# Approaches to Solving Problems of Markov Modeling Training in Speech Recognition

D. T. Muxamediyeva[1](✉), N. A. Niyozmatova[1], R. A. Sobirov[1], B. N. Samijonov[2], and E. Kh. Khamidov[1]

[1] Digital Technologies and Artificial Intelligence, "Tashkent Institute of Irrigation and Agricultural Mechanization Engineers" National Research University, Tashkent, Uzbekistan
dilnoz134@rambler.ru

[2] Sejong University, Seoul, South Korea

**Abstract.** The article discusses approaches to solving problems of learning Markov modeling in speech recognition. A Markov process is a stochastic process consisting of a sequence of random states, where the probability of transition from one state to another depends only on the current state and does not depend on previous states. The result of observing such a process is a sequence of states that the system goes through during the observation period. The task of model training is considered the most difficult when using Markov models in recognition systems, since there is no known unique and universal way to solve it, and the quality of recognition depends on the result of model training. Therefore, special attention must be paid to training the model.

**Keywords:** genetic algorithm · training · automatic speech recognition · speech · Hidden Markov Models · Mel-cepstral coefficients · ant colony algorithm · immune algorithm

## 1 Introduction

At present, speech recognition technologies are intensively developing. There are many areas where speech technologies can find effective practical applications, such as medicine, banking, telephony, robotics, the automotive industry, forensics, etc. [1].

Currently, special attention is paid to the following promising areas of speech technologies:

1. Speech recognition for human-machine interfaces: such technologies are used in smart homes, voice assistants, automotive systems, mobile and other devices where voice control is required.
2. Speech synthesis technologies: These technologies are used to create an artificial voice, for example, for people with a speech impairment.
3. Systems that determine the personality of a speaker can be used for two main purposes: verification and identification.


F. P. García Márquez et al. (Eds.): ICETAI 2023, LNNS 960, pp. 344–357, 2024.
https://doi.org/10.1007/978-3-031-56728-5_29

Speaker verification systems are used to verify a person's identity. Typically, such systems are used to authorize users to access computer and information systems, as well as to control access. In this case, the system checks whether there is a voice in a pre-created voice database that the system expects to hear. Speaker identification systems, on the other hand, allow the identification of a person. Such systems are used in areas such as military affairs, communications, forensics, etc. The above-mentioned systems use methods and algorithms for speech recognition and signal processing, which make it possible to extract unique characteristics of the voice, such as frequencies, intonations, and rhythms [2–11]. One such method is Hidden Markov Models (HMMs).

HMMs are a mathematical tool for modeling data sequences where the next state depends only on the current state and is independent of previous states. HMMs can be used for speech recognition, where data sequences represent speech commands and model states correspond to speech phonemes [12].

The task of model training is one of the key and complex tasks in Markov modeling. Model training is based on the existing training data set to create a statistical model that will reflect the dependencies between the states of the system and its input data. The learning problem can be solved using the Baum-Welch algorithm, which uses the maximum likelihood method to estimate model parameters and evolutionary algorithms. Such algorithms make it possible to determine the optimal values for the model parameters, maximizing the probability of obtaining an observed sequence for a given model [13].

Let, given a system that at an arbitrary point in time can be in one of N different states $S_1$, $S_2$, ..., $S_N$.

At discrete times $t = 1, 2, ...$ The system passes from one state to another, but can remain in the existing state (the current state at time t will be described as $q_t$) [3–5]. In this case, the transitions are carried out in accordance with a certain probability matrix, described as

$$p_{ij} = p\left[q_t = S_j | q_{t-1} = S_i\right], \quad 1 \le i, j \le N. \tag{1}$$

In this case, $p_{ij}$ has the following properties:

$$p_{ij} \ge 0, \tag{2}$$

$$\sum_{j=1}^{N} p_{ij} = 1. \tag{3}$$

A Markov process is a stochastic process consisting of a sequence of random states, where the probability of transition from one state to another depends only on the current state and does not depend on previous states. The result of the observation of such a process is a sequence of states that the system goes through during the observation period [14].

