Мухамедиева Дилноз Тулкуновна

КВАНТОВЫЕ ВЫЧИСЛЕНИЯ ПРИ ИНТЕЛЛЕКТУАЛЬНОМ АНАЛИЗЕ ДАННЫХ

Монография

Мухамедиева Дилноз Тулкуновна

КВАНТОВЫЕ ВЫЧИСЛЕНИЯ ПРИ ИНТЕЛЛЕКТУАЛЬНОМ АНАЛИЗЕ ДАННЫХ

Монография

Национальный исследовательский университет «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Мухамедиева Дилноз Тулкуновна

КВАНТОВЫЕ ВЫЧИСЛЕНИЯ ПРИ ИНТЕЛЛЕКТУАЛЬНОМ АНАЛИЗЕ ДАННЫХ

Ташкент-2024

Издательство «Fan ziyosi»

УДК 519.71(575.1)

Д.Т.Мухамедиева. «Квантовые вычисления при интеллектуальном анализе данных». Монография – Т.: Изд. «Fan ziyosi», 2024. 302 с.

В работе рассмотрены актуальные теоретико-методические проблемы квантовых вычислений при интеллектуальном анализе данных. Благодаря способности квантовых компьютеров решать определенные вычислительные задачи экспоненциально быстрее классических, они могут найти применение в самых различных областях, таких как криптография, машинное обучение, искусственный интеллект, биология и т.д. В связи с нетрадиционностью применяемого математического аппарата в книге приводятся основные положения теории квантовых вычислений, нейронных сетей в объеме, необходимом для понимания постановок задач, рассмотренных в последующих главах.

Рекомендовано к печати НТС

Национально-исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства»

Рецензенты:

Маматов Н.С. - д.т.н., профессор Национально- исследовательского университета «Ташкентский институт инженеров ирригации и механизации сельского хозяйства».

Рахимов Н.О. – д.т.н., профессор Ташкентского университета информационных технологий.

С О Д Е Р Ж А Н И Е

Введение

Искусственный интеллект (ИИ) и квантовые вычисления (КВ) – это две области, которые тесно связаны и могут взаимодействовать друг с другом. ИИ относится к разработке компьютерных систем, способных выполнять задачи, которые обычно требуют интеллекта человека. КВ, с другой стороны, являются новым подходом к вычислениям, основанным на принципах квантовой механики.

Одна из основных связей между ИИ и КВ заключается в том, что КВ могут предоставить новые вычислительные возможности, которые могут быть использованы для улучшения алгоритмов и моделей ИИ. Классические компьютеры ограничены в своих вычислительных возможностях, особенно при работе с большими объемами данных и сложными задачами. КВ, с другой стороны, могут обрабатывать и анализировать информацию параллельно и использовать принципы квантовой суперпозиции и квантового взаимодействия для решения сложных задач.

Использование КВ в ИИ может привести к разработке новых алгоритмов и моделей, которые могут быть более эффективными и точными. Например, квантовые нейронные сети могут быть использованы для обработки и анализа больших объемов данных с высокой скоростью и точностью. Квантовые алгоритмы также могут быть применены для решения оптимизационных задач, которые являются важными в области ИИ.

Однако, в настоящее время КВ все еще находятся в ранней стадии развития, и существует множество технических и теоретических проблем, которые нужно решить, прежде чем они станут практически применимыми в ИИ. Некоторые из этих проблем включают ошибки квантовых вычислений, сложность программирования квантовых алгоритмов и необходимость разработки новых алгоритмов и моделей, специально адаптированных для КВ.

Тем не менее, совмещение ИИ и КВ представляет большой потенциал для развития новых технологий и приложений. Использование КВ в ИИ может привести к созданию более интеллектуальных и эффективных систем, способных решать сложные задачи и преодолевать ограничения классических компьютеров. Это может иметь значительное влияние на различные области, такие как медицина, финансы, транспорт и многие другие.

Квантовые технологии одно из наиболее динамически развивающихся направлений. Квантовые технологии открывают новые возможности для целого ряда областей. За счет своих уникальных свойств квантовые системы могут стать основой нового поколения высокопроизводительных вычислительных устройств квантовых компьютеров, методов защиты информации с использованием квантовой криптографии, а также высокоточных измерительных устройств квантовых сенсоров и квантовых метрологических устройств. Обзор посвящен прогрессу, наблюдаемому в основных сферах современных квантовых технологий: квантовой обработки информации, квантовой криптографии, а также квантовой метрологии и квантовой сенсорики.

Последние десятилетия наблюдается стремительный прогресс в области информационных технологий. С каждым годом вычислительные устройства становятся как более компактными, так и более производительными. то же движет столь стремительным прогрес сом вычислительных технологий? Основным стимулирующим фактором развития компьютеров считается совершенствование технологий, связанных с миниатюризацией элементной базы. Иными словами, мощность компьютеров растет, так как в каждом новом поколении компьютеров на чипе той же площади можно разместить примерно вдвое больше транзисторов. Это эмпирическое правило, известное сегодня как закон Мура, с достаточной степенью точности описывало развитие информационных технологий. Чтобы поддерживать дальнейший рост производительности компьютеров, необходимо будет создавать транзисторы атомных размеров. Стоит отметить, что рост других параметров, которые сопутствуют развитию вычислительных технологий, таких как тактовая частота процессоров, уже завершился. Таким образом, на этот раз развитие компьютеров сталкивается с новой физической парадигмой: не с привычной классической физикой, а с квантовой механикой.

Строго говоря, появление таких технологий, как транзисторы и лазеры также явилось результатом исследований в области квантовой физики. Однако в случае рассмотрения принципов функционирования лазера или транзистора речь идет о коллективных квантовых явлениях, проявляющихся на уровне коллектива большого числа квантовых объектов атомов, фотонов или электронов. Задача же управления

индивидуальными кванто выми объектами, такими как одиночные фотоны, атомы, ионы оказывается гораздо сложнее: она находится на переднем крае развития науки. Для ее решения в случае рассмотрения сложных квантовых систем на уровне индивидуальных частиц нужно разработать эффективные методы создания, контроля и измерения. Отметим, что Нобелевская премия по физике была вручена С. Арошу и Д. Вайленду с формулировкой: За создание прорывных технологий манипулирования квантовыми системами, которые сделали возможными измерение отдельных квантовых систем и управление ими.

Квантовые технологии, т. е. технологии, основанные на управлении индивидуальными квантовыми свойствам частиц, активно развиваются по всему миру. Они разрабатываются в ведущих университетах, исследовательских центрах и компаниях. В Российской Федерации квантовые технологии входят в перечень основных сквозных цифровых технологий.

В технологически развитых странах в США, Китае, Канаде, ЕС, Великобритании, Японии, Австра лиии т. д. исследования и разработки в области квантовой физики находятся под бдительным вниманием со стороны государства: для их развития создаются специализированные центры компетен ций, а финансирование обеспечивается специаль ными целевыми программами. На сегодняшний день главным потребителем квантовых техноло гий является государство. Во многом это объясняется стратегической важностью квантовых техно логий для обеспечения защищенности интересов государства, например, для обеспечения безопас ности в информационной сфере. В Евросоюзе создана специальная программа quantum Flagship – после завершения предыдущей программы по развитию квантовых технологий. В Китае запущена программа по квантовым технологиям. Конгрессом США утверждена долгосрочная программа развития квантовых технологий National Quantum Initiative. Аналогичные программы рассматриваются в Великобритании, Японии, Канаде, Австралии и ряде других стран.

Глава 1. МАТЕМАТИЧЕСКИЕ ОСНОВЫ КВАНТОВЫХ ВЫЧИСЛЕНИЙ

1.1. Основные понятия и определения

1.1.1. Комплексные числа

$N \subset Z \subset Q \subset R \subset C$

N – множество натуральных чисел;

Z – множество целых чисел;

Q – множество рациональных чисел;

R – множество вещественных чисел;

C – множество комплексных чисел.

Определение 1.1

Комплексное число z – это упорядоченная пара вещественных чисел $z=(x,y)$ записанное в виде $z=x+i\cdot y$, где $i=\sqrt{-1}$, называемая мнимой единицей [1-10,33-40].

Определение 1.2

Функция комплексной переменной $z=(x,y)$ в общем случае имеет вид:

$\omega=u(x,y)+i\cdot v(x,y)$, где $u=(x,y)-$ вещественная часть функции, ω, $v=(x,y)-$ мнимая часть функции ω.

Примером функции комплексной переменной является показательная функция $\omega=e^{Z}$, определяется сходящимся рядом:

$$e^{Z}=\sum_{n=0}^{\infty}\frac{z^{n}}{n!},$$

$$e^{i\varphi}=\sum_{n=0}^{\infty}\frac{i^{n}\varphi^{n}}{n!}=\sum_{n=0}^{\infty}(-1)^{n}\frac{\varphi^{2n}}{(2n)!}+i\cdot\sum_{n=0}^{\infty}(-1)^{n}\frac{\varphi^{2n+1}}{(2n+1)!},$$

$$e^{i\varphi}=\cos(\varphi)+i\cdot\sin(\varphi).$$

Формулы Эйлера:

$$\cos(\varphi)=\frac{e^{i\varphi}+e^{-i\varphi}}{2},$$

$$\sin(\varphi)=\frac{e^{i\varphi}-e^{-i\varphi}}{2\cdot i}.$$

Алгебраическая форма записи: $z=x+i\cdot y$.

$x=\mathrm{Re}(z)$ -вещественная часть комплексного числа z.

$y=\mathrm{Im}(z)$ -мнимая часть комплексного числа z.

Тригонометрическая форма записи: $z=r\cdot(\cos(\varphi)+i\cdot\sin(\varphi))$, где $r=|z|=\sqrt{x^{2}+y^{2}}-$ модуль комплексного числа, а $r=Arg(z)=\arg(z)+2\cdot\pi\cdot k-$ аргумент комплексного числа $(k=0,\pm1,\pm2,...)$.

Показательная форма записи: $z = r \cdot e^{i\varphi}$.

Определение 1.3

Комплексное число вида $\overline{z} = x - i \cdot y$ —· называется комплексно-сопряженным комплексному числу вида $z = x + i \cdot y$.

Тогда можно записать:

$$\mathrm{Re}(z) = \frac{1}{2}(z + \overline{z}),$$

$$\mathrm{Im}(z) = \frac{1}{2 \cdot i}(z - \overline{z}).$$

Определение 1.4

Комплексная плоскость $\tilde{N}$ – плоскость, на которой каждому комплексному числу $z = x + i \cdot y$ соответствует точка $z(x, y)$.

Операции над комплексными числами:

1. Равенство комплексных чисел z_1 и z_2.

$$\mathrm{Re}(z_1) = \mathrm{Re}(z_2)$$

$$\mathrm{Im}(z_1) = \mathrm{Im}(z_2)$$

2. Сумма/разность комплексных чисел $z_1 = x_1 + i \cdot y_1$ и $z_2 = x_2 + i \cdot y_2$.

$$z = z_1 \pm z_2 = (x_1 \pm x_2) + i \cdot (y_1 \pm y_2).$$

3. Произведение комплексных чисел $z_1 = x_1 + i \cdot y_1$ и $z_2 = x_2 + i \cdot y_2$.

Алгебраическая запись: $z = z_1 \cdot z_2 = (x_1 \cdot x_2 - y_1 \cdot y_2) + i \cdot (x_1 \cdot y_2 + x_2 \cdot y_1)$.

Тригонометрическая запись:

$$z_1 = r_1 \cdot (\cos(\varphi_1) + i \cdot \sin(\varphi_1)),$$

$$z_2 = r_2 \cdot (\cos(\varphi_2) + i \cdot \sin(\varphi_2)),$$

$$z_1 \cdot z_2 = r_1 \cdot r_2 \cdot (\cos(\varphi_1 + \varphi_2) + i \cdot \sin(\varphi_1 + \varphi_2)),$$

$$|z_1 \cdot z_2| = |z_1| \cdot |z_2|,$$

$$Arg(z_1 \cdot z_2) = Arg(z_1) + Arg(z_2).$$

4. Деление комплексных чисел $z_1 = x_1 + i \cdot y_1$ и $z_2 = x_2 + i \cdot y_2$.

Алгебраическая запись (при $z_2 \neq 0$):

$$\frac{z_1}{z_2} = \frac{z_1 \cdot \overline{z}_2}{z_2 \cdot \overline{z}_2} = \frac{z_1 \cdot \overline{z}_2}{|z_2|^2},$$

$$z_1 = r_1 \cdot (\cos(\varphi_1) + i \cdot \sin(\varphi_1)),$$

$$z_2 = r_2 \cdot (\cos(\varphi_2) + i \cdot \sin(\varphi_2)),$$

$$\frac{z_1}{z_2} = \frac{r_1}{r_2} \cdot (\cos(\varphi_1 - \varphi_2) + i \cdot \sin(\varphi_1 - \varphi_2)),$$

$$\left|\frac{z_1}{z_2}\right| = \frac{|z_1|}{|z_2|},$$

$$Arg\left(\frac{z_1}{z_2}\right) = Arg(z_1) - Arg(z_2).$$

5. Возведение в степень $z = x + i \cdot y$.

Формула Муавра – $z^n = r^n \cdot (\cos(n \cdot \varphi) + i \cdot \sin(n \cdot \varphi))$.

$$\sqrt[n]{z} = \sqrt[n]{r} \cdot \left(\cos\left(\frac{\varphi}{n} + \frac{2\pi k}{n}\right) + i \cdot \sin\left(\frac{\varphi}{n} + \frac{2\pi k}{n}\right)\right).$$

6. Обратный элемент комплексного числа z.

По сложению: $-z \to z + (-z) = 0$.

По умножению: $\forall z \neq 0 \exists z^{-1} \equiv \frac{1}{z}$, $z \cdot z^{-1} = 1$.

1.1.2. Линейные пространства

Определение 1.5

Линейное пространство (векторное пространство) – множество Φ элементов $\vec{x}, \vec{y}, ..., \vec{z}$ (произвольной природы), определяемое следующими свойствами [34]:

1) Для любых двух элементов $\vec{x}, \vec{y} \in \Phi$ определен элемент $\vec{z} \in \Phi$, называемый суммой элементов $\vec{x}$ и $\vec{y}$ обозначаемый $\vec{z} = \vec{x} \oplus \vec{y}$.
2) Для любого элемента $\vec{x} \in \Phi$ и любого вещественного числа α определен элемент $\vec{u} \in \Phi$, называемый произведением элемента $\vec{x}$ на число α и обозначаемый $\vec{u} = \alpha \cdot \vec{x}$.
3) Указанные два свойства подчинены следующим аксиомам:

3.1. Коммутативность сложения:

$$\vec{x} \oplus \vec{y} = \vec{y} \oplus \vec{x}.$$

3.2. Ассоциативность сложения:

$$(\vec{x} \oplus \vec{y}) \oplus \vec{z} = \vec{x} \oplus (\vec{y} \oplus \vec{z}).$$

3.3. Особая роль нулевого элемента:

$\exists$ нулевой элемент $\vec{0}$ такой что $\vec{x} \oplus \vec{0} = \vec{x} \forall \vec{x}$.

3.4. Существование противоположного элемента:

$\forall \vec{x} \exists$ противоположный элемент $\vec{x}'$ такой, что $\vec{x} \oplus \vec{x}' = \vec{0}$.

3.5. Особая роль числового множителя на 1:

$$1 \oplus \vec{x} = \vec{x} \forall \vec{x}.$$

3.6. Ассоциативность относительно числового множителя:

$$\alpha(\beta\vec{x}) = (\alpha\beta)\vec{x}.$$

3.7. Дистрибутивность относительно числового множителя:

$(\alpha + \beta)\vec{x} = \alpha\vec{x} \oplus \beta\vec{x}$ для любых $\alpha, \beta \in R$.

3.8. Дистрибутивность относительно суммы элементов:

$\alpha(\vec{x} \oplus \vec{y}) = \alpha\vec{x} \oplus \alpha\vec{y}$ для любых $\alpha, \beta \in R$.

Определение 1.6

Линейное подпространство M линейного пространства Φ – подмножество $M \subset \Phi$ такое, что $\forall \vec{x}, \vec{y} \in M$ и любого числа α выполняются условия: $\vec{x} \oplus \vec{y} \in M$ и $\alpha\vec{x} \in M$.

Определение 1.7

Линейная комбинация элементов $\vec{x}, \vec{y}, ..., \vec{z}$ пространства Φ – сумма произведений элементов на произвольные вещественные числа:

$$\alpha\vec{x} + \beta\vec{y} + ... + \gamma\vec{z}.$$

Определение 1.8

Тривиальная линейная комбинация элементов $\vec{x}, \vec{y}, ..., \vec{z}$ – линейная комбинация $\alpha\vec{x} + \beta\vec{y} + ... + \gamma\vec{z}$, в которой все коэффициенты $\alpha, \beta, ..., \gamma$ равны 0.

Определение 1.9

Множество векторов $\vec{x}, \vec{y}, ..., \vec{z}$ называется линейно зависимым, если существует нетривиальная линейная комбинация этих векторов $\alpha\vec{x} + \beta\vec{y} + ... + \gamma\vec{z}$, равная 0.

Определение 1.10

Множество векторов $\vec{x}, \vec{y}, ..., \vec{z}$ называется линейно независимым, если никакая нетривиальная линейная комбинация этих векторов $\alpha\vec{x} + \beta\vec{y} + ... + \gamma\vec{z}$, не равная 0.

Определение 1.11

Базис в пространстве V – это множество n линейно независимых векторов $\{\vec{e}_1, ..., \vec{e}_n\}$, а любые $n+1$ векторов линейно зависимы.

Определение 1.12

Размерность пространства V – число векторов базиса n, обозначаемое $\dim V$.

Определение 1.13

Конечномерное пространство V – пространство, в котором имеется конечный базис.

Определение 1.14

Бесконечномерное пространство V – пространство V, в котором можно найти системы линейно независимых векторов $\{\vec{e}_1, ..., \vec{e}_n\} \forall n \in N$

Определение 1.15

В случае конечномерного пространства V любой $\vec{x} \in V$ является линейной комбинацией базисных векторов $\{\vec{e}_1, ..., \vec{e}_n\}$, т.е. однозначно представляется в виде

$$\vec{x} = x^1\vec{e}_1 + ... + x^n\vec{e}_n = \sum_{i=1}^{n} x^i\vec{e}_i .$$

Данное выражение является разложением вектора $\vec{x}$ по базису $\{\vec{e}_1,...,\vec{e}_n\}$.

Определение 1.16

Компоненты (координаты) вектора $\vec{x}$ в базисе $\{\vec{e}_1,...,\vec{e}_n\}$ – числовые коэффициенты $x^i, i = 1,...,n$ разложения этого вектора по базису.

Определение 1.17

Пусть $\{\vec{e}_1,...,\vec{e}_n\}$ и $\{\vec{e}'_1,...,\vec{e}'_n\}$ – два базиса в V. Пусть $\vec{e}'_i (i = 1,...,n)$ разложен по базису $\{\vec{e}_1,...,\vec{e}_n\}$:

$$\vec{e}'_i = c_i^1\vec{e}_1 + ... + c_i^n\vec{e}_n = \sum_{j=1}^{n} c_i^j\vec{e}_j , \quad i = 1,...,n ,$$

где коэффициенты разложения $c_i^j, i = 1,...,n$ – суть координаты i-го вектора базиса $\{\vec{e}'_1,...,\vec{e}'_n\}$ в базисе $\{\vec{e}_1,...,\vec{e}_n\}$.

Квадратная матрица n-го порядка $C = c_i^j$ – матрица перехода от базиса $\{\vec{e}'_1,...,\vec{e}'_n\}$ к базису $\{\vec{e}_1,...,\vec{e}_n\}$ в V.

Определение 1.18

Линейная оболочка множества $Ls\{U\}$ – это множество всех линейных комбинаций векторов, входящих в множество U.

Определение 1.19

Сумма $M + N$ линейных подпространств M и N линейного пространства V – это линейная оболочка векторов из $M \cup N$, т.е. множество всевозможных линейных комбинаций элементов из M и N:

$$M + N = Ls\{M \cup N\} .$$

Теорема 1.1 Пусть M и N – линейные пространства, дающие в сумме пространство V. Тогда для них эквивалентны следующие условия [34-40]:

- $V = M + N$;
- всякий $\vec{x} \in V$ может быть однозначно представлен в виде $\vec{x} = \vec{z} + \vec{y}$, для некоторых $\vec{y} \in M, \vec{z} \in N$;
- если $\{\vec{e}_1,...,\vec{e}_a,...\}$ – базис в M и $\{\vec{f}_1,...,\vec{f}_b,...\}$ – базис в N, то множество $\{\vec{e}_1,...,\vec{e}_a,...,\vec{f}_1,...,\vec{f}_b,...\}$ образуют базис в V.

В результате: $\dim V = \dim M + \dim N$.

Теорема 1.2

Для любого подпространства M конечномерного линейного пространства V найдется такое линейное подпространство N, что $V = M + N$.

Определение 1.20

Линейные пространства V и W называются изоморфными, если между их элементами установлено такое взаимное соответствие, при котором сумме любых двух элементов одного пространства отвечает сумма соответствующих элементов другого пространства, а произведению элемента одного пространства на некоторое число отвечает произведение на то же число соответствующего элемента другого пространства.

Обозначение: $V \cong W$.

Теорема 1.3

Все конечномерные комплексные (вещественные) линейные пространства одинаковой размерности n изоморфны.

1.1.3. Линейные операторы

Определение 1.21

Линейное преобразование (оператор) или эндоморфизм в линейном пространстве V – отображение $\hat{A}: V \to V$, линейное по аргументу, т.е.

$\hat{A}(\alpha\vec{x} + \beta\vec{y}) = \alpha\hat{A}\vec{x} + \beta\hat{A}\vec{y} \quad \forall\vec{x}, \vec{y} \in V$ и $\forall\alpha, \beta \in N$.

Обозначение: $End\, V$ – множество всех линейных операторов в линейном пространстве V.

Определение 1.22

Нулевой оператор – оператор $\hat{0}: \vec{x} \to 0\vec{x} = \vec{0}$, т.е. оператор, отображающий все векторы пространства V в ноль.

Определение 1.23

Тождественный (единичный) оператор – оператор $\hat{1}: \vec{x} \to 1\vec{x} = \vec{x}$, т.е. оператор, отображающий каждый вектор пространства V в самого себя.

Определение 1.24

Пусть $\{\vec{e}_1, ..., \vec{e}_n\}$ – базис в V, а $\hat{A} \in End\, V$ – линейный оператор в V. Подействовав на каждый базисный вектор , $\vec{e}_i, i = \overline{1,n}$ оператором $\hat{A}$, получим некоторые векторы $\hat{A}\vec{e}_i \in V$, $i = \overline{1,n}$. Разложив каждый из полученных векторов по базису получим:

$$\hat{A}\vec{e}_i = \sum_{j=1}^{n} a_i^j \vec{e}_j,\ i = \overline{1,n}\ ,$$

где коэффициенты, α_i^j, $j = \overline{1,n}$ – суть координаты вектора $\hat{A}\vec{e}_i$ в базисе $\{\vec{e}_1,...,\vec{e}_n\}$.

Матрица линейного оператора $\hat{A}$ в базисе $\{\vec{e}_1,...,\vec{e}_n\}$ – матрица A, матричными элементами которой являются коэффициенты $a_i^j, i, j = \overline{1,n}$, определяется как

$$A = (\alpha_i^j) \in Mat(n, C).$$

Матричный элемент линейного оператора $\hat{A}$ в базисе $\{\vec{e}_1,...,\vec{e}_a\}$ – коэффициент α_i^j.

Формула, связывающая матрицы одного и того же линейного оператора $\hat{A}$ в разных базисах

$$A' = C^{-1}AC,$$

где A и A' – матрицы одного и того же оператора $\hat{A}$ в базисах $\{\vec{e}_1,...,\vec{e}_n\}$ и $\{\vec{e}'_1,...,\vec{e}'_n\}$ соответственно, C – матрица перехода от базиса $\{\vec{e}'_1,...,\vec{e}'_n\}$ к базису $\{\vec{e}_1,...,\vec{e}_n\}$, т.е.

$$C = (c_i^j),$$

$$\vec{e}_i = \sum_{j=1}^{n} c_i^j \vec{e}'_i\ ,$$

$$\vec{e}'_i = \sum_{j=1}^{n} (c_i^{-1})_i^j \vec{e}_i,\ \ i = \overline{1,n}.$$

Определение 1.25

Определитель (*детерминант*) линейного оператора $\hat{A}$ – определитель матрицы линейного оператора $\hat{A}$.

$$\det \hat{A} = \det A.$$

Обозначение: $\det \hat{A}$.

Определение 1.26

След линейного оператора $\hat{A}$ – след матрица линейного оператора $\hat{A}$.

$$Tr\hat{A} = Tr\, A.$$

Обозначение: $Tr\hat{A}$.

Определение 1.27

Если определитель оператора $\hat{A}$ не равен 0, то оператор называется невырожденным линейным оператором $\hat{A}$. В противном случае оператор $\hat{A}$ называется вырожденным линейным оператором $\hat{A}$

Алгебра линейных операторов

Пусть $\hat{A}$ и $\hat{B}$ – два линейных оператора в произвольном пространстве V, α – вещественное число. Можно определить их сумму $\hat{A}+\hat{B}\in End\,V$, произведение $\hat{A}\cdot\hat{B}\in End\,V$, а также произведение $\alpha\hat{A}$ следующим образом:

1) $(\hat{A}+\hat{B})\vec{x}=\hat{A}\vec{x}+\hat{B}\vec{x}$;

2) $\hat{A}\cdot\hat{B}\cdot\vec{x}=\hat{A}\cdot(\hat{B}\cdot\vec{x})$;

3) $(\alpha\hat{A})\vec{x}=\alpha(\hat{A}\vec{x}),\forall\vec{x}\in V$.

Свойства умножения операторов $\forall\hat{A},\hat{B},\hat{C}\in End\,V$ и $\forall$ чисел α,β:

1. Дистрибутивность умножения операторов по сложению

$\hat{A}(\alpha\hat{B}+\beta\hat{C})=\alpha\hat{A}\hat{B}+\beta\hat{A}\hat{C}$,

$(\alpha\hat{A}+\beta\hat{B})\hat{C}=\alpha\hat{A}\hat{C}+\beta\hat{B}\hat{C}$.

2. Ассоциативность умножения операторов

$\hat{A}(\hat{B}\hat{C})=(\hat{A}\hat{B})\hat{C}$.

3. Свойство единичного оператора

$\hat{1}\hat{A}=\hat{A}\hat{1}=\hat{A}$.

4. Некоммутативность умножения операторов

$\hat{A}\hat{B}\neq\hat{B}\hat{A}$.

Определение 1.28

Обратный оператор $\hat{B}\in End\,V$ к оператору $\hat{A}\in End\,V$ – оператор такой, что $\hat{A}\hat{B}=\hat{B}\hat{A}=1$. Обозначение: $\hat{A}^{-1}$.

Свойство обратного оператора: в конечномерном линейном пространстве матрица оператора $\hat{A}^{-1}$ – является обратной матрицей к матрице оператора $\hat{A}$, т.е.

$$\hat{A}\hat{A}^{-1}=\hat{A}^{-1}\hat{A}=1_n.$$

Определение 1.29

k-я степень оператора $\hat{A}$ – оператор $\hat{A}^k=\hat{A}\hat{A}...\hat{A}$ (k-сомножителей $k=1,2,3,...$).

Определение 1.30

Диагонализируемый (полупростой) оператор $\hat{A}$ – оператор такой, что в некотором базисе $\{\vec{e}_1,...,\vec{e}_n\}$ матрица A оператора диагональная, т.е.

$$A=\begin{pmatrix}\lambda_1 & 0 & ... & 0\\ 0 & \lambda_2 & ... & 0\\ ... & ... & ... & ...\\ 0 & 0 & ... & \lambda_n\end{pmatrix}=diag\{\lambda_1,\lambda_2,...,\lambda_n\}.$$

Определение 1.31

Собственное значение λ оператора $\hat{A}\in EndV$ – такое число, что $\exists\hat{x}\neq 0$ такой, что $\hat{A}\vec{x}=\lambda\vec{x}$.

Определение 1.32

Собственный вектор оператора $\hat{A}$, отвечающий собственному значению λ – нетривиальное решение уравнения $\hat{A}\vec{x} = \lambda\vec{x}$.

Определение 1.33

Множество всех собственных векторов линейного оператора, соответствующих собственному числу λ, дополненное нулевым вектором, называется собственным подпространством этого оператора.

Определение 1.34

Геометрическая кратность собственного значения λ оператора $\hat{A}$ – размерность M_λ, т.е. $\dim M_\lambda$

Определение 1.35

Спектр оператора $\hat{A}$ – множество всех различных собственных значений оператора $\hat{A}$.

Теорема 1.4

Пусть $\vec{x}_1 \in M_{\lambda_1}, \vec{x}_2 \in M_{\lambda_2}, ..., \vec{x}_k \in M_{\lambda_k}$ – собственные векторы оператора $\hat{A} \in EndV$, отвечающие попарно различным собственным значениям $\lambda_1, \lambda_2, ..., \lambda_n \ \ (\lambda_i \neq \lambda_j \ при \ i \neq j)$. Тогда векторы $\vec{x}_1, \vec{x}_2, ..., \vec{x}_k$ линейно независимы.

Теорема 1.36

Уравнение $(\hat{A} - \lambda\hat{1})\vec{x} = \vec{0}$ имеет ненулевые решения $\vec{x} \neq \vec{0}$ тогда и только тогда, когда $\det(\hat{A} - \lambda\hat{1}) = 0$.

Характеристические уравнения линейного оператора $\hat{A}$/ матрицы A

$$f_{\hat{A}}(\lambda) = \det(\hat{A} - \lambda\hat{1}) = \begin{pmatrix} a_1^1 - \lambda & a_2^1 & a_3^1 & ... & a_n^1 \\ a_1^2 & a_2^2 - \lambda & a_3^2 & ... & a_n^2 \\ a_1^3 & a_2^3 & a_3^3 - \lambda & ... & a_n^3 \\ ... & ... & ... & ... & ... \\ a_1^n & a_2^n & a_3^n & ... & a_n^n - \lambda \end{pmatrix} = 0 \quad ,$$

где $A = (a_j^i)$ – матрица оператора $\hat{A}$ в каком-либо базисе.

Многочлен n-й степени имеет ровно n комплексных корней, если каждый корень учитывать столько раз, какова его алгебраическая кратность. Для характеристического многочлена это означает:

1. Характеристическое уравнение $f_{\hat{A}}(\lambda) = 0$ всегда имеет не менее одного и не более n различных комплексных корней $\lambda_1, \lambda_2, ..., \lambda_k$ для некоторого $k = \overline{1,n}$.

2. Характеристический многочлен представляется в виде

$$f_{\hat{A}}(\lambda) = (-1)^n (\lambda - \lambda_1)^{n_1} (\lambda - \lambda_2)^{n_2} ... (\lambda - \lambda_k)^{n_k} = 0,$$

где $n_1, n_2, ..., n_k$ – целые положительные числа, причем $n_1 + n_2 + ... + n_k = n$, n_i – алгебраическая кратность собственного значения λ_i .

Определение 1.37

Алгебраическая кратность n_i собственного значения λ_i – кратность λ_i как корня характеристического многочлена.

Определение 1.38 Собственное значение λ_i некоторого линейного оператора называется простым, если его алгебраическая кратность $n_i = 1$.

Определение 1.39

Собственное значение λ_i некоторого линейного оператора называется *вырожденным*, если его алгебраическая кратность $n_i > 1$.

Определение 1.40

Оператор $\hat{A} \in EndV$ называется оператором с простым спектром, если все его собственные значения являются простыми.

Теорема 1.5

Пусть $\hat{A} \in EndV$ – линейный оператор в конечном комплексном пространстве. Тогда справедливо:

1. $\forall$ собственного значения λ_i геометрическая кратность не превосходит алгебраическую, т.е. $r_i \le n_i$.

2. Оператор $\hat{A}$ диагонализируем тогда и только тогда, когда $\forall \lambda_i r_i \le n_i$.

Следствие: линейный оператор с простым спектром в конечномерном комплексном пространстве диагонализируется.

Определение 1.41

Нильпотентный оператор $\hat{A} \in EndV$ – оператор, некоторая степень которого является нулевым оператором, т.е. $\langle$ |для некоторого $k > 0$.

Теорема 1.6.

Единственное собственное значение нильпотентного оператора $\hat{A}$ равно 0. Характеристический многочлен любого нильпотентного оператора равен $f_{\hat{A}}(\lambda) = (-1)^n \lambda^n$, где $n = \dim V$.

Теорема 1.7

Если оператор $\hat{A}$ в конечномерном линейном пространстве имеет единственное значение $\lambda_1 = 0$, то он нильпотентен.

Теорема 1.8

Ненулевой нильпотентный оператор не диагонализируем.

Теорема 1.9

Пусть $\alpha \in C$ – некоторое комплексное число. Тогда:

1. Операторы $\hat{A}$ и $\hat{A}+\alpha\hat{1}$ диагонализируемы и не диагонализируемы одновременно.

2. Если $\{\lambda_1, \lambda_2, ..., \lambda_k\}$ – спектр оператора $\hat{A}$, то $\{\lambda_1+\alpha, \lambda_2+\alpha, ..., \lambda_k+\alpha\}$ – спектр оператора $\hat{A}+\alpha\hat{1}$.

3. Каждое собственное пространство M_{λ_i} оператора $\hat{A}$, отвечающее собственному значению λ_i, является собственным пространством оператора $\hat{A}+\alpha\hat{1}$, отвечающим собственному значению $\lambda_i+\alpha$.

4. Характеристические многочлены операторов $\hat{A}$ и $\hat{A}+\alpha\hat{1}$ связаны соотношением $f_{\hat{A}}(\lambda) = f_{\hat{A}+\alpha\hat{1}}(\lambda-\alpha)$.

5. Геометрическая и алгебраическая кратности каждого собственного значения λ_i оператора $\hat{A}$ совпадают с геометрической и алгебраической кратностями соответственно собственного $\lambda_i+\alpha$ оператора $\hat{A}+\alpha\hat{1}$.

Следствие: пусть $\hat{A} \in EndV$ – ненулевой нильпотентный оператор. Тогда оператор $\hat{A}+\alpha\hat{1}$ также не диагонализируем и имеет единственное вырожденное собственное значение α, для которого алгебраическая кратность равна n, а геометрическая кратность – 1. Характеристический многочлен оператора $\hat{A}+\alpha\hat{1}$ равен $f_{\hat{A}}(\lambda) = f_{\hat{A}+\alpha\hat{1}}(\alpha-\lambda)^n$

1.1.4. Эрмитовы и симметричные формы

Определение 1.42

Эрмитова форма (эрмитово скалярное произведение) на комплексном линейном пространстве V – правило, которое каждой паре векторов $\vec{x}, \vec{y} \in V$ ставит в соответствие комплексное число, т.е. $\langle \vec{x}, \vec{y} \rangle \in C$, причем выполняются следующие свойства $\forall \vec{x}, \vec{y}, \vec{z} \in V$ и $\forall \alpha, \beta \in C$

1. Линейность по второму аргументу
$$\langle \vec{x}, \alpha\vec{y} + \beta\vec{z} \rangle = \alpha \langle \vec{x}, \vec{y} \rangle + \beta \langle \vec{x}, \vec{z} \rangle .$$
2. Полулинейность по первому аргументу
$$\langle \alpha\vec{x} + \beta\vec{y}, \vec{z} \rangle = \alpha \langle \vec{x}, \vec{z} \rangle + \beta \langle \vec{y}, \vec{z} \rangle .$$
3. Эрмитовость
$$\langle \vec{x}, \vec{y} \rangle = \overline{\langle \vec{y}, \vec{z} \rangle} .$$

Определение 1.43

Матрица эрмитовой формы H в базисе $\{\vec{e}_1,...,\vec{e}_n\}$ – комплексная матрица n-го порядка, образованная попарным эрмитовым скалярным произведением базисных векторов, т.е.

$$h_{i,j} = \langle \vec{e}_i, \vec{e}_j \rangle, i, j = \overline{1,n},$$

$$H = (h_{i,j}) \in Mat(n, C).$$

Формула, связывающая матрицы одной и той же эрмитовой формы в разных базисах,

$$H' = C^{\dagger} HC,$$

где H – матрица эрмитовой формы в базисе $\{\vec{e}_1,...,\vec{e}_n\}$, H' – матрица той же эрмитовой формы в базисе $\{\vec{e}'_1,...,\vec{e}'_n\}$, C – матрица перехода от базиса $\{\vec{e}'_1,...,\vec{e}'_n\}$ к базису $\{\vec{e}_1,...,\vec{e}_n\}$, $C^{\dagger} = \overline{C}^T$.

Определение 1.44

Ортогональные векторы $\vec{x}, \vec{y} \in V$ – векторы такие, что $\langle \vec{x}, \vec{y} \rangle = 0$.

Определение 1.45

Изотропный вектор $\vec{x} \in V$ – вектор ортогональный сам себе, т.е. $\langle \vec{x}, \vec{y} \rangle = 0$

Определение 1.46

В пространстве C задан стандартный базис $\vec{e}_1 = (1,0,...,0)$, $\vec{e}_2 = (0,1,...,0)$, …, $\vec{e}_n = (0,0,...,1)$. В данном базисе зададим эрмитово скалярное произведение векторов $\vec{x} = (x^1,...,x^n) \in \tilde{N}$ и $\vec{y} = (y^1,...,y^n) \in \tilde{N}$ следующим образом:

$$\langle \vec{x}, \vec{y} \rangle = \overline{x}^1 y^1 + \overline{x}^2 y^2 + ... + \overline{x}^p y^p - \overline{x}^{p+1} y^{p+1} - \overline{x}^{p+2} y^{p+2} - ... - \overline{x}^{p+q} y^{p+q},$$

где $p + q = r \le n$

Матрица этой эрмитовой формы в рассматриваемом базисе имеет вид

$$H = diag(1,1,...,1,-1,-1,...,-1,0,0,...,0).$$

Ортонормированный базис – базис, в котором матрица эрмитовой формы имеет вид

$$H = diag(1,1,...,1,-1,-1,...,-1,0,0,...,0).$$

Теорема 1.10

В любом конечномерном комплексном пространстве V с эрмитовой формой существует ортонормированный базис.

Определение 1.47

Эрмитово скалярное произведение на комплексном линейном пространстве V называется невырожденным, если в V нет ненулевых векторов, ортогональных векторам из V, т.е. $\langle \vec{x}, \vec{y} \rangle = 0$, $\forall \vec{y} \neq 0 \Leftrightarrow \vec{x} = 0$.

Определение 1.48

Эрмитово скалярное произведение называется положительно определенным, если $\langle \vec{x}, \vec{y} \rangle > 0$, $\forall \vec{x} \neq 0$.

Теорема 1.11

Матрица положительно определенной эрмитовой формы в ортонормированном базисе равна единичной матрице.

Определение 1.49 Унитарное пространство $(V, \langle , \rangle)$ – конечномерное комплексное линейное пространство V с положительно определенной эрмитовой формой $\langle , \rangle$.

Определение 1.50

Эрмитов (самосопряженный) оператор $\hat{A} \in EndV$ – оператор в унитарном пространстве $(V, \langle , \rangle)$ такой, что $\left\langle \vec{x}, \hat{A}\vec{y} \right\rangle = \left\langle \hat{A}\vec{x}, \vec{y} \right\rangle, \forall \vec{x}, \vec{y} \in V$.

Теорема 1.12 (свойства эрмитова оператора)

1. Оператор $\hat{A}$ эрмитов тогда и только тогда, когда матрица $A \in Mat(n, C)$ этого оператора в произвольном ортонормированном базисе эрмитова, т.е. $A = A^{\dagger}$.

2. Все собственные значения эрмитова оператора вещественные.

3. Собственные векторы эрмитова оператора, отвечающие различным собственным значениям, ортогональны друг другу.

4. Эрмитов оператор диагонализируем, а также в пространстве V существует ортонормированный базис, состоящий из собственных векторов эрмитова оператора $\hat{A}$.

Определение 1.51

Унитарный оператор $\hat{U} \in EndV$ – оператор в унитарном пространстве $(V, \langle , \rangle)$ такой, что $\left\langle \hat{U}\vec{x}, \hat{U}\vec{y} \right\rangle = \langle \vec{x}, \vec{y} \rangle$, $\forall \vec{x}, \vec{y} \in V$.

Теорема 1.13 (свойства унитарного оператора)

1. Оператор $\hat{U}$ унитарен тогда и только тогда, когда $U \in Mat(n, C)$ этого оператора в произвольном ортонормированном базисе унитарна, т.е. $U^{\dagger}U = UU^{\dagger} = I_n$.

2. Все собственные значения унитарного оператора по модулю равны единице.

3. Собственные векторы унитарного оператора, отвечающие различным собственным значениям, ортогональны друг другу.

4. Унитарный оператор диагонализируем, а также в пространстве V существует ортонормированный базис, состоящий из собственных векторов унитарного оператора $\hat{U}$.

Понятие квантового бита

Единицей хранения информации является бит. Ячейка памяти классического компьютера объемом в 1 бит может находиться в одном из двух различных состояниях. Эти состояния принято обозначать 0 и 1, саму такую ячейку также принято называть битом. Если бит находится в состоянии 0, то говорят, что он хранит значение 0, если же он находится в состоянии 1, то говорят, что он хранит значение 1.

При выполнении вычислений над битами можно совершать различные двоичные операции, например такие:x

– отрицание (NOT): $y = \bar{x}$.

x	y
0	1
1	0

– конъюнкция («И», AND): $y = x_1 \wedge x_2$.

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

– дизъюнкция («ИЛИ», OR): $y = x_1 \vee x_2$.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Данные базовые операции являются основой любых дискретных вычислений. Любая более сложная логическая операция – это определенная комбинация базовых операций. С использованием набора базовых побитовых операций работает обычный компьютер.

Квантовый компьютер – это средство вычислительной техники, в основе которого лежат законы квантовой механики [34–38]. В квантовых компьютерах для вычисления применяются так называемые квантовые алгоритмы, которые используют эффекты

квантовой механики, например, квантовый параллелизм и квантовая запутанность [34– 38].

Квантовый компьютер оперирует так называемыми квантовыми битами. Можно определить квантовый бит, или сокращенно ***q-бит (кубит)***, как квантово-механическую систему, имеющую два состояния, обозначаемых, соответственно, как $|0\rangle$ и $|1\rangle$. Однако в отличие от классического случая в квантовой механике эти два состояния могут находиться в ***состоянии суперпозиции***, т.е. наиболее общее состояние квантового бита может быть записано как:

$$\langle\psi\rangle = \alpha|0\rangle + \beta|1\rangle$$

где α и β – комплексные коэффициенты. Другими словами, можно сказать, что законы квантовой механики допускают другие значения ***кубита***, которые называются состояниями суперпозиции. Таким образом, состояние суперпозиции представляет собой значения между экстремумами 0 и 1, а квантовый бит может принимать бесконечно много значений.

Например, проведем аналогию с выключателем света. Классический бит может принимать только одно из двух состояний – «включено» или «выключено» (рис. 2.1, а, б). Кубиты похожи на светильник с возможностью регулировки яркости (рис. 1.1, в) [6].

а б в

Рис. 1.1. Различие бита и кубита

Кубит можно определить как вектор единичной длины в двумерном гильбертовом пространстве над полем комплексных чисел [1–4]. Состояния $|0\rangle$ и $|1\rangle$ вместе представляют собой базисные вектора. Как и все векторы, они указывают направление и имеют величину. Для записи двух состояний кубитов можно использовать обозначения ***бра*** ($\langle\ |$) и ***кет*** ($|\ \rangle$) – обозначения Дирака. Векторы вида $|\ \rangle$ называются

кет-векторами, а вида $\langle\,|$ ***бра***-векторами. Обозначения кет соответствует следующим векторам: Кет-вектора, соответствующие нулевому и единичному состоянию кубита будут иметь следующий вид:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \text{ и } |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

При измерении состояния системы с волновой функцией $\langle\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ вероятность обнаружить ее в состоянии $|0\rangle$ равна α^2 , а вероятность обнаружить ее в состоянии $|1\rangle$ равна β^2. Сумма этих вероятностей равна единице:

$$|\alpha|^2 + |\beta|^2 = 1.$$

Данное соотношение называется ***условием нормализации***. То есть формула для волновой функции $|\psi\rangle$ описывает, в какой пропорции бесконечное множество всех вариантов значений квантового состояния $|\psi\rangle$ содержит варианты базисных состояний $|0\rangle$ и $|1\rangle$. Визуализация состояния кубита возможна с помощью специального инструмента, называемого сферой Блоха.

Сфера Блоха – это сфера с единичным радиусом, при этом точка на ее поверхности соответствует состоянию кубита.

Когда кубит находится в суперпозиции $|0\rangle$ и $|1\rangle$, вектор будет располагаться между двумя этими точками на сфере (угол θ). Вращение вокруг оси Z описывается углом φ, и отвечает за изменение фазы кубита. Сфера Блоха показана на рисунке 1.2.

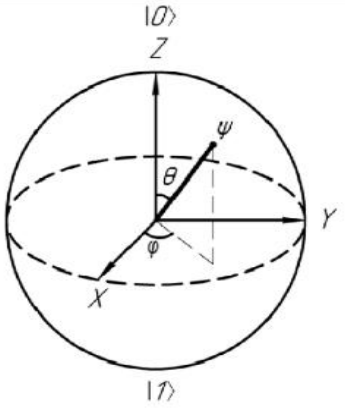


Рис. 1.2. Сфера Блоха. Состояние в верхней части сферы представляет $|0\rangle$, а состояние в нижней части сферы представляет $|1\rangle$

Физические реализации подобной системы с двумя состояниями могут быть разнообразными:

– электрон или ядро со спином 1/2, ориентированным по или против направления магнитного поля;

– атом с двумя различными энергетическими состояниями;

– фотон с горизонтальной или вертикальной поляризацией;

– и т.д.

Понятие квантовой системы

Для квантовых вычислений, как правило, требуется больше одного кубита. Система, состоящая из нескольких кубитов, представляет собой тензорное произведение составляющих ее систем. Такая система называется ***квантовой системой*** [34–38]. Состояние квантовой системы, которая состоит из n кубитов можно представить следующим выражением:

$$|q_1\rangle ... |q_n\rangle = (\alpha_0|0\rangle + \beta|1\rangle) \otimes (\alpha_1|0\rangle + \beta|1\rangle) \otimes ... \otimes (\alpha_{n-1}|0\rangle + \beta_{n-1}|1\rangle)$$

Приведем пример для системы двух кубитов. В данном случае мы имеем четырехмерный вектор единичной длины. Полностью смешанное состояние системы двух кубитов можно описать следующим образом:

$$|\psi\rangle = (\alpha_0|0\rangle + \beta|1\rangle) \otimes (\alpha_1|0\rangle + \beta|1\rangle) = \alpha_0\alpha_1|00\rangle + \alpha_0\beta_1|01\rangle + \beta_0\alpha_1|10\rangle + \beta_0\beta_1|11\rangle).$$

При этом сумма вероятностей нахождения в том или ином состоянии по-прежнему равна 1.

$$|\alpha_0\alpha_1|^2 + |\alpha_0\beta_1|^2 + |\beta_0\alpha_1|^2 + |\beta_0\beta_1|^2 = 1.$$

Как и в бинарном случае, этот набор возможных результатов называется измерительным базисом, а приводящие к ним комбинации значений – базисными состояниями. Тогда любое системное квантовое состояние мы можем записать как суперпозицию базисных состояний:

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle.$$

Физический смысл коэффициентов, стоящих в формуле перед базисными состояниями, тот же, что и для одного кубита – это пропорция вариантов значений. Однако, в отличие от одного кубита, это значения не одиночного измерения, а комбинация двух измерений.

Таким образом, квантовая система из двух кубитов может находиться в одном из четырех состояний:

– $|00\rangle$ – оба кубита при измерении дают результат $|0\rangle$.

– $|01\rangle$ – первый кубит при измерении дает $|0\rangle$, второй кубит дает $|1\rangle$.

– $|10\rangle$ – первый кубит при измерении дает $|1\rangle$, второй кубит дает $|0\rangle$.

– $|11\rangle$ – оба кубита при измерении дают результат $|1\rangle$.

Другими словами, вектор состояния n -кубитной системы существует в 2^n -мерном комплексном пространстве и представляет собой сумму 2^n базисных векторов – базисных состояний. Вероятностная природа квантовой механики проявляется в процессе измерений. Измерение является единственным способом извлечения данных, определяющих квантовое состояние. В результате измерения кубит немедленно коллапсирует. Допустим, одномерный кубит находится в состоянии суперпозиции. Если его измерить, то он примет одно конкретное значение – $|0\rangle$ или $|1\rangle$. После измерения кубита коэффициенты α и β, которыми характеризовалось его предыдущее состояние, будут утеряны. Рассмотрим n-кубит:

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle .$$

Так как длина вектора остается равной единицы, то можно записать

$$\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1.$$

Когда мы будем выполнять измерение состояния такого кубита, то получим одно из классических значений совокупности n битов от 000...0 до 111...1, где значение k (в двоичном представлении) появится с вероятностью $|\alpha_k|^2$. Следует отметить, что для каких-то состояний вероятность может оказаться нулевой!

Измерение квантовой системы

Чтобы получить информацию о состоянии квантовой системы необходимо выполнить его измерение. Для проведения измерений мы должны активно воздействовать на квантовую систему. В процессе измерения квантовой системы в выбранном базисе мы получим один из векторов этого базиса. В результате сразу же после измерения состояние квантовой системы разрушается, то есть система переходит в состояние, соответствующее наблюденному значению [34, 35].

Приведем пример возможных исходов измерения для одно-кубитной системы: – Если система находилась в состоянии $|0\rangle$, то при ее измерении мы получим тоже $|0\rangle$.

– Если система находилась в состоянии $|1\rangle$, то при ее измерении мы получим тоже $|1\rangle$.

– Если система находилась в состоянии суперпозиции $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, то в результате ее измерения мы получим вектор $|0\rangle$ с вероятностью $|\alpha|^2$, а вектор $|1\rangle$, с вероятностью $|\beta|^2$.

Следует отметить, что вероятность получения в процессе измерения конкретного вектора выбранного базиса ($|0\rangle$ или $|1\rangle$) равна квадрату модуля скалярного произведения вектора данной системы на этот вектор [1]. При этом в результате измерений невозможно определить значения α и β, а также невозможно измерить повторно ту же самую систему, потому что после измерения состояние квантовой системы разрушается. Таким образом, для получения максимально достоверной информации о состоянии системы при ее измерении необходимо выбирать базис, один из векторов которого наиболее близок с измеряемым кубитом [34–37].

В общем случае вероятностный процесс измерения квантовой системы происходит следующим образом:

– Пусть у нас имеется квантовое состояние системы из n кубитов ψ и стандартный базис пространства состояний системы $k = \{|0\rangle, ..., |2^n - 1\rangle\}$

$$|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle .$$

– При измерении состояния системы по отношению к базису k с вероятностью $|\alpha_i|^2$ результат измерения будет – $|i\rangle$.

– После измерения квантовая система будет находиться в состоянии – $|i\rangle$.

– После измерений амплитуды состояний отличных от $|i\rangle$ будут равны нулю: $\alpha_k = 0$ для $k \neq i$. Рассмотрим пример измерения 2-кубита.

Пример 1.1: Выполним измерение следующего 2-кубита

$$|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle,$$
$$\alpha_0 = 0.3;\ \ \alpha_1 = 0.1;\ \alpha_2 = 0.9;\ \alpha_3 = 0.3;$$

При измерении данного 2-кубита возможны четыре варианта:

– С вероятностью 0.09 получится значение 00 и 2-кубит перейдет в состояние $|00\rangle$.

– С вероятностью 0.01 получится значение 01 и 2-кубит перейдет в состояние $|01\rangle$.

– С вероятностью 0.81 получится значение 10 и 2-кубит перейдет в состояние $|10\rangle$.

– С вероятностью 0.09 получится значение 11 и 2-кубит перейдет в состояние $|11\rangle$.

В некоторых задачах требуется измерить состояние лишь некоторых кубитов в большом n-кубите. Приведем пример подобной ситуации для 3-кубита.

Пример 1.2: Выполним измерение первых двух битов кубита, но не будем трогать третий бит.

$$|\psi\rangle = \alpha_0|000\rangle + \alpha_1|001\rangle + \alpha_2|010\rangle + \alpha_3|011\rangle + \alpha_5|101\rangle + \alpha_6|110\rangle,$$
$$\alpha_0 = 0.3;\ \alpha_1 = -0.6;\ \alpha_2 = -0.1;\ \alpha_3 = -0.7;\ \alpha_5 = 0.1;\ \alpha_6 = -0.2;$$

Видно, что в заданном нами примере отсутствуют состояния $|100\rangle$ и $|111\rangle$, т.е. их коэффициенты α_4 и α_7 равны 0. Так как нам необходимо измерить только два первых кубита, то возможны четыре варианта исхода: 00, 01, 10, 11. После измерения первые два кубита получат определенные значения, а значение третьего кубита не будет фиксировано.

– Вероятность наблюдения значения 00 будет определяться как $\alpha_0^2 + \alpha_1^2 = 0.09 + 0.36 = 0.45$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_0^2 + \alpha_1^2}}(\alpha_0|000\rangle + \alpha_1|001\rangle),$$
$$\alpha_0 = 0.3;\ \alpha_1 = -0.6;$$

– Вероятность наблюдения значения 01 будет определяться как $\alpha_2^2 + \alpha_3^2 = 0.01 + 0.49 = 0.5$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_2^2 + \alpha_3^2}}(\alpha_2|010\rangle + \alpha_3|011\rangle),$$
$$\alpha_2 = -0.1;\ \alpha_3 = -0.7;$$

Стоит отметить, что для нормализации состояния третьего ненаблюдаемого кубита необходимо коэффициенты этих двух состояний разделить на корень квадратный из вероятности появления данного исхода.

– Вероятность наблюдения значения 10 будет определяться как $\alpha_5^2 = 0.01$, при этом новое состояние системы будет $|101\rangle$.

– Вероятность наблюдения значения 11 будет определяться как $\alpha_6^2 = 0.04$, при этом новое состояние системы будет $|110\rangle$. Заметьте, учитывая особенности нашего 3-кубита (отсутствие состояний $|101\rangle$ и $|110\rangle$), в последних двух случаях вероятность того, что третий бит соответственно принимает значения 1 и 0, равна 1.

Пример 1.3: Выполним измерение последних двух кубитов, но не будем трогать первый кубит.

$$|\psi\rangle = \alpha_0|000\rangle + \alpha_1|001\rangle + \alpha_2|010\rangle + \alpha_3|011\rangle + \alpha_5|101\rangle + \alpha_6|110\rangle,$$

$$\alpha_0 = 0.3;\ \alpha_1 = -0.6;\ \alpha_2 = -0.1;\ \alpha_3 = -0.7;\ \alpha_5 = 0.1;\ \alpha_6 = -0.2;$$

– Вероятность наблюдения значения 00 будет определяться как $|\alpha_0|^2 = 0.09$, при этом новое состояние системы будет $|000\rangle$.

– Вероятность наблюдения значения 01 будет определяться как $\alpha_1^2 + \alpha_5^2 = 0.36 + 0.01 = 0.37$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_1^2 + \alpha_5^2}}(\alpha_1|001\rangle + \alpha_5|101\rangle),$$

$$\alpha_5 = 0.1;\ \alpha_1 = -0.6;$$

– Вероятность наблюдения значения 10 будет определяться как $\alpha_2^2 + \alpha_6^2 = 0.01 + 0.04 = 0.05$, при этом новое состояние системы будет следующим:

$$|\psi\rangle = \frac{1}{\sqrt{\alpha_2^2 + \alpha_6^2}}(\alpha_2|010\rangle + \alpha_6|110\rangle),$$

$$\alpha_2 = 0.1;\ \alpha_6 = -0.2;$$

– Вероятность наблюдения значения 11 будет определяться как $\alpha_6^2 = 0.04$, при этом новое состояние системы будет $|011\rangle$.

1.2. Квантовые гейты

Классические дискретные цепи представляют собой совокупность проводников и набора логических гейтов (совокупности полупроводниковых приборов). Логические гейты осуществляют преобразование информации, поступающей на их входы. В квантовых компьютерах преобразование информации осуществляется с использованием так называемых квантовых гейтов [34, 35].

1.2.1. Гейт Адамара

Одиночный кубит по определению является суперпозицией двух квантовых состояний $|0\rangle$ и $|1\rangle$, каждое из которых может рассматриваться как носитель одного бита классической информации $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Чтобы создать состояния суперпозиции, используется гейт, называемый ***гейт Адамара (H). Гейт Адамара*** является одним из наиболее полезных квантовых гейтов. Он задается матрицей:

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Учитывая, что состояния кубита могут быть представлены в виде:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix};\ |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix};\ |\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

запишем выражения показывающие действия оператора Адамара на кубиты.

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ -1 \end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle,$$

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix} \alpha+\beta \\ \alpha-\beta \end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle.$$

Таким образом, действие оператора H на произвольный кубит можно описать формулой:3

$$H|\psi\rangle = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle.$$

В геометрической интерпретации кубита на сфере Блоха можно заметить, что в результате действия гейта Адамара на кубит в состоянии $|0\rangle$ переводит его в положение между состояниями $|0\rangle$ и $|1\rangle$, т.е. в состояние $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. Соответственно, в результате действия гейта Адамара состояние кубита $|1\rangle$ переводит его в положение между состояниями $|0\rangle$ и $|1\rangle$, только в другой полусфере, т.е. в состояние $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$.

Действие гейта Адамара на сфере Блоха показано на рисунке 1.3.

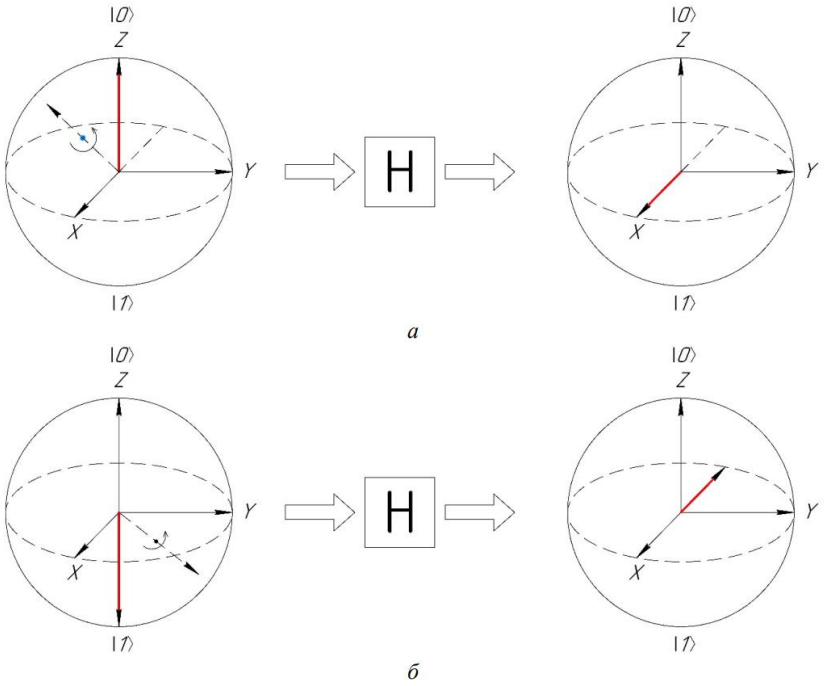


Рис. 1.3. Действие гейта Адамара: а) начальное состояние кубита $|0\rangle$; б) начальное состояние кубита $|1\rangle$

Оператор H является самосопряженным, а, следовательно, обратным к себе самому, его повторное применение к базису Адамара вернет нам обычный базис. Другими словами, алгебраические вычисления дают $H^2 = 1$. То есть двукратное применение гейта H возвращает систему в исходное положение. Покажем самосопряженность гейта Адамара на примерах.

Пример 1.4. Для кубита в состоянии $|0\rangle$:

$$H|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$HH|0\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \left(\frac{1}{\sqrt{2}}\right)^2\begin{pmatrix}2\\0\end{pmatrix} = \begin{pmatrix}1\\0\end{pmatrix} = |0\rangle.$$

Пример 1.5. Для кубита в состоянии $|1\rangle$:

$$H|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle,$$

$$HH|1\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix} = \left(\frac{1}{\sqrt{2}}\right)^2\begin{pmatrix}0\\2\end{pmatrix} = \begin{pmatrix}0\\1\end{pmatrix} = |1\rangle.$$

Пример 1.6. Для кубита в состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$:

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\\alpha-\beta\end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle,$$

$$HH|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \left(\frac{1}{\sqrt{2}}\right)^2\begin{pmatrix}2\alpha\\ 2\beta\end{pmatrix} = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle .$$

Если у нас имеется система из n кубитов, то применив ***оператор Адамара*** ко всем кубитам индивидуально, мы получим суперпозицию всех 2^n состояний:

$$(H \otimes H \otimes H \otimes H \ldots \otimes H \otimes H)|0000\ldots00\rangle =$$

$$= \frac{1}{\sqrt{2^n}}((|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle) \otimes \ldots \otimes (|0\rangle + |1\rangle)) = \frac{1}{\sqrt{2^n}}\sum_{z=0}^{2^n-1}|z\rangle$$

1.2.2. Оператор NOT (Гейт X)

В классических дискретных вычислениях оператор NOT – это оператор инверсии. В соответствии с этим определением классического оператора NOT, квантовый гейт X (т.е. гейт преобразующий информацию внутри кубита) может быть определен по аналогии:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Rightarrow NOT|\psi\rangle = \alpha|1\rangle + \beta|0\rangle .$$

Квантовым аналогом классического оператора NOT является матрица вида:

$$X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}.$$

Покажем действия гейта X на кубит в различных состояниях.

$$X|0\rangle = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} = |1\rangle ,$$

$$X|1\rangle = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix} = |0\rangle ,$$

$$X|\psi\rangle = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha\end{pmatrix} = \beta|0\rangle + \alpha|1\rangle .$$

С точки зрения интерпретации действия данного гейта на состояние ***кубита*** с помощью сферы Блоха можно заметить, что происходит поворот вектора состояния на 180 градусов вокруг оси X (рис. 1.4).

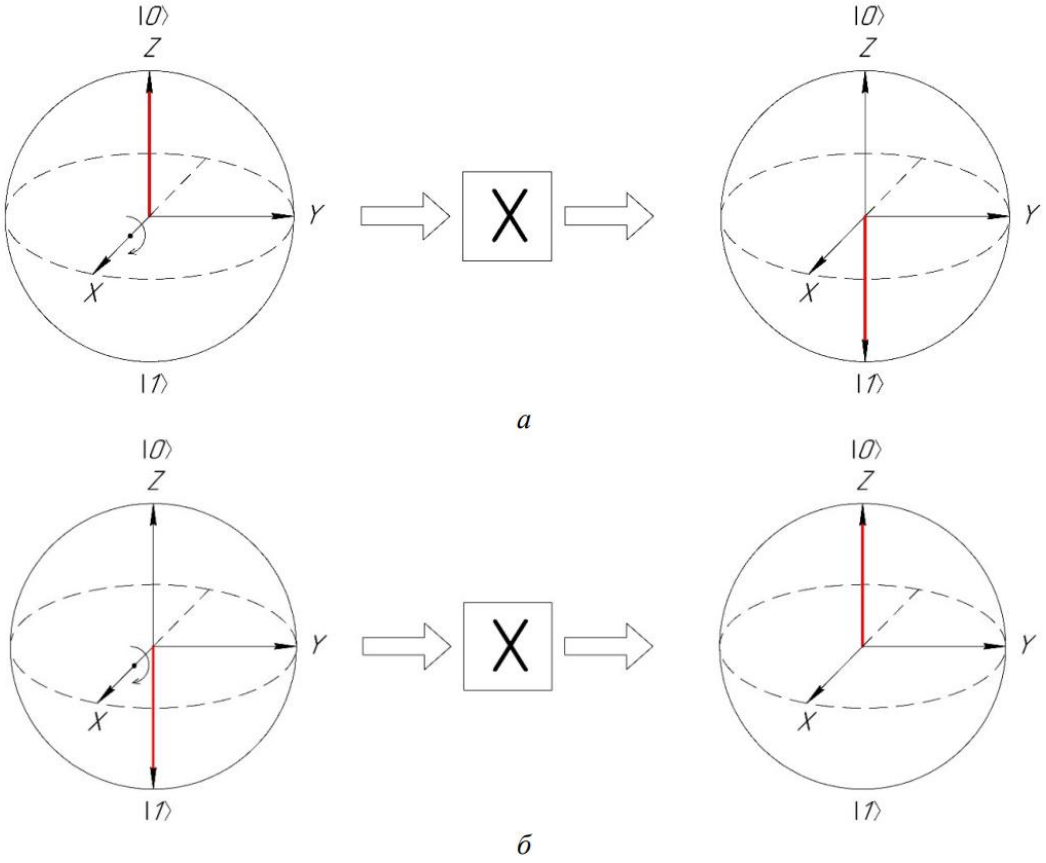


Рис. 1.4. Действие гейта X: а) начальное состояние кубита $|0\rangle$; б) начальное состояние кубита $|1\rangle$

1.2.3. Гейт Z

Определим сначала действие ***гейта*** Z на базисные вектора. Потребуем, чтобы он не изменял 0, а 1 переводил в –1:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \Rightarrow Z|\psi\rangle = \alpha|0\rangle - \beta|1\rangle$$

Тогда действию данного ***гейта*** Z отвечает матрица:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Покажем действия ***гейта*** Z на кубит в различных состояниях.

$$Z|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = |0\rangle,$$

$$Z|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} = -|1\rangle,$$

$$Z|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ -\beta \end{pmatrix} = \alpha|0\rangle - \beta|1\rangle.$$

На сфере Блоха действие гейта Z соответствует повороту вектора вокруг оси Z на угол π (рис. 1.5).

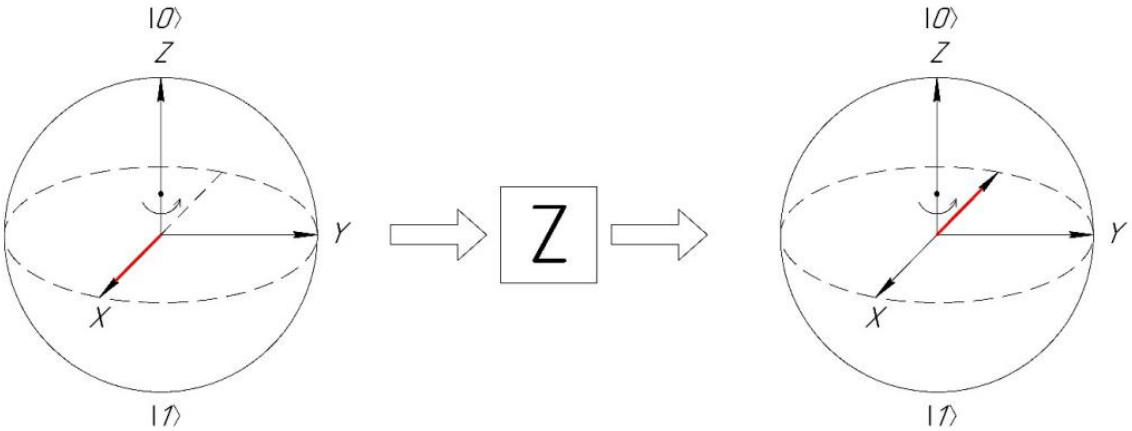


Рис. 1.5. Действие ***гейта*** Z

Отметим следующие два свойства гейтов X и Z**:**

$$HXH = Z,$$

$$HZH = X.$$

Приведем примеры показывающие данные свойства.

Пример 1.7. Для кубита в состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ применим операции HXH.

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle,$$

$$XH|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{\alpha-\beta}{\sqrt{2}}|0\rangle + \frac{\alpha+\beta}{\sqrt{2}}|1\rangle,$$

$$HZH|\psi\rangle = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha+\beta\\ \beta-\alpha\end{pmatrix} = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix}2\beta\\ 2\alpha\end{pmatrix} = \beta|0\rangle + \alpha|1\rangle.$$

Как было показано ранее $X|\psi\rangle = \beta|0\rangle + \alpha|1\rangle$, а, следовательно, можно заметить, что $HZH = X$.

1.2.4. Гейт Y

Гейт Y задается матрицей:

$$Y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}.$$

В отличие от предыдущих рассмотренных нами элементов, ***гейт*** Y является комплексным. Покажем действия ***гейта*** Y на кубит в различных состояниях.

$$Y|0\rangle = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}0\\ i\end{pmatrix} = i|1\rangle, \quad Y|1\rangle = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}-i\\ 0\end{pmatrix} = -i|1\rangle,$$

$$Y|\psi\rangle = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}-i\beta\\ i\alpha\end{pmatrix} = -i\beta|0\rangle + i\alpha|1\rangle.$$

На сфере Блоха действие ***гейта*** Y соответствует повороту вектора вокруг оси Y на угол π (рис. 1.6).

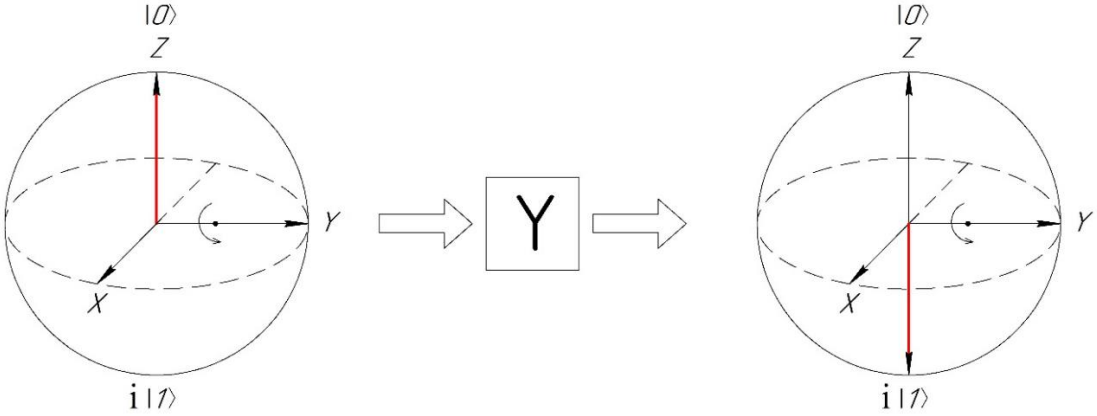


Рис. 1.6. Действие ***гейта*** Y

Отметим следующее свойства ***гейта*** Y:

$$HYH = -Y.$$

Приведем пример, показывающий данное свойство.

Пример 1.8. Для кубита в состоянии $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ применим операции HYH.

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{\alpha+\beta}{\sqrt{2}}|0\rangle + \frac{\alpha-\beta}{\sqrt{2}}|1\rangle,$$

$$YH|\psi\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}-i(\alpha-\beta)\\ i(\alpha+\beta)\end{pmatrix} = \frac{-i(\alpha-\beta)}{\sqrt{2}}|0\rangle + \frac{i(\alpha+\beta)}{\sqrt{2}}|1\rangle,$$

$$HYH|\psi\rangle = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}-i(\alpha-\beta)\\ i(\alpha+\beta)\end{pmatrix} = \frac{1}{\sqrt{2}}\frac{1}{\sqrt{2}}\begin{pmatrix}2i\beta\\ -2i\alpha\end{pmatrix} = i\beta|0\rangle - i\alpha|1\rangle.$$

Как было показано ранее $Y|\psi\rangle = -i\beta|0\rangle + i\alpha|1\rangle$, а, следовательно, можно заметить, что $HYH = -Y$.

1.2.5. Гейт S

Гейт S задается матрицей:

$$S = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}.$$

Покажем действия гейта S на кубит в различных состояниях.

$$S|0\rangle = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}\begin{pmatrix}1\\ 0\end{pmatrix} = \begin{pmatrix}1\\ 0\end{pmatrix} = i|0\rangle,$$

$$S|1\rangle = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}\begin{pmatrix}0\\ 1\end{pmatrix} = \begin{pmatrix}0\\ i\end{pmatrix} = i|1\rangle,$$

$$S|\psi\rangle = \begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\alpha\\ i\beta\end{pmatrix} = \alpha|0\rangle + i\beta|1\rangle.$$

Покажем действие ***гейта*** S на сфере Блоха (рис. 1.7).

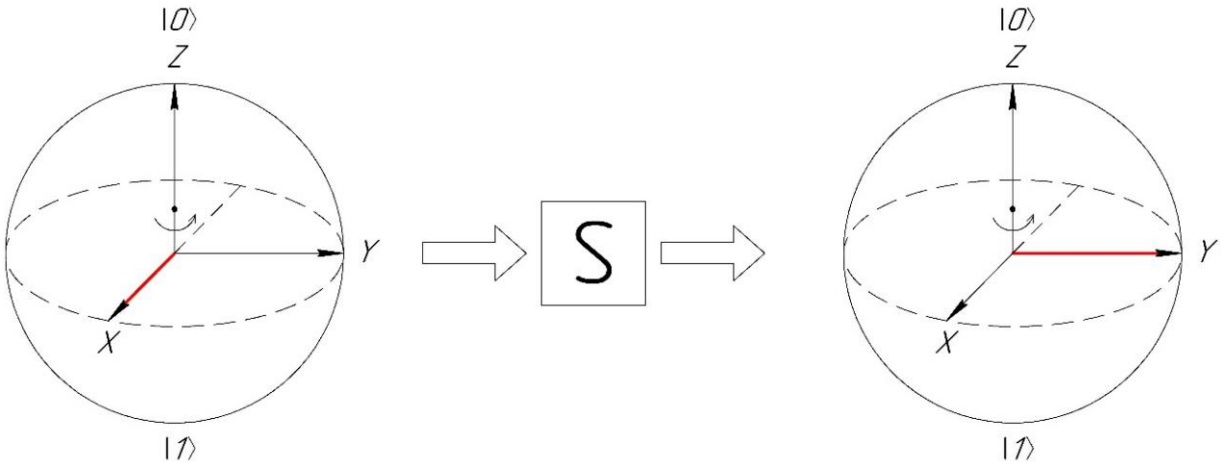


Рис. 1.7 Действие гейта S

1.2.6. Гейт T

Рассмотрим еще один комплексный логический ***гейт*** T, который часто обозначается как $\frac{\pi}{4}$. ***Гейт*** T задается матрицей:

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix}.$$

Название ***гейта*** T определяется историческими причинами и возможностью представления матрицы этого ***гейта*** с точностью до общего фазового множителя $e^{i\frac{\pi}{8}}$ в виде.

$$T = e^{i\frac{\pi}{8}} \begin{pmatrix} e^{-i\frac{\pi}{8}} & 0 \\ 0 & e^{i\frac{\pi}{8}} \end{pmatrix}.$$

Покажем действия ***гейта*** T на ***кубит*** в различных состояниях.

$$T|0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} = i|0\rangle,$$

$$T|1\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ e^{i\frac{\pi}{4}} \end{pmatrix} = e^{i\frac{\pi}{4}}|1\rangle,$$

$$T|\psi\rangle = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \alpha \\ e^{i\frac{\pi}{4}}\beta \end{pmatrix} = \alpha|0\rangle + e^{i\frac{\pi}{4}}\beta|1\rangle.$$

Покажем действие ***гейта*** T на сфере Блоха (рис. 1.8).

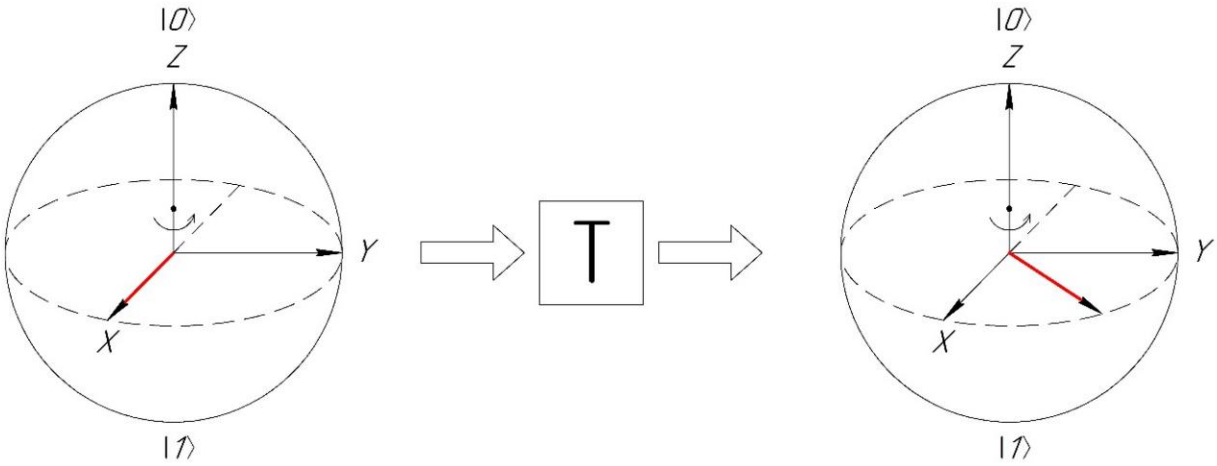


Рис. 1.8. Действие ***гейта*** T

Таким образом, ***гейт*** T не изменяет коэффициент при базисном векторе 0 и меняет фазу коэффициента при базисном векторе 1.

1.2.7. Гейты поворота Rx, Ry, Rz

Гейт Rx на сфере Блоха соответствует вращению кубита вокруг оси X на заданный угол. В матричном виде данный ***гейт*** можно записать как [1, 2, 6]:

$$Rx(\theta) = e^{-i\frac{\theta}{2}X} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\cdot\sin\left(\frac{\theta}{2}\right) \\ -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}.$$

По аналогии с ***гейтом*** Rx, ***гейты*** Ry и Rz соответствуют вращению ***кубита*** вокруг осей Y и Z. При этом их можно определить как:

$$Ry(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix},$$

$$Rz(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}.$$

1.2.8. Произвольные однокубитные унитарные гейты U

Произвольный однокубитовый унитарный оператор может быть записан в виде [34,35, 40]:

$$U = e^{i\alpha} R_{\vec{n}}(\theta),$$

где $R_{\vec{n}}(\theta)$ – оператор поворота на угол θ вокруг оси, определенной единичным вектором $\vec{n}$, α и θ – действительные числа.

Гейт U1 осуществляет вращение одного кубита вокруг оси Z. В матричном виде данный гейт можно записать как:

$$U1(\lambda) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\lambda} \end{pmatrix}.$$

В зависимости от значения угла λ данный гейт является эквивалентом других гейтов:

$$U1(\pi) = Z,$$
$$U1(\pi/2) = S,$$
$$U1(\pi/4) = T.$$

Гейт U2 осуществляет вращение одного кубита вокруг $X+Y$ осей. Согласно теореме $X-Y$ разложения для однокубитового гейта, данный оператор определяется как:

$$U2(\varphi,\lambda) = Rz\left(\varphi+\frac{\pi}{2}\right)Rx\left(\frac{\pi}{2}\right)Rz\left(\lambda-\frac{\pi}{2}\right).$$

В матричном виде данный гейт можно записать как:

$$U2(\varphi,\lambda) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & -e^{i\lambda} \\ e^{i\varphi} & e^{i(\varphi+\lambda)} \end{pmatrix}.$$

Можно заметить, что $U2(0,\pi) = H$.

Гейт U3 – это универсальный однокубитный поворотный затвор с тремя углами Эйлера. Данный гейт определяется как:

$$U3(\theta,\varphi,\lambda) = Rz\left(\varphi-\frac{\pi}{2}\right)Rx\left(\frac{\pi}{2}\right)Rz(\pi-0)Rx\left(\frac{\pi}{2}\right)Rz\left(\lambda-\frac{\pi}{2}\right),$$

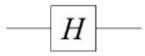
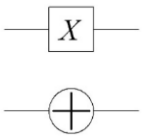
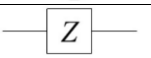
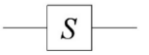
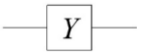
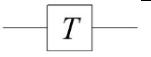
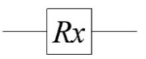
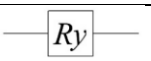
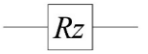
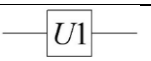
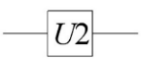
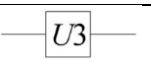
$$U3(\theta,\varphi,\lambda) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}.$$

Можно заметить, что $U3(\theta,-\frac{\pi}{2},\frac{\pi}{2}) = Rx(\theta)$ и $U3(\theta,0,0) = Ry(\theta)$. В таблице 1.1 представлены графические обозначение однобитовых гейтов и результат их воздействия на произвольный кубит $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Используя гейты поворота R и унитарные гейты U, можно получать произвольные состояния суперпозиции $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$. Приведем примеры получения различных состояний суперпозиции кубита.

Таблица 1.1.

Однокубитные гейты

Название	**Графическое обозначение**	**Матрица**	**Результат воздействия**
Гейт **H**		$H=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}$	$\frac{\alpha+\beta}{\sqrt{2}}\|0\rangle+\frac{\alpha-\beta}{\sqrt{2}}\|1\rangle$
Гейт **X**		$X=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$	$\beta\|0\rangle+\alpha\|1\rangle$
Гейт **Z**		$Z=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$	$\alpha\|0\rangle-\beta\|1\rangle$
Гейт **S**		$S=\begin{pmatrix}1 & 0\\ 0 & i\end{pmatrix}$	$\alpha\|0\rangle+i\beta\|1\rangle$
Гейт **Y**		$Y=\begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$	$-i\beta\|0\rangle+i\alpha\|1\rangle$
Гейт **T**		$T=\begin{pmatrix}1 & 0\\ 0 & e^{i\frac{\pi}{4}}\end{pmatrix}$	$\alpha\|0\rangle+e^{i\frac{\pi}{4}}\beta\|1\rangle$
Гейт **Rx**		$Rx(\theta)=\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -i\cdot\sin\left(\frac{\theta}{2}\right)\\ -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix}$	$\left(\alpha\cos\left(\frac{\theta}{2}\right)-i\beta\sin\left(\frac{\theta}{2}\right)\right)\|0\rangle+$ $+\left(\beta\cos\left(\frac{\theta}{2}\right)+i\alpha\sin\left(\frac{\theta}{2}\right)\right)\|1\rangle$
Гейт **Ry**		$Ry(\theta)=\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right)\\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix}$	$\left(\alpha\cos\left(\frac{\theta}{2}\right)-\beta\sin\left(\frac{\theta}{2}\right)\right)\|0\rangle+$ $+\left(\alpha\sin\left(\frac{\theta}{2}\right)+\beta\cos\left(\frac{\theta}{2}\right)\right)\|1\rangle$
Гейт **Rz**		$Rz(\theta)=\begin{pmatrix}e^{-i\frac{\theta}{2}} & 0\\ 0 & e^{i\frac{\theta}{2}}\end{pmatrix}$	$\alpha e^{-i\frac{\theta}{2}}\|0\rangle+\beta e^{i\frac{\theta}{2}}\|1\rangle$
Гейт **U1**		$U1(\lambda)=\begin{pmatrix}1 & 0\\ 0 & e^{i\lambda}\end{pmatrix}$	$\alpha\|0\rangle+\beta e^{i\lambda}\|1\rangle$
Гейт **U2**		$U2(\varphi,\lambda)=\frac{1}{\sqrt{2}}\begin{pmatrix}1 & -e^{i\lambda}\\ e^{i\varphi} & e^{i(\varphi+\lambda)}\end{pmatrix}$	$\frac{\left(\alpha-\beta e^{i\lambda}\right)}{\sqrt{2}}\|0\rangle+\frac{\left(\alpha e^{i\varphi}+\beta e^{i(\varphi+\lambda)}\right)}{\sqrt{2}}\|1\rangle$
Гейт **U3**		$U3(\theta,\varphi,\lambda)=\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right)\\ e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right)\end{pmatrix}$	$\left(\alpha\cos\left(\frac{\theta}{2}\right)-\beta e^{i\lambda}\sin\left(\frac{\theta}{2}\right)\right)\|0\rangle+$ $+\left(\alpha e^{i\varphi}\sin\left(\frac{\theta}{2}\right)+\beta e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right)\right)\|1\rangle$

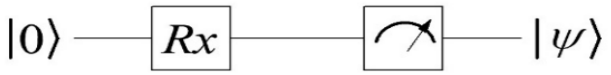
Пример 1.9: Получим кубит в состоянии суперпозиции $|\psi\rangle=\sqrt{0.8}|0\rangle+\sqrt{0.2}|1\rangle$, применив воздействие гейта *Rx* к ***кубиту***, который находится в исходном состоянии $|0\rangle$:

$$Rx(\theta)|0\rangle = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ -i\cdot\sin\left(\frac{\theta}{2}\right) \end{pmatrix} = \cos\left(\frac{\theta}{2}\right)|0\rangle - i\cdot\sin\left(\frac{\theta}{2}\right)|1\rangle .$$

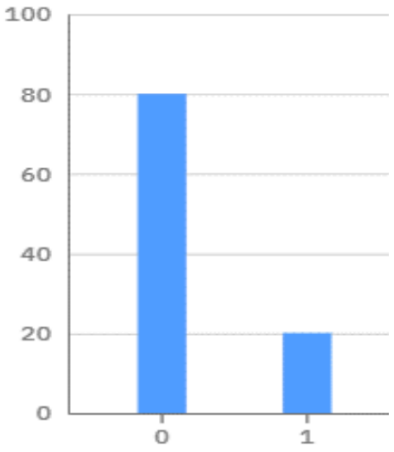
Таким образом, для получения необходимого состояния нужно рассчитать правильный угол поворота θ . Для этого решим уравнение:

$$\cos\left(\frac{\theta}{2}\right)^2 = 0.8 \Rightarrow \theta \approx 0.93 рад .$$

На рисунке 1.9 представлена квантовая схема, реализующая подобное состояние суперпозиции и результат измерений состояния ***кубита*** с использованием системы **IBM** Quantum Experience.



а



б

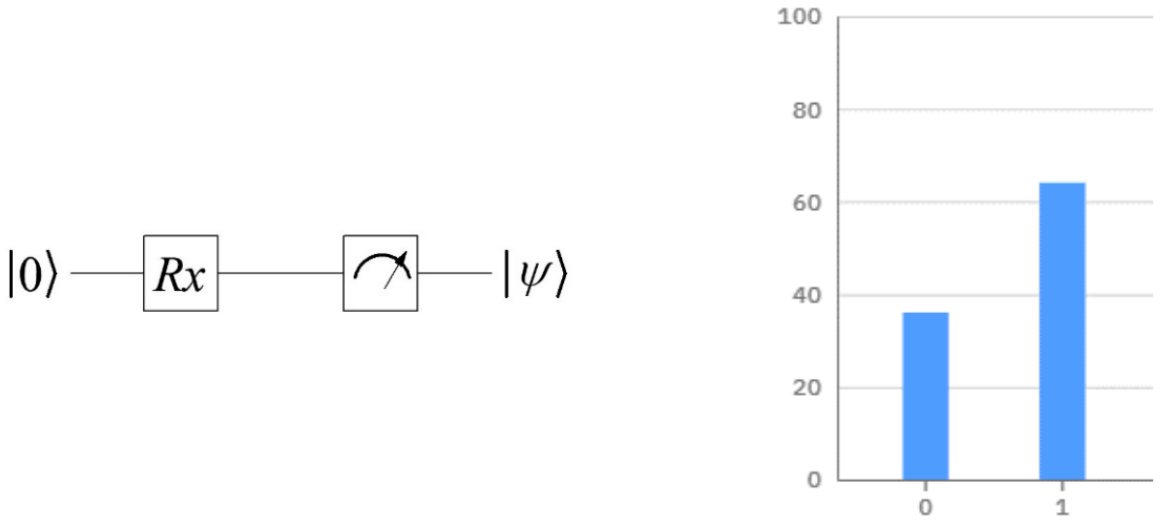
Рис. 1.9 Квантовая схема получения кубита в состоянии суперпозиции $|\psi\rangle = \sqrt{0.8}|0\rangle + \sqrt{0.2}|1\rangle$ (а) и результат симуляции (б)

Пример 1.10: Получим кубит в состоянии суперпозиции $|\psi\rangle = 0,6|0\rangle + 0,8|1\rangle$, применив воздействие гейта Rx к кубиту, который находится в исходном состоянии $|0\rangle$. Состояние $|\psi\rangle = 0,6|0\rangle + 0,8|1\rangle$ соответствует тому, что данный кубит будет принимать значение $|0\rangle$ с вероятностью 36% и значение $|1\rangle$ с вероятностью 64%.

Решим уравнение:

$$\cos\left(\frac{\theta}{2}\right)^2 = 0.36 \Rightarrow \theta \approx 1,85 рад .$$

На рисунке 1.10 представлена квантовая схема, реализующая подобное состояние суперпозиции и результат измерений состояния ***кубита*** с использованием системы **IBM** Quantum Experience.



а *б*

Рис. 1.10. Квантовая схема получения кубита в состоянии суперпозиции $|\psi\rangle = 0,6|0\rangle + 0,8|1\rangle$ (а) и результат симуляции (б)

1.2.9. Контролируемые гейты

Для квантовых вычислений, как правило, требуется больше одного кубита:

$$|q_1\rangle ... |q_n\rangle = (\alpha_0|0\rangle + \beta|1\rangle) \otimes (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes ... \otimes (\alpha_{n-1}|0\rangle + \beta_{n-1}|1\rangle).$$

Например, система двух кубитов описывается как:

$$|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle = (\alpha_0|0\rangle + \beta_0|1\rangle) \otimes (\alpha_1|0\rangle + \beta_1|1\rangle) =$$
$$= \alpha_0\alpha_1|00\rangle + \alpha_0\beta_1|01\rangle + \beta_0\alpha_1|10\rangle + \beta_0\beta_1|11\rangle$$
$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \lambda|10\rangle + \delta|11\rangle.$$

Простейшим двукубитным контролируемым гейтом является ***гейт*** *CNOT*. Данный гейт имеет два кубита на входе и два кубита на выходе. При этом один из кубитов называется контролирующим, а второй – контролируемым. Процесс выполнения ***гейта*** *CNOT* является следующим: если контролирующий кубит находится в состоянии $|1\rangle$, тогда контролируемый кубит подвергается квантовой операции *NOT*, если контролирующий кубит находится в состоянии $|0\rangle$, тогда контролируемый кубит остается без изменения. Графическое обозначение квантового ***гейта*** *CNOT* представлено на рисунке 1.11.

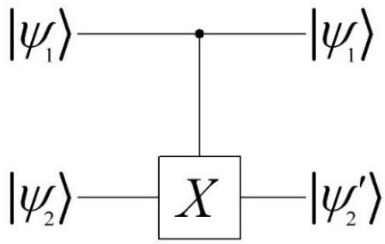


Рис. 1.11. Графическое обозначение квантового ***гейта*** *CNOT*

Для пары кубитов $|\psi_1\rangle$ и $|\psi_2\rangle$ в качестве базисных можно выбрать вектора, являющиеся произведением базисных векторов отдельных кубитов [34, 35]:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \qquad |01\rangle = |0\rangle \otimes |1\rangle = \begin{pmatrix}1\\0\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\1\\0\\0\end{pmatrix},$$

$$|10\rangle = |1\rangle \otimes |0\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \qquad |11\rangle = |1\rangle \otimes |1\rangle = \begin{pmatrix}0\\1\end{pmatrix} \otimes \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

В общем случае можно записать:

$$|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \lambda|10\rangle + \delta|11\rangle = \begin{pmatrix}\alpha\\\beta\\\lambda\\\delta\end{pmatrix}.$$

Исходя из этого, можно определить матрицу ***гейта*** *CNOT*:

$$CNOT = \begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}.$$

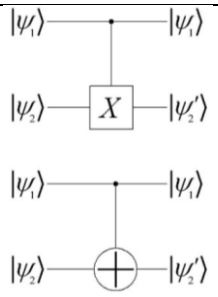
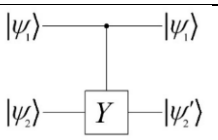
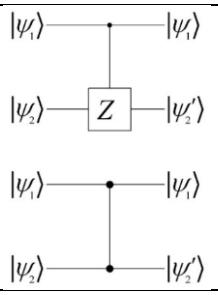
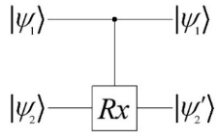
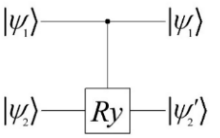
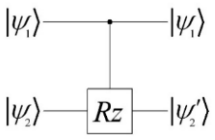
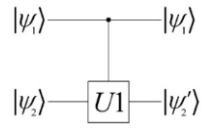
В данном случае для состояния $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \lambda|10\rangle + \delta|11\rangle$ первый (слева) кубит будет контролирующим, а второй кубит контролируемым. При этом действия ***гейта*** *CNOT* будет следующим:

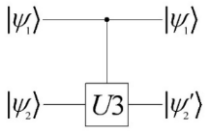
$$CNOT|00\rangle \Rightarrow |00\rangle,$$
$$CNOT|01\rangle \Rightarrow |01\rangle,$$
$$CNOT|10\rangle \Rightarrow |10\rangle,$$
$$CNOT|11\rangle \Rightarrow |11\rangle.$$

Аналогичным образом может быть определен произвольный контролируемый унитарный оператор (табл. 1.2).

Таблица 1.2.

Контролируемые квантовые гейты

Название	Графическое обозначение	Матрица
CXgate CNOT		$CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
CYgate		$CY = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \end{pmatrix}$
CZgate		$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$
CRxgate		$CRx = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -i\cdot\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & -i\cdot\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$
CRygate		$CRy = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}$
CRzgate		$CRz = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{-i\frac{\lambda}{2}} & 0 \\ 0 & 0 & 0 & e^{i\frac{\lambda}{2}} \end{pmatrix}$
CU1gate		$CU1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{pmatrix}$

CU3gate		$CU3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\left(\frac{\theta}{2}\right) & -e^{i\lambda}\sin\left(\frac{\theta}{2}\right) \\ 0 & 0 & e^{i\varphi}\sin\left(\frac{\theta}{2}\right) & e^{i(\varphi+\lambda)}\cos\left(\frac{\theta}{2}\right) \end{pmatrix}$

1.3 Квантовые алгоритмы

Квантовый параллелизм – это фундаментальное свойство квантовых вычислений. Данное свойство позволяет квантовым компьютерам вычислять функцию $f(x)$ для различных значений x одновременно. Как отмечалось ранее, в классических компьютерах состояние бита представляется либо 0, либо 1. В квантовом же компьютере состояние кубита можно представить не только как 0 или 1, а как комбинацию двух этих значений. Поэтому применение какой-либо операции кубиту происходит сразу со всеми его значениями одновременно. В качестве примера рассмотрим действие ***гейта*** X. Допустим, изначально кубит находился в состоянии $|\psi\rangle = 0.8|1\rangle + 0.6|0\rangle$, после действия гейта инверсии кубит будет находиться в состоянии $|\psi\rangle = 0.8|1\rangle + 0.6|0\rangle$. Таким образом, одна операция повлияла сразу на оба значения кубита. Это и есть квантовый параллелизм. Однако для получения информации, которая хранится в суперпозиционном состоянии, нужно выполнить операцию измерения состояний кубитов. При этом измерение кубита всегда дает только один вариант из всего множества возможных вариантов. Квантовый параллелизм лежит в основе всех квантовых алгоритмов. Благодаря возможности воздействовать сразу на все состояния кубитов системы эффективность квантовых алгоритмов значительно выше, чем у классических компьютерных алгоритмов.

Перепутанные состояния двух кубитов. Базис Белла

Рассмотрим два незапутанных кубита. Незапутанность двух кубитов подразумевает, что измерение первого кубита не влияет на результат измерения второго кубита. Зададим их состояния:

$$|\psi_1\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle,$$
$$|\psi_2\rangle = \beta_0|0\rangle + \beta_1|1\rangle.$$

Состояние пары кубитов получается перемножение одиночных состояний:

$$|\psi\rangle = |\psi_1\psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = (\alpha_0|0\rangle + \alpha_1|1\rangle)(\beta_0|0\rangle + \beta_1|1\rangle),$$

$$|\psi\rangle = |\psi_1\psi_2\rangle = \alpha_0\beta_0|00\rangle + \alpha_0\beta_1|01\rangle + \alpha_1\beta_0|10\rangle + \alpha_1\beta_1|11\rangle),$$
$$|\psi\rangle = |\psi_1\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle).$$

В данном случае, как отмечалось в главе 1, вероятность событий $|00\rangle$, $|01\rangle$, $|10\rangle$ и $|11\rangle$. равна произведению вероятностей его составляющих – $\alpha_0\beta_0$, $\alpha_0\beta_1$, $\alpha_1\beta_0$ и $\alpha_1\beta_1$ соответственно. Перепутанное состояние двух кубитов – это состояние, при котором измерение одного из кубитов однозначно определяет состояние второго кубита. Например, если в результате измерения одного кубита получилось, что состояние $|0\rangle$, то измерение другого кубита однозначно даст значение $|0\rangle$, а если в результате измерения одного кубита получилось, что состояние $|1\rangle$, то измерение другого кубита также даст значение $|1\rangle$.

В этом примере амплитуды вероятности групп $|01\rangle$ и $|10\rangle$ равны нулю. Такое состояние системы двух кубитов можно записать следующим образом [34, 35]:

$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right),$$
$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right),$$
$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle - |11\rangle\right),$$
$$|\psi\rangle = |\psi_1\psi_2\rangle = \frac{1}{\sqrt{2}}\left(|01\rangle + |10\rangle\right).$$

Для понимания специфики подобных состояний можно подробно рассмотреть процессы измерения, например, для состояний:

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle,$$
$$|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle).$$

Сначала рассмотрим процесс измерения двухкубитного состояния $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$. Допустим, что сначала мы проводим измерение первого кубита. В ходе проведения измерения первого кубита мы получим состояние $|0\rangle$, в том случае если мы попадем в группы $|00\rangle$ и $|01\rangle$. Вероятность такого события будет составлять $|\alpha_{00}|^2 + |\alpha_{01}|^2$. При этом первый кубит перешел в состояние $|0\rangle$, а состояния $|10\rangle$ и $|11\rangle$ становятся нереализуемы.

После получения этого результата двухкубитное состояние выглядит так:

$$|\psi\rangle = \frac{\alpha_{00}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|00\rangle + \frac{\alpha_{01}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|01\rangle,$$

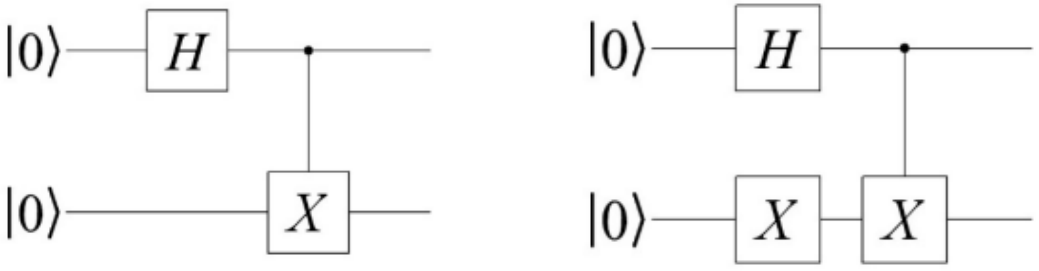
$$|\psi_1\rangle = |0\rangle; \quad |\psi\rangle = \frac{\alpha_{00}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|0\rangle + \frac{\alpha_{01}}{\sqrt{|\alpha_{00}|^2 + |\alpha_{01}|^2}}|1\rangle.$$

Теперь рассмотрим процесс измерения двухкубитного состояния $|\psi\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$.

В данном случае при измерении реализуемы только два состояния: $|01\rangle$ и $|10\rangle$. Если мы проведем измерение первого кубита, то реализуемой остается только одна группа: $|01\rangle$ или $|10\rangle$. Отсюда следует, что состояние второго кубита тоже определится, хотя его мы пока не провели его измерение. Например, если при измерении первого кубита мы получим состояние $|\psi_1\rangle = |0\rangle$ тогда реализуемой остается только группа $|01\rangle$, т.е. второй кубит переходит в определенной состояние – $|\psi_2\rangle = |1\rangle$. Если же при измерении первого кубита мы получим состояние $|\psi_1\rangle = |1\rangle$ тогда реализуемой остается только группа $|10\rangle$, то есть второй кубит переходит в определенной состояние – $|\psi_2\rangle = |0\rangle$. Также можно описать процессы измерения других перепутанных состояний.

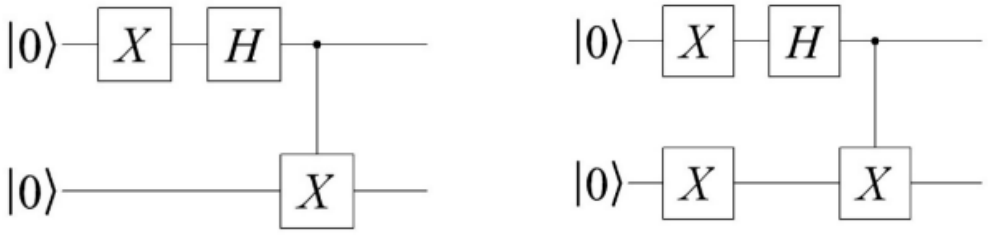
Аналогично обстоит дело и с более сложными системами: трехкубиными, четырехкубитными и так далее. Измерение одного кубита всегда выводит его из запутанности с остальными кубитами и приводит в одно из двух чистых базисных состояний. Квантовое состояние оставшейся части системы кубитов скачкообразно изменяется строго определенным образом. Если же измеряемый кубит не запутан с прочими кубитами системы, то при его измерении с остальными кубитами ничего не происходит.

На рисунке 1.12 представлены способы реализации состояний Белла системы двух кубитов.



$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$

а *б*



$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$ $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle)$

в *г*

Рис. 1.12. Примеры реализации состояний Белла

1.3.1.Сверхплотное кодирование

Приведем пример применения перепутанности квантовых состояний. Допустим, Алиса хочет передать Бобу одну из цифр от 0 до 3. Для организации передачи информации между Бобом и Алисой каждому из них пересылается один из двух кубитов приготовленных в запутанном состоянии, например, $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ Пусть Алиса получает первый кубит, а Боб второй. При этом Алиса может осуществлять преобразование на своем кубите, а Боб на своем[2].

Алгоритм обмена информации будет следующим: 1. Алиса получает извне два классических бита, которые кодируют цифры от 0 до 3. В зависимости от значения числа Алиса совершает одно из преобразований I, X, Y или Z: Для цифры 0:

$$|\psi_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix};$$

$$0 \Rightarrow (I \otimes I)|\psi_0\rangle = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Таким образом, получаем: $0 \Rightarrow (I \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |11\rangle)$.

Для цифры 1:

$$0 \Rightarrow (X \otimes I)|\psi_0\rangle = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} =$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

Таким образом, получаем: $1 \Rightarrow (X \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle)$.

Аналогично можно получить преобразования для цифр 2 и 3:

$$2 \Rightarrow (Y \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(-|10\rangle + |01\rangle);$$

$$3 \Rightarrow (Z \otimes I)|\psi_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle).$$

2. Далее Алиса передает свой кубит Бобу.

3. Боб применяет ***гейт*** $CNOT$ к двум запутанным кубитам:

$$0 \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |10\rangle);$$

$$1 \Rightarrow \frac{1}{\sqrt{2}}(|10\rangle + |01\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(|11\rangle + |01\rangle);$$

$$2 \Rightarrow \frac{1}{\sqrt{2}}(-|10\rangle + |01\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(-|11\rangle + |01\rangle);$$

$$3 \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \Rightarrow CNOT \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle - |10\rangle).$$

4. Далее Боб производит измерение второго кубита. В результате измерений он получит состояние $|0\rangle$ для цифр 0 и 3, состояние $|1\rangle$ для цифр 1 и 2:

$$0 \Rightarrow \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \\ \textit{Второй кубит } |0\rangle \end{cases};$$

$$1 \Rightarrow \frac{1}{\sqrt{2}}(|11\rangle + |01\rangle) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(|1\rangle + |0\rangle) \\ \textit{Второй кубит } |1\rangle \end{cases};$$

$$2 \Rightarrow \frac{1}{\sqrt{2}}\left(-|11\rangle + |01\rangle\right) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(-|1\rangle + |0\rangle) \\ \textit{Второй кубит } |1\rangle \end{cases};$$

$$3 \Rightarrow \frac{1}{\sqrt{2}}\left(|00\rangle - |10\rangle\right) \Rightarrow \begin{cases} \textit{Первый кубит } \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \\ \textit{Второй кубит } |0\rangle \end{cases}.$$

Результат измерения вотрого кубита:

$$|0\rangle \Rightarrow \begin{cases} \textit{Либо } 0 \\ \textit{Либо } 3 \end{cases} \qquad |1\rangle \Rightarrow \begin{cases} \textit{Либо } 1 \\ \textit{Либо } 2 \end{cases}$$

5. Далее Боб применяет ***преобразование Адамара*** к первому кубиту и измеряет его:

$$|0\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) + \frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right)\right) = |0\rangle;$$

$$|1\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle - |1\rangle\right) + \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\right) = |0\rangle;$$

$$|2\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(-|0\rangle + |1\rangle\right) + \frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right)\right) = |1\rangle;$$

$$|3\rangle \Rightarrow \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) + \frac{1}{\sqrt{2}}\left(-|0\rangle + |1\rangle\right)\right) = |1\rangle.$$

6. Таким образом, выполнив преобразования и измерив два ***кубита***, Боб понимает, какую цифру передавала Алиса:

7.

$$\left.\begin{matrix} \textit{Первый кубит} \to |0\rangle \\ \textit{Второй кубит} \to |0\rangle \end{matrix}\right\} \to \textit{цифра } 0;$$

$$\left.\begin{matrix} \textit{Первый кубит} \to |0\rangle \\ \textit{Второй кубит} \to |1\rangle \end{matrix}\right\} \to \textit{цифра } 1;$$

$$\left.\begin{matrix} \textit{Первый кубит} \to |1\rangle \\ \textit{Второй кубит} \to |1\rangle \end{matrix}\right\} \to \textit{цифра } 2;$$

$$\left.\begin{matrix} \textit{Первый кубит} \to |1\rangle \\ \textit{Второй кубит} \to |0\rangle \end{matrix}\right\} \to \textit{цифра } 3.$$

Исходя из представленного выше описания сврехплотного кодирования, можно построить квантовую схему (рис. 3.2).

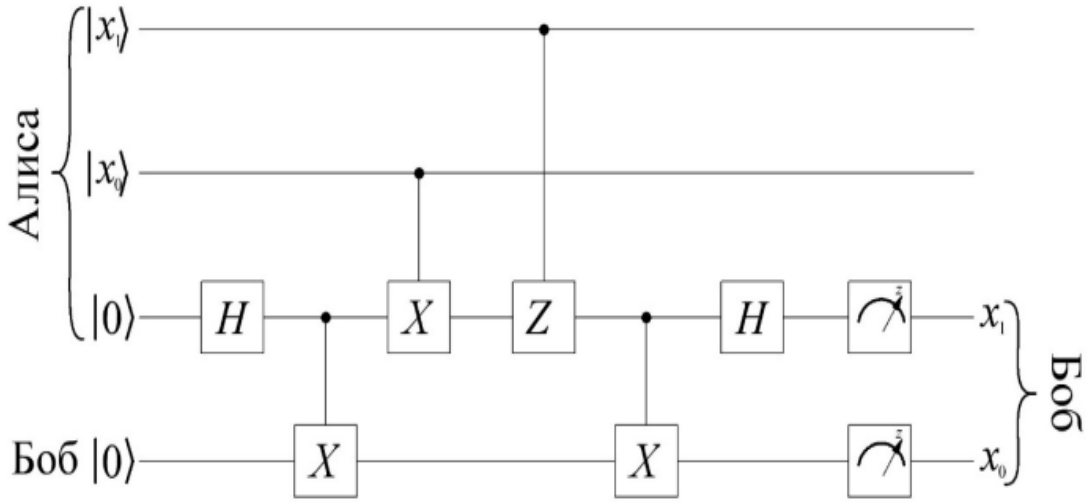


Рис. 1.13. Квантовая схема сверхплотного кодирования

На схеме, представленной на рисунке 1.13, передаваемая цифра кодируется в двоичном коде 1 x_1x_0. При этом один из запутанных ***кубитов*** принадлежит Алисе и в зависимости от передаваемой цифры Алиса применяет к нему соответствующее преобразование. После преобразования своего кубита Боб применяет операцию распутывания кубитов посредствам операции контролируемого NOT и гейта Адамара H. После этого производится операция измерения двух кубитов.

1.3.3. Квантовая телепортация

Задача квантовой телепортации заключается в переносе неизвестного квантового состояния с одной системы на другую с использованием квантового канала связи. Так как квантовое состояние не может быть скопировано (теорема о неклонируемости), то в процессе передачи исходное состояние кубита будет утеряно. При осуществлении квантовой телепортации исходное состояние кубита будет утеряно, но при этом оно будет воссоздано у получателя [34, 35, 40].

Рассмотрим пример квантовой телепортации. Допустим, Алиса хочет передать Бобу кубит в состоянии суперпозиции $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$. Кубит в данном состоянии находится у Алисы. Для организации передачи информации между Бобом и Алисой каждому из них пересылается один из двух кубитов приготовленных в запутанном состоянии, например, $|\psi_{23}\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$.

На рисунке 1.14 показано распределение кубитов между Бобом и Алисой.

$|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle \rightarrow$ *Первый бит – Алиса*

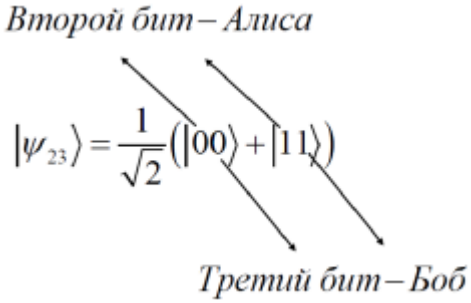


Рис. 1.14 Кубиты Алисы и Боба для квантовой телепортации

Таким образом, можно говорить, что мы имеем трехкубитовую систему:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|100\rangle + \beta|111\rangle)$$

Далее Алиса применяет операцию контролируемой инверсии, причем передаваемый кубит 1 $|\psi_1\rangle = \alpha|0\rangle + \beta|1\rangle$ будет являться контролирующим. Тогда трехбитовая система будет иметь вид:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|011\rangle + \beta|110\rangle + \beta|101\rangle).$$

После операции *CNOT* Алиса выполняет преобразование Адамара с передаваемым кубитом. Для дальнейшего описания системы вспомним, что $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ и $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Следовательно, состояние системы после преобразования Адамара можно представить в виде:

$$\alpha|000\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\alpha|000\rangle + \alpha|100\rangle),$$

$$\alpha|011\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\alpha|011\rangle + \alpha|111\rangle),$$

$$\beta|110\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\beta|010\rangle - \beta|110\rangle),$$

$$\beta|110\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(\beta|010\rangle - \beta|110\rangle),$$

$$|\psi\rangle = \frac{1}{2}(\alpha|000\rangle + \alpha|100\rangle + \alpha|011\rangle + \alpha|111\rangle + \beta|010\rangle - \beta|110\rangle + \beta|001\rangle - \beta|101\rangle).$$

Напомним, что в данной системе два кубита принадлежат Алисе и один кубит – Бобу. Если выделить в волновой функции системы кубиты Алисы, то ее можно записать следующим образом:

$$|\psi\rangle = \frac{1}{2}\left(|00\rangle(\alpha|0\rangle + \beta|1\rangle) + |11\rangle(\alpha|1\rangle + \beta|0\rangle) + |10\rangle(\alpha|0\rangle - \beta|1\rangle) + |11\rangle(\alpha|1\rangle - \beta|0\rangle)\right).$$

Из представленной выше функции можно заметить, что кубит, который принадлежит Бобу, может быть преобразован в исходный (передаваемый) кубит. Причем необходимое преобразование зависит от значений двух кубитов Алисы. Распишем необходимые преобразования для получения Бобом исходного кубита:

1. Если кубиты Алисы при измерении принимают значения $|00\rangle$, то в данном случае Бобу не нужно выполнять никаких преобразований и его кубит будет иметь волновую функцию $\alpha|0\rangle + \beta|1\rangle$.

2. Если кубиты Алисы при измерении принимают значения $|01\rangle$, то в данном случае кубит, который принадлежит Бобу, находится в состоянии $\alpha|1\rangle + \beta|0\rangle$. Можно заметить, что для получения исходного кубита Бобу необходимо выполнить операцию инверсии:

$$\alpha|1\rangle + \beta|0\rangle \xrightarrow{X} \alpha|0\rangle + \beta|1\rangle.$$

3. Если кубиты Алисы при измерении принимают значения $|10\rangle$, то в данном случае кубит, который принадлежит Бобу, находится в состоянии $\alpha|0\rangle - \beta|1\rangle$. Можно заметить, что для получения исходного кубита Бобу необходимо применить гейт Z:

$$\alpha|0\rangle - \beta|1\rangle \xrightarrow{Z} \alpha|0\rangle + \beta|1\rangle.$$

4. Если кубиты Алисы при измерении принимают значения $|11\rangle$, то в данном случае кубит, который принадлежит Бобу, находится в состоянии $\alpha|1\rangle - \beta|0\rangle$. Можно заметить, что для получения исходного кубита Бобу необходимо применить гейты Z и X:

$$\alpha|1\rangle - \beta|0\rangle \xrightarrow{X} \alpha|0\rangle - \beta|1\rangle,$$
$$\alpha|0\rangle - \beta|1\rangle \xrightarrow{Z} \alpha|0\rangle + \beta|1\rangle.$$

Исходя из алгоритма телепортации, можно построить квантовые схемы ее реализующие (рис. 1.15).

В представленной на рисунке 1.15 схеме передаваемым кубитом является $|\psi_1\rangle$. При разработке схемы телепортации данный кубит должен находиться в состоянии суперпозиции. Получить подобное состояние можно с использованием гейтов поворота U или R.

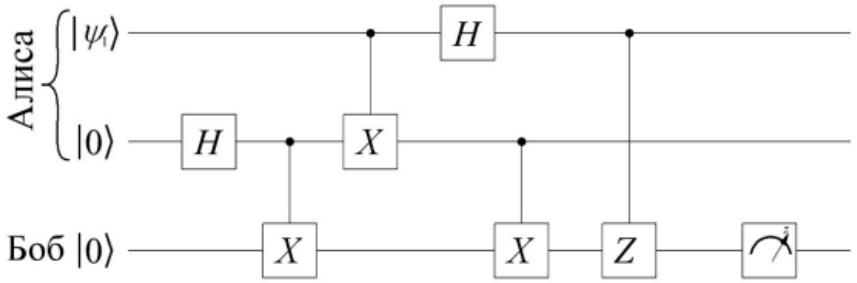


Рис. 1.15. Схема алгоритма квантовой телепортации

1.3.4. Алгоритм Дойча. Задача Дойча-Джозы

Принцип работы квантового компьютера и преимущества квантовых вычислений наиболее просто показать на примере простейшего квантового алгоритма – алгоритма Дойча [34, 35, 40]. Опишем задачу, которую решает данный алгоритм. Предположим, что у нас имеется «черный ящик», который вычисляет неизвестную нам функцию одной переменной – $f(x)$. Так как это функция одной переменной, то на вход можно подать 0 или 1 и на выходе также можно получить или 0, или 1. Функции одной переменной можно разделить на две группы: константные и сбалансированные. На выходе первых мы всегда будем получать постоянное значение 0 или 1 не зависимо от того, что подано на вход. Константных функций одной переменной являются функции вида: $f(x)=0$ и $f(x)=1$. Сбалансированными функциями одной переменной являются функции тождественного равенства и инверсии: $f(x)=x$ и $f(x)=\overline{x}$.

Задача Дойча состоит в том, чтобы определить к какой из двух групп (константная или сбалансированная) относится функция, реализуемая «черным ящиком». Решение подобной задачи с использованием классического компьютера сводится к тому, что необходимо на вход схемы подать сначала 0, а потом 1. То есть на классическом компьютере нам необходимо будет два раза вызвать функцию и измерить выходную реакцию. После двух данных операций мы однозначно можем идентифицировать не только к какой группе относится функция, но и вид данной функции. Если же мы на классическом компьютере вызовем функцию один раз, то не сможем определить даже группу, к которой она относится. Однако использование квантового компьютера позволит определить группу за один вызов функции. В квантовой системе в качестве функции в «черном ящике» должен быть реализован унитарный оператор U_f.

Данный оператор называется квантовым оракулом. Квантовый оракул – это многокубитный гейт, который соответствует бинарной функции, содержит в себе информацию о функции $f(x)$ и позволяет одновременно вызвать ее на всех возможных аргументах. Квантовый оракул определяется как преобразование $U_f|x\rangle|y\rangle=|x\rangle|y\oplus f(x)\rangle$, где, в общем случае, множество кубитов $|x\rangle$ несут в себе информацию об аргументах функции (остаются неизменными), а кубит $|y\rangle$ – результат. При этом размерность квантового оракула будет составлять $n+1$при размерности функции $f(x)-n$.

Определив квантовый оракул как преобразование $U_f|x\rangle|y\rangle=|x\rangle|y\oplus f(x)\rangle$ можно показать его воздействие на состояни $\frac{1}{\sqrt{2}}|x\rangle(|0\rangle-|1\rangle)$:

$$U_f\frac{1}{\sqrt{2}}|x\rangle(|0\rangle-|1\rangle)=\frac{1}{\sqrt{2}}|x\rangle\left(|0\oplus f(x)\rangle-|1\oplus f(x)\rangle\right)=\frac{1}{\sqrt{2}}(-1)^{f(x)}|x\rangle\left(|0\rangle-|1\rangle\right).$$

Давайте получим матрицы и схемы оракулов для всех бинарных функций одной переменной:

$$f(x)=0,\ f(x)=1,\ f(x)=x \text{ и } f(x)=\bar{x}.$$

1. Функция $f(x)=0$. В соответствии с воздействием $|x\rangle|y\oplus f(x)\rangle$ рассмотрим в какие состояния должны перейти все возможные базисные состояния регистра:

$$|00\rangle=|0\rangle|0\rangle\xrightarrow{U_f}|0\rangle|0\oplus f(0)\rangle=|0\rangle|0\oplus 0\rangle=|00\rangle,$$
$$|01\rangle=|0\rangle|1\rangle\xrightarrow{U_f}|0\rangle|1\oplus f(0)\rangle=|0\rangle|1\oplus 0\rangle=|01\rangle,$$
$$|10\rangle=|1\rangle|0\rangle\xrightarrow{U_f}|1\rangle|0\oplus f(1)\rangle=|1\rangle|0\oplus 0\rangle=|10\rangle,$$
$$|11\rangle=|1\rangle|1\rangle\xrightarrow{U_f}|1\rangle|1\oplus f(0)\rangle=|1\rangle|1\oplus 0\rangle=|11\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f=\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}.$$

Исходя из полученной матрицы оракула, можно представить его квантовую схему (рис. 1.16).

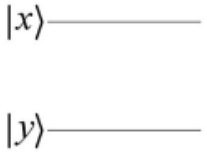


Рис. 1.16. Квантовая схема оракула функции $f(x)=0$

2. Функция $f(x)=1$. Рассмотрим, в какие состояния должны перейти все возможные базисные состояния регистра

$$|00\rangle=|0\rangle|0\rangle\xrightarrow{U_f}|0\rangle|0\oplus f(0)\rangle=|0\rangle|0\oplus 1\rangle=|01\rangle,$$
$$|00\rangle=|0\rangle|1\rangle\xrightarrow{U_f}|0\rangle|1\oplus f(0)\rangle=|0\rangle|1\oplus 1\rangle=|00\rangle,$$
$$|10\rangle=|1\rangle|0\rangle\xrightarrow{U_f}|1\rangle|0\oplus f(1)\rangle=|1\rangle|0\oplus 1\rangle=|11\rangle,$$
$$|11\rangle=|1\rangle|1\rangle\xrightarrow{U_f}|1\rangle|1\oplus f(0)\rangle=|1\rangle|1\oplus 1\rangle=|10\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f=\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}.$$

Как видно из полученной матрицы оракула, – в данном случае схема будет представлять собой инверсию $|y\rangle$ (рис. 1.17).

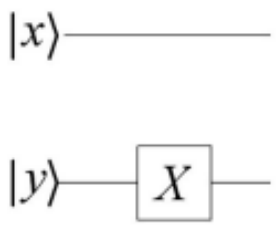


Рис. 1.17. Квантовая схема оракула функции $f(x)=1$

3. Функция $f(x)=x$. Рассмотрим, в какие состояния должны перейти все возможные базисные состояния регистра:

$$|00\rangle=|0\rangle|0\rangle\xrightarrow{U_f}|0\rangle|0\oplus f(0)\rangle=|0\rangle|0\oplus 0\rangle=|00\rangle,$$
$$|01\rangle=|0\rangle|1\rangle\xrightarrow{U_f}|0\rangle|1\oplus f(0)\rangle=|0\rangle|1\oplus 0\rangle=|01\rangle,$$
$$|10\rangle=|1\rangle|0\rangle\xrightarrow{U_f}|1\rangle|0\oplus f(1)\rangle=|1\rangle|0\oplus 1\rangle=|11\rangle,$$
$$|11\rangle=|1\rangle|1\rangle\xrightarrow{U_f}|1\rangle|1\oplus f(0)\rangle=|1\rangle|1\oplus 1\rangle=|10\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f=\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&0&1\\0&0&1&0\end{pmatrix}.$$

В данном случае квантовый оракул представляет собой контролируемый гейт X, причем кубит $|x\rangle$ является контролирующим (рис. 1.18).

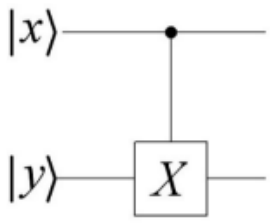


Рис. 1.18. Квантовая схема оракула функции $f(x)=x$

4. Функция $f(x)=\bar{x}$. Рассмотрим, в какие состояния должны перейти все возможные базисные состояния регистра:

$$|00\rangle=|0\rangle|0\rangle\xrightarrow{U_f}|0\rangle|0\oplus f(0)\rangle=|0\rangle|0\oplus 1\rangle=|01\rangle,$$
$$|01\rangle=|0\rangle|1\rangle\xrightarrow{U_f}|0\rangle|1\oplus f(0)\rangle=|0\rangle|1\oplus 1\rangle=|00\rangle,$$
$$|10\rangle=|1\rangle|0\rangle\xrightarrow{U_f}|1\rangle|0\oplus f(1)\rangle=|1\rangle|0\oplus 0\rangle=|10\rangle,$$
$$|11\rangle=|1\rangle|1\rangle\xrightarrow{U_f}|1\rangle|1\oplus f(0)\rangle=|1\rangle|1\oplus 0\rangle=|11\rangle.$$

Таким образом, матрицу оракула U_f можно представить в виде:

$$U_f=\begin{pmatrix}0&1&0&0\\1&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}$$

Подобный оператор можно реализовать различными схемами (рис. 1.19).

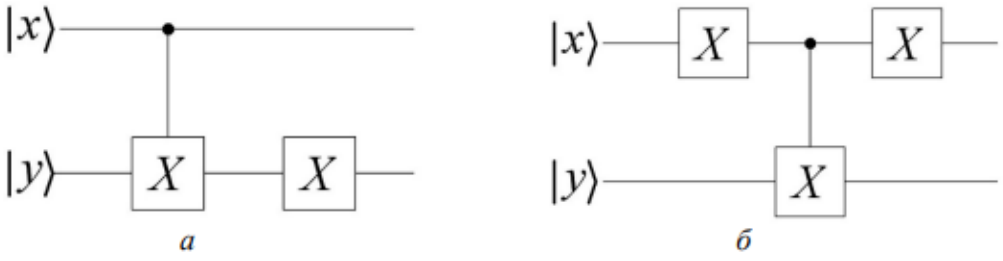


Рис. 1.19. Квантовая схема оракула функции $f(x)=\bar{x}$

Итак, описав оракулы, мы можем вернуться к разработке алгоритма, реализующего задачу Дойча. Схема данного алгоритма представлена на рисунке 1.20.

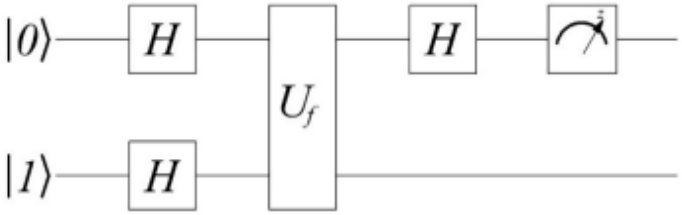


Рис. 1.20. Схема алгоритма Дойча

Приведем описание, как данная схема работает. Применим преобразование Адамара к каждому кубиту двукубитовой системы:

$$|0\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle);$$

$$|1\rangle \xrightarrow{H} \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle);$$

$$|xy\rangle = |01\rangle \xrightarrow{H} \frac{1}{2}(|0\rangle + |1\rangle)(|0\rangle - |1\rangle) = \frac{1}{2}|0\rangle(|0\rangle - |1\rangle) + \frac{1}{2}|1\rangle(|0\rangle - |1\rangle).$$

Далее на систему действует оператор U_f.

$$U_f \frac{1}{\sqrt{2}}|x\rangle(|0\rangle - |1\rangle) = \frac{1}{\sqrt{2}}(-1)^{f(x)}|x\rangle(|0\rangle - |1\rangle);$$

$$\frac{1}{2}|0\rangle(|0\rangle - |1\rangle) + \frac{1}{2}|1\rangle(|0\rangle - |1\rangle) \xrightarrow{U_f} \frac{1}{2}\left((-1)^{f(0)}|0\rangle + (-1)^{f(1)}|1\rangle\right)(|0\rangle - |1\rangle).$$

Далее по алгоритму применяется преобразование Адамара к первому кубиту, и после этого производится его измерение. Рассмотрим оба случая, т.е. когда функция является константной или сбалансированной.

- В случае если функция является константной, то $f(0) = f(1)$. В данном случае после воздействия оператора U_f первый кубит будет находиться либо в состоянии $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$, либо в состоянии $-\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. При этом воздействие оператора Адамара приведет его в состояния $|0\rangle$ или $-|0\rangle$ соответственно. В любом случае измерение покажет, что кубит находится в нулевом состоянии.
- В случае если функция является сбалансированной, то $f(0) \neq f(1)$. После воздействия оператора U_f первый кубит будет находиться либо в состоянии $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, либо в состоянии $-\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. Последующее воздействие оператора Адамара приведет его в состояния $|1\rangle$ или $-|1\rangle$, соответственно, и измерение покажет, что кубит находится в единичном состоянии.

Далее рассмотрим с вами решения задачи Дойча-Джозы, которая является обобщенной задачей Дойча. Данная задача формулируется следующим образом: пусть имеется бинарная функция: $f : \{0,1\}^n \to \{0,1\}$, и известно, что данная функция может быть константной или сбалансированной. Функция с n входными переменными будет сбалансированной, если она принимает значение 0 на 2^{n-1} входных наборах и значение 1 на 2^{n-1} входных наборах. Задача Дойча-Джозы,

как и задача Дойча, заключается в том, что необходимо определить, является ли функция в «черном ящике» сбалансированной или константной. По аналогии с предыдущей задачей, схема алгоритма Дойча-Джозы будет иметь вид, представленный на рисунке 1.21.

Выполним описание данного алгоритма. Для этого рассмотрим действие оператора Адамара на произвольный квантовый регистр $|x\rangle^n$:

$$|x\rangle^n \to |x_1 x_2 ... x_n\rangle \xrightarrow{H} H|x_1\rangle \otimes H|x_2\rangle \otimes ... \otimes H|x_n\rangle =$$

$$= \frac{1}{\sqrt{2^n}} \left(\sum_{y_1 \in \{0,1\}} (-1)^{x_1 y_1} |y_1\rangle \otimes \sum_{y_2 \in \{0,1\}} (-1)^{x_2 y_2} |y_2\rangle \otimes ... \otimes \sum_{y_n \in \{0,1\}} (-1)^{x_n y_n} |y_n\rangle \right) =$$

$$= \frac{1}{\sqrt{2^n}} \sum_{y \in B_n} (-1)^{x_1 y_1 \otimes x_2 y_2 \otimes ... \otimes x_n y_n} |y\rangle^n$$

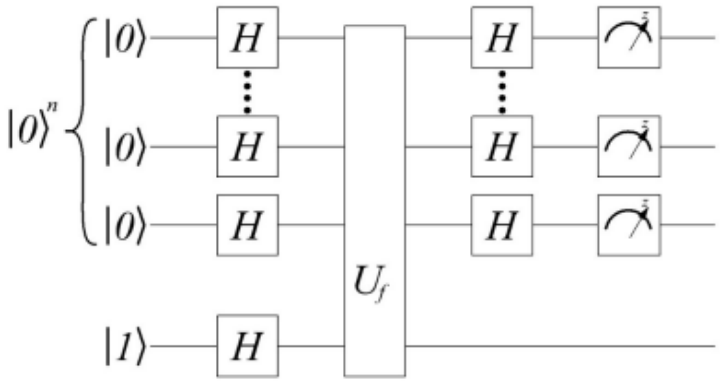


Рис. 1.21. Схема алгоритма Дойча-Джозы

В начальный момент времени система (рис. 3.10) находится в состоянии $|0\rangle^n |1\rangle$. Если применить операцию H_n к состоянию $|0\rangle^n$, то мы получим следующий результат $H_n |0\rangle^n = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$, следовательно, для состояния $|0\rangle^n |1\rangle$ получим:

$$|0\rangle^n |1\rangle \xrightarrow{H_{n+1}} = \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle).$$

Воздействие оракула U_f переведет систему в состояние:

$$\frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} |x\rangle (|0\rangle - |1\rangle) \xrightarrow{U_f} \frac{1}{\sqrt{2^{n+1}}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle (|0\rangle - |1\rangle).$$

Если функция $f(x)$ является константной и, учитывая, что $H_n |0\rangle^n = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle$, то первые n кубитов находятся в состоянии

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} (-1)^{f(x)} |x\rangle = (-1)^{f(x)} H_n |0\rangle^n .$$

Повторное применение оператора Адамара к состоянию $H_n|0\rangle^n$ приведет к тому, что при измерении (рис. 3.10) с вероятностью 100% будет получено состояние $|0\rangle^n$.

Если же функция $f(x)$ является сбалансированной, то $(-1)^{f(x)}$ – нельзя вынести из под суммы и состояние первых n кубитов можно представить как [1].

$$\frac{1}{\sqrt{2^n}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle=\frac{1}{\sqrt{2^n}}\left(\sum_{x:f(x)=0}|x\rangle-\sum_{x:f(x)=1}|x\rangle\right).$$

Отметим, что

$$H_n|x\rangle^n=\frac{1}{\sqrt{2^n}}\sum_{y=0}^{2^n-1}(-1)^{x_1y_1\otimes x_2y_2\otimes\ldots\otimes x_ny_n}|y\rangle=\frac{1}{\sqrt{2^n}}\left(|0\rangle^n+\sum_{y=1}^{2^n-1}(-1)^{x_1y_1\otimes x_2y_2\otimes\ldots\otimes x_ny_n}|y\rangle\right).$$

Если функция $f(x)$ сбалансирована, то число слагаемых в суммах выражения: $\frac{1}{\sqrt{2^n}}\left(\sum_{x:f(x)=0}|x\rangle-\sum_{x:f(x)=1}|x\rangle\right)$ одинаково, и при повторно применении оператора Адамара векторы $|0\rangle^n$ будут отсутствовать. Таким образом, в случае сбалансированной функции при измерении системы вероятность получения состояния $|0\rangle^n$ будет равно нулю.

1.3.5. Алгоритм Гровера

Алгоритм Гровера направлен на решение следующей задачи: в базе неупорядоченных данных, состоящей из n элементов, требуется найти элемент с заданными свойствами [34, 35, 40]. Для решения данной задачи на классическом компьютере в среднем потребуется перебрать $\frac{n}{2}$ элементов базы, а в наихудшем случае – $n-1$ элементов. Квантовый алгоритм Гровера позволит выполнить данную задачу всего за $\sqrt{n}$ операций [35]. Задачу поиска можно сформулировать следующим образом: пусть имеется бинарная функция: $f:\{0,1\}^n\to\{0,1\}$, и известно, что данная функция принимает значение 0 на всех входных наборах, кроме x_0, и задача состоит в том, чтобы найти x_0. Очевидно, что для реализации алгоритма поиска необходимо получит оракул данной функции:

$$U_f|x\rangle|y\rangle=|x\rangle|y\otimes f(x)\rangle,$$
$$|x\rangle-\textit{входной регистр},$$
$$|y\rangle-\textit{входной регистр}.$$

Вычисление функции $f(x)$ для всевозможных входных наборов $|x\rangle$ за счет применения оператора U_f, при начальном значении

выходного регистра равному 0, приведет к суперпозиции $\frac{1}{\sqrt{2^n}}\sum_x |x\rangle|f(x)\rangle$. Для входного набора x_0, при котором $f(x_0)=1$, состояние $|x_0\rangle|1\rangle$ будет иметь вероятность 2^{-n}. Суть алгоритма Гровера состоит в том, чтобы максимально повысить амплитуду состояния $|x_0\rangle|1\rangle$ и уменьшить амплитуды состояний $|x_0\rangle|0\rangle$.[2] Представим последовательность реализации алгоритма Гровера:

1. Подготавливаем входной регистр $|x\rangle^n$, который будет содержать суперпозицию всех возможных значений входных наборов.

2. Вычисляем функцию $f(x)$.

3. Изменяем знак коэффициентов для тех значений x при которых функция $f(x)$ равна 1.

4. Далее необходимо увеличить амплитуду коэффициентов тех значений x, при которых функция $f(x)$ равна 1. Увеличение амплитуд возможно за счет применения операции, которая носит название «инверсия относительно среднего». После применения данной операции амплитуды входных значений, при которых функция равна 1 вырастут, а амплитуды входных значений, при которых функция равно 0 – уменьшаться.

5. Далее пункты 2–4 повторяются $\frac{\pi}{4}\sqrt{2^n}$ раз. Изменение знака коэффициентов выполняется следующим образом. Возьмем значение выходного регистра, равное 1, и применим преобразование Адамара:

$$|0\rangle^n|1\rangle \xrightarrow{H_{n+1}} = \frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\big(|0\rangle-|1\rangle\big).$$

Затем к полученному состоянию суперпозиции применяется преобразование U_f и, как и в алгоритме Дойча, это приводит к следующему результату:

$$\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}|x\rangle\big(|0\rangle-|1\rangle\big)\xrightarrow{U}\frac{1}{\sqrt{2^{n+1}}}\sum_{x=0}^{2^n-1}(-1)^{f(x)}|x\rangle\big(|0\rangle-|1\rangle\big).$$

Отсюда можно заметить, что в полученном состоянии суперпозиции амплитуды значений, при которых функция равна 1, будут иметь отрицательное значение. Инверсия относительно среднего – это унитарное преобразование, которое воздействует на систему следующим образом:

$$\sum_i \alpha_i|x_i\rangle \to \sum_{x=0}^{2^n-1}(2A-\alpha_i)|x_i\rangle,$$

где A – среднее значение амплитуды. По сути амплитуда осуществляется путем переворачивания вероятностей относительно их среднего. Если число выше среднего, оно переворачивается и становится ниже среднего и наоборот. Преобразование инверсии относительно среднего описывается матрицей:

$$D=\begin{pmatrix} \frac{2}{N}-1 & \frac{2}{N} & \dots & \frac{2}{N} \\ \frac{2}{N} & \frac{2}{N}-1 & \dots & \frac{2}{N} \\ \dots & \dots & \dots & \dots \\ \frac{2}{N} & \frac{2}{N} & \dots & \frac{2}{N}-1 \end{pmatrix},$$

где $T=2^n$.

Пример матрицы преобразования «инверсия относительно среднего» для $N=4$, т.е. для $n=2$, будет выглядеть следующим образом:

$$D=\begin{pmatrix} -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{pmatrix}=\frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}.$$

Пример схемы, реализующей данное преобразование, представлен на рисунке 1.22 [34, 35].

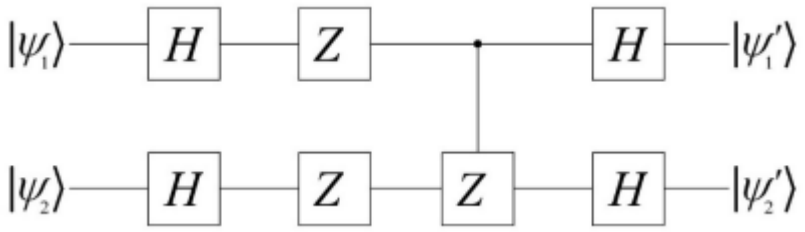


Рис. 1.22. Квантовая схема инверсии относительно среднего

Покажем, что данная схема реализует преобразование «инверсия относительно среднего:

$$H\otimes H=\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}=\frac{1}{2}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix},$$

$$Z \otimes Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

$$(H \otimes H)(Z \otimes Z) = \frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix},$$

$$CZ = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix},$$

$$(H \otimes H)(Z \otimes Z)CZ = \frac{1}{2}\begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix},$$

$$(H \otimes H)(Z \otimes Z)CZ(H \otimes H) = \frac{1}{4}\begin{pmatrix} -2 & 2 & 2 & 2 \\ 2 & -2 & 2 & 2 \\ 2 & 2 & -2 & 2 \\ 2 & 2 & 2 & -2 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} -1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}.$$

В общем виде пример схемы алгоритма Гровера можно представить, как показано на рисунке 1.23 [34].

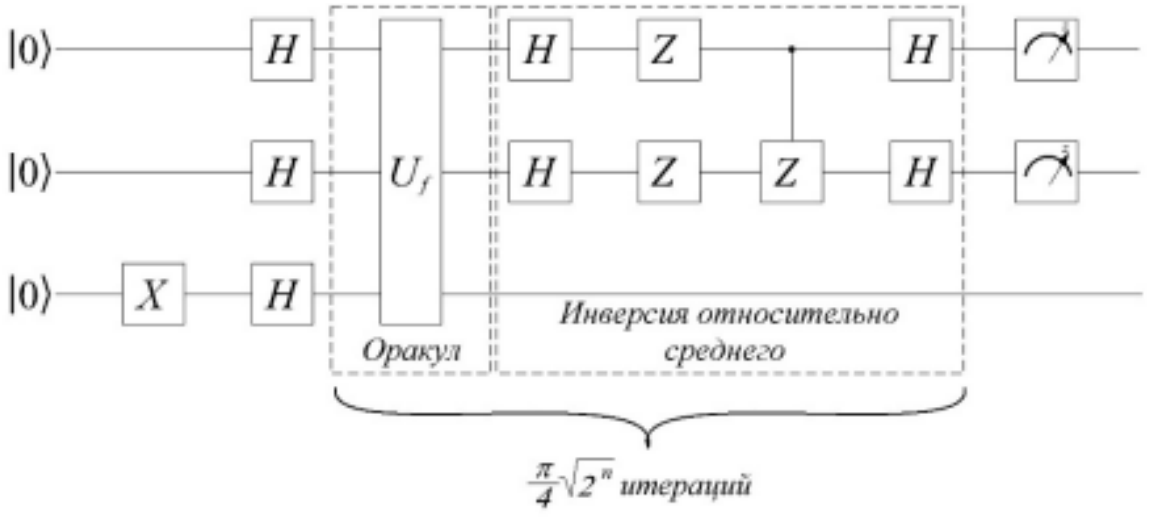


Рис. 1.23 Схема алгоритма Гровера

1.3.6. Квантовое преобразование Фурье

Квантовое преобразование Фурье (КПФ) – это аналог дискретного преобразования Фурье (ДПФ), которое применяется к вектору амплитуд квантовых состояний. Квантовое преобразование Фурье

широко применяется во многих квантовых алгоритмах, в частности, в алгоритме Шора. Квантовое преобразование Фурье осуществляет преобразование квантового состояния следующим образом:

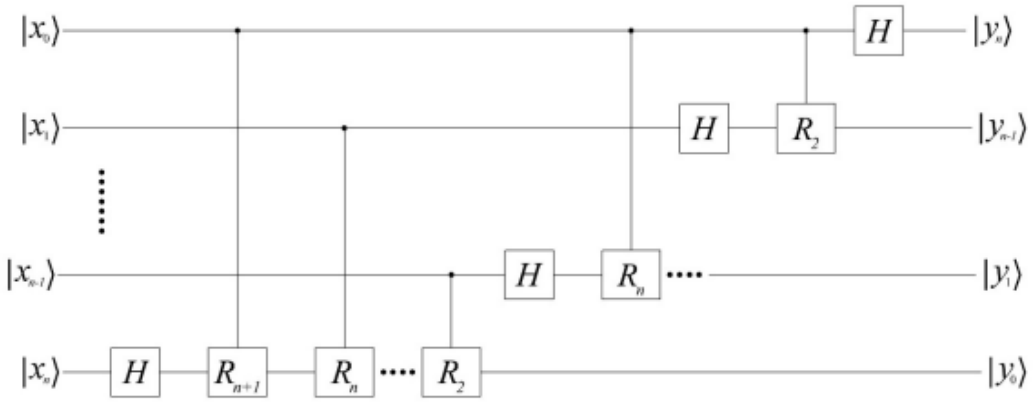
$$\sum_x f(x)|x\rangle = \sum_y F(y)|y\rangle .$$

При измерении состояния после преобразования Фурье с вероятностью $|F(y)|^2$ мы получим состояние $|y\rangle$. ДПФ осуществляет преобразование функции с периодам r в функцию, которая имеет нулевые значения на всех частотах не кратных $\frac{1}{r}$. Применив квантовое преобразование Фурье к функции $f(x)$ с периодом r, мы получим функцию $F(y)$, которая будет равна нулю, кроме значений кратных $\frac{N}{r}$. То есть проведя измерение, мы получим результат кратный $\frac{N}{r}$ [35].

Квантовое преобразование Фурье определяется следующим образом:

$$|x\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2^n}} \sum_{y=0}^{2^n-1} e^{\frac{2\pi i y x}{2^n}} |y\rangle .$$

В общем случае квантовая схема квантового преобразования Фурье представлена на рисунке



3.13.

Рис. 1.24. Схема квантового преобразования Фурье

Следует отметить, что схема меняет порядок кубитов. В схеме, представленной на рисунке 1.24, используются унитарные операторы поворота R_k, которые можно определить как:

$$R_k = \begin{pmatrix} 1 & 0 \\ 0 & e^{\frac{2\pi i}{2^k}} \end{pmatrix}.$$

В общем случае выражение для n-й линии можно выражением [34]:

$$|x_n\rangle \xrightarrow{QFT} \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i\left(\frac{x_n}{2}+\ldots+\frac{x}{2^n}+\frac{x_0}{2^{n+1}}\right)}|1\rangle\right).$$

1.3.7. Алгоритм Шора

Алгоритм Шора является одним из наиболее распространенных алгоритмов и направлен на решение задачи факторизации. Факторизация – это разложение чисел на множители. Факторизация широко применяется в современной теории чисел и криптоанализа. Следует отметить, что эта задача весьма сложная, так как трудоемкость классических алгоритмов растет экспоненциально с увеличением размера факторизуемых чисел. Так, при фактризации числа N путем простого перебора необходимо будет перебрать все варианты от 2 до $\sqrt{N}$, а это весьма сложно, если мы имеем дело с очень большими числами [34, 35, 40].

Пример использования факторизации – это алгоритмы шифрования с открытым ключом (RSA, El Gamal и т.д.). В данном случае ключ представляет собой пару больших чисел, а взлом шифра, который заключается в нахождении по открытому ключу приватного и наоборот, требует решения задачи факторизации. Алгоритм Шора позволяет выполнить решение задачи факторизации за полиномиальное время. Это осуществляется за счет использования свойства квантового параллелизма и сведения задачи к поиску периода функции. Допустим нам необходимо разложить на множители некоторое число N. Изначально выберем произвольное число $\alpha < N$ и рассматриваем функцию $f_a(x) = a^x \bmod N$.

Функция $f_a(x)$ является периодической с периодом r. Период r является порядком числа $a: a^r = 1 \bmod N$ и $\forall r_1 < r \to a^{r_1} \neq 1 \bmod N$. Если число N простое, то период r будет равно $N-1$. Этот случай весьма простой и легко реализуется проверкой на простоту классическими методами.

В общем случае $f_a(x+r) = f_a(x)$. Если период r известен, то разложение на множители числа N легко можно определить классическими методами. В частности, если период r является четным числом, то из соотношения $a^r - 1 = 0 \bmod N$ можно записать:

$$\left(\alpha^{\frac{r}{2}} - 1\right)\left(\alpha^{\frac{r}{2}} + 1\right) = 0 \bmod N.$$

Так как $\left(\alpha^{\frac{r}{2}}-1\right)\left(\alpha^{\frac{r}{2}}+1\right)$ делится на N, то оба сомножителя имеют общие с N делители. Эти делители можно определить классическим алгоритмом Евклида по поиску наибольшего общего делителя. Если же период r является нечетным или $\left(\alpha^{\frac{r}{2}}-1\right)\left(\alpha^{\frac{r}{2}}+1\right)$ вырождается в ноль, то следует выбрать другое число α. Для больших чисел N это случается редко.

Квантовый алгоритм Шора предназначен для быстрого поиска периода r. Для реализации алгоритма необходимо использовать квантовый компьютер с двумя квантовыми регистрами размера n. Причем размер должен быть таким, что $M = 2^n > N$, $M \approx N^2$. Алгоритм определения периода r будет следующим:

1. На первом этапе необходимо приготовить два входных регистра в состоянии $|0\rangle$.

2. Далее воздействием на каждый кубит входного регистра преобразованием Адамара:

$$|0\rangle^n |0\rangle^n \xrightarrow{H_n} = \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle^n .$$

3. Применим к обоим регистрам квантовую схему, реализующую функцию $f_a(x)$. Теперь регистры перейдут в состояние:

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |0\rangle^n \xrightarrow{f_a(x)} \frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f_a(x)\rangle .$$

Таким образом, в выходном регистре будет суперпозиция всех возможных значений функции $f_a(x)$.

4. Далее производится измерение выходного регистра. В результате измерений мы получи

$$\frac{1}{\sqrt{2^n}} \sum_{x=0}^{2^n-1} |x\rangle |f_a(x)\rangle \xrightarrow{\text{измерение}} \frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |k + jr\rangle |f_a(k)\rangle .$$

После измерения мы получим значение функции $f_a(k)$ при некотором случайном значении k, а выходной регистр перейдет в состояние $|f_a(k)\rangle$. Измеренное значение выходного регистра нас не интересует, но важно, что состояние входного регистра редуцируется до комбинации только тех состояний, которые совместимы с измеренным на выходе значением: $f_a(x) = f_a(k)$, $x = k$, $x = k + r$, $x = k + 2r$ и т.д.

5. Для извлечения периодичности в выходном регистре необходимо выполнить квантовое преобразование Фурье и измерение результата преобразования.

$$\frac{1}{\sqrt{M}}\sum_{i=0}^{M-1}\left|k+jr\right\rangle \xrightarrow{QFT} \frac{1}{\sqrt{MN}}\sum_{i=0}^{M-1}\sum_{y=0}^{2^n-1} e^{2\pi i\frac{ky}{N}} e^{2\pi i\frac{jry}{N}}\left|y\right\rangle = \frac{1}{\sqrt{MN}}\sum_{y=0}^{2^n-1} e^{2\pi i\frac{ky}{N}}\sum_{y=0}^{M-1} e^{2\pi i\frac{jry}{N}}\left|y\right\rangle .$$

При преобразовании Фурье период r будет преобразован в $\frac{M}{r}$ и измерение приведет к тому, что с высокой долей вероятности мы получим $\frac{jM}{r}$ для $j=1,2,...$ Далее, применив классический алгоритм разложения в непрерывную дробь (алгоритм Евклида), можно извлечь из полученного результата период r.

На рисунке 1.25 представлена структурная схема алгоритма Шора [1].

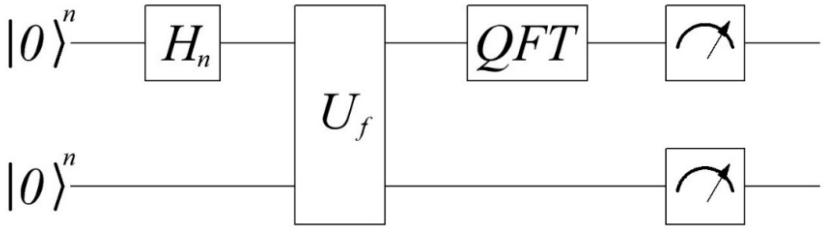


Рис. 1.25. Схема алгоритма Шора

Приведем классический пример реализации алгоритма Шора [1].

Пример 1.11: Допустим, нам необходимо выполнить факторизацию числа $N=pq=3\cdot 5=15$. Используя функцию Эйлера, можно определить функцию $f_a(x)$:

$$\Phi(N)=(q-1)(p-1)=(5-1)(3-1)=8\rightarrow a=7\rightarrow f_a(x)=7^x \bmod 15.$$

Функцию $f_a(x)$ можно записать в следующем виде:

$$f_a(x)=\left(7^8\right)^{x_3}\cdot\left(7^4\right)^{x_2}\cdot\left(7^2\right)^{x_1}\cdot\left(7^1\right)^{x_0} \bmod 15.$$

Поскольку $\left(7^8\right)^{x_3} \bmod 15=1$, $\left(7^4\right)^{x_2} \bmod 15=1$ и $\left(7^2\right)^{x_1} \bmod 15=(4)^{x_1} \bmod 15$, то можно переопределить функцию как:

$$f_a(x)=(4)^{x_1}\cdot(7)^{x_0} \bmod 15.$$

На рисунке 1.26 представлена квантовая схема, реализующая алгоритм Шора для данного примера.

На рисунке 1.26 блок под номером **1** реализует умножение регистра $|y\rangle$ на 7, а блок под номером 2 – умножение на 4. Далее выполняется квантовое преобразование Фурье (блок QFT) и измерение результата преобразования.

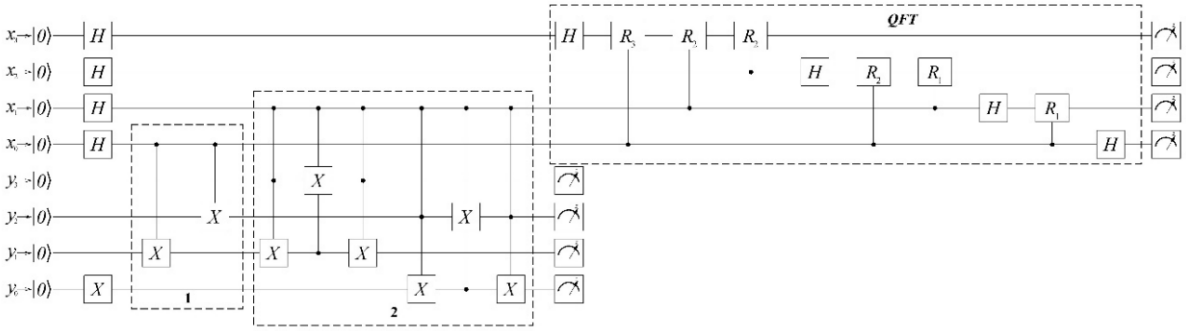


Рис. 1.26. Квантовая схема алгоритма Шора для $N=15$.

Выполним моделирование данной схемы в системе IBM Quantum Experience. Результаты серии измерений представлены на рисунке 1.27.

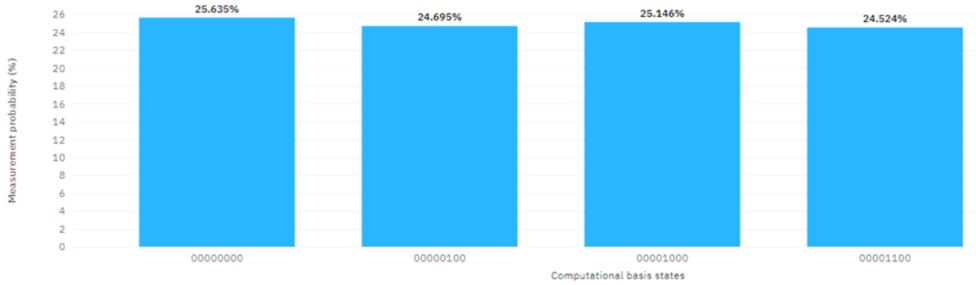


Рис. 1.27. Результаты симуляции алгоритма Шора для $N=15$.

Из рисунка 1.27 видно, что в результате измерений мы получили 0, 4, 8 и 12. Отбросив нулевой результат и вспомнив, что преобразование Фурье приведут к тому, что с высокой долей вероятности мы получим $\frac{jM}{r}$ для $j=1,2,...,$ можно вычислить период r:

$$\frac{jM}{r}\to\frac{j2^n}{r}\to\begin{cases} j=1\to\dfrac{2^4}{r}=4\to r=4 \\ j=2\to\dfrac{2\cdot 2^4}{r}=8\to r=4 \\ j=3\to\dfrac{3\cdot 2^4}{r}=4\to r=4 \end{cases}$$

Таким образом, мы определили период $r=4$. Зная период, мы можем определить числа p и q:

$$НОД\left(7^{\frac{r}{2}}+1,15\right)=НОД\left(50,15\right)=5,$$

$$НОД\left(7^{\frac{r}{2}}-1,15\right)=НОД\left(49,15\right)=3.$$

Глава 2. КВАНТОВЫЕ ТЕХНОЛОГИИ И ИНФРАСТРУКТУРА BIG DATA

2.1.Системы управления Большими данными

Чтобы отделить большие данные от обычных, нужно ответить на вопрос: «Big Data — это сколько?». Таблица в Excel на 500 000 строк — это большие данные? А если строк миллиард? Текстовый файл на тысячи слов, который весит 2 мегабайта, — это много? А распечатки графиков температуры всех метеостанций области — много или еще недостаточно? Тут многие скажут, что эти примеры представляют собой довольно внушительное количество информации. Действительно, с такой точки зрения, все перечисленное — большие данные. Но что вы скажете про таблицу в Excele на миллиард строк? Это тоже большие данные — и куда побольше тех!

На интуитивном уровне специалисты, далекие от Big Data, привыкли называть большими данными любой объем информации, который сложно удержать в голове и/или который занимает много места. И такое интуитивное определение, конечно же, неправильно [1].

Однозначно отделить формат больших данных от обычных помогут три критерия. Данные должны быть цифровыми. Книги в национальной библиотеке или стопки документов в архиве компании — это данные, и часто их много. Но термин Big Data означает только цифровые данные, которые хранятся на серверах. Данные должны поступать в объективно больших объемах и быстро накапливаться. Например, база заказов интернет-магазина по продаже колясок может быть большой: 10 миллионов заказов за 20 лет, но пополняется она со скоростью 100 заказов в сутки — это не большие данные. Фильм в высоком качестве может занимать десятки гигов, но со временем его размер не растет — это тоже не Big Data.

А вот записи показателей пары сенсоров в двигателе Боинга, поступающие в количестве несколько гигабайт в час и загружаемые на диагностический сервер производителя авиатехники — это уже big data.

Данные должны быть разнородными и слабо структурированными. Заказы в онлайн-магазине упорядочены, из них легко извлечь дополнительные статистические параметры, например, средний чек или самые популярные товары. Поэтому эти данные не относят к Big Data. Показания датчиков температуры с корпуса самолета, записанные за последние 6 месяцев, — информация, в

которой есть польза, но не очень понятно, как ее извлечь. Можно, конечно, рассчитать средние значения температуры за бортом самолета за полгода, но какой в этом смысл? А если погрузиться в анализ этих данных глубоко — можно вытащить много неочевидной информации. Например, о длительности перелетов, скорости набора высоты, климатических условиях за бортом и так далее. Информация интересная и полезная, но трудноизвлекаемая, значит, это большие данные [2].

Этот критерий не всегда обязательный. Иногда большие объемы структурированных данных, которые постоянно пополняются, относят к формату Big Data, особенно если их используют для машинного обучения или выявления неочевидных закономерностей. То есть если к структурированным данным применяют методы анализа Big Data, можно сказать, что это они и есть. Итак, большие данные — это трудноанализируемая цифровая информация, накапливаемая со временем и поступающая к вам.

Когда в любом IT-проекте начинают работать с данными, в первую очередь анализируют наиболее очевидные, значимые и понятные показатели. Так, если речь идет об онлайн-торговле, сначала смотрят на средние чеки заказов, топ продаж и объемы складских запасов. Когда речь идет о самолетах — смотрят скорость, высоту, расход топлива. Сбор и анализ очевидных метрик позволяет вносить в систему простые и понятные корректировки. Такие улучшения практически сразу дают ощутимый результат. Это называется «сбор фруктов с нижних веток дерева» [3].

По мере эволюции системы инженеры прорабатывают все видимые узкие места в проекте. После этого начинается стагнация продукта: для поиска новых путей развития нужно лезть выше, чтобы собрать плоды с более высоких веток. Инженеры и аналитики начинают собирать и анализировать косвенные данные, напрямую не связанные с основными метриками проектов. Например, в онлайн-торговле можно собирать со страниц магазина данные о перемещении курсора (или пальца) по экрану. Или собирать данные с большого числа сенсоров самолета, например: число оборотов двигателя, состав топливно-воздушной смеси, забортную температуру и температуру выхлопа. Или анализировать слова в комментариях клиентов в соцсетях для оценки их лояльности.

Это означает, что технологии Big Data чаще всего нужны тогда, когда требуется более глубокий анализ процессов. Такие данные напрямую не связаны с основными метриками IT-системы и бизнеса, но при правильном анализе могут рассказать много интересного о возможных точках оптимизации в проекте. Работа с такими данными — как поиск нефти. Нужно пробовать разные места, применять различные стратегии поиска и извлечения скрытых ресурсов, спрятанных в данных. Далеко не все попытки будут успешны, но в итоге находки могут принести массу выгоды. Большие данные в основном помогают решать четыре задачи:

Анализировать текущее положение дел и оптимизировать бизнес-процессы. С помощью больших данных можно понять, какие товары предпочитают покупатели, оптимально ли работают станки на производстве, нет ли проблем с поставками товаров. Обычно для этого ищут закономерности в данных, строят графики и диаграммы, формируют отчеты.

Например, с помощью больших данных компания Intel обнаружила, что делает много лишних тестов при производстве процессоров. Они проанализировали данные, отказались от лишних тестов и сэкономили около 30 миллиардов долларов.

Делать прогнозы. Данные о прошлом помогают сделать выводы о будущем. Например, примерно прикинуть продажи в новом году или предсказать поломку оборудования до того, как оно действительно сломается. Чем больше данных, тем точнее предсказания. Например, логистическая компания ПЭК запустила Центр управления перевозками с использованием big data. В итоге они стали прогнозировать загрузку складов — предсказывать, когда склады будут заполнены, а когда пусты. Это помогло планировать маршруты транспорта и избегать простоев [4].

Строить модели. На основе больших данных можно собрать компьютерную модель магазина, оборудования или нефтяной скважины. Потом с этой моделью можно экспериментировать: что-то в ней изменять, отслеживать разные показатели, ускорять или замедлять разные процессы для их анализа. Например, «Газпром нефть» смоделировала ситуацию аварийного отключения электричества, чтобы понять, почему возникает сбой автоматического перезапуска оборудования. Модель помогла обнаружить неожиданные причинно-следственные связи и устранить проблемы.

Автоматизировать рутину. На больших данных учатся автоматические программы, которые умеют выполнять определенные задачи, например, сортировать документы или общаться в чатах. Это могут быть как примитивные алгоритмы, так и искусственный интеллект: голосовые помощники или нейросети. Так, компания Stafory разработала робота-рекрутера Веру. Этот робот выполняет простую рекрутерскую работу: распознает голос, сортирует резюме, задает простые вопросы и принимает ответы. В итоге рекрутерам-людям остаются только более сложные и творческие задачи — реальные собеседования и окончательный отбор кандидатов [5].

Мы разобрались, что такое большие данные и какую пользу они могут принести. Теперь посмотрим, как в общих чертах работают системы анализа больших данных и какие инструменты нужны для их работы. Упрощенно работа с Big Data происходит по следующей схеме: информацию собирают из разных источников → данные помещают на хранение в базы и хранилища → данные обрабатывают и анализируют → обработанные данные выводят с помощью средств визуализации или используют для машинного обучения. Для технологий, которые работают с большими данными, базовым принципом считают горизонтальную масштабируемость, то есть возможность обрабатывать данные сразу на множестве узлов (серверов, компьютеров). Если обрабатывать такой массив информации на одном узле, это займет слишком много времени.

Итак, к основным технологиям для работы с большими данными относят:

MapReduce. Это модель распределенных вычислений, разработанная Google. Ее суть в том, что обработка больших объемов информации происходит на большом количестве серверов (узлов), которые образуют кластер. На каждом сервере производятся одинаковые элементарные задания по обработке, потом все результаты обработки сводят вместе. Если копнуть чуть глубже, мы увидим, что в основе технологии лежат две процедуры функционального программирования. Первая — map, она применяет нужную функцию к каждому элементу данных. Вторая — reduce, она объединяет результаты работы. Такой подход позволяет быстрее обрабатывать большие данные.

NoSQL — термин расшифровывается как Not Only SQL, «не только SQL». Это подход к реализации систем управления базами

данных. В общих чертах — особенность в том, что для хранения информации в базах данных NoSQL не требуется заранее заданная схема данных. Это значит, что любые данные можно легко помещать в хранилище и быстро извлекать оттуда. Когда у вас большое количество разнородных данных, именно это и нужно [6].

Hadoop — инструмент для разработки решений, которые работают по модели MapReduce. По сути, это конструктор, из которого можно создавать хранилища данных под потребности бизнеса. Технология лежит в основе многих облачных решений для обработки больших данных. Например, сервис для анализа big data от Mail.ru Cloud Solutions построен на базе Hadoop, Spark и ClickHouse.

R. Язык программирования для работы с графикой и статистической обработки данных. Стандарт для создания аналитических и статистических программ, без которых по определению невозможен анализ big data. Еще аналитики часто используют языки Python, Scala, Java.

McKinsey также включает в этот список технологии Business Intelligence и реляционные системы управления базами данных с поддержкой языка SQL

По данным отчетов, в 2020 году мировой рынок Big Data составляет 138,9 млрд долларов, к 2025 году он вырастет до 229,4 млрд долларов — будет расти по 10,6% в год. Вплоть до 2025 года лидерство на рынке будет удерживать Северная Америка, в частности США.

В основном такой рост вызван повышением интереса к IoT — сейчас к интернету вещей подключено 30,73 млрд устройств, а к 2025 году их будет 75,44 млрд. Кроме того, уже сейчас без больших данных компании не выдерживают конкуренцию с теми, кто использует Big Data, так как не могут обеспечивать достаточный уровень клиентского сервиса.

При этом нельзя сказать, что большие данные просто тренд, которому компании следуют бездумно. Опросы показывают, что big data помогают бизнесу на 8% увеличить прибыль и на 10% снизить расходы.

Чтобы работать с большими данными, придется учесть несколько моментов:

Готовьте много места. Если вернуться к определению биг даты, то видно, что данных будет немало, значит, нужно быть готовыми где-то их хранить. Также информация может поступать с высокой

скоростью, поэтому заранее смотрите, чтобы ширины входного канала и скорости дисков хватало для обработки входящего потока байтов [7].

Готовьте больше серверов. Данные нужно не только хранить, но и как-то обрабатывать. Из-за больших объемов вам, скорее всего, придется разбивать информацию на порции и обрабатывать их параллельно на разных машинах, то есть использовать упомянутые выше технологии MapReduce. Для этого придется заранее озаботиться достаточным количеством железа для вычислений.

Готовьте правильные инструменты. IT-специалисты много лет занимаются поиском крупиц золота в горах разнообразных больших данных. Для их расчетов создано много надежных, классных и быстрых инструментов, например: Hadoop, Spark и другие. Познакомьтесь с основными продуктами на рынке и выберите, что подойдет вам.

Подготовка инфраструктуры занимает много времени, поэтому лучше переложить ее на плечи профессиональных администраторов и присмотреться к облачным решениям по обработке Big Data.

Простыми словами Big Data — это большие объемы цифровой информации, которая непрерывно пополняется. Обычно такая информация слабо структурированная и разнородная. Также под этим термином могут объединять технологии хранения и обработки больших данных.

Большие данные помогают анализировать текущее состояние бизнеса, строить прогнозы и автоматизировать рутинные процессы. Для работы с ними используют специальные технологии, которые позволяют быстро обрабатывать огромные массивы информации и извлекать из них пользу. Кроме инструментов, вам понадобится много дискового пространства и много серверов. Необходимые мощности можно арендовать в облаке. Рынок Big Data в мире растет на 10,6% в год, в основном за счет роста популярности интернета вещей (IoT).

67 лет назад, когда был разработан первый транзистор, никто не мог предсказать, какую роль компьютер будет играть в нашем обществе сегодня через Интернет. Интернет не является мгновенным успехом. Он проделал путь ок. 24 года. За этот короткий промежуток времени это затрагивает почти все аспекты жизни. Мы не можем представить себе банковскую систему, систему бронирования железнодорожных билетов, данные авиакомпаний, бизнес-данные и т. д. без централизованного хранилища данных в Интернете. Из-за

быстрого использования Интернета через социальные сети, службы электронной почты, веб-инструменты связи, веб-конференции и т. д. объем данных в Интернете быстро увеличился. Данные, которые выходят за рамки емкости хранения и возможностей обработки классического компьютера, называются большими данными, и получение некоторой информации из большого количества данных является очень большой проблемой. ИТ-компании, такие как Facebook, Google, Amazon, Salesforce и т. д., управляют своими данными в крупных быстроэластичных центрах обработки данных. Эти организации также предоставляют по запросу различные услуги, такие как хранилище (SaaS), платформа (PaaS), приложения (AaaS) и т. д. на условиях аренды, называемые облачными вычислениями. Облачная инфраструктура очень хороша для небольших организаций, и они обращаются к поставщикам облачных услуг. Из-за огромного увеличения объема данных в облаке ИТ-отрасли сталкиваются с очень серьезной проблемой анализа больших данных. Аналитика больших данных позволяет получить представление об огромном наборе данных, получить бизнес-аналитику, найти закономерности, сделать выводы и сделать некоторые прогнозы. На рынке доступно так много инструментов анализа больших данных, но все еще существует проблема более высокого ранга, которую не могут решить в оптимальное время даже самые совершенные классические компьютеры. Для анализа больших данных нам нужна система с огромными возможностями обработки. Как мы знаем, скорость обработки обычного компьютера зависит от количества транзисторов, которые мы используем. Увеличивая количество транзисторов, мы можем увеличить производительность обработки или использовать HDFS и MapReduce для более быстрой обработки данных [8].

2.2. Квантовые вычисления в BigData

Когда мы говорим о высокопроизводительных вычислениях, термин «квантовые вычисления» радикально изменил наше мышление и предоставил возможности обработки вычислений в порядке 2^n для ввода n кубитов. Перспективы квантовых компьютеров заключаются в том, что вычисления, на которые обычные компьютеры тратят часы, квантовые компьютеры могут выполнить за секунды.

Большие данные. Данные, которые выходят за рамки возможностей хранения и обработки классического компьютера, называются большими данными.

Источники генерации данных: данные социальных сетей, сенсорная сеть, поиск в Интернете, геномика, астрономия, данные авиакомпаний и т. д.

Типы данных:

Структурированные данные: данные заданного формата, адресная книга, каталоги продуктов, банковские переходы.

Неструктурированные данные: данные, которые не имеют заранее установленного формата. Фильмы, аудио, текстовые файлы, веб-страницы, компьютерные программы, социальные сети.

Полуструктурированные данные: неструктурированные данные, которые можно структурировать с помощью доступных описаний форматов.

Причины больших данных:

I. Недорогое хранилище для хранения данных, которые были удалены ранее.

II. Мощные многоядерные процессоры [9].

III. Низкая задержка возможна благодаря распределенным вычислениям: компьютерный кластер и сети соединены через высокоскоростную сеть.

IV. Виртуализация: разделяйте, агрегируйте, изолируйте ресурсы любого размера и динамически изменяйте их.

V. Доступное хранилище и вычисления с минимальными затратами рабочей силы через облака.

VI. Лучшее понимание распределения задач (карта сокращения), вычислительной архитектуры (Hadoop).

VII. Передовые аналитические методы (машинное обучение).

VIII. Управляемые платформы больших данных: поставщики облачных услуг.

IX. Программное обеспечение с открытым исходным кодом: открытое

стек, PostGresSQL.

X. 12 марта 2012 г. Обама объявил о выделении 200 миллионов долларов на исследования больших данных для NSF, NIH, DOE, DoD, DARPA и USGS (геологическая разведка) [10-15].

Квантовые вычисления: Квантовый компьютер — это компьютер, который использует законы квантовой механики для выполнения вычислений. Он может решать быстрее, чем современный самый быстрый компьютер. Он обещает более мощные вычислительные возможности, чем любой обычный компьютер.