Having a process model in the form of a probability matrix $A = p_{ij}$ and an initial state matrix $\Pi_i = p[q_i = S_i]$, $1 \le i \le N$, one can calculate the probability of any sequence of states, i.e. any observation [15].

In practice, the so-called hidden Markov models are most common. They are used to model sequences of observed data that are the result of some hidden process. In this case, the hidden process is modeled as a sequence of states that change over time in accordance with the probabilistic transitions between them. Each state of the hidden process generates some observation, which becomes visible to the observer. The probabilities of transitions between states and the probabilities of generating observations are determined by the parameters of the model, where they can be estimated from the observed data using training methods. In such models, the observed events are probabilistic functions of the real state of the system. In Hidden Markov Models, the observed events are called "observable variables" or "outputs", and the states of the system not directly observed are called "hidden variables" or "internal states". The output data is the result of some process, depending on the internal state of the system. The probability of observing each specific sequence of output data for a given sequence of internal states is determined using conditional probabilities [16].

Typically, Markov models can be described by the following sets of parameters [17]:

1) $N$ is the number of model states.
2) $M$ is the number of different observables in each state by the symbol (i.e. the size of the alphabet) $V = \{V_1, V_2, \ldots V_N\}$..
3) $A = \{p_{ij}\}$ - matrix of transition probabilities, where

$$p_{ij} = p\left[q_{t+1} = S_j | q_t = S_i\right], \quad 1 \le i, j \le N. \tag{4}$$

4) $G = \{g_j(k)\}$ - probability distributions of observed symbols in state $j$, where

$$g_j(k) = p[v_k | q_t = S_j], \quad 1 \le j \le N, \quad 1 \le k \le M. \tag{5}$$

5) $\Pi = \{\Pi_i\}$ is the probability of each initial state, where

$$\Pi_i = P[q_1 = S_i], \quad 1 \le i \le N. \tag{6}$$

By setting the values of the parameters $N$, $M$, $A$, $G$ and $\Pi$ for Markov models, it can be used as a sequence generator.

To generate a sequence with a given number of elements, you must specify the initial state of the system (that is, one of the possible states that the hidden process can take) and sequentially generate new states and corresponding observable events. This process is carried out in the following steps [18]:

1. Set the initial state of the system by selecting one of the possible states of the hidden process.
2. Using the matrix of transition probabilities $A$, select the next state of the system based on the current state.
3. Using the probability matrix $G$, generate an observable event corresponding to the current state.
4. Repeat steps 2 and 3 the specified number of times (until the required sequence length is reached).

Based on a given algorithm, it is possible to generate a sequence of observed events corresponding to a given statistical model. The resulting sequence is used to test speech recognition algorithms.

In addition, when using HMM for pattern recognition, it is necessary to solve the following three problems [19]:

1. **The task of training the model.** The HMM parameters are determined based on the training set. To determine the parameters, it is recommended to use the EM-algorithm (expectation-maximization algorithm).

1.1. Initialization of model parameters (transition_matrix, emission_matrix and initial_distribution). Model parameters in the context of HMM describe the probabilities of transitions between states and the probabilities of symbol emission in each state.

The transition matrix determines the transition probabilities between model states. If the HMM has $N$ states, then the transition matrix will be $NxN$ in size. Each element $(i, j)$ of the matrix represents the probability of transition from state $i$ to state $j$. It is important that the sum of the probabilities in each row (or column) of the transition matrix be equal to 1, since a state must always transition to one of the states (including itself).

The emission matrix determines the probabilities of observing symbols for each state. If the HMM has $N$ states and $M$ possible symbols (for example, in a speech recognition problem, $M$ can be equal to the number of different sounds), then the emission matrix will be $N \times M$ in size. Each element $(i, k)$ of the matrix represents the probability of observing character $k$ while in state $i$.

The initial distribution determines the probabilities of the initial states of the model. If the HMM has $N$ states, then the initial distribution will be a vector of size $N$. Each element $i$ of the vector represents the probability that the model will start in state $i$.

They must be normalized so that the sum of the rows in the matrices is 1.

1.2. E-step (Expectation step):

The Forward-Backward algorithm is implemented to calculate the posterior probabilities.

Forward algorithm:

Initialization of the alphas array with zero values.