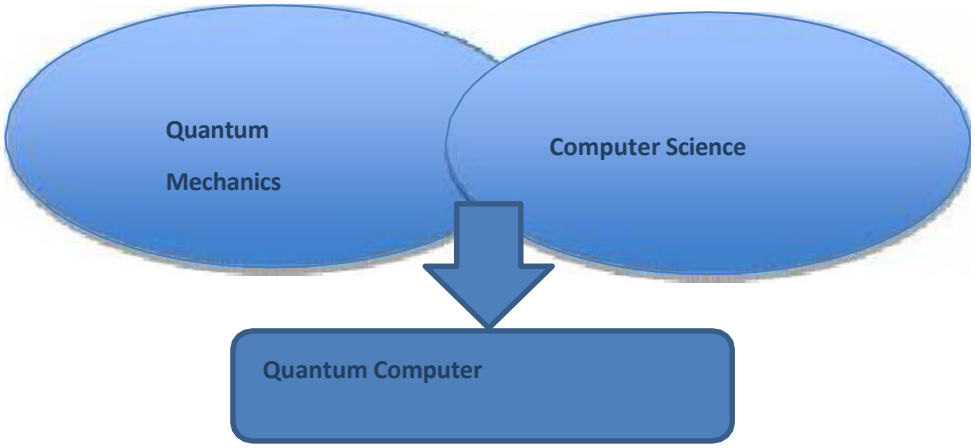


Рис. 2.1. Связь квантового компьютера с различными областями

Кубит (квантовый бит). В квантовом компьютере мы используем кубит (битовый эквивалент обычного компьютера) для хранения данных.

Генерация кубита: Маленькие частицы, такие как электроны и фотоны, имеют спин, и этот спин можно измерить с помощью магнитного поля. Если мы поместим электрон в магнитное поле, то он будет вращаться в разных направлениях и одновременно, что называется квантовой суперпозицией. Если у нас есть n бит, то в результате их суперпозиции мы получим 2^n кубитов.

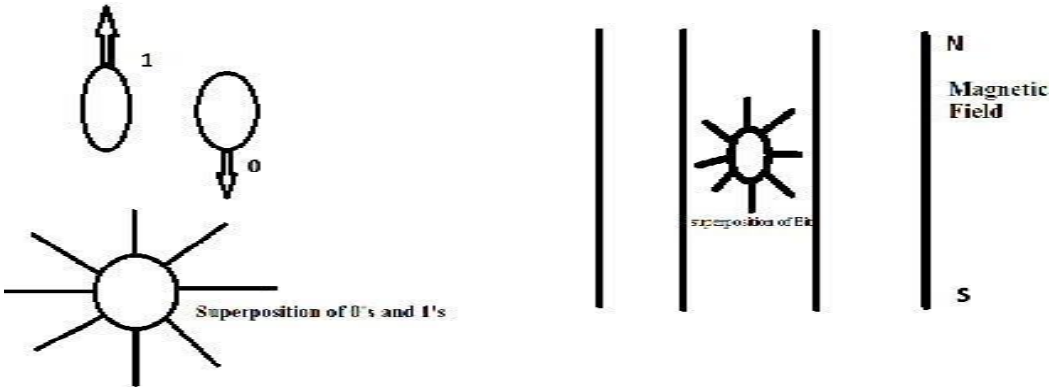


Рис. 2.2 Генерация кубита

Кубит — это аналог бита для квантовых вычислений. Это означает, что кубит можно представить как линейную комбинацию |0> и |1>:

$$|\varphi\rangle = a|0\rangle + b|1\rangle,$$

где a и b — амплитуды вероятности и комплексные числа. Когда мы измеряем этот кубит, вероятность результата |0> равна $|a|^2$, а вероятность результата |1> равна $|b|^2$. Поскольку абсолютные квадраты амплитуд равны вероятностям, из этого следует, что a и b должны быть ограничены уравнением

$$|a|^2 + |b|^2 = 1$$

То есть мы должны измерять либо одно состояние, либо другое [16-18].

Сфера Блоха: кубит $|\varphi\rangle = a|0\rangle + b|1\rangle$ можно представить в виде точки (θ, φ) на единичной сфере, называемой сферой Блоха. Определите углы θ и φ, разрешив $a = \cos\left(\theta/2\right)$ и $b = e^{i\varphi}\sin\left(\theta/2\right)$. Здесь a считается действительным, что всегда можно сделать реальным, умножив на $|\psi\rangle$ общий фазовый коэффициент (который ненаблюдаемый). Тогда $|\psi\rangle$ представляется единым вектором, называемый вектором Блоха.

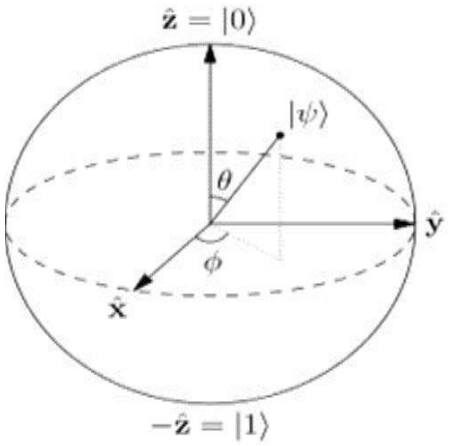


Рис. 2.3 Сфера Блоха

Квантовая запутанность: два объекта, если они квантово-механически запутаны, то они сильно связаны друг с другом, даже если они находятся на огромном расстоянии друг от друга. Это значит, что суперпозиция битов и все одновременно. Электроны внутри атома существуют на квантованных энергетических уровнях. Качественно эти электронные орбиты можно рассматривать как резонирующие стоячие волны, что находится в тесной аналогии с вибрирующими волнами, которые можно наблюдать на туго удерживаемой струне. Два таких отдельных уровня могут быть изолированы для настройки базовых состояний кубита [19-25].

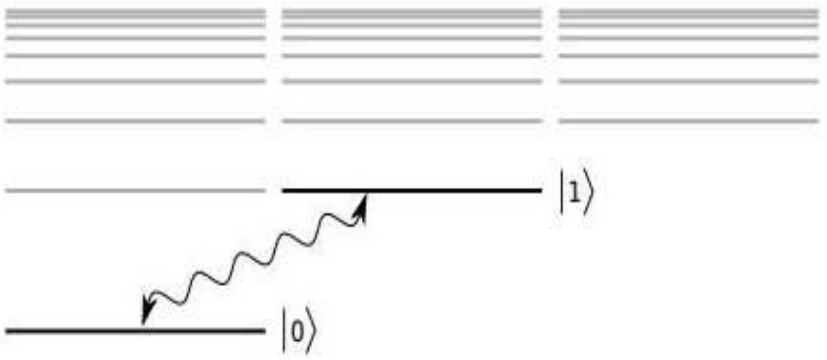


Рис. 2.4 Диаграмма энергетических уровней атома. Основное состояние и первое возбужденное состояние соответствуют уровням кубита |0> и |1>

Типы квантового компьютера:

I.Кремниевый квантовый компьютер. Использование спина электрона в качестве квантового бита или кубита.

Рис. 2.5 Кремниевые квантовые компьютеры

II. Оптический квантовый компьютер. Он использует фотон света в качестве кубита.

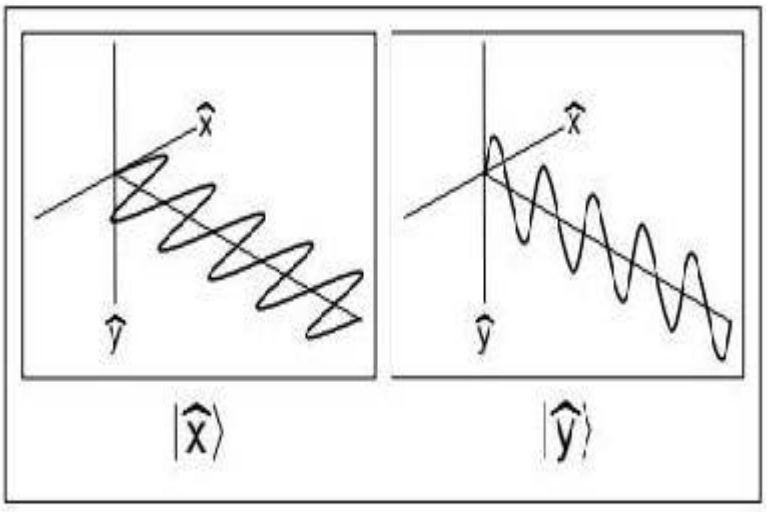


Рис. 2.6. Горизонтальная поляризация соответствует состоянию кубита |x>, а вертикальное состояние соответствует состоянию кубита |y>

Поляризацию фотона можно измерить с помощью поляроидной пленки или кристалла кальцита. Подходящим образом ориентированный поляроидный лист пропускает фотоны с x-поляризацией и поглощает фотоны с y-поляризацией. Поэтому фотон, находящийся в суперпозиции

$|\varphi\rangle = a|0\rangle + b|1\rangle$ передается с вероятностью $|a|^2$. Если фотон теперь встречает другой лист поляроида с той же ориентацией, то он передается с вероятностью 1.

III. Secure Quantum Computer – Квантовый компьютер для безопасной связи.

Рис. 2.7 Защищенный квантовый компьютер

IV. Квантовые вычисления модели шлюза — копируют цифровые вентили, которые являются строительными блоками современного компьютера, и создают квантовую эквивалентность для этих вентилей. Квантовые вычисления с крупнейшей моделью вентилей, которые позволяют сделать это сегодня, с учетом чисел 21, 7 и 3, поскольку мы все знаем, что это очень мелкомасштабный эксперимент.

Квантовые вычисления с ионной ловушкой: в качестве кубита используется ион. Вольфганг Пауль получил Нобелевскую премию по физике за работу в области квантовых вычислений на ионных ловушках. Ионы, или заряженные атомные частицы, можно удерживать и подвешивать в свободном пространстве с помощью электромагнитных полей. Кубиты хранятся в стабильных электронных состояниях каждого иона, а квантовая информация может обрабатываться и передаваться посредством коллективного квантованного движения ионов в ловушке (взаимодействующих посредством кулоновской силы). Лазеры применяются для создания связи между состояниями кубита (для операций с одним кубитом) или связи между внутренними состояниями кубита и внешними двигательными состояниями (для запутывания между кубитами).

V.D WAVE Quantum Computer: Совместный проект Google и НАСА.

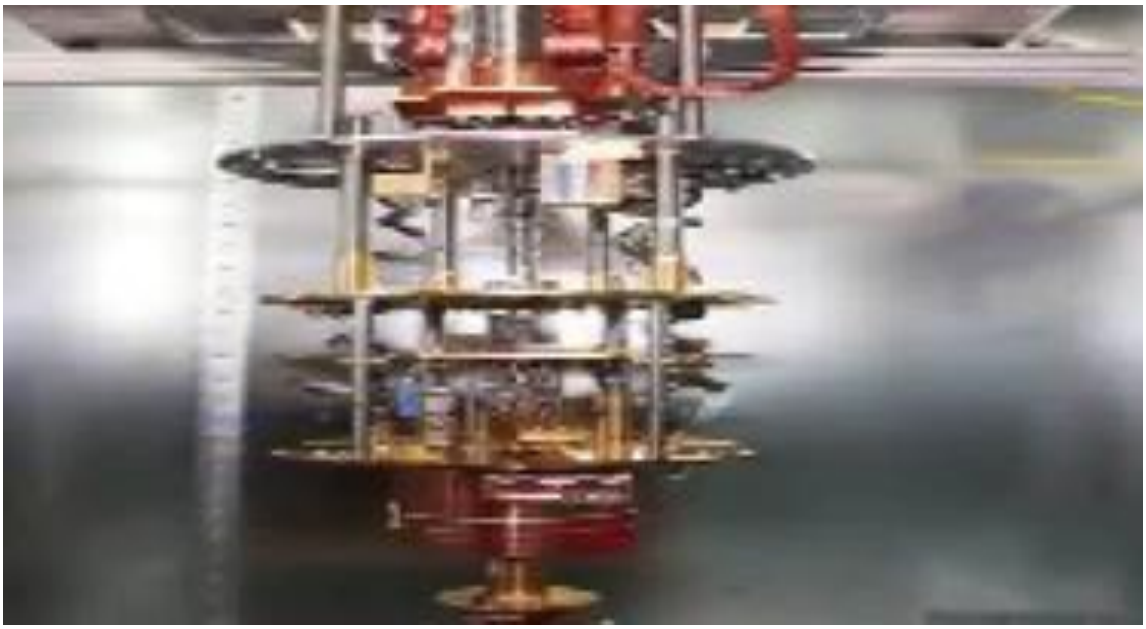


Рис. 2.8 Квантовый компьютер D WAVE

11 мая 2011 года компания D-Wave Systems анонсировала D-Wave One, интегрированную квантовую компьютерную систему, работающую на 128-кубитном процессоре. Процессор, используемый в D-Wave One под кодовым названием «Rainier», выполняет одну математическую операцию — дискретную оптимизацию. Ренье использует квантовый отжиг для решения задач оптимизации. D-Wave One — первая в мире коммерчески доступная квантовая компьютерная система. Цена составит около 10 000 000 долларов США.

Основными задачами исследования являются

I. Построить сверхбыстрые квантовые вычисления.

II. И сверхбезопасная квантовая связь.

III. Применение квантовых вычислений для бизнес-аналитики.

IV. Выяснить возможности квантовых вычислений в различных областях.

V. Разобраться в проблемах квантовых вычислений.

Потребность в квантовых вычислениях и их потенциальные преимущества перед обычными компьютерами изучались несколькими авторами (например, Канамори [1], Девитт [2], Прантош [3]). Чен и Чаян (2012) подробно обсудили процесс бизнес-аналитики и то, как анализ больших данных полезен для принятия бизнес-решений. Девитт и Манро (2011) реализовали модель высокопроизводительного квантового компьютера. Прантош (2011) обсудил будущее квантовой науки и ее масштабы в различных областях. Ааронсон обсудил в своем проекте ограничения квантовых вычислений при их практической реализации. Он подчеркивает, почему квантовому компьютеру для измерения импульса атома необходима абсолютная нулевая температура. В статье «Квантовый компьютер D WAVE» рассказывается о преимуществах и проблемах, связанных с квантовым компьютером D WAVE. В нем рассказывается, как можно использовать D WAVES при решении задачи коммивояжера и других сложных задач принятия решений.

В этом качественном исследовании мы пытаемся выяснить некоторые потенциальные преимущества и проблемы квантовых вычислений. Обычные компьютеры состоят из кремниевых чипов, на которых выгравированы более миллиардов транзисторов. Благодаря большему количеству транзисторов компьютеры становятся все быстрее и быстрее. Квантовая концепция кубита изменила всю концепцию вычислений. Обычный компьютер использует 8 бит только

для хранения одного числа от 0 до 256, где, как и в квантовом компьютере, 8 кубитов могут одновременно хранить 256 чисел, что значительно ускоряет вычислительную мощность. Давайте рассмотрим все возможные комбинации 2-битной системы данных с 4 возможными состояниями 00, 01, 10 и 11. Классический 2-битный компьютер может одновременно выполнять максимум одну из этих 4 возможных функций. Чтобы проверить их все, компьютеру пришлось бы повторить каждую операцию отдельно, тогда как 2-битный квантовый компьютер благодаря явлению суперпозиции способен анализировать все эти возможности одновременно за одну операцию. Это связано с тем, что 2 кубита содержат информация о 4 состояниях, тогда как 2 бита содержат информацию об одном состоянии.

Таким образом, машина с n кубитами может одновременно находиться в суперпозиции 2^n состояний. Компьютер с 4 кубитами мог анализировать 16 параллельных состояний за одну операцию; для сравнения, 4-битный классический компьютер может анализировать только одно состояние. Чтобы получить то же решение, что и квантовый компьютер, классический компьютер должен повторить эту операцию 16 раз.

10 кубитов — могут хранить 1024 числа.

11 кубитов – могут хранить 2048 чисел.

100 кубитов – можно хранить

1 267 650 600 228 229 401 496703205 376 чисел.

Мы можем сказать, что квантовый компьютер может решить проблему в масштабе, превосходящем любой обычный компьютер. Таким образом, мы обнаруживаем, что квантовые вычисления могут использоваться в огромных масштабах для анализа проблемы быстро растущих данных в сети, называемых большими данными. Его можно использовать для предварительного прогнозирования погоды, прогнозирования стихийных бедствий, таких как цунами, землетрясение, для бизнес-аналитики и многого другого. Врагом квантовых вычислений является среда, близкая к абсолютному нулю, очень чистая среда. Одной из самых больших проблем, с которыми сталкиваются ученые, работающие в области квантовых вычислений, является проблема декогерентности (изолировать систему от внешней среды), проблема оптимизации (выбора кратчайшего пути) и квантового туннелирования (вероятность исчезновения электрона на другой стороне) [26-29].

В этой исследовательской работе показано несколько преимуществ квантовых вычислений и проблемы их реализации. Акцент сегодняшних вычислений делается на разработке и производстве такого компьютера, который обладает огромными вычислительными возможностями для анализа быстро растущих больших объемов данных в сети. Использование квантовых вычислений позволяет нам прогнозировать статистические выводы, принимать бизнес-решения, прогнозировать погоду, сопоставление шаблонов, анализ веб-данных и многое другое. Хотя квантовые вычисления находятся на начальной стадии, будущее вычислений и анализа больших данных больше зависит от квантовых компьютеров. Использование квантовых вычислений очень экологично по своей природе. Это может сэкономить огромное количество тепла в центрах обработки данных и снизить энергопотребление с МВт до КВт. Существующие сегодня квантовые компьютеры, скажем, квант D WAVE НАСА, являются специфичными для конкретной области и используются для некоторых сложных приложений, а универсальный квантовый компьютер все еще остается для нас мечтой. Центр квантовых вычислений и коммуникационных технологий Австралии возглавляет глобальную гонку по разработке квантового компьютера и квантовой защищенной сети связи, а также имеет планы по разработке универсального квантового компьютера. Вполне возможно, что компьютер на 500 кубитов однажды сможет анализировать больше данных, чем атомов в наблюдаемой Вселенной [30-33].

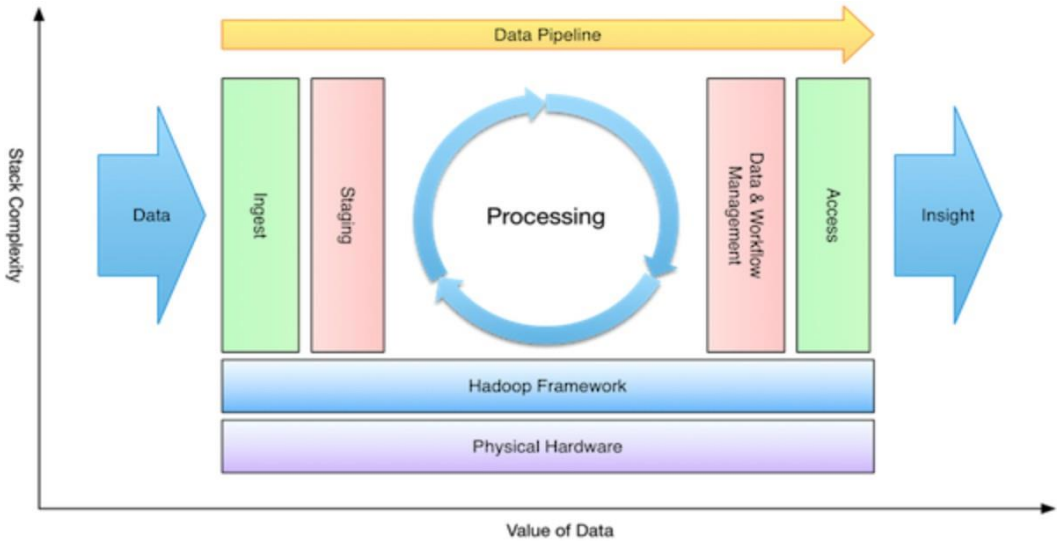


Рис. 2.9 – Стек работы с Большими данными [15]

Для работы с Большими данными разработчиками систем создаются модели данных, содержательно связанные с реальным миром. Разработка адекватных моделей данных представляет собой сложную аналитическую задачу, выполняемую системными архитекторами и аналитиками. Модель данных позволяет создать математическую модель взаимодействий объектов реального мира и включает в себя описание структуры данных, методы манипуляции данными и аспекты сохранения целостности данных. Описание разработки моделей данных не является задачей настоящего руководства.

Для хранения данных используются распределенные системы различных типов. Это могут быть файловые системы, базы данных, журналы, механизмы доступа к общей виртуальной памяти. Большинство систем хранения ориентированы исключительно на работу с Большими данными, они имеют крайне ограниченное число функций (например, может отсутствовать возможность не только модификации, но и удаления поступивших данных) что объясняется внутренней сложностью создания высокоэффективных распределенных систем. В конце текста приведены ссылки на несколько используемых в настоящий момент систем.

Для того, чтобы работа с данными происходила быстрее системы хранения и обработки данных распараллеливаются в кластере (cluster, группа компьютеров, объединенных сетью для выполнения единой задачи). Однако, согласно гипотезе Брюера невозможно обеспечить одновременную согласованность (непротиворечивость) данных, доступность данных и устойчивость системы к отделению отдельных узлов. Гипотеза доказана для транзакций типа ACID (Atomic, Consistent. Isolated, Durable) и известна под названием CAP теоремы (Consistency, Availability, Partition tolerance).

Прием данных (Data Ingestion)

Источники данных имеют различные параметры, такие как частоту поступления данных из источника, объём порции данных, скорость передачи данных, тип поступающих данных и их достоверность.

Для эффективного сбора данных необходимо установить источники данных. Это могут быть хранилища данных, поставщики агрегированных данных, API каких-либо датчиков, системные журналы, сгенерированный человеком контент в социальных сетях, в

корпоративных информационных системах, геофизическая информация, научная информация, унаследованные данные из других систем. Источники данных определяют исходный формат данных [34-40].

Например, мы можем самостоятельно проводить погодные на территории аэропорта, использовать данные, поступающие с взлетающих и садящихся самолетов, закупить данные со спутников, пролетающих над аэропортом и у местной метеослужбы, а также найти их где-то в сети в другом месте. В общем случае для каждого источника необходимо создавать собственный сборщик (Data Crawler для сбора информации в сети и Data Acquisition для проведения измерений).

Прием данных заключается в начальной подготовке данных от источников с целью приведения данных к общему формату представления данных. Этот единый формат выбирается в соответствии с принятой моделью данных. Выполняются преобразования систем измерения, типов (типизация), верификация. Обработка данных содержательно не затрагивает имеющуюся в данных информацию, но может изменять ее представление (например, приводить координаты к единой системе координат, а значения к единой размерности).

Сбор данных (Data Staging)

Этап сбора данных характеризуется непосредственным взаимодействием с системами хранения данных. Устанавливается точка сбора, в которой собранные данные снабжаются локальными метаданными и помещаются в хранилище либо передаются для последующей обработки. Данные, по каким-либо причинам не прошедшие точку сбора, игнорируются. Для структурированных данных проводится преобразование из исходного формата по заранее заданным алгоритмам. Это наиболее эффективная процедура в случае, если структура данных известна. Однако если данные представлены в двоичном виде, структура и связи между данными утеряны, то разработка алгоритмов и основанного на них программного обеспечения для обработки данных может оказаться крайне затруднительной.

Для полуструктурированных данных требуется интерпретация поступающих данных и использование программного обеспечения, умеющего работать с используемым языком описания

данных. Существенным плюсом полуструктурированных данных является то, что в них зачастую содержатся не только сами данные, но метаданные в виде информации о связях между данными и способах их получения. Разработка программного обеспечения для обработки полуструктурированных данных представляет собой достаточно сложную задачу. Однако имеется значительное количество готовых конвертеров, которые могут, например, извлечь данные из формата XML в сформированное табличное представление.

Наибольшего объема работ требует обработка неструктурированных данных. Для их перевода к заданному формату может потребоваться создание специального ПО, сложная ручная обработка, распознавание и выборочный ручной контроль. На этапе сбора проводится контроль типов данных и может выполняться базовый контроль достоверности данных. Например, координаты молекул газа, содержащихся в какой-либо области, не могут лежать за пределами этой области, а скорости – существенно превышать скорость звука. Для того, чтобы избежать ошибок типизации, необходимо проверять правильно ли заданы единицы измерения. Например, в одном наборе данных высота может измеряться в километрах, а в другом – в футах. В этом случае необходимо произвести преобразование высоты в те единицы измерения, которые приняты в используемой модели.

При сборе данные систематизируются и снабжаются метаданными, хранимыми в связанных метаданных. При наличии большого количества источников данных может потребоваться управление сбором данных для того, чтобы сбалансировать объемы информации, поступающие из различных источников. Собранные данные либо сохраняются в системах хранения, либо (в особенности, для потоковых данных) передаются для анализа в реальном времени.

Анализ данных (Analysis Layer)

Анализ данных, в отличии от сбора данных, использует информацию, содержащуюся в самих данных. Анализ может проводиться как в реальном времени, так и в пакетном режиме. Анализ данных составляет основную по трудоемкости задачу при работе с Большими данными. Существует множество методик обработки данных: предиктивный анализ, запросы и отчетность, реконструкция по математической модели, трансляция, аналитическая

обработка и другие. Методики используют специфические алгоритмы в зависимости от поставленных целей. Например, аналитическая обработка может являться анализом изображений, социальных сетей, географического местоположения, распознавания по признакам, текстовым анализом, статистической обработкой, анализом голоса, транскрибированием.

Алгоритмы анализа данных также, как и алгоритмы обработки данных, опираются на модель данных. При этом при анализе может быть использовано несколько моделей, задающих общий формат данных, но по-разному моделирующие содержательные процессы, данные о которых мы обрабатываем. При использовании при анализе методов искусственного интеллекта, в частности нейронных сетей, производится динамическое обучение моделей на различных наборах данных. При анализе данных производится идентификация сущностей, описываемых данными на основании имеющейся в данных информации и используемых моделей. Сущностью анализа является аналитических механизм, использующий аналитические алгоритмы, управление моделями и идентификацию сущностей для получения новой содержательной информации, являющейся результатом анализа.

Для анализа данных также используются методы искусственного интеллекта на нейронных сетях, не рассматриваемые в данном учебном пособии.

Представление результатов (Consumption Layer)

Результаты анализа данных предоставляются на уровне потребления. Имеется несколько механизмов, позволяющих использовать результаты анализа больших данных.

Мониторинг метаинформации.

Подсистема отображения в реальном времени существенных параметров работы системы, загруженности вычислителей, распределение задач в кластере, распределение информации в хранилищах, наличие свободного места в хранилищах, поступление данных от источников, активности пользователей, отказов оборудования и тд.

Мониторинг данных.

Подсистема отображения в реальном времени процессов приема, сбора и анализа данных, навигация по данным.

Генерация отчетов, запросы к данным, представление данных в виде визуализации на дэшбордах (Dashboard), в формате PDF,

инфографике, сводных таблицах и кратких справках

Преобразование данных и экспорт в другие системы, интерфейс с BI- системами.

2.3.Параллельные алгоритмы для работы с данными

Операторы Map и Reduce.

Для параллельной обработки данных в интерфейсе MPI (Message Passing Interface), являющимся распространённым стандартом для обмена данными при организации параллельных вычислений, была предложена ныне широко используемая во множестве реализаций парадигма параллельной обработки наборов данных при помощи использования операторов Map и Reduce.

Оператор Reduce (свертка).

Свертка в простейшем случае получает на входе функцию от двух параметров из набора данных, состоящего из элементов одного типа. На выходе она возвращает один элемент такого же типа. В общем случае от функции требуется коммутативность и ассоциативность на множестве определения. В этом случае порядок вычислений несущественен, алгоритм позволяет эффективно распараллеливаться, так как есть возможность выбрать произвольные пары элементов набора данных, от выбранных пар при помощи переданной в Reduce функции параллельно вычислить значения и исходные пары элементов заменить на вычисленные значения. Алгоритмы, в которых необходимая нам функция свертки коммутативна и ассоциативна, высокоэффективна при обработке Большого неупорядоченного (или упорядоченного, для вычисления это несущественно) набора данных.

Для ассоциативной, но некоммутативной функции свертка определена для набора данных, элементы которого упорядочены. В этом случае мы можем разбить исходный набор на несколько упорядоченных между собой последовательных поднаборов, например, по числу имеющихся вычислителей в кластере. С математической точки зрения эта процедура соответствует расстановке скобок. Затем, получив по одном элементу данных из каждого поднабора, мы имеем промежуточный упорядоченный набор данных, и выполняем свертку над ним и далее рекурсивно до получения итогового значения.

В более сложном случае мы хотим при работе Reduce получить данные другого типа, например, на входе мы имеем набор с массой и скоростями молекул, а на выходе нам нужно получить суммарную массу и импульс (чтобы узнать среднюю скорость ветра разделив импульс на массу). Для решения такой задачи функция, передаваемая в свертку, существенно видоизменяется следующим образом: она получает один параметр результирующего типа (называемый аккумулятором) и один параметр исходного типа (итератор). В самом начале аккумулятор имеет некоторое начальное значение. Для нашего примера с молекулами передаваемое в свертку начальное значение – это структура из четырёх элементов: масса равна нулю и три компонента (по координатам *x*, *y* и *z*) импульса тоже равны нулю.

Затем элементы данных перебираются и производятся вычисления, определённые функцией. В нашем примере масса молекулы, переданной в итераторе, прибавляется к массе в аккумуляторе, далее вычисляется импульс молекулы (вектор из произведений массы на компоненты скорости) и прибавляется к компонентам импульса аккумулятора.

Разбив исходный набор данных на поднаборы мы для каждого вычислим массу и импульс. Однако далее у нас появляется проблема: имеющаяся функция умеет добавлять к данным имеющим типа аккумулятора (масса, импульс) данные имеющие тип итератора (масса, скорость). Чтобы произвести итоговые вычисления, нам необходимо произвести переопределение функции: в том случае, если передаётся два параметра, имеющих тип аккумулятора, то функция начинает вести себя совсем по-другому. А именно складывает массы и импульсы. Иначе говоря, по сути у нас появляются две функции: одна работает с мастер-данными, а другая с промежуточными данными. Таким образом, мы можем провести эффективное распараллеливание. Обратим внимание что этот метод также допускает работу с некоммутирующими, но упорядоченными данными в случае ассоциативных функций.

Иногда для удобства вычислений, кроме описанной выше прямой (левой) свертки, при которой функция вычисляется от первых двух элементов (либо от начального значения и первого элемента данных), затем от полученного аккумулятора и следующего элемента данных (итератора) и так далее до завершения вычисления, вводят также правую (обратную) свертку. При обратной свертке

функция вычисляется сначала от начального значения (если оно задано) и последнего элемента, затем от аккумулятора и предпоследнего элемента, и так далее, пока не дойдём до вычисления от аккумулятора первого элемента.

Эффективное распараллеливание вычисления свертки от некоммутативных неассоциативных функций в общем случае невозможно. Заметим, что Гвидо ван Россум (BDFL, Benevolent Dictator For Life, великодушный пожизненный диктатор языка Python) сознательно перевел свертку из общего пространства имен в модуль functools поскольку в большинстве приложений ее использовали либо для таких тривиальных вещей, как суммирование массивов, либо для получения совершенно непонятного нечитаемого кода.

Оператор Map.

Этот оператор известен во многих языках программирования под различными именами, кроме Map используется также названия Select и Transform.

В простейшем случае Map создаёт из переданной ему функции и упорядоченного набора данных другой упорядоченный набор данных, в котором каждый элемент является значением функции от соответствующего элемента исходного набора данных.

Существуют расширения оператора Map на несколько наборов данных, при этом функция должна быть определена от соответствующего количества элементов. Возвращается набор данных, элементы которого являются значениями функции от соответствующих наборов. При этом есть два способа определить поведение расширенного оператора Map в случае различия в размерах передаваемых наборов данных – либо результирующий набор имеет размер равный размеру наименьшего набора из передаваемых, либо размеру максимального, при этом функции передаются неопределённые значения.

Лямбда-архитектура.

Архитектура, позволяющая проводить работу с данными при помощи эквивалентных алгоритмов одновременно в пакетном режиме и в реальном времени. Аналитика в реальном времени может быть неточной, выполняется в оперативной памяти, но предоставляются быстро. Расчеты в пакетном режиме обеспечивают хранение полученных результатов и выдает достоверные данные, но выполняется долго.

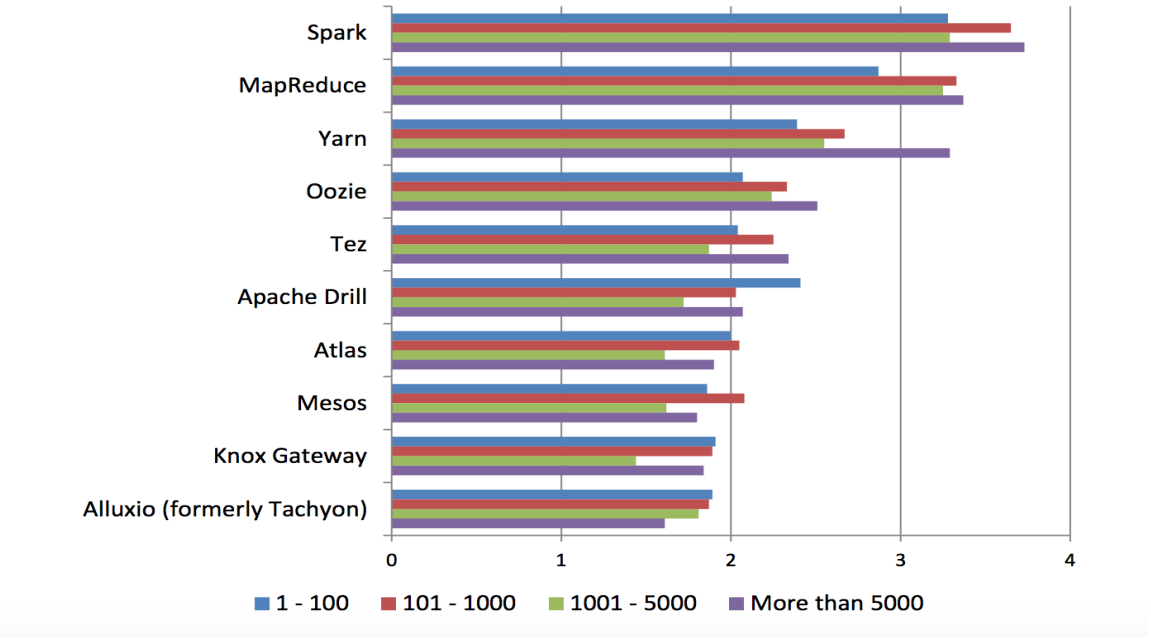


Рис. 2.10 – Инфраструктуры Больших данных в разрезе размера предприятий [18]

Многокомпонентная система от Dell EMC

SAP Predictive Analytics https://www.sap.com/products/predictive-analytics.html

Система от мирового лидера ERP, интегрируется с SAP HANA

Oracle Big Data Preparation https://cloud.oracle.com/bigdatapreparation Облачное решение от мирового лидера баз данных

Оборудование для обработки Больших данных

Комплект оборудования для обработки Больших данных монтируется в ЦОД. Основными компонентами системы являются система управления, вычислительные ресурсы, система хранения данных, локальная сеть. Электропитание, мониторинг, доступ к интернету и другие внешние ресурсы предоставляют центры обработки данных (ЦОД).

Система управления предназначена для общего управления системой, внешнего доступа, обеспечения аутентификации, авторизации и аккаунтинга и представления результатов пользователям. Выполняется на базе обычной серверной платформы, специфических требований не имеет.

На рис. 2.11 представлен кластер Hadoop от Yahoo 10 летней давности [16].

Рис. 2.11 – Кластер Hadoop от Yahoo 10 летней давности

Вычислительные ресурсы кластера состоят из узлов (nodes), основными параметрами которых является объем оперативной памяти с контролем четности (ECC) и максимальное количество ядер CPU. Расчётным параметрами является размер оперативной памяти на ядро и скорость работы процессора. Высокая отказоустойчивость узлов желательна, но в целом не требуется, некоторое количество отказов является нормальным и компенсируется при помощи программного обеспечения – отказавший узел автоматически выводится из эксплуатации, и его работа перераспределяется на другие узлы.

В некоторых случаях для обработки Больших данных, особенно при использовании нейронных сетей, используют вычислители на базе GPU, аналогичные используемым для HPC.

Распределённая система хранения данных состоит из дисковых полок, обеспечивающих максимально быстрый доступ к данным. Резервирование выполняется при помощи программного обеспечения систем хранения и в общем случае не требуется. Зачастую также используются компьютеры, в которых подключено большое количество локальных дисков.

Рис. 2.12 – Центр хранения данных производства Titan Power [21]

Сетевая инфраструктура связывает вычислительные ресурсы, систему хранения и систему управления в единое целое. Основное требование к локальной сети – низкие задержки. При большом количестве узлов используются сетевые коммутаторы, начинающие передачу пакета данных сразу после обработки заголовка пакета данных. При малом количестве узлов распространено прямое соединение компьютеров по схеме гиперкуб, где размерность куба соответствуют числу портов на интерфейсных платах. Ранее массово использовались сети на базе различных вариантов интерфейса Infiniband, сейчас, в основном, используется 100 Gigabit Ethernet. Сравнительно недавно появился 200 и 400 Gigabit Ethernet.

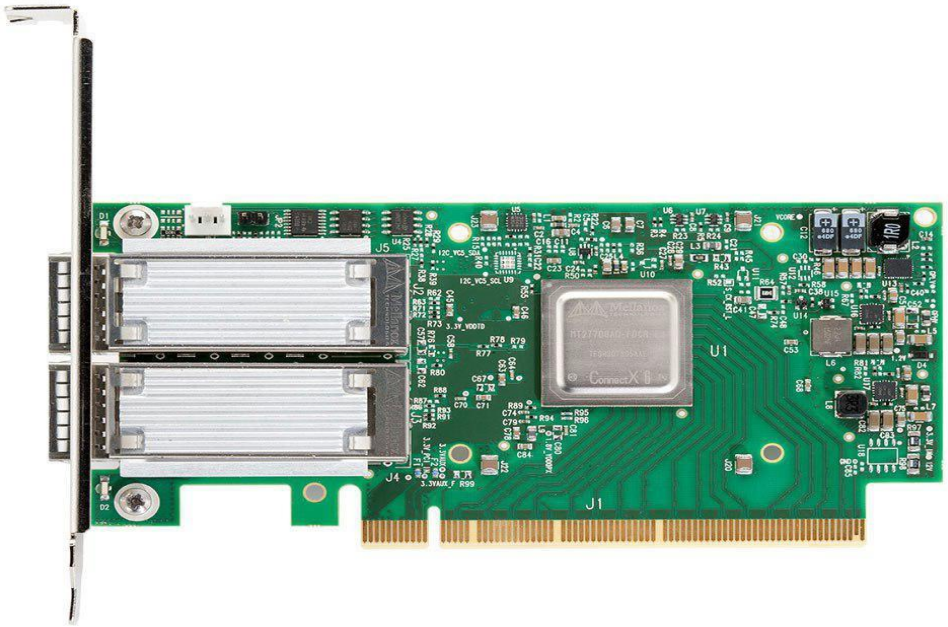


Рис. 2.13 – ConnectX®-6 EN двухпортовой сетевой адаптер 200Gb/s Ethernet [22]

Имеет несколько активных каналов распределения нагрузки и охлаждения с резервными компонентами (2 (N+1), т.е. 2 ИБП с избыточностью N+1 каждый).

Время развертывания – от 15 до 20 месяцев.

Годовое время простоя – 0,4 часа.

Рис. 2.14 – Инженер обслуживает оборудование в центре обработки данных [23]

2.4. Связь между искусственным интеллектом и квантовыми вычислениями

Применение квантовых вычислений в искусственном интеллекте

Квантовые вычисления (КВ) представляют собой новую область в информатике, которая использует принципы квантовой механики для обработки информации. Искусственный интеллект (ИИ) включает в себя различные методы и алгоритмы, которые позволяют компьютерам имитировать интеллектуальное поведение.

Применение квантовых вычислений в искусственном интеллекте может привести к созданию более мощных и эффективных систем, способных решать сложные задачи. Вот некоторые из областей, где квантовые вычисления могут быть применены в искусственном интеллекте:

Машинное обучение

Машинное обучение является ключевой областью искусственного интеллекта, где компьютеры обучаются на основе данных и опыта. Квантовые вычисления могут улучшить процесс обучения, позволяя обрабатывать большие объемы данных и решать сложные задачи классификации и кластеризации.

Оптимизация

Квантовые вычисления могут быть использованы для решения оптимизационных задач, таких как поиск оптимального решения или нахождение наилучшего пути. Это может быть полезно в различных областях, таких как логистика, финансы и производство.

Распознавание образов

Квантовые вычисления могут быть применены для улучшения процесса распознавания образов, что является важной задачей в области компьютерного зрения. Квантовые алгоритмы могут обрабатывать большие объемы данных и находить сложные закономерности в изображениях.

Генетические алгоритмы

Генетические алгоритмы используются для решения оптимизационных задач, имитируя процесс естественного отбора. Квантовые вычисления могут улучшить эффективность генетических алгоритмов, позволяя обрабатывать большие объемы данных и находить оптимальные решения.

Это лишь некоторые примеры применения квантовых вычислений в искусственном интеллекте. С развитием технологий и исследований в этой области, ожидается, что будут открыты новые возможности и применения.

Преимущества и ограничения использования квантовых вычислений в искусственном интеллекте

Преимущества:

1. Обработка больших объемов данных: Квантовые вычисления могут обрабатывать и анализировать огромные объемы данных значительно быстрее, чем классические компьютеры. Это позволяет искусственному интеллекту работать с большими наборами данных и находить более точные и сложные закономерности.

2. Решение сложных оптимизационных задач: Квантовые вычисления могут эффективно решать сложные оптимизационные задачи, которые являются ключевыми в искусственном интеллекте. Они могут найти оптимальные решения в большом пространстве возможных вариантов.

3. Ускорение обучения моделей: Квантовые вычисления могут ускорить процесс обучения моделей искусственного интеллекта. Они могут обрабатывать большое количество данных параллельно и выполнять операции с большой точностью, что позволяет моделям быстрее учиться и адаптироваться к новым данным.

4. Решение проблемы экспоненциального роста вычислительной сложности: Некоторые задачи в искусственном интеллекте имеют экспоненциальный рост вычислительной сложности при увеличении размера входных данных. Квантовые вычисления могут решать эти задачи более эффективно, сокращая время выполнения и уменьшая вычислительную сложность.

Ограничения:

1. Технические сложности: Квантовые вычисления требуют специального оборудования и инфраструктуры, которые до сих пор находятся в стадии разработки и совершенствования. Это ограничивает доступность и применимость квантовых вычислений в искусственном интеллекте.

2. Чувствительность к ошибкам: Квантовые вычисления подвержены различным типам ошибок, таким как декогеренция и квантовые шумы. Это может привести к неточным результатам и

ограничить надежность и точность искусственного интеллекта, основанного на квантовых вычислениях.

3. Сложность программирования: Разработка и программирование квантовых алгоритмов требует специальных знаний и навыков. Это ограничивает доступность и использование квантовых вычислений в искусственном интеллекте для широкого круга разработчиков.

4. Ограниченное количество квантовых алгоритмов: На данный момент существует ограниченное количество квантовых алгоритмов, которые могут быть применены в искусственном интеллекте. Это ограничивает возможности использования квантовых вычислений для решения различных задач в искусственном интеллекте.

Несмотря на ограничения, квантовые вычисления представляют большой потенциал для развития искусственного интеллекта. С развитием технологий и исследований в этой области, ожидается, что будут найдены решения для преодоления этих ограничений и расширения возможностей квантовых вычислений в искусственном интеллекте.

Текущие исследования и разработки в области искусственного интеллекта и квантовых вычислений

Разработка квантовых алгоритмов для машинного обучения

Одной из активных областей исследований является разработка квантовых алгоритмов, которые могут быть применены в задачах машинного обучения. Квантовые алгоритмы могут предложить новые подходы к решению сложных задач, таких как классификация данных, кластеризация и оптимизация.

Создание квантовых нейронных сетей

Исследователи также работают над созданием квантовых нейронных сетей, которые могут использоваться для обработки и анализа данных. Квантовые нейронные сети могут предложить новые возможности в области распознавания образов, обработки естественного языка и других задач машинного обучения.

Разработка квантовых алгоритмов для оптимизации

Квантовые вычисления также могут быть применены для решения задач оптимизации, таких как поиск оптимальных решений или оптимизация параметров моделей машинного обучения. Исследователи работают над разработкой квантовых алгоритмов,

которые могут эффективно решать такие задачи и достигать лучших результатов по сравнению с классическими методами оптимизации.

Исследование квантовых алгоритмов для обработки больших объемов данных

С ростом объемов данных, требующих обработки, становится все более важным разработка эффективных алгоритмов для их анализа. Квантовые вычисления могут предложить новые подходы к обработке больших объемов данных, таких как анализ графов, обработка изображений и анализ текстов. Исследователи работают над разработкой квантовых алгоритмов, которые могут обрабатывать данные более эффективно и быстро, чем классические методы.

Разработка квантовых компьютеров и квантовых симуляторов

Одним из ключевых направлений исследований является разработка квантовых компьютеров и квантовых симуляторов. Квантовые компьютеры представляют собой устройства, которые могут выполнять квантовые вычисления, а квантовые симуляторы позволяют моделировать и анализировать поведение квантовых систем. Исследователи работают над созданием более мощных и стабильных квантовых компьютеров и симуляторов, чтобы расширить возможности квантовых вычислений в искусственном интеллекте.

В целом, исследования и разработки в области искусственного интеллекта и квантовых вычислений продолжаются, и ожидается, что в будущем будут найдены новые методы и алгоритмы, которые позволят использовать квантовые вычисления для решения сложных задач в искусственном интеллекте.

Таблица 2.1

Связь между искусственным интеллектом и квантовыми вычислениями

Тема	Описание	Пример
Искусственный интеллект	Область науки, изучающая создание интеллектуальных систем, способных выполнять	Программа, способная распознавать

Тема	Описание	Пример
	задачи, требующие человеческого интеллекта.	образы на изображениях.
Квантовые вычисления	Методы и технологии обработки информации, основанные на принципах квантовой механики.	Алгоритм Шора, использующий квантовые вычисления для факторизации больших чисел.
Связь между искусственным интеллектом и квантовыми вычислениями	Исследование и применение квантовых вычислений в области искусственного интеллекта для решения сложных задач.	Использование квантовых алгоритмов для обучения нейронных сетей.
Применение квантовых вычислений в искусственном интеллекте	Использование квантовых алгоритмов и квантовых компьютеров для решения задач искусственного интеллекта.	Разработка квантовых алгоритмов для оптимизации параметров нейронных сетей.
Преимущества и ограничения использования квантовых вычислений в искусственном интеллекте	Анализ преимуществ и ограничений квантовых вычислений в контексте их применения в искусственном интеллекте.	Большая вычислительная мощность квантовых компьютеров, но сложность в реализации и управлении

Тема	Описание	Пример
		квантовыми системами.
Текущие исследования и разработки в области искусственного интеллекта и квантовых вычислений	Обзор актуальных исследований и разработок, связанных с применением квантовых вычислений в искусственном интеллекте.	Разработка новых квантовых алгоритмов для обработки естественного языка.

2.5. Квантовые и классические нейронные сети

Классические нейронные сети — это алгоритмические модели, вдохновленные человеческим мозгом, которые можно обучить распознавать закономерности в данных и научиться решать сложные проблемы. Они основаны на ряде взаимосвязанных узлов или *нейронов* , организованных в многоуровневую структуру, с параметрами, которые можно изучить, применяя стратегии машинного или глубокого обучения.

Мотивацией квантового машинного обучения (QML) является интеграция понятий квантовых вычислений и классического машинного обучения, чтобы открыть путь для новых и улучшенных схем обучения. КНС применяют этот общий принцип, комбинируя классические нейронные сети и параметризованные квантовые схемы. Поскольку они лежат на пересечении двух полей, QNN можно рассматривать с двух точек зрения:

С точки зрения машинного обучения QNN снова представляют собой алгоритмические модели, которые можно обучить находить скрытые закономерности в данных аналогично их классическим аналогам. Эти модели могут загружать классические данные (входные данные) в квантовое состояние, а затем обрабатывать их с помощью квантовых вентилей, параметризованных обучаемыми весами . На рисунке 2.15 показан общий пример QNN, включая этапы

загрузки и обработки данных. Результаты измерения этого состояния затем можно подключить к функции потерь для обучения весов посредством обратного распространения ошибки.

С точки зрения квантовых вычислений , QNN — это квантовые алгоритмы, основанные на параметризованных квантовых схемах, которые можно обучать вариационным способом с использованием классических оптимизаторов. Эти схемы содержат карту признаков (с входными параметрами) и анзац (с обучаемыми весами), как показано на рисунке 2.15.

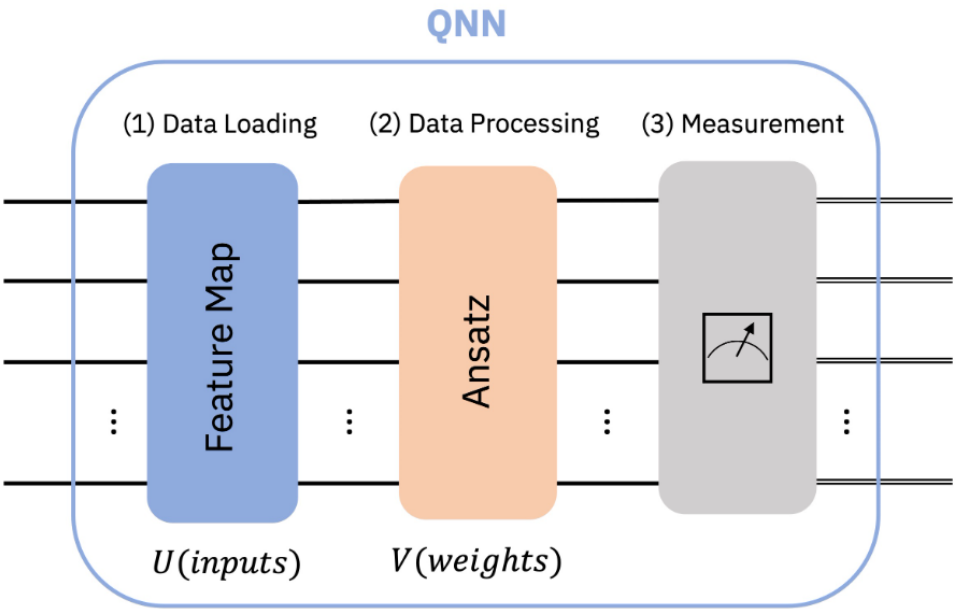


Рисунок 2.15. Общая структура квантовой нейронной сети (QNN)

Как видите, эти две точки зрения дополняют друг друга и не обязательно основаны на строгих определениях таких понятий, как «квантовый нейрон» или того, что представляет собой «слой» QNN. Модель машинного обучения использует квантовые вычисления для классификации данных. Она использует комбинацию методов, таких как сопоставление четности (parity matching) и горячее кодирование (one-hot encoding), чтобы обработать и классифицировать битовые строки.

VQC (Variational Quantum Classifier) является специальным вариантом с использованием расширения SamplerQNN (Quantum Neural Network Sampler). Суть VQC заключается в использовании квантовых вычислений для обучения модели классификации данных. Она принимает битовые строки в качестве входных данных и

генерирует вектор вероятностей для различных классов, интерпретируемый как результат горячего кодирования. Функция CrossEntropyLoss используется для обучения модели и ожидает, что метки будут представлены в формате горячего кодирования. Таким образом, модель возвращает прогнозы также в этом формате, что обычно удобно для обработки и интерпретации результатов классификации.

Variational Quantum Classifier (VQC):

VQC - это метод машинного обучения, который использует квантовые вычисления для классификации данных.

Основная идея заключается в том, чтобы использовать квантовую схему (quantum circuit), которая может быть параметризована, чтобы выполнять классификацию данных.

В VQC, квантовая схема принимает на вход некоторые данные и параметры, и выдаёт результат, который затем используется для принятия решения о классе, к которому относится входные данные.

SamplerQNN (Quantum Neural Network Sampler):

Это расширение для VQC, которое добавляет возможность использования различных алгоритмов выборки (sampling) в квантовой схеме. Вместо того, чтобы просто вычислять вероятности классов, как это делается в традиционной VQC, SamplerQNN позволяет использовать квантовые вычисления для генерации выборок из распределений вероятностей. Эти выборки могут затем использоваться для аппроксимации различных статистических характеристик или для других задач, таких как генерация прогнозов с учётом неопределённости.

Таким образом, расширение SamplerQNN предоставляет дополнительные возможности для работы с квантовыми вычислениями в контексте классификации данных, что может привести к улучшению производительности и эффективности модели.

В основе VQC (Variational Quantum Classifier) лежит использование квантовых вычислений для обучения модели классификации данных. Классификация данных - это задача машинного обучения, в которой модель должна принимать входные данные и определять к какому классу или категории они относятся. Например, классификация может использоваться для определения, является ли изображение кошкой или собакой. Квантовые вычисления используют принципы квантовой механики для обработки

информации. В отличие от классических битов, которые могут находиться в состоянии 0 или 1, квантовые биты (qubits) могут находиться в суперпозиции этих состояний, что позволяет выполнять параллельные вычисления над множеством возможных состояний. В контексте машинного обучения обучение модели - это процесс настройки параметров модели на основе предоставленных данных (набора обучающих данных). Цель состоит в том, чтобы модель могла делать точные прогнозы для новых данных, которые не использовались в процессе обучения. В VQC квантовая схема, параметризованная определенными параметрами, используется для обработки входных данных. Затем параметры этой схемы настраиваются с помощью оптимизации, чтобы минимизировать ошибку классификации на обучающих данных. Это позволяет модели адаптироваться к данным и обучаться на них. Таким образом, VQC использует принципы квантовых вычислений для эффективного обучения модели классификации данных, что может привести к созданию более мощных и эффективных моделей машинного обучения.

2.6. Алгоритмы машинного обучения

Машинное обучение, раздел искусственного интеллекта – это процесс, в ходе которого вычислительная система обрабатывает большое число примеров, выявляет закономерности и использует их, чтобы прогнозировать характеристики новых данных. Машинное обучение – дисциплина, которая зародилась как часть теории статистики. Но сегодня она актуальна как никогда, так как позволяет извлекать знания из данных.

Машинное обучение (machine learning - ML) — передовая технологическая дисциплина, класс методов искусственного интеллекта, особенностью которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме. Основная задача машинного обучения – восстановление заранее неизвестной зависимости по выборке, составленной из пар «вход-выход» [41,42].

Цель машинного обучения — предсказать результат по входным данным. Чем разнообразнее входные данные, тем проще машине найти закономерности и тем точнее результат. На основе данных требуется построить неявную зависимость, т.е. построить алгоритм, способный для любого сочетания входных данных выдать квалифицирующий ответ. Реализуется принцип не аналитического, а эмпирического формирования решения. Для измерения точности ответов вводится оценочный функционал качества. На рис. 2.16. представлена общая схема процесса машинного обучения.

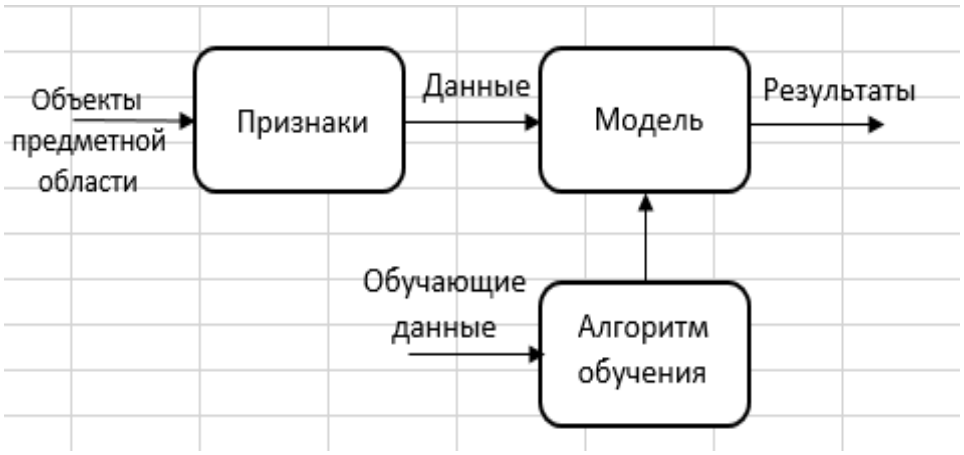


Рис.2.16. Схема машинного обучения

На вход системы поступают параметры объекта исследуемой предметной области. На первом этапе из набора параметров выделяются признаки, которые затем в качестве исходных данных используются моделью обработки. Модель работает на базе алгоритма обучения, который построен на обучающих данных. Так как цель машинного обучения — предсказать результат по входным данным, то чем разнообразнее входные данные, тем проще машине найти закономерности и тем точнее результат. При обучении компьютера нужны три вещи: данные, признаки и алгоритм обработки.

Входные данные являются набором параметров, характеризующих исследуемый объект. Если это изображение – то параметром будет набор пикселей, если это сигнал, то параметрами являются значения дискретных отсчетов во времени. Это могут быть статистические показатели в виде временных рядов. Данные могут быть представлены в виде цифровых показателей возраста, геометрических размеров тел, цвета, спектра.

В результате предварительного анализа из измеренных или

считанных входных данных формируются признаки или наборы признаков исследуемого объекта. Признак – это некоторое количественное измерение, характеристика объекта произвольной природы. Совокупность признаков, относящихся к одному образу, называется вектором признаков. Считается, что каждому образу ставится в соответствие единственное значение вектора признаков и наоборот: каждому значению вектора признаков соответствует единственный образ объекта.

На этапе предварительного анализа реализуется генерация и селекция признаков: выбор тех признаков, которые с достаточной полнотой описывают объект исследования (генерация), отбор наиболее информативных признаков (селекция). Наиболее информативные признаки объекта играют роль входных данных для математической модели обработки.

Модель – это численный метод, в результате работы которого происходит отображение признаков на результат. Математическая модель содержит переменные в виде входных данных и весовых регулируемых коэффициентов (обучающие данные). Алгоритмы машинного обучения можно описать как обучение целевой функции f, которая наилучшим образом соотносит входные переменные X и выходную переменную Y, т.е. реализует функцию Y = f(X). Мы не знаем, что из себя представляет функция f, если бы знали, то использовали бы её напрямую, а не пытались обучить с помощью различных алгоритмов. В этом и есть суть эмпирически формируемого решения с использованием обучающих данных. Как было показано ранее, для оценки точности ответов вводится установленный функционал качества.

В реальных прикладных задачах входные данные об объектах могут быть неполными, неточными, нечисловыми, разнородными. Эти особенности приводят к большому разнообразию моделей и методов машинного обучения.

Основные методы машинного обучения

К настоящему времени разработано большое количество методов и алгоритмов машинного обучения. Разнообразие объектов моделирования и управления, вероятностные характеристики и количественное разнообразие входных данных способствовали разработке самых разных по сложности алгоритмов обучения. Такое же разнообразие наблюдается и при классификации методов. Мы

рассмотрим наиболее широко применимые, относительно простые для понимания и успешно реализуемые с помощью нейронных сетей классические алгоритмы машинного обучения (рис.2.17). Из них наиболее широко применяемыми являются: классификация, регрессия и кластеризация [41,43].

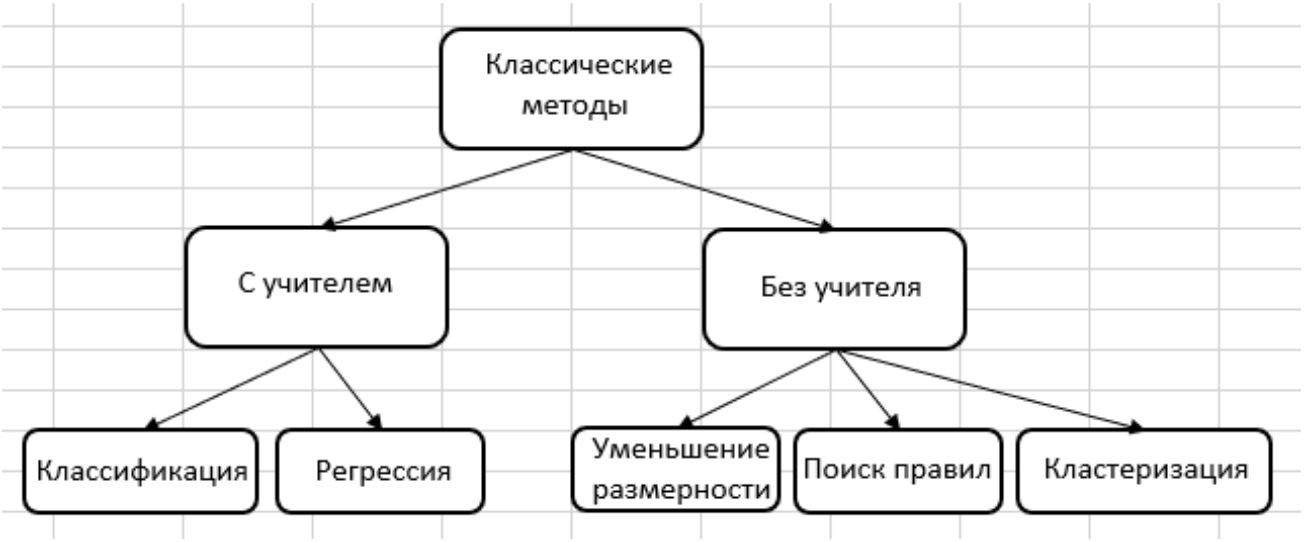


Рис.2.17. Методы машинного обучения

Если выход представляет вещественную переменную, то задача называется задачей восстановления – регрессией.

Если выход принимает конечное число значений – это задача классификации.

Если входы и выходы – значения величин в некоторые моменты времени – это задача прогнозирования.

Приложения задач машинного обучения:

- распознавание образов;
- интеллектуальный анализ данных (Data Mining);
- обработка естественных языков;
- компьютерное зрение, робототехника;
- медицинская диагностика;

Задачи классификации объектов

При исследовании широкого типа задач считается, что все объекты или явления разбиты на конечное число классов [24,26,30]. Для каждого класса известно и изучено конечное число объектов – прецедентов. Прецедент – это образ ранее классифицированного объекта, принимаемый как образец при решении задач классификации. Идея принятия решений на основе прецедентности – основополагающая в естественно-научном мировоззрении. При такой

постановке задачи каждому образу объекта ставится в соответствие единственное значение вектора признаков и наоборот: каждому значению вектора признаков соответствует единственный образ исследуемого объекта. Классификатором или решающим правилом называется правило отнесения образа к одному из классов на основании его вектора признаков.

В зависимости от наличия или отсутствия прецедентной информации различают задачи классификации с обучением и без обучения. Задача классификации на основе имеющегося множества прецедентов называется классификацией с обучением (или с учителем). В том случае, если имеется множество векторов признаков, полученных для некоторого набора образов, но правильная классификация этих образов неизвестна, возникает задача разделения этих образов на классы по сходству соответствующих векторов признаков. Эта задача называется кластеризацией или распознаванием без обучения (без учителя).

Очевидно, что с учителем машина обучится быстрее и точнее, потому в реальных задачах его используют намного чаще. Эти задачи делятся на два типа: классификация — предсказание категории объекта, и регрессия — предсказание места на числовой прямой.

При классификации алгоритм разделяет объекты по заранее известному признаку. Примером может быть распознавание рукописных цифр или букв по клеткам изображения.

Для реализации процедур классификации в зависимости от характера решаемой задачи использовали известный из теории вероятностей «алгоритм Байеса» или алгоритм «дерево решений», где компьютер автоматически разделяет все данные по вопросам, ответы на которые «да» или «нет». Сегодня для классификации всё чаще используют нейронные сети, первые разработки которых были ориентированы на решение задач классификации. Преимущество нейронных сетей заключается в том, что они предполагают наличие правил, с помощью которых сеть может программироваться автоматически.

Качество работы нейронной сети сильно зависит от вводимого в процессе обучения набора учебных данных, поэтому в начале обучения весовые коэффициенты устанавливаются равными случайным малым значениям. Расхождение между входными и выходными, требуемыми данными, есть величина ошибки, которая

может использоваться для корректировки весовых коэффициентов.

Среди задач обучения математические методы классификации включают несколько самостоятельных моделей. Рассмотрим наиболее широко применяемые методы.

Метод k-ближайших соседей.

Сознание человека при изучении объектов окружающего мира, сопоставляет их с уже известными объектами и оценивает степень их сходства. На основе это оценки объект относится к определённой группе, классу объектов. Классификация является наиболее естественным для человеческого интеллекта способом получения знаний о процессах и явлениях.

Представителем методов классификации, является метод «k-ближайших соседей» (k-nearest neighbors algorithm - KNN).

При реализации данного метода предполагается, что уже имеется какое- то количество объектов, для которых известно, какому классу они принадлежит. Нужно выработать правило, позволяющее отнести новый объект к одному из возможных классов, классы известны. В основе k-NN лежит правило: объект считается принадлежащим тому классу, к которому относится большинство его ближайших соседей. Под «соседями» здесь понимаются объекты, близкие к исследуемому в установленном смысле.

При решении данной задачи необходимо уметь определять, насколько объекты близки друг к другу, т.е. уметь измерять «расстояние» между объектами. Это не обязательно евклидово расстояние. Это может быть мера близости объектов, например, по цвету, форме, вкусу, запаху, весу, то есть по тем параметрам, которые характеризуют эти объекты. Следовательно, для применения метода k-NN в пространстве признаков объектов должна быть введена некоторая метрика, т.е. функция расстояния. Предполагается, что объекты с близкими значениями признаков относятся к одному классу.

Предсказание для новой точки делается путём поиска k ближайших соседей в наборе данных и суммирования выходной переменной для этих k экземпляров (рис.2.18).

На рисунке изображено пространство признаков двух классов (белые и черные кружки), неизвестный объект для распознавания изображен в виде круга с утолщенной окружностью.

Согласно методу k-NN неизвестная модель будет отнесена к тому классу, к которому принадлежит большинство из k - ближайших

соседей. Расстояние между объектами будет измеряться по Евклидовой норме, т.е. расстояние между объектами с координатами $(x1, y1)$ и $(x2, y2)$ равно:

$$L = \sqrt{((x_1 - y_1) + (x_2 - y_2))^2}.$$

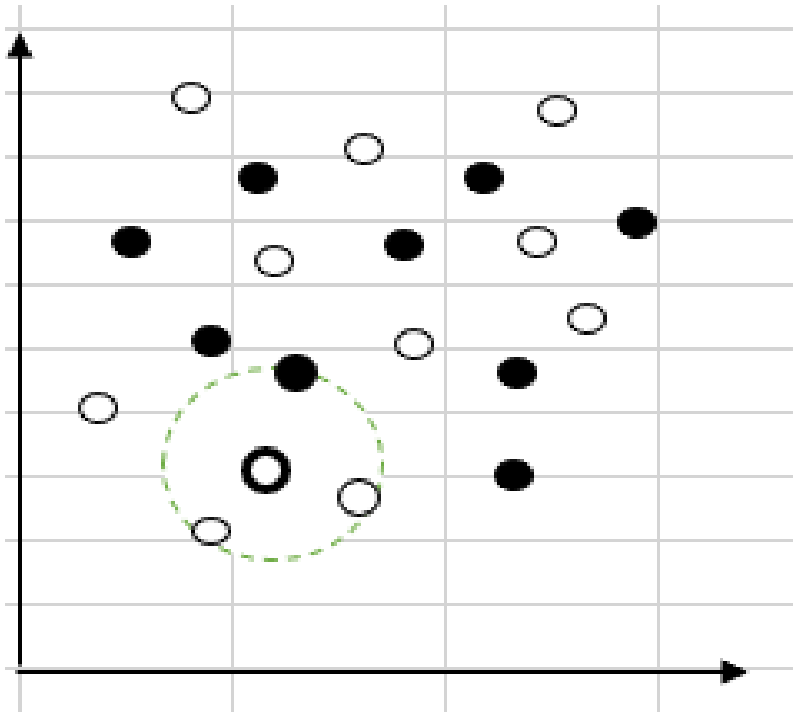


Рис.2.18. Иллюстрация задачи классификации k-nearest

Здесь значения x и y соответствуют координатам скорости и веса. По приведенной формуле рассчитываются все расстояния от неизвестной модели до всех приведенных на рисунке автомобилей.

Очевидно результат, получаемый методом k-NN, сильно зависит от выбора параметра k. При малых значениях k результат может быть случайным, при больших значениях k результат будет зависеть числа автомобилей в той или иной группе. Обычно k берется равным квадратному корню из общего числа автомобилей. В приведенном примере k взято равным 4 и ответом будет
«легковой автомобиль».

Решение задачи классификации упрощается, если классы признаков имеют четко разделенные области представления. Задача заключается в выработке правил разделения областей, т.е. вывести функцию классификации путем построения линии, разделяющей эти два класса.

Бинарная классификация методом опорных векторов.

В данном случае решается задача бинарной (когда класса всего два) классификации. Сначала алгоритм тренируется на объектах из обучающей выборки, для которых заранее известны метки (признаки) классов. Далее уже обученный алгоритм предсказывает признак класса для каждого объекта из тестовой выборки. Метки классов могут принимать значения Y= (+1,-1). Объектом является вектор с N признаками x=(x1, x2, x3,..., xn). При обучении алгоритм должен построить функцию F(x)=y, которая принимает аргумент x (объект из пространства) и выдает метку класса y.

Данная задача относится к обучению с учителем, применяется алгоритм метода опорных векторов (SVM – Support Vector Machines). Данный алгоритм может применяться и для задач регрессии, но в данном случае он выполняет функции классификатора [32].

Теорема Байеса (метод Maive Bayes).

Теорема Байеса – один из законов теории вероятности [28]. Она помогает нам пересматривать и понимать вероятности, когда возникают новые свидетельства. То есть, теорема дает количественную оценку того, насколько должны измениться наши убеждения в связи со вновь обнаруженными доказательствами.

Прежде чем рассмотреть реализацию алгоритма по формуле Байеса, исследуем простой способ представить саму идею метода без формул, т.к. применение теоремы не интуитивно, по крайней мере, для большинства людей. Визуализация использования теоремы намного упрощает понимание сути идеи Байеса и способствует применению ее в жизни.

Для упрощения графического представления задачи и ее решения опытные объекты лучше представить в виде квадратных фигур с неким содержимым.

Пусть имеется несколько возможностей, которые одинаково вероятны, их удобно представить в виде прямоугольника, разделенного на равные части (рис.2.19). Возможные события и их вероятности представлены в виде фигур.

Рис.2.19. Возможности событий и их вероятности

Представим, что каждая возможность – это коробка с 10 конфетами. В коробке А - 10 шоколадных конфет. Чтобы продемонстрировать это, мы заштрихуем ее. В коробке В – 5 шоколадных конфет и 5 ирисок. Мы разделим коробку пополам, и заштрихуем только часть с шоколадными конфетами.

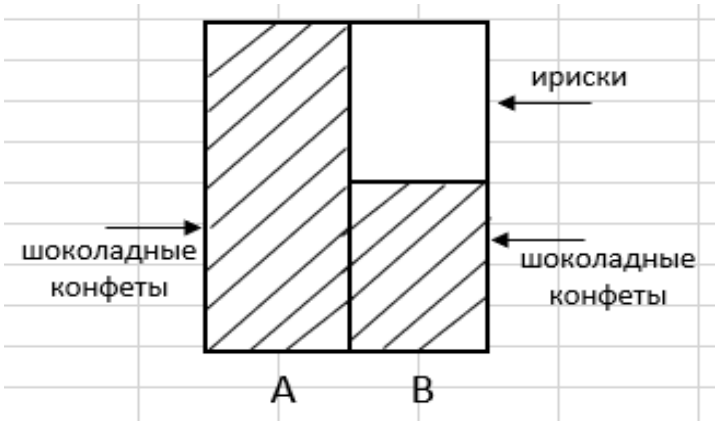


Рис.2.20. Две коробки с разными типами конфет

Из графического представления видно, что заштрихованные области представляют все шоколадные конфеты в обеих коробках, в то время как белая область представляет все ириски. Представим себе, что содержимое обеих коробок перемешано, наугад вытащена одна конфета и она оказалась шоколадной. Если бы была необходимость угадать, из какой коробки вытащена шоколадная конфета, большинство выбирало бы коробку А, так как в коробке А их в два раза больше! Это результат оценки мозга человека. Это пример очень простого, естественного использования теоремы Байеса. В результате выборки шоколадной конфеты исчезла вероятность

выбрать ириску. Чтобы визуализировать это, удалим часть коробки, которая относилась к ирискам (рис.2.21).

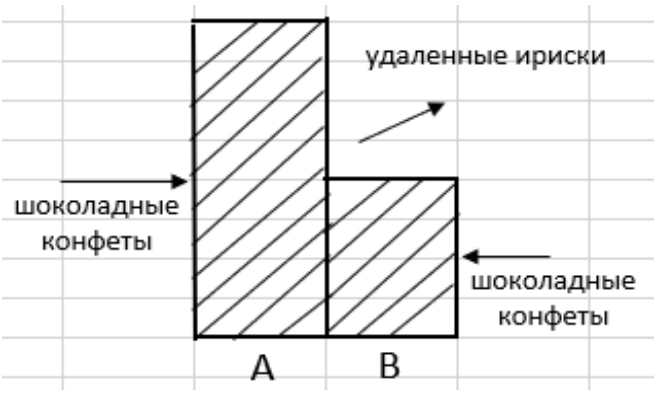


Рис.2.21. Части объемов коробок с шоколадными конфетами

Теперь в коробке А вдвое больше конфет, чем в коробке В. Если разбить коробки на равные секции, образуется 3 равные области: 2 секции в коробке А и 1 секция в коробке В (рис.3.22).

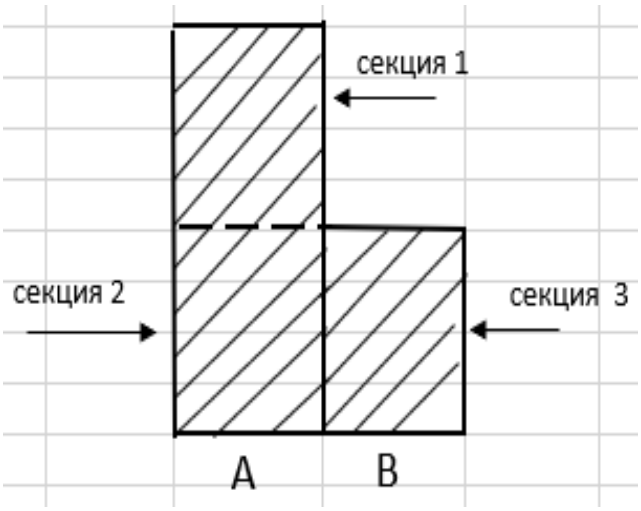


Рис.2.22. Три равные секции в двух коробках

При произвольном извлечении шоколадная конфета из коробки В может быть выбрана с вероятностью 1/3, или 33%. Конфета из коробки А может быть выбрана с вероятностью 2/3, или 66%. Эта разница в вероятности – то, что человеческий мозг примерно рассчитал раньше. Именно это – причина, по которой была выбрана коробка А. Только что была продемонстрировали концепция теоремы Байеса и решена проблема без использования формулы. Теперь, прежде чем решать

эту же проблему с помощью формулы, введем определение теоремы Байеса.

Формула содержит три компонента, комбинирование которыми позволяет решать задачи:

$$P(A/B) = P(B/A) * P(A) / P(B).$$

Обозначения:

- P – вероятность наступления события, знак - условная вероятность;
- A, B – сами события;
- P(A), P(B) – вероятности независимых и не влияющих друг на друга событий;
- P(A/B) – условная вероятность того, что событие A истинно, если истинно событие B;
- P(B/A) – условная вероятность того, что событие B истинно, если истинно событие A.

Использование формулы Байеса предполагает, что нужно определить все три компонента формулы, включить их в формулу и получить обновленную вероятность, основанную на основе новой информации. Искомый ответ будет P(A|B), который называется апостериорной вероятностью и является нормализованным взвешенным значением.

Во вступлении задача с конфетами решалась без использования формулы. Сейчас мы используем Байеса и помня, что всего в обеих коробках 20 конфет, из них 15 шоколадных (рис. 6.9).

Используем четыре шага, чтобы найти ответ с помощью формулы Байеса. Шаг 1. Для начала мы должны определить, что мы хотим найти. Мы хотим знать вероятность того, что выбрали коробку A с учетом того, что мы выбрали

шоколадную конфету.

Шаг 2. Запишем поиск решения в виде формулы для данной задачи: P(кор.A шок.конф) = P(шок.конф кор.A) / P(шок.конф)

Шаг 3. Найдем каждый компонент.

P (коробка A) = 0,5. Чтобы ответить на этот вопрос, необходимо определить какова вероятность выбрать из коробки A. Эта вероятность не зависит от других событий. Поскольку существует только две коробки – A и B, и вероятности их выбора равны, ответ 0,5.

P (шок. конфета) = 0,75. Для ответа на этот вопрос необходимо определить какова вероятность того, что мы выберем шоколадную

конфету. Эта вероятность не зависит от других событий. Всего в обеих коробках 20 конфет, из них 15 шоколадных (рис. 3.20). Итак, 15/20 = 0,75.

P (шок. конфета | коробка А) = 1. Чтобы ответить на этот вопрос необходимо определить какова вероятность выбора шоколадной конфеты, учитывая, что мы выбрали из коробки А. Поскольку в коробке А есть только шоколадные конфеты, вероятность равна 1. Подставим все компоненты в формулу:

P(кор.А шок.конф) = 1* 0,5 / 0,75 =0,66= 0,66%

Итак, существует вероятность 66%, что мы выбрали из коробки А, если выбранная конфета была шоколадной.

Автор знаменитой формулы Томас Байес родился в Лондоне в 1702 году. Он изучал логику и теологию в Эдинбургском университете, любил математику и увлекался теорией вероятности. Именно в 1740-х годах сформулировал решение, которое можно понимать следующим образом: первоначальное мнение + новое доказательство = новое суждение. В 1774 году французский математик Пьер-Симон Лаплас повторил решение Байеса. В то время он ничего не знал об открытии Байеса и пришел к нему независимо.

Во время Второй Мировой войны формула была использована Аланом Тьюрингом, чтобы взломать немецкий шифрокод, но эта информация была засекречена в течение длительного времени. Только в 1980-х годах с появлением персональных компьютеров теорема Байеса получила широкое признание. Сейчас она широко используется во многих отраслях и касается нашей повседневной жизни.

Логистическая регрессия (Logistic Regression).

В математической статистике логистическая регрессия является широко применяемой статистической моделью. Регрессия в общем виде применяется, когда входные и выходная переменные непрерывные. А логистическая регрессия лучшим образом подходит, когда выходная переменная принимает только два значения.

В отличие от множественной регрессии, которая игнорирует ограничения на диапазон значений выходной переменной, задача логистической регрессии может быть сформулирована иначе: вместо предсказания бинарной переменной мы предсказываем непрерывную переменную со значениями на отрезке [0,1] при любых значениях независимых переменных. Это достигается применением следующего регрессионного уравнения:

$$P = 1 / 1+e-y$$

где p — вероятность того, что произойдет интересующее событие; e — основание натуральных логарифмов 2,71…; y — стандартное уравнение регрессии:

$$y = W0 + W1x1 + W2x2 + \ldots + WnXn$$

Это уравнение позволяет комплектовать все n признаков на одну ось абсцисс и вычислять значение признака.

При построении графической зависимости регрессионного уравнения учитываются особенности сигмоиды: при y=0 значение e-y обращается в 1, величина P становится равной 0,5. Кроме того, результаты подсчета коэффициентов влияют на характер кривизны сигмоиды. Массив X – параметры объекта наблюдения, W – это массив коэффициентов. Путем побора коэффициентов выбирается характер кривой, наилучшим образом описывающей конкретные данные машинного обучения.

Зависимость, связывающая вероятность события и величину X, показана на следующем графике:

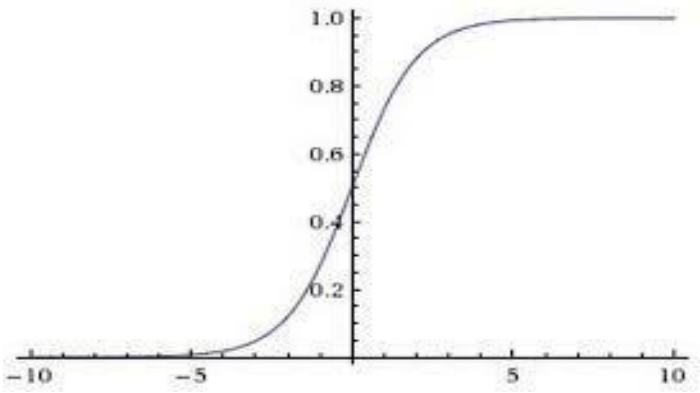


Рис.2.23. График зависимости между вероятностью и переменной

Сигмоида, представленная на рис.2.23, берет какое-то число от + ∞ до - ∞ и переводит его в число от 0 до 1.

Данная модель преобразования решает задачу классификации. В теории машинного обучения (Machine Learning – ML) если выход алгоритма преобразования представляет вещественную переменную, то задача называется задачей восстановления - регрессией. Если же выход принимает конечное число значений – это задача классификации. В задачах такого вида множество ответов является

конечным, каждый ответ выражения y = f (WX) соответствует некоторому классу объектов, и задача ML заключается в вычислении по каждому объекту соответствующего ему класса.

Алгоритм «дерево решений».

Деревья решений являются одним из наиболее эффективных инструментов интеллектуального анализа данных, которые позволяют решать задачи классификации и регрессии. Поскольку правила в деревьях решений получаются путём обобщения множества отдельных наблюдений (обучающих примеров), описывающих предметную область, то по аналогии с соответствующим методом логического вывода их называют индуктивными правилами, а сам процесс обучения — индукцией деревьев решений. Метод относится к категории обучения с учителем [41].

Дерево решений — это простой метод представления решающих правил в иерархической структуре, состоящей из элементов двух типов — узлов (node) и листьев (leaf). В узлах находятся решающие правила и производится проверка соответствия примеров этому правилу по какому- либо атрибуту обучающего множества.

В простейшем случае, в результате проверки, множество примеров, попавших в узел, разбивается на два подмножества, в одно из которых попадают примеры, удовлетворяющие правилу, а в другое — не удовлетворяющие. Затем к каждому подмножеству вновь применяется правило и процедура рекурсивно повторяется пока не будет достигнуто некоторое условие остановки алгоритма. В результате в последнем узле проверка и разбиение не производится и он объявляется листом. Лист определяет решение для каждого попавшего в него примера. Для дерева классификации — это класс, ассоциируемый с узлом, а для дерева регрессии соответствующий листу модальный интервал целевой переменной. Таким образом, в отличие от узла, в листе содержится не правило, а подмножество объектов, удовлетворяющих всем правилам ветви, которая заканчивается данным листом.

Основная сфера применения деревьев решений — поддержка процессов принятия управленческих решений, используемая в статистике, анализе данных и машинном обучении. Задачами, решаемыми с помощью данного аппарата, являются:

- классификация — отнесение объектов к одному из заранее

известных классов, при этом целевая переменная должна иметь дискретные значения;

- регрессия — предсказание числового значения независимой переменной для заданного входного вектора.
- описание объектов — набор правил в дереве решений позволяет компактно описывать объекты.

Процесс построения деревьев решений заключается в последовательном, разбиении обучающего множества на подмножества с применением решающих правил в узлах. Процесс разбиения продолжается до тех пор, пока все узлы в конце всех ветвей не будут объявлены листьями. Объявление узла листом может произойти естественным образом (когда он будет содержать единственный объект, или объекты только одного класса), или по достижении некоторого условия остановки, задаваемого пользователем (минимально допустимое число примеров в узле или максимальная глубина дерева).

При формировании правила для разбиения в очередном узле дерева необходимо выбрать атрибут, по которому это будет сделано. Общее правило для этого можно сформулировать следующим образом: выбранный атрибут должен разбить множество наблюдений в узле так, чтобы результирующие подмножества содержали примеры с одинаковыми метками класса, или были максимально приближены к этому, т.е. количество объектов из других классов («примесей») в каждом из этих множеств было как можно меньше.

Как было отмечено выше, если «рост» дерева не ограничить, то в результате будет построено сложное дерево с большим числом узлов и листьев. Как следствие оно будет трудно интерпретируемым. В то же время решающие правила в таких деревьях, создающие узлы, в которые попадают два-три примера, оказываются малозначимыми с практической точки зрения.

Гораздо предпочтительнее иметь дерево, состоящее из малого количества узлов, которым бы соответствовало большое число примеров из обучающей выборки. Поэтому представляет интерес подход, альтернативный ранней остановке — построить все возможные деревья и выбрать то из них, которое при разумной глубине обеспечивает приемлемый уровень ошибки распознавания, т.е. найти наиболее выгодный баланс между сложностью и точностью дерева.

Рассмотрим основные достоинства алгоритма:

- относительно быстрый процесс обучения;

- извлечение правил на естественном языке; -высокая точность предсказания,

- построение непараметрических моделей.

В силу этих и многих других причин, деревья решений являются важным инструментом в работе каждого специалиста, занимающегося анализом данных. Основные области применения: банковское дело (оценка кредитоспособности клиентов), оценка качества продукции и выявление дефектов, диагностика заболеваний.

Пример реализации.

Рассмотрим, как законы энтропии используются при построении дерева решений. Это очень важное понятие, используемое в физике, теории информации и других областях. Опуская предпосылки введения этого понятия, отметим, что, интуитивно, энтропия соответствует степени хаоса в системе. Чем выше энтропия, тем менее упорядочена система и наоборот.

Для иллюстрации того, как энтропия поможет определить хорошие признаки для построения дерева, приведем пример с черными и белыми шариками. Будем предсказывать цвет шарика по его координате, это позволит показать, как энтропия используется для построения дерева решений (рис.2.24).

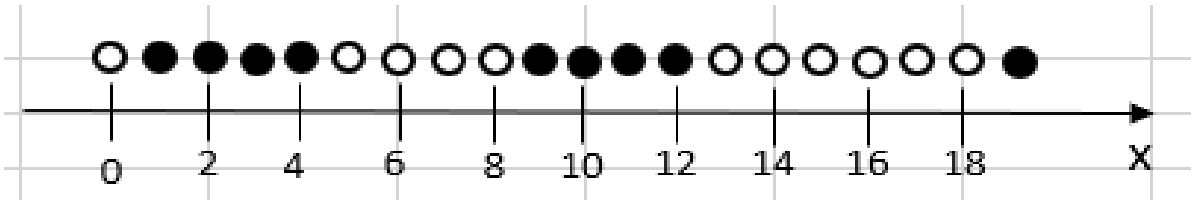


Рис.2.24. Произвольное расположение черных и белых шариков

На рисунке 9 черных шариков и 11 белых шариков. Если наудачу вытащить шарик, то он с вероятностью p1=9/20 будет черным и с вероятностью p2=11/20 будет белым. Значит, энтропия состояния:

$$S = - 9/20*\log 9/20 – 11/20*\log 11/20 \approx 1.$$

Само значение «1» ни о чем не говорит. Посмотрим, как изменится энтропия если разбить шарики на две группы – с координатой равной или меньше 12 (рис.2.25).

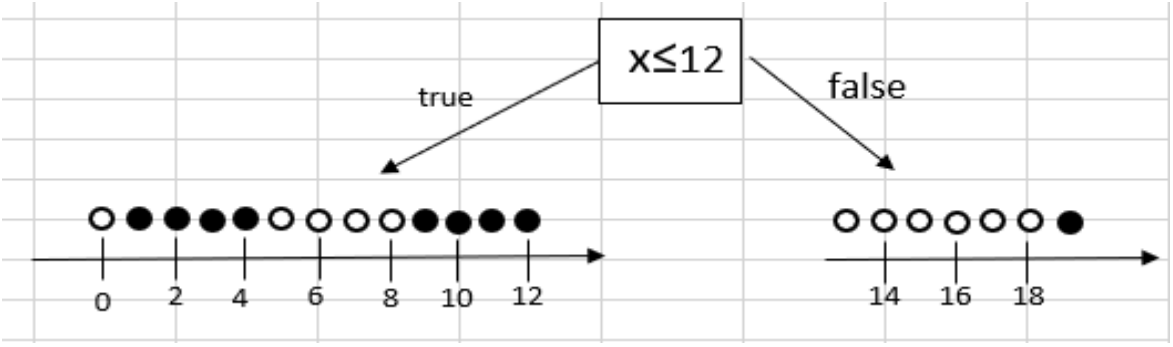


Рис.2.25. Разделение шариков на две группы

В левой группе оказалась 13 шаров, из которых 8 черных и 5 белых.

Энтропия этой группы равна:

$$S1 = -5/13*\log 5/13 – 8/13*\log 8/13 \approx 0{,}96.$$

В правой группе оказалось 7 шаров, из которых 1 черный и 6 белых.

Энтропия правой группы равна:

$$S2 = - 1/7*\log 1/7 – 6/7*\log 6/7 \approx 0{,}6.$$

Как видно из формул, энтропия уменьшилась в обеих группах, особенно во второй. Поскольку энтропия – это степень хаоса (неопределенности) в системе, уменьшение энтропии называют приростом информации.

В данном случае после разделения получилось две группы (q=2) – одна из 13 элементов (N1=13), вторая группа – из 7 элементов (N2=7). Прирост информации получился:

$$IG(x\leq 12) = S0 – 13/20*S1 – 7/20*S2 \approx 0{,}16$$

Получается, что разделив шарики на две группы по признаку «координата меньше либо равна 12» получилась более упорядоченная система, чем в начале. Продолжим деление шариков на группы до тех пор, пока в каждой группе шарики не будут одного цвета (рис.2.26).

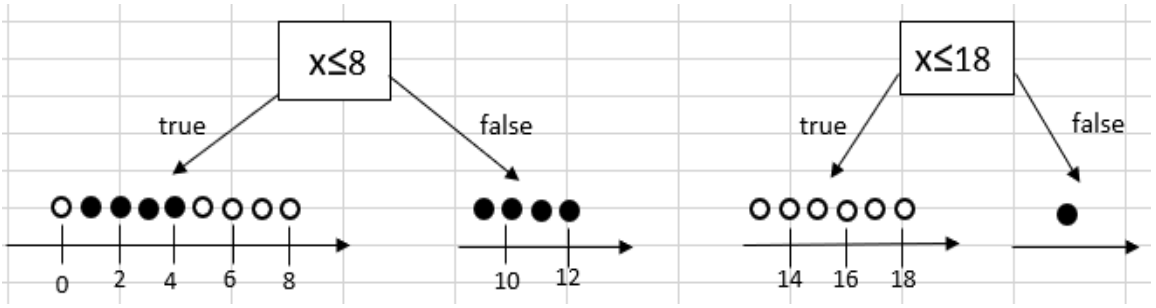


Рис. 2.26. Второй шаг разбиения на группы

На следующем шаге разбиения для правой группы (рис.6.14) потребовалось всего одно дополнительное разбиение по признаку «координата меньше либо равна18». Очевидно, энтропия правой группы с шариками одного цвета равна 0 (т.к. log 1=0). Другими словами, группа шариков одного цвета является упорядоченной.

Для левой группы на рис.2.26 процесс разбиения не закончен, т.к. нет четкого разбиения на две группы с шариками одного цвета, поэтому эта группа подлежит дальнейшему разбиению.

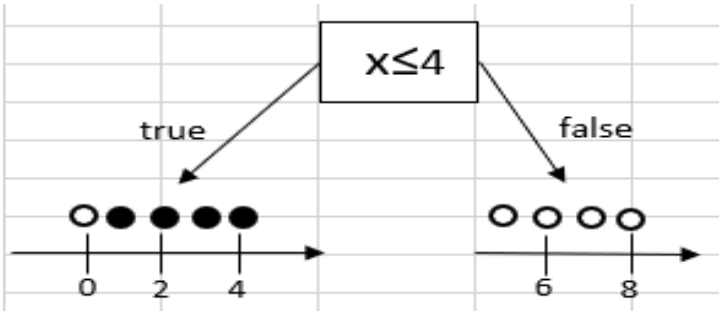


Рис.2.27. Третий шаг разбиения на группы

На третьем шаге (рис.2.28) однозначного решения не достигнуто, требуется еще один шаг для получения групп с шариками одного цвета.

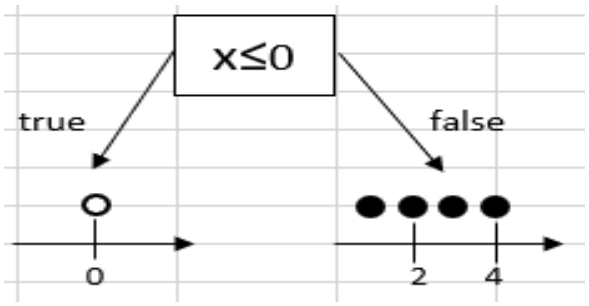


Рис.2.28. Заключительный шаг разбиения

В итоге разбиения на группы удалось построить дерево решений, предсказывающее цвет шарика по его координате. Итоговое изображение дерева решений выглядит следующим образом.

Таким образом, дерево решения задачи было построено в результате выполнения четырех шагов разбиения на группы с целью получения групп с шариками одного цвета, т.е. имеющими нулевую

энтропию. Данное решение идеально подстроилось под обучающую выборку в 20 шариков, при этом потребовалось всего пять шагов алгоритма. При других исходных данных дерево решений может иметь другую конфигурацию (шире и глубже) с другим числом шагов.
Линейный дискриминантный анализ.

Логистическая регрессия используется, когда нужно отнести образец к одному из двух классов. Если классов больше, чем два, то лучше использовать алгоритм линейного дискриминантного анализа LDA.

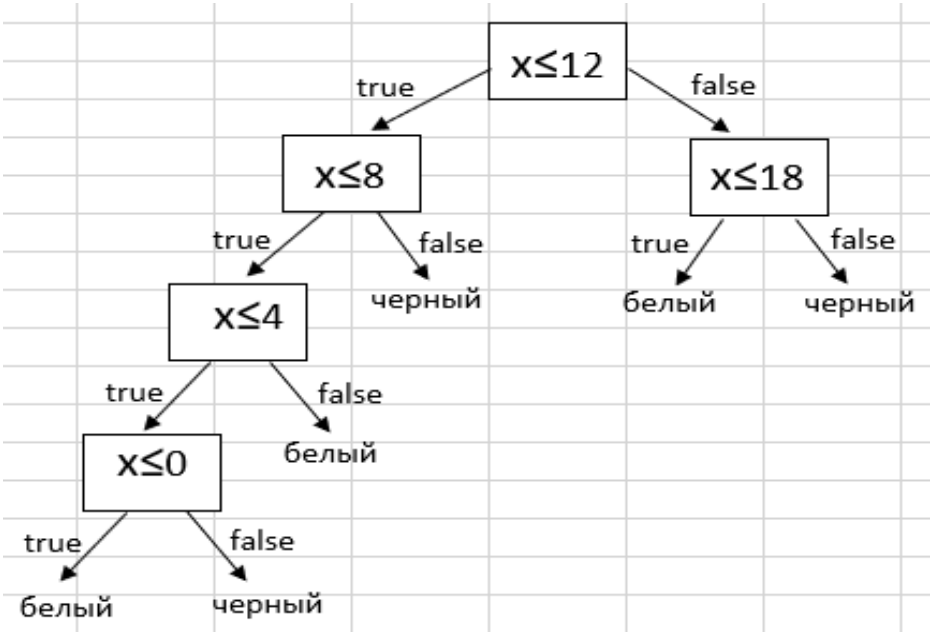


Рис.2.29. Схема алгоритма «дерево решения»

Представление LDA довольно простое. Оно состоит из статистических свойств данных, рассчитанных для каждого класса. Для каждой входной переменной это включает:

- среднее значение для каждого класса;
- дисперсию, рассчитанную по всем классам

Предсказания производятся путём вычисления дискриминантного значения для каждого класса и выбора класса с наибольшим значением. Предполагается, что данные имеют нормальное распределение, поэтому перед началом работы рекомендуется удалить из данных аномальные значения. Это простой и эффективный алгоритм для задач классификации.

Задачи регресии

Регрессия. Иногда естественно вообще отказаться от дискретных классов, а предсказывать некое вещественное число. Например, если требуется оценить срочность входящего параметра по некоторой шкале. Такая задача называется регрессией.

Если классификация — предсказание категории объекта, то регрессия — предсказание места на числовой прямой. Суть регрессии состоит в том, чтобы построить вещественную функцию на основании обучающих примеров, помеченных истинными значениями функции. Обычно для решения этой задачи выбирают некий класс функций (скажем, функции, линейно зависящие от каких-то числовых признаков) и строят функцию из этого класса, которая минимизирует разность между предсказанными и истинными значениями. Задача, в которой строится функция f^ в результате обучения на примерах, помеченных истинными значениями функции (x, f(x)), называется регрессией. Когда регрессия функцию как прямую линию, её называют линейной, когда как кривую — полиномиальной. Это два основных вида регрессии. В математической статистике линейная регрессия есть метод аппроксимации зависимостей между входными и выходными переменными не основе линейной модели. Она является частью более широкой статистической методики, называемой регрессионным анализом.

Линейная регрессия.

Если рассматривается зависимость между одной входной и одной выходной переменными, то имеет место простая линейная регрессия. Для этого определяется уравнение регрессии y = ax + b и строится прямая, известная как линия регрессии. Коэффициенты a и b, называемые также параметрами модели, определяются таким образом, чтобы сумма квадратов отклонений точек, соответствующих реальным наблюдениям данных, от линии регрессии была бы минимальной. Коэффициенты обычно оцениваются методом наименьших квадратов. Если ищется зависимость между несколькими входными и одной выходной переменными, то имеет место множественная линейная регрессия. Соответствующее уравнение имеет вид: y = b0 + b1x1 + b2x2 +⋯+ bnxn, где n — число входных переменных. Очевидно, что в данном случае модель будет описываться не прямой, а гиперплоскостью. Коэффициенты уравнения множественной линейной регрессии подбираются так, чтобы минимизировать сумму

квадратов отклонения реальных точек данных от этой гиперплоскости.

Линейная регрессия была первым видом регрессионного анализа, который был хорошо изучен и начал широко использоваться в реальных приложениях. Это связано с тем, что в линейных моделях оценивание параметров проще, а также с тем, что статистические свойства полученных оценок легче определить.

В качестве примера реализации алгоритма регрессии рассмотрим относительно простую задачу: вычислить отклонение металлической пластины в зависимости от приложенной к ее середине нагрузки, например гири определенного веса. Необходимо также начертить график отклонения при изменении веса нагрузки.

Из законов физики известно, что при однородном материале металлической пластины зависимость ее прогиба линейно зависит от прикладываемого веса, поэтому график в идеале должен образовать прямую линию. Используя метод наименьших квадратов по измеренным точкам можно построить прямую линию, с помощью которой в дальнейшем можно будет предсказывать величину отклонения для точек, в которых измерения не проводились.

Такие задачи широко применяются при статистических исследованиях, когда необходимо определить объем выпускаемой продукции между периодами официальной статистической отчетности, или предсказать объем выпускаемой продукции в последующие периоды работы.

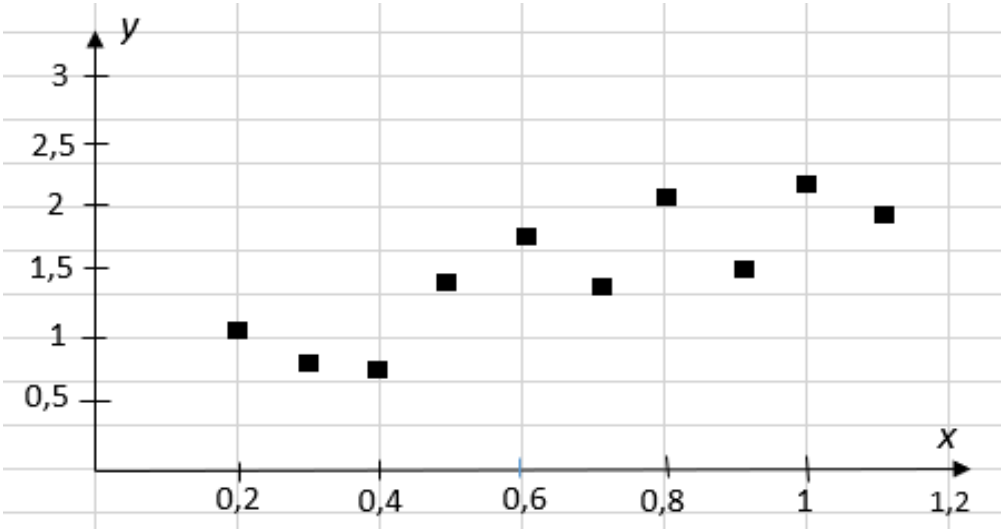


Рис.2.30. Расположение точек на линии с ошибками измерения

Однако, в реальных ситуация всегда присутствуют ошибки измерений, которые изменяют характер прямой линии, появляются случайные ошибки. Это хорошо видно на рис.2.30.

$$y = mx + c$$

где y и x являются переменными (отклонением и нагрузкой), m определяет наклон прямой, а c – определяет значение сдвига (точку пересечения с осью). Метод наименьших квадратов дает возможность вычислить m и c и найти уравнение прямой для всех точек, сумма квадратов ошибок в которых будет наименьшей. Чтобы найти сумму квадратов ошибок следует возвести значение каждой ошибки в квадрат и просуммировать все полученные таким образом значения.

Линейная регрессия — пожалуй, один из наиболее известных и понятных алгоритмов в статистике и машинном обучении. Для регрессии подходят задачи, где есть зависимость от времени, алгоритм регрессии популярен среди финансистов и аналитиков. При полиномиальной регрессии для построения уже нелинейной кривой используется полином более высоких степеней по методу наименьших квадратов.

Логистическая регрессия — этот алгоритм используется для задач бинарной классификации, когда на выходе получается один из двух классов.

Логистическая регрессия похожа на линейную тем, что в ней тоже требуется найти значения коэффициентов для входных переменных, только выходное значение преобразуется с помощью стандартной логистической функции. Логистическая функция «S» и преобразует любое входное значение в число в пределах от 0 до 1 (рис.2.31).

Благодаря тому, как обучается модель на логистической функции, предсказания логистической регрессии можно использовать для отображения вероятности принадлежности образца к классу 0 или 1. Это полезно в тех случаях, когда нужно иметь больше обоснований для прогнозирования. Модель логистической регрессии быстро обучается и хорошо подходит для задач бинарной классификации.

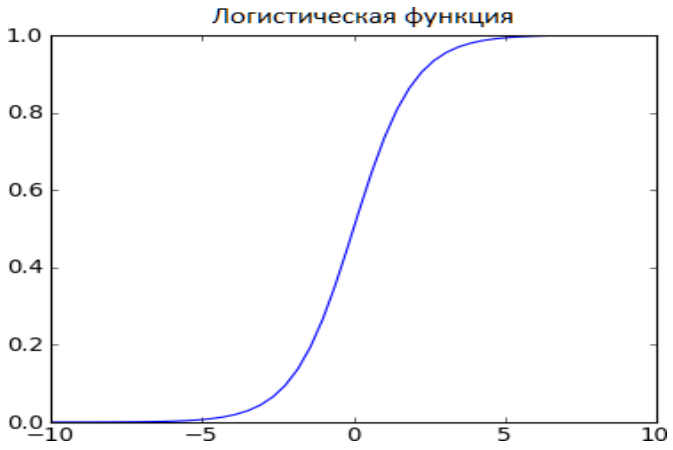


Рис.2.31. Пример логистической функции

Алгоритм обучения без учителя

Кластеризация. Кластеризация (или кластерный анализ) — это задача разбиения множества объектов на группы, называемые кластерами. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно более отличны.

Основными целями кластерного анализа являются: понимание данных путём их обобщения, упрощение дальнейшей обработки, сжатие данных и выделение нетипичных объектов.

Главное отличие кластеризации от классификации состоит в том, что перечень групп четко не задан и определяется в процессе работы алгоритма. Применение кластерного анализа в общем виде сводится к следующим этапам:

- отбор выборки объектов для кластеризации;

определение множества переменных, по которым будут оцениваться объекты в выборке(при необходимости – нормализация значений переменных);

- вычисление значений меры сходства между объектами;, ,
- применение метода кластерного анализа для создания групп сходных объектов (кластеров);
- представление результатов анализа.

Для начала нужно составить вектор характеристик для каждого объекта

— как правило, это набор числовых значений, например, рост-вес человека, габариты товаров, расстояния между объектами, числовые

параметры. В процессе нормализации все значения приводятся к некоторому диапазону, например, [-1, -1] или [0, 1].

Типы входных данных. Каждый объект описывается набором своих характеристик, называемых признаками. Признаки могут быть числовыми или нечисловыми. Признак, по которому собираются предметы в одну группу, а другие в другую, называется метрикой. Грубо говоря, метрика – это просто способ отделить один предмет от другого.

Самый распространненый вариант – это когда метрикой является расстояние. Каждый объект описывается расстояниями до всех остальных объектов обучающей выборки, функцией расстояния является метрика, часто используется евклидова метрика. Евклидово расстояние - наиболее распространенная функция расстояния.

Выбор метрики полностью лежит на исследователе, поскольку результаты кластеризации могут существенно отличаться при использовании различных мер.

Описание алгоритма кластеризации. Для выборки данных, содержащей n записей (объектов), задается число кластеров k , которое должно быть сформировано. Затем алгоритм разбивает все объекты выборки на k разделов ($k < n$), которые и представляют собой кластеры.

2.7. Нейронные сети

Биологический нейрон

Нервная система и мозг человека состоят из нейронов, соединенных между собой нервными волокнами, которые способны передавать электрические импульсы между нейронами. Все процессы передачи раздражений от кожи, ушей и глаз к мозгу, процессы мышления и управления действиями - все это реализовано в живом организме как передача электрических импульсов между нейронами.

Изучение механизмов функционирования отдельных нейронов и их взаимодействия важно для познания протекающих в нервной системе процессов [35]. Это позволяет лучше понимать процессы передачи и обработки информации, происходящие в нейронных сетях. Нейроны, соединенные разнообразными связями в сеть, определяют интеллект, творческие способности и память человека. Количество

нейронов в головном мозге человека больше, чем у всех остальных известных форм жизни.

Нейрон представляет собой особый вид клеток, которые обладают электрической активностью. (рис.2.32). Он получает информацию при помощи сильно разветвленных отростков, называемых дендритами и передает информацию вдоль тонкого волокна – аксона. Аксон имеет множество ответвлений, на конце каждого из которых находится область, называемая синапсом**.** Посредством синапсов осуществляется связь между различными нейронами.

Каждый нейрон может иметь тысячи связей с соседними нейронами. Информация по аксонам передается в виде электрических импульсов, амплитуда которых составляет около 100 мВ, а длительность – 1 мс. На участках контакта между нейронами (синапсы) электрические импульсы превращаются в химические сигналы, которые стимулируют проникновение в клетку нейрона положительных зарядов. Когда достигается критическое значение потенциала, называемое пороговым, в ядре нейрона возникает электрический импульс, распространяемый, как волна, по аксону на следующий нейрон. Вклад одного синапса в установление соответствующего потенциала на выходе нейрона очень маленький. Для возникновения электрического импульса необходимо, чтобы нейрон непрерывно интегрировал множество синаптических входов. Импульсы одинаковой величины, поступающие на входы нервной клетки через различные синапсы, могут возбуждать ее в разной степени.

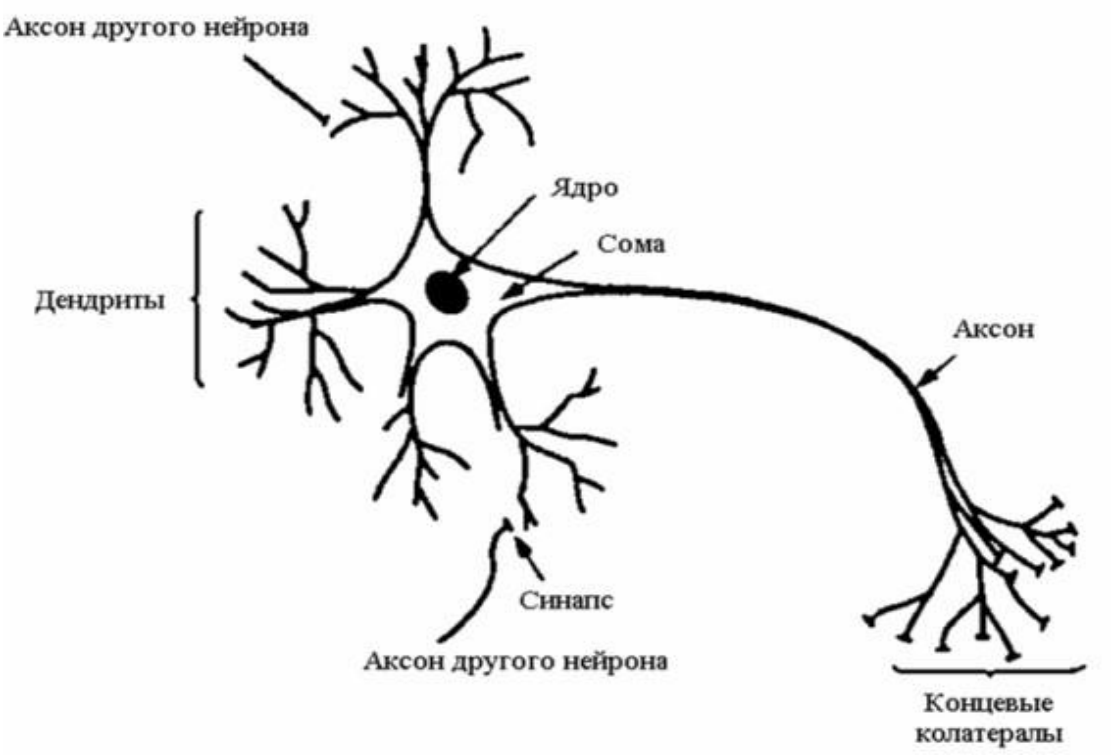


Рис.2.32. Модель биологического нейрона

Мерой возбуждения клетки считается уровень поляризации ее мембраны, зависящий от суммарного количества нейроэнергии, выделенной на всех синапсах [19]. При подходе нервного импульса к синаптическому контакту происходит химическая реакция, следует выброс из аксона биологически активного вещества, которое, через синаптическую щель, достигает клетки, реализуя синаптическую передачу. Передача возбуждений и торможений между нейронами происходит через химические и электрические синапсы.

Дендриты и аксоны вступают в связи с участками мембран других нейронов, образуя сети. Эти сети и служат системами обработки информации. Тело нейрона в данном случае лишь поддерживает функционирование этих участков мембраны.

Каждому входу нейрона соответствуют коэффициенты (веса), пропорциональные количеству нейроэнергии, однократно выделяемой на соответствующем синапсе. В математической модели нейрона входные сигналы должны умножаться на эти коэффициенты для учета влияния каждого сигнала на состояние клетки.

Синапсические веса - натуральные числа, принимающие как положительные, так и отрицательные значения. В первом случае синапс оказывает возбуждающее, а во втором - тормозящее действие, препятствующее возбуждению клетки другими сигналами.

Считается, что человеческий мозг содержит млн. нейронов, каждый из нейронов выполняет относительно примитивные функции суммирования весовых коэффициентов входных сигналов и сравнения суммы с пороговым значением. Каждым нейрон имеет свои веса и свое пороговое значение.

Нейронная организация мозга

В процессе развития мозга непрерывно происходит динамическая перестройка нейронных сетей, при этом формируется сознание. Само сознание – это процесс непрерывного развития и структурной организации нейронов мозга, который происходит под воздействием внутренних и внешних факторов, во времени и в пространстве, при этом формирование мозга происходит под воздействием генетической программы и факторов внешней среды. Каждый индивид имеет неповторимую генетическую информацию и развивается в разных условиях то архитектура мозга формируется у каждого человека по-своему. Это составляет его индивидуальность. Поэтому, несмотря даже на одинаковую генетическую информацию, нельзя полностью воспроизвести архитектуру мозга и интеллект индивида. У различных индивидов это происходит по-разному в зависимости от генетической программы и внешних воздействий, стимулирующих нервную активность мозга. По мере взросления человека общее число нейронов в мозге снижается, однако путем перестройки соответствующих нейронных структур мозг стремится эту утрату компенсировать [19].

Одним из важнейших свойств человеческого мозга является его способность к обучению. Обучение – это процесс непрерывного развития и формирования сознания через взаимодействие с внешней средой и учетом индивидуальности организма. В результате обучения происходит динамическая перестройка нейронных сетей головного мозга, увеличивается число связей между нейронами, совершенствуются взаимодействие между ними. Способность синапсов и нейронных сетей динамически изменяться в результате воздействий называется пластичностью. Механизм пластичности лежит в основе обучения и создает в коре головного мозга соответствующие структуры нейронных сетей, которые

определяют интеллект, память и эмоции индивида.

Пластичность возникает в результате изменения эффективности и количества связей между нейронами. К канадский психолог Д. Хебб сформировал принцип обучения, согласно которому для усиления связи между нейронами необходимо совпадение их активности во времени. Усиление связи приводит к повышению эффективности синаптической передачи между двумя нейронами.

Французские исследователи Л. Тауц и Э. Кендел обнаружили, что если на пресинаптическую (передающую) клетку действует некий третий нейрон, то для усиления синаптической связи между двумя нейронами не обязательна активность постсинаптического (принимающего) нейрона. Этот третий нейрон назвается модулирующим. Если одновременно с пресинаптическим нейроном активен модулирующий нейрон, то происходит усиление связи между пресинаптическим и постсинаптическим нейроном (рис. 2.33).

Данный механизм обучения участвует в образовании условных рефлексов, требует участия сознания и сводится к механическому повторению определенных действий.

Итак, в процессе обучения происходит усиление синаптических связей между соответствующими нейронами головного мозга, вследствие чего возникает кратковременное запоминание информации, которая хранится от нескольких минут до нескольких часов. При долговременном запоминании информации, длящемся месяцами, наблюдается активация и экспрессия генов, синтез соответствующих белков и рост новых связей [41].

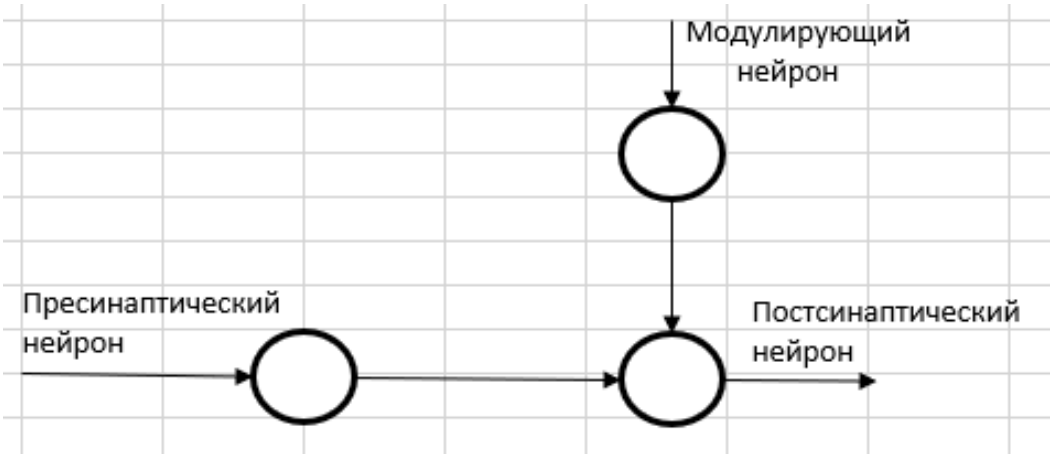


Рис.2.33. Обучение с модулирующим нейроном

Важной характеристикой процесса обучения является обобщающая способность, характеризующая способность мозга интегрировать частные данные для определения закономерностей. К этому относится способность после обучения на одних данных применять полученные знания для других данных или рассуждения от частного к общему. Обобщающая способность – важная черта нейронной организации мозга. Особенно она проявляется в процессе обучения.

В зависимости от вида взаимодействия обучающегося с внешней средой можно условно выделить обучение с учителем и без учителя. Обучение с учителем происходит при взаимодействии индивида-ученика с конкретным индивидом (учителем), с которым он находится в состоянии обратной связи. В процессе взаимодействия реальная реакция ученика сравнивается с целевой функцией - эталонной реакцией учителя. В зависимости от величины их несовпадения (целевая функция ошибки) происходит перестройка синаптических связей в целях минимизации ошибки. При обучении без учителя в качестве учителя выступает внешняя среда, обучение сводится к адаптации индивида к внешней среде.

В обоих типах обучения используются как положительные, так и отрицательные обратные связи в соответствующих нейронных структурах головного мозга. Так, обучение с отрицательной обратной связью происходит для минимизации ошибки целевой функции. Положительная обратная связь может интенсифицировать процесс обучения при успешном взаимодействии индивида со средой.

Искусственный нейрон

Основной строительный блок искусственных нейронных сетей (ИНС) — искусственные нейроны. Считается, что искусственный нейрон имитирует поведение природной нервной клетки мозга [37]. Обобщенная схема i-ого нейрона представлена на рис.2.34.

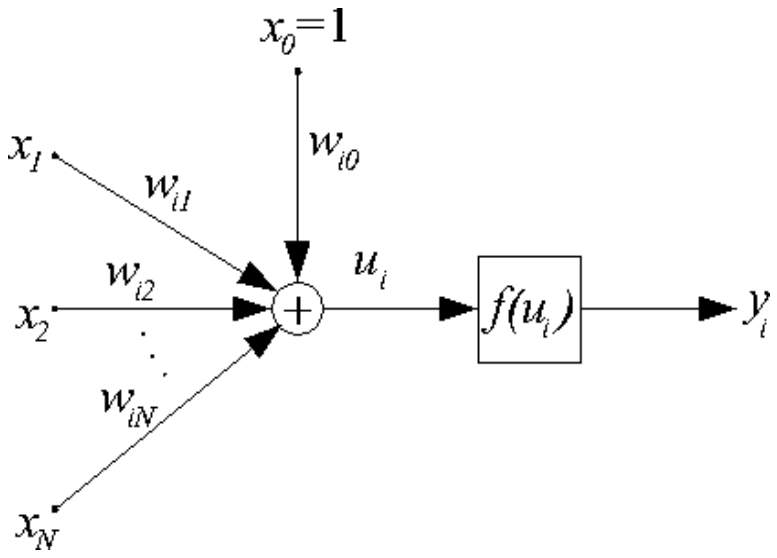


Рис.2.34. Обобщенная структурная схема искусственного нейрона

Искусственный нейрон выполняет операцию нелинейного преобразования суммы произведений входных сигналов на весовые коэффициенты. Оператор нелинейного преобразования f(*ui*) называется функцией активации нейронного элемента. Каждому входу нейрона соответствует весовой коэффициент.
(синапс), который характеризует силу синаптической связи по аналогии с биологическим нейроном. Сумма произведений входных сигналов на весовые коэффициенты называется взвешенной суммой. Она представляет собой скалярное произведение вектора весов на входной вектор.

Рассмотренная модель искусственного нейрона используется для построения искусственных нейронных сетей.

Элементы искусственных нейронных сетей

Для описания алгоритмов и устройств в нейроинформатике выработана специальная "схемотехника", в которой представляются элементарные устройства – различные типы сумматоров, нелинейные преобразователи. Они затем объединяются в нейронные сети, предназначенные для решения различных задач [45].

Простой сумматор (рис.2.35) не имеет настраиваемых параметров. Он имеет векторный вход и выдает на выходе сумму координат вектора входного сигнала *x*.

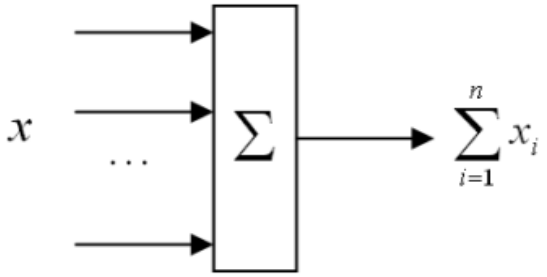


Рис.2.35. Простой сумматор

Важный элемент нейросетей – это адаптивный сумматор.

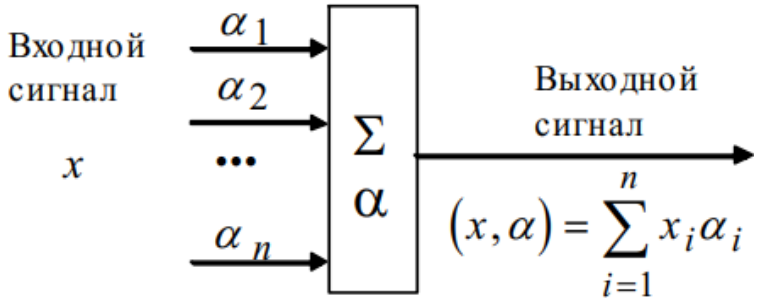


Рис.2.36. Адаптивный сумматор

Для многих задач полезно иметь неоднородную линейную функцию выходных сигналов. Ее вычисление также можно представить с помощью адаптивного сумматора, имеющего n+1 вход и получающего на 0-й вход постоянный единичный сигнал. Здесь главное – подавать на один из входов постоянную. Использование единицы чаще всего удобнее, хотя не обязательно. Сумматор с таким дополнительным входом называют неоднородным адаптивным сумматором (рис. 2.37).

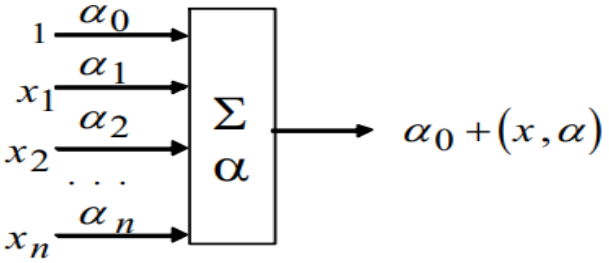


Рис.2.37. Неоднородный адаптивный сумматор

Как правило, набор синапсов рассматривается вместе с сумматором, к которому они подключены, а веса синапсов одновременно служат весами адаптивного сумматора. Веса синапсов сети образуют набор параметров, настраивая которые, нейронная сеть обучается решению задачи. На диапазон изменения весов синапсов накладываются ограничения, например, принадлежности веса синапса диапазону [-1, 1] или другой удобный диапазон.

Адаптивный сумматор может быть представлен с использованием n линейных связей и простого сумматора (рис.2.38).

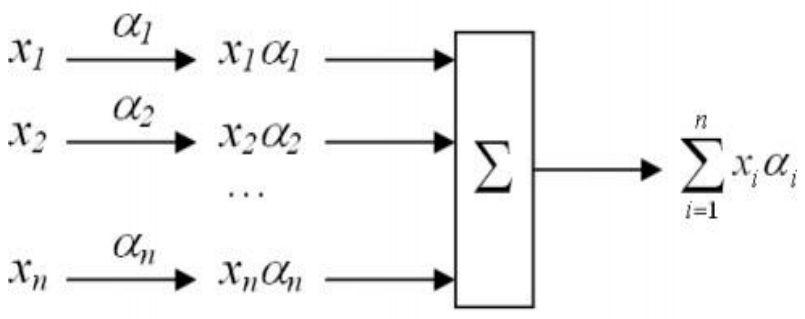


Рис.2.38. Комбинированный адаптивный сумматор

Нелинейный преобразователь сигнала изображен на рис.2.39. Как правило, она нелинейна, но может быть линейной, ступенчатой (sign**)** или гладкой.

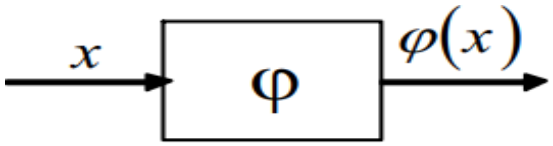


Рис.2.39. Нелинейный преобразователь сигнала

Точка ветвления служит для рассылки одного сигнала по нескольким адресам (рис. 2.40). Она получает скалярный входной сигнал x и передает его всем своим выходам.

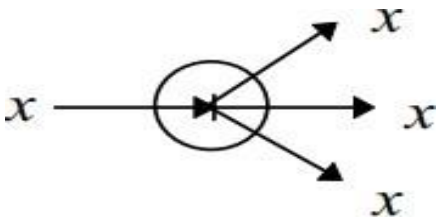


Рис.2.40. Точка ветвления

Объединяет эти элементы так называемый формальный нейрон. Он состоит из входного сумматора, нелинейного преобразователя и точки ветвления на выходе (рис.2.41). Существуют варианты, в которых нейрон имеет два сумматора, возможны также и другие типы нейронов и сумматоров.

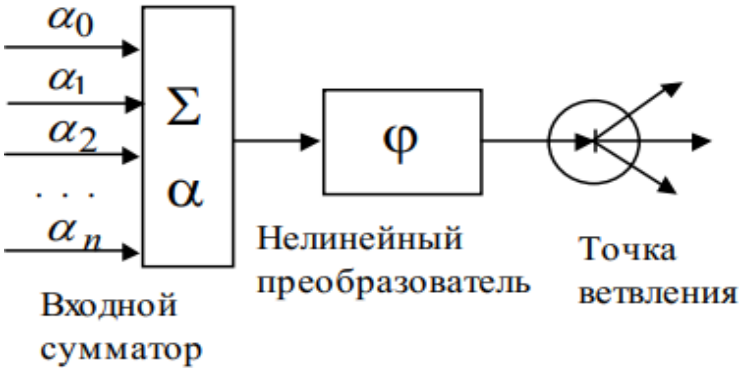


Рис.2.41. Обобщенная схема формального нейрона

Перечисленные выше простые элементы нейронных сетей могут быть представлены в более удобном виде и объединены в более сложные элементы: блок, слой, колонка. Чаще всего используется слой. **Слой** (layer) - набор нейронов или сумматоров, одновременно воспринимающий входную информацию и одновременно генерирующих выходные сигналы.

Структура и свойства искусственного нейрона

На рис. 2.42 показана структура искусственно созданного нейрона, который является приблизительным аналогом настоящего биологического нейрона и отображает аппаратно-программную сущность искусственного нейрона. Он состоит из элементов трех типов: умножителей (синапсов), сумматора и нелинейного преобразователя. Синапсы осуществляют связь между нейронами, умножают входной сигнал на число вес синапса. Сумматор выполняет сложение сигналов, поступающих по синаптическим связям от других нейронов, и внешних входных сигналов.

Нелинейный преобразователь реализует нелинейную функцию одного аргумента - выхода сумматора. Эта функция называется функцией активации или передаточной функцией нейрона. Нейрон в целом реализует скалярную функцию векторного аргумента.

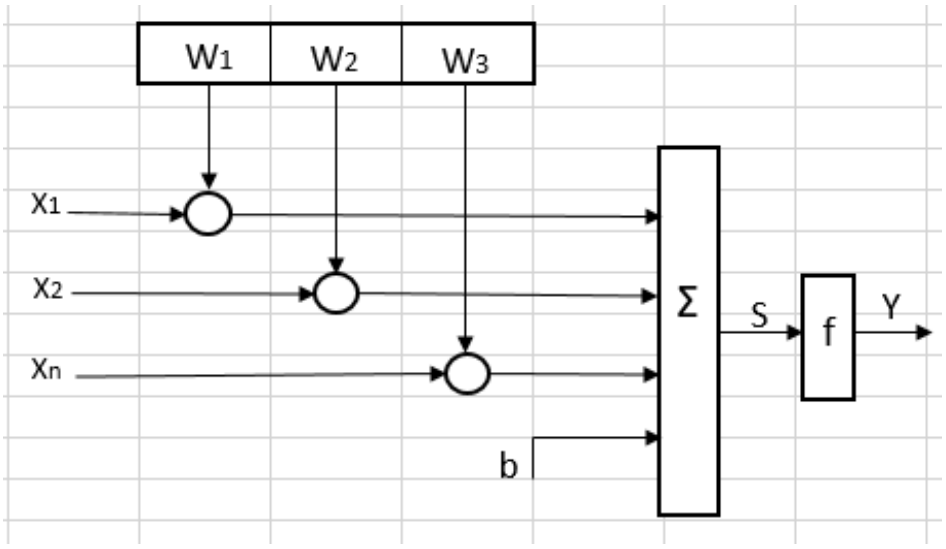


Рис.2.42. Структура искусственного нейрона

В общем случае входной сигнал, весовые коэффициенты и смещение могут принимать действительные значения, а во многих практических задачах - лишь некоторые фиксированные значения.

Выход (*y*) определяется видом функции активации и может быть как действительным, так и целым. Синаптические связи с положительными весами называют возбуждающими, с отрицательными весами - тормозящими.

В качестве оператора нелинейного преобразования могут использоваться различные функции, которые определяются в соответствии с решаемой задачей и типом нейронной сети. Функции активации осуществляют линейные и нелинейные преобразования над выходными значениями сумматора.

Пусть *T* – порог нейронного элемента, характеризующий расположение функции активации по оси абсцисс.

Рассмотрим примеры распространенных функций активации [31.32].

Линейная функция активации.

В этом случае выходное значение нейронного элемента равняется взвешенной сумме Изменение порога линейного элемента эквивалентно сдвигу функции активации по оси абсцисс S (рис. 2.43).

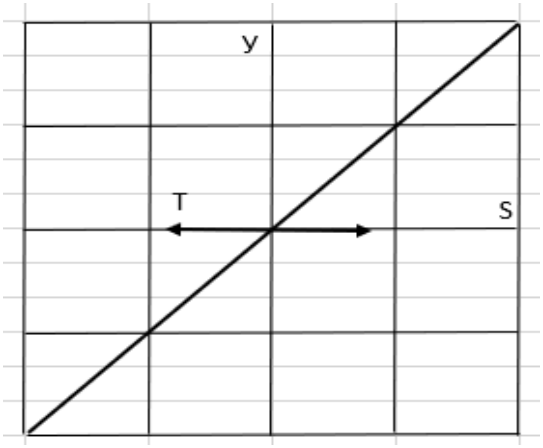


Рис.2.43. Линейная функция активации

Пороговая функция активации.

В качестве пороговой функции активации может использоваться или биполярная, или бинарная пороговая функция. В случае использования пороговой бинарной функции активации (рис.2.44):

y = sign (S)

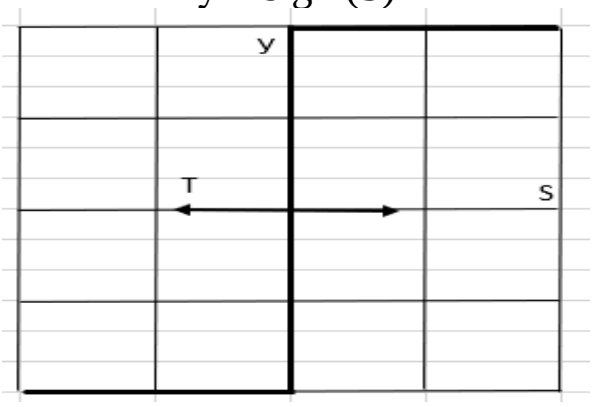


Рис. 2.44.Пороговая функция активации

Линейная ограниченная функция.

В этом случае выходное значение нейрона определяется следующим образом (рис.2.45):

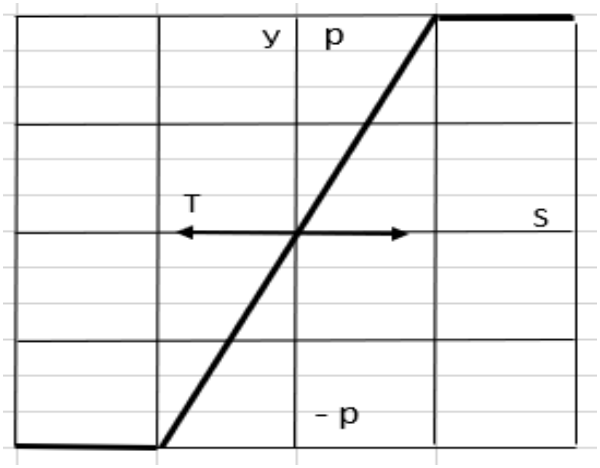


Рис.2.45. Линейная ограниченная функция

Сигмоидная функция.

Эта функция является непрерывным аналогом бинарной пороговой функции активации и представляет собой возрастающую, ограниченную функцию в диапазоне значений [0, 1]. На практике используются как униполярные (2.46), так и биполярные (2.47) функции активации.

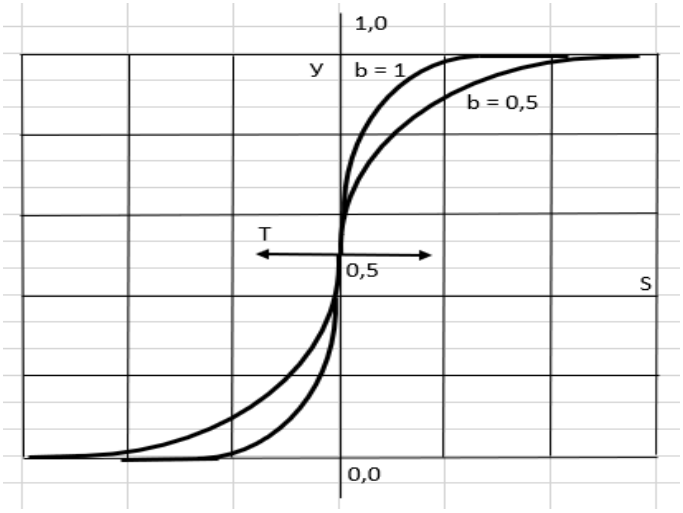


Рис.2.46. Униполярная сигмоидная функция

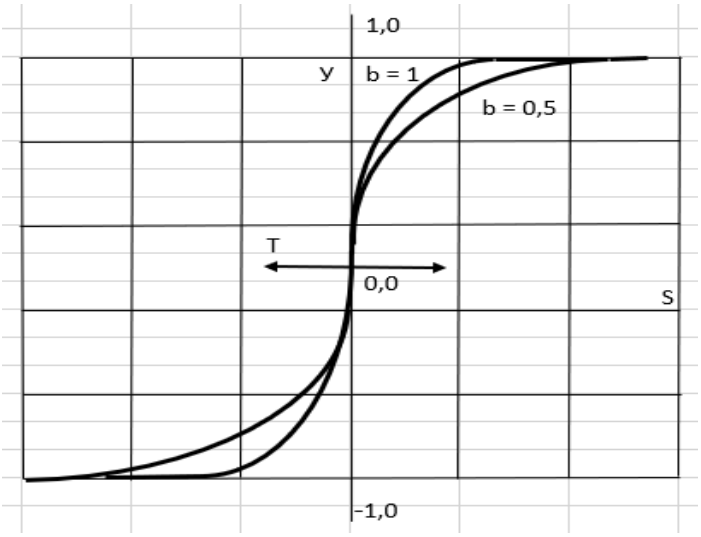


Рис.2.47. Биполярная сигмоидная функция

Коэффициент *b* определяет "крутизну" функций и выбирается разработчиком сети. На практике *b* для упрощения назначают обычно равным 1. По аналогии с электронными системами активационная функция считается нелинейной усилительной характеристикой искусственного нейрона. Коэффициент усиления вычисляется как отношение приращения величины *y* к вызвавшему его небольшому приращению величины s**.** Он выражается наклоном кривой при определенном уровне возбуждения и изменяется от малых значений при больших отрицательных возбуждениях (кривая почти горизонтальна) до максимального значения при нулевом возбуждении и снова уменьшается, когда возбуждение становится большим положительным.

Подобная нелинейная характеристика решает поставленную им дилемму шумового насыщения, благодаря чему одна и та же сеть может обрабатывать как слабые, так и сильные сигналы.

Нейрон возбуждается при достижении входным сигналом порогового значения. Сигнал возбуждения на выходе нейрона может меняться скачкообразно или нелинейно в зависимости от величины входного сигнала S, который в свою очередь является суммой сигналов на входах нейрона. Сглаженная **S**-образная сигмоида — это и есть то, что мы будем использовать при изучении нейронной сети. Сигмоида, которую иногда называют также логистической функцией, описывается следующей формулой:

$$y = 1/1+e^{-x}, \quad \text{где } e = 2{,}71828.$$

При нулевом значении *x* выражение **e** принимает значение 1,

поскольку возведение любого числа в нулевую степень всегда дает 1**.** Поэтому *y* становится равным 1 / (1 + 1) или просто 1/2, т.е. половине. Следовательно, базовая сигмоида пересекает ось *y* тогда, когда *y* = 1/2.

Реальные биологические нейроны имеют несколько входов, а не только один. При более чем одном входе необходимо комбинировать, суммируя соответствующие значения, а результирующая сумма будет служить входным значением для сигмоиды, управляющей выходным значением. Такая схема отражает принцип работы нейронной сети.

Гиперболический тангенс.

Функция гиперболического тангенса (рис.2.48) аналогична биполярной сигмоидной функции.

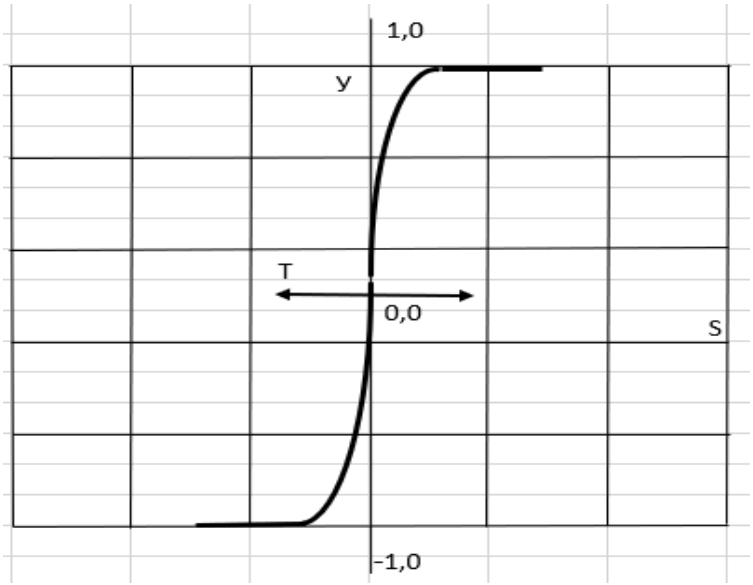


Рис.2.48. Функция гиперболический тангенс

Радиально-базисная сигмоидная функция.

Эта функция определяется функцией Гаусса для нормального закона распределения (рис.2.49), в соответствии с которой:

$$y = \exp(-S^2/2b^2),$$

где b – среднеквадратическое отклонение ширины радиально-базисной

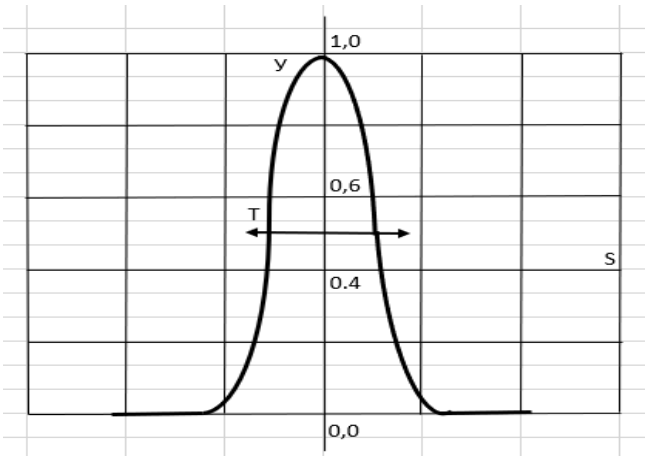


Рис.2.49. Радиально-базисная функция

Величина *S* определяется, как правило, в соответствии с евклидовым расстоянием между входным и весовым вектором:

Ректификационная функция активации.

Эта функция используется в глубоких нейронных сетях (рис. 2.50).

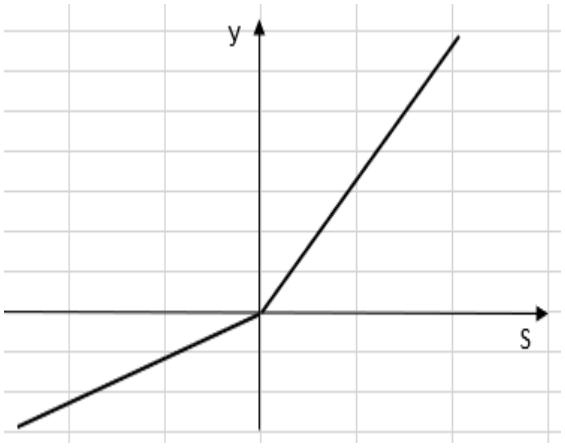


Рис.2.50. Ректификационная функция

Общие вопросы функционирования нейронных сетей

Нейронной сетью называется некоторая последовательность нейронов, объединённых между собой синапсами. Каждый нейрон способен иметь множество синапсов, которые ослабляют или усиливают сигнал. Нейроны способны менять свои характеристики в течение определённого времени. Правильно выбрав параметры синапсов, мы сможем получать на выходе правильные результаты

преобразования входного вектора данных.

Связи между искусственными нейронами называются синаптическими, или просто синапсами. У синапса имеется единственный параметр — весовой коэффициент, в зависимости от его значения происходит то или иное изменение информации, когда она передается от одного нейрона к другому. Именно благодаря этому входная информация обрабатывается и превращается в результат, а обучение нейронной сети основано на подборе такого весового коэффициента для каждого синапса, который и приводит к получению требуемого результата.

Таким образом, на результат обработки оказывают влияние конкретно синапсы, которые дают совокупность веса входных данных, собственно сами нейроны постоянно выполняют абсолютно одинаковые вычисления. Выставление весов осуществляется в случайном порядке или с учетом опыта решения подобных задач.

Искусственная нейронная сеть (ИНС) — это математическая модель, а также ее программное или аппаратное воплощение, построенная по принципу организации и функционирования биологических нейронных сетей — сетей нервных клеток живого организма. Совокупность нейронных элементов и связей между ними называется нейронной сетью. Слой нейронной сети – это множество нейронных элементов, на которые в каждый такт времени параллельно поступает информация от других нейронных элементов сети [42-44].

В нейронных сетях выделяют несколько видов слоев – входной, выходной, скрытый внутренний рабочий слой **(**hidden), не получающий входных сигналов и не генерирующий выходных. В зарубежных источниках принято особо выделять нейронные сети с двумя скрытыми слоями. Чаще всего используется от 1 до 3 скрытых слоев, хотя встречается использование нейронных сетей с 9 скрытыми слоями.

Перечисленные выше элементы, структуры и свойства отдельных нейронов при решении сложных вычислительных задач с применением алгоритмов искусственного интеллекта как правило объединяются в структуры искусственных нейронных сетей

Таким образом, алгоритм работы сети следующий:

— на входной слой нейронов происходит поступление данных;

— информация передаётся с помощью синапсов следующему слою, причём каждый синапс имеет собственный коэффициент веса, а

любой следующий нейрон способен иметь несколько входящих синапсов;

— данные, полученные следующим нейроном, это сумма всех данных для предыдущих узлов, которые перемножены на коэффициенты весов;

— полученное в итоге значение подставляется в функцию активации, в результате чего происходит формирование выходной информации;

— информация передаётся дальше, пока не дойдёт до конечного выхода.

Выбор структуры нейронной сети осуществляется в соответствии с особенностями и сложностью задачи. Если же задача не может быть сведена ни к одному из известных типов, приходится решать сложную проблему синтеза новой конфигурации. При этом необходимо руководствоваться следующими основными правилами:

- возможности сети возрастают с увеличением числа нейронов сети, плотности связей между ними и числом слоев;
- введение обратных связей наряду с увеличением возможностей сети поднимает вопрос о динамической устойчивости сети;
- сложность алгоритмов функционирования сети, введение нескольких типов синапсов способствует усилению мощности нейронной сети.

Еще одна классификация делит нейронные сети на синхронные и асинхронные. В первом случае в каждый момент времени лишь один нейрон меняет свое состояние, во втором – состояние меняется сразу у целой группы нейронов, как правило, у всего слоя.

Основными преимуществами нейронных сетей перед традиционными вычислительными методами являются.

1. Возможность решение задач в условиях неопределенности. Благодаря способности к обучению нейронная сеть позволяет решать задачи с неизвестными закономерностями и зависимостями между входными и выходными данными, что позволяет работать с неполными данными.

2. Устойчивость к шумам во входных данных. Нейронная сеть может самостоятельно выявлять неинформативные для анализа параметры и производить их отсев, в связи с чем отпадает необходимость в предварительном анализе входных данных.

1. Гибкость структуры нейронных сетей. Компоненты

нейрокомпьютеров (нейроны и связи между ними) можно комбинировать различными способами. За счет этого один нейрокомпьютер можно применять для решения различных задач, зачастую никак не связанных между собой.

2. Высокое быстродействие. Входные данные обрабатываются многими нейронами одновременно, благодаря чему нейронные сети решают задачи быстрее, чем большинство других алгоритмов.

3. Адаптация к изменениям окружающей среды. Нейронные сети, обучаясь на данных, способны подстраиваться под изменяющуюся окружающую среду (например, под изменениях входных параметров приборов или оборудования в системах контроля и управления). Если необходимо решать какую-то задачу в условиях нестационарной среды, то могут быть созданы нейронные сети, переучивающиеся в режиме реального времени. Чем выше адаптивные способности системы, тем более устойчивой будет ее работа в нестационарной среде.

4. Отказоустойчивость нейронных сетей. На неблагоприятное изменение условий нейросеть реагирует лишь незначительным снижением производительности. Эта особенность объясняется распределенным характером хранения информации в нейронной сети, поэтому только серьезные повреждения структуры могут существенно повлиять на работоспособность нейросети.

Основными недостатками нейронных сетей являются.

1. Ответ, выдаваемый ИНС, всегда приблизительный. Нейронные сети не способны давать точные и однозначные ответы. Но задачи, в которых надо применять ИНС и одновременно получать точные ответы, встречаются довольно редко.

2. Неспособность принятия решений в несколько этапов. Нейронная сеть не может решать задачи, которые требуют последовательного выполнения нескольких шагов; она способна решать задачу только "в один заход". Поэтому нейросеть не может, например, доказать математическую теорему.

3. Неспособность решать некоторые вычислительные задачи. В ИНС нельзя загрузить, допустим, математическое уравнение и получить его решения для различных параметров. Но это и не является предназначением нейронных сетей.

4. Трудоемкость и длительность обучения. Для того чтобы

нейронная сеть могла корректно решать поставленные задачи, требуется провести ее обучение на десятках и сотен наборов входных данных. Но уже разработаны различные технологии ускоренного обучения, современные видеокарты позволяют обучать нейросети в сотни раз быстрее, а недавно появились готовые, предобученные нейросети, в частности, распознающие образы. На основе таких нейросетей можно создавать приложения, не занимаясь длительным обучением.

Нейронные сети с одним обрабатывающим слоем

В данном разделе будут рассматриваться нейронные сети, состоящие из одного слоя нейронных элементов, осуществляющего обработку входной информации. Такие сети принято изображать в виде двуслойной нейронной сети, где первый слой нейронных элементов является распределительным, а второй – обрабатывающим. Распределительный слой передает входные сигналы на обрабатывающий слой нейронных элементов, который преобразует входную информацию в соответствии с синаптическими связями и функцией активации (рис.2.51). При этом каждый нейрон распределительного слоя имеет синаптические связи со всеми нейронами обрабатывающего слоя. Тогда выходное значение *j*-го нейронного элемента второго слоя можно представить как [41,44]:

$$y_j = F(S_j) = F\left(\sum_{i=1}^{n} \omega_{ij} x_i - T_j\right),$$

где *Tj* – порог *j*-го нейронного элемента выходного слоя; w*ij* – сила синаптической связи между *i*-м нейроном распределительного слоя и *j*-м нейроном обрабатывающего слоя.

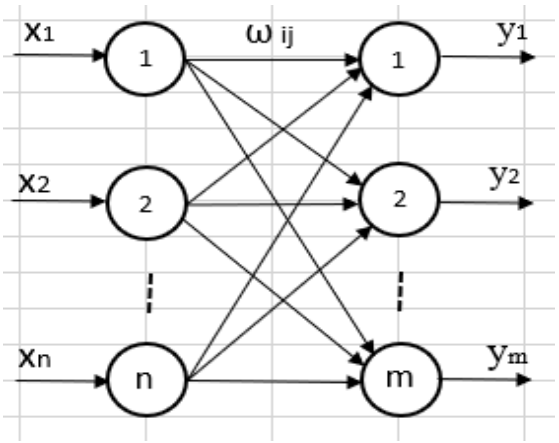


Рис.2.51. Топология однослойной нейронной сети

Совокупность весовых коэффициентов этой сети можно представить матрицей размерностью n x m :

$$W = \begin{bmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1m} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2m} \\ \dots & \dots & \dots & \dots \\ \omega_{n1} & \omega_{n2} & \dots & \omega_{nm} \end{bmatrix}.$$

Тогда вектор-столбец взвешенной суммы в матричном виде определяется следующим образом:

$$S = W^T X - T,$$

где T – вектор-столбец порогов нейронных элементов второго слоя.

Однослойная нейронная сеть обладает достаточно ограниченными возможностями. Рассмотрим одну из них, пример которой дан в гл.6. в разделе бинарной классификации.

Бинарная классификация. Рассмотрим однослойный персептрон с двумя нейронами входного и одним нейроном выходного слоя (рис.2.52). Тогда взвешенная сумма вычисляется как

$$S = \omega 11\ x1 + \omega 21\ x2 - T1 \quad ,$$

а выходное значение нейронной сети $y_1 = F(S)$ соответственно.

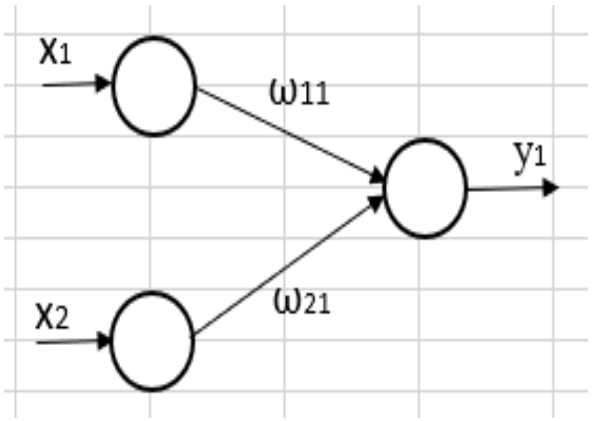


Рис.2.52. Сеть с одним выходным нейроном

Пороговое значение можно вынести за знак нейронного элемента и использовать в качестве весового коэффициента. В этом случае необходимо добавить в схему сети один входной нейрон и подать на его вход постоянное значение –1 (рис.2.53).

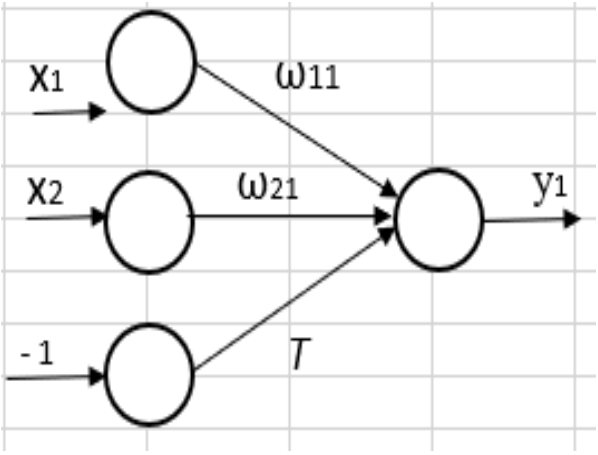


Рис.2.53. Представление порога в качестве весового элемента

Пусть используется пороговая функция активации в выходном нейронном элементе. Тогда

$$y_1 = F(S) = \begin{cases} 1, S > 0, \\ 0, S \le 0. \end{cases}$$

Такая сеть осуществляет линейное разбиение входного пространства образов на два класса (класс нулей и единиц) и может использоваться для решения задач бинарной классификации образов. Уравнение разделяющей линии отделяет область решений, соответствующую одному классу, от другого класса и называется дискриминантной (разделяющей) линией.

Многослойные (слоистые) нейронные сети

Нейронная сеть представляет собой совокупность нейроподобных элементов, определенным образом соединенных друг с другом и с внешней средой с помощью связей, определяемых весовыми коэффициентами. В зависимости от функций, выполняемых нейронами в сети, можно выделить три их типа:

- входные нейроны, на которые подается вектор, кодирующий входное воздействие или образ внешней среды; в них обычно не осуществляется вычислительных процедур, а информация передается с входа на выход путем изменения их активации;
- выходные нейроны, выходные значения которых представляют выходы нейронной сети;
- промежуточные нейроны, составляющие основу нейронных сетей, преобразования в которых выполняются также по выше приведенным выражениям.

В процессе функционирования сети осуществляется

преобразование входного вектора в выходной, некоторая переработка информации. Конкретный вид выполняемого сетью преобразования данных обусловливается не только характеристиками нейроподобных элементов, но и особенностями ее архитектуры, а именно топологией межнейронных связей [44].

В слоистых сетях нейроны расположены в несколько слоев. Нейроны первого слоя получают входные сигналы, преобразуют их и через точки ветвления передают нейронам второго слоя. Далее срабатывает второй слой и т.д. до k-го, который выдает выходные сигналы для интерпретатора и пользователя. Каждый выходной сигнал i-го слоя подается на вход всех нейронов i+1-го. Число нейронов в каждом слое может быть любым и никак заранее не связано с количеством нейронов в других слоях.

Общепринятый способ подачи входных сигналов: все нейроны первого слоя получают каждый входной сигнал. Особое распространение получили двухслойные и трехслойные сети прямого распространения, в которых нет обратных связей. Теоретически число слоев и число нейронов в каждом слое может быть произвольным, однако фактически оно ограничено ресурсами компьютера. Чем сложнее сеть, тем более сложные задачи она может решать. Рассмотрим персептроны с различным количеством слоев.

Персептрон с одним скрытым слоем.

Количество слоев характеризует способность многослойной нейронной сети к разбиению входного пространства образов на подпространства меньшей размерности. Персептрон с одним скрытым слоем и нелинейной функцией активации нейронных элементов позволяет формировать в пространстве решений любые разделяющие поверхности, как выпуклые, так и невыпуклые. Прежде всего следует отметить, что возможности персептрона с одним скрытым слоем различаются в зависимости от используемой в нем функции активации.

Персептрон с одним скрытым слоем является универсальным аппроксиматором. Он способен с высокой степенью точности аппроксимировать непрерывную функцию, если в качестве функции активации нейронных элементов скрытого слоя используется непрерывная, монотонно возрастающая, ограниченная функция. При этом точность аппроксимации функции зависит от количества нейронов в скрытом слое, чем больше нейронов, тем больше точность

аппроксимации. Однако при слишком большой размерности скрытого слоя может наступить явление, которое называется перетренировкой сети, когда сеть имеет плохую обобщающую способность.

В качестве функции активации могут использоваться функции активации: сигмоидная, биполярная сигмоидная и гиперболический тангенс.

Проведем анализ персептрона с одним скрытым слоем (рис.2.54) в качестве классификатора при использовании пороговой функции активации нейронных элементов [31].

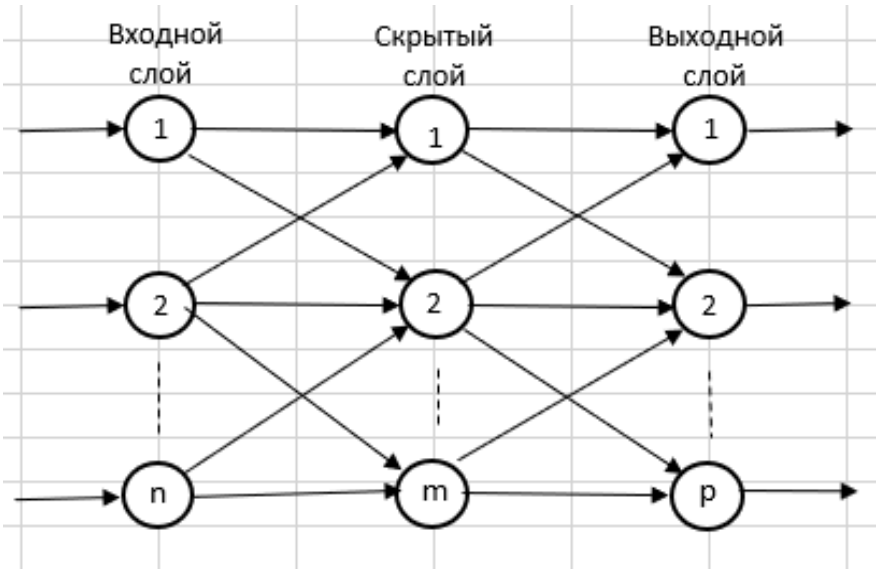


Рис.2.54. Персептрон с одним скрытым слоем

Входной слой обеспечивает прием входных переменных. Нейроны скрытого слоя представляют собой локальные детекторы признаков. Они разбивают входное пространство образов на области с помощью гиперплоскостей (прямых в двумерном случае). Число нейронов в скрытом слое характеризует количество таких областей. Выходной слой нейронных элементов производит объединение полученных областей в соответствующие классы.

Рассмотрим конкретный пример организации сети. Пусть p – количество классов, на которые необходимо разбить входное пространство образов. Число классов равно количеству нейронов выходного слоя. Для разбиения входного пространства образов на m классов необходимо, чтобы число областей было больше числа классов,. Тогда количество нейронов в скрытом слое можно оценить

следующим образом:

$$m > \log_2 p.$$

Основной формой работы НС является так называемое «обучение» сети, т.е. настройка (подбор) весов сети на решение конкретной задачи. Подбор ведется в интерактивном режиме до тех пор, пока на выходе не будет получен ожидаемый результат.

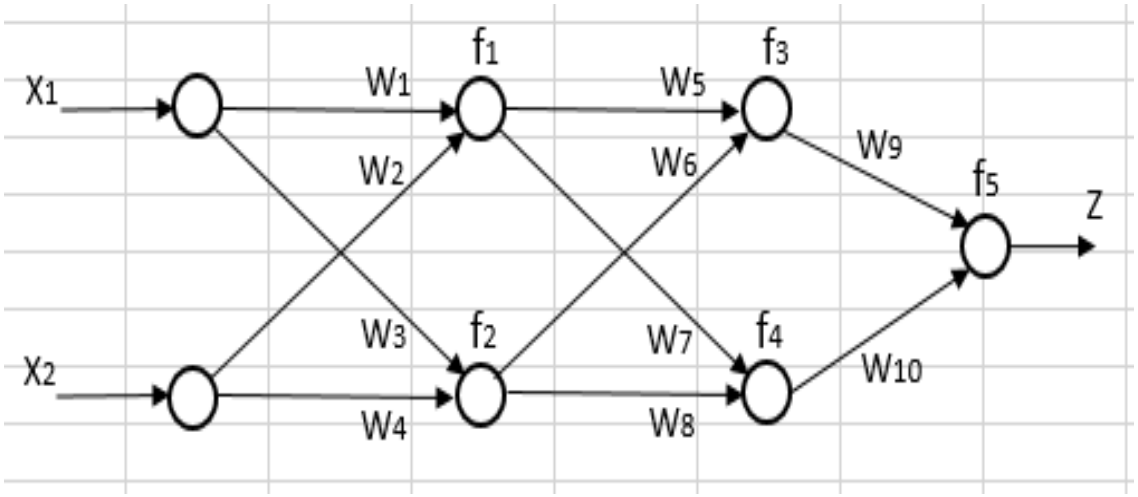


Рис.2.55. Модель трехслойной сети

Сеть считается обученной, если все ее веса w1, w2, …, w10 имеют конкретные значения, подобранные таким образом, что если на входы x_1 и x_2 подать входные данные, то на выходе **z** получим некоторый результат, согласующийся с нашей задачей.

Таблица 2.2.

$t_1 = w_1 x_1 + w_2 x_2$	$y_1 = f_1(t_1)$
$t_2 = w_3 x_1 + w_4 x_2$	$y_2 = f_2(t_2)$
$t_3 = w_5 y_1 + w_6 y_2$	$y_3 = f_3(t_3)$
$t_4 = w_7 y_1 + w_8 y_2$ $t_5 = w_9 y_3 + w_{10} y_4$	$y_4 = f_4(t_4)$ $z = y_5 = f_5(t_5)$

Таким образом, вычисление функции z по нейросети

идет слева направо, от слоя к слою. Если сохранять промежуточные результаты в каждом нейроне (значения **t**i и *y***i**), то вычисления в каждом нейроне такой сети сводятся к нахождению скалярного произведения вектора входов на соответствующие веса синапсов и последующему вычислению функции нелинейного преобразования f**i** от полученного результата.

Нельзя подавать сети входные сигналы в их истинном диапазоне величин и получать от сети выходные сигналы в требуемом диапазоне. Поэтому перед подачей сети входных сигналов их необходимо нормировать, например, в диапазон значений [-1,1] или [0,1] в тех случаях, когда существенна положительность, либо делать так, чтобы входные сигналы не слишком сильно выходили за пределы этих отрезков. Нормирование можно выполнить по указанным ниже формулам, в которых каждая i-я компонента входного вектора данных *x*i заменяется новой величиной.

Функционирование нейронных сетей прямого распространения

Далее будут рассматриваться двухслойные и трехслойные синхронные сети прямого распространения.

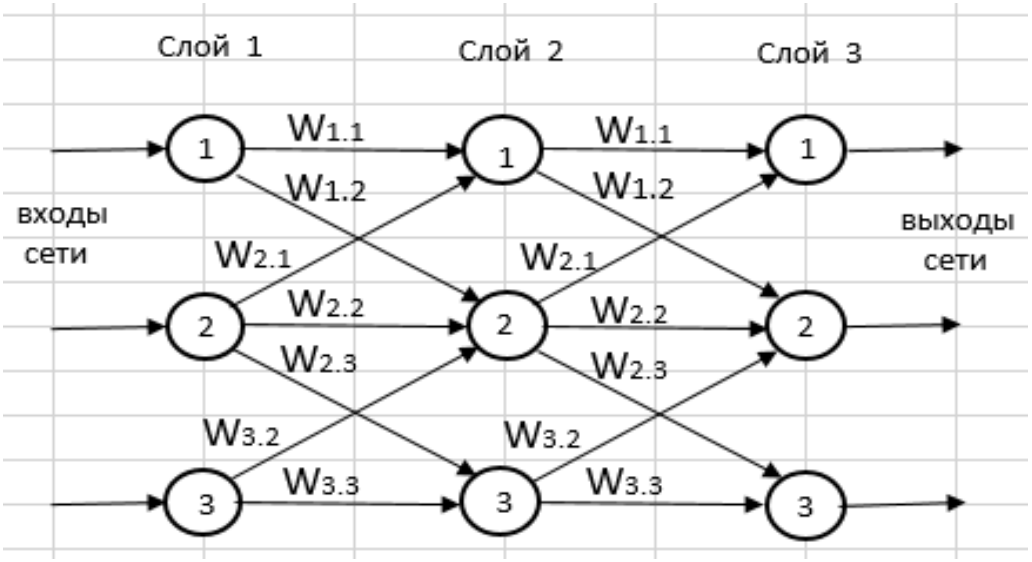


Рис.2.56. Трехслойная сеть прямого распространения

На рис.2.56 представлены три слоя, каждый из которых включает три искусственных нейрона, или узла, каждый узел соединен с каждым из узлов предшествующего и последующего слоев. Параметром регулирования такой сети является сила связи между узлами. с каждым соединением (связью) ассоциируется определенный вес **(**W**i,j)**. Низкий

весовой коэффициент ослабляет сигнал, высокий — усиливает его.

Символ W**2,3** обозначает весовой коэффициент, связанный с сигналом, который передается от узла 2 данного слоя к узлу 3 следующего слоя. Следовательно, $W_{1,2}$ — это весовой коэффициент, который ослабляет или усиливает сигнал, передаваемый от узла 1 к узлу 2 следующего слоя.

Приведенные на рис.3.56 типы соединений узлов не являются обязательными, связи между узлами могут быть произвольными, здесь дан упрощенный и широко применимый вариант связей.

Распространение сигналов в узлах нейронной сети.

Возьмем простой вариант построения сети (рис.2.57). Значениям сигналов на входе соответствуют 1,0 и 0,5. Возьмем произвольно начальные значения весовых коэффициентов:
$W_{1,1} = 0,9$; $W_{1,2} = 0,2$; $W_{2,1} = 0,3$; $W_{2,2} = 0,9$.

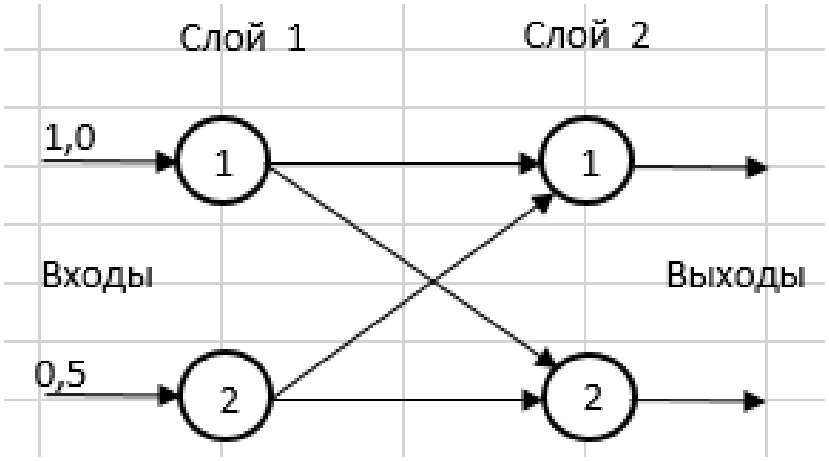


Рис.2.57. Нейронная сеть из двух слоев по два нейрона

Случайное значение улучшается с каждым очередным тренировочным примером, используемым для обучения классификатора. То же самое должно быть справедливым и для весовых коэффициентов связей в нейронных сетях. В данном случае, когда сеть небольшая, имеем всего четыре весовых коэффициента, поскольку таково количество всех возможных связей между узлами при условии, что каждый слой содержит по два узла [39].

Ниже приведена схема (рис.2.58), на которой все связи промаркированы соответствующим образом.

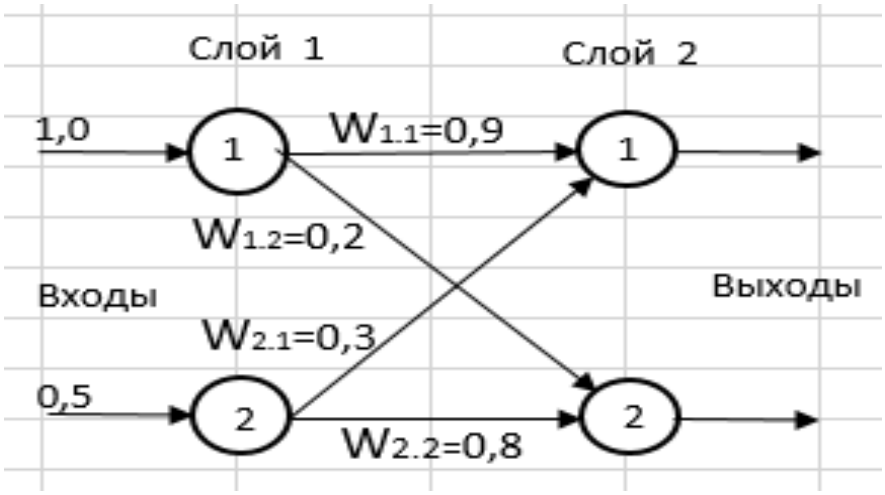


Рис.2.58. Сеть с установленными весовыми коэффициентами

Первый слой узлов — входной, и его единственное назначение — представлять входные сигналы, по правилам во входных узлах функция активации к входным сигналам не применяется. С первым слоем все просто — никаких вычислений. Предстоит определить входной сигнал для каждого узла в этом слое.

В ранее приведенной сигмоиде

$$y = 1/1+e^{-x}$$

значение x — комбинированный сигнал на входе каждого узла. Данная комбинация образуется из необработанных выходных сигналов связанных узлов предыдущего слоя, сглаженных весовыми коэффициентами связей.

Сосредоточим внимание на узле 1 слоя 2. С ним связаны оба узла первого, входного слоя. Исходные значения на этих входных узлах равны 1,0 и 0,5.

Связи первого узла назначен весовой коэффициент 0,9, связи второго — 0,3. Поэтому сглаженный входной сигнал на входе узла 1 слоя 2 вычисляется с помощью следующего выражения:

$x = (1,0 \times 0,9) + (0,5 \times 0,3) = 0,9 + 0,15 = 1,05$

Имеем значение $x = 1,05$ для комбинированного сглаженного входного сигнала первого узла второго слоя и теперь необходимо

рассчитать для этого узла выходной сигнал (рис.2.59) с помощью выше приведенной функции активации $y = 1/1+e^{-x}$:
$y = 1/(1 + e^{-1,05}) = 1/1,3499 = 0,7408$.