Calculate the first element of alphas[0] as the product of initial_distribution and emission_matrix for the first observation.

For each next time step $t$:

alphas[$t$] is calculated as the product of the emission_matrix for the current observation and the scalar product of alphas[$t-1$] and transition_matrix.

Backward algorithm:

Initializing the betas array with units.

The last element of betas[num_observations - 1] is calculated to be 1.

For each time step $t$ from num_observations - 2 to 0:

Calculate betas[$t$] as the product of the emission_matrix for the next observation times the scalar product of betas[$t + 1$] and the transition_matrix.

Calculation of gamma and xi:

posterior probabilities of states (gamma) by normalizing the product of alphas and betas for each time step;

posterior transition probabilities (xi) for each time step using alphas, betas, transition_matrix and emission_matrix.

1.3. M-step (Maximization Step):

The model parameters are updated to maximize the expected likelihood:

Initial Distribution Update:

initial_distribution as average gamma values for the first time steps.

Transition matrix update:

each transition_matrix element as the sum of the xi values for the transition from state *i* to state *j* divided by the sum of the gamma values for state i (excluding the last time step).

Emissions matrix update:

each element of the emission_matrix as the sum of the gamma values for state *i* when the observation matches symbol *k*, divided by the sum of the gamma values for state *i*.

1.4. Repetition of E-step and M-step:

Steps E and M are repeated for a certain number of iterations or until parameter changes are negligible.

1.5. Model Rating:

After reaching the stopping criterion, evaluate the trained model parameters transition_matrix, emission_matrix and initial_distribution.

2. The problem of estimating the probability of observation. Let the HMM and the observed sequence be given. It is necessary to determine the probability of observing a given sequence for a given model. To solve this problem, you can use the Forward-Backward algorithm, which is carried out in the following steps:

2.1. Initialization:

Model parameters are set: transition_matrix, emission_matrix, and initial_distribution.

Assigning an observed sequence of characters to observations.

2.2. Forward pass:

The alphas array of size (num_observations, num_states) is initialized with zeros.

The first element of alphas[0] is computed as the product of initial_distribution and the corresponding emission probability from the emission_matrix for the first observation observations[0].

For each time step *t* from 1 to num_observations - 1:

For state *j*:

Alphas[*t,j*] is calculated as the sum of the products of alphas[*t-1, i*], transition_matrix[i, j] and emission_matrix[ *j*, observations[*t*]] for all states *i*.

2.3. Backward pass:

The betas array is initialized with a size of (num_observations, num_states) units.

Assigned to the last element betas[num_observations - 1] 1.

For each time step *t* from num_observations - 2 to 0:

For state *i*:

Betas[*t, i*] is calculated as the sum of the products of transition_matrix[*i, j*], emission_matrix[ *j*, observations[$t + 1$]] and betas[$t + 1, j$] for all states *j*.

2.4. Observation Probability Calculation:

The overall probability of the observation is calculated as the sum of the probabilities in the last time step alphas[−1].

2.5. Result output

Output the probability of observation obtained in the previous step.

It should be noted that the algorithm uses two passes: forward and reverse. The forward pass computes alphas (forward passes), which represent the probabilities of being in a particular state at each time step, given observations up to that point. The back pass computes betas (back passes), which represent the probabilities of remaining in the state after observing all subsequent symbols. After performing the backward pass, the total probability of the observation is calculated as the sum of the probabilities in the last time step alphas[-1]. This value represents the desired probability of observing a given sequence given a hidden Markov model.

3. The task of decoding. Let the HMM and the observed sequence be given. It is necessary to determine the most probable sequence of hidden states. To solve this problem, the Viterbi algorithm is used, which is carried out in the following steps:

3.1. Parameter initialization:

The number of hidden states (num_states) and the number of possible observations (num_symbols) are determined.

The transition matrix (transition_matrix) is initialized with the probabilities of transitions between hidden states, the emission matrix (emission_matrix) with the probabilities of generating observations from hidden states, and the initial distribution (initial_distribution) with the probabilities of initial states.

3.2. Viterbi algorithm:

Input data: a sequence of observations (observations).