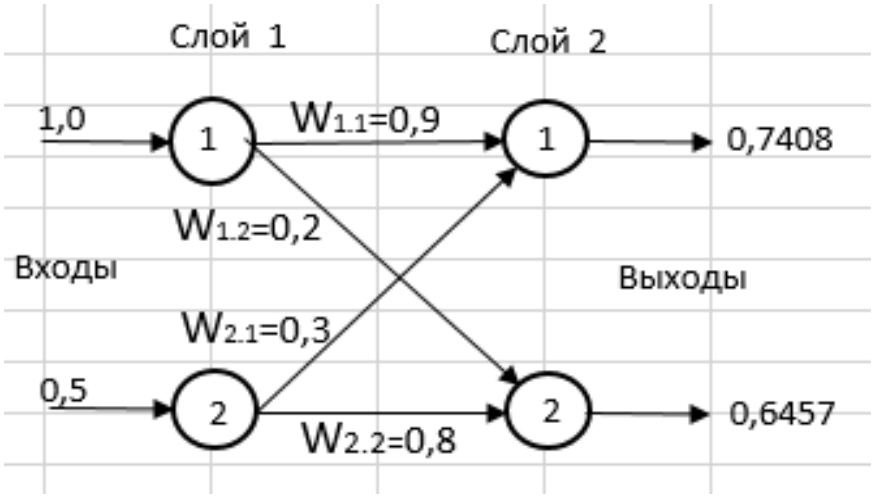


Рис.2.59. Сеть с установленными выходными сигналами

Повторим те же вычисления для оставшегося узла 2 второго слоя, т.е. вновь вычислим сглаженный входной сигнал с помощью следующего выражения:

$$x = (1,0 \times 0,2) + (0,5 \times 0,8) = 0,2 + 0,4 = 0,6$$

$$y = 1/(1+ e^{-0,6}) = 1/1,5488 = 0,6457$$

Вычисления для нейронных сетей, особенно многослойных, удобнее проводить с помощью матриц. Во-первых, они обеспечивают сжатую запись операций, придавая им компактную форму. Во-вторых, в языках программирования предусмотрена работа с матрицами, при этом многие операции повторяются. Чтобы матрицы можно было умножить, количество столбцов в первой из них должно совпадать с количеством строк во второй.

Реализация вычислений с помощью матриц

В реальном программировании, естественно, никаких нейронов и связей не пишут, всё представляют матрицами и считают матричными произведениями, потому что нужна скорость [30].

В рамках матричного подхода все вычисления, которые необходимы для расчета входных сигналов, поступающих на каждый из узлов второго слоя, могут быть выражены с использованием матричного умножения, обеспечивающего чрезвычайно компактную форму записи соответствующих операций:

$$x = W \times I$$

Здесь W — матрица весов, I — матрица входных сигналов, а x— результирующая матрица комбинированных сглаженных сигналов, поступающих в слой 2.

Увеличение количества слоев приводит лишь к увеличению размера матриц, но количество символов в записи при этом не увеличивается. Она остается по-прежнему компактной, и мы просто записываем произведение матриц в виде W x I, независимо от количества элементов в каждой из них. Если используемый язык программирования распознает матричную нотацию, то компьютер выполнит все трудоемкие расчеты, связанные с вычислением выражения $x = W \times I$, без предоставления ему отдельных инструкций для каждого узла в каждом слое.

Видно, что верхний левый элемент результата (19) вычисляется с использованием верхней строки первой матрицы и левого столбца второй, а правый верхний элемент вычисляется с использованием верхней строки первой матрицы и правого столбца второй (22). Такой же порядок и для нижней строки первой матрицы: нижняя строка на левый столбец (43) и нижняя строка на правый столбец (50).

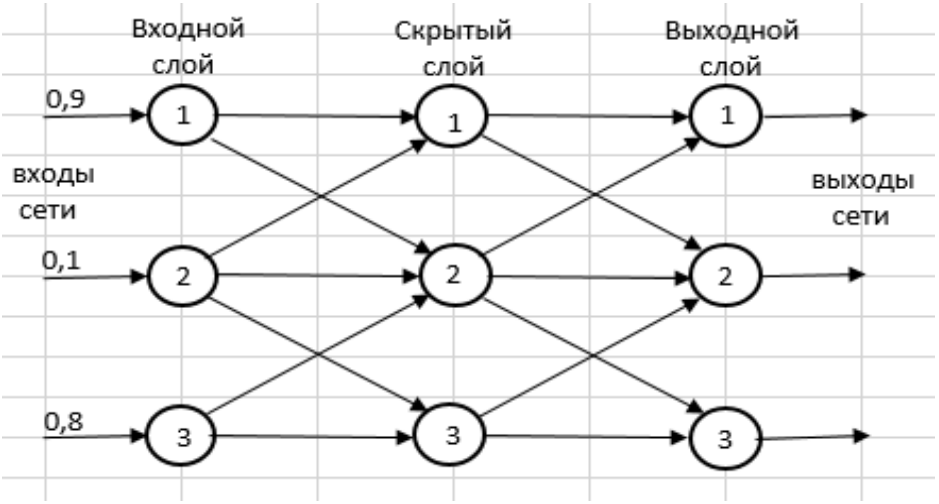


Рис.2.60. Нейронная сеть с тремя слоями

На следующей (рис.2.60) диаграмме представлен пример нейронной сети, включающей три слоя по три узла. Чтобы избежать загромождения диаграммы лишними надписями веса связей представлены отдельно.

Обучение многослойной нейронной сети. Это автоматический поиск закономерности между совокупностью обучающих данных и заранее известным результатом (примером). Обучающие данные образуют обучающую выборку – электронную таблицу или базу данных, каждая строка которых содержит один обучающий пример - вектор входных данных размерностью, равной числу входов нейросети, и вектор выходных данных, соответствующий выходам нейросети. Задача обучения состоит в том, чтобы нейросеть в ответ на вектор входных данных выдавала такой вектор выходных данных, который был бы наиболее близок к выходным данным из примера. Задача обучения будет рассмотрена отдельно.

Классификация искусственных нейронных сетей

Существует много вариантов построения искусственных нейронных сетей ИНС [44-48]. В данном случае классификация осуществлена в соответствии с качественными характеристиками нейронной сети. Их детальное описание будет рассмотрено в последующих разделах.

Нейронные сети можно классифицировать в зависимости от различны качеств, свойственных их архитектуре и обучению.

Обучение с учителем.

В этом случае заранее известно выходное пространство решений нейронной сети. Предполагается, что имеются входные сигналы и эталонные реакции на них [44]. В процессе обучения происходит целенаправленная корректировка синаптических связей нейронной сети (NN) до достижения наилучшего соответствия между реальными выходными значениями сети Y и их эталонными значениями e (рис.2.61).

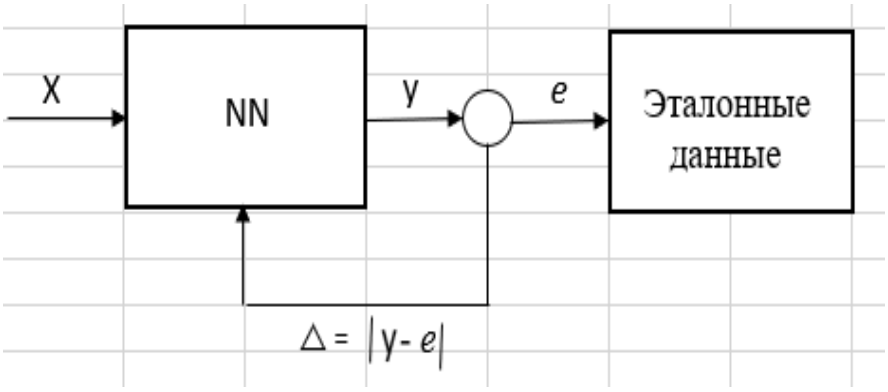


Рис.2.61. Обучение с учителем

Обучение без учителя.

В этом случае нейронная сеть формирует выходное пространство решений только на основе входных воздействий. Такие сети называются самоорганизующимися (рис.2.62).

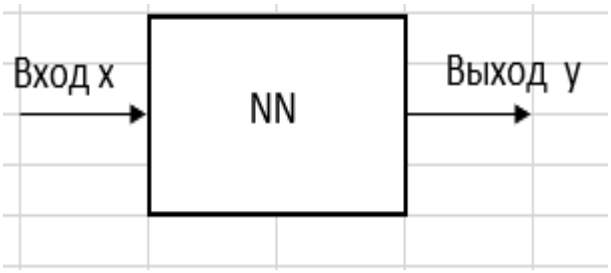


Рис.2.62. Обучение без учителя

Подкрепляющее обучение.

При подкрепляющем обучении выходной сигнал формируется с использованием сигнала подкрепления *r* от внешней среды (рис.2.63).

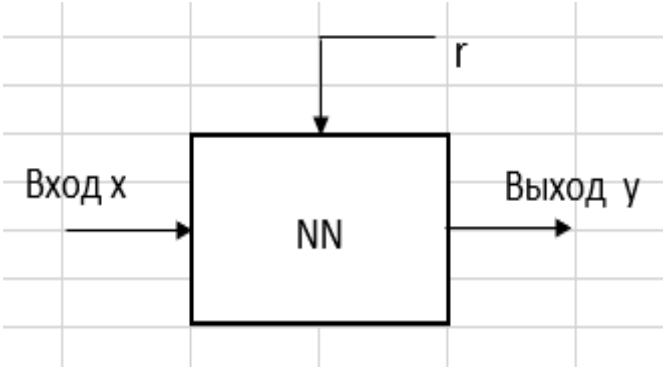


Рис.2.63. Подкрепляющее обучение

По характеру настройки синапсов.

Сети с фиксированными связями. В этом случае весовые коэффициенты нейронной сети выбираются сразу, исходя из условия задачи. Выбор весовых коэффициентов осуществляется в соответствии с ранее решаемыми аналогичными задачами.

Сети с изменяемыми (динамическими) связями. В них в процессе обучения происходит настройка синаптических связей.

По архитектуре и обучению.

I. Персептронные нейронные сети характеризуются, как правило, обучением с учителем.

Многослойные персептроны. Здесь, помимо выходного и входного слоёв, имеются ещё несколько скрытых промежуточных слоёв. Число этих слоёв зависит от степени сложности нейронной сети. Она в большей степени напоминает структуру биологической нейронной сети. Если сравнивать с однослойными, т.к. в процессе обработки данных каждый промежуточный слой — это промежуточный этап, на котором осуществляется обработка и распределение информации.

Рекуррентные нейронные сети, в которых присутствует обратная связь между входом и выходом. Выходное значение при этом определяется в зависимости как от входных, так и предшествующих выходных значений нейронной сети (рис.2.64). Этим нейросетям присуща функция кратковременной памяти, на основании чего сигналы восстанавливаются и дополняются во время их обработки.

Сверточные нейронные сети, представляющие дальнейшее развитие персептрона для обработки изображений. Сверточные

нейронные сети (convolutional neural networks, CNN) являются дальнейшим развитием многослойного персептрона и широко используются для обработки изображений

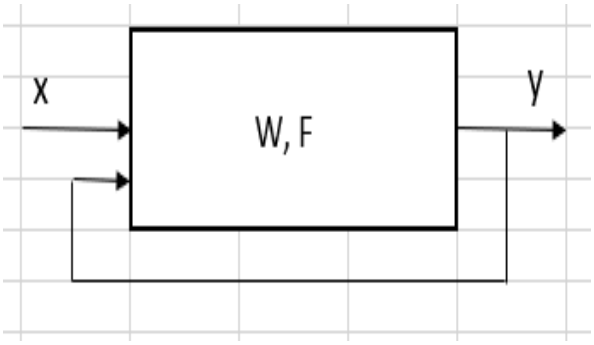


Рис.2.64. Рекуррентная нейронная сеть

Разновидностью рекуррентных сетей являются рециркуляционные (автоэнкодерные) нейронные сети (PCA-сети,), которым присуще как прямое $y = f(x)$, так и обратное $x = f^{-1}(y)$ преобразование информации (рис.2.65).

Рис.2.65. Рециркуляционная нейронная сеть

Релаксационные нейронные сети, в которых циркуляция информации происходит до тех пор, пока не перестанут изменяться выходные значения нейронной сети (состояние равновесия).

К ним относятся:

синхронные и асинхронные нейронные сети Хопфилда.

нейронные сети Хэмминга.

Самоорганизующиеся нейронные сети, реализующие алгоритмы обучения без учителя. Такое обучение основывается только на сигналах от внешней среды.

Для обучения сетей с самоорганизацией на основе конкуренции наибольшее распространение получил алгоритм Кохонена. В таких нейронных сетях, соседство нейронов носит топологический характер.

Глубокие нейронные сети, осуществляющие глубокое нелинейное иерархическое преобразование информации.

Гибридные нейронные сети. Отличаются применением двух подходов к обучению – с учителем и без учителя.

Нечеткие нейронные сети, характеризуемые применением нечеткой логики и нейронных сетей.

2.7.1. Обучение нейронных сетей
Основные правила обучения

Основным свойством биологического мозга является его способность к обучению, а поскольку искусственная нейронная сеть является моделью мозга, понятие «обучение» является также ключевым в теории ИНС. Процесс обучения рассматривается как адаптация параметров, а возможно и архитектуры сети для решения поставленной задачи путем оптимизации принятого критерия качества.

Обучение нейронной сети представляет собой автоматический поиск закономерности между совокупностью обучающих данных и заранее известным результатом. Обучающий пример состоит из вектора входных данных размерностью, равной числу входов нейросети, и вектора выходных данных, соответствующего выходам нейросети. Задача обучения состоит в том, чтобы нейросеть в ответ на вектор входных данных выдавала такой вектор выходных данных, который был бы наиболее близок к выходным данным из примера [41,45,49].

Таким образом, задача обучения нейросети сводится к минимизации ошибки как функции весов нейросети $w1$, $w2$, …, wn. Это означает, что весь мощный арсенал методов оптимизации может быть испытан для обучения.

Различают алгоритмы обучения с учителем и без учителя. Обучение

с учителем предполагает, что для каждого входного вектора существует целевой вектор, представляющий собой требуемый выход. Вместе они называются обучающей парой. Обычно сеть обучается на некотором числе таких обучающих пар. Предъявляется выходной вектор, вычисляется выход сети и сравнивается с соответствующим целевым вектором, разность (ошибка) с помощью обратной связи подается в сеть и веса изменяются в соответствии с алгоритмом, стремящимся минимизировать ошибку. Векторы обучающего множества предъявляются последовательно, вычисляются ошибки и веса подстраиваются для каждого вектора до тех пор, пока ошибка по всему обучающему массиву не достигнет приемлемо низкого уровня.

Обучение с учителем обладает биологической неправдоподобностью. Трудно вообразить обучающий механизм в мозге, который бы сравнивал желаемые и действительные значения выходов, выполняя коррекцию с помощью обратной связи.

Обучение без учителя является намного более правдоподобной моделью обучения в биологической системе. Такое обучение не нуждается в целевом векторе для выходов и, следовательно, не требует сравнения с предопределенными идеальными ответами. Обучающее множество состоит лишь из входных векторов. Обучающий алгоритм подстраивает веса сети так, чтобы получались согласованные выходные векторы, т. е. чтобы предъявление достаточно близких входных векторов давало одинаковые выходы. Предъявление на вход вектора из данного класса даст определенный выходной вектор, который может не полностью соответствовать требуемому выходному вектору, но ошибка не оценивается.

Алгоритм обучения персептрона

Простая нейронная модель, показанная на рис.2.66, использовалась в большей части начальных исследований нейронных сетей. Суммирующий элемент умножает каждый вход x на вес w и суммирует взвешенные входы. Если эта сумма больше заданного порогового значения, выход равен единице, в противном случае – нулю. Эти системы (и множество им подобных) получили название персептронов. Они состоят из одного слоя искусственных нейронов, соединенных с помощью весовых коэффициентов с множеством входов, хотя в

принципе описываются и более сложные системы.

Речь идет об уточнении наклона разграничительной линии (рис.2.67), отделяющей на графике одну группу точек данных, соответствующих парам значений длины и ширины, от другой [49]. Имеются два объекта: Ф1, имеющий ширину 3,0 и длину 1,0, и Ф2, имеющий большую длину — 3,0 и ширину — 1,0 (табл.2.3).

Таблица 2.3

Пример	Ширина	Длина	Объект
1	3,0	1,0	Ф1
2	1,0	3,0	Ф2

Нам известно, что данные (размеры объектов) в этом наборе примеров являются истинными. Именно с их помощью будет уточняться значение константы в функции классификатора. Примеры с истинными значениями, которые используются для обучения предиктора или классификатора, называют тренировочными данными. Отобразим эти два примера тренировочных данных на графике.

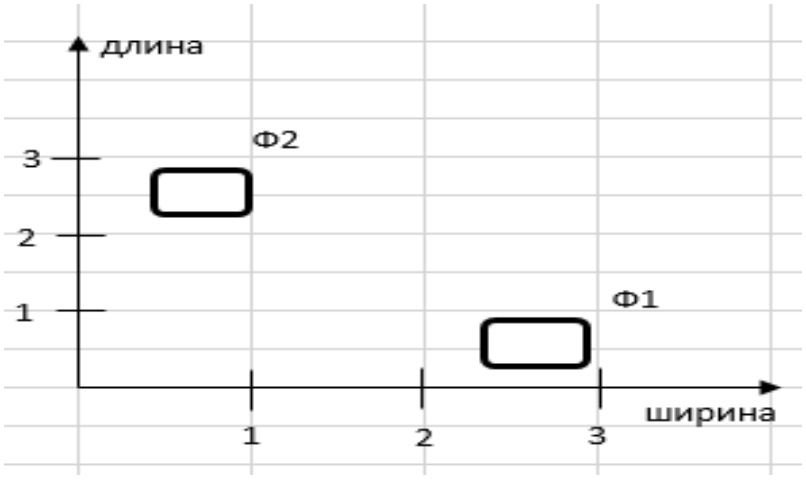


Рис.2.66. Истинные значения объектов в графическом представлении

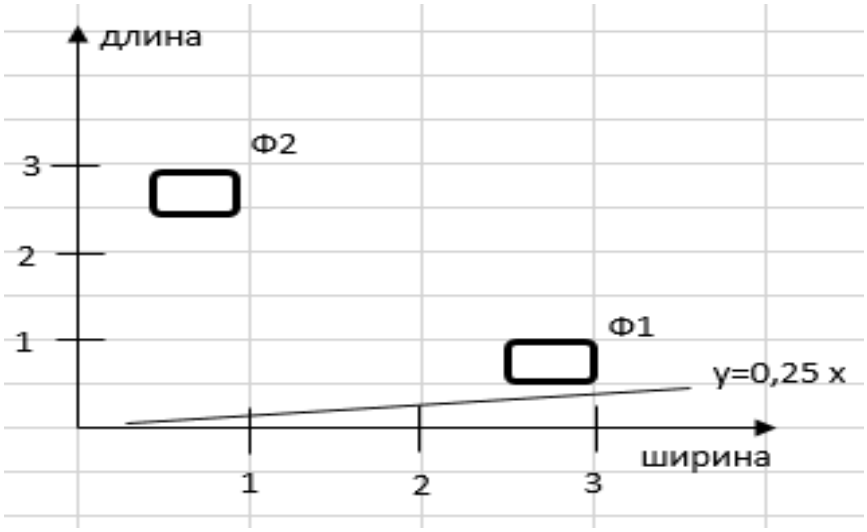


Рис.2.67. Условная разделительная линия классификатора объектов

Условная разделительная линия классификатора представляется в виде:

$$y = Ax$$

Это разделительная линия классификатора, а не зависимость между длиной и шириной. Параметр А управляет наклоном разделительной линии. В представленном варианте рис.2.67 выбран первоначальный (пробный) вариант наклона линии: А=0,25. Однако, при таком варианте линия не отделяет один тип объектов от другого, а решение будет иметь вид:

$$y = (0.25) * (3,0) = 0,75.$$

То есть функция обозначает, для объекта Ф1 шириной 3,0 длина должна составлять 0,75 (на самом деле длина объекта **y** =1,0). Налицо ошибка в выборе величины А. Нам желательно, чтобы линия классификации проходила над точкой 3,0;1,0, чтобы точки данных объекта Ф1 лежали под линией. В связи с этим можно выбрать целевое значение длины объекта y =1,1 и определяем ошибку Е с помощью формулы: ошибка = желаемое целевое значение + фактический результат.

После подстановки значений:

$$E = 1,1 – 0,75 = 0,35.$$

Мы хотим использовать ошибку в значении y, которую назвали

E, для нахождения искомого изменения параметра A. Для этого нам нужно знать, как эти две величины связаны между собой. Каково соотношение между A и E ? Если бы это было нам известно, то мы могли бы понять, как изменение одной величины влияет на другую.

Пусть t — корректное, расчетное целевое значение. Чтобы получить его, мы должны ввести в **A** небольшую поправку. Для таких поправок в математике принято испольать символ $\triangle$ **,** означающий "небольшое изменение" .

Запишем соответствующее уравнение разделительной линии:

$$\mathbf{t} = (A + \Delta A)\, x$$

Ошибку E мы определили как разность между желаемым корректным значением y и расчетным значением, полученным для текущего пробного значения A. Таким образом, E = t - y. Запишем это в явном виде:

$$t - y = (A+\Delta A)\, x - Ax$$

Раскрыв скобки и приведя подобные члены, получаем:

$E = t - y = Ax + (\Delta A)\, x - Ax$ $E = (\Delta A)\, x$,

тогда $\Delta A = E / x$

Ошибка E связана с ΔA очень простым соотношением, что значительно упрощает работу. Теперь мы можем использовать ошибку E для изменения наклона классифицирующей линии на величину ΔA в нужную сторону.

Когда x был равен 3,0, ошибка была равна 0,35. Таким образом, формула $\Delta A = E / x$ превращается в 0,35 / 3,0 = 0,1167. Это означает, что текущее значение A = 0,25 необходимо изменить на величину 0,1167. Отсюда следует, что новое, улучшенное значение A равно **(**A + ΔA) т.е. 0,25 + 0,1167 = 0,3667. Не составляет труда убедиться в том, что расчетное значение **y** при новом значении A равно, как и следовало ожидать, 1,1 — желаемому целевому значению.

Продолжая наш пример обратимся к фигуре Ф2. Он дает нам следующие истинные данные: x = 1,0 и y = 3,0 . Посмотрим, что получится, если вставить x = 1,0 в линейную функцию, в которой теперь используется обновленное значение A = 0,3667. Мы получаем y = 0,3667 *1,0 = 0,3667. Это очень далеко от значения y = 3,0 в тренировочном примере.

Используя те же рассуждения, что и перед этим, когда мы искали путь к построению такой линии, которая не пересекала бы тренировочные данные, а проходила над ними или под ними, мы можем задать желаемое целевое значение равным 2,9. При этом данные тренировочного примера, соответствующего Ф2, находятся над линией, а не на ней. Тогда ошибка равна E = (2,9 - 0,3667) = 2,5333.

Опять обновим A, как делали до этого. Соотношение $\Delta A = E / x$ дает нам 2,5333/1,0 = 2,5333. Это означает, что после очередного обновления параметр A принимает значение 0,3667 + 2,5333 = 2,9. Отсюда следует, что для x = 1,0 функция возвращает в качестве ответа значение 2,9, которое и является желаемым целевым значением (рис.2.68)

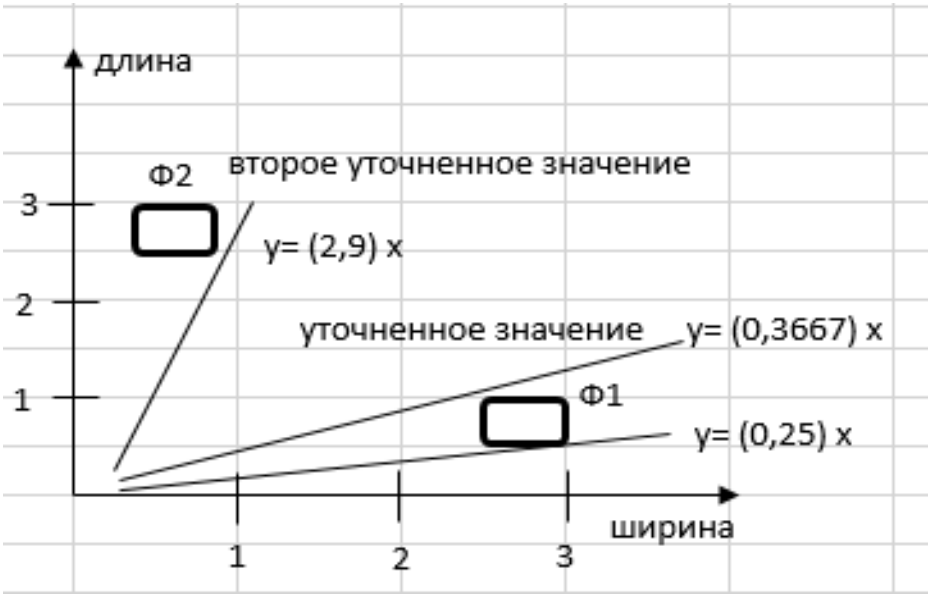


Рис.2.68. Значения разделительной линии классификации

Нам не удалось добиться того наклона прямой, которого мы хотели. Она не обеспечивает достаточно надежное разделение областей диаграммы, занимаемых точками данных объектов Ф1 и Ф2, потому что линия обновляется, подстраиваясь под то целевое значение **y,** которое мы задаем. Если мы будем продолжать так и далее, т.е. просто обновлять наклон для очередного примера тренировочных данных, то все, что мы будем каждый раз получать в конечном счете, — это линию, проходящую вблизи точки данных последнего тренировочного примера. В результате этого мы **о**тбрасываем весь предыдущий опыт обучения, который могли бы использовать, и учимся лишь на самом последнем примере. Исправить

эту ситуацию можно, эта идея играет ключевую роль в машинном обучении**.** Необходимо сглаживать обновления, т.е. немного уменьшать величину поправок.

Вместо того чтобы каждый раз заменять А новым значением, мы используем лишь некоторую долю поправки ΔА, а не всю ее целиком. Благодаря этому мы движемся в том направлении, которое подсказывает тренировочный пример, но делаем это осторожно, сохраняя некоторую часть предыдущего значения, которое было получено в результате, возможно, многих предыдущих тренировочных циклов. У такого сглаживания есть еще один полезный эффект. Если тренировочные данные не являются надежными и могут содержать ошибки или шум (а в реальных измерениях обычно присутствуют оба этих фактора), то сглаживание уменьшает их влияние.

Сделаем перерасчет, добавив сглаживание в формулу обновления:

$$\Delta A = L\,(E/x)$$

Фактор сглаживания, обозначенный здесь как L**,** часто называют коэффициентом скорости обучения.

Выберем L= 0,5 в качестве начального приближения, используем поправку вдвое меньшей величины, чем без сглаживания.

Повторим все расчеты, используя начальное значение А=0,25**.** Первый тренировочный пример дает нам **y** = O,25x3,0 = 0,75. При целевом значении 1,1 ошибка равна 0,35. Поправка равна ΔА = L (E/*x*) **=** 0,5x0,35/3,0 = 0,0583. Обновленное значение А равно 0,25 + 0,0583 = 0,3083. Проведение расчетов с этим новым значением А для тренировочного примера при *x* = 3,0 дает *y* = 0,3083 x 3,0 = 0, 9250. Как видим, расположение этой линии относительно тренировочных данных оказалось неудачным — она проходит ниже значения 1,1, но была всего лишь первая попытка. Главное то, что мы движемся в правильном направлении от первоначальной линии.

Перейдем ко второму набору тренировочных данных при *x*=1,0**.** Используя А=0,3083, мы получаем **y** = 0,3083x1,0 = 0,3083. Желаемым значением было 2,9**,** поэтому ошибка составляет

(2,9 - 0,3083**)** **=** 2,5917**.**

Поправка ΔА = L (E/*x*) = 0,5x2,5917/1,0 = 1,2958. Теперь обновленное значение А равно 0,3083 + 1,2958 = 1,6042**.**

Отображение на диаграмме начального, улучшенного и окончательного вариантов линии говорит о том, что сглаживание обновлений приводит к более удовлетворительному расположению разделительной линии между областями данных объектов Ф1 и Ф2 (рис.2.69).

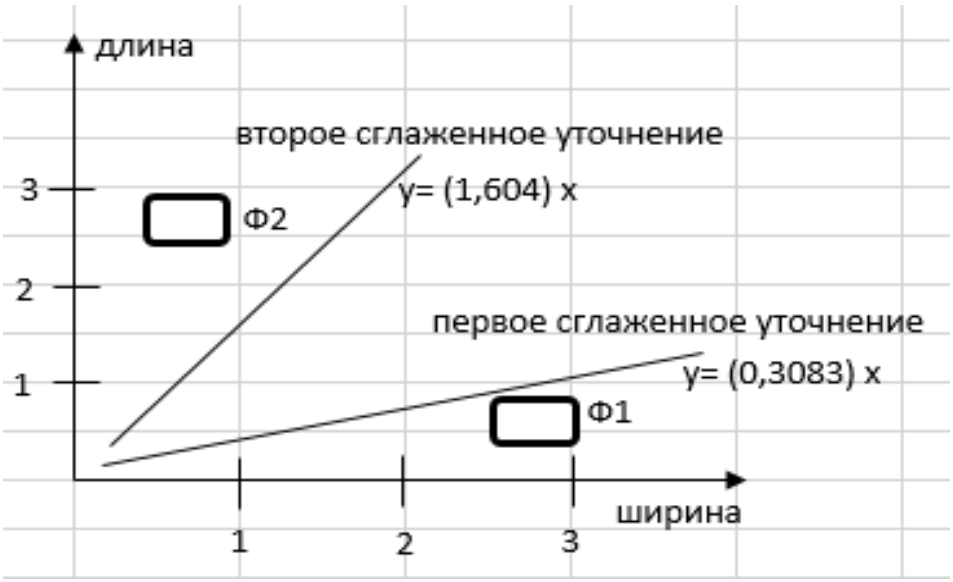


Рис.2.69. Окончательное сглаженное значение линии классификации

Корректировка весовых коэффициентов при обучении

Улучшение показателей простого линейного классификатора выполнялось путем регулирования параметра наклона линейной функции узла, при этом использовалась текущей величиной ошибки, т.е. разности между ответом, вырабатываемым узлом, и известным истинным значением. Когда на выходной узел поступал сигнал только от одного узла, все было намного проще, задача усложняется когда необходимо использовать величину ошибки выходного сигнала при наличии двух входных узлов.

Основная идея заключается в том, чтобы распределять ошибку между узлами так, чтобы большая доля ошибки приписывалась вкладам тех связей, которые имеют больший вес. Это делается потому, что они оказывают большее влияние на величину ошибки. Идея адаптивного распределения ошибки может использоваться при большом числе узлов пропорционально их вкладам, размер которых определяется весами связей [49].

Изменение весов коэффициентов используется для распространения ошибки в обратном направлении (обратное распространение ошибки).

На рис.2.70 представлена простейшая сеть с двумя входными и двумя выходными узлами.

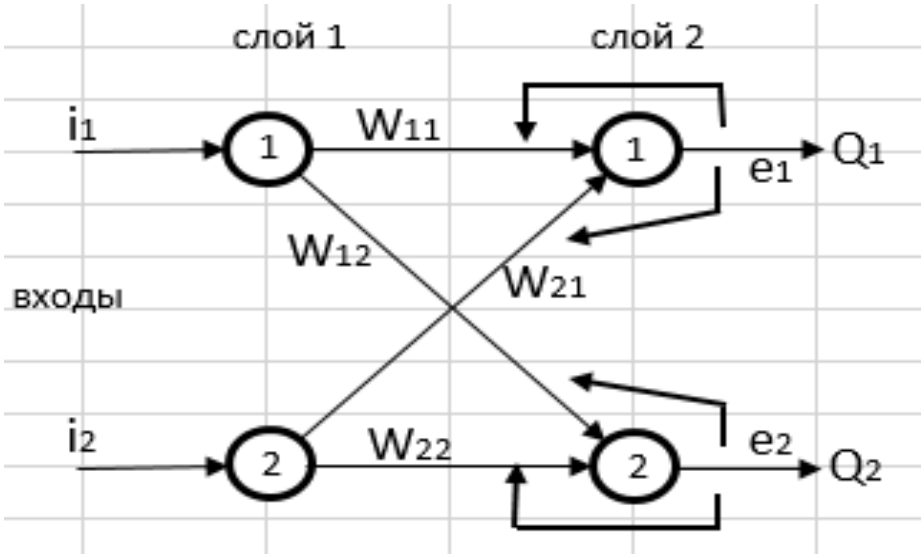


Рис.2.70. Сеть с двумя входными и двумя выходными узлами Q1, Q2 – выходные узлы, e1, e2 – выходные ошибки.

Ошибка может формироваться на обоих узлах — фактически эта ситуация очень похожа на ту, которая возникает, когда сеть еще не обучалась. Для коррекции весов внутренних связей нужна информация об ошибках в обоих узлах. Можно распределять ошибку выходного узла между связанными с ним узлами пропорционально весовым коэффициентам соответствующих связей. Это объясняется тем, что связи одного выходного узла не зависят от связей другого. Между этими двумя наборами связей отсутствует какая-либо зависимость.

На диаграмме видно, что ошибка e1 распределяется пропорционально весам связей, обозначенным как W**11** и W**21.** Точно так же ошибка e**2** должна распределяться пропорционально весам W**12** и W**22.** Чтобы у вас не возникало никаких сомнений в правильности получаемых результатов, запишем эти доли в явном виде. Ошибка e**1** информирует о величинах поправок для весов W**11** и W**21**. При ее

распределении между узлами доля е**1,** информация о которой используется для обновления W**11**, определяется следующим выражением:

W11 / W11 + W21.

Доля **е**1**,** используемая для обновления W**21,** определяется аналогичным выражением:

W21/ W11 + W21.

Узлы, сделавшие больший вклад в ошибочный ответ, получают больший сигнал об ошибке, тогда как узлы, сделавшие меньший вклад, получают меньший сигнал.

Например: если W1**1** = 2W **1,** то доля е**1** для W**11** составляет 2/3, а доля е**1** для W**21** составляет 1/3. Как и следует ожидать, при равных весах будут равны и соответствующие доли. Рассуждая аналогичным образом, можно получить выражения и для второго выхода Q**2** и его ошибки е**2.** Доля е**2** для W**12** составляет W12/W12 + W22. Доля е**2** для W**22** составляет W22/W22 + W12. Таким образом, в результате всех вычислений для сети с двумя входами получены выражения для коэффициентов обратного распространения от обеих выходов е**1** и е**2**.

Обратное распространение ошибок при большем числе слоев

Продвигаясь в обратном направлении от последнего, выходного слоя (крайнего справа), видно, как информация об ошибке в выходном слое, используется для определения величины поправок к весовым коэффициентам связей, со стороны которых к нему поступают сигналы. При наличии большего количества слоев описанная процедура повторяется для каждого слоя, продвигаясь в обратном направлении от последнего, выходного слоя. Идея распространения этого потока информации об ошибке (сигнала об ошибке) достаточно понятна.

Следует помнить, у нас нет целевых или желаемых выходных значений для скрытых узлов. Мы располагаем лишь целевыми значениями узлов последнего, выходного слоя, и эти значения приходиться использовать.

На рис.2.71. представлена сеть с двумя входами и тремя слоями.

Введены следующие обозначения

е вых 1,2 – ошибки выходного слоя сети;

е скр 1,2 – ошибки скрытого слоя;

W вс – связь между входным и скрытым слоем; Wсв – связь между скрытым и входным слоем.

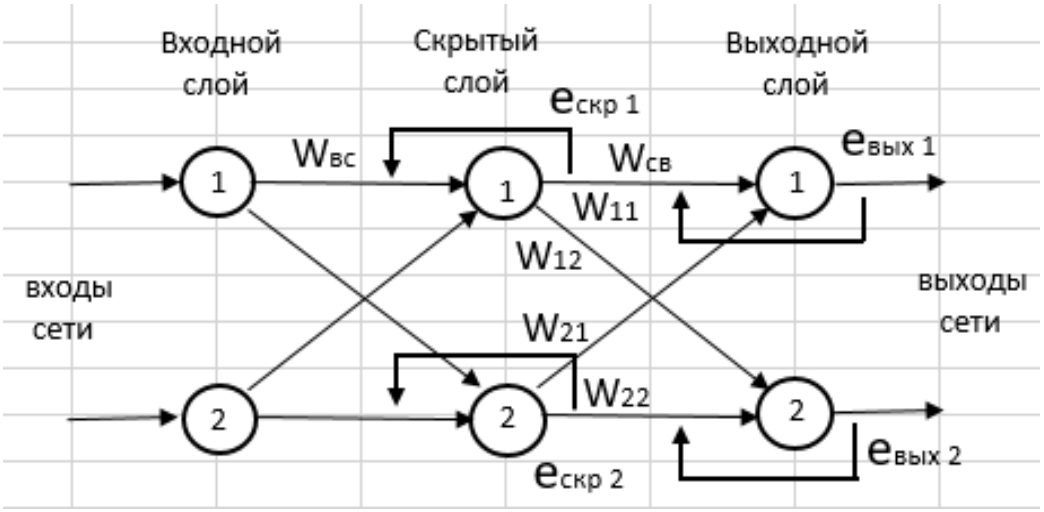


Рис.2.71 Сеть с двумя входами и тремя слоями

Вычисляются конкретные ошибки каждой связи путем ее распределения пропорционально весам. Ошибка на первом скрытом узле представляет собой сумму ошибок, распределенных по всем связям, исходящим из этого узла в прямом направлении [30].

На данной схеме имеется доля выходной ошибки **евых1**, приписываемая связи с весом W**11**, и некоторая доля выходной ошибки W**вых 2**, приписываемая связи с весом W**12.**

Как и раньше, значения ошибок евых**1** и евых**2** равно разнице между целевым и фактическим значениями выходов сети.

Вышесказанное можно записать в виде следующего выражения:

е**скр1=** сумма ошибок, распределенных по связям

W**11** и W**12** =

евых1 * (W11/ W11 + W21) + евых 2 * (W12/ W12 + W22).

Аналогично:

ескр**2** = сумма ошибок, распределенных по связям

W**12** и W**22** =

евых 1 * (W12/ W12 + W22) + евых 2 * (W22/ W22+ W12).

Нейронные сети обучаются посредством уточнения весовых

коэффициентов своих связей. По сути это процесс приобретения знаний. Этот процесс управляется ошибкой — разностью между правильным ответом, предоставляемым тренировочными данными, и фактическим выходным значением.

Ошибка на выходных узлах определяется простой разностью между желаемым и фактическим выходными значениями.

Величина ошибки, связанной с внутренними узлами, не столь очевидна. Одним из способов решения этой проблемы является распределение ошибок выходного слоя между соответствующими связями пропорционально весу каждой связи с последующим объединением соответствующих разрозненных частей ошибки на каждом внутреннем узле.

Пример вычисления ошибок скрытых слоев

Чтобы проиллюстрировать, как эта теория выглядит на практике, приведем схему (рис.2.72), демонстрирующую обратное распространение ошибок в простой трехслойной сети на примере конкретных данных.

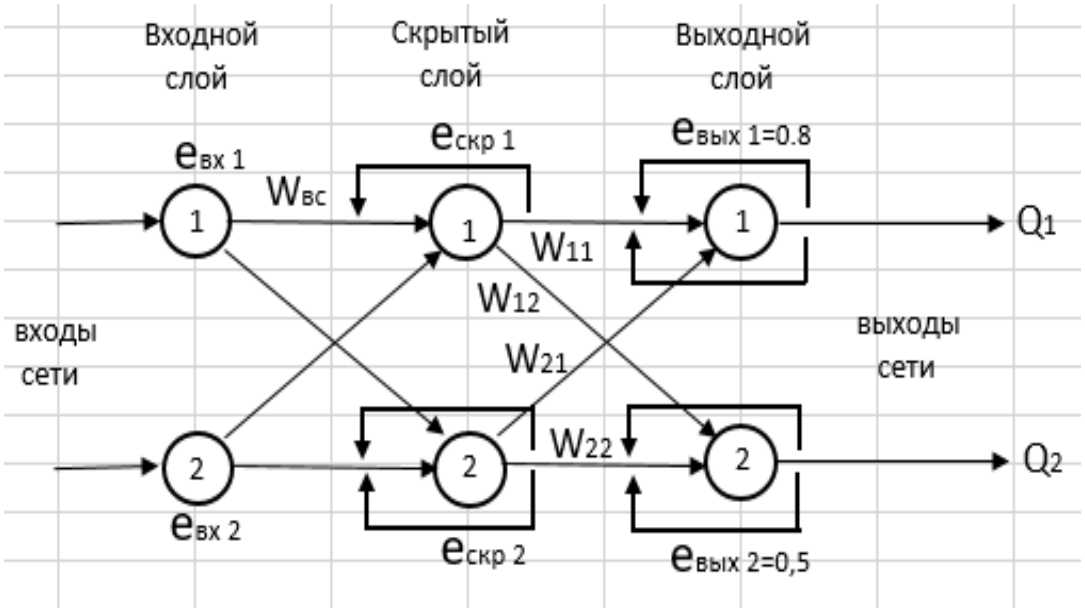


Рис.2.72. Конкретный пример с трехслойной сетью

Исходные данные, взятые для связи первого выходного и скрытого слоев.

Заданные веса связей верхнего узла:

евых 1 = 0,8; W11 = 2,0; W21 = 3,0.

Вычисление ошибок скрытого слоя:

ескр 1 = евых 1 х (W11/ W11 + W21) + евых 2 х (W12/ W12 + W22) = 0,8 х (2,0/5,0) + 0,5 х (1,0/5,0) = 0,32 + 0,1 = 0,42.

Заданные веса связей нижнего узла:

евых2 = 0,5; W12 = 1,0; W22 = 4,0.

Вычисление ошибок скрытого слоя:

ескр2 = евых 1 х (W12/ W12 + W22) + евых 2 х (W22/ W22+ W12) = 0,8 х (3,0/5,0) + 0,5 (4,0/5,0) = 0,48 + 0,4 = 0,88.

Проделав аналогичные расчеты можно получить величины ошибок для входных узлов сети евх1, евх2.

Архитектуры нейронных сетей на основе персептрона

Ранее была представлена классификация нейронных сетей, включающая все основные виды современных архитектур с учетом особенностей обучения и применения. В настоящее время существует большое количество разнообразных по построению и применению нейронных сетей. Изучение их всех их заняло бы очень много времени, те более, что многие из них являются модификациями известных, широко применяемых архитектур [41-50].

Ранее были подробно рассмотрены вопросы построения, функционирования и обучения многослойного персептрона. Это одна из базовых архитектур искусственных нейронных сетей, на примере которой можно объяснять принципы организации, обучения и моделирования вычислительного процесса. Поэтому далее будут рассмотрены архитектуры многослойных нейронных сетей на базе персептрона: рекуррентные, сверточные, автоэнкодерные нейронные сети.

Рекуррентные нейронные сети

Рекуррентными нейронными сетями называются такие сети, в которых выходы нейронных элементов последующих слоев имеют синаптические соединения с нейронами предшествующих слоев. Это приводит к возможности учета результатов преобразования

нейронной сетью информации на предыдущем этапе для обработки входного вектора на следующем этапе функционирования сети [19,31,32].

В нейросетях прямого распространения информация передается только вперед по сети, от слоя к слою. Недостатком таких сетей является сложность анализа временных последовательностей, особенно при решении задач обработки звуков, речи, текста. В этих приложениях важно знать временные последовательности, например, при переводе текста нужно знать не только сами слова, но и их порядок следования.

Для задач такого типа били разработаны сети другого класса - рекуррентные нейронные сети (Recurrent Neural Network - RNN). В рекуррентных нейросетях нейроны обмениваются информацией между собой: вдобавок к новой порции входящих данных нейрон также получает некоторую информацию о предыдущем состоянии сети. Таким образом в сети реализуется «память», что принципиально меняет характер ее работы и позволяет анализировать любые последовательности данных, в которых важно, в каком порядке идут значения [43,44].

Наряду с сетью Хопфилда (1982г.) первой современной рекуррентной стала сеть Д.Элмана (1990г.). Они имеют следующую структуру.

Сеть содержит входы, рекуррентный слой и выходы. В рекуррентном слое на входы нейронов подаются как входной вектор данных, так и информация, вычисленная на предыдущем шаге и задержанная на один такт (Z^{-1}).

В сети Элмана функция активации нейронов – линейная, имеется один скрытый слой с набором обратных связей. Именно эти обратные связи образуют рекуррентность сети. На основе такой схемы создаются простейшие рекуррентные сети.

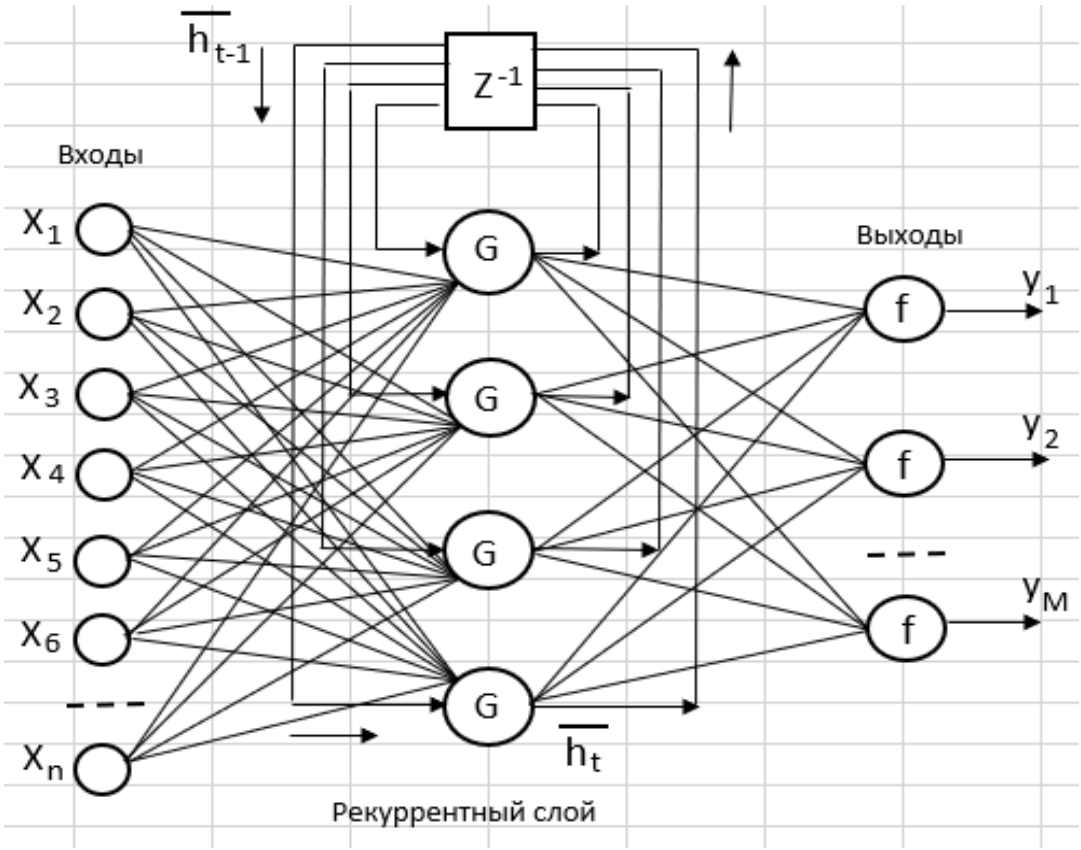


Рис.2.73. Рекуррентная нейронная сеть

Упрощенное представление такой сети выглядит следующим образом:

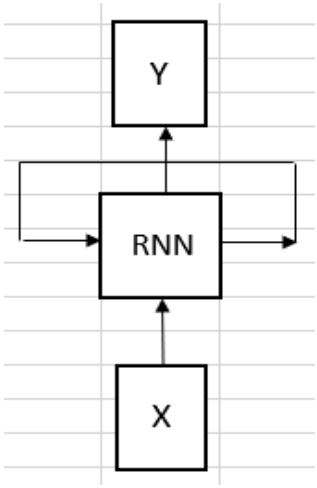


Рис.2.74. Упрощенная графическая схема рекуррентной сети

На рисунке 3.74 обозначены: X – вектор входных данных, Y – выходные значения рекуррентной сети, RNN – рекуррентный слой с обратной связью. Математическая модель такой сети представляется следующим образом:

$$ht = G (h\ t\text{-}1, xt).$$

Выходной сигнал слоя представляется через функцию активации G, а на вход G подается предыдущее состояние рекуррентной сети h_{t-1} плюс выходной сигнал x_t в момент времени t. В качестве начального условия произвольно выбирается:

$$h_0 = [0,0,0,.....,0_N]^T$$

В качестве начального условия для RNN сетей принято использовать нулевой вектор. Для каждого шага итерации зная выходные значения h_t для каждого входного вектора x_n можно вычислить соответствующие выходные значения:

$$yt = Nhy * ht$$

Так как функция активации линейна вектор выходных значений определяется как произведение матрицы весовых коэффициентов N_{hy} на выходные значения нейронов рекуррентного слоя. В результате получается вектор y_t. В качестве функции активации могут использоваться не только линейные, но и нелинейные: гиперболический тангенс, сигмоида, softmax.

Представим вычислительную структуру в виде графа.

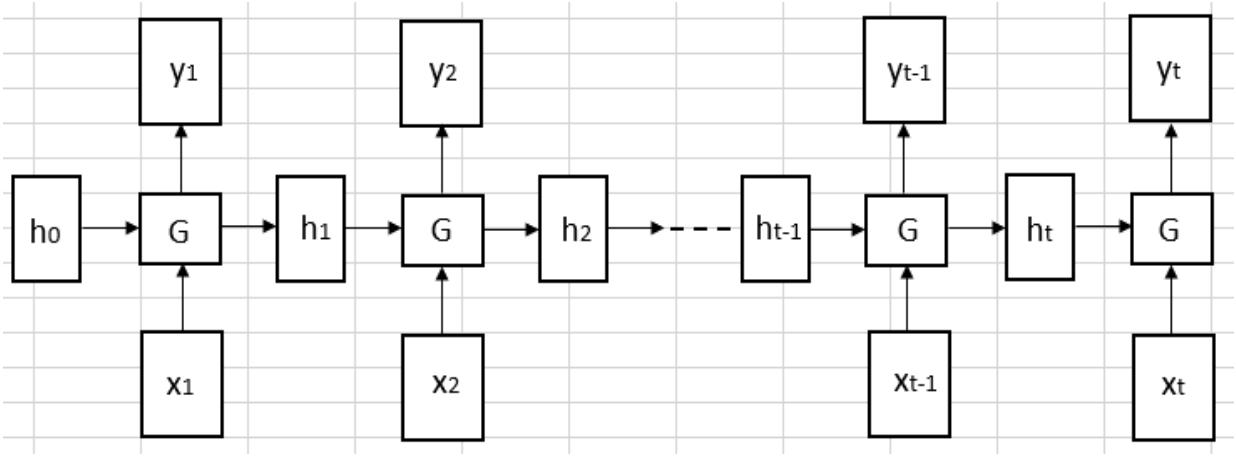


Рис.2.75. Графическое представление рекуррентной сети

Когда множество входных векторов X1,2,..., Xt-1, Xt соответствует множеству выходных значений Y1,2,..., Yt-1, Yt, технология построения рекуррентной сети называется «Many to Many».

Используя графическое представление можно построить и другие архитектуры, например архитектуру «Many to One»:

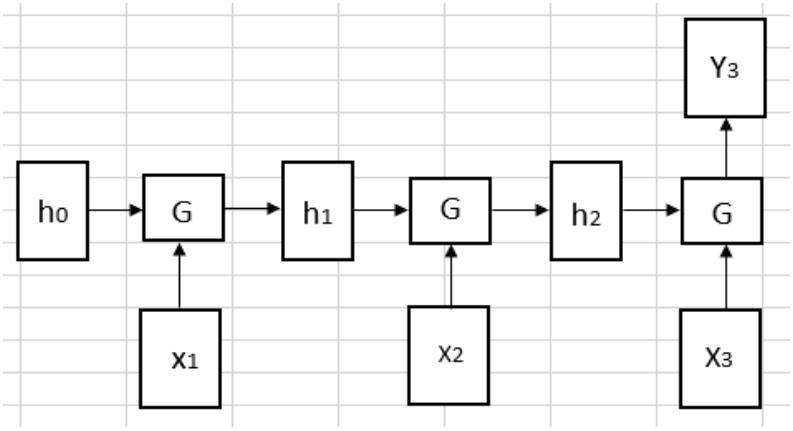


Рис.2.76. Представление архитектуры «Many to One»

В данном варианте архитектуры рекурсия проходит 3 раза и только потом, один раз вычисляется выходное значение.

Если на вход подается один вектор данных X_1 , а на выходе в результате рекурсии получается три выходных значения Y_1, Y_2, Y_3, то такая архитектура называется «One to Many»:

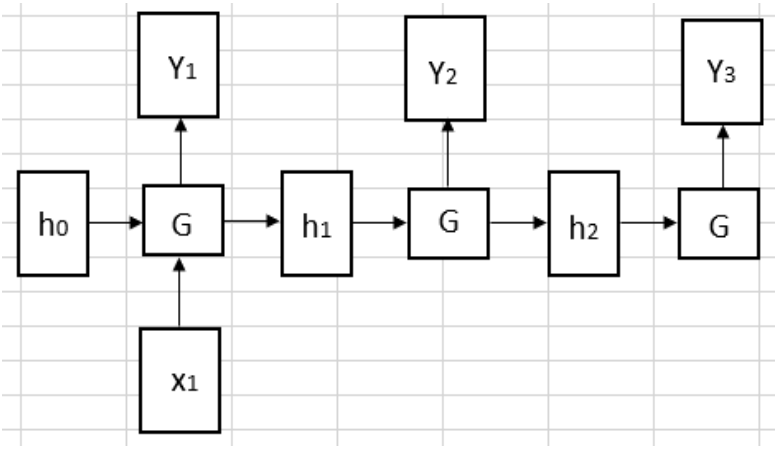


Рис.2.77. Представление архитектуры «One to Many»

В данном варианте на каждой итерации рекурсии вычисляются выходные значения.

Выбор той или иной архитектуры зависит от типа решаемой сетью задачи:

- для перевода тестов используется «Many to Many»;
- для эмоциональной окраски произносимого диктором текста (вход-текст, выход – положительный, отрицательный, нейтральный) используется «Many to One»;
- для генерации описания изображений, когда на вход подаются карты признаков изображения, а на выходе его детальное описание (например, текст), используется архитектура «One to Many».

Обучение рекуррентной сети может осуществляться известным алгоритмом «Backpropagation Through Time - BPTT». Задача обучения рекуррентной сети более трудоемкая в вычислительном плане и требует больший объем памяти, чем для сетей прямого распространения.

Сверточные нейронные сети

Сверточные нейронные сети (convolutional neural networks, CNN) являются дальнейшим развитием многослойного персептрона и неокогнитрона и широко используются для обработки изображений. В отличие от многослойного персептрона, сверточные нейронные сети позволяют учитывать топологию изображений и инвариантны к сдвигам, масштабированию, смещению, поворотам, смене ракурса и другим искажениям входного образа [41]. На данный момент сверточная нейронная сеть и ее модификации считаются лучшими по точности и скорости алгоритмами нахождения объектов на изображении.

Сверточная нейронная сеть состоит из разных видов слоев: сверточные (convolutional) слои, слои подвыборки (subsampling) и слои обычной нейронной сети – персептрона. Первые два типа слоев (convolutional, subsampling), чередуясь между собой, формируют входной вектор признаков для многослойного персептрона.

Первые модели сверточных нейронных сетей были предложены для решения задач классификации графических образов. Набольшего успеха сверточные нейронные сети достигли при обработке изображений [45,46,47].

Рассмотрим построение и принципы функционирования

сверточных нейронных сетей. В качестве обобщенного образца рассмотрим процесс обработки изображения.

Процедура свертки. Входной сигнал изображения подается на вход нейрона только в пределах ограниченной (как правило квадратной) области представления (рис.2.78). Из общего изображения, в данном примере размером 7x7 квадратов, в начале выбирается крайний слева участок размером 3x3 (оба размера выбираются произвольно из условия задачи и общего объема данных). Для простоты данные представлены в виде битов информации в квадратах одинакового размера.

Каждая маска 3x3 подается на соответствующий нейрон, затем эта маска смещается вправо на один шаг и подается на второй нейрон. Происходит сканирование изображения с шагом 1 по горизонтали. Операция сканирования заключается в перемножении одноименных позиций маски и весовых коэффициентов нейрона (wi,j) и их сложении. Для показанной на примере позиции сумма по одной фиксированной будет равна:

$$(1x1+0x0+0x1) + (1x1 +1x0+0x1) + (1x1+1x0+1x1) = 1=1+1+1 = 4$$

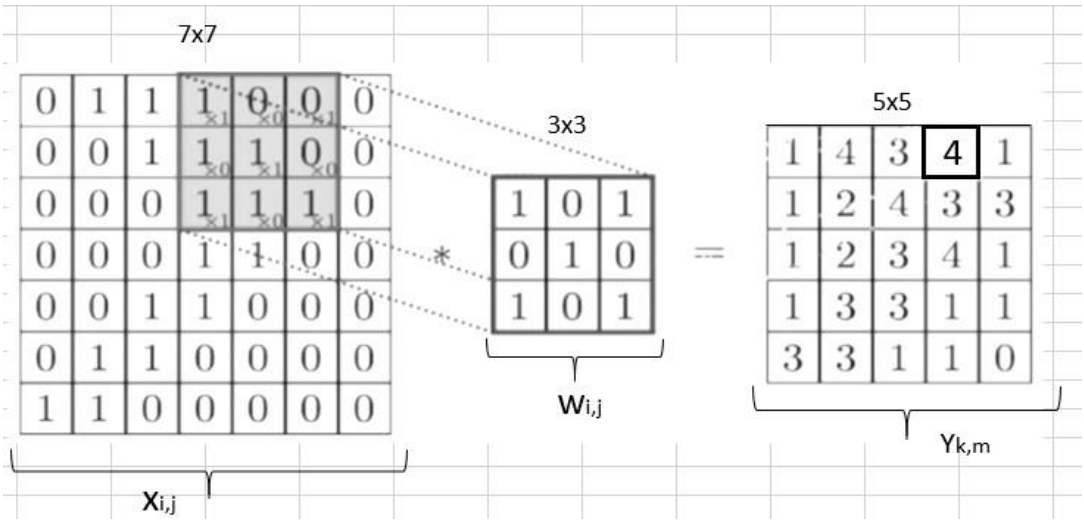


Рис.2.78. Схема процесса сканирования изображения

При сканировании крайние позиции получаются неполными и не учитываются, получается в одном ряду 5 смещений позиции маски. Затем маска смещается вниз и процесс сканирования повторяется для второго ряда пикселей изображения.

Один проход по изображению позволяет охватить все нейроны первого скрытого слоя и сформировать выходные значения первой группы нейронов. При этом весовые коэффициенты берутся одинаковыми для всех нейронов первой группы, т.е. значения внутри матрицы 3x3 для всех позиций одного прохода будут постоянными. К каждой сумме добавляется смещение +1xw0.

После процесса первого сканирования всего изображения процесс сканирования повторяется, но уже с другим набором весовых wi,j. в матрице 3x3. Этот набор подается на другую группу нейронов. Так формируются все остальные группы нейронов со своими весовыми коэффициентами. Они идентичны и отличаются только весами коэффициентов Wk.

Так как при перемещении окна весовые коэффициенты wi,j остаются неизменными, общее число настраиваемых параметров в пределах одной группы нейронов в нашем случае равно 3x3+1=10 (вместе со смещением). Так как имеется несколько групп (n), то получается 10n - число настраиваемых параметров.

В задачах цифровой обработки сигналов сумма умножений со сдвигом называется сверткой, а окно с коэффициентами wi,j называется импульсным откликом фильтра или ядром.

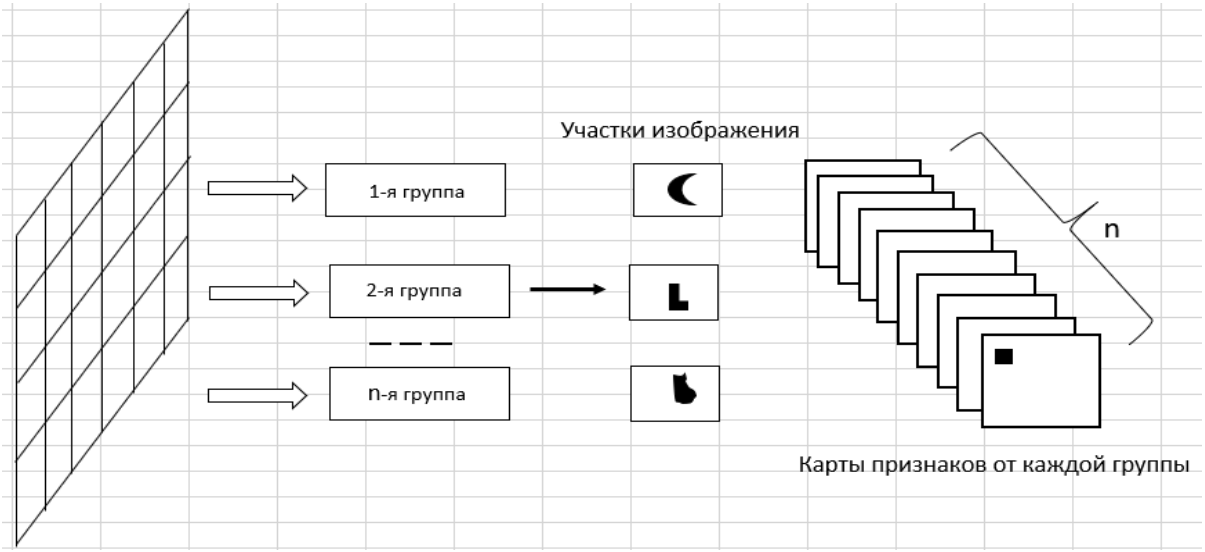


Рис.2.79. Формирование карт признаков групп

Таким образом, процедура свертки с использованием ядра в виде группы весов (в данном примере 3x3) позволяет выделить характерные участки на исходном изображении в соответствии с конфигурацией весовых коэффициентов ядра.

Благодаря такому подходу нейроны каждой группы активируются на том участке изображения, где появляется фрагмент, подходящий под их ядро. Получается n групп, и каждая группа имеет свое ядро, которое выделяет свои особенные, характерные закономерности на изображении при его сканировании. В результате получается карта признаков, которые называются каналами (изображены в виде квадратов). Число каналов соответствует группе нейронов. Изображенный черный квадратик – это выходное значение каждого нейрона из соответствующей группы.

Значимые величины в каждой карте показывают наличие признака в строго определенном месте изображения. Используя дальше карты признаков следующая группа нейроном может их обрабатывать и делать дальнейшие обобщения и выделять признаки сложных фигур: эллипсов, прямоугольников. Выходные значения карт признаков Yk,m определяются в соответствии с формулой, которая приведена выше. В формуле сумма получается на каждом нейроне и эта сумма пропускается через функцию активации. В результате
получается такое выходное значение:

$$Qk,m = f(Yk,m).$$

Это и есть отдельные значения на карте признаков.

Рассмотрен простейший случай плоского, черно-белого изображения. Если изображение будет цветным с тремя компонентами R,G,B, тогда каждый цвет обрабатывается отдельным фильтром и для каждой цветной компоненты образуется своя карта признаков. Затем они складываются между собой поэлементно, после чего сумма пропускается через функцию активации.

Так выглядит в общем обработка на первом скрытом слое сверточной нейронной сети.

Следующие слои работают по такому же принципу. На вход следующего слоя сверточной нейронной сети подается многоканальное (n-каналов) изображение, которое представляет собой n-карт признаков и оно обрабатывается так же. Каждая карта признаков прогоняется через

свой фильтр и суммируется, добавляется +1 и все это подается на вход нейрона следующего слоя.

После обработки каждый канал формирует свое новое изображение намного меньшего размера. Если исходное изображение имеет размер 128x128, то после того, как фильтр масками 3x3 пройдет через все изображение, на выходе получится карта признаков размером 127x127. Это потому, что граничные значения не захватываются. Целесообразнее было бы иметь на выходе тот же самый начальный размер 128x128.

Для этого добавляются новые значения по краям, эти граничные значения приравниваются нулю, центр маски размещается на первым пикселем изображения. Далее смещаясь по изображению на выходе получается размерность 128x128. Это если смещение идет по одному шагу изображения. Если смещение идет по два шага (2 клетки сразу), то выходные карты признаков будут иметь размер:

(128:2) x (128:2) = 64x64

Операция пулинга. Кроме таких непосредственных (традиционных) сверток в таких сетях применяется еще одна очень важная и широко применимая операция, которая называется «пулинг» (Pooling – изменение масштаба).

Сушествует 3 типа операций пулинга:

- Max Pooling – отбор наибольших значений;
- Min Pooling – отбор наименьших значений;
- Average Pooling – отбор средних значений.

На рис.3.80. показан процесс реализации операции пулинга. Представлен вариант карты признаков 6x6. Проводится анализ карты признаков непересекающимися окнами размером 2x2. Процесс смещения, в отличие от сверточного процесса, идет отдельными, не связанными шагами. В каждом из окон отбирается максимальное значение (33,88,65 и т.д.).

Вся карта признаков прогоняется такими окнами и в каждом окне выбирается максимальный элемент. В итоге карта признаков уменьшается в два раза и получается новая карта признаков меньшего масштаба. При использовании других операций пулинга (Min Pooling, Average Pooling) масштабы сохраняются, изменяются только значения чисел в клетках

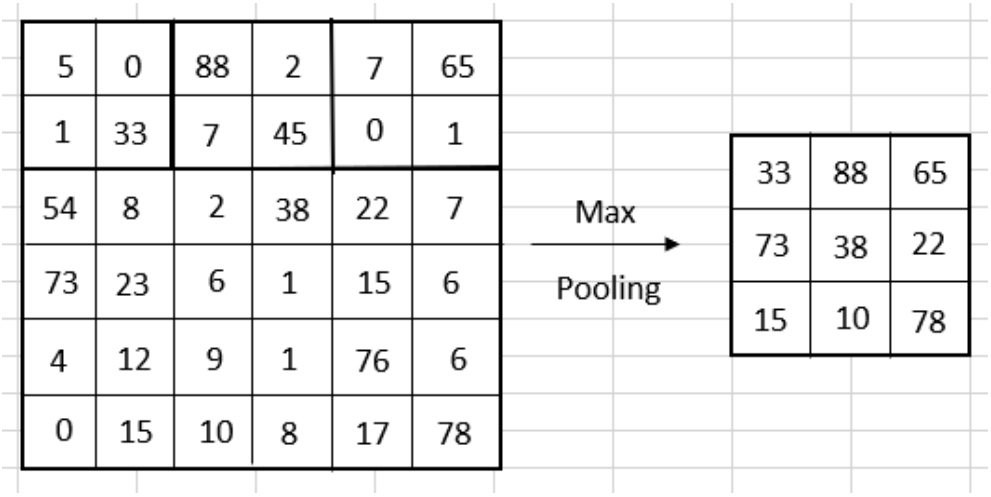


Рис.2.80. Реализация операции пулинга (вариант Max Pooling)

Операция Max Pooling позволяет анализировать изображение на разных масштабах. Чем крупнее масштаб, тем более общие детали подвергаются анализу. Благодаря этому нейроны следующего слоя способны выделить более общие признаки изображения. Большие значения чисел соответствуют наличию информативного признака. Отбор максимальных чисел позволяет сохранять ранее найденные информационные признаки для дальнейшего анализа на более крупном масштабе.

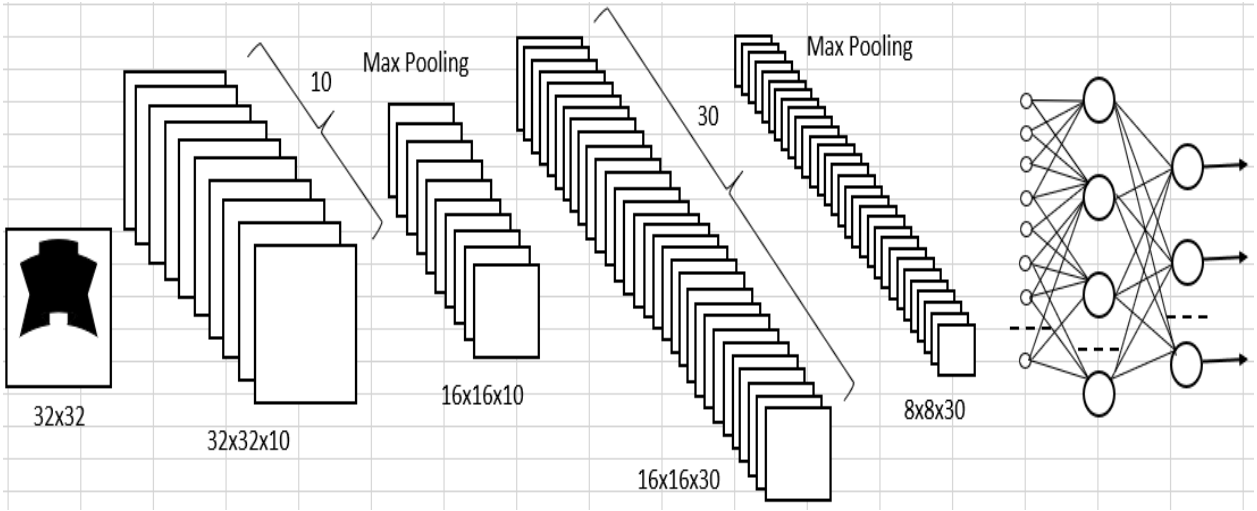


Рис.2.81. Построение сверточных сетей

Общая архитектура сверточной нейронной сети. Для упрощения в качестве примера (рис.2.81) предполагается, что на вход нейронной сети подается изображение размером 32x32 пиксела. Оно проходит через 10 каналов первого скрытого слоя. В результате получается карта признаков размером 32x32x10. Далее по условию задачи весь этот тензор (карта признаков) пропускается через процедуру Max Pooling, и на выходе получается карта размером 16x16x при 10 каналах. Размер 32x32 уменьшен в 2 раза. Max Pooling может уменьшать и в 64 и в 8 раз, все зависит от размеров окон и шагов смещения.

Далее для анализа берется полученный тензор и воспринимается как многоканальное изображение, которое анализируется с помощью 30 разных групп нейронов, т.е. получается на выходе 30 каналов. Для каждой карты признаков применяется свой фильтр, затем они все суммируются, прогоняются через нейроны соответствующей группы и функции активации и получается выходное значение 16x16x30.

Полученный тензор из 30 каналов пропускается через операцию Max Pooling и получается тензор 8x8x30. Такой процесс выявления информативных признаков и сжатия можно продолжать и далее, пока каждый выходной сегмент не станет размером в один пиксел. Чаще всего, в зависимости от необходимого решения делается остановка на нужном этапе.

После всех этих процедур свертки вычисленная карта признаков подается на вход обычной полносвязной нейронной сети. Таким образом, в задачах классификации [45] конечный этап сверточной сети обычно завершается полносвязной сетью, на выходе которой определяется вероятность принадлежности исследуемого объекта к тому или иному классу.

Автоэнкодерные нейронные сети

В самом простом варианте автоэнкодер – это нейронная сеть, которая характеризуются прямым $Y=f(X)$, так и обратным $X= f^{-1}(Y)$ преобразованием информации (рис.2.82). При прямом преобразовании данных происходит ее сжатие, а при обратном – восстановление. Автоэнкодерные нейронные сети называются также рециркуляционными, автоассоциативными, NPCA (nonlinear principal component analysis) нейронными сетями.

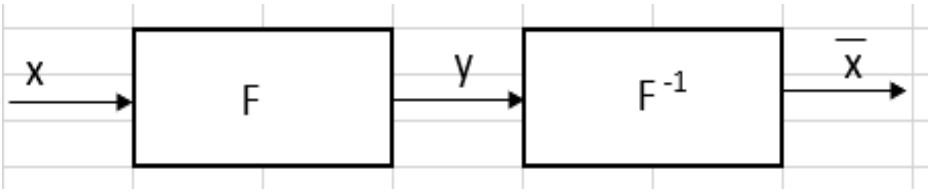


Рис.2.82. Преобразование информации в автоэнкодере

Автоэнкодерные нейронные сети применяются в общем случае для выделения наиболее важных информативных признаков (feature extraction) во входном пространстве образов, сжатия информации, очистки данных от шумов, визуализации и классификации данных [45]. Сжатие информации осуществляется таким образом, чтобы восстановить информацию с наименьшими потерями. Обучение данных сетей производится без учителя в соответствии с критерием достижения минимальной ошибки между входными и выходными данными.

Впервые использование автоэнкодерных нейронных сетей было предложено в 1982 г. в работе [48]. В настоящее время их эволюция происходит по пути увеличения количества слоев (deep autoencoder – глубокий автоэнкодер). Теоретической основой данного класса сетей является метод главных компонент (principial component analysis).

Автоэнкодер сначала кодирует входной сигнал в скрытое состояние, размерность которого меньше размерности входного сигнала, а затем их этого скрытого состояния снова разворачивает (декодирует) данные в другое, новое состояние формируя выходной сигнал. Размеры входных и выходных векторов могут отличаться. В задачах масштабирования и сжатия сигналов декодер должен как можно точнее воспроизвести входное изображение, опираясь только на вектор скрытого состояния. Сам вектор представляет сжатое изображение.

Принцип работы автоэнкодера [43,44] объясняется на рис.10.11.

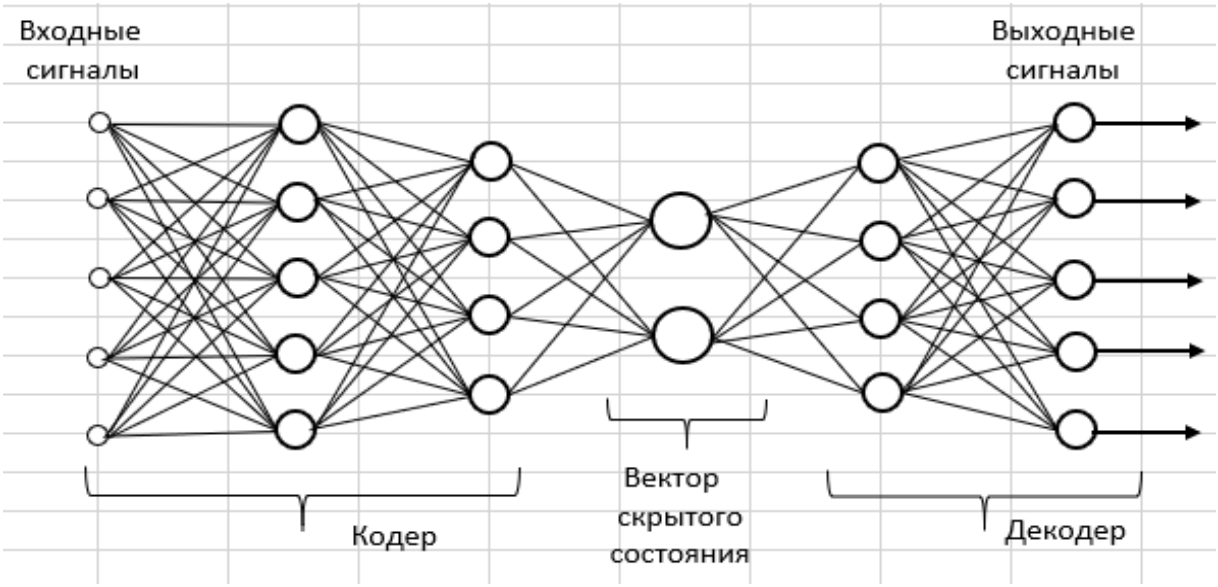


Рис.2.83. Структура автоэнкодерной сети

По архитектуре автоэнкодер похож на персептрон. Автоэнкодеры сжимают входные данные для представления их в latent-space (скрытое состояние), а затем восстанавливают из этого представления входные данные. Цель — получить на выходном слое отклик, наиболее близкий к входному. Отличительная особенность автоэнкодеров — количество нейронов на входе и на выходе совпадает. Автоэнкодер состоит из двух частей:

- кодера, который отвечает за сжатие входа и формирование вектора скрытого состояния и который представлен функцией кодирования $h = F(x)$;
- декодера, предназначенного для восстановления ввода из скрытого состояния и представленного функцией декодирования $h = F^{-1}(x)$.

Одной из самых распространенных архитектур автоэнкодера является нейронная сеть прямого распространения без обратных связей, содержащая входной, скрытый и выходной слои. Данные на входном слое сжимаются на скрытом слое и восстанавливаются на выходном слое, таким образом выделяются «скрытые признаки».

Рассмотрим принцип работы автоэнкодера, где каждый нейрон имеет линейную функцию активации. На рис.2.83 представлена простейшая схема

«кодер-скрытый слой-декодер».

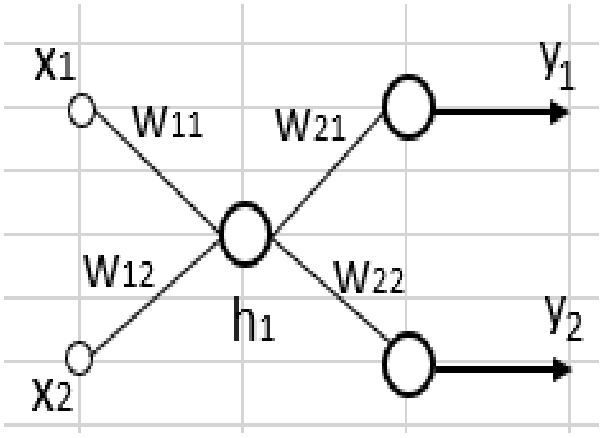


Рис.2.84. Схема простейшего автоэнкодера

Кодер выполняет очень простую операцию – формирует выходное число (скрытое состояние) на основе входных данных x1, x2 используя весовые коэффициенты w1, w2:

$$h1= w11* x1 + w12*x2.$$

А декодер разворачивает значения h1 снова в двумерный вектор.

Математически это можно записать как:

$$y1 = w21 * h1 \; y2 = w22 * h1.$$

Предположим, что значения h1 – это просто сумма двух величин x1, и x2 : h1 = x1 + x2; w11, w12 = 1.

По условиям функционирования автоэнкодера, декодер должен восстановить входные значения. Это можно сделать следующим образом:

$$y1 = 0,5 * h1,$$

$$y2 =0,5 * h1, \quad \text{где } w21, w22 = 0,5.$$

В итоге получается модель представления входных данных скрытого состояния в виде прямой линии, наклоненной под углом в 450.

Пока входные данные $x_1 + x_2$ соответствуют этой модели и лежат на одной линии, то декодер может их восстановить. Как только их положение меняется, например таким образом, что $x_1 = 1$, $x_2 = 3$, тогда получается точка, которая уже не лежит на прямой и не соответствует этой модели. То есть, получается значение скрытого состояния $h_1 = 1+3$

= 4, и декодер вычисляет эту точку со значением $x_1 = 2$, $x_2 = 2$. Тогда на выходе, где должны были бы повториться значения входа, получатся значения $y_1 = 2$ и $y_2 = 2$, хотя на входе было $x_1 = 1$, $x_2 = 3$.

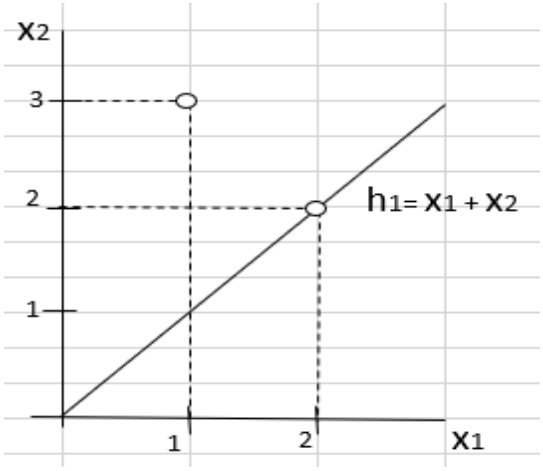


Рис.2.85. Линейная модель представления входных данных

Это принципиальный момент. Вектор скрытого состояния описывает принятую модель представления данных. И чем точнее эта модель описывает входные значения, тем лучше декодер сможет их восстанавливать.

Нейронная сеть с линейной функцией активации может формировать модель только в виде прямой линии (если это одномерный случай) или в виде гиперплоскости (если это двумерный случай). Но используя нелинейную функцию активации, например сигмоиду или RELU, можно сформировать практически любую модель.

Эта модель создается в процессе обучения автоэнкодерной сети.

Как уже отмечалось, автоэнкодерная нейронная сеть может применяться для решения задач классификации и визуализации образов, сжатия и восстановления изображений.

Другие типы рекуррентных сетей на базе персептрона

Сеть RLMP (Recurent MultiLayer Perceptron) - это рекуррентный многослойный персептрон, является одним из простейших способов построения однонаправленной рекуррентной сети с введением в персептронную сеть обратной связи. Она отличается от рассмотренной рекррунтной сети тем, что обработная связь идет не от каждого нейрона, а от одного общего выходного нейрона. Это динамическая сеть,

характеризующаяся запаздыванием входных и выходных сигналов, объединяемых во входной вектор сети. Свойство относится только к одному входному узлу, скрытому слою и одному выходному узлу.

Сеть RMPL применяется для моделирования динамических процессов в режиме «онлайн». Типичным примером ее приложения может служить имитация нелинейных динамических объектов, для которых сеть RMLP выступает в роли модели, а алгоритмы уточнения весов – в роли процедуры идентификации параметров этой модели. Идентифицированная модель объекта может в последующем использоваться для управления данным объектом. По этой причине сети RMLP применяются для имитации систем управления машинами, устройствами и динамическими процессами

Сеть RTRN (Real Time Recurent Network) – предназначена для обработки сигналов в реальном масштабе времени. В такой сети сигналы обратной связи в отличие от сети Элмана, рассмотренной ранее, поступают с задержкой только на вход части нейронов.

Сеть RTRN – это частный случай сети Элмана, в которой веса выходного слоя имеют бинарные значения.

Релаксационные нейронные сети

В предыдущем разделе были детально рассмотрены нейронные сети, которые в процессе своей работы реализуют широко применяемые модели обработки: прямого распространения, обратной связи, свертки, а также кодирования/декодирования. Далее будут рассмотрены сети релаксационного типа. Релаксационные нейронные сети характеризуются прямым и обратным распространением информации между слоями нейронной сети. В основе функционирования таких сетей лежит итерационный принцип работы. Он заключается в том, что на каждой итерации процесса происходит обработка данных, полученных на предыдущем шаге. Такая циркуляция информации продолжается до тех пор, пока не установится состояние равновесия. При этом состояния нейронных элементов перестают изменяться и характеризуются стационарными значениями. Поскольку релаксация – это процесс установления равновесия, то такие сети называются релаксационными. К релаксационным сетям относятся: нейронные сети Хопфилда, Хэмминга [44,49].

Сеть Хопфилда

Среди моделей нейронных сетей существуют такие, которые по способу обучения нельзя отнести к традиционным (обучающихся с учителем или без него). В таких сетях весовые коэффициенты синапсов рассчитываются только однажды перед началом функционирования сети на основе информации об обрабатываемых данных. Из сетей с подобной логикой работы наиболее известна сеть Хопфилда [41,44], которая обычно используется для организации ассоциативной памяти. Человеку для порождения некоторого воспоминания необходимо провести ассоциацию связанного с ним события или объекта, в случае компьютера сеть Хопфилда обеспечивает работу ассоциативной памяти – поиск в памяти образца данных, наиболее схожего с предъявляемым образцом. Обычная же компьютерная память является адресуемой, то есть информация извлекается по предъявляемому адресу.

Структурная схема сети Хопфилда приведена на рис. 2.86. Она состоит из единственного слоя нейронов, число которых является одновременно числом входов и выходов сети. Каждый нейрон связан синапсами со всеми остальными нейронами, а также имеет один входной синапс, через который осуществляется ввод сигнала. Выходные сигналы, как обычно, образуются на аксонах [41,50].

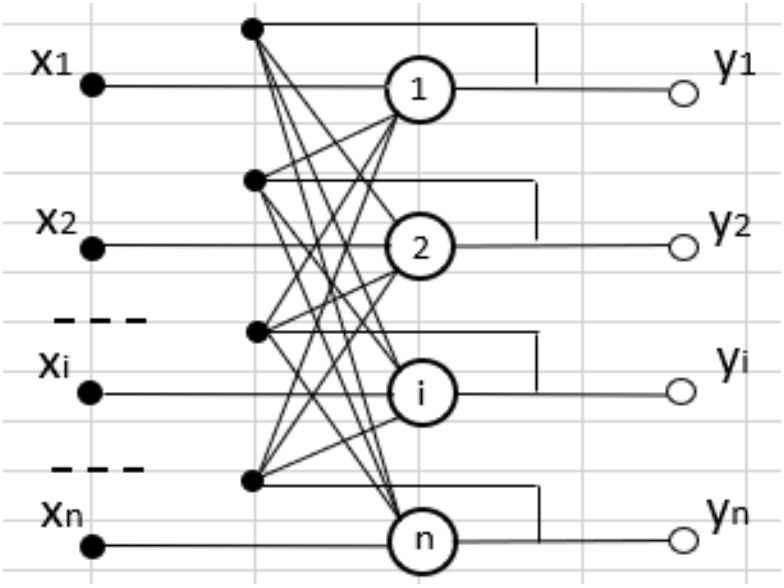


Рис.2.86. Построение сети Хопфилда

Как видно из рис. 2.86. сеть Хопфилда имеет обратные связи, т.е. сеть рекуррентная. В результате отклик таких сетей является динамическим, то есть после предъявления нового входа вычисляется выход и, передаваясь по обратной связи, модифицирует вход. Этот процесс повторяется многократно. Задача, решаемая данной сетью в качестве ассоциативной памяти, как правило, формулируется следующим образом.

Известен некоторый набор двоичных сигналов (изображений, звуковых оцифровок, бинарных последовательностей, описывающих объекты), которые считаются образцовыми. Сеть должна уметь из произвольного неидеального сигнала, поданного на ее вход, выделить («вспомнить» по частичной информации) соответствующий образец (если такой есть) или «дать заключение» о том, что входные данные не соответствуют ни одному из образцов. Если, например, сигналы представляют собой некие изображения, то, отобразив в графическом виде данные с выхода сети, можно будет увидеть картинку, полностью совпадающую с одной из образцовых (в случае успеха), или же «вольную импровизацию» сети (в случае неудачи).

Сеть является биполярной, то есть оперирует только величинами {–1, 1}.

В сети имеется один слой настраиваемых весов нейронов wji. Все нейроны единственного слоя возвращают свои выходы на свой вход и входы всех остальных нейронов сети посредством распределителей. Каждый нейрон реализует следующие шаги:

1) вычисляет взвешенную сумму a своих входов;
2) к сумме применяется нелинейная пороговая функция активации.

Работа сети Хопфилда.

Входные точки xi отображают первоначальный проход – ввод данных при обучении сети. Выходные значения будут сформированы тогда, когда сеть выдаст результат, наиболее похожий на один из обучающих примеров. Выходные сигналы подаются на входы всех соседних нейронов, кроме самого себя. На всех выходах yi и всех входах xi данные имеют формат +1 или -1.

Рассмотрим как работает сеть Хопфилда для задачи распознавания образов в виде черно-белых фото. Простейшее изображение

представлено в виде некоторой матрицы (рис.3.87). На практике строк и столбцов может быть намного больше. В этом примере закодирована буква «L». Сканируя по строкам ее изображение можно представить в виде вектора. При биполярном кодировании первой строки получается вектор: [1, -1, -1, -1, -1]. Продолжая сканирование далее по строкам можно получить:

[1, -1, -1, -1, -1]
[1, -1, -1, -1, -1]
[1, -1, -1, -1, 1]
[1, 1, 1, 1, 1].

Таким образом изображение выбранной буквы представляется как набор векторов.

На рисунке представлены два изображения: а) правильное в виде буквы «L» и в) искаженное. Алгоритм должен распознать правильный образ, даже если он немного искажен (зашумлен). Сеть Хопфилда предназначена для того, чтобы распознавать такие образы.

Сначала алгоритму предъявляются правильные образцы (один или несколько). Образцы кодируются и запоминаются, подаются на вход сети. В процессе обработки данные с выходных точек сети опять подаются на входы, т.к. сеть рекуррентная. Выход сети будет сформирован тогда, когда сеть выдаст результат, наиболее похожий на один из предъявленных примеров.

Обучение сети Хопфилда. Пусть имеется три образца векторов изображений, подлежащих распознаванию. Необходимо по итогам обучения создать квадратную матрицу преобразования, с помощью которой будет реализоваться алгоритм распознавания искаженного образца. Через x1, x2, x3 обозначен «правильные» образцы, через y обозначен искаженный образец.

x1 = [-1, 1, -1, 1]
x2 =[1, -1, 1, 1]
x3 =[-1, 1, -1, -1]
y = [1, -1, 1, -1]

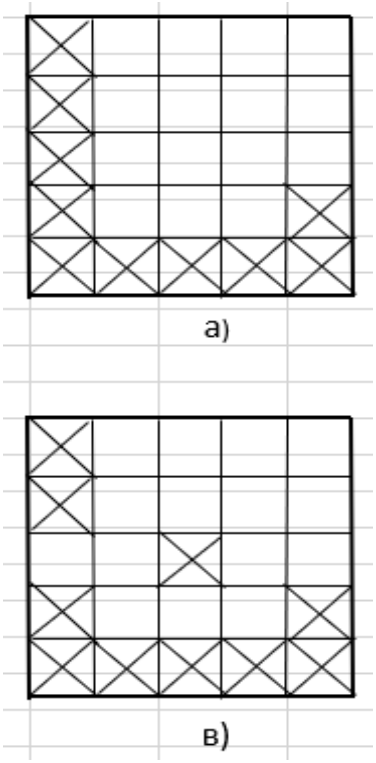


Рис.2.87. Закодированные в виде матрицы изображения

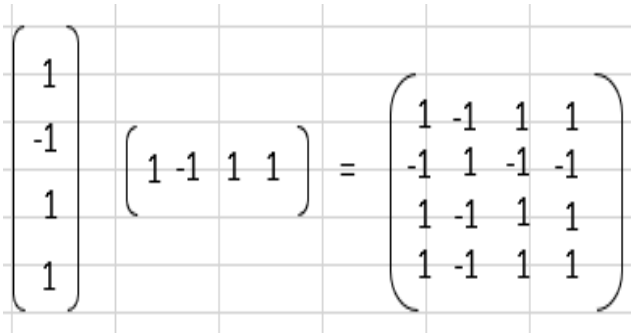
Формула для обучения создается суммой образцов в виде:

$$W = \sum_{1}^{3} x^T \bar{x}$$

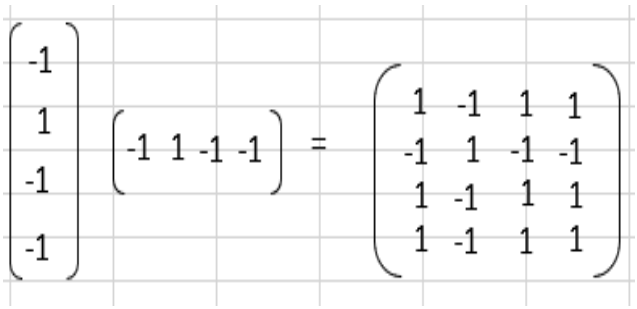
Получаемая по формуле квадратная матрица представляет собой результат произведения транспонированной матрицы первой строки x1 на ее же представление в виде строки:

$$\begin{pmatrix} -1 \\ 1 \\ -1 \\ 1 \end{pmatrix} \begin{pmatrix} -1 & 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \end{pmatrix}$$

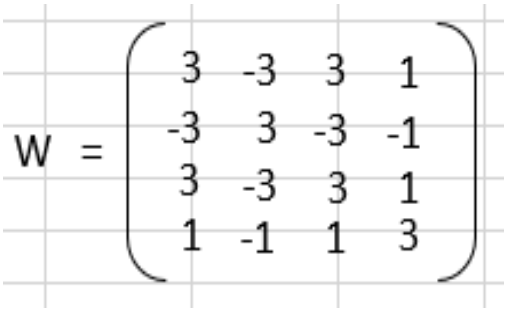
Такую же матрицу создаем для второй строки x2:



Для третьей матрицы x3 получаем:

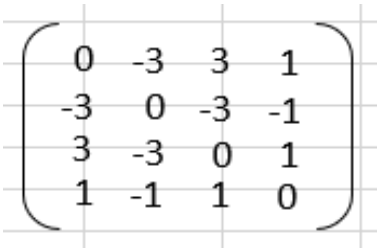


Вычисленные три матрицы поэлементно суммируются (1,1,1=3). Полученная в результате суммированная матрица будут симметричная, по диагонали все значения равны 3:



По правилам модели и алгоритма необходимо обнулить диагональ:

Полученная в результате обучения матрица умножается на



распознаваемый (искаженный) образец, полученный результат

умножается на активационную функцию (рис.2.88). В данном случае выбрана активационная функция в виде сигнатуры (знаковая), в результате умножения на которую формируется первый результирующий образец y* (рис.11.3):

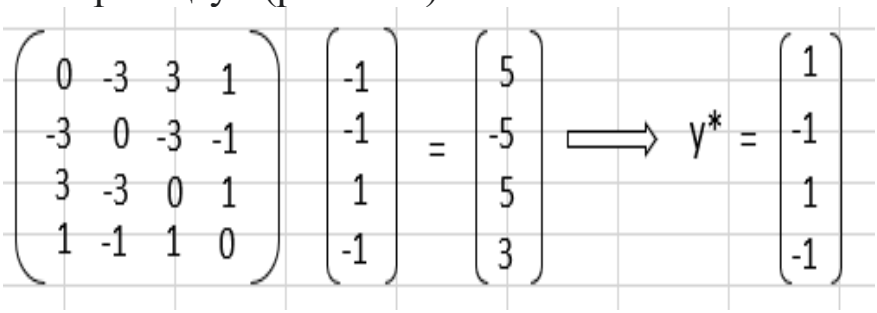


Рис.2.88. Процесс формирования первого результирующего образца

Далее процесс продолжается следующим образом. Полученный первый образец сравнивается с одним из «правильных» образцов, в случае удачного решения процесс останавливается, в случае значительного расхождения полученный последний образец вновь подается на матрицу обучения для реализации следующего вычислительного цикла. Это повторяется до тех пор, пока не будет получен нужный результат.

В данном примере один из правильных образцов был получен на первом вычислительном шаге, благодаря простоте изображений. При сложных снимках с десятками входных битов может занимать несколько вычислительных циклов обработки [45]. Для зашумленных изображений такой процесс позволяет практически восстановить первоначальный, правильный вариант.

Таким образом, общий алгоритм работы и процесс формирования результата сети Хопфилда состоит в следующем.

Первоначально все запоминаемые образцы кодируются биполярными векторами.

Веса сети Хопфилда настраиваются следующим образом: фактически вычисляется произведение каждого запоминаемого вектора с самим собой и суммированием полученных матриц.

После запоминания образцов становится возможным восстановление. Входам придают значение образа, возможно частично искаженного, и сети дается возможность достичь одного из запомненных

состояний. Значения выходов и есть восстановленный образец.

Пример использования сети Хопфилда. На рисунке 2.89 **п**редставлено несколько черно-белых эталонов, каждый из которых имеет размер 6x6 пикселей. В соответствии с поставленной задачей, требуется по введенному искусственно зашумленному символу восстановить его до наиболее подходящего эталона.

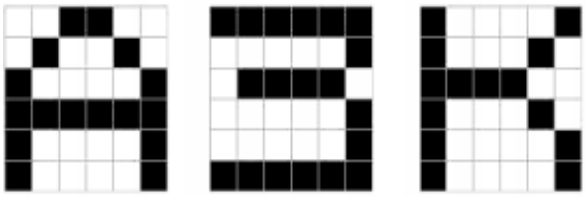
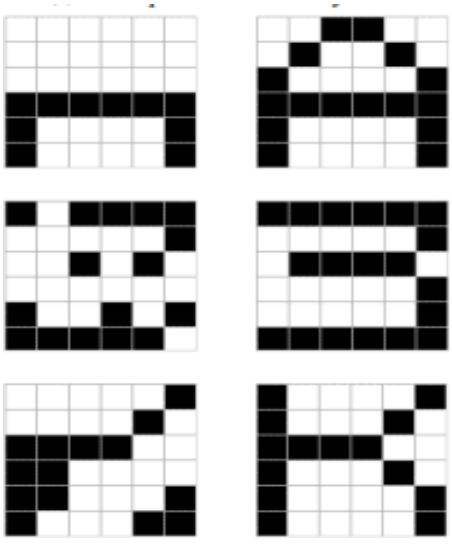


Рис.2.89. Эталоны изображений

После запоминания эталонов на вход сети Хопфилда для проверки функционирования были представлены зашумленные образцы. По завершении всех вычислений для был получен результат, представленный на рис. 2.90.



а) б)

Рис.2.90. Начальные образцы а), и их распознавание б).

Результаты показывают, что первоначальные образцы изображений восстановились полностью.

Нейронная сеть Хэмминга

Сеть Хэмминга многослойная и состоит из различных классов нейронных сетей. Пусть задано m образов, каждый из которых имеет размерность n:

$$X^1 = [x^1_1, x^1_2, \dots, x^1_n],$$
$$X^2 = [x^2_1, x^2_2, \dots, x^2_n],$$
$$\dots\dots\dots\dots$$
$$X^m = [x^m_1, x^m_2, \dots, x^m_n]$$

Тогда нейронная сеть Хэмминга будет состоять из сети с прямыми связями, сети Хопфилда и слоя выходных нейронов (рис. 22.91).

Сеть с прямыми связями состоит из n входных распределительных и m выходных нейронных элементов. Она вычисляет меру подобия между входным и эталонным образом, хранящимися в сети. В качестве меры подобия используется количество одинаковых разрядов между входным и эталонным образом [41,50].

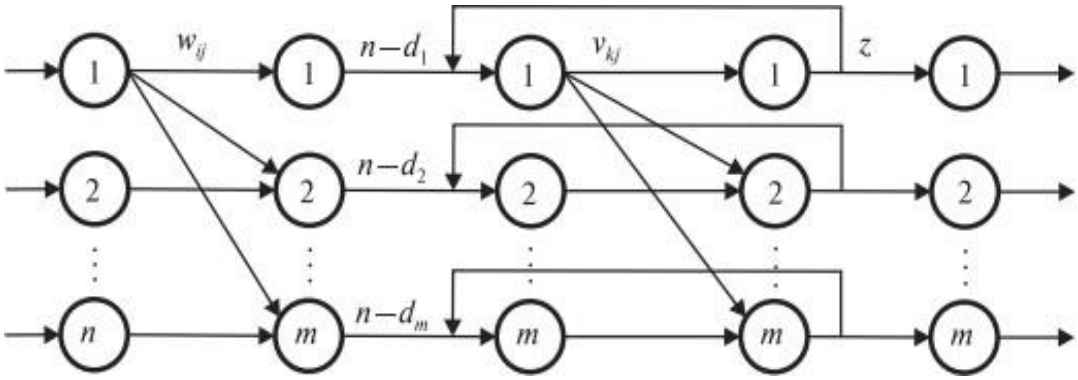


Рис.2.91. Архитектура нейронной сети Хэмминга

Тогда выходное значение i-го нейрона второго слоя представляет собой меру подобия Pi между входным и i-м эталонным образом:

$$P_i = n - d_i,$$

где di – расстояние Хэмминга между входным и i-м эталонным паттерном.

Сеть Хопфилда используется для разрешения возникающих конфликтов, когда входной паттерн подобен нескольким эталонным образам, хранящимся в сети. В процессе релаксации сети Хопфилда на ее выходе остается только один нейронный элемент с положительной выходной активностью.

Выходной слой нейронной сети состоит из *m* нейронов, каждый из которых имеет соответствующую пороговую функцию активации. Он предназначен для преобразования положительной выходной активности нейрона сети Хэмминга в единичное значение. При этом значения всех остальных нейронов выходного слоя устанавливаются в нулевое состояние.

Таким образом, происходит идентификация входного паттерна, который кодируется номером нейрона выходного слоя, имеющим единичное значение. Если входной образ не совпадает с эталонным, то на выходе сети Хэмминга будет формироваться эталонный паттерн с минимальным расстоянием Хэмминга по отношению к выходному образу.

Исходя из предыдущего функционирование нейронной сети Хэмминга состоит из следующих шагов:

1. Определяются весовые коэффициенты и пороговые значения для соответствующих слоев нейронной сети.

2. На вход сети подается неизвестный образ и производится инициализация нейронных элементов сети Хопфилда.

3. Производится вычислительная итерационная процедура расчета выходных значений нейронной сети Хопфилда до тех пор, пока она не стабилизируется. В этом случае на выходе сети Хэмминга один нейронный элемент из сети будет иметь единичное состояние, а остальные – нулевое состояние.

4. Если в выходном слое существует несколько нейронных элементов- победителей, то выбор одного из них производится случайным образом.

Нейронная сеть Кохонена

Основным их назначением сетей Кохонена является кластеризация образцов, то есть разделение образцов на группы (кластеры) по тем или иным признакам.

Нейронная сеть Кохонена в отличие от рассмотренных сетей обучается без учителя, поэтому назвается самоорганизующейся картой Кохонена [41,42,43].

Применение кластерного анализа в общем виде сводится к следующим этапам:

- отбор выборки объектов для кластеризации;
- определение множества переменных с нормализацией их значений;
- вычисление значений меры сходства между объектами;, ,
- реализация алгоритма метода кластерного анализа для создания групп сходных объектов;
- представление результатов анализа.

В процессе нормализации все значения приводятся к некоторому диапазону, например, [-1, -1] или [0, 1]. Признак, по которому собираются предметы в одну группу, а другие в другую, называется метрикой**.** Самый распространенный вариант – это когда метрикой является расстояние. Каждый объект описывается расстояниями до всех остальных объектов обучающей выборки, функцией расстояния является метрика, часто используется и здесь используется евклидова метрика.

Карты Кохонена имеют набор входных элементов, количество которых совпадает с размерностью подаваемых на вход векторов, и набор выходных элементов, каждый их которых соответствует одному кластеру (рис.2.92).

Обычно стараются задавать количество выходных элементов меньшим, чем количество входных, в таком случае сеть позволяет получить упрощенную характеристику объектов для дальнейшей работы с ними.

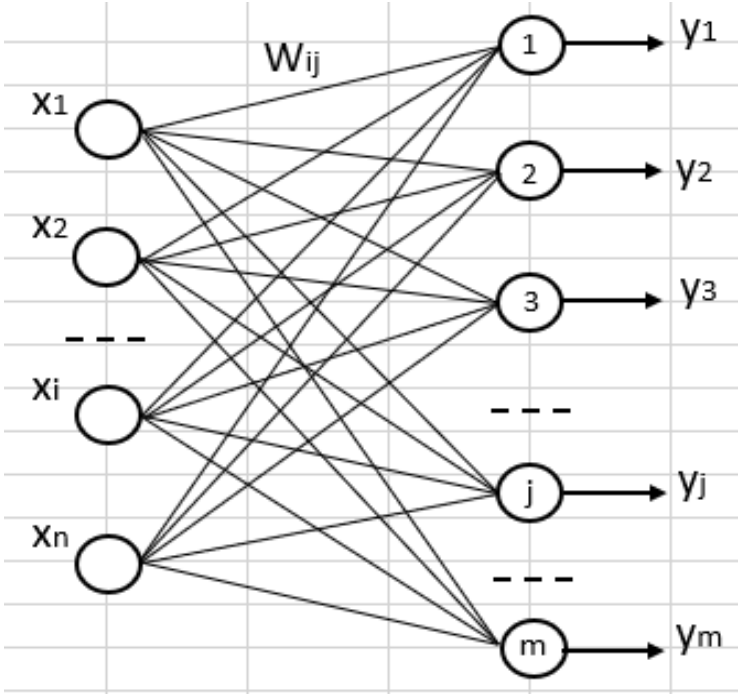


Рис.2.92. Сеть Кохонена

Также необходимо остановиться на структуре связей между элементами сети. Тут все просто – каждый входной элемент соединяется с каждым выходным, и все связи, так же, как и для других нейронных сетей, имеют определенный вес wij, который корректируется в процессе обучения.

Работа сети. При подаче какого-либо вектора на вход сеть должна определить, к какому из кластеров этот вектор ближе всего. В качестве критерия близости был выбран критерий минимальности квадрата евклидова расстояния. Рассмотрим входной вектор как точку в n-мерном пространстве (n – количество координат вектора, это число равно числу входных нейронов, как мы обсуждали ранее). Тогда нужно вычислить расстояние между этой точкой и центрами разных кластеров и определить, расстояние до какого из кластеров окажется минимальным. Тогда этот кластер (и соответствующий ему выходной нейрон) объявляется победителем.

Координатами центра кластера являются величины весов всех связей, которые приходят к данному выходному нейрону от входных элементов. Поскольку каждый выходной нейрон (кластер) соединен с каждым входным нейроном, то мы получаем, n связей, то есть n

координат для точки, соответствующей центру кластера. Таким образом, подав вектор на вход сети мы получим один кластер-победитель, соответственно, этот вектор будет принадлежать именно к этому кластеру (группе). Именно так карты Кохонена решают задачу классификации.

В этой модели обработки победителем в конкурентной борьбе является такой нейронный элемент с номером ***k***, который в результате подачи на вход сети определенного образа имеет максимальное значение взвешенной суммы:

$$S_k = \max S_j ,$$

где взвешенная сумма j-го нейрона вычисляется как:

$$S_j = \Sigma\, wij\ xi = Wj\ X^T.$$

Здесь X = (x1,x2,...,xn) - входной образ, Wj = (w1j, w2j, ...,wnj) – вектор- столбец весовых коэффициентов j-го выходного нейрона.

Тогда активность выходных нейронов:

yj = F(Sj) = 1, если j=k и 0 в остальных случаях.

Таким образом, после обучения нейронной сети при подаче входного образа активность нейрона-победителя принимается равной единице, а остальных нейронов – нулю.

Для определения нейрона-победителя необходимо определить евклидово расстояние между входным образом и весовыми векторами нейронов обрабатывающего слоя. Так, для *j*-го нейрона второго слоя евклидово расстояние вычисляется по формуле:

$$Dj = | X - Wj | = \sqrt{(x1- w1j)^2 + (x2 – w2j)^2 + ...+ (xn – wnj)}.$$

Далее находится нейрон-победитель с номером *k*, который соответствует минимальному евклидовому расстоянию между входным и весовым вектором:

$$Dk = \min | X - Wj |.$$

В процессе обучения сети возникает необходимость корректировки весов нейронов. Для корректировки весов используется следующая формула:

$$wij\ (t+1) = wij\ (t) + q\ (t)\ (xi – wij(t))$$

В этой формуле wij (t+1) – вес на шаге t+1, а wij (t) – вес на предыдущем t-шаге, q – норма обучения, а xi – координата входного вектора. Норма обучения зависит от шага обучения и меняется в процессе

обучения.

Рассмотрев модель процесса обучение можно написать конкретный алгоритм для этого процесса:

- берется учебный вектор и вычисляется квадрат евклидова расстояния от него до каждого из кластерных элементов сети;
- находится минимальное из полученных значений и определяется элемент-победитель;
- для нейрона-победителя, а также для тех нейронов, которые попали в заданный радиус, выполняется корректировка весов связей;
- обновляются значения нормы обучения и радиуса;
- продолжаем обучение, если не выполнено условие остановки обучения. Остановка обучения происходит в том случае, если величины изменения весов становятся очень маленькими.

Более подробно алгоритм кластеризации описан в гл.6 на примере реализации алгоритма k-средних (k-means).

Глубокие нейронные сети

Название “глубокие нейронные сети” пошло от использования множества скрытых слоёв, которые и делают нейросеть способной обучаться более сложным алгоритмам обработки. Истории успешного применения глубокого обучения только-только начали появляться в последние годы, т.к. процесс обучения нейронной сети сложный по части вычислений и требует больших объёмов данных [41,44].

Глубокие искусственные нейронные сети — это сети с более чем тремя слоями между входным и выходным слоями. Такие слои называются скрытыми. Нейронная сеть глубокого обучения являет собой своего рода черный ящик: трудно сказать точно, какой логикой руководствуется хорошо обученная нейронная сеть. Самая сложная часть в глубоких нейронных сетях — это обучение сети. Для этого через такие сети прогоняют огромные массивы данных. В конечном итоге сеть обучается решать узкопрофильные задачи разного рода. Существует огромное число разновидностей глубоких сетей, различающихся архитектурой: числом слоев, числом нейронов в каждом слое, а также структурой связей. Фактически глубокие нейронные сети автоматизировали труд исследователей, которые раньше занимались конструированием признаков [45,46].

В общем случае глубокие нейронные сети осуществляют иерархическое преобразование входного пространства образов и представляют собой нейронные сети с множеством слоев нейронных элементов. Существуют следующие глубокие нейронные сети (DNN), обычные варианты которых были ранее рассмотрены:

- глубокий персептрон;
- глубокая сверточная нейронная сеть;
- глубокая рекуррентная нейронная сеть;
- глубокий автоэнкодер;

Исторически первыми появились нейронные сети глубокого доверия и глубокий персептрон, которые в общем случае представляют собой многослойный персептрон с более чем двумя скрытыми слоями. До 2006 г. в научной среде было понимание, что многослойный персептрон с одним, максимум двумя скрытыми слоями более эффективен для нелинейного преобразования входного пространства образов в выходное по сравнению с персептроном с большим количеством скрытых слоев. Считалось, что не имеет смысла применять персептрон с более чем двумя скрытыми слоями. Проблема заключается в том, что все попытки применять алгоритм обратного распространения ошибки для обучения персептрона с тремя и более скрытыми слоями не приводили к улучшению решения различных задач. Это связано с тем, что алгоритм обратного распространения ошибки неэффективен для обучения персептронов с тремя и более скрытыми слоями при использовании сигмоидной функции активации. В 2006 г. Дж. Хинтон предложил алгоритм послойного обучения [47], который стал эффективным средством обучения глубоких нейронных сетей. Было показано, что глубокая нейронная сеть имеет большую эффективность нелинейного преобразования и представления данных по сравнению с традиционным персептроном.

Архитектура глубокой нейронной сети. На рис.2.93 представлена обобщенная схема построения глубокой нейронной сети [31].

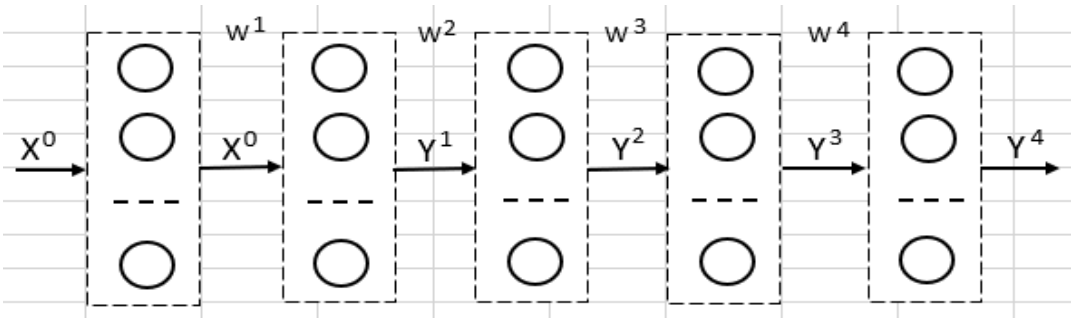


Рис.2.93. Обобщенная структура глубокой нейронной сети

В матричном виде выходной вектор k-го слоя

$$Y^k = F(S^k) = F(W^k Y^{k-1} + T^k)$$

где W – матрица весовых коэффициентов; Y^{k-1} – выходной вектор $(k-1)$ – го слоя; T^k – вектор пороговых значений нейронов k – го слоя.

Если глубокая нейронная сеть используется для классификации образов, то выходные значения сети часто определяются на основе функции активации Softmax:

Несмотря на архитектурные различия глубоких нейронных сетей, принципы их обучения являются идентичными. В настоящее время принята следующая парадигма для обучения глубоких нейронных сетей. Если обучающая выборка большая, т. е. размерность обучающей выборки намного больше, чем количество настраиваемых параметров сети, то используется метод стохастического градиента с функцией активации ReLU нейронных элементов. Если размерность обучающей выборки сравнима с количеством настраиваемых параметров сети, то применяется предварительное обучение нейронной сети и алгоритм обратного распространения ошибки для точной настройки синаптических связей сети.

Глубокое обучение. Искусственный интеллект - это очень широкая область исследований, посвященная когнитивным способностям машин: обучение определенному поведению, упреждающее взаимодействие с окружающей средой, способность к

логическому выводу и дедукции, компьютерное зрение, распознавание речи, решение задач, представление знаний, восприятие действительности и многое другое (за подробностями отсылаем к книге S. Russell, P. Norvig «Artificial Intelligence: A Modern Approach», Prentice Hall, 2003). Менее формально под ИИ понимается любая ситуация, в которой машины имитируют *интеллектуальное* поведение, считающееся присущим человеку. Искусственный интеллект заимствует методы исследования из информатики, математики и статистики.

Глубокое обучение – подмножество методов машинного обучения, в которых применяются искусственные нейронные сети (ИНС), построенные на базе аналогии со структурой нейронов человеческого мозга. Термин «глубокий» подразумевает наличие большого числа слоев в ИНС, но его интерпретация со временем менялась. Если еще четыре года назад считалось, что 10 слоев достаточно, чтобы называть сеть глубокой, то теперь глубокой обычно называется сеть, содержащая сотни слоев. На рис.2.94 представлена иерархия средств искусственного интеллекта.

Рис.2.94. Компоненты искусственного интеллекта

Глубокое обучение - это сравнительно небольшое число численных методов, которые с огромным успехом применяются в самых разных областях (обработка изображений, текста, видео и речи, компьютерное зрение), что позволило добиться значительного

прогресса по сравнению с результатами, достигнутыми за предшествующие десятки лет.

Для построения эффективных процедур обучения в архитектурах глубоких нейронных сетей применяются графические процессоры (GPU), которые многократно увеличивают скорость обучения.

В компаниях Google, Microsoft, Amazon, Apple, Facebook и многих других методы глубокого обучения постоянно используются для анализа больших массивов данных. Методы глубокого машинного обучения позволяют формировать базы знаний и они стали достоянием крупных промышленных компаний, стали составной частью современной программной продукции, и владение ими обязательно для программиста.

Построение нейронной сети

При построении модели ИНС необходимо точно определить задачи, которые будут решаться с её помощью. Нет необходимости придумывать нейронную сеть с «нуля», так как существует несколько десятков различных нейросетевых архитектур, эффективность многих из которых доказана математически.

Первым этапом построения нейросетевой модели является тщательный отбор входных данных, влияющих на ожидаемый результат. Из исходной информации необходимо исключить все сведения, не относящиеся к исследуемой проблеме, а также следует располагать достаточным количеством примеров для обучения сети. Существует эмпирическое правило, которое устанавливает рекомендуемое соотношение X между количеством обучающих примеров, содержащих входные данные и правильные ответы, и числом соединений в нейронной сети:

$$X \leq 10$$

Для факторов, которые включаются в обучающую выборку, целесообразно предварительно оценить их значимость, проведя анализ диапазоны их возможных изменений.

На втором этапе осуществляется преобразование исходных данных с учётом характера и типа решаемой проблемы, выбираются способы представления информации. Эффективность нейросетевой модели и достоверность ее результатов повышается, если диапазоны изменения входных и выходных величин приведены к стандарту

[0,1] или [–1,1].

Третий этап заключается в конструировании сети, т.е. в проектировании её архитектуры с определением числа слоёв и число входящих нейронов в каждом слое. Структура нейронной сети формируется до начала обучения, поэтому успешное решение этой задачи определяется опытом специалиста.

Четвёртый этап связан с обучением сети, которое может проводиться на основе конструктивного или деструктивного подхода. В соответствии с первым подходом обучение ИНС начинается на сети небольшого размера, который постепенно увеличивается до достижения требуемой точности. Деструктивный подход базируется на принципе «прореживания дерева», в соответствии с которым из сети с заведомо избыточным объёмом постепенно удаляют «лишние» нейроны и их связи. Этот подход даёт возможность исследовать влияние удалённых связей на точность работы сети.

Процесс обучения нейронной сети представляет собой уточнение значений весовых коэффициентов и для отдельных узлов на основе постепенного увеличения объёма входной и выходной информации. Началу обучения должна предшествовать процедура выбора функции активации нейронов, учитывающая характер решаемой задачи. В трёхслойных перцептронах на нейронах скрытого слоя применяется в большинстве случаев логистическая функция, а тип передаточной функции нейронов выходного слоя определяется на основе анализа результатов вычислительных экспериментов на сети.

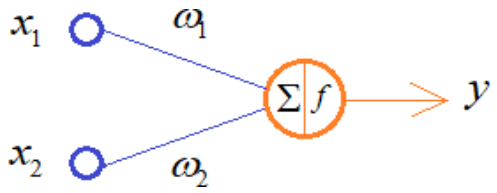
На пятом этапе проводится тестирование полученной модели ИНС на независимой выборке.

Пример алгоритма классификации «исключающее ИЛИ»

При решении задач классификации, изложенных в главе 6, выполняется бинарной алгоритм разделения двух множеств сходных объектов с разными параметрами. В них множества четко разделены на два класса для случая линейно-разделимых образов.

Рассмотрим более сложный пример на классификацию, приведенную в гл.6. Необходимо вывести функцию классификации путем построения прямой, разделяющей два класса объектов. Данная задача также относится к обучению с учителем. Рассмотрим простейший

персептрон для задачи классификации двух классов образов, представленных двумя характеристиками $x1$ и $x2$:



с активационной функцией:

$$f(x)=\begin{cases}1, & x\geq 0 \quad \rightarrow C_1\\ -1, & x<0 \quad \rightarrow C_2\end{cases}$$

То есть, если значение суммы Σ больше или равно 0, то вектор принадлежит классу 1:

$$[x_1, x_2]^T \in C_1$$

иначе классу 2:

$$[x_1, x_2]^T \in C_2$$

Далее применяя активационную функцию можно увидеть, что граница разделения двух классов проходит на уровне 0, то есть если:

$$\begin{cases}\omega_1 x_1+\omega_2 x_2 \geq 0 \rightarrow C_1\\ \omega_1 x_1+\omega_2 x_2 < 0 \rightarrow C_2\end{cases}$$

значит сумма:

$$\omega_1 x_1+\omega_2 x_2=0$$

определяет границу разделения одного класса объектов от другого. Ее еще можно записать в виде:

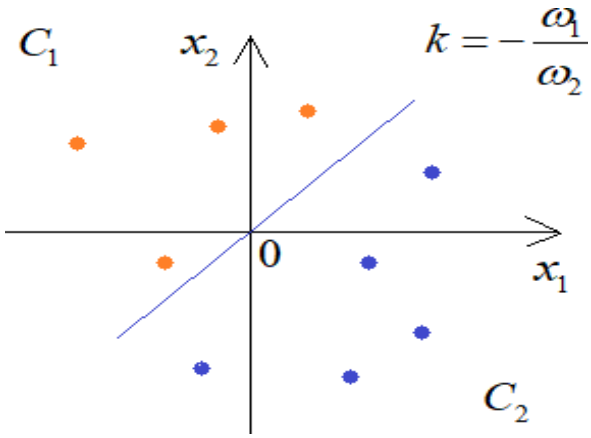
$$x_2=-\frac{\omega_1}{\omega_2}\cdot x_1$$

то есть это прямая с угловым коэффициентом:

$$k = -\frac{\omega_1}{\omega_2}$$

,

проходящая через начало системы координат:



Все точки по одну сторону от этой прямой будут относиться к одному классу, а по другую сторону – к другому классу. Такая прямая называется разделяющей прямой (в многомерном случае – разделяющей плоскостью). Этот двумерный график иллюстрирует возможность правильной классификации простейшим персептроном только линейно-разделимых объектов.

В качестве примера смоделируем два класса линейно-разделимых образов разделяющей прямой:

$$x_2 = 1 \cdot x_1$$

то есть, прямой, идущей под 45 градусов к осям координат. В этом случае, для корректной классификации, мы должны выбрать веса нейронной сети равными, но с противоположными знаками:

$$k = 1 = -\frac{\omega_1}{\omega_2} \Rightarrow \omega_1 = -0,3; \quad \omega_2 = 0,3$$

Здесь взяты коэффициенты по 0,3 с противоположным знаком. Можно выбрать и любые другие, например, -1 и 1. Данная нейронная сеть успешно классифицирует эти образы. Но, если коэффициенты взять не равными:

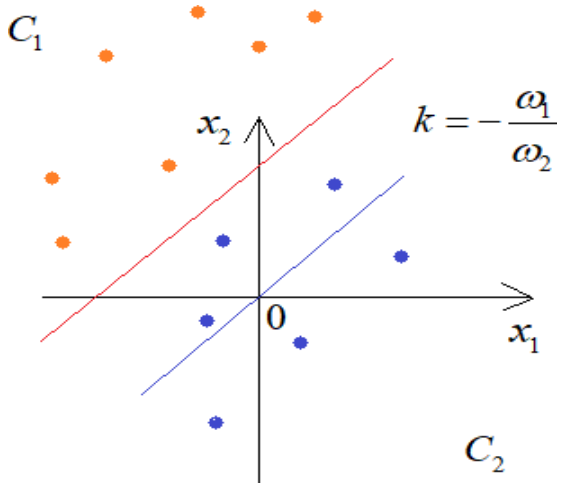
$$\omega_1 = -0,2; \quad \omega_1 = 0,1$$

то первый класс будет неверно распознаваться. Это, в частности, означает, что НС неверно настроена на классификацию таких образов и ее весовые коэффициенты нуждаются в корректировке.

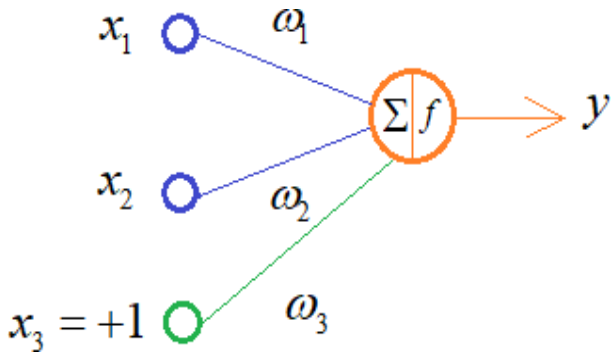
Вернемся к рабочему состоянию с весами:

$$\omega_1 = -0,5; \quad \omega_1 = 0,5$$

и предположим, что все наши образы сдвигаются вверх по оси $x2$:



Теперь, наша разделяющая прямая не сможет верно классифицировать такие образы, т.к. она проходит через начало координат. И как бы мы ее ни крутили, корректного разделения не получится, т.к. необходимо смещение. Поэтому, в нейронную сеть дополнительно определяют еще один вход для смещения разделяющей гиперплоскости.



С этим введенным дополнительным входом, разделяющая прямая принимает вид:

$$x_2 = -\frac{a_1}{a_2} x_1 - \frac{a_3}{a_2} \cdot 1$$

то есть, мы можем теперь сдвинуть ее на любое требуемое значение.

Предположим, что все образы сдвинуты вверх по оси x_2 на величину b. Тогда третий весовой коэффициент нейронной сети следует выбрать из уравнения:

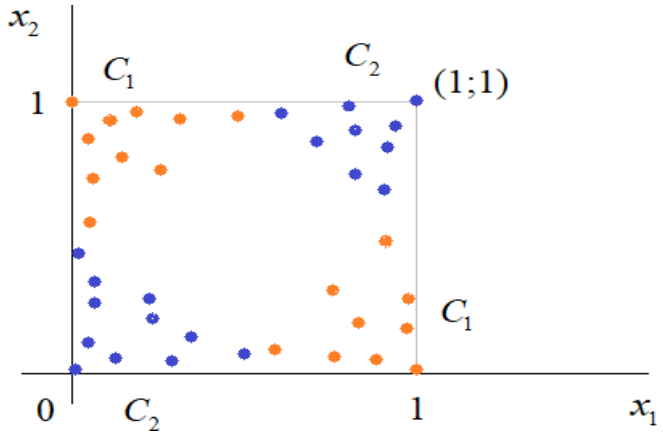
$$-\frac{a_3}{a_2} = b \Rightarrow a_3 = -b \cdot a_2$$

Теперь к входному вектору добавляется третье значение +1 и теперь нейронная сеть корректно классифицирует такие смещенные образы.

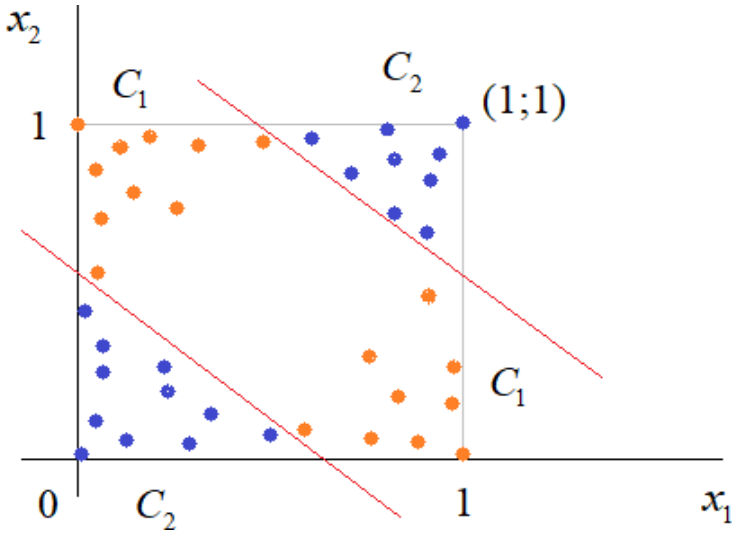
Теперь мы знаем, зачем нужно было смещение в нейронной сети. Это смещение используется во всех современных сетях, а не только в персептроне. Здесь лишь показан пример необходимости его использования. Но та же самая картина сохраняется и для других видов нейронных сетей с большими размерностями.

Реализация алгоритма «исключающее ИЛИ»

Рассмотренные выше нейронная сети с одним нейроном, могут классифицировать только линейно-разделимые образы. Однако, на практике чаще встречаются более сложные задачи. Например, представим, что классы наших образов распределены образом и не могут быть классифицированы с помощью одной линейной функции.

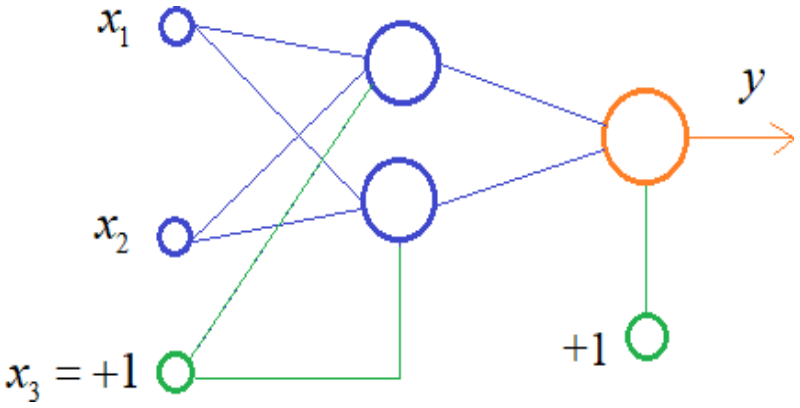


Здесь для правильной классификации невозможно ограничиться одной линией. Можно провести две линии следующим образом:



Все, что будет попадать между этими двумя линиями будем относить к первому классу, а все объекты за пределами этих линий – ко второму классу.

При создании архитектуры нейронной сети необходимо учитывать, что каждая разделительная линия может быть реализована на отдельном нейроне, а результат классификации может быть объединен результирующим нейроном выходного слоя.



Для простоты предположим, что на вход такой сети подаются только два значения: 0 или 1.

$$x_1, x_2 \in [0,1]$$

Тогда все объекты будут лежать в четырех углах квадрата:

$$\begin{array}{ccc} 0 & 0 & C_2 \\ 0 & 1 & C_1 \\ 1 & 0 & C_1 \\ 1 & 1 & C_2 \end{array}$$

Анализ полученной таблицы показывает, что при $C_2 = 0$ и $C_1 = 1$ получена таблица истинности «исключающее ИЛИ». В этом случае активационная функция каждого нейрона будет иметь вид:

$$f(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

Для решения поставленной задачи классификации необходимо определить веса связей нейронной сети. На начальном этапе можно предположить, что первый нейрон скрытого слоя будет формировать границу:

$$x_2 = -1 \cdot x_1 + 1,5$$

Учитывая ранее приведенную формулу:

$$x_2 = -\frac{a_1}{a_2} x_1 - \frac{a_3}{a_2} \cdot 1$$

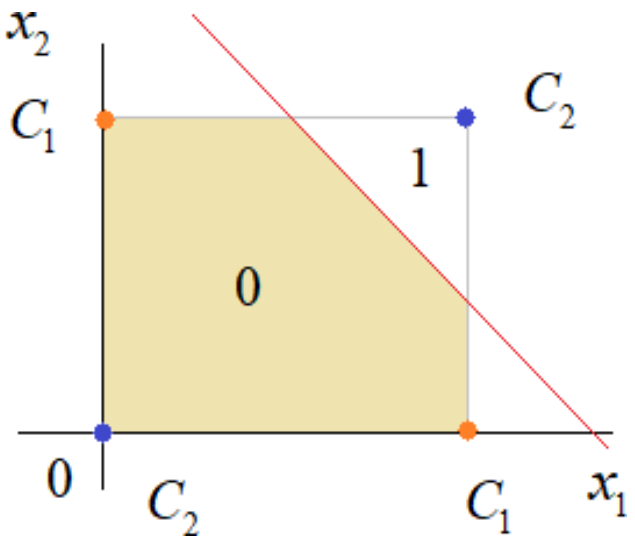
веса входов первого нейрона $x1, x2$ можно взять равными:

$$a_1 = a_2 = 1$$

При этом вес третьей связи будет:

$$a_3 = -b \cdot a_2 = -1,5$$

В результате получена прямая, которая формирует следующее разделение на плоскости:



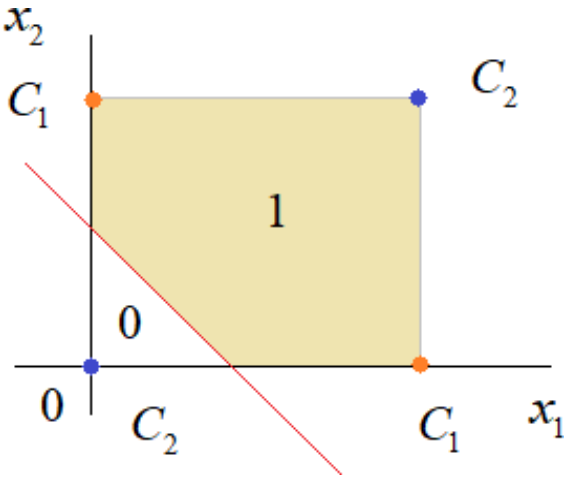
Второй нейрон скрытого слоя будет формировать параметры прямой:

$$x_2 = -1 \cdot x_1 + 0,5$$

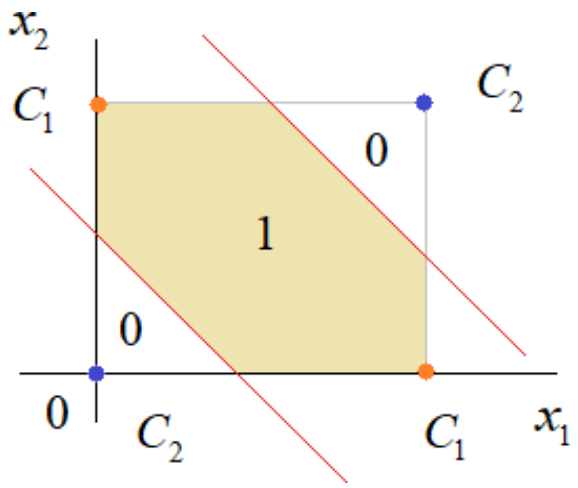
И веса связей второго скрытого слоя нейронной сети можно взять равными:

$$a_1 = a_2 = 1 \qquad a_3 = -0,5$$

В результате получается следующее разделение на плоскости:

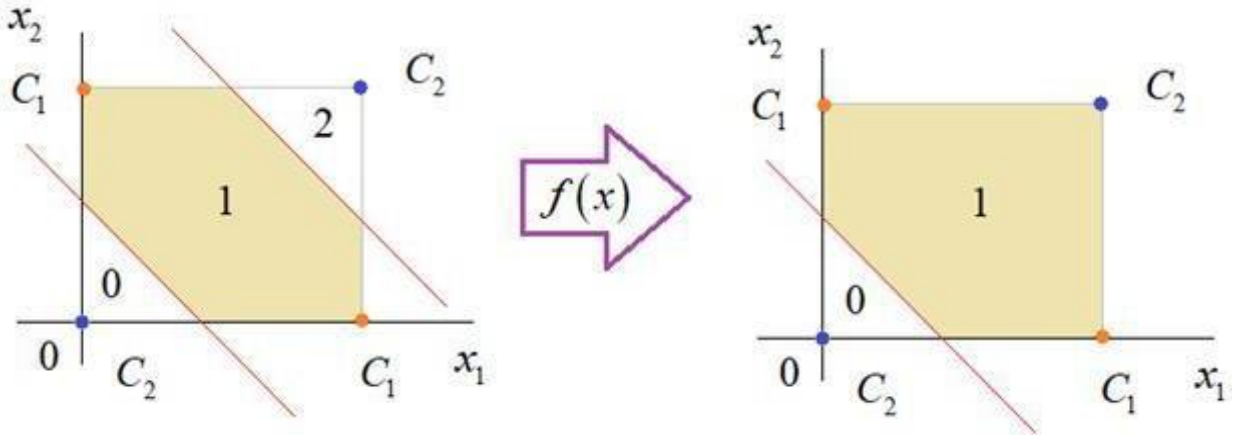


На следующем шаге алгоритма необходимо объединить результаты двух предыдущих шагов чтобы получилась ранее задуманная область разделения.

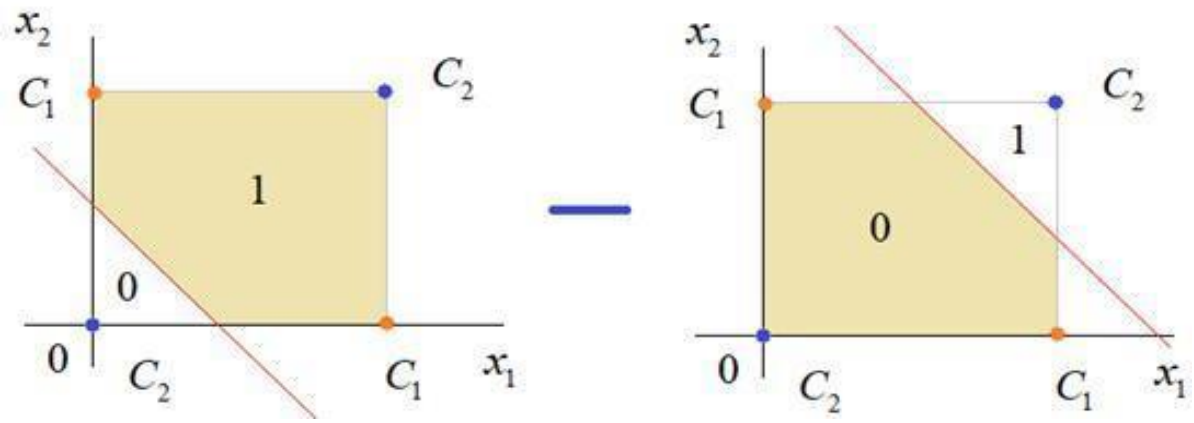


Для получения такой картины разделения областей и соответствующей картины классификации необходимо найти алгоритм объединения двух ранее полученных областей.

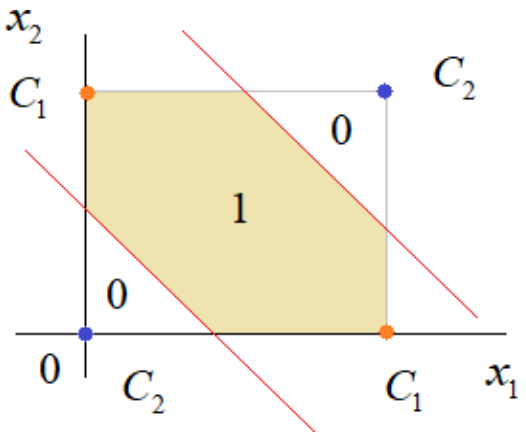
Если результаты просто сложить, то выходе активационной функции получим одну большую область с 1 и одну маленькую область с 0. Такой результат не может удовлетворить, т.к. в начальном представлении имеется четыре области расположения объектов, и одна из областей при классификации оказывается исключенной.



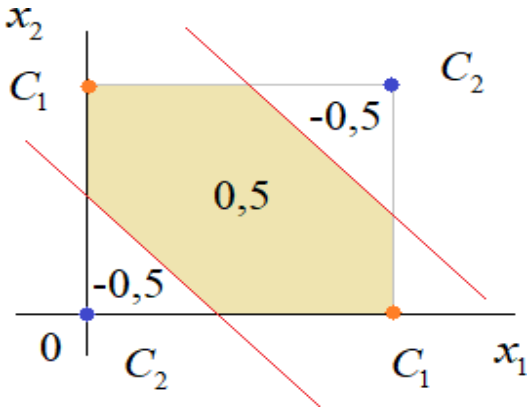
Этот результат не соответствует ожидаемому, поэтому лучше из второго вычесть первое.



Тогда на входе выходного нейрона можно получить следующие значения:

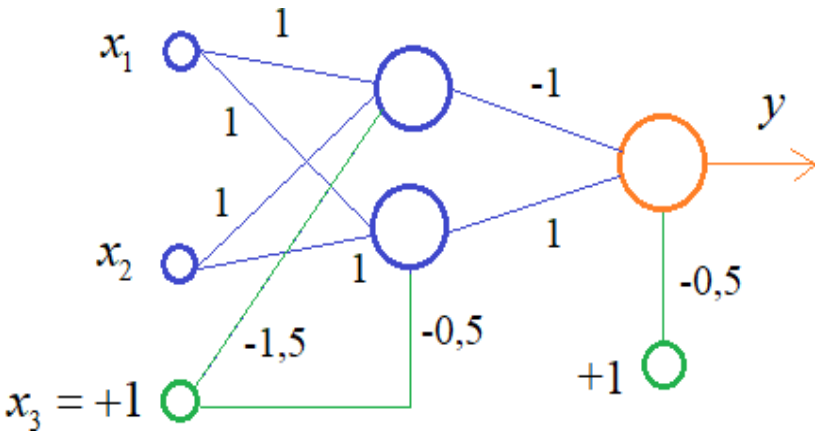


Можно сместить эти значения на величину -0,5 и тогда получить окончательный результат:



В результате проделанной работы по подбору линий разделения и

весовых коэффициентов нейронной сети, получим окончательную схему построения нейронной сети, разделяющей области размещения объектов и их классификации:



В итоге работы получена структура, которая решает задачу разделения двух множеств объектов на классы, то есть выполняет процедуру классификации – одну из основных процедур машинного обучения. Добавление одного скрытого слоя позволило решать уже задачи, выходящие за пределы линейно-разделимых множеств распознаваемых объектов.

Приведенный пример хорошо показывает, что добавляя новые нейроны, можно получать все более сложные формы разделяющих выпуклых областей, полученные комбинацией разделяющих линий или гиперплоскостей. Это приводит к более сложным схемам классификации.

Глава 3. КВАНТОВЫЕ АЛГОРИТМЫ ПРИ ИНТЕЛЛЕКТУАЛЬНОМ АНАЛИЗЕ ДАННЫХ

3.1. Инновационные квантовые технологии в сельском хозяйстве для оценки плодородия земли

Сельское хозяйство является ключевой отраслью, обеспечивающей продовольственную безопасность. В условиях современных вызовов, таких как изменение климата и устойчивое использование ресурсов, становится необходимым внедрение инновационных технологий для повышения эффективности сельского хозяйства. Оценка плодородия почвы играет решающую роль в оптимизации использования удобрений и ресурсов. Одним из инновационных подходов является использование квантовых технологий для оценки плодородия почвы. Вариационные квантовые цепи (VQC) предоставляют уникальную возможность эффективного решения задач классификации в контексте анализа данных о почвенных характеристиках. В данном исследовании мы использовали данные о химических и физических свойствах почвы, включая плотность, влажность, уровень pH, содержание азота, фосфора и калия. Для построения модели VQC мы преобразовали эти данные в квантовые состояния, используя различные анзацы, такие как ZZFeatureMap и RealAmplitudes. Для сравнения результатов мы использовали традиционные методы классификации, такие как метод опорных векторов (SVM), и сравнили их с результатами, полученными с использованием VQC. Мы разбили данные на обучающую и тестовую выборки, обучили модели на обучающих данных и оценили их производительность на тестовых данных. Обсуждались преимущества и ограничения применения вариационных квантовых цепей в оценке плодородия земли. Рассматривались перспективы дальнейшего развития и улучшения методологии. Вариационные квантовые цепи представляют собой перспективное направление для разработки инновационных методов оценки плодородия почвы в сельском хозяйстве. Результаты нашего исследования подчеркивают потенциал квантовых технологий в сельском хозяйстве и необходимость дальнейших исследований в этом направлении.

Сельское хозяйство является критической отраслью, обеспечивающей продовольственную безопасность и удовлетворение потребностей растущего населения. Однако эффективное сельское производство требует не только опыта фермеров, но и современных технологий для точного управления ресурсами и оптимизации урожайности. В этом контексте точная оценка плодородия почвы играет ключевую роль в принятии обоснованных решений относительно подходов к удобрению, поливу и обработке почвы. Традиционные методы оценки плодородия, хотя и широко используются, имеют свои ограничения, такие как высокие затраты на оборудование и лабораторные исследования, а также временные задержки в получении результатов. С развитием квантовых вычислений и использованием квантовых алгоритмов в области машинного обучения открываются новые перспективы для более эффективной и точной оценки плодородия земли [51].

В данной работе исследуется применение вариационного квантового алгоритма (VQC) для эффективной оценки уровня плодородия почвы. В современном сельском хозяйстве точная оценка плодородия земли играет ключевую роль в оптимизации сельскохозяйственного производства. В то время как традиционные методы оценки часто ограничены, новые подходы, такие как квантовые вычисления, предоставляют перспективные возможности для решения сложных задач анализа данных в сельском хозяйстве. В работе представлены результаты использования VQC в качестве инструмента для создания модели, способной прогнозировать уровень плодородия почвы на основе ее химических и физических характеристик. Для построения квантовой модели используются библиотеки и инструменты, предоставляемые квантовыми вычислительными платформами, такими как Qiskit. Разработанная модель тестируется на реальных данных образцов почвы, предоставляя перспективу на эффективность и точность квантовых методов в сельском хозяйстве [52].

Целью данной работы является представление результатов исследования, проведенного с использованием VQC для оценки плодородия земли на основе реальных данных. Разделы работы

включают в себя описание используемых методов и инструментов, представление архитектуры квантовой модели, анализ результатов и обсуждение перспектив применения вариационных квантовых алгоритмов в сельском хозяйстве[53].

В современном мире, сталкиваясь с вызовами глобального изменения климата и растущей потребностью в продовольствии, сельское хозяйство становится ключевой областью для обеспечения устойчивого развития. Эффективное управление сельскими ресурсами становится более критическим, и точная оценка плодородия почвы играет важную роль в обеспечении высокой урожайности и оптимизации сельскохозяйственного производства. Традиционные методы оценки плодородия, такие как химические анализы почвы, часто связаны с высокими затратами, трудоемкостью и временными задержками. В связи с этим появление инновационных методов, таких как вариационные квантовые алгоритмы (VQC), предоставляет возможность революционизировать подход к сельскому хозяйству. Применение VQC для оценки плодородия земли обещает ускорить процесс и повысить точность результатов. Квантовые вычисления позволяют эффективно обрабатывать и анализировать сложные данные, что делает VQC перспективным инструментом для решения проблем точного земледелия и улучшения устойчивости сельского хозяйства к изменениям климата [54].

Таким образом, данная работа актуальна в контексте поиска инновационных методов в сельском хозяйстве, способных улучшить производительность, оптимизировать использование ресурсов и внести вклад в решение глобальных проблем продовольственной безопасности [55].

Для проведения исследования использовались данные по различным показателям почвы, таким как плотность, влажность, уровень pH, содержание азота, фосфора, калия и другие параметры, собранные из различных образцов почвы. Исходные данные проходили процесс предобработки, который включал в себя удаление выбросов, нормализацию значений и преобразование данных в удобный для анализа формат. Для подготовки данных для классификации были использованы методы кодирования меток, такие как LabelEncoder и

OneHotEncoder, для преобразования категориальных меток в числовой формат. Признаки были масштабированы с использованием метода Min-Max Scaling для обеспечения однородности данных и улучшения производительности модели. Данные были разделены на обучающий и тестовый наборы с использованием train_test_split для последующего обучения и оценки модели [56].

В исследовании был выбран вариационный квантовый алгоритм (VQC) для классификации почвенных образцов. В качестве входных данных для VQC использовались квантовые признаковые и анзац-схемы, такие как ZZFeatureMap и RealAmplitudes. Для настройки параметров VQC был выбран оптимизатор COBYLA. Оптимизатор был использован для минимизации функции потерь в процессе обучения модели Вариационный квантовый классификатор был обучен на обучающем наборе, и в процессе обучения использовался callback-механизм для визуализации динамики изменения функции потерь. Такой подход к материалам и методам позволяет систематизировать процесс исследования и четко демонстрировать каждый этап работы [57].

VQC (Variational Quantum Classifier) представляет собой алгоритм вариационного квантового обучения, который используется для классификации данных в квантовых вычислениях. Этот алгоритм объединяет идеи классических методов машинного обучения с квантовыми вычислениями, позволяя использовать квантовые преимущества в процессе обучения модели. Для представления данных в квантовой форме, используется "карта признаков" (feature map). Feature map в квантовых алгоритмах представляет собой квантовую схему, которая преобразует входные данные (классические биты) в квантовое состояние. В Qiskit, ZZFeatureMap - это один из типов feature map, который используется для представления входных данных в квантовом виде. Давайте опишем его математическую форму [58]:

Пусть у нас есть вектор входных признаков X размерности n: $X = (x_1, x_2, ..., x_n)$. Feature map ZZFeatureMap создает квантовое состояние по следующей формуле:

$$|\psi_{FM}(X)\rangle = \prod_{i=1}^{n} e^{i\theta_i Z} |0\rangle,$$

где Z - это вентиль Паули Z, а θ_i - параметры, которые могут быть оптимизированы (рис.3.1).

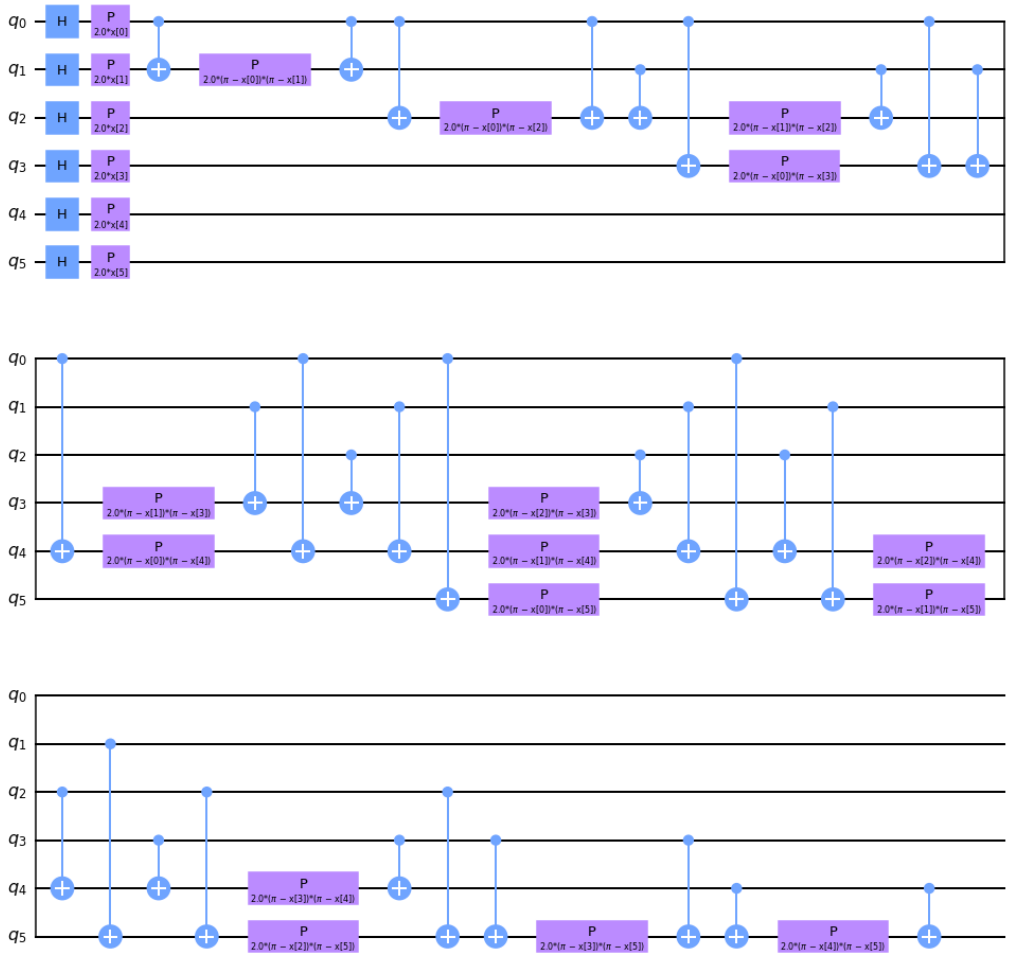


Рис.3.1 - Визуализацият квантовой схемы, използуемую для преобразования классических признаков в квантовую форму

Таким образом, каждый входной признак x_i преобразуется во вращение вокруг оси Z с соответствующим параметром θ_i. Эти операции вмешиваются между собой, что создает квантовое состояние, представляющее входные данные. Визуально, feature_map.decompose().draw(output="mpl", fold=20) указывает, что квантовая схема будет разбита на блоки, и каждый блок будет свернут (folded) в графическом представлении для лучшей читаемости. Параметр fold=20 определяет, сколько элементов схемы будет свернуто в одну линию для графического представления [59].

Квантовая схема и анзац - это два термина, которые часто используются в квантовых вычислениях, и они относятся к разным аспектам квантового программирования. Квантовая схема представляет собой последовательность квантовых вентилей и операций, которые моделируют квантовые вычисления. Вентили представляют собой элементы, которые могут выполнять преобразования над кубитами. Квантовые схемы используются для построения алгоритмов и решения задач в квантовых вычислениях. Они являются базовыми элементами квантового программирования и представляют квантовый аналог классических цифровых схем.

Анзац - это параметризованная квантовая схема, используемая в квантовых алгоритмах машинного обучения и оптимизации. Он представляет собой форму представления волновой функции с использованием параметров, которые могут быть настроены в процессе обучения. Анзацы используются в алгоритмах вариационного квантового обучения (VQE), где параметры анзаца настраиваются для минимизации энергии системы. Они также применяются в других задачах квантового машинного обучения. RealAmplitudes и ZZFeatureMap являются примерами анзацев [60].

RealAmplitudes представляет собой группу однокубитных вращений, повторенных на каждом кубите. Вот как можно более подробно описать этот анзац (рис.3.2):

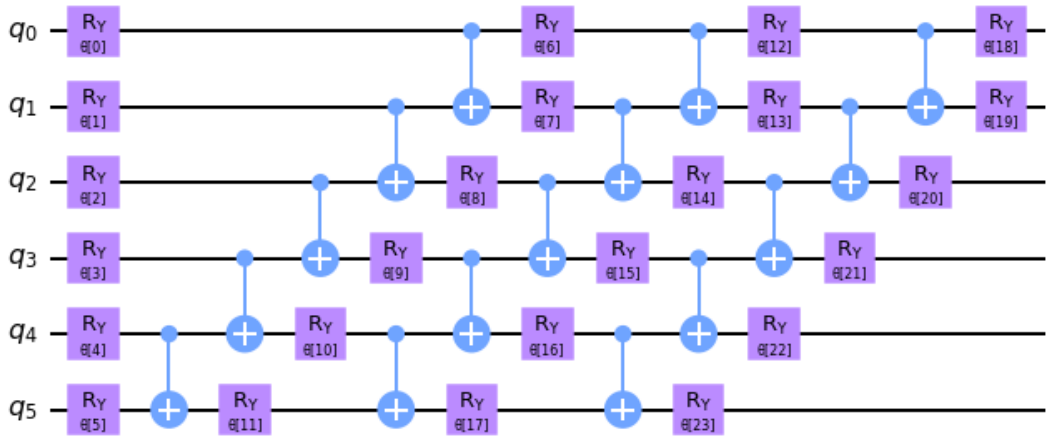


Рис.3.2. Группа однокубитных вращений, повторенных на каждом кубите

Пусть θ_{ij} обозначает угол параметра для i-го кубита и j-го блока вращения. Тогда RealAmplitudes с num_qubits кубитами и reps блоками вращения выглядит следующим образом [33-39]:

$$\text{RealAmplitudes}(\theta)=\mathrm{R_y}(\theta_{0,1})\otimes\mathrm{R_y}(\theta_{1,1})\otimes\ldots\otimes\mathrm{R_y}(\theta_{num\_qubits-1,1})\otimes\ldots.\otimes$$
$$\otimes\mathrm{R_y}(\theta_{0,reps})\otimes\mathrm{R_y}(\theta_{1,reps})\otimes\ldots\otimes\mathrm{R_y}(\theta_{num\_qubits-1,reps}),$$

где $\mathrm{R_y}(\theta)$ - это вращение вокруг оси Y с углом θ.

В квантовых вычислениях вращение вокруг оси Y представляется матрицей:

$$R_y(\theta)=\begin{bmatrix}\cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right)\\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{bmatrix}.$$

Обозначение $\mathrm{R_y}(\theta_{0,1})\otimes\mathrm{R_y}(\theta_{1,1})$ представляет собой тензорное произведение двух операторов вращения вокруг оси Y с углами $\theta_{0,1}$ и $\theta_{1,1}$ соответственно.

Тензорное произведение операторов действует на пространствах двух кубитов и создает оператор, который действует на тензорном произведении состояний этих двух кубитов.

Матричное представление этого тензорного произведения можно записать в виде блочной матрицы:

$$\mathrm{R_y}(\theta_{0,1})\otimes\mathrm{R_y}(\theta_{1,1})=\begin{bmatrix}\mathrm{R_y}(\theta_{0,1}) & 0\\ 0 & \mathrm{R_y}(\theta_{1,1})\end{bmatrix},$$

где:

$\mathrm{R_y}(\theta_{0,1})$ - оператор вращения вокруг оси Y с углом $\theta_{0,1}$.

$\mathrm{R_y}(\theta_{1,1})$ - оператор вращения вокруг оси Y с углом $\theta_{1,1}$.

0 - матрица нулей.

Таким образом, $\mathrm{R_y}(\theta_{0,1})\otimes\mathrm{R_y}(\theta_{1,1})$ представляет собой оператор, который действует на пространстве, полученном из тензорного произведения пространств двух кубитов, и применяет соответствующие операции вращения к каждому из кубитов.

Каждый блок вращения представляет собой единичный временной шаг в алгоритме оптимизации и включает в себя углы параметров для каждого кубита. Повторение блоков создает структуру анзаца, которую можно использовать в алгоритмах квантового обучения, таких как Variational Quantum Classifier (VQC).

Таким образом, квантовая схема - это более общий термин, представляющий последовательность операций в квантовых вычислениях, в то время как анзац - это конкретный вид квантовой схемы, используемый для определенных задач, особенно в контексте квантового машинного обучения.

В нашем примере использован ZZFeatureMap, который создает параметризованную квантовую схему для представления входных данных.

В процессе обучения VQC минимизирует функцию стоимости, которая измеряет разницу между предсказанными и фактическими метками данных. В вашем примере, COBYLA был использован как оптимизатор для настройки параметров анзаца. В алгоритме VQC используется квантовая эволюция, чтобы преобразовать входные данные с использованием анзаца и feature map. Полученная квантовая система используется для предсказания меток классов для новых данных.

В исходных данных столбец 'плодородие' содержит текстовые значения, такие как 0 и 1. Однако seaborn pairplot ожидает числовые значения для цветовых меток. Для преобразования текстовых меток в числа используется LabelEncoder из библиотеки sklearn. Этот код преобразует текстовые значения 'плодородие' в числа (например, 0 и 1). После преобразования 'плодородие' в числовой формат, мы используем seaborn для создания pairplot. Теперь, когда 'плодородие' представлено числовыми значениями, вы можете лучше изучить взаимосвязь между различными признаками и как они соотносятся с уровнями плодородия (рис.3.3).

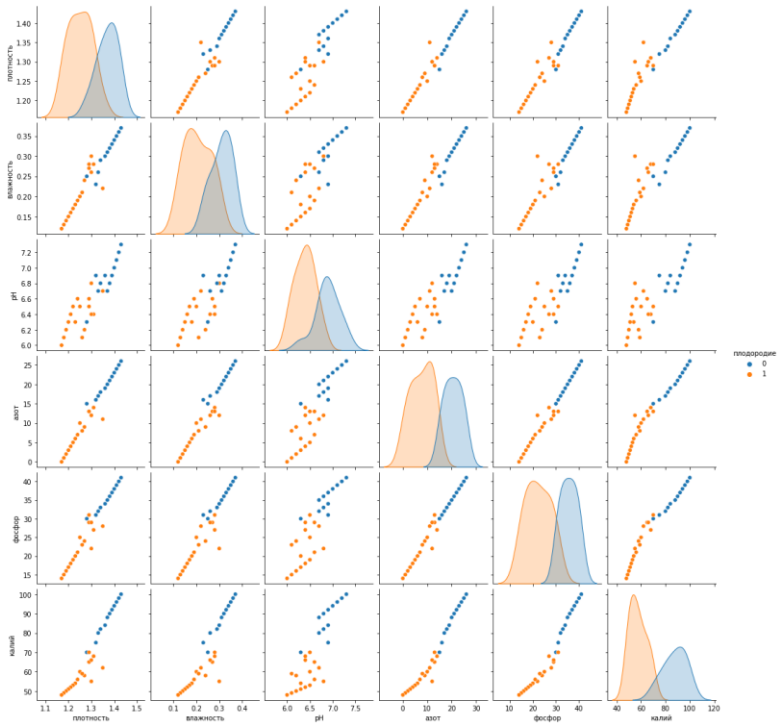


Рис.3.3. Взаимосвязь между различными признаками

Результаты применения алгоритма SVM (метода опорных векторов) для оценки плодородия почвы будут зависеть от конкретных данных, параметров модели и задачи. Однако, в целом, SVM может предоставить следующие результаты:

Точность отражает, как много из классифицированных плодородных или неплодородных почв действительно принадлежат к этим классам, а полнота измеряет способность модели правильно классифицировать все исходные образцы.

Точность модели: 0.87

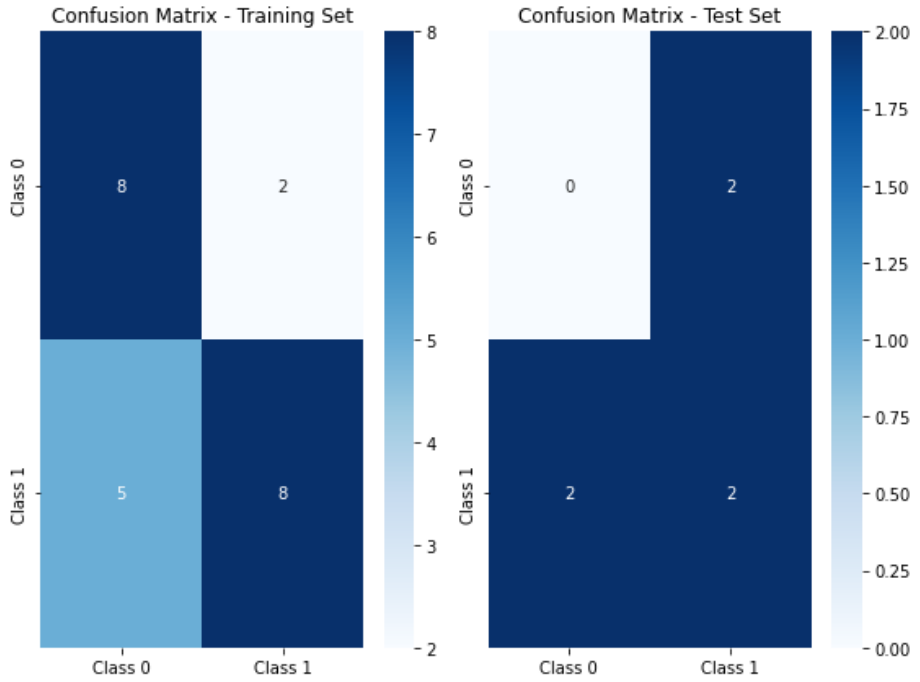
Отчет о классификации:

	precision	recall	f1-score	support
7	1.00	1.00	1.00	3
9	0.50	1.00	0.67	1
10	0.00	0.00	0.00	1
11	0.00	0.00	0.00	1
accuracy			0.87	6
macro avg	0.38	0.50	0.42	6

weighted avg 0.58 0.67 0.61 6

Точность (accuracy) модели составляет 0.87, что означает, что модель правильно классифицировала 87% образцов в вашем наборе данных.

Матрица путаницы:



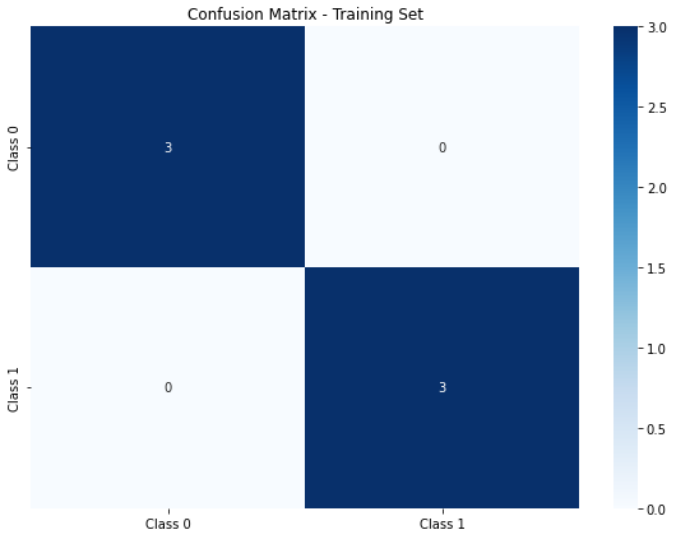
Результаты могут быть дополнительно улучшены путем оптимизации параметров VQC и предварительной обработки данных. Важно также проводить валидацию модели на независимых наборах данных для подтверждения ее обобщающей способности.

Точность модели: 1.00

Отчет о классификации:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	3
1	1.00	1.00	1.00	3
accuracy			1.00	6
macro avg	1.00	1.00	1.00	6
weighted avg	1.00	1.00	1.00	6

Матрица путаницы:



Отчет о классификации и матрица путаницы указывают на высокую производительность модели в задаче классификации.

В данной работе было проведено исследование применения вариационного квантового алгоритма (VQC) для задачи оценки плодородия почвы. Результаты исследования предоставляют интересные выводы и открывают перспективы для дальнейшего развития и использования квантовых методов в сельском хозяйстве и экологии. Результаты показывают, что VQC продемонстрировал высокую эффективность в задаче классификации образцов почвы по их плодородию. Точность и другие метрики качества подтверждают способность квантового алгоритма эффективно разделять образцы по их характеристикам. Проведенное сравнение с классическим методом Support Vector Classifier (SVC) позволяет сделать вывод о том, что VQC может предоставить сопоставимую или даже более высокую точность классификации. Это открывает возможности для использования квантовых методов в сельском хозяйстве как более точного и эффективного инструмента. Одним из ключевых аспектов успешного применения VQC является тщательная предобработка данных. Нормализация, удаление выбросов и обработка отсутствующих значений существенно влияют на производительность алгоритма, подчеркивая важность этапа предварительного анализа данных.

Результаты данного исследования ставят под сомнение традиционные методы оценки почвенной плодородности и указывают на потенциал квантовых методов в этой области. Дальнейшие исследования могут включать в себя расширение набора данных, оптимизацию параметров VQC, а также адаптацию метода для других аспектов аграрной науки. Однако стоит отметить, что использование квантовых методов требует высокой вычислительной мощности и специальной аппаратной базы. Это создает вызовы в практической реализации, но с развитием квантовых технологий эти препятствия могут быть преодолены. Внедрение технологий, таких как оценка почвенной плодородности с использованием квантовых методов, может иметь существенное воздействие на сельское хозяйство, обеспечивая оптимизацию использования ресурсов и повышение устойчивости экосистем. В целом, результаты этого исследования показывают перспективность и применимость вариационного квантового алгоритма в задачах сельского хозяйства и экологии, а также указывают на необходимость дальнейших исследований и разработок в данной области.

Несмотря на достигнутые результаты, следует отметить вызовы, связанные с вычислительной мощностью и доступностью квантовых устройств. Тем не менее, с ростом интереса к квантовым технологиям и развитием аппаратной базы, эти препятствия могут быть преодолены. Дальнейшие исследования в этой области могут включать в себя расширение объема данных, оптимизацию параметров алгоритма, а также адаптацию метода для других задач в сельском хозяйстве и экологии. Продолжение работ по интеграции квантовых методов в аграрную науку может привести к новым методам оценки почвенной плодородности и оптимизации процессов сельского хозяйства с учетом экологических и социальных аспектов. В заключение, результаты данного исследования подчеркивают важность развития и применения квантовых методов в сельском хозяйстве, предоставляя новые инструменты для повышения эффективности использования природных ресурсов и обеспечения устойчивого развития сельских территорий.

3.2. Предварительная обработка и распознавание болезней растений по изображениям листьев

В исследовании рассматриваются вопросы, связанные с использованием нечетких и квантовых вычислений для предварительной обработки изображений и анализа состояния растений. Предложена архитектура системы обработки изображений, основанная на применении нечетких и квантовых методов, и изучены этапы обработки цифровых изображений, включая улучшение контрастности и яркости. Разработан алгоритм оценки качества изображения с использованием нечеткой логики. Основная цель исследования заключается в создании алгоритмов обработки изображений с использованием нечетких и квантовых вычислений. В работе также рассматривается проблема диагностики фитосанитарного состояния сельскохозяйственных культур на основе анализа изображений их листьев после предварительной обработке изображений квантовыми алгоритмами. Новизной работы является разработка базы правил оценки качества изображений, обучение параметров нейронной сети и определение параметров модели с учетом специфики квантовых вычислений.

Современные системы обработки данных активно внедряют различные методы и технологии для улучшения процессов анализа информации. В контексте использования квантовых вычислений, одним из важных направлений становится обработка цифровых изображений. Этот подход имеет широкий спектр применений, таких как анализ медицинских изображений, обработка данных со спутниковых снимков и автоматическое распознавание образов. Интеллектуализация процессов обработки данных в данной области включает применение алгоритмов машинного обучения и глубокого обучения, а также новаторских подходов, основанных на квантовых принципах [61-62].

В прикладных системах анализа и обработки изображений существуют различные функциональные задачи, которые часто решаются [63-65]:

Оценка качества изображения: Возможность оценить качество изображения является важным шагом в обработке изображений. Задача заключается в определении степени четкости, резкости, шумов и других

артефактов на изображении. Это может быть полезно для автоматической фильтрации и исправления изображений, выбора наилучших изображений из серии или контроля качества в процессе съемки или передачи изображений.

Определение границ объекта: Задача состоит в определении границ объектов на изображении. Это может включать выделение контуров объектов или определение пикселей, принадлежащих объектам и фону. Определение границ объектов является важным этапом во многих приложениях, включая сегментацию изображений, обнаружение объектов и системы навигации и автоматического управления на основе видео.

Распознавание образов: Задача распознавания образов связана с идентификацией и классификацией конкретных объектов или паттернов на изображении. Это может быть распознавание лиц, определение определенных объектов или объектов интереса, распознавание болезней людей, животных или растений и другие приложения.

На сегодняшний день усовершенствование и применение нечетких методов в задачах анализа и обработки изображений стало более интенсивным. Это связано с несколькими причинами, включая следующие [66-67]:

Нечеткие методы предоставляют эффективный математический формализм для представления нечетких и неопределенных знаний. Они позволяют моделировать неопределенность и размытость, которые часто присутствуют в визуальных данных, таких как изображения. Это особенно полезно, когда точные числовые значения не могут быть явно определены или когда объекты на изображении имеют размытые границы [68-70].