The dp matrix is initialized to store the highest probabilities and the prev_state matrix to store the previous states and the first time step of the dp matrix with initial probabilities.

Based on time steps and states, the highest probabilities and previous states are calculated according to the Viterbi algorithm.

The path is decoded starting from the last time step based on the prev_state matrix.

3.3. Result output:

The decoded path of hidden states (decoded_states) obtained by the Viterbi algorithm is displayed.

When using Markov models, training with test sequences is time consuming. It should give a method for finding model parameters such as to maximize the probability of a sequence of observations matching a given model. Consider an iterative procedure for choosing model parameters [12].

In the procedure of iterative refinement of the parameters of Markov models, it is necessary to determine the probability of the system being in state $S_i$ at time $t$ and in state $S_j$ at time $t+1$ for given model parameters and a sequence of observations, i.e.

$$\xi_t(i,j) = p(q_t = S_i,\ \ q_{t+1} = S_j | O, \lambda) \tag{7}$$

In direct and inverse variables, the probability is defined as

$$\xi_t(i,j) = \frac{\alpha_t(i) p_{ij} g_j(O_{t+1}) \beta_{t+1}(j)}{p(O|\lambda)} = \frac{\alpha_t(i) p_{ij} g_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) p_{ij} g_j(O_{t+`1}) \beta_{t+1}(j)}. \tag{8}$$

Expressing $\gamma_t(i)$ in terms of summation over $\xi_t(i,j)$, we get:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(ij). \tag{10}$$

Based on the above formulas for refining Markov models, the following can be written:

Expected frequency of being in state $S_i$ at time $t = 1$:

$$\overline{\pi_i} = \gamma_1(i); \tag{11}$$

$$\overline{p_{ij}} = \frac{\sum\limits_{t=1}^{T-1} \xi_t(i,j)}{\sum\limits_{t=1}^{T-1} \gamma_t(i)}; \tag{12}$$

$$\sum_{t=1}^{T} \gamma_t(j), \quad \overline{g_j}(k) = \frac{O_t = v_k}{\sum\limits_{t=1}^{T} \gamma_t(j)} \tag{13}$$

Certain parameters are most likely to represent a given sequence of observations.

One of the main problems in HMM training is the problem of getting into the local optimum in the process of model optimization. This can cause the model optimization to stop at a loss function value that is not the global optimum, and therefore the model will not have the best recognition accuracy.

To overcome this problem, many approaches have been proposed so far. One is to use a stochastic optimization method such as Markov Chain Monte Carlo (MCMC) or Stochastic Gradient Descent (SGD). These methods make it possible to get out of the local optimum and provide a wider search for the parameter space.

In addition, global optimization methods such as genetic algorithms or ant colony based global optimization methods or other evolutionary algorithms can be used. Such methods also make it possible to avoid local optimal.

To do this, at the first stage of the evolutionary algorithm, an initial population of solutions is randomly generated, which are various combinations of model parameter values. Then the quality of each solution is evaluated using an evaluation function, which can be, for example, the likelihood function of the model. Next comes the selection of the most adapted solutions (the best). The crossover operator combines parts of two parent solutions to create new offspring, and the mutation operator randomly changes the parameter values in the new solution.

The evolutionary algorithm repeats the process of creating new generations of the population until a given stopping criterion is reached (for example, reaching the maximum value of the evaluation function or reaching a certain number of iterations).

The use of evolutionary algorithms for HMM training has a number of advantages, such as the possibility of finding a global optimum and reducing the dependence on starting parameters. And also, the evolutionary algorithm allows you to optimize not only the parameters of the model, but also the choice of training sequences, which can improve the quality of recognition.

## 2 Results

HMM-based speech recognition requires the formation of informative speech features. In literatures [20–31], interesting works have been carried out on the formation of features and the identification of informative features. To form features in this work, MFCC coefficients were used to represent speech in the form of a vector of features, which are obtained as a result of the Fourier transform of temporal speech signals presented in the form of a spectrogram into a Mel-frequency representation. MFCC coefficients take into account the peculiarities of the perception of sounds by the human ear, since the chalk-frequency representation better matches the perception of high and low frequencies. Next, the obtained coefficients are transformed using cepstral analysis, which helps to highlight important signal characteristics and reduce the dimensionality of the data.