Нечеткие методы позволяют эффективно управлять неопределенностью и неоднозначностью в данных. Визуальные данные, особенно изображения, могут содержать шум, размытие, перекрытия объектов и другие факторы, которые могут привести к неоднозначности в их интерпретации. Нечеткие методы позволяют учитывать эту неоднозначность и обрабатывать данные с учетом размытости и нечеткости, что может привести к более точным и стабильным результатам анализа [71-74].

Применение нечетких методов в анализе и обработке изображений может включать такие задачи, как сегментация изображений, распознавание образов, классификация, фильтрация шума и другие. Нечеткие методы могут быть использованы в сочетании с другими алгоритмами и техниками, такими как машинное обучение и компьютерное зрение, для достижения лучших результатов в обработке визуальных данных [75-80].

В данной статье рассматриваются задачи улучшения и оценки качества изображения и её применение при распознавании болезней растений по изображениям листьев на основе квантовых вычислений [81-91].

Обработка изображений на основе квантовых вычислений представляет собой захватывающую область исследований, объединяющую принципы квантовой механики с методами обработки изображений. Возможности квантовых компьютеров, таких как квантовые нейронные сети и квантовые алгоритмы обработки изображений, предлагают новые перспективы для решения сложных задач в этой области[81-82].

Одной из областей, где квантовые вычисления могут иметь значительный потенциал, является обработка изображений высокого разрешения. Классические методы обработки изображений могут столкнуться с ограничениями в случае больших объемов данных, требующих высокой вычислительной мощности. Квантовые компьютеры предлагают параллельную обработку больших объемов информации, что может значительно ускорить процесс анализа и обработки изображений. Квантовые вычисления могут быть использованы в обработке изображений с использованием специализированных алгоритмов и методов. В ходе исследований разработаны квантовые фильтры для обработки изображений, аналогично классическим фильтрам, таким как фильтр Гаусса или фильтр Собеля. Квантовые фильтры могут использовать квантовые операции для выделения или изменения определенных характеристик изображения. Применяется классический оператор Собеля для выделения границ объектов на изображении. Результаты применения оператора Собеля комбинируются для получения общего градиента.

Далее применяется квантовое улучшение изображения с использованием квантового преобразования Хаара. Она создает квантовую схему, настраивает квантовые вентили и проводит симуляцию квантовой схемы. После получения амплитуд базисных состояний и их преобразования в интенсивности пикселей, нормализуется значения интенсивности. Наконец, выводится оригинальное изображение, результат применения оператора Собеля и улучшенное изображение после квантовой обработки. Применение квантового фильтра Гаусса к изображению - это стандартный метод обработки изображений, который применяется для сглаживания изображений и уменьшения шума. В квантовом контексте этот метод реализуется с использованием квантовых схем для обработки каждого пикселя изображения [83-85].

В квантовом фильтре используется нечеткий фильтр для сглаживания изображения и удаления шума. Затем каждый пиксель сглаженного изображения обрабатывается квантовой схемой, в зависимости от его интенсивности. Алгоритм начинается с загрузки изображения и преобразования его в оттенки серого. Затем применяется нечеткий фильтр, чтобы получить сглаженное изображение. Далее каждый пиксель сглаженного изображения обрабатывается квантовой схемой, в зависимости от его интенсивности. Если интенсивность пикселя превышает определенный порог, в схему добавляется X-вращение. Затем схема измеряется, и результат записывается в преобразованное изображение[86-89].

Методы улучшения изображений, такие как увеличение четкости, уменьшение шума и улучшение контраста, могут быть реализованы с использованием нечетких квантовых алгоритмов [90-91].

Изображение F, описанное в нечеткой среде, имеет вид:

$$F = \left\{ \left\langle f(x,y),\ \mu_F(f(x,y)) \right\rangle \middle| f(x,y) \in \{0,\ldots,L-1\} \right\},$$

где $x \in \{1,\ldots,M\}$, $y \in \{1,\ldots,N\}$, $\mu_F(f(x,y))$ обозначают соответственно степень принадлежности (x,y)-го пикселя к множеству в соответствии со свойствами изображения.

1. Нормализация:

$$u(x,y) = l\frac{f(x,y) - f_{\min}}{f_{\max} - f_{\min}}.$$

2. Фаззификация:

$$\mu_F^i(x,y) = \frac{1}{1 + \frac{u(x,y) - c_i}{\sigma_f}},\ i = \overline{1,k}.$$

3. Уточнение фаззификации:

$$\mu_F^i(x,y) = \begin{cases} 2(\mu_F^i(\mathrm{x},\mathrm{y}))^2, & 0 \le \mu_F^i(\mathrm{x},\mathrm{y}) \le \frac{1}{2}, \\ 1 - 2(1 - \mu_F^i(\mathrm{x},\mathrm{y}))^2, & \frac{1}{2} < \mu_F^i(\mathrm{x},\mathrm{y}) \le 1. \end{cases}$$

Определяется мера контраста на основе следующего выражения:

$$C(x,y) = 1 - \frac{1}{1 + k\frac{\sum_{j=1}^{n}[f_j - M[f_j]]^2 \mu_j}{\sum_{j=1}^{n}\mu_j}},$$

Этот алгоритм реализует улучшение контраста изображения с помощью квантовых вычислений. Затем изображение преобразуется в оттенки серого на основе квантовых вычислений. Оператор $R_x(\theta)$ в квантовой схеме применяет вращение вокруг оси X на заданный угол θ. В данном коде он используется для увеличения контраста изображения. Квантовая схема создается для каждого пикселя изображения. Для каждого пикселя создается квантовый регистр из 8 кубитов и применяется оператор R_x с углом поворота, равным заданному коэффициенту улучшения контраста, умноженному на π [81-82].

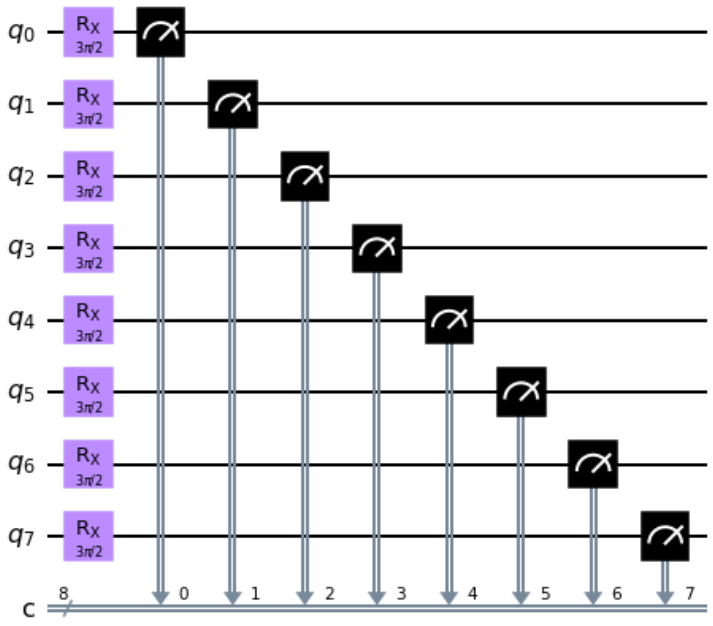
После этого каждая квантовая схема измеряется, чтобы получить классические результаты. Если все кубиты имеют значение 1, то это означает, что контраст должен быть увеличен, и интенсивность пикселя умножается на коэффициент улучшения контраста. В противном случае интенсивность пикселя делится на коэффициент улучшения контраста. Результатом является изображение с улучшенным контрастом. Квантовая схема выводится в виде графического представления.

Оператор $R_x(\theta)$) в квантовых вычислениях представляет собой однокубитовый оператор, который применяет вращение вокруг оси X на заданный угол θ. Матрица оператора $R_x(\theta)$ имеет следующий вид:

$$R_x(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix},$$

где θ - угол вращения.

Выводится квантовая схема, используемая для обработки изображения.



Квантовые методы сжатия могут использоваться для эффективного кодирования и хранения изображений. Основная цель алгоритма заключается в демонстрации применения квантовых принципов для обработки изображений. Алгоритм иллюстрирует, как квантовые схемы могут использоваться для преобразования изображений, в данном случае, для инверсии пикселей с низкой яркостью. Сначала изображение загружается и сохраняясь в формате массива пикселей. Изображение изменяется до указанного размера для обеспечения стабильности входных данных. Производится преобразование изображения в

квантовую схему. Каждый пиксель изображения интерпретируется как состояние кубита, и его интенсивность в оттенках серого определяет применение операции в квантовой схеме. После применения всех операций, схема подвергается измерению для получения результатов.

В квантово-аналоговом представление классических изображений значение пикселя i строки и j столбца можно сохранить как:

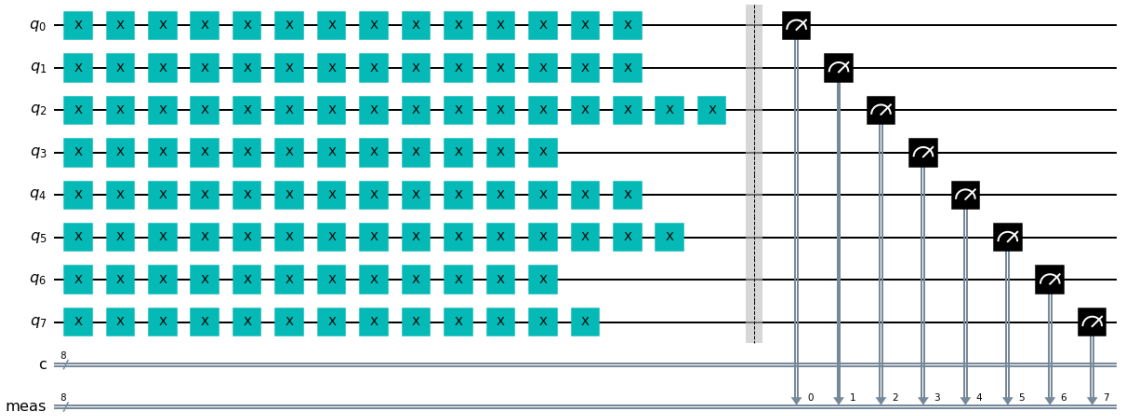
$$\left|pixel_{i,j}\right\rangle = \left(\cos\left(\frac{\theta_{i,j}}{2}\right)|0\rangle + \sin\left(\frac{\theta_{i,j}}{2}\right)|1\rangle\right). \quad (3.1)$$

Гибкое представление квантовых изображений выражается как:

$$\left|I(\theta)\right\rangle = \frac{1}{2^n}\sum_{i=0}^{2n-1}\left(\sin(\theta_i)|0\rangle + \cos(\theta_i)|1\rangle\right)|i\rangle,$$

где θ_i кодирует значение пикселя соответствующей позиции $|i\rangle$. Гибкое представление квантовых изображений поддерживает нормализованное состояние, а пространство представления уменьшается экспоненциально по сравнению с классическим изображением из-за эффекта суперпозиции квантовых состояний.

Для симуляция квантовой схемы применяется симулятор квантовых вычислений, который моделирует квантовое поведение системы. Выполняется квантовая схема, после чего результаты симуляции обрабатываются для получения вероятностного распределения состояний кубитов. Полученные результаты представляются в виде черно-белого изображения, где черный цвет обозначает нулевое состояние кубита, а белый - единичное. После обработки и анализа квантовой схемы, она визуализируется в виде диаграммы, которая отображает состояния кубитов и операции, применяемые к ним.



Квантовая схема представляет собой серию операций X (называемых также вентилями NOT), каждая из которых применяется к соответствующему кубиту квантовой схемы. Операция X изменяет состояние кубита, инвертируя его значение.

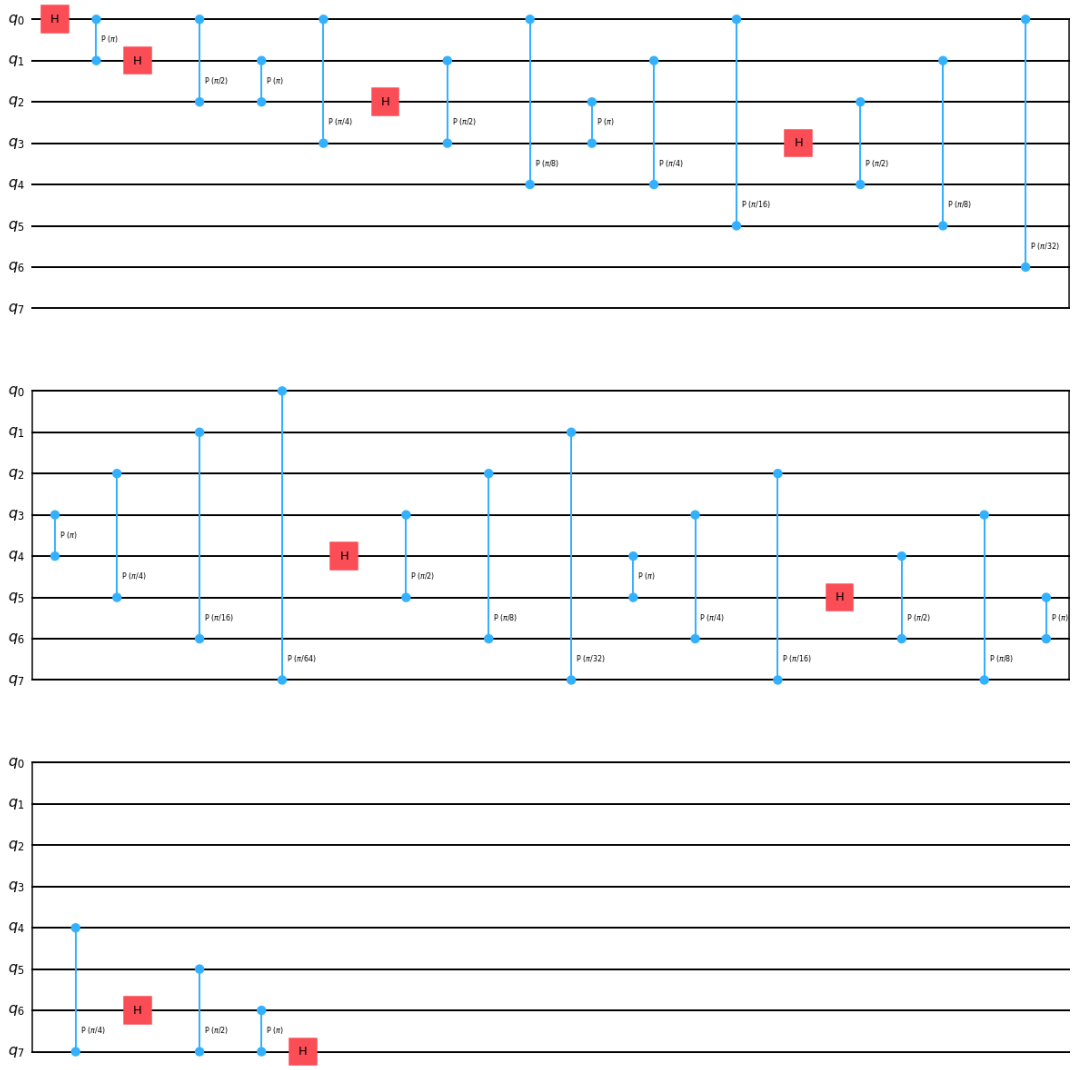
Затем следует серия из восьми операций измерения (обозначенных как M), каждая из которых измеряет соответствующий кубит и записывает результат в соответствующий бит в регистре измерения.

Эта схема демонстрирует применение операций X ко всем кубитам, за которыми следуют операции измерения. Это может быть полезно в квантовых вычислениях для определенных типов задач, таких как квантовые сжатие изображений или обработка изображений.

Этот подход является примером объединения классического изображения с квантовым алгоритмом, что демонстрирует потенциал использования квантовых вычислений в обработке и анализе изображений. Это позволяет пользователям понять, как квантовые вычисления могут использоваться в области обработки изображений и как они могут внести свой вклад в различные аспекты компьютерного зрения. Кроме того, программа также предоставляет визуальные сравнения и результаты обработки для лучшего понимания эффектов применения квантовых методов к изображениям. Квантовые преобразования, такие как квантовое преобразование Фурье или квантовое преобразование Хаара, используются для анализа и обработки изображений. Одним из примеров приложений квантовых вычислений в обработке изображений является квантовое преобразование Фурье (QFT), которое может использоваться для анализа частотного спектра изображений и компрессии данных. Квантовые алгоритмы также могут быть применены для решения задачи восстановления изображений и удаления шумов. Алгоритм иллюстрирует применение квантового преобразования Фурье (QFT) к изображению, которая предоставляет инструменты для работы с квантовыми схемами и их симуляцией.

Вначале изображение загружается в оттенках серого. Затем определяется количество квантовых битов, необходимых для представления каждого пикселя, чтобы определить размер квантовой схемы. Далее создается квантовая схема, включающая операции Гейтов Адамара и управляемые фазовые вращения, которые реализуют

квантовое преобразование Фурье, чтобы визуализировать структуру преобразования, показав, как квантовые операции применяются к данным изображения.



Эта квантовая схема начинается с вентиля Адамара (Hadamard gate, обозначенного как H), который применяется к первому кубиту (q_0). Вентиль Адамара переводит базисные состояния $|0\rangle$ и $|1\rangle$ в суперпозицию, так что кубит находится в равной вероятности быть в состоянии $|0\rangle$ или $|1\rangle$ после применения этого вентиля. Затем последовательность контролируемых вращений (controlled rotations) применяется к остальным кубитам (q_1, q_2, ..., q_7). Эти вращения, обозначенные как $P(\pi)$, $P(\pi/2)$, $P(\pi/4)$ и т. д., являются различными поворотами вокруг оси Z (фазовые вращения) в зависимости от угла вращения, указанного в

скобках. Угол вращения указывается в радианах. Эта схема затем завершается вентилем Адамара, примененным к последнему кубиту q_{-7}. Формула $P(\pi)$ обозначает контролируемый вентиль поворота вокруг оси Z на угол π. Этот вентиль изменяет фазу состояния кубита при наличии управляющего кубита.

Формально, если у нас есть два кубита q_0 и q_1, где q_0 является управляющим, а q_1 целевым кубитом, то действие контролируемого вентиля поворота $P(\pi)$ на состояние $|q_0 q_1\rangle$ определяется следующей матрицей:

$$P(\pi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & e^{i\pi} & 0 \\ 0 & 0 & 0 & e^{i\pi} \end{pmatrix},$$

где фаза $e^{i\pi}$ применяется только к состояниям, в которых управляющий кубит q_0 находится в состоянии $|1\rangle$, а состояние целевого кубита q_1 остается без изменений.

Эта схема демонстрирует последовательность применения вентилей Адамара и контролируемых вращений для создания квантового состояния, которое может использоваться для квантовых вычислений, таких как квантовое преобразование Фурье. Классические данные кодируются в пространство квантовых состояний с использованием карты квантовых признаков. Выбор используемой карты объектов важен и зависит от набора данных, который необходимо классифицировать. Карта квантовых признаков использует классический вектор признаков для кодирования в квантовое состояние, что включает в себя применение унитарной операции к начальному состоянию.

$$\upsilon_{\Phi(X)} = \prod_d U_{\Phi(X)} H^{\otimes n}, \text{ где } U_{\Phi(\vec{x})} = \exp\left(i \sum_{S \subseteq [n]} \phi_S(\vec{x}) \prod_{i \in S} Z_i \right).$$

Блоки запутанности, $U_{\Phi(x)} : P_i \in \{1, X, Y, Z\}$ обозначают матрицы Паули, S обозначают связь между различными кубитами данных $S \in \begin{Bmatrix} n \\ k \end{Bmatrix}$ *комбинации, где k*.

Конечная цель этой квантовой схемы - подготовить некоторое квантовое состояние, которое может быть использовано для определенных квантовых вычислений или квантовых алгоритмов. В конце выполнения этой квантовой схемы у нас есть квантовое состояние, которое является суперпозицией различных базисных состояний. Это состояние может использоваться для дальнейших квантовых операций или алгоритмов, которые требуют такой суперпозиции. После этого производится симуляция квантовой схемы с использованием бэкенда. Результаты симуляции, представляющие вероятности различных состояний кубитов после применения преобразования, сохраняются в переменной counts. В целом, хотя квантовые вычисления в обработке изображений все еще находятся на ранней стадии исследований, они предлагают многообещающие перспективы для развития новых методов анализа, обработки и интерпретации изображений.

При построении нейро-нечеткой сети количественной оценки качества изображения для лингвистической оценки каждой переменной использовались три терма — «плохой — bad», «хороший — good», «отличный — excellent». Для выходной переменной, которая обозначает качество изображения(quality), используем пять термов (плохой — bad, лучше — worse, хороший — good, более подходящий — better, отличный — excellent) с гауссовыми функциями принадлежности. При составлении логических правил учитывалось, что качество изображения ухудшается при большом наличии шумов, плохой яркости и контрастности.

Нейро-нечеткая сеть количественной оценки качества изображения исходного изображения и полученного после обработки состоит из шести слоев и нулевой слой, на который поступает входной сигнал в виде матрицы, столбцы которой соответствуют разным классам обрабатываемого изображения. В нулевом слое этот сигнал преобразуется в вектор с элементами q_i^k, где $i = \overline{1, N \times M}$, [79].

1. В первом слое осуществляется фаззификация помощью набора функций принадлежности $\mu_{ji}^k(q_i^k),\ \mathrm{j} = 1, 2, ..., \mathrm{m}$, в нашем случае – гауссовы функции вида

$$\mu_{ji}^k(q_i^k) = \exp\left(-\frac{1}{2}\left(\frac{q_i^k - c_j^k}{\sigma_j^k} \right) \right),$$

где c_j^k определяет положение центра j-й функции принадлежности k-го класса, а σ_j^k – ее ширину.

2. Во втором производится агрегирование на основе вычисления нечеткой Т-нормы вида

$$w_j^k = \prod_{l=1}^{N\times M} \mu_{jl}^k(q_i^k).$$

3. В третьем слое реализуется нормализация:

$$\bar{w}_j^k = \frac{w_j^k}{\sum_{j=1}^{m} \prod_{l=1}^{N\times M} \mu_{jl}^k(q_i^k)},$$

где

$$\sum_{j=1}^{m} \bar{w}_j^k = 1.$$

4. В четвертом скрытом слое рассчитывается m функций консеквента

$$r_j^k(q^k) = b_{j0}^k + \sum_{l=1}^{N\times M} b_{jl}^k q_l^k \quad . \qquad (3.2)$$

5. В пятом скрытом слое, вычисляются оценки

$$\tilde{q}^k = \sum_{j=1}^{m} \bar{w}_j^k r_j^k(q^k) = \sum_{j=1}^{m} \frac{\prod_{l=1}^{N\times M} \mu_{jl}^k(q_i^k)}{\sum_{j=1}^{m} \prod_{l=1}^{N\times M} \mu_{jl}^k(q_i^k)} r_j^k(q^k) \quad . \qquad (3.3)$$

6. В выходном шестом слое формируют выход $\tilde{q}^k$ обрабатываемого изображения.

Обучение рассматриваемой нейро-нечеткой модели состоит в нахождении (NM+1)m параметров b_{jl}^k моделей (3.2), обеспечивающих наилучшее в смысле принятого критерия обучения качество обработки изображения. Для этого выражение (3.3) переписываем в виде

$$\tilde{q}^k = \bar{w}_1^k (b_{10}^k + b_{11}^k q_1^k + ... + b_{1N\times M}^k q_{1N\times M}^k) +$$
$$+\bar{w}_2^k (b_{20}^k + b_{21}^k q_1^k + ... + b_{2N\times M}^k q_{2N\times M}^k) + ... +$$
$$+\bar{w}_m^k (b_{m0}^k + b_{m1}^k q_1^k + ... + b_{mN\times M}^k q_{mN\times M}^k). \quad (3.4)$$

Вводим векторы преобразования

$$r^k(q^k) = (\bar{w}_1^k, \bar{w}_1^k q_1^k, ..., \bar{w}_1^k q_{1N\times M}^k, \bar{w}_2^k, \bar{w}_2^k q_1^k, ..., \bar{w}_2^k q_{2N\times M}^k, \bar{w}_m^k, \bar{w}_m^k q_1^k, ..., b_{mN\times M}^k q_{mN\times M}^k)^T.$$

Вводим векторы оцениваемых параметров $b^k = (b_{10}^k, b_{11}^k, ..., b_{1N\times M}^k, b_{20}^k, b_{21}^k ..., b_{2N\times M}^k, ..., b_{m0}^k, b_{m1}^k, ..., b_{mN\times M}^k)$ размерности $(\mathrm{N}\times M+1)m$.

Тогда обобщенную модель можно записать в векторной форме

$$\tilde{q}^k = b^{kT} r^k(q^k)$$

или в матричной форме

$$\tilde{Q} = BR.$$

Тогда задача в матричной форме ставится следующим образом: найти вектор B, чтобы [40, 41]:

$$E = (Q - \tilde{Q})^T (Q - \tilde{Q}) \to \min,$$

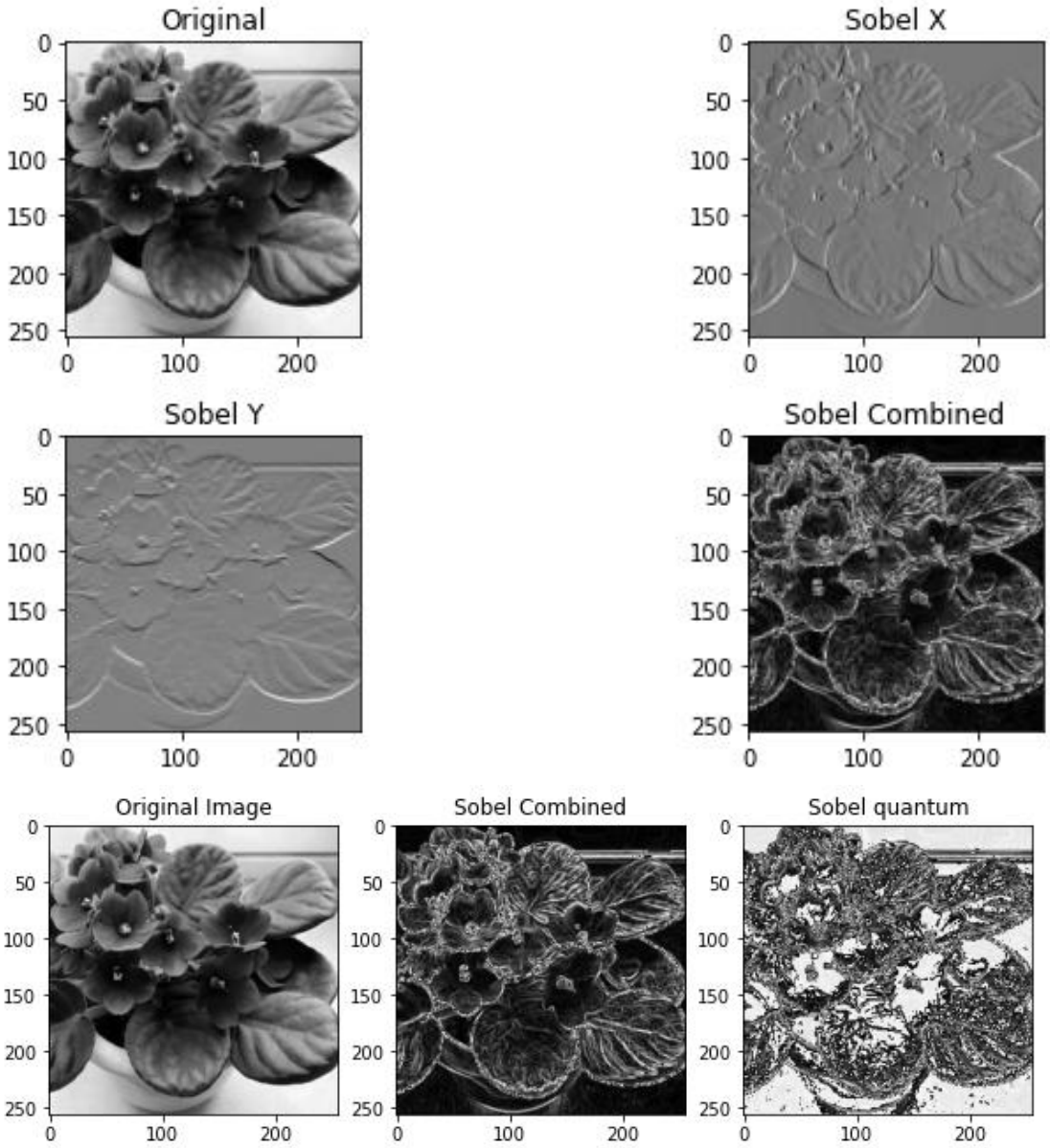
где $\tilde{Q} = BR$.

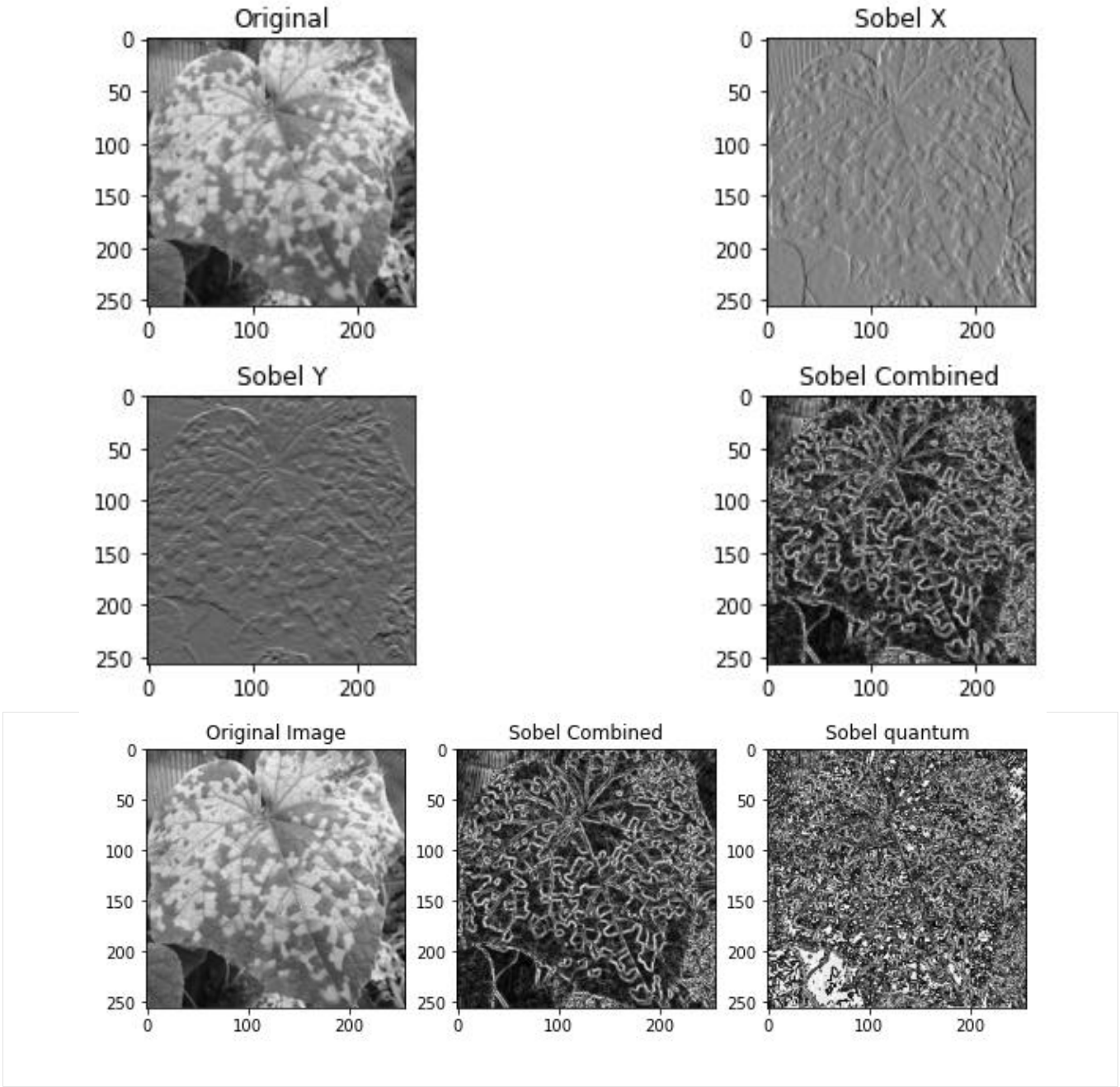
$$B = \left(R^T R\right)^{-1} R^T Q. \quad (3.5)$$

Проблемы нахождения решения (3.5) связаны с возможной сингулярностью матрицы $\left(R^T R\right)$.

Далее производится вычислительный эксперимент.

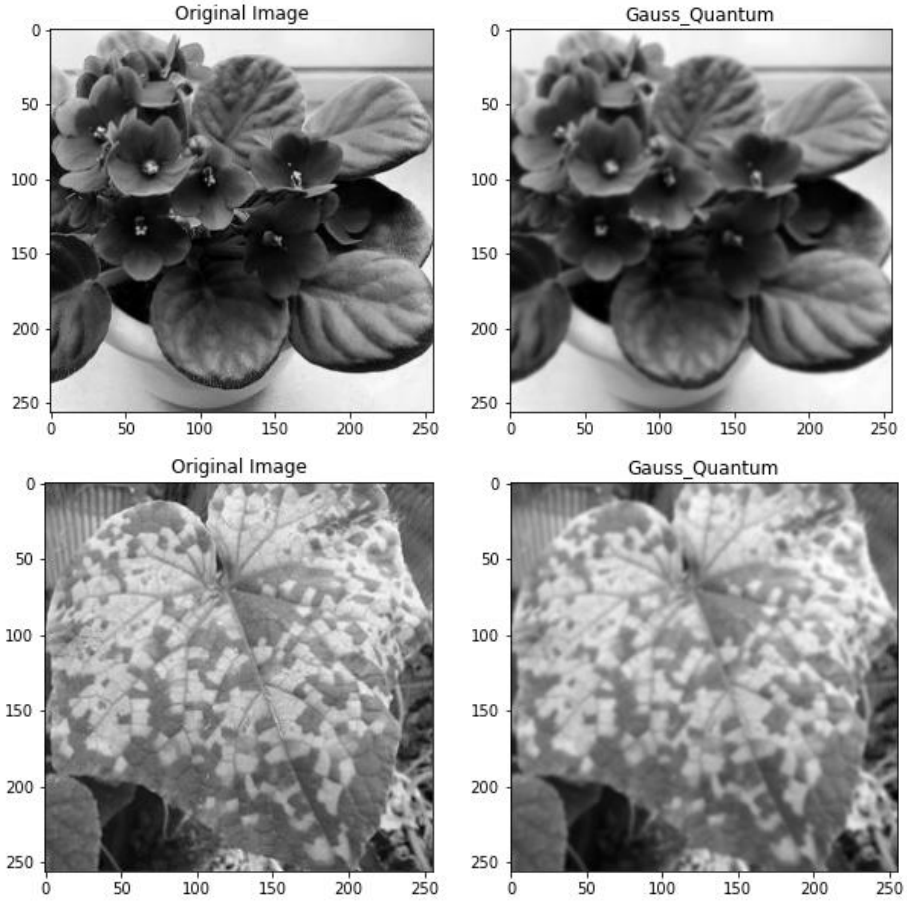
Целью вычислительного эксперимента является проверка, как квантовые алгоритмы могут использоваться для обработки изображений, в частности, для усиления границ и улучшения контрастности. В этом контексте программа применяет классический оператор Собеля для выделения границ объектов и квантовое преобразование Хаара для улучшения изображения. Программа выводит оригинальное изображение, результат применения оператора Собеля и улучшенное изображение после квантовой обработки.


Original
Sobel X
Sobel Y
Sobel Combined
Original Image
Sobel Combined
Sobel quantum

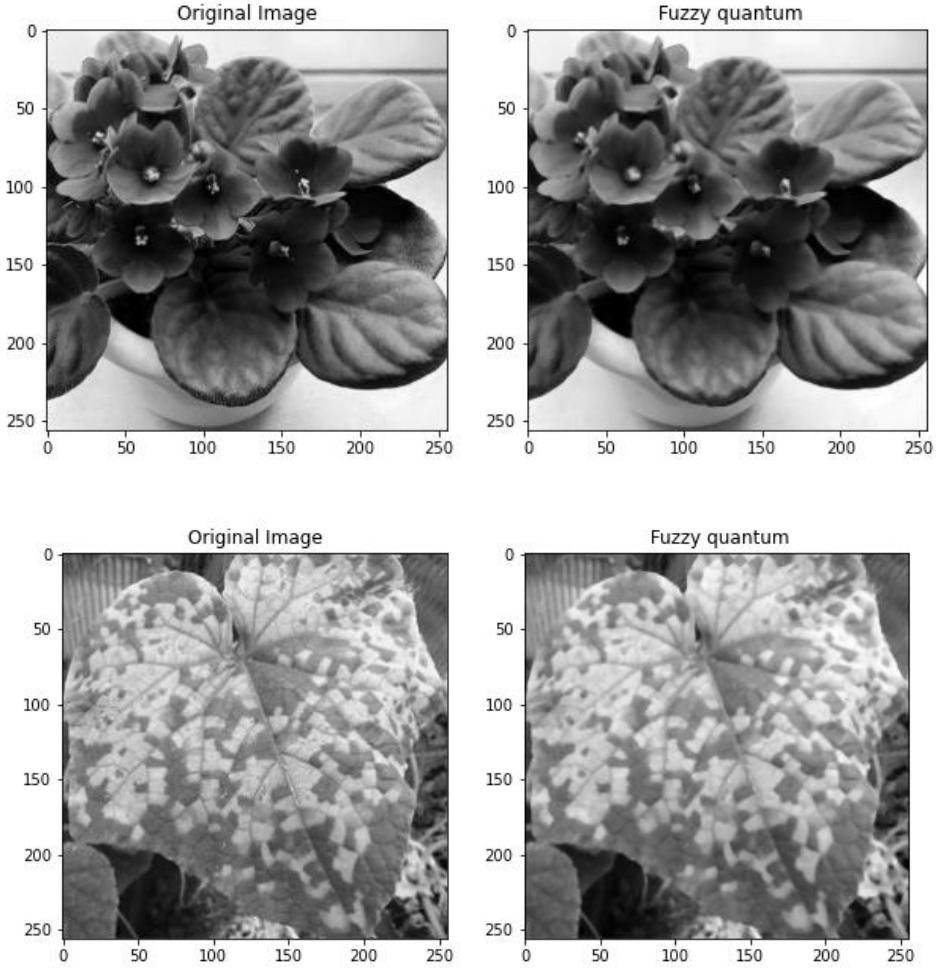


Программа квантового фильтра Гаусса к изображению начинается с загрузки изображения и преобразования его в оттенки серого. Затем применяется классический фильтр Гаусса, чтобы получить сглаженное изображение. Далее каждый пиксель сглаженного изображения обрабатывается квантовой схемой, в зависимости от его интенсивности. Если интенсивность пикселя превышает определенный порог, в схему добавляется X-вращение. Затем схема измеряется, и результат записывается в преобразованное изображение. Наконец, программа

выводит исходное изображение, преобразованное изображение и квантовую схему, которая использовалась для обработки изображения.

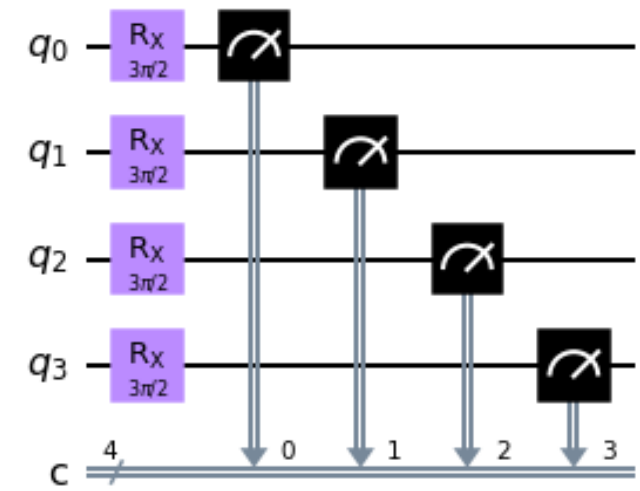


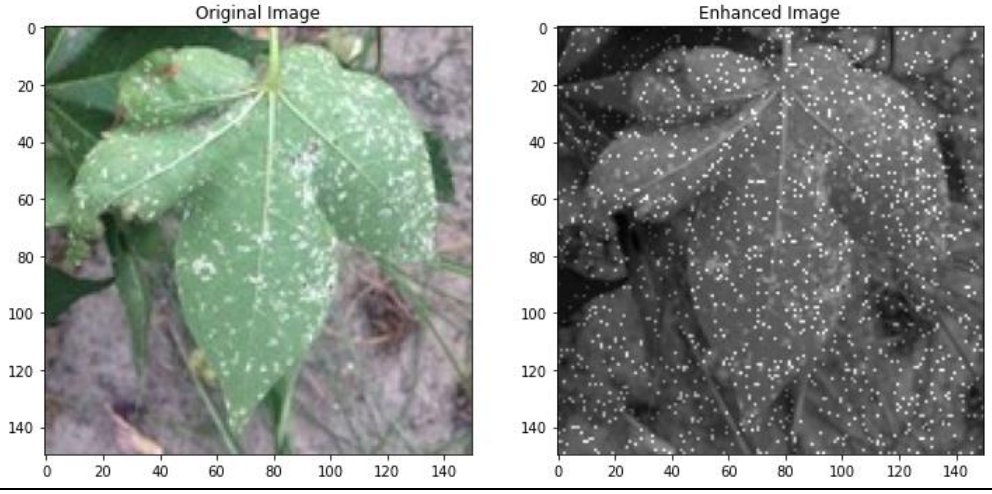
Программа нечеткого фильтра начинается с загрузки изображения и преобразования его в оттенки серого. Затем применяется нечеткий фильтр, чтобы получить сглаженное изображение. Далее каждый пиксель сглаженного изображения обрабатывается квантовой схемой, в зависимости от его интенсивности. Если интенсивность пикселя превышает определенный порог, в схему добавляется X-вращение. Затем схема измеряется, и результат записывается в преобразованное изображение. Программа выводит исходное изображение, преобразованное изображение и квантовую схему, которая использовалась для обработки изображения.

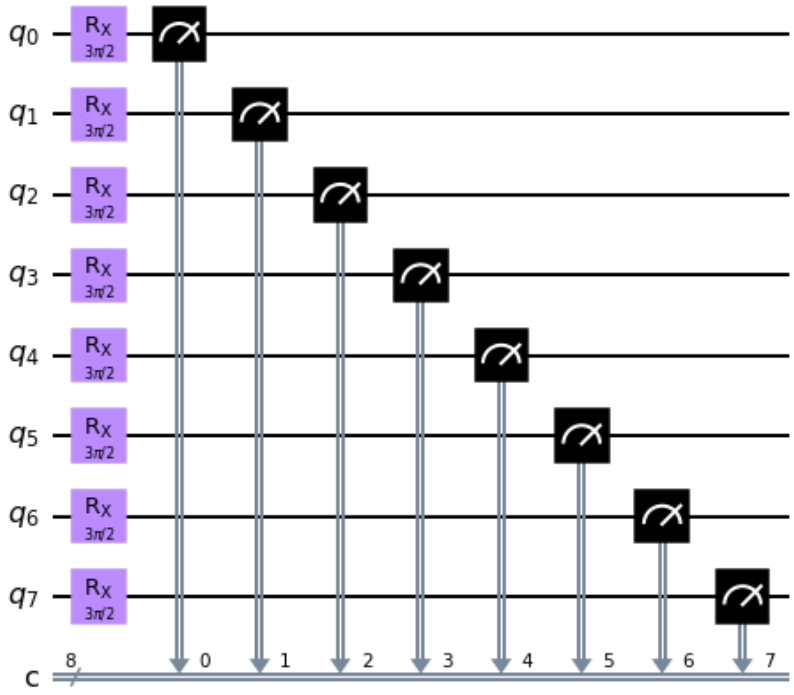


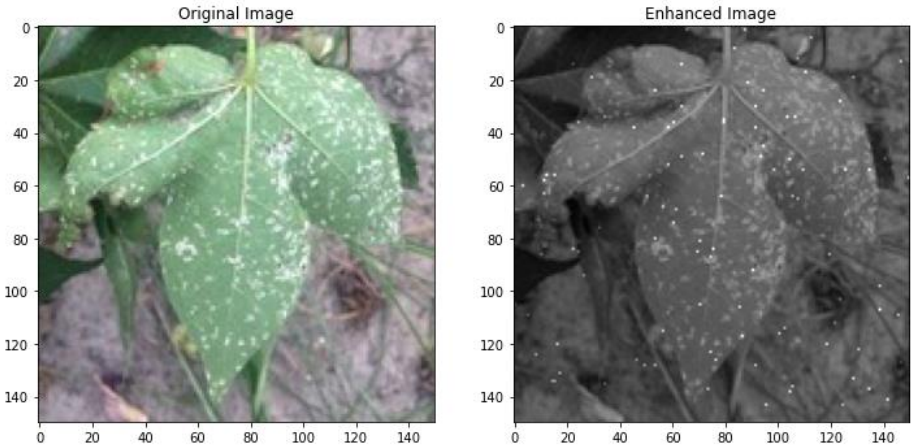
При улучшении контраста каждый пиксель изображения обрабатывается квантовой схемой. Для каждого пикселя создается квантовая схема с 2, 4 и 8 кубитами. Применяется операция поворота R_x с увеличивающим коэффициентом контрастности. Затем схема измеряется, и результат записывается в преобразованное изображение

Программа выводит исходное изображение и улучшенное. Кроме того, выводится квантовая схема, используемая для обработки изображения.


q_0
q_1
q_2
q_3
Rx
3π/2
c
4
0
1
2
3

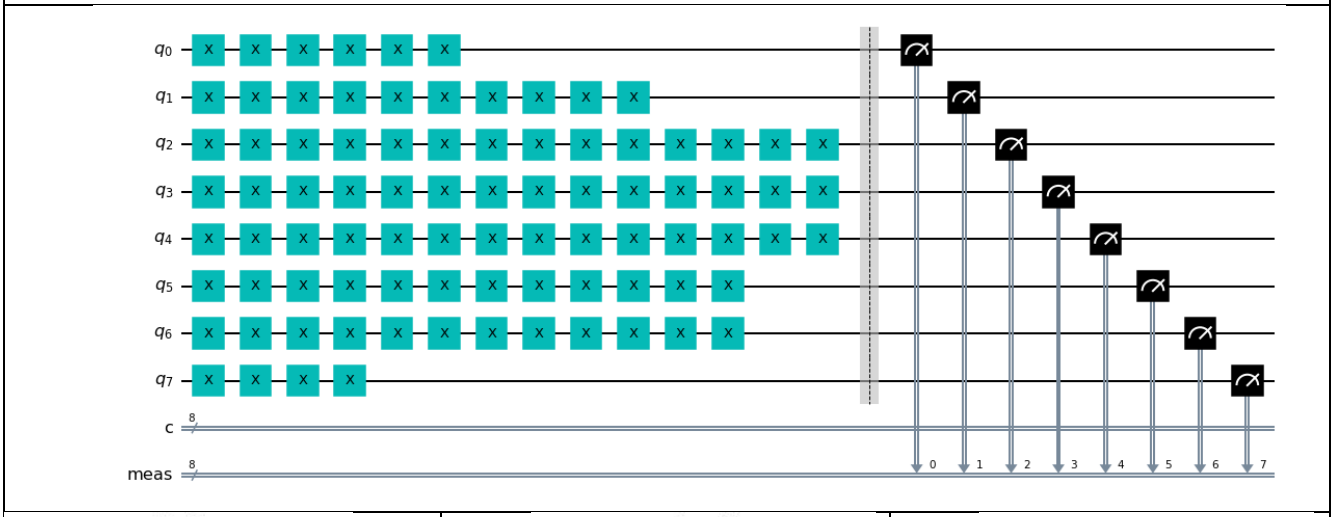

Original Image
Enhanced Image
0
20
40
60
80
100
120
140

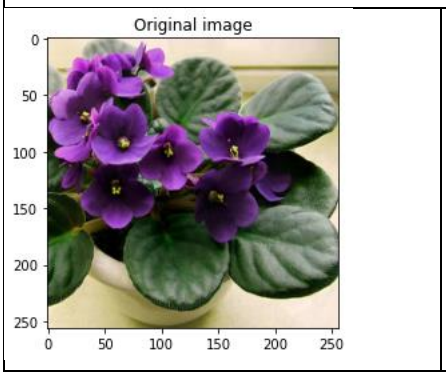

q_0
q_1
q_2
q_3
q_4
q_5
q_6
q_7
Rx
3π/2
c
8
0
1
2
3
4
5
6
7

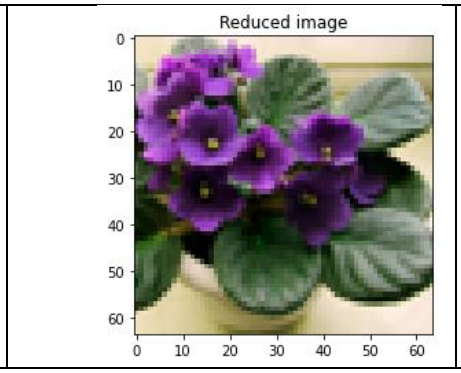

Original Image
Enhanced Image
0
20
40
60
80
100
120
140

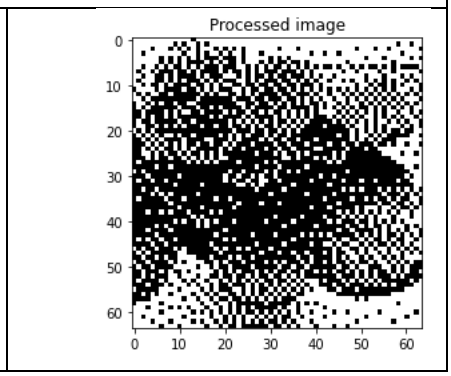
Использования квантовых вычислений для обработки изображений путем изменения размера их до 64x64 пикселей позволяет продемонстрировать, как квантовые методы могут применяться к процессу сжатия изображений и как это может влиять на результаты обработки и качество изображения.

Квантовая схема

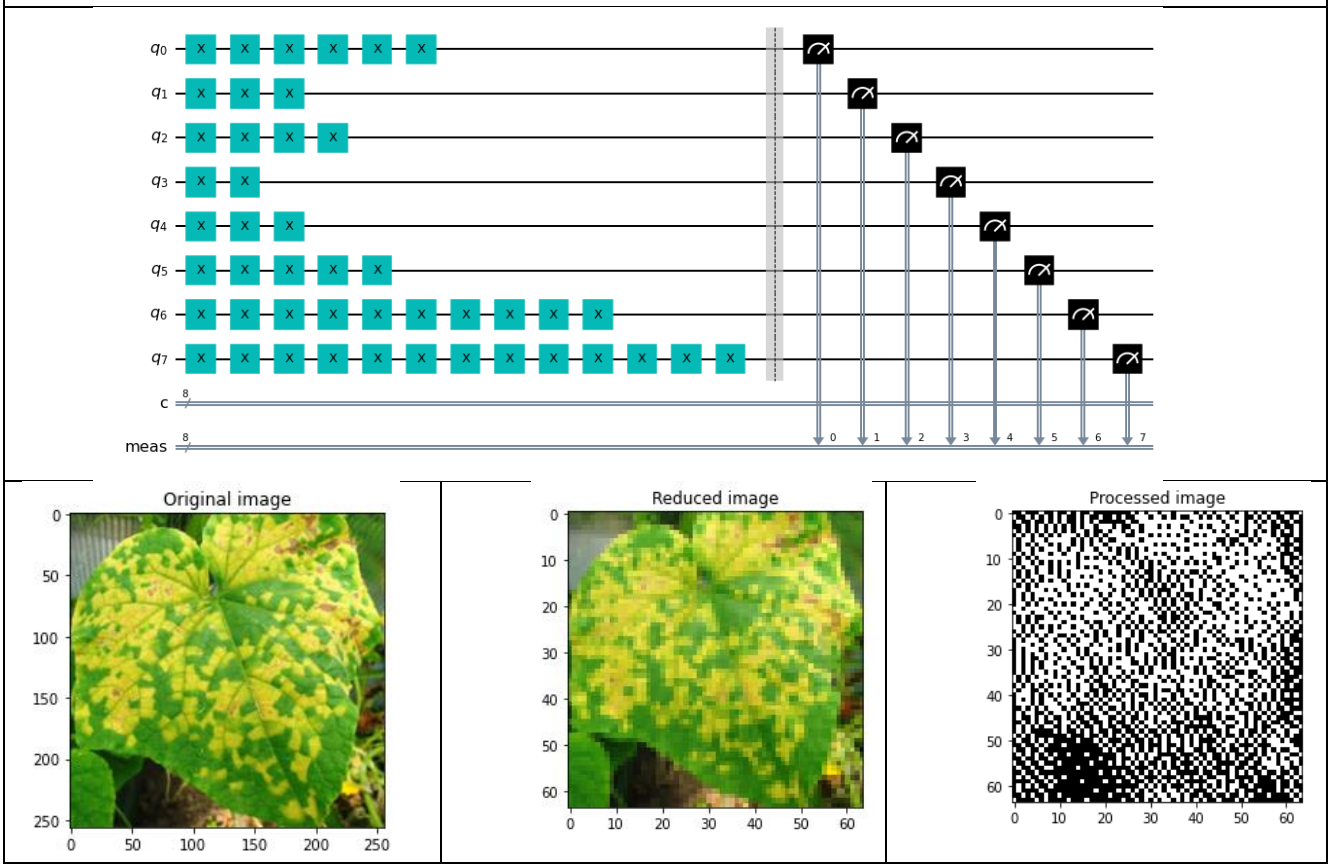






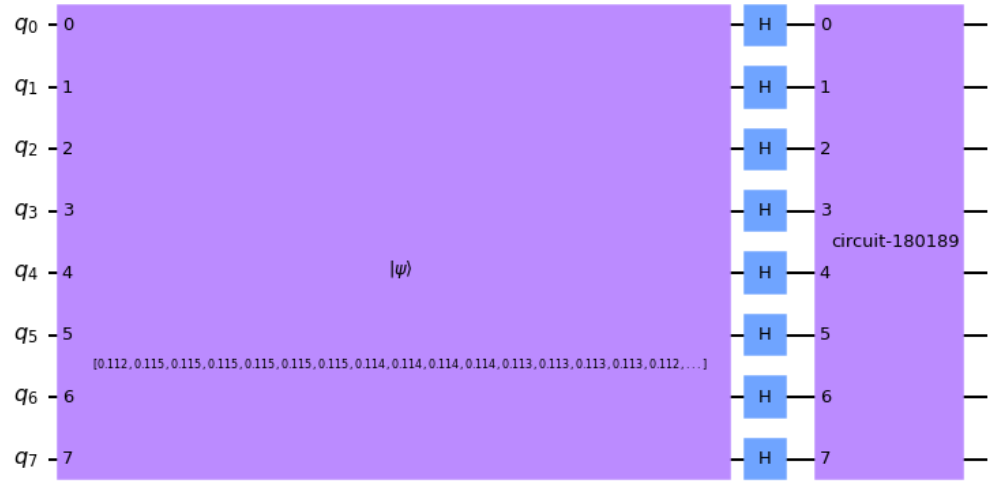


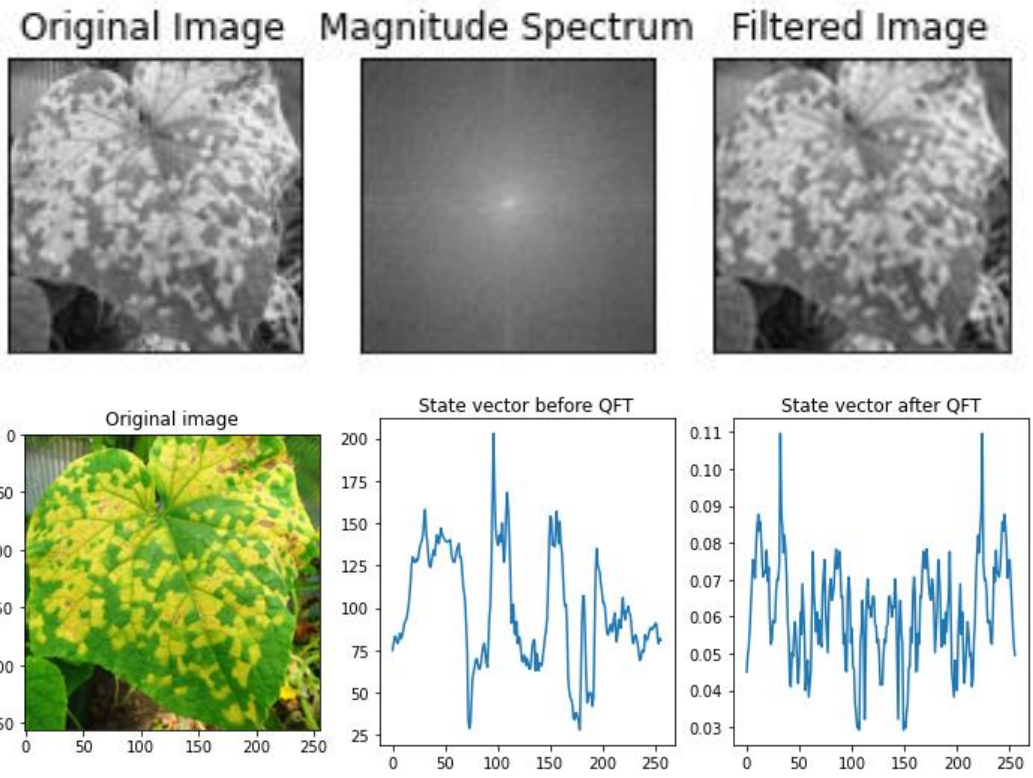
Квантовая схема

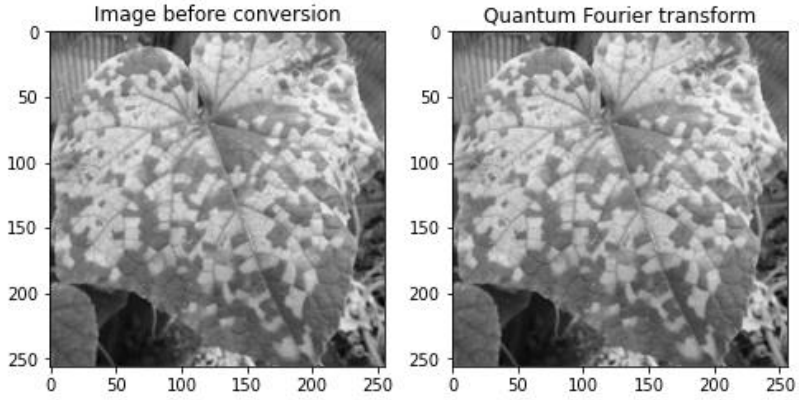


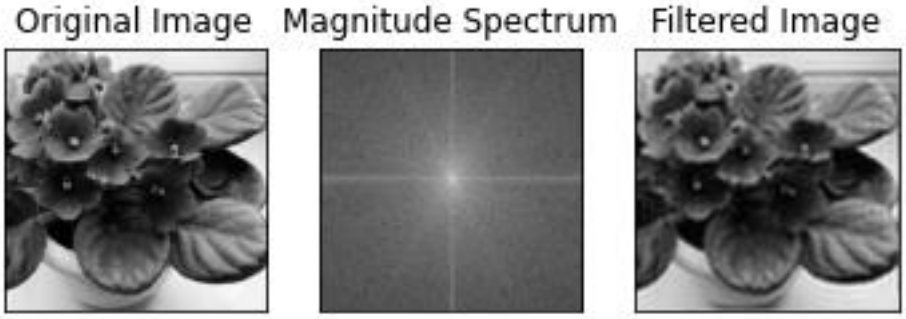
Разработанная программа демонстрирует применение квантового преобразования Фурье к изображению. Сначала загружается изображение в оттенках серого. Затем определяется количество бит, необходимых для представления каждого пикселя, чтобы установить размер квантовой схемы. Далее создается квантовая схема с уровнями Гейтов Адамара и управляемыми фазовыми вращениями, которые реализуют квантовое преобразование Фурье. После этого происходит симуляция квантовой схемы. Результаты симуляции, т.е. вероятности различных состояний, сохраняются в переменной counts.

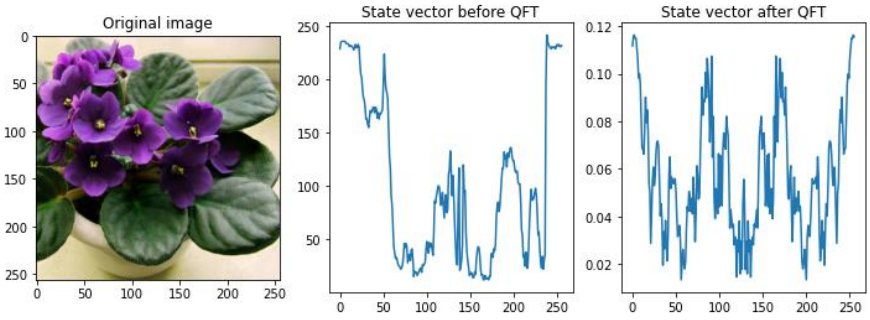
Выводятся изображения до преобразования и после него. Квантовая схема также отображается с помощью circuit_drawer.

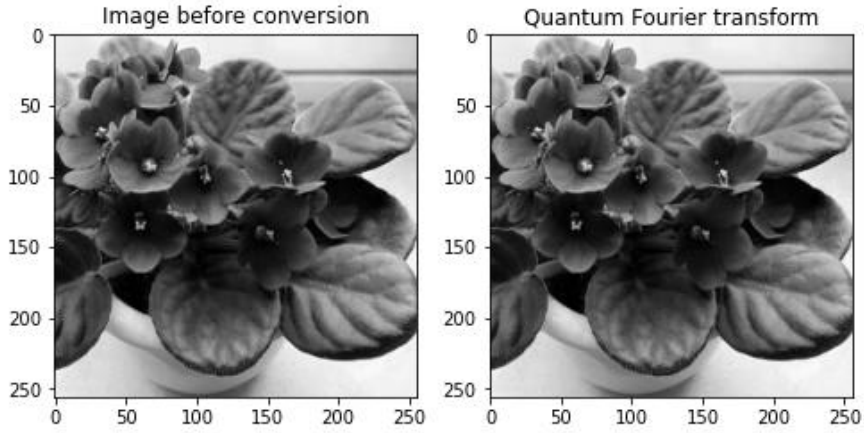

q_0 0
q_1 1
q_2 2
q_3 3
q_4 4
q_5 5
q_6 6
q_7 7
$|\psi\rangle$
[0.112, 0.115, 0.115, 0.115, 0.115, 0.115, 0.115, 0.114, 0.114, 0.114, 0.114, 0.113, 0.113, 0.113, 0.113, 0.112, ...]
H
circuit-180189


Original Image
Magnitude Spectrum
Filtered Image
Original image
State vector before QFT
State vector after QFT









Разработана программа для предварительной обработки изображений на основе квантовых вычислений. Предложен общий псевдокод для этой задачи:

1. Загрузка изображения из исходного файла.

2. Предварительная обработка изображения:

а. Преобразование изображения в формат, пригодный для квантовых вычислений.

Преобразование изображения в формат, пригодный для квантовых вычислений: В этой программе мы просто преобразуем изображение в черно-белый формат. Это пример преобразования, которое делает изображение более подходящим для дальнейших операций

Применение квантового преобразования Хаара к черно-белому изображению.

После применения квантового преобразования мы обратно преобразуем полученные коэффициенты в изображение с помощью обратного преобразования

в. Преобразование обработанного изображения обратно в классический формат. Здесь мы просто конвертируем изображение обратно в формат RGB

3. Сохранение обработанного изображения в новый файл.

разберем эту программу по шагам:

Мы указываем путь к целевому файлу изображения (output_image_path), в который мы сохраним обработанное изображение.

Затем мы используем метод save() для сохранения обработанного изображения в указанный файл.

Вывод информации об успешном завершении:

По завершении программы выводится сообщение о том, что изображение успешно обработано и сохранено.

После предварительной обработки изображений на основе аппарата нечетких множеств проводится распознавание болезней растений с использованием нейронных сетей.

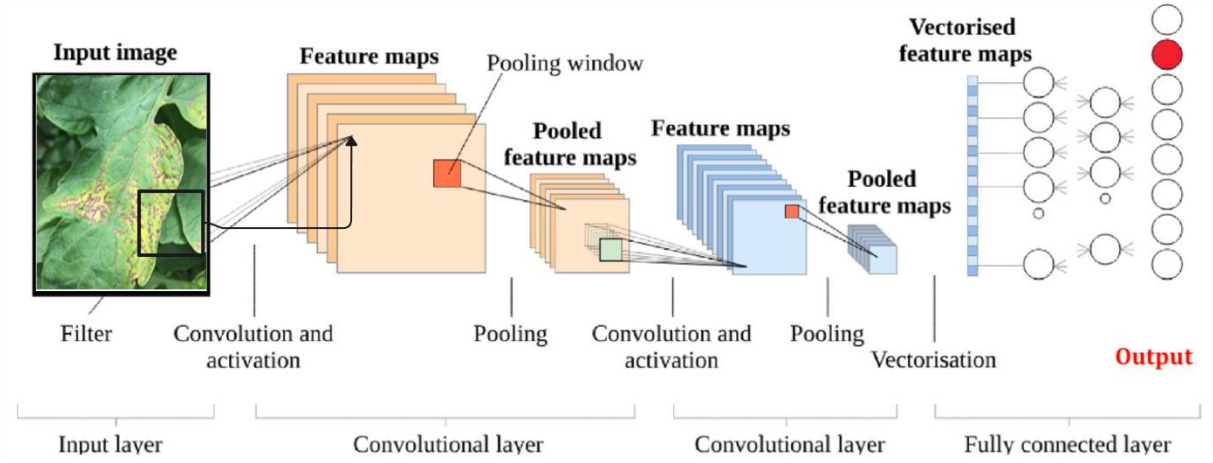
Глубокие сверточные нейронные сети (CNN) привнесли значительный прогресс в области визуального распознавания и классификации изображений, включая диагностику заболеваний растений на основе изображений их листьев [42-43].

CNN - это класс нейронных сетей, специально разработанный для анализа визуальных данных, таких как изображения. Они обладают способностью автоматически извлекать признаки из изображений на

различных уровнях абстракции, начиная с простых локальных особенностей и заканчивая более сложными концепциями. Это позволяет им моделировать иерархию признаков и эффективно обрабатывать сложные данные, такие как изображения листьев растений [44-47].

Во время обучения CNN находит оптимальные веса и связи между нейронами, чтобы оптимально классифицировать изображения на основе признаков, которые она сама извлекает из данных[48-50].

Схема 1. Работа глубоких CNN для диагностики заболевание сельскохозяйственных культур



Даем описание работы работа глубоких CNN для диагностики заболевание сельскохозяйственных культур.

1. Входной слой:

- Принимает цветные изображения листьев растений размером 256x256 пикселей и тремя цветовыми каналами (RGB).

2. Сверточные блоки:

- Начальный сверточный блок с одним сверточным слоем.

- Последующий сверточный блок с объединением для уменьшения размерности.

3. Остаточные блоки:

- Два остаточных блока с двумя сверточными слоями каждый, использующие пакетную нормализацию и активацию ReLU.

- Остаточные блоки помогают обучать более глубокие сети, смягчая проблему исчезновения градиента.

4. Классификатор:

- Слой глобального максимального объединения для уменьшения пространственных размеров.

— Сведение слоя для преобразования 2D-карт объектов в 1D-тензор.

- Полностью связный слой, создающий выходные логиты для классификации заболеваний.

- Количество выходных нейронов соответствует общему количеству классов заболеваний.

5. Функции активации:

- Активация ReLU применяется после каждого сверточного слоя.

6. Функция потерь:

- Перекрестная энтропийная потеря, используемая для многоклассовой классификации.

7. Оптимизация:

- Оптимизатор Адама для обновления параметров модели.

- Планировщик скорости обучения OneCycle для динамической регулировки скорости обучения.

8. Обучение и оценка:

- Эпохи обучения: модель проходит обучение в течение определенного количества эпох (в данном случае 2 эпохи).

- Размер пакета: обучающие данные обрабатываются пакетами по 32 изображения за итерацию.

- Регулировка скорости обучения: планировщик скорости обучения OneCycle регулирует скорость обучения во время обучения для лучшей сходимости.

- Постепенное отсечение. Градиентное отсечение применяется для предотвращения взрыва градиентов во время обратного распространения ошибки.

- Распад веса: для регуляризации используется регуляризация L2 с затуханием веса (1e-4).

- Метрики обучения: точность и потери отслеживаются в периоды обучения.

- Проверка: эффективность модели оценивается на отдельном наборе данных проверки, чтобы обеспечить обобщение.

- Тестирование: обученная модель тестируется на тестовом наборе данных, и для каждого изображения делаются прогнозы.

- Сохранение модели: обученная модель сохраняется как в виде словаря состояния, так и в виде всей модели для использования в будущем.

Подробности обучения:

- Оптимизатор: Адам
- Планировщик скорости обучения: OneCycle
- Градиентное отсечение: 0,1
- Распад веса (регуляризация): 1e-4.После обучения, глубокая CNN может использоваться для классификации новых изображений листьев и определения их фитосанитарного состояния. Модель автоматически анализирует признаки на изображениях, находит соответствующие паттерны и принимает решения на основе этих паттернов [51-53].

Вычислительный эксперимент. База правил количественной оценки качества изображений приведены в таблице 1. При составлении логических правил учитывалось, что качество изображения ухудшается при большом наличии шумов, плохой яркости и контрастности. При отсутствии этих факторов или их снижении качество изображения улучшается. Модель оценки качества изображения основывается на нечетком логическом выводе Сугено, в котором все значения входных и выходных переменных заданы нечеткими множествами.

Входной сигнал задается в виде матрицы, столбцы которой соответствуют разным классам обрабатываемого изображения.

Рассмотрена задача диагностики заболеваний сельскох

озяйственных культур. Исходной информацией в задаче определения фитосанитарного состояния культурных растений являются изображения их листьев. Использование цифровых фотоизображений, в сочетании с обработкой изображений, распознаванием образов и автоматическими инструментами классификации является мощным подходом для решения проблемы мониторинга и диагностики заболеваний растений в сельском хозяйстве.

Таблица 3.2

База правил количественной оценки

Если «входы»				То
Яркость q_1	Контраст q_2	Шум q_3	Смещение фокуса q_4	Качество изображения r
B	B	B	B	$r=1,6+4,7\,q_1-66,9\,q_2+37,4\,q_3+2,8\,q_4$
B	B	B	G	$r=1,6+30,4\,q_1-71,7\,x_2+52,1\,q_3-0,7\,q_4$
B	B	B	E	$r=2,0+392,5\,q_1-354,5\,q_2+12,8\,q_3-4,7\,q_4$
B	B	G	B	$r=0,3+0,1\,q_1+50,1\,q_2+3,0\,q_3-4,7\,q_4$
B	B	G	G	$r=-3,2-13,5\,q_1+35,4\,q_2+9,9\,q_3-2,0\,q_4$
B	B	G	E	$r=1,2-3,1\,q_1+1,1\,q_3-0,6\,q_4$
B	B	E	B	$r=2,1+28,7\,q_1+56,1\,q_2-0,1\,q_3-8,7\,q_4$
B	B	E	G	$r=-0,7+64,0\,q_1+31,1\,q_2+2,7\,q_3-9,5\,q_4$
B	B	E	E	$r=5,5+6,4\,q_1-10,6\,q_2-3,4\,q_3+0,7\,q_4$
B	G	B	B	$r=2,7-64,4\,q_1-1,9\,q_2-6,6\,q_3+7,9\,q_4$
B	G	B	G	$r=0,8+28,9\,q_1+2,0\,q_2-10,2\,q_3-1,2\,q_4$
B	G	B	E	$r=5,5-3,8\,q_1-7,4\,q_2+2,0\,q_3+1,1\,q_4$
B	G	G	B	$r=-1,5+158,3\,q_1+5,5\,q_2+1,6\,q_3-16,2\,q_4$
B	G	G	G	$r=-2,5-99,7\,q_1+8,8\,q_2+0,2\,q_3+9,4\,q_4$
B	G	G	E	$r=-6,3-7\,q_1-1,2\,q_2+17,9\,q_3-1,6\,q_4$
B	G	E	B	$r=1,6+124,0\,q_1-0,3\,q_2+0,7\,q_3-12,0\,q_4$
B	G	E	G	$r=3,5+31,7\,q_1-1,9\,q_2-0,4\,q_3-2,8\,q_4$

Цифровые фотоизображения, могут быть подвергнуты обработке, например, для выделения характеристик листьев и других признаков,

которые могут свидетельствовать о наличии заболевания. Затем, с использованием методов обработки изображений и распознавания образов, можно автоматически анализировать и классифицировать эти изображения. Это может быть достигнуто с помощью алгоритмов машинного обучения, которые обучаются на размеченных данных, состоящих из изображений с известным фитосанитарным состоянием растений. Таким образом, система может автоматически определять признаки, характерные для заболеваний, и делать предположения о состоянии растений на основе этих признаков.

Набор данных воссоздан с использованием автономного дополнения из исходного набора данных. Этот набор данных состоит примерно из 87 тыс. RGB-изображений здоровых и больных листьев сельскохозяйственных культур, которые подразделяются на 46 различных классов. Общий набор данных разделен на обучающий и проверочный наборы в соотношении 80/20 с сохранением структуры каталогов. Новый каталог, содержащий 33 тестовых изображения, создается позже для целей прогнозирования.

Примеры из набора данных

Хлопчатник:Здоровые

Хлопчатник: Альтернариоз

Хлопчатник:Мучнистая роса

Хлопчатник:Вертициллезное увядание

Пшеница:Здоровые

Пшеница:Черный бактериоз

Пшеница:Бурая ржавчина

Пшеница:Мучнистая роса

Помидоры:Здоровые

Помидоры:Бактериальная пятнистость

Помидоры:Серая гниль

Помидоры:Фитофтороз

Помидоры:Бурая пятнистость

На основании полученных данных была построена графическая зависимость, показывающая среднее значение точности и ошибки распределения исходя из количества эпох обучения (рис. 3.4,3.5).

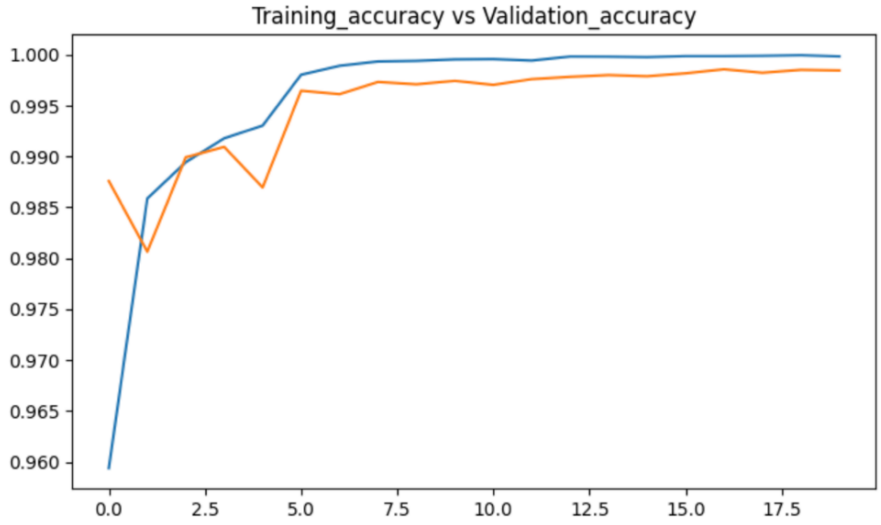


Рис. 3.4. График зависимости точности распределения от количества пройденных эпох обучения

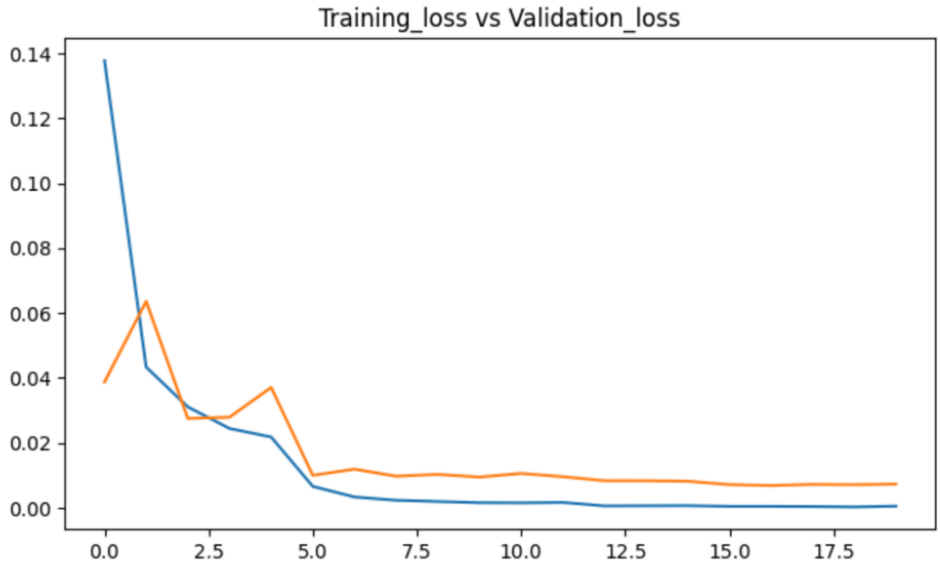
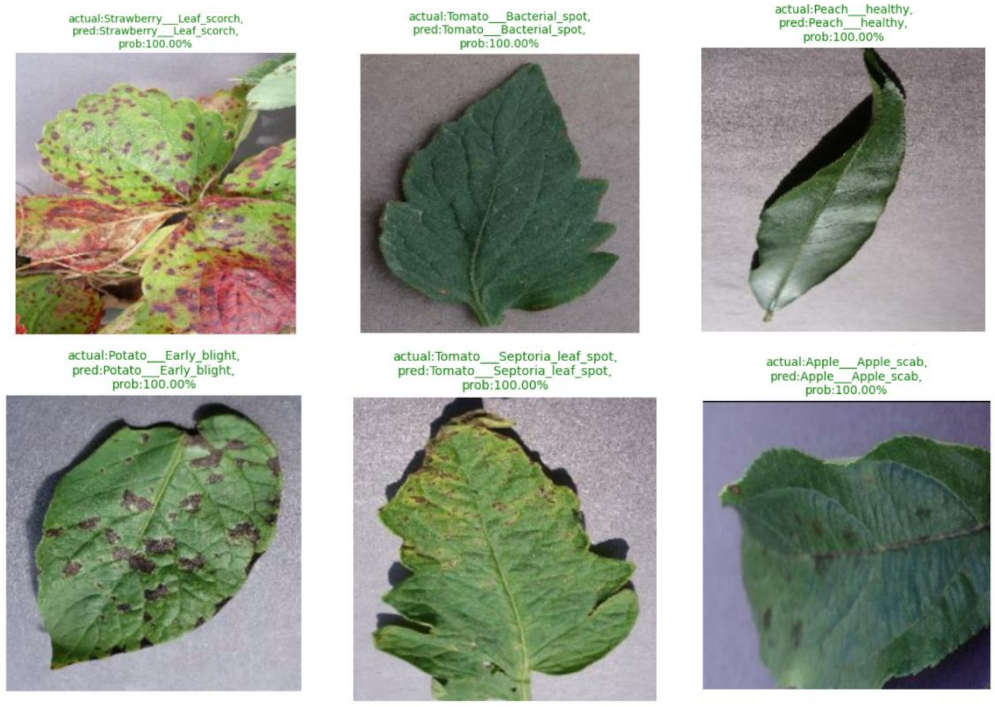
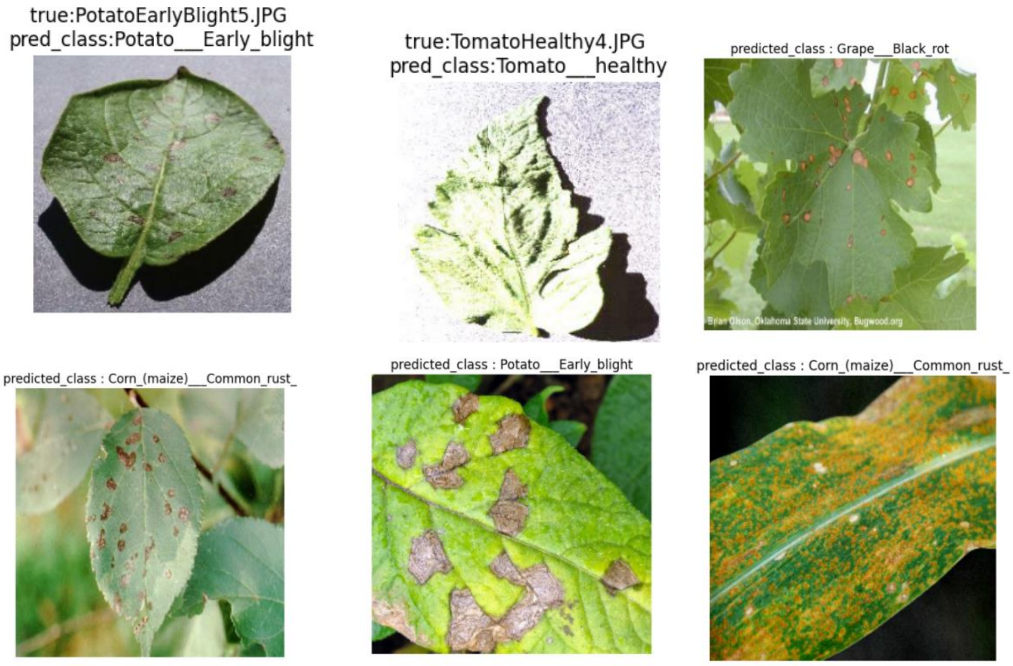


Рис. 3.5. График зависимости ошибки распределения от количества пройденных эпох обучения

Примеры полученных результатов





Исходя из проведенного анализа существующих технологий машинного обучения был разработан программный комплекс на основе CNN, предназначенной для анализа изображений и последующего определения заболевших растений. Традиционная сверточная нейронная сеть или полносвязная нейронная сеть характеризуются потерей информации и другими проблемами при передаче данных, что приведит к исчезновению или всплеску градиента, а последнее, в свою очередь, способно вызывать проблемы при обучении. В 2009 году Грейвс и Шмидхубер представили многонаправленные рекуррентные нейронные сети, способные обрабатывать входные данные в четырех направлениях с использованием рекуррентных слоев. В 2011 году Миколов и коллеги предоставили свободно доступный инструмент с открытым исходным кодом для обучения языковых моделей на основе рекуррентных нейронных сетей. Обучение этих сетей осуществляется с использованием алгоритма обратного распространения ошибки по времени, который считается естественным расширением стандартного алгоритма обратного распространения ошибки, выполняющего неполный градиентный спуск. Этот метод обеспечивает эффективное обучение моделей с учетом временных зависимостей.

Ранее, в 1994, Робинсон использовал простые рекуррентные нейронные сети для распознавания речи, а в 1995 и 1998 годах Ли, Ким и

Сеньор с Робинсоном применили подобные сети для распознавания речи и рукописного текста соответственно. Общая ошибка в процессе обучения должна быть обобщена по каждому образцу последовательности, аналогично многослойному перцептрону. Однако при обучении рекуррентных нейронных сетей необходимо учитывать общий временной контекст.

Название алгоритма обучения - обратное по времени распределение, происходит из того, что при обучении некоторых искусственных нейронных сетей, использующих методы, основанные на градиенте (например, обратное распространение ошибки), возникает сложная проблема исчезающего градиента. Эта проблема связана с перекрытием многих слоев и использованием функций активации (например, сигмоидной или тангенсоидной), которые нелинейно сжимают входные данные в очень маленьком выходном диапазоне. При наложении нескольких слоев даже значительные изменения входных данных приводят к незначительным изменениям выходных данных, что приводит к небольшим градиентам. Это делает невозможным улавливание больших временных зависимостей, что ограничивает применение искусственных нейронных сетей для моделирования временных последовательностей.

В 2006 году Грейвс и соавторы представили критерий подхода ранжирующей классификации (CTC) для обучения рекуррентных нейронных сетей. Этот критерий позволяет сопоставлять входные последовательности с выходными последовательностями символов от предсказаний N к N без необходимости дополнительной постобработки. Однако стандартные целевые функции нейронной сети обычно определяются для каждого кадра в последовательности обучения, что создает сложности при обучении рекуррентных нейронных сетей для генерации последовательности независимых классификаций признаков. Для использования CTC необходимо предварительно сегментировать данные обучения, и результаты сети требуют дополнительной обработки для получения окончательной последовательности признаков. Использование нейронной сети, основанной на архитектуре CNN, в некоторой степени решает проблему исчезающего градиента. Эта сеть обеспечивает защиту целостности данных, прямо проходя входную

информацию и выводя ее. Обучение всей сети требуется лишь для изучения различий между входом и выходом, что значительно упрощает процесс обучения [100].

В контексте обработки изображений, алгоритмы на основе CNN часто улучшают качество изображения путем обработки и шумоподавления. Для решения этой задачи применяются различные нечеткие алгоритмы, которые эффективно удаляют шум из изображений. Исследования, проводимые в развитых странах, сосредоточены на использовании нечетких методов в обработке изображений из-за их способности эффективного управления неопределенностью в условиях:

Нечеткие методы представляют собой мощное средство отражения и обработки знаний;

Они могут эффективно управлять в условиях неопределенности.

Многие программы обработки изображений требуют специальных знаний для преодоления трудностей. Теория нечетких множеств и нечеткая логика могут отражать и обрабатывать человеческие знания, представленные как нечеткие правила ЕСЛИ. Важно отметить, что многие трудности при обработке изображений связаны с случайностью и неопределенностью данных, используемых в решаемых задачах [92].

Применение аппаратного обеспечения нечеткой логики предоставляет более точное представление нечетких особенностей изображения в оттенках серого, что может быть использовано для повышения четкости изображения. В улучшении качества изображения одним из первых шагов является проблема распознавания. Методы улучшения изображения часто включают удаление шума, сглаживание областей с незначительными изменениями уровней серого и выделение резких изменений уровней серого [93].

Использование нечеткой логики является эффективным подходом для создания систем улучшения изображений, поскольку ее математическая основа позволяет интегрировать эвристические знания о конкретном применении в виде правил. Это привело к разработке различных методов улучшения изображений, использующих нечеткую логику. Ниже рассмотрим краткое описание некоторых из них.

Манкузо М., Полуцци Р., Риццотто Г.А. [90] предложили метод для динамического уменьшения диапазона значений контрастности

изображения и повышения четкости с использованием нечетких правил. Пели Т. предложил алгоритм, основанный на методе Лима [91]. Пэн С. и Лакке предложили линейный нечеткий фильтр для обработки изображений [92]. Известно, что медианные фильтры эффективно удаляют гауссовский шум, а фильтры, основанные на статистике порядка (например, медианный фильтр), успешно применяются для удаления импульсного шума. Для объединения этих фильтров был предложен подход, основанный на нечеткой логике, что привело к неоднозначной морфологической фильтрации. После применения нечеткой морфологической фильтрации точность выявления болезней растений увеличилась с 64,5% до 90,2%, что демонстрирует улучшение точности вычислений в результате использования этого метода.

Квантовая обработка изображений представляет собой захватывающую область в сфере квантовых технологий, обещающую новые возможности в обработке и анализе изображений. Изображения на квантовом компьютере представляются в виде нормализованных состояний, что открывает путь к эффективному решению широкого спектра задач обработки изображений [81-82].

Использование квантовых состояний для представления изображений позволяет эффективно кодировать и обрабатывать информацию о яркости, цвете и пространственной структуре изображения. Квантовые алгоритмы могут обрабатывать большие объемы данных параллельно и эффективно решать задачи, которые были бы сложны для классических компьютеров. Некоторые из потенциальных задач обработки изображений, которые могут быть решены с использованием квантовой обработки, включают в себя улучшение качества изображений, сжатие изображений, анализ изображений для распознавания образов и объектов, а также решение задач в области компьютерного зрения и машинного обучения [83].

Таким образом, квантовая обработка изображений представляет собой многообещающую область, которая может привнести новые методы и подходы в обработку и анализ изображений, открывая новые горизонты для развития технологий в этой области.

Модель решетки кубитов, предложенная Венегасом-Андрака и Бозе в 2003 году, отображает пространственную информацию изображения с

использованием амплитуды одного кубита без активации квантовых свойств. В этой модели количество кубитов соответствует количеству пикселей изображения, что позволяет ей эффективно кодировать и хранить пространственную информацию. Одним из ключевых аспектов этой модели является использование состояний кубитов для представления информации о яркости каждого пикселя изображения. В отличие от классических битов, которые могут быть в состояниях 0 или 1, кубиты могут находиться в суперпозиции состояний, что позволяет им хранить более сложные данные. Таким образом, решетка кубитов позволяет эффективно представлять изображение в формате, который сохраняет пространственную структуру изображения и его яркостные характеристики, используя квантовые принципы, но без активации квантовых свойств, таких как суперпозиция или квантовые взаимодействия.

Гибкое представление квантовых изображений, разработанное Le и его коллегами, представляет собой инновационный подход к кодированию изображений в квантовых состояниях. Этот метод ассоциирует значения яркости каждого пикселя с амплитудой в квантовом состоянии, а также учитывает их пространственное положение на изображении, интегрируя их в квантовое состояние [81].

Гибкое представление квантовых изображений позволяет гибко представлять изображения в квантовом виде, что обеспечивает эффективную обработку и анализ изображений с использованием квантовых алгоритмов. Каждый пиксель изображения кодируется в виде квантового состояния, что позволяет эффективно хранить и обрабатывать большие объемы информации о изображении с использованием квантовых преобразований. Одной из ключевых особенностей гибкого представления квантовых изображений является его способность сохранять пространственную структуру изображения в квантовом состоянии. Это позволяет эффективно решать задачи обработки изображений, такие как улучшение качества изображения, сжатие данных и анализ изображений для распознавания объектов [84-86].

В работе Чжана и его коллег [87] описывается новый подход к представлению изображений, который использует базовое состояние

последовательности кубитов для сохранения информации о значениях оттенков серого каждого пикселя. В отличие от гибкое представление квантовых изображений, где информация кодируется в амплитудах вероятности квантовых состояний, предложенный метод Чжана и др. применяет непосредственное использование базовых состояний кубитов для этой цели.

Этот подход позволяет эффективно хранить информацию о значениях яркости пикселей изображения, используя кубиты как носители данных. Вместо кодирования информации в амплитудах вероятности, как это делается в гибкое представление квантовых изображений, значения оттенков серого сохраняются непосредственно в состояниях кубитов. Преимущество этого метода заключается в его простоте и эффективности, поскольку он не требует сложных манипуляций с амплитудами вероятности и обеспечивает прямой доступ к значениям пикселей через состояния кубитов. Это делает его привлекательным инструментом для представления и обработки изображений с использованием квантовых вычислений. Таким образом, предложенное представление изображений Чжана и др. открывает новые возможности для развития методов обработки изображений с применением квантовых технологий, обеспечивая более эффективное и гибкое решение задач обработки и анализа изображений [87-89].

Алгоритм квантового обнаружения краев Адамара, предложенный Яо и соавторами в 2017 году [89], представляет собой метод анализа квантовых изображений, основанный на применении преобразования Адамара для выявления границ между пикселями. Суть алгоритма заключается в том, что значения пикселей изображения кодируются в амплитудах вероятности квантовых состояний, а их положения – в состояниях вычислительного базиса. Преобразование Адамара применяется к этим квантовым состояниям для вычисления разницы между соседними пикселями, что позволяет обнаружить границы и переходы между областями различной яркости или цвета. Этот метод обладает рядом преимуществ, так как использует преимущества квантовых вычислений для обработки изображений. В частности, он может быть более эффективным по сравнению с классическими методами обработки изображений, особенно при работе с большими

объемами данных или при выполнении сложных операций обработки. Таким образом, алгоритм представляет собой перспективный подход к обработке квантовых изображений, который может быть использован в различных приложениях, включая распознавание образов, медицинскую диагностику, анализ изображений и многое другое.

Анализ литературных источников по применению нечетких методов в задачах обработки цифровых изображений показывает, что эти методы используются в основном в следующих задачах: повышение контрастов изображений, выделение контуров и сегментация. Однако, эти методы не нашли широкого распространения при решении прикладных задач обработки изображений из-за их вычислительной сложности. Учитывая это, в работе определена цель исследования и сформулированы задачи обработки изображений с использованием аппарата нечетких множеств. Определены состав и структура системы обработки изображений, основанной на теории нечетких множеств. Разработана структурная схема основных функциональных подсистем с учетом применения «мягких вычислений». Предложена схема предварительной обработки и оценки качества изображений с использованием аппарата нечетких множеств. Проработаны вопросы формирования баз правил на модельном примере. Найдены параметры нечеткой модели с учетом сингулярности матрицы оценки. Построена нейро-нечеткая сеть количественной оценки качества изображения. Улучшены нечеткие операции с изображениями. Предложенные нечеткие операции могут быть использованы для разработки новых методов предварительной обработки и анализа изображений. Создана база данных по болезням растений и разработаны программы выявления болезней по изображению листа растения. Программа позволяет проверить адекватность предложенных алгоритмов

Квантовая обработка изображений представляет собой перспективную область исследований с большим потенциалом для применения в различных областях, включая обнаружение краев и классификацию изображений. В сравнении с классическими методами обработки, квантовые методы обладают преимуществами во времени выполнения и пространственной сложности, что делает их привлекательными для решения некоторых задач обработки

изображений. Однако существуют ограничения, связанные с обработкой изображений большего размера при использовании квантовых методов. Усложнение проектирования схем с увеличением количества требуемых кубитов может привести к увеличению шума и неточностей в результатах обработки. Это ограничивает применение квантовых методов для обработки изображений большего разрешения или содержащих большое количество деталей. Для преодоления этих ограничений необходимо проведение дальнейших исследований по разработке более эффективных алгоритмов и архитектур квантовых схем, способных обрабатывать изображения больших размеров с минимальным уровнем шума. Также важно улучшение технологий квантовых вычислений и развитие методов коррекции ошибок для повышения точности результатов обработки изображений. Несмотря на текущие ограничения, перспективы квантовой обработки изображений остаются многообещающими, и дальнейшие исследования в этой области могут привести к разработке более эффективных и мощных методов обработки изображений с использованием квантовых вычислений.

Путем компьютерного эксперимента показано, что применение предложенной нейро-нечеткой сети существенно снижает затраты на обучение. Для нейронной сети число итераций обучения составляет 8400, а время 7 мин; для нейро-нечеткой сети число итераций составляет 1150, а время 1,2 мин. При осмотре поля мгновенная идентификация избавляет от необходимости тратить время на выявление проблем и регистрацию результатов. Кроме того, записанные изображения могут быть просмотрены другими и использованы в качестве эталона в будущем. Садоводы, которым трудно поставить точный диагноз, используют инструмент в качестве помощника или как способ получить второе мнение, когда есть сомнения. Это особенно важно, когда другие источники рекомендаций недоступны, например, в странах, где соотношение количества производителей к количеству агрономов очень велико. Визуальный осмотр является важным аспектом, когда речь идет о здоровье растений. Поскольку лабораторные тесты не являются практическим инструментом для ежедневной диагностики, из-за затрат и времени оборота производители принимают решения, основываясь на симптомах, которые можно увидеть на растениях. Осмотр растений

занимает много времени и часто оставляет у производителей сомнения. Консультанты по растениеводству готовы поддержать производителей в принятии решений и убедиться, что ошибки сведены к минимуму. Но что делать, если такой помощи нет? К счастью, на помощь могут прийти технологии. Приложение на основе искусственного интеллекта для выявления болезней растений стало возможным благодаря большому скачку производительности, достигнутому сообществом исследователей искусственного интеллекта. Исходя из вышеизложенного можно сделать вывод, что используемая сверточная нейронная сеть CNN на высоком уровне справилась с экспериментальной задачей. В свою очередь, для дальнейшего обучения нейронной сети необходимо создать обширную информационную базу по заболеваниям растений.

3.3. Применение квантовой технологии Variational Quantum Classifier в сельском хозяйстве для классификации сортов пшеницы

В данном исследовании предлагается использование Variational Quantum Classifier для автоматизированной классификации сортов пшеницы. Модель, обученная на обширном наборе данных, будет способна выявлять уникальные паттерны и зависимости между характеристиками семян и принадлежностью к конкретному сорту. Это позволит сельскохозяйственникам и исследователям более точно определять сорта пшеницы, что, в свою очередь, может улучшить процессы выращивания и управления урожаем. Такой подход обосновывается не только потребностью в оптимизации сельскохозяйственного производства, но и в контексте использования передовых технологий для достижения точности и эффективности в аграрной сфере. В результате этого исследования ожидается повышение качества и устойчивости производства пшеницы, что имеет важное значение для обеспечения продовольственной безопасности и устойчивого развития сельского хозяйства. Цель задачи является классификация сортов пшеницы на основе характеристик семян. VQC обучается на обучающем наборе данных, а затем оценивается на тестовом наборе данных. Для оценки производительности модели используются различные метрики, такие как точность (accuracy), precision, recall, F1-score и матрица ошибок (Confusion Matrix).

Сельское хозяйство является одной из ключевых отраслей мировой экономики, обеспечивая продовольственную безопасность и экономическое развитие. Одним из важных аспектов в этой области является выращивание сельскохозяйственных культур, таких как пшеница, которая является основным источником пищи для миллионов людей по всему миру. Сорта пшеницы различаются по своим характеристикам, таким как размер, форма и другие атрибуты семян. Понимание и классификация этих сортов становятся важными задачами для оптимизации процессов сельскохозяйственного производства. В сельском хозяйстве важно иметь надежные методы для идентификации сортов сельскохозяйственных культур. Классификация сортов пшеницы может быть полезной для селекционеров и фермеров при выборе оптимальных семян для посева. Различные сорта пшеницы могут обладать разными характеристиками, влияющими на урожайность и качество урожая. Путем классификации семян по их характеристикам можно определить оптимальные условия для роста и получения максимального урожая. Знание сорта пшеницы может иметь экономическое значение для аграрных предприятий. Разные сорта могут иметь различные рыночные цены, влияя на доходы фермеров. Классификация сортов пшеницы может быть важной частью научных исследований в области сельского хозяйства, биологии и генетики. Это может способствовать лучшему пониманию разнообразия сортов и их адаптации к различным условиям. С учетом современных тенденций в сельском хозяйстве, где внедряются технологии машинного обучения и искусственного интеллекта, задачи классификации становятся более актуальными для автоматизации процессов и улучшения эффективности. Таким образом, задача классификации сортов пшеницы на основе характеристик семян имеет широкий спектр применения и может принести практическую пользу в сельском хозяйстве, научных исследованиях и принятии решений в области сельского хозяйства.

В этом контексте применение методов машинного обучения, в частности, VQC (Variational Quantum Classifier) предоставляет уникальные возможности для эффективной классификации сортов пшеницы на основе их характеристик семян. VQC (Variational Quantum Classifier) - это квантовый классификатор, предназначенный для решения

задач машинного обучения. Он основан на идее вариационного квантового подхода, который использует параметризованный квантовый схематичный анзац для представления входных данных и параметризованный классический алгоритм для обучения этих параметров [111-113].

Вариационные квантовые алгоритмы, включая VQC, представляют собой гибридные модели, которые объединяют классический и квантовый вычислительный ресурс. Они используют квантовую схему для представления информации и классическую часть для обучения параметров этой квантовой схемы. Это позволяет использовать квантовый аппарат для решения определенных задач в рамках применения машинного обучения. Квантовая схема, представляющая входные данные в квантовом пространстве. Входные данные сначала отображаются в квантовые состояния, которые затем подвергаются эволюции на квантовом компьютере. Параметризованный квантовый схематичный блок, который представляет параметризованные квантовые вентили анзац обеспечивает гибкость в выборе формы представления квантового состояния. Классический оптимизатор, который обновляет параметры анзаца на основе обратной связи от обучающих данных. Различные оптимизаторы могут использоваться для нахождения оптимальных параметров. После обучения модели VQC используется для классификации новых данных. Входные данные сначала подвергаются тому же процессу карты признаков и эволюции анзаца, а затем производится классификация на основе результатов. Преимущества VQC включают возможность использования квантового вычислительного ускорения для решения определенных задач классификации, особенно там, где квантовые эффекты могут обеспечить выигрыш в производительности по сравнению с классическими методами. VQC в настоящее время применяется преимущественно в контексте гибридных вычислений, используя квантовые и классические ресурсы вместе [114-117].

В VQC, квантовый схематичный блок анзац является параметризованной моделью, аналогичной параметризованным слоям в нейронных сетях. Эти параметры подлежат обучению, чтобы адаптировать модель к конкретной задаче. Как и в нейронных сетях, для

обучения параметров VQC используется обратное распространение ошибки. В процессе обучения классический оптимизатор обновляет параметры анзаца, чтобы минимизировать ошибку между предсказанными и фактическими значениями. Как и в нейронных сетях, VQC использует функцию потерь для измерения разницы между предсказанными и фактическими значениями. Оптимизатор используется для настройки параметров модели с целью минимизации функции потерь. После завершения обучения VQC используется для классификации новых данных. Результаты, полученные в процессе эволюции квантового состояния, используются для определения класса, к которому относится входной образ. В отличие от традиционных нейронных сетей, где данные представлены числовыми векторами, VQC использует квантовую схему для представления данных в квантовом пространстве. Это позволяет использовать квантовые вычислительные преимущества в решении определенных задач классификации. Таким образом, VQC представляет собой гибридную модель, в которой квантовая часть отвечает за представление и эволюцию данных в квантовом пространстве, а классическая часть - за обучение параметров этого представления [118-120].

Датасет "Seeds" представляет собой набор данных, описывающих различные характеристики семян трех различных сортов пшеницы. Этот датасет используется в задачах классификации для идентификации сорта пшеницы на основе их характеристик.

Каждая запись в датасете содержит следующие атрибуты:

Площадь семени

Периметр семени

Длина семени

Ширина семени

Коэффициент компактности семени

Длина желоба семени

Площадь желоба семени

Длина круга семени

Класс (реальное значение)

Классификация направлена на три сорта пшеницы: Kama, Rosa и Canadian. Датасет содержит информацию о 210 семенах, по 70 семян для

каждого из трех сортов пшеницы. Задача состоит в том, чтобы по характеристикам семян предсказать их сорт. Важно провести предобработку данных, такую как стандартизация или нормализация, перед обучением модели машинного обучения на этом датасете. Этот датасет предоставляет возможность решать задачу классификации в контексте машинного обучения и исследования влияния различных характеристик на сорта пшеницы.

V1 V2 V3 V4 V5 V6 V7
15.26, 14.84, 0.871, 5.763, 3.312, 2.221, 5.22,1
14.88, 14.57, 0.8811, 5.554, 3.333, 1.018, 4.956,1
...
11.26, 13.01, 0.8355, 5.186, 3.308, 2.356, 5.096,3

Данные были предварительно обработаны, включая стандартизацию характеристик семян с использованием методов, таких как StandardScaler из библиотеки scikit-learn. Также была проведена кодировка меток классов для использования в VQC. Производится масштабирование признаков с использованием MinMaxScaler из библиотеки sklearn.preprocessing. Строится график в парах для визуализации взаимодействия признаков. Данные разделяются на тренировочный и тестовый наборы.

Строится квантовая схема с использованием различных квантовых фич (feature map) и анзацов (ansatz) (рис.3.6 и рис.3.7). Сначала мы импортируем необходимый класс ZZFeatureMap из библиотеки Qiskit, который предоставляет инструменты для создания квантовых фич-карт. Затем мы определяем количество признаков в наших данных. Это число будет использоваться при создании квантовой фич-карты. Мы создаем объект ZZFeatureMap, который представляет собой конкретный вид квантовой фич-карты. Мы указываем feature_dimension равным числу признаков в данных, и reps (количество повторений) равным 1.

После создания квантовой фич-карты мы можем произвести ее декомпозицию (если необходимо) и визуализировать квантовую схему, которая ей соответствует. Визуализации квантовой фич-карты, которая может использоваться в квантовых алгоритмах машинного обучения для представления классических данных в квантовой форме.

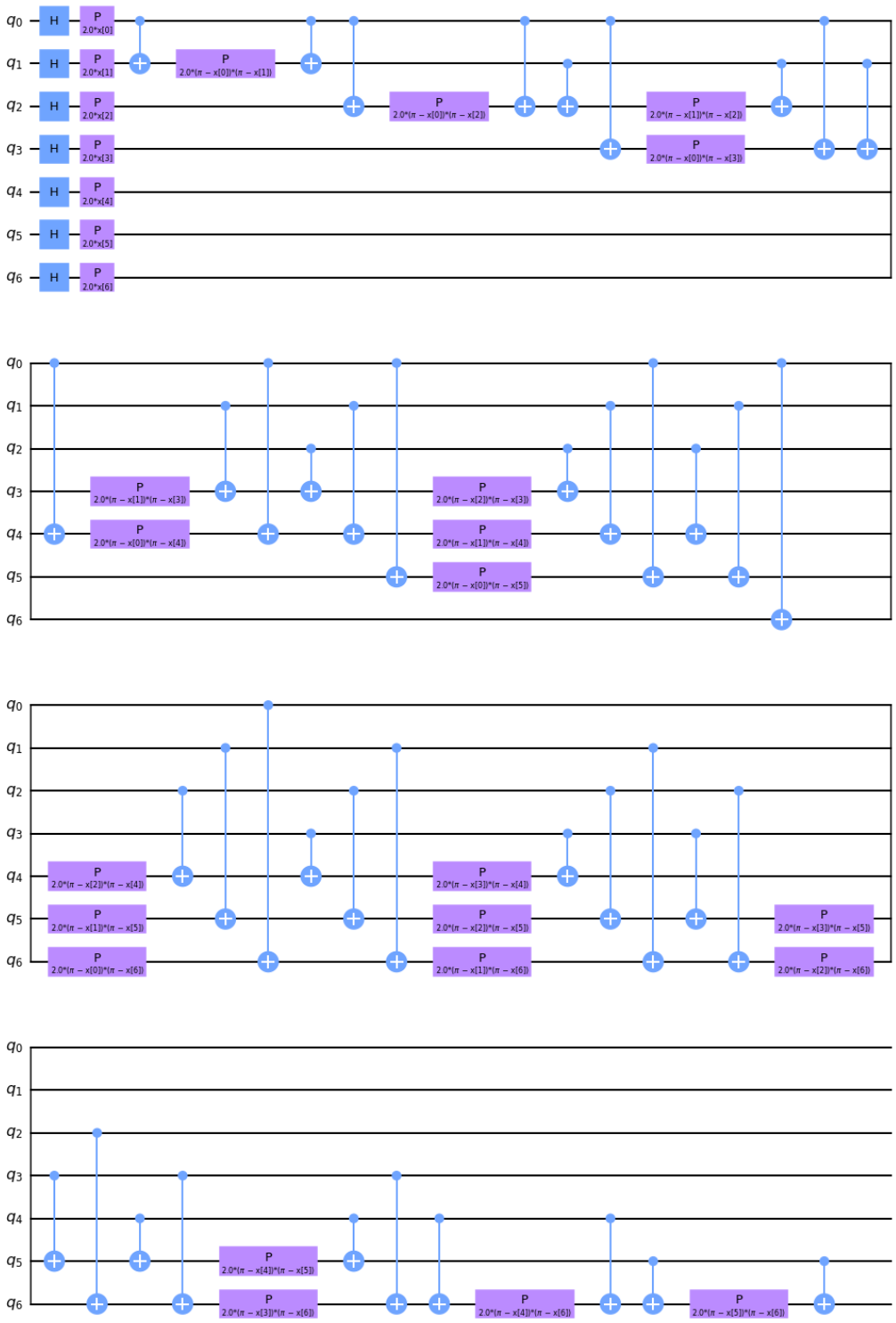


Рис.3.6. Квантовая схема с использованием различных квантовых фич (feature map) и анзацов (ansatz)

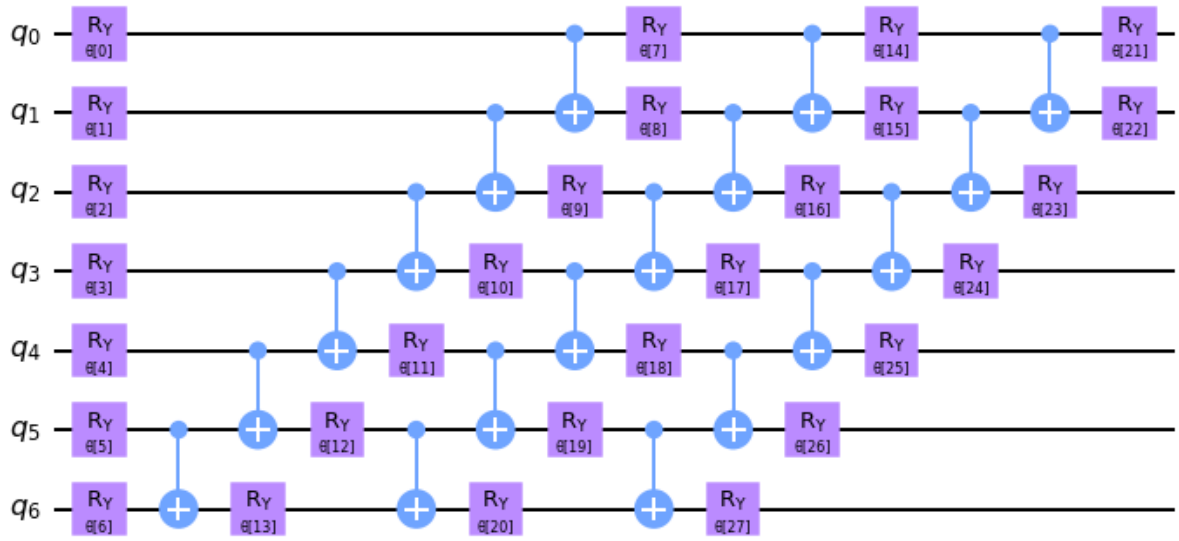


Рис.3.7. Квантовая схема с использованием различных квантовых фич (feature map) и анзацов (ansatz)

Затем производится оптимизация весов модели с использованием квантового классификатора VQC. Визуализируется процесс обучения с использованием функции обратного вызова. Производится оценка точности квантовой модели на тренировочном и тестовом наборах данных. Сравниваются результаты классической и квантовой моделей на различных наборах признаков и различных анзацах. Выводится таблица сравнения результатов.

Разработанная программа представляет собой пример гибридной модели машинного обучения, которая использует как классические, так и квантовые методы для решения задачи классификации. Проведен статистический анализ полученных результатов для оценки степени достоверности классификации и значимости выявленных паттернов.

Этот подход позволяет создать эффективную модель классификации сортов пшеницы на основе их характеристик семян с использованием методов глубокого обучения и нейронных сетей.

В ходе обучения VQC сети на обучающем наборе данных наблюдалось улучшение значений функции потерь с увеличением числа эпох. Это свидетельствует о том, что модель успешно адаптируется к обучающим данным. В представленной программе графики используются для визуализации данных. В данном случае мы создаем

scatter plot для каждой пары признаков из набора данных. Каждая точка на графике представляет собой экземпляр данных с соответствующими значениями двух признаков. Различные цвета точек соответствуют различным классам (видам). Таким образом, графики позволяют нам визуально оценить, как различные виды распределены в пространстве признаков. Красные кресты (x) на графиках представляют собой экземпляры данных, которые были неправильно классифицированы нашей VQC. Таким образом, они представляют ошибки модели (рис.3.8).

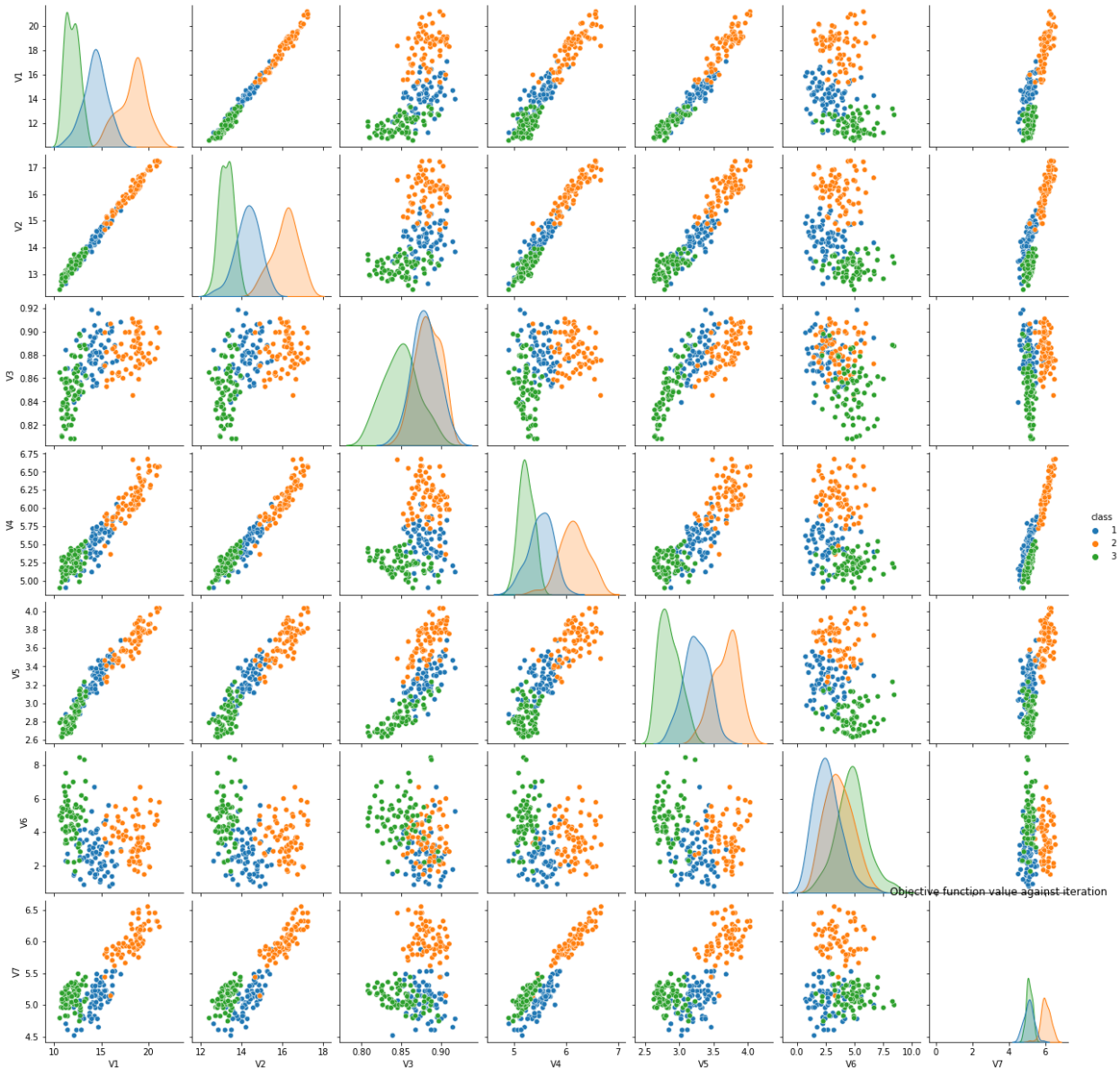


Рис.3.8.Визуальная оценка, как различные виды распределены в пространстве признаков

Эти графики предоставляют визуальное представление о том, как модель классификации работает на тестовом наборе данных. Визуализация ошибок позволяет нам понять, где именно модель допускает ошибки, что может быть полезно для анализа ее производительности и улучшения обучения.

Проведен статистический анализ полученных результатов для оценки степени достоверности классификации и значимости выявленных паттернов.Этот подход позволяет создать эффективную модель классификации сортов пшеницы на основе их характеристик семян с использованием методов глубокого обучения и нейронных сетей (рис.3.9).

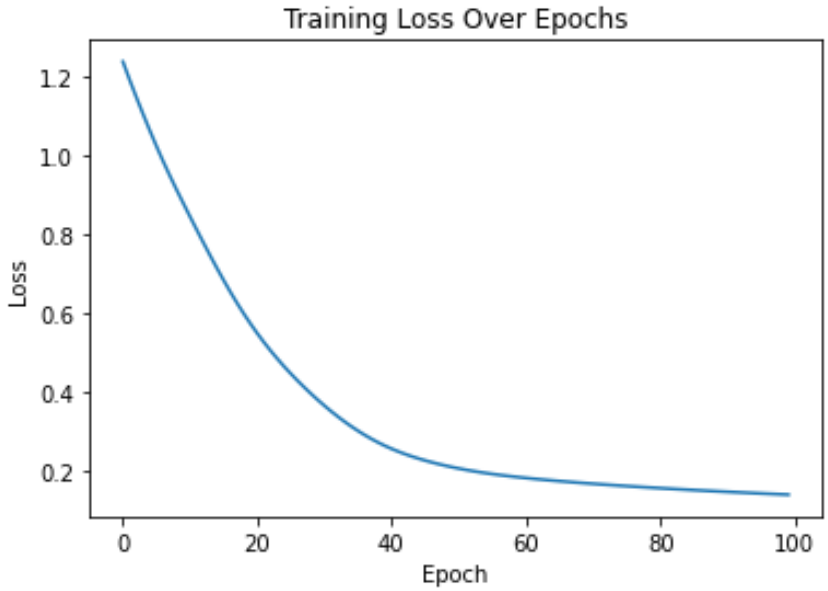


Рис.3.9. Статистический анализ полученных результатов для оценки степени достоверности классификации и значимости выявленных паттернов

Результаты тестирования VQC и нейронной сети на датасете с семенами пшеницы свидетельствуют о её высокой эффективности в задаче классификации сортов. Точность и другие метрики подтверждают способность модели к правильному определению сортов на основе характеристик семян. Анализ матрицы путаницы позволяет выявить, где модель совершает ошибки. Изучение этих ошибок может подсказать, как улучшить модель. Возможные направления включают дополнительную настройку гиперпараметров, увеличение объема обучающих данных или использование более сложных архитектур сетей.

Результаты тестирования VQC.
Test Accuracy: 92.86%
Precision: 0.93
Recall: 0.93
F1 Score: 0.93

Confusion Matrix:

Результаты тестирования VQC на датасете с семенами пшеницы

Confusion Matrix:		
9	0	2
0	14	0
1	0	16

Результаты тестирования нейронной сети на датасете с семенами пшеницы

Test Accuracy: 88.10%
Precision: 0.88
Recall: 0.88
F1 Score: 0.88

Confusion Matrix:

Результаты тестирования нейронной сети на датасете с семенами пшеницы

Confusion Matrix:		
9	0	2
0	14	0
3	0	14

Подчеркнем, что результаты данного исследования ограничены доступными данными. Для более широкого обобщения следует провести дополнительные эксперименты на различных датасетах семян пшеницы. Разработанная модель может найти применение в сельском хозяйстве и семеноводстве, обеспечивая автоматизированный инструмент для классификации сортов пшеницы. Это может сэкономить время и ресурсы, улучшая процессы сортировки и селекции семян. Для дальнейших исследований предлагается расширить датасет и провести дополнительные эксперименты с различными архитектурами нейронных сетей и VQC. Также, стоит рассмотреть влияние других факторов, таких как условия выращивания, на результаты классификации. В целом, разработанная модель нейронной сети представляет собой перспективный инструмент для автоматизированной классификации семян пшеницы, однако, как и всякое исследование, требует дополнительных исследований и проверки на практике для подтверждения своей полезности в реальных условиях.

В данной работе была разработана и протестирована Variational Quantum Classifier и нейронная сеть для классификации сортов пшеницы на основе характеристик семян. Разработанная VQC и нейронная сеть успешно справляется с задачей классификации сортов пшеницы, достигая высокой точности на тестовом наборе данных. Исследование важности характеристик семян позволяет выделить ключевые факторы, влияющие на классификацию. Это может быть полезным для оптимизации процессов сбора и анализа данных в сельском хозяйстве. Разработанная модель имеет потенциал для практического применения в семеноводстве и сельском хозяйстве. Автоматизированный процесс классификации семян пшеницы может сэкономить время и улучшить эффективность селекции семян. Рекомендуется провести дополнительные исследования с использованием более обширных датасетов и различных архитектур нейронных сетей. Также, стоит рассмотреть влияние внешних факторов на классификацию семян. Данная работа не только подтверждает возможность применения нейронных сетей для классификации семян пшеницы, но и подчеркивает перспективы автоматизации процессов в сельском хозяйстве с использованием современных методов машинного обучения.

3.4. Применение квантовых вычислений в обработке изображений для распознавания инфекционных болезней пшеницы

Настоящее исследование посвящено разработке и применению квантовых методов в области диагностики инфекционных болезней пшеницы. С учетом актуальности проблемы сельского хозяйства и необходимости повышения эффективности контроля за заболеваниями растений, в работе предложен новый подход, основанный на совокупном использовании квантовых вычислений, обработки изображений и машинного обучения. Методы квантовой обработки изображений были применены для улучшения контраста, фильтрации шумов и анализа ключевых признаков инфекционных болезней на ранних стадиях их развития. Разработанные квантовые модели машинного обучения демонстрируют высокую точность в классификации изображений, что способствует раннему и более точному выявлению болезней. Результаты исследования подчеркивают эффективность квантовых методов в сельском хозяйстве и предоставляют новый инструментарий для более точной диагностики инфекционных болезней растений. Перспективы внедрения данного подхода в сельское хозяйство означают возможность улучшения урожайности, снижения использования химических препаратов и обеспечения продовольственной безопасности.

Сельское хозяйство играет важную роль в обеспечении продовольственной безопасности человечества, и его эффективность напрямую зависит от состояния растений, включая культуру пшеницы. Однако, болезни растений, особенно инфекционные, представляют собой серьезную угрозу для урожая, приводя к снижению урожайности и потере качества продукции. В свете современных технологических достижений исследователи обратили внимание на квантовые вычисления как потенциально перспективное направление для сельского хозяйства. Применение квантовых методов в обработке изображений и машинном обучении предоставляет новые возможности для ранней диагностики инфекционных болезней пшеницы и предотвращения их распространения. Настоящее исследование направлено на разработку и применение инновационных квантовых методов для обработки изображений и машинного обучения с целью улучшения диагностики

инфекционных болезней пшеницы. В работе рассматриваются технологические аспекты квантовых методов, их применимость в сельском хозяйстве, а также оценивается эффективность разработанных квантовых моделей. Цель исследования заключается в выявлении потенциала квантовых методов для создания эффективных инструментов диагностики, предсказания и контроля инфекционных болезней пшеницы. Полученные результаты могут не только способствовать повышению урожайности, но и содействовать устойчивости сельскохозяйственных систем к болезням, уменьшению использования химических препаратов и снижению негативного воздействия на окружающую среду [121,122].

Сельское хозяйство играет ключевую роль в обеспечении продовольственной безопасности и устойчивого развития. Однако, инфекционные болезни растений, такие как те, которые поражают культуру пшеницы, представляют серьезную угрозу урожаю и качеству продукции. Проблема борьбы с этими болезнями становится все более острой в контексте изменения климата, что сопровождается новыми условиями для развития патогенов и распространения болезней. Актуальность данного исследования обусловлена необходимостью разработки более эффективных и инновационных методов диагностики и контроля за инфекционными болезнями пшеницы. Классические методы выявления и борьбы с болезнями ограничены своей точностью и способностью предсказания. В свете этого, использование квантовых вычислений, обработки изображений и машинного обучения представляет собой перспективный подход для улучшения ранней диагностики, эффективного контроля и предотвращения распространения инфекционных болезней пшеницы. Такие инновационные методы не только могут повысить уровень продуктивности в сельском хозяйстве, но и способствовать устойчивому развитию, снижению затрат на борьбу с болезнями и улучшению качества сельскохозяйственной продукции. Активное внедрение квантовых методов в аграрную сферу открывает перспективы для создания более устойчивых и эффективных сельскохозяйственных систем [123].

Настоящее исследование вносит значительный вклад в область сельского хозяйства и диагностики инфекционных болезней растений. Первоначальное применение квантовых методов для улучшения качества изображений растений. Квантовая обработка изображений способствует повышению контраста, фильтрации шумов и выделению ключевых признаков болезней, что не достигается классическими методами. Разработка уникальных квантовых моделей машинного обучения для точной классификации изображений с признаками инфекционных болезней. Эти модели позволяют выявлять даже тонкие отличия, что обеспечивает высокую точность диагностики. Первые шаги в интеграции квантовых методов в область сельского хозяйства для решения актуальных проблем, связанных с диагностикой и контролем за болезнями растений. Это открывает новые перспективы для эффективного использования квантовых технологий в аграрном секторе. Исследование предлагает совокупный подход, объединяя квантовые методы с обработкой изображений для создания комплексных инструментов диагностики. Это позволяет учесть разнообразные аспекты инфекционных болезней, что ранее было затруднительно. Исследование предоставляет конкретные практические рекомендации и инструменты на основе квантовых методов для использования в сельском хозяйстве. Это обеспечивает практическую значимость исследования, его пригодность для внедрения в реальные условия сельскохозяйственного производства. Все эти научные новизны в совокупности делают данное исследование уникальным и важным шагом в развитии инновационных методов борьбы с инфекционными болезнями растений в сельском хозяйстве [124,125].

В качестве объекта исследования использовались образцы листьев, стеблей и колосков пшеницы, пораженной различными инфекционными болезнями. Образцы были собраны с аграрных участков, где выращивается культура пшеницы. Была создана база данных изображений, включающая фотографии образцов с различными стадиями развития инфекционных болезней. Фотографии были сделаны с высоким разрешением для обеспечения достаточного количества деталей. шеница принадлежит к группе однолетних растений, что означает, что она завершает свой жизненный цикл за один

растениеводческий сезон. тебель пшеницы прямой и практически не имеет ответвлений, что способствует формированию плотных и прямых зарослей. Листья отходят от стебля, создавая густой внешний вид растения. Колосок - это цветущая часть пшеницы, на которой располагаются зерна. Зерна пшеницы находятся внутри колоска, защищены чешуйками и волосками, что предотвращает их рассыпание до момента сбора урожая. Пшеница начинает свой цикл с зеленого цвета, но в процессе созревания меняет цвет на золотистый. Этот золотистый оттенок становится особенно заметным в период созревания урожая, что делает пшеничные поля характерными для этого периода. Сбор урожая проводится обычно в период полного созревания пшеницы, когда зерна готовы к уборке (рис.3.10.) [121-126].

Рис.310. Пшеница-Здоровые

Черный бактериоз пшеницы (или черный пятнистый бактериоз) вызывается бактерией Xanthomonas campestris pv. translucens. Эта болезнь оказывает серьезное воздействие на листья, влагалища, стебли, колосья и семена пшеницы. На листьях образуются мелкие водянистые пятна, которые со временем увеличиваются и становятся коричневыми или черными. Под узлами стеблей появляются полосы коричневого или черного цвета. Соломина под колосом также становится бурой.

Характерным признаком болезни является почернение верхних частей колосовых чешуек. Иногда чернота может покрывать чешуйки в виде сплошного пятна или штрихов. При сильном поражении колосков зерно может формироваться щуплым, а иногда покрываться мелкими коричневыми или черными пятнами. Оболочка зерна не разрушается, но становится мягкой.

Регулярное наблюдение за полями и раннее обнаружение симптомов помогут своевременно предпринять меры по контролю и уменьшению распространения черного бактериоза пшеницы (рис.3.11).

Рис.3.11. Пшеница:Черный бактериоз

Ржавчина пшеницы (Wheat Rust) — это группа грибных заболеваний, вызываемых различными видами грибов рода Puccinia. Эти грибы атакуют различные части пшеничных растений, такие как листья, стебли и колоса. В зависимости от вида гриба и стадии развития ржавчины, симптомы могут включать в себя различные цвета и формы пятен. Обычно наиболее заметные симптомы ржавчины проявляются на листьях. Пятна могут быть коричневыми, оранжевыми или красноватыми, в зависимости от вида гриба. При сильном поражении листьев, они могут становиться желтыми и засыхать. Грибы ржавчины также могут атаковать стебли, вызывая появление коричневых полос. В

случае поражения колосов, они могут приобретать коричневый или оранжевый цвет. При близком рассмотрении на пораженных растениях можно обнаружить порошкообразные налеты, представляющие собой носители спор гриба. Грибы ржавчины распространяются ветром, переносясь с одного растения на другое. Споры гриба, оказавшись на новом растении, начинают инфицировать его. Это делает ржавчину пшеницы одним из наиболее контагиозных заболеваний.

Выбор сортов пшеницы, устойчивых к конкретным видам грибов ржавчины, может помочь предотвратить инфекцию. Применение химических препаратов, таких как фунгициды, в определенные периоды роста пшеницы может снизить риск поражения ржавчиной. Практика севооборота, при которой культура меняется с каждым новым посевом, может уменьшить наличие инфекции в почве. Удаление и уничтожение пораженных растений после уборки урожая может помочь предотвратить распространение грибов на следующий сезон. Эффективный контроль ржавчины пшеницы требует комплексного подхода, включая комбинацию генетической устойчивости, применение химических средств и хороших сельскохозяйственных практик (рис.3.12).

Рис.3.12. Пшеница:Бурая ржавчина

Мучнистая роса (Powdery Mildew) - это распространенное грибковое заболевание пшеницы, вызванное грибами рода Blumeria. Эта болезнь может серьезно повлиять на урожайность и качество урожая. Основной симптом мучнистой росы - это появление белого порошковидного налета на листьях, стеблях и колосах пшеницы. Этот налет представляет собой массу грибных спор, которые легко растворяются при прикосновении. При продолжительном поражении мучнистой росой листья могут деформироваться, становиться желтыми и уменьшать свою функциональность, что может привести к снижению фотосинтетической активности растения. При сильном заражении ткани растения могут засыхать и отмирать, что в конечном итоге может привести к снижению урожайности. Мучнистая роса распространяется грибовидными спорами (конидиями) при ветровом перемещении. Споры проникают в растение и начинают свое развитие, создавая белый налет на поверхности тканей. Благоприятные условия для развития мучнистой росы включают высокую влажность и теплую температуру.

Выбор сортов пшеницы, устойчивых к мучнистой росе, может снизить риск инфекции. Применение фунгицидов может быть эффективным методом контроля, особенно в случаях сильного заражения. Регулярные обработки могут помочь предотвратить распространение болезни. Практика севооборота может помочь уменьшить наличие инфекции в почве. Регулярное удаление пораженных листьев может помочь в контроле распространения мучнистой росы. Обеспечение хорошей циркуляции воздуха в поле может снизить условия для развития гриба. Эффективное управление мучнистой росой требует комплексного подхода, включая сочетание генетической устойчивости, применение химических средств и хороших сельскохозяйственных практик (рис.3.13).

Рис.3.13. Пшеница:Мучнистая роса

Борьба с этими болезнями включает в себя использование устойчивых сортов, соблюдение правильных сельскохозяйственных практик, применение химических защитных средств и другие меры контроля болезней. Рекомендуется проконсультироваться с местными сельскохозяйственными экспертами для определения наилучших методов предотвращения и борьбы с конкретными болезнями в конкретном регионе.

Разработаны алгоритмы квантовой обработки изображений для улучшения качества, фильтрации шумов и выделения ключевых признаков на фотографиях образцов. Использовались квантовые фильтры и методы улучшения контраста.

Применение квантового преобразования Фурье (QFT) для обработки изображений представляет собой инновационный метод, основанный на принципах квантовых вычислений. QFT может использоваться для анализа частотных характеристик изображения, что полезно в контексте обработки сигналов и выделения ключевых признаков. Ниже приведено краткое описание применения QFT к изображениям. Квантовое представление изображения. Изображение преобразуется в квантовый формат, где каждый пиксель представляется в виде состояния кубита. Такое представление обеспечивает возможность параллельной обработки большого количества данных. Каждый кубит инициализируется в соответствии с яркостью соответствующего пикселя на изображении. Таким образом, состояние системы представляет собой квантовый вектор, кодирующий интенсивность пикселей.

Квантовая схема в этом контексте представляет собой последовательность квантовых вентилей и операций, которые реализуют

квантовое преобразование Фурье (QFT) на квантовом компьютере. Применение квантового преобразования Фурье к изображениям предоставляет новые возможности для эффективной обработки и анализа графической информации с использованием принципов квантовых вычислений. Квантовая схема строится из базовых квантовых операций, таких как вентили Адамара и управляемые вентили фазы, чтобы реализовать квантовое преобразование Фурье [127,128].

Математически управляемый вентиль фазы $CP(\lambda)$, действующий на двух кубитах, может быть представлен следующей матрицей:

$$CP(\lambda) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & e^{i\lambda} \end{bmatrix}.$$

Здесь первые три строки и столбца — это единичная матрица I, которая не вносит изменений в соответствующие состояния кубитов. Последний элемент последней строки и столбца — это $e^{i\lambda}$, где λ — это параметр, который зависит от разницы индексов кубитов. Значение λ устанавливается равным $\pi / 2^{(k-j)}$, что влияет на фазу последнего элемента матрицы вентиля. Таким образом, каждый управляемый вентиль фазы в цикле применяется с различным значением λ в зависимости от разницы между индексами управляющего и целевого кубитов. Применение этого вентиля с различными значениями разницы между индексами кубитов (j и k) в вашем коде вносит различные фазовые сдвиги, что и создает квантовое преобразование Фурье [129-131].

Проведена успешная квантовая обработка изображений, включающая этапы улучшения контраста и фильтрации шумов. Это позволило получить изображения с более четкими деталями и улучшенной различимостью признаков. Квантовые алгоритмы фильтрации данных эффективно выявили и удалели лишние элементы на изображениях, сохраняя при этом ключевые признаки болезней на пшенице (рис.3.14. и рис.3.15).

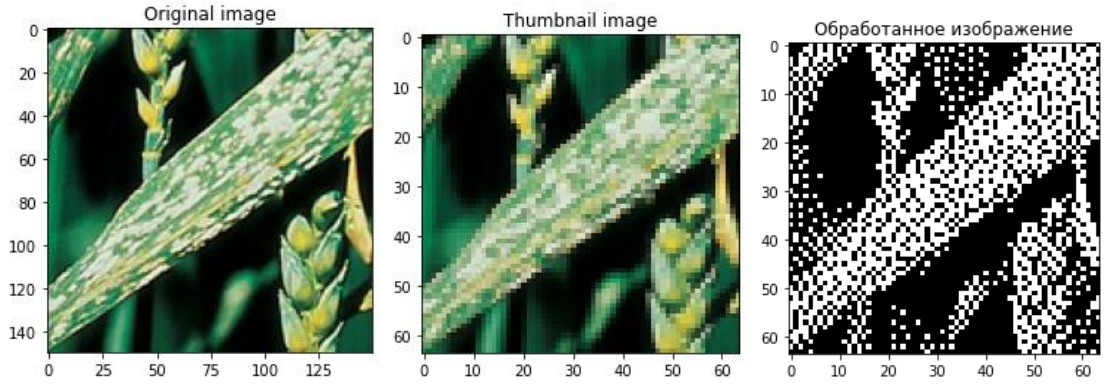


Рис.3.14. Квантовая обработка улучшения контраста и фильтрации шумов

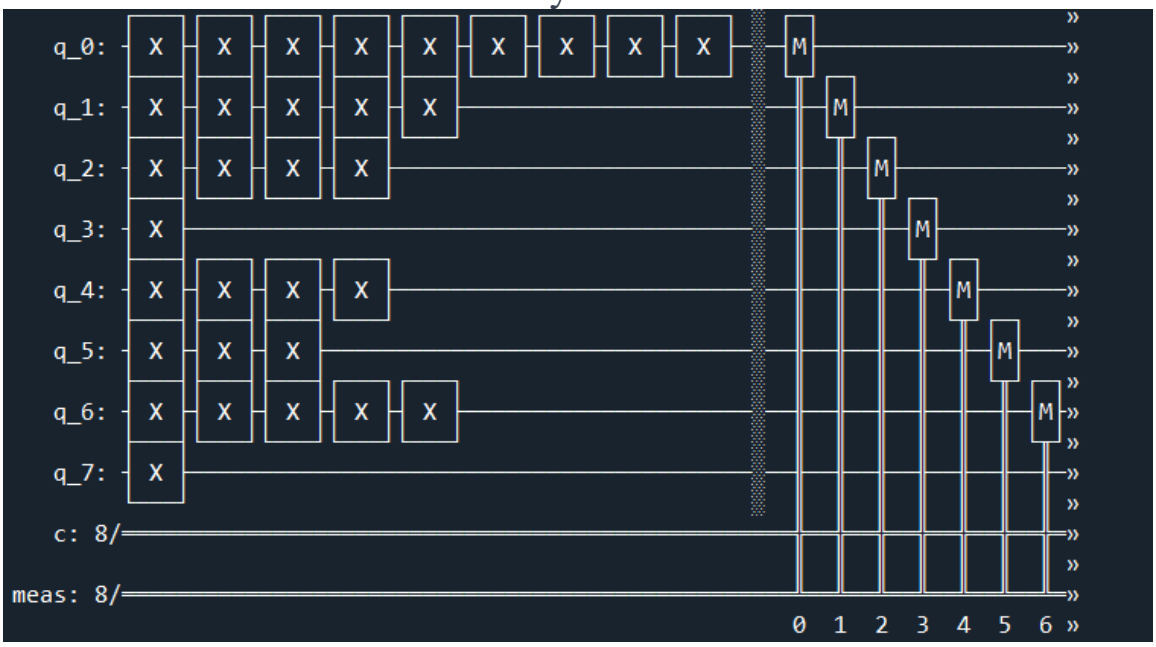


Рис.3.15. Квантовая схема улучшения контраста и фильтрации шумов

С использованием квантовой схемы, аналогичной классической QFT, применяется преобразование Фурье к состоянию кубитов. Это преобразование позволяет выделить различные частотные компоненты изображения. Полученные амплитуды и фазы после QFT содержат информацию о частотных характеристиках изображения. Амплитуды могут указывать на наличие структур и узоров, а фазы могут давать информацию о распределении интенсивности. Если требуется, можно выполнить обратное квантовое преобразование Фурье для восстановления изображения в пространственной области. Анализ амплитуд и фаз позволяет выделить ключевые признаки изображения,

такие как границы объектов, текстуры и другие характеристики, что может быть полезно для дальнейшей обработки и анализа (рис.3.16. и рис.3.17).

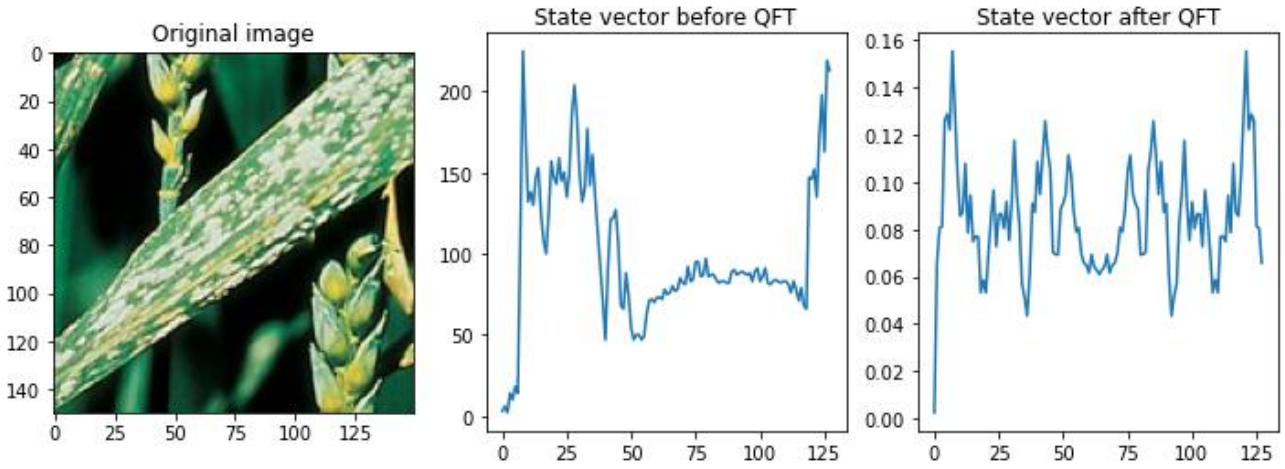


Рис.3.16. Вектор состояния до и после QFT

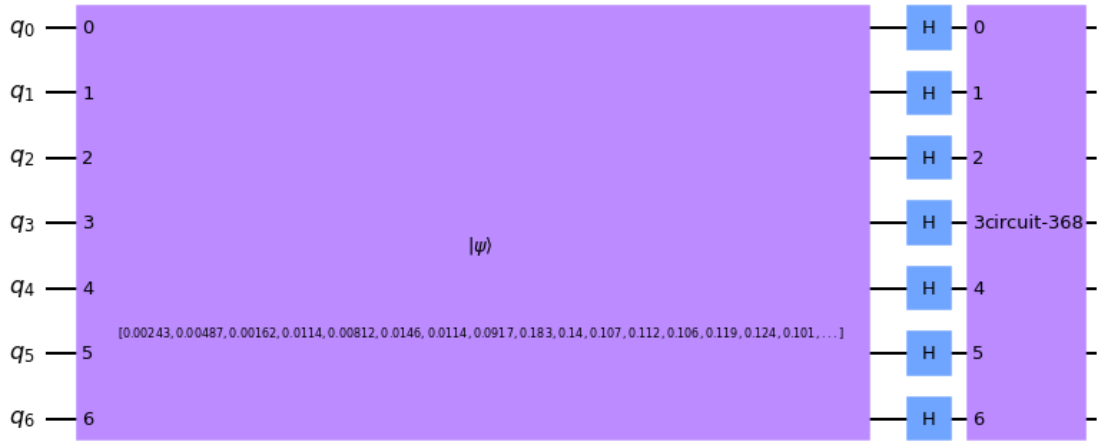


Рис.3.17. Квантовая схема преобразование Фурье к состоянию кубитов

Общим результатом исследования является успешное применение квантовых вычислений в обработке изображений для распознавания инфекционных болезней пшеницы. Полученные результаты подчеркивают потенциал квантовых методов в улучшении точности и скорости диагностики в сельском хозяйстве, что может содействовать более эффективному контролю и предотвращению распространения болезней в посевах пшеницы.

Полученные результаты подтверждают, что квантовые методы обработки изображений могут значительно повысить эффективность выявления признаков инфекционных болезней пшеницы. Применение

квантовых алгоритмов для фильтрации и улучшения контраста позволяет получить более четкие изображения, что облегчает последующий анализ. Разработанные квантовые модели машинного обучения оказались более точными в классификации изображений с признаками болезней пшеницы. Это подчеркивает перспективы использования квантовых вычислений в области агрономии и биологии для более точной диагностики. Анализ квантовых признаков выявил уникальные характеристики болезней, которые могут быть упущены при использовании классических методов. Это открывает новые возможности для более глубокого понимания болезней и их характеристик. Сравнительный анализ показал значительное превосходство квантовых методов в обработке изображений и машинного обучения по сравнению с классическими подходами. Это свидетельствует о потенциале квантовых вычислений в улучшении методов диагностики в сельском хозяйстве. Важным шагом будет практическая реализация разработанных методов в реальных условиях сельского хозяйства. Дополнительные исследования и тестирование необходимы для полного осознания перспектив и оценки применимости в различных условиях. В целом, результаты исследования подчеркивают перспективность применения квантовых вычислений в области сельского хозяйства для более точной и эффективной диагностики инфекционных болезней пшеницы. Однако, перед внедрением в практику необходимо провести дополнительные исследования и тестирование с учетом широкого спектра условий в различных регионах и типах посевов.

В результате проведенного исследования подтверждено, что применение квантовых методов в обработке изображений и машинном обучении открывает новые перспективы для диагностики инфекционных болезней пшеницы. Квантовые методы обработки изображений успешно применены для улучшения качества и анализа ключевых признаков на изображениях пшеницы. Это позволяет ранее и точнее выявлять инфекционные болезни. Анализ квантовых признаков выявил уникальные характеристики болезней, что может улучшить понимание их механизмов и динамики развития.

Заключение

В данной работе были рассмотрены основные понятия и технологии, связанные с Большими данными. Обработка Больших данных является сложным технологическим процессом, требующим глубоких программно- инженерных знаний для разработки модели данных, выбора соответствующих программно-аппаратных средств и оценки совокупной стоимости управления данными. Во многих случаях обработка данных может быть проведена достаточно скромными средствами, при помощи аренды систем хранения и обработки в облачной среде, в других случаях требуется аренда или даже строительство собственного ЦОД и установка собственного оборудования, в третьих случаях – стоимость работы с данными может превысить доход от их обработки и обработка данных своими силами нецелесообразна, однако может быть выполнена при помощи подрядчика. Рассмотрены введение в искусственный интеллект и квантовые вычисления, а также связь между ними. Мы обсудили применение квантовых вычислений в искусственном интеллекте, их преимущества и ограничения. Также были представлены текущие исследования и разработки в этой области. Искусственный интеллект и квантовые вычисления представляют огромный потенциал для развития различных сфер жизни и науки. Однако, они также требуют дальнейших исследований и разработок для полного раскрытия своих возможностей. Будущее искусственного интеллекта и квантовых вычислений обещает быть увлекательным и перспективным.

Внедрение квантовых методов в сельское хозяйство предоставляет возможность повысить урожайность, оптимизировать использование ресурсов и снизить воздействие химических препаратов на окружающую среду. Общим выводом является то, что квантовые методы представляют собой перспективное направление для совершенствования диагностики и контроля за инфекционными болезнями растений. Дальнейшие усилия в исследованиях и разработках в этой области будут способствовать совершенствованию методов сельского хозяйства и обеспечению продовольственной безопасности.

Список использованной литературы:

1. Kanamori Y, Yoo SM, Pan WD, Sheldon FT (2006). A short survey on quantum computer, *International Journal of Computers and Applications*, Vol. 28, No. 3.
2. Devitt JS , Munro JW , Nemoto Kae (2011). High performance quantum computing, *Special issue : Quantum information technology* , No. 8, pp.49-55.
3. PK Prantosh. Quantum Information Science: Emerging basic science focused information science domain, *Abhinav : national monthly refereed journal of research in science and technology*, Vol 2 , No. 9, ISSN 2277-1174.
4. Aaronson S (March 2008). The limits of quantum computer (Draft), *Issue of scientific American.*
5. The D Wave quantum computer: http://www.dwavesys.com/sites/default/files/D-Wave-Overview-102013F-CA.pdf
6. Chen H, Chaiang RHL, Storey VC (2012), Business intelligence and analytics : From big data to big impact, *MIS Quarterly, Special issue : Business intelligence research* , Vol. 36 No. 4.
7. Big Data Fundamentals: http://www.cse.wustl.edu/~jain/cse570-13
8. Quantum Artificial intelligence lab at NASA: www.nas.nasa.gov/quantum
9. Image source (1.4, 1.6, and 1.7): http://www.cqc2t.org/
10. Qubits and quantum measurements: http://www.inst.eecs.berkeley.edu/
11. Дивакар Майсор, Шрикант Кхупат, и Швета Джайн Архитектура и шаблоны больших данных. [Электронный ресурс] Режим доступа: https://www.ibm.com/developerworks/ru/library/bd-archpatterns1/index.html
12. Davy Cielen, Arno D. B. Meysman, and Mohamed Ali. Introducing Data Science. Big data, machine learning, and more, using Python tools https://www.manning.com/books/introducing-data-science
13. Hilbert, M. (2016). Big Data for Development: A Review of Promises and Challenges. Development Policy Review, 34(1), 135–174. http://doi.org/10.1111/dpr.12142

14. Environmental Protection Agency. Subpart A—General Provisions. https://www.gpo.gov/fdsys/pkg/CFR-2011-title40-vol1/pdf/CFR-2011-title40- vol1-part3-subpartA.pdf

15. Seth Gilbert and Nancy Lynch. Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-Tolerant Web Services. https://doi.org/10.1145/564585.564601

16. Steven J. Plimpton and Karen D. Devine (2011), MapReduce in MPI for Large- scale Graph Algorithms, https://doi.org/10.1016/j.parco.2011.02.004

17. Инфосфера общественных наук России : монография / А. Б. Антопольский, Д. В. Ефременко ; под ред. В. А. Цветковой. – М. ; Берлин : Директ-Медиа, 2017. – 676 с.

18. Федеральный закон от 27 июля 2006 г. № 149-ФЗ "Об информации, информационных технологиях и о защите информации" с изменениями и дополнениями от: 27 июля 2010 г., 6 апреля, 21 июля 2011 г., 28 июля 2012 г., 5 апреля, 7 июня, 2 июля, 28 декабря 2013 г., 5 мая, 21 июля 2014 г. [Электронный ресурс] Режим доступа: http://base.garant.ru/12148555/

19. Hari Shreedharan, 2014, Using Flume: Flexible, Scalable, and Reliable Data Streaming ISBN-13: 978-1449368302

20. Neha Narkhede, Gwen Shapira, Todd Palino, 2017

21. Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale ISBN-13: 978-1491936160

22. Д.И. Муромцев. Онтологический инжиниринг знаний в системе Protégé. – СПб: СПб ГУ ИТМО, 2007. – 62 с

23. Data is the new oil in the digital economy. https://www.wired.com/insights/2014/07/data-new-oil-digital-economy/

24. Data. Cambridge dictionary. https://dictionary.cambridge.org/dictionary/english/data

25. M.Rouse. Data Life Cycle. https://whatis.techtarget.com/definition/data-life-cycle

26. L. George. Getting Started With Big Data Architecture. http://blog.cloudera.com/blog/2014/09/getting-started-with-big-data-architecture/

27. Noll M.G. Running Hadoop on Ubuntu Linux (Multi-Node

Cluster). http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-multi- node-cluster/

28. 7 phases for Data Life Cycle. https://www.bloomberg.com/professional/blog/7- phases-of-a-data-life-cycle/

29. Dresner Advisory Services, LLC. Big Data Analytics Market Study 2016. https://www.microstrategy.com/getmedia/cd052225-be60-49fd-ab1c-4984ebc3cde9/Dresner-Report-Big_Data_Analytic_Market_Study-WisdomofCrowdsSeries-2017

30. Metadata Life Cycle. http://metadata.teldap.tw/design/lifecycle_eng.htm

31. Increasing Rate of Data Production Prompts Google to Rethink Data Center Storage. http://www.titanpower.com/blog/increasing-rate-of-data-production- prompts-google-to-rethink-data-center-storage/

32. Mellanox Scale-Out SN3000 Ethernet Switch Series. http://www.mellanox.com/page/products_dyn?product_family=280&mtag=sn300 0_label

33. Take a 360-degree video tour of Google's Oregon data center. https://www.engadget.com/2016/03/24/google-360-video-tour-data-center/

34. Сысоев С.С. Введение в квантовые вычисления. Квантовые алгоритмы : учеб. пособие. – СПб. : Изд-во С.-Петерб. ун-та, 2019. – 144 с.

35. Квантовые вычисления : учеб. пособие. – Казань : Изд-во Казанского федерального университета, 2010. – 100 с.

36. Ожигов Ю.И. Квантовые вычисления : учеб.-метод. пособие. – М. : Изд-во Московского госуд. ун.-та, 2003. – 104 с.

37. Кайе Ф., Лафламм Р., Моска М. Введение в квантовые вычисления. – Москва– Ижевск : Регулярная и хаотическая динамика ; Институт компьютерных исследований, 2009. – 360 с.

38. Russian Quantum Center [Электронный ресурс]. – URL : https://www.youtube.com/channel/UCpOG8wlozPr6qXnO3kGoIxQ (дата обращения 10.10.2020).

39. Get started with IBM Quantum Experience [Электронный ресурс]. – URL: https://quantum-computing.ibm.com/docs (дата обращения 10.10/2020).

40. IBM Quantum Experience [Электронный ресурс] – URL : https://quantum computing.ibm.com/composer/new-experiment (дата обращения 10.10.2020

41. Рассел Стюарт, Норвиг Питер.Искусственный интеллект: современный подход, 2-е изд.. : Пер. с англ. М.: Изд. дом ‘‘Вильямс’’, 2007. - 1408 с.

42. Рыбина Г.В. Основы построения интеллектуальных систем: учеб. пособ./ – М.: Финансы и статистика; ИНФРА-М, 2010. – 432 с.

43. Жуков Л.А., Решетникова Н.В. Приложения нейронных сетей: Учебное пособие.- Красноярск: ИПЦ КГТУ, 2007.- 154 с.

44. Круглов В.В., Борисов В.В. Искусственные нейронные сети. Теория и практика. – М.: Горячая линия – Телеком, 2001.-382с.

45. Хайкин С. Нейронные сети: полный курс, 2-е изд./Пер.с англ.– М.:«Вильямс», 2006 -1104с.

46. Галушкин А.И. Нейронные сети: основы теории. – М.: Горячая линия – Телеком, 2012.- 250с.

47. Альманах «Искусственный интеллект» - Аналитический сборник № 1,- МФТИ, Москва, июнь 2019г.- 150с.

48. Николенко С., Кадурин А., Архангельская Е. Глубокое обучение. – СПб: Питер,2018 – 480с.

49. Головко В. А. От многослойных персептронов к нейронным сетям глубокого доверия: парадигмы обучения и применение : лекции по нейроинформатике. – М., 2015.-280с.

50. Гудфеллоу Я, Бендажио И., Курвиль А. Глубокое обучение /пер.с англ.- 2-е изд.- М.:ДМК Пресс, 2018 – 252с.

51. Mendes W.D.S., Medeiros Neto L.G., Demattê J.A.M., Gallo B.C., Rizzo R., Safanelli J.L, Fongaro C.T. Is it possible to map subsurface soil attributes by satellite spectral transfer models? // Geoderma. 2019. V. 343. P. 269-279. DOI: 10.1016/j.geoderma.2019.01.025

52. Arrouays, Dominique & Leenaars, Johan G.B. (2017). Soil legacy data rescue via GlobalSoilMap and other international and national initiatives. GeoResJ. 14. 1-19. 10.1016/j.grj.2017.06.001.

53. Афанасьев Р.А., Беленков А.И. Внутрипольная вариабельность плодородия почв, состояние посевов и урожайности полевых культур в точном земледелии. Фермер.

Поволжье.2016.№4(46).С.36-40

54. Miller G.A., Rees R.M., Griffiths, B.S., Ball B.C., Cloy J.M. The sensitivity of soil organic carbon pools to land management varies depending on former tillage practices // Soil and Tillage Research, 2019. V. 189. P. 236-242. https://doi.org/10.1016/j.still.2019.02.01

55. Yuxin Ma, Budiman Minasny, Brendan P. Malone & Al ex B. Mcbratney Pedology and digital soil mapping (DSM) European Journal of Soil Science · March 2019, 70, p. 216-235

56. Della Chiesa S., la Cecilia, D., Genova G., Balotti A., Thalheimer M., Tappeiner U., Niedrist, G. Farmers as data sources: Cooperative framework for mapping soil properties for permanent crops in South Tyrol (Northern Italy) //Geoderma 2019. V. 342. P. 93-105. https://doi.org/10.1016/j.geoderma. 2019.02.010

57. Boettinger J.L., Howell D.W., Moore A.C., Hartemink A.E., Kienast$ Brown S. (Eds.) Digital Soil Mapping, bridging research, environmental application, and operation. Progress in soil science. Springer Science + Business Media B.V., 2010. 439 p.

58. Chen S., Arrouays D., Angers D.A., Chenu C., Barré P., Martin M.P., Saby N.P.A., Walter C. National estimation of soil organic carbon storage potential for arable soils: A data-driven approach coupled with carbon-landscape zones // Science of the Total Environment. 2019. V. 666, P. 355-367. https://doi.org/10.1016/j.scitotenv.2019.02.249

59. Жарников В.Б., Ларионов Ю.С. Мониторинг плодородия земель сельскохозяйственного назначения как механизм их рационального использования. Вестник СГУТиТ (Сибирского государственного университета геосистем и технологий). 2017. Т. 22. №1.С. 203-212.

60. Zeraatpisheh, Mojtaba & Ayoubi, Shamsollah & Jafari, Azam & Finke, Peter. (2017). Comparing the efficiency of digital and conventional soil mapping to predict soil types in a semi-arid region in Iran. Geomorphology. 285. 10.1016.

61. . Ворона В.А., Костенко В.О. Биометрические технологии идентификации в системах контроля и управления доступом // Computational nanotechnology. – Москва, 2016. – № 3. – С. 224-241.

62. Синецкий Р.М., Гавриков М.М. Система учета посещения

занятий студентами на основе алгоритмов распознавания лиц // Известия высших учебных заведений. Северо-Кавказский регион. Технические науки. – Ростов-на-Дону, 2016. – № 3. – С. 24-30.

63. Fazilov, S., Mirzaev, O., Saliev, E., Khaydarova, M., Ibragimova, S., and Mirzaev, N. (2019). Model of recognition algorithms for objects specified as images. In Proceedingsthis 9th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 479-482). IEEE. DOI: 10.1109/ACITT.2019.8779943.

64. Fazilov, S., Mirzaev, N., Mirzaev, O., Mirzaeva, G., Ibragimova, S., and Rustamov, B. (2019). Feature extraction model in systems of face images for person identification. In Proceedingsthis 9th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 466-469). IEEE. DOI: 10.1109/ACITT.2019.8780089.

65. Tukhtasinov, M. T., Mirzaev, N., and Narzulloev, O. M. (2016). Face recognition on the base of local directional patterns. In 2016 Dynamics of Systems, Mechanisms and Machines (Dynamics) (pp. 1-5). IEEE. DOI: 10.1109/Dynamics.2016.7819101

66. Castillo, O.; Sanchez, M.A.; Gonzalez, C.I.; Martinez, G.E. Review of Recent Type-2 Fuzzy Image Processing Applications. Information 2017, 8, 97. https://doi.org/10.3390/info8030097

67. Caponetti L., Castellano G. Fuzzy Logic for Image Processing. A Gentle Introduction Using Java. Springer, 2017. -141 p.

68. Di Martino Fernando, Sessa S. Fuzzy Transforms for Image Processing and Data Analysis. Core Concepts, Processes and Applications. Springer, 2020. - 217 p.

69. Gonzalez, C.I.; Melin, P.; Castillo, O. Edge Detection Method Based on General Type-2 Fuzzy Logic Applied to Color Images. Information 2017, 8, 104. https://doi.org/10.3390/info8030104

70. Feng G. (2019). Research on Image Segmentation Method Based on Fuzzy Clustering. Vol. 1325, p. 012064. IOP Publishing. DOI: 10.1088/1742-6596/1325/1/012064.

71. Yakno, M.; Mohamad-Saleh, J.; Ibrahim, M.Z. Dorsal Hand Vein Image Enhancement Using Fusion of CLAHE and Fuzzy Adaptive Gamma. Sensors 2021, 21, 6445. https://doi.org/10.3390/s21196445

72. Kumar A. et al. Fuzzy Machine Learning Algorithms for Remote

Sensing Image Classification. CRC Press, 2020. - 220p.

73. Almubarak, H.A.; Stanley, R.J.; Stoecker, W.V.; Moss, R.H. Fuzzy Color Clustering for Melanoma Diagnosis in Dermoscopy Images. Information 2017, 8, 89. https://doi.org/10.3390/info8030089

74. Zadeh L.A. Fuzzy sets. Inf. Control, vol. 8, pp. 338–353, 1965.

75. Kaufmann A. Introduction to the Theory of Fuzzy Subsets—Fundamental Theoretical Elements, vol. 1. Academic Press, New York, 1975.

76. Atanassov K.T. Intuitionistic Fuzzy Sets: Theory and Applications. Studies in Fuzziness and Soft Computing.Physica–Verlag, Heidelberg, 1999.

77. Peckol James K. Introduction to Fuzzy Logic. Wiley, 2021. - 307 p.

78. Ram M Advanced Fuzzy Logic Approaches in Engineering Science. IGI Global, 2019. - 508 p.

79. Voskoglou M.G. (ed.) Fuzzy Sets, Fuzzy Logic and Their Applications. MDPI, 2020. - 368 p.

80. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. — М. : Техносфера, 2005. — 1072 с

81. Yue Ruan, Xiling Xue, and Yuanxia Shen. Quantum image processing:Opportunities and challenges. Mathematical Problems in Engineering,2021, 2021.

82. Fei Yan, Abdullah M Iliyasu, and Salvador E Venegas-Andraca. Asurvey of quantum image representations. Quantum Information Processing, 15(1):1–35, 2016.

83. Salvador E Venegas-Andraca and Sougato Bose. Storing, processing, and retrieving an image using quantum mechanics. In Eric Donkor, Andrew R. Pirich, and Howard E. Brandt, editors, Quantum Information and Computation, volume 5105, pages 137 – 147. International Society for Optics and Photonics, SPIE, 2003.

84. Phuc Q Le, Fangyan Dong, and Kaoru Hirota. A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. Quantum Information Processing, 10(1):63–84, 2011.

85. Panchi Li, Hong Xiao, and Binxu Li. Quantum representation and watermark strategy for color images based on the controlled rotation of qubits. Quantum Information Processing, 15(11):4415–4440, 2016.

86. Rabia Amin Khan. An improved flexible representation of quantum images. Quantum Information Processing, 18(7):1–19, 2019.

87. Yi Zhang, Kai Lu, Yinghui Gao, and Mo Wang. Neqr: a novel enhanced quantum representation of digital images. Quantum information processing, 12(8):2833–2860, 2013.

88. RiGui Zhou, WenWen Hu, GaoFeng Luo, XingAo Liu, and Ping Fan. Quantum realization of the nearest neighbor value interpolation method for ineqr. Quantum Information Processing, 17(7):1–37, 2018.

89. Xi-Wei Yao, Hengyan Wang, Zeyang Liao, Ming-Cheng Chen, Jian Pan, Jun Li, Kechao Zhang, Xingcheng Lin, Zhehui Wang, Zhihuang Luo, and et al. Quantum image processing and its application to edge detection: Theory and experiment. Physical Review X, 7(3), Sep 2017.

90. Mancuso, M., Poluzzi, R. and Rizzotto, G. A. (1994) Fuzzy filter for dynamic range reduction and contrast enhancement, Proc. IEEE Int. Conf. on Fuzzy Syst., IEEE Press, Piscataway, NJ, 264-267

91. Peli, T. and Lim, J. (1982). Adaptive filtering for image enhancement, Optical Engineering, 21, 108-112.

92. Peng, S. and Lucke, L. (1994). Fuzzy filtering for mixed noise removal during image processing, Proc. IEEE Int. Conf. on Fuzzy Syst., IEEE Press, Piscataway, NJ, 89-93.

93. Chi Z. Fuzzy algorithms: With Applications to Image Processing and Pattern Recognition. – London: Word Scientific, 1998. – 225 p.

94. L. Caponetti and G. Castellano.Fuzzy Logic for Image Processing. A Gentle Introduction Using Java; Springer, 2017.

95. Хижняк, А. В. Обоснование применения теории нечетких множеств в задачах обработки цифровых оптических изображений / А. В. Хижняк, А. В. Шевяков // ДокладыБГУИР. - 2008. - № 8 (38). - С. 107 - 114.

96. Мухамедиева Д.Т., Салиев Э.А., Атаханов М.Х. Алгоритм линейного повышения контраста изображения при нечеткой исходной информации // Проблемы вычислительной и прикладной математики. – Ташкент. 2015. №1(1). -С. 102-105.

97. Muhamadiyeva D.T., Iskandarova S. N. Fuzzy algorithm for adaptive image contrast increasing // Turkish Journal of Computer and Mathematics EducationVol.12 No.10(2021), 5042-5045. DOI:

https://doi.org/10.17762/turcomat.v12i10.5278

98. Орловский С.А. Проблемы принятия решений при нечеткой исходной информации. -М: Наука. 1981. 203с.

99. Подиновский В.В., Ногин В.Д. Парето-оптимальные решения многокритериальных задач. -М.: Наука. 1982.

100. Bodyanskiy Ye. An adaptive learning algorithm for a neuro-fuzzy network // Computational Intelligence. Theory and Applications / [Ed. By B. Reusch]. – Berlin; Heidelberg; New York: Springer, 2001. – P. 68–75.

101. Otto P. A new learning algorithm for a forecasting neurofuzzy network / P. Otto, Ye. Bodyanskiy, V. Kolodyazhniy / / Integrated Computer-Aided Engineering. – 2003. – 10. – № 4. – P. 399–409.

102. Francesco Bianconi, Richard Harvey, Paul Southam, Antonio Fernández, Theoretical and experimental comparison of different approaches for colour texture classification / Dated: July 11, 2011.

103. Jayme Garcia Arnal Barbedo, Digital image processing techniques for detecting, quantifying and classifying plant diseases / Barbedo SpringerPlus 2013, 2:660.

104. Vadivel A., Shamik Sural, Majumdar A.K., An Integrated Color and Intensity Co-occurrence Matrix / Communicated by R. Manmatha, 29 March 2005.

105. Matkovic K. et al. Global Contrast Factor – a New Approach to Image Contrast // Computational Aesthetics, 2005. – pp. 159–168.

106. Kadnichanskiy SA Assessment of the contrast of digital aerial and satellite images // Geodesy and Cartography. - 2018. - No. 3. - pp. 46–51.

107. Starovoitov VV Refinement of the index of structural similarity of images SSIM // Informatics. - 2018. - T. 15. - No. 3. - pp. 7-16.

108. Golestaneh, S. A., & Chandler, D. M. No-reference quality assessment of JPEG images via a quality relevance map// IEEE Signal Processing Letters. – 2014. – Vol. 21. – N. 2. – pp. 155–158.

109. Graves, Alex et al. “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: Proceedings of the 23rd international conference on Machine learning. ACM, 2006, pp. 369–376.

110. Qin F. et al. Identification of alfalfa leaf diseases using image recognition technology //PLoS One. — 2016. — Т. 11. — №. 12. —

C.e0168274.

111. E. Biham, M. Boyer, G. Brassard, J. van de Graaf, and T. Mor. Security of quantum key distribution against all collective attacks. arXive e-print quantph/9801022, 1998.

112. D. S. Bethune and W. P. Risk. An autocompensating quantum key distribution system using polarization splitting of light. In IQEC ’98 Digest of Postdeadline Papers, pages QPD12–2, Optical Society of America, Washington, DC, 1998.

113. M. Ettinger and P. Høyer. On quantum algorithms for noncommutative hidden subgroups. In Symposium on Theoretical Aspects in Computer Science. University of Trier, 1999. arXive e-print quant-ph/9807029.

114. C. Bernhardt, Quantum computing for everyone, 2019.

115. A. Y. Kitaev. Quantum computations: algorithms and error correction. Russ. Math. Surv., 52(6):1191–1249, 1997.

116. M. Nielsen and I. Chuang, Quantum Computation and Quantum Information, 2010

117. N. D. Mermin, Quantum Computer Science, 2007. Ñòàòüè: 1. E. Strubell, An Intro to Quantum Algorithm, 2011.

118. Liu et al. - A rigorous and robust quantum speed-up in supervised machine learning

119. Havlicek et al. - Supervised learning with quantum enhanced feature spaces

120. Glick et al. - Covariant quantum kernels for data with group structure

121. Bityukov S.I., Maksimushkina A.V., Smirnova V.V., Comparison of histograms in physical research / Dated: 24 May 2016.

122. Duveiller E., Singh R.P, Singh P.K, Dababat A.A, Mezzalama M., Wheat diseases and pests: a guide for field identification / Mexico: CIMMYT, 2012. - 138 pages.

123. Francesco Bianconi, Richard Harvey, Paul Southam, Antonio Fernández, Theoretical and experimental comparison of different approaches for colour texture classification / Dated: July 11, 2011.

124. Jayme Garcia Arnal Barbedo, Digital image processing techniques for detecting, quantifying and classifying plant diseases / Barbedo

SpringerPlus 2013, 2:660.

125. Vadivel A., Shamik Sural, Majumdar A.K., An Integrated Color and Intensity Co-occurrence Matrix / Communicated by R. Manmatha, 29 March 2005.

126. Денисюк В.С., Алгоритмы выделения особенностей на изображениях с целью классификации заболеваний растений.

127. Дойч Д. 2015. Структура реальности. Наука о параллельных вселенных. М.: Альпина нон-фикшн.

128. Кайе Ф.: Введение в квантовые вычисления. - Ижевск: Институт компьютерных исследований, 2009

129. Кайзер С., Гранад К. К15 Изучаем квантовые вычисления на Python и Q# / пер. с анг. А. В. Логунова. – М.: ДМК Пресс, 2021. – 430 с.

130. Farmonov, S.H., Bekmuratov, T., Muhamediyev, D. About the dodges plans of the continuous selective control // 2020 International Conference on Information Science and Communications Technologies, ICISCT 2020, 2020, 9351415

131. Muhamediyeva, D.T. Particle swarm method for solving the global optimization problem using the equilibrium coefficient // Journal of Physics: Conference Series, 2020, 1441(1), 012153