The number of features (MFCC coefficients) depends on the specific task and may be different. Typically 12 to 20 MFCCs are used, although more may be used in some cases. The number of MFCC coefficients can be chosen experimentally based on the quality of speech recognition on test data.

The algorithm for extracting MFCC coefficients consists of the following steps:

The audio signal is split into overlapping frames.

Applies a window function (such as Hamming) to each frame.

The Fourier transform is applied to the windowed frame.

The magnitude spectrum is calculated for each frame.

Mel filters are calculated to convert the magnitude spectrum into a Mel spectrum.

Mel filters are applied to the magnitude spectrum.

The logarithm of the resulting Mel spectrum is calculated.

A discrete cosine transform (DCT) to the logarithm of the Mel spectrum is performed.

The output of this algorithm is the MFCC coefficients, which are represented as vectors for each frame.

[[0.7422303 0.5532607 0.68049966… 0.95503917 0.32147816]
[0.0786073 0.9745327 0.52009533 … 0.71336791 0.79554746]
[0.0726853 0.0393421 0.02926857 … 0.85601778 0.20057767]
[0.5490482 0.3676609 0.15398193 … 0.29524076 0.48544261]
[0.4670125 0.2525328 0.21170367 … 0.76408712 0.64643327]]

At the next stage of speech recognition, the creation of HMM and training is carried out. Below is the result of this step:

Trained Transition Matrix:
[[8.58531871e−01 1.41467622e−01 … 5.06242437e−07]
[4.01853960e−10 6.00927764e−01 … 3.99072235e−01]
[2.71153867e−01 1.12751763e−09 … 7.28846132e−01]]
Trained Emission Matrix:
[[3.74174969e−01 1.84769465e−01 … 4.15451791e−20 4.41055566e−01]
[5.09872773e−06 6.52057991e−01 … 2.85676830e−01 6.22600806e−02]
[1.64715419e−01 2.93611122e−01 … 3.31506967e−01 2.10166492e−01]]
Trained Initial Distribution:
[1.14160670e−96 1.00000000e+00 … 7.05398021e−48]

This step uses the EM (Expectation-Maximization) algorithm, which is a powerful and commonly used technique in machine learning. It has several advantages that make

it useful for solving various problems, since the EM algorithm allows you to work with data in which some information is missing or hidden (for example, observable and hidden variables). This can be useful when training models when certain variables cannot be directly observed. The EM-algorithm seeks to maximize the likelihood function, which allows obtaining the most probable model parameters for a given sample, can be used for data clustering, as well as for training models of a mixture of distributions, which makes it useful for analyzing unstructured data and is resistant to the presence of outliers in the data, since it is based on a probabilistic model and averages the influence of different observations. The algorithm works iteratively, improving the current solution at each iteration. This allows reaching local optima and achieving convergence. The EM algorithm can be applied even if there is not enough data to fully train the model, as it uses an E-step to estimate latent variables and an M-step to update the parameters.

However, it has some drawbacks: the EM algorithm is sensitive to initial approximations of the parameters. Different initial conditions can lead to different local optima. This can make it difficult to obtain globally optimal model parameters. The EM algorithm does not guarantee convergence to the global optimum. It can get stuck at local extremes even if there is a global optimum. At each iteration of the EM algorithm, it is required to calculate the latent variable expectations (step E) and update the model parameters (step M). This can be computationally expensive, especially for complex models or large amounts of data. The EM algorithm requires multiple repetitions of steps E and M until convergence. In some cases, convergence may take a long time or even not be achieved. The EM algorithm uses optimization methods that often require the likelihood function to be differentiable. This limits its applicability to models for which it is difficult to calculate derivatives. If the initial parameters of the model are poorly chosen or the model is too complex, the EM algorithm may converge to local extrema where the model likelihood is close to zero. The EM algorithm does not always work well if the data has a complex structure, for example, when there are outliers that are out of the general distribution.

The next step is to estimate the observation probability based on the Forward-Backward algorithm. This algorithm allows you to calculate the probability of observing a sequence of characters, as well as the distribution of hidden states in each time step. One of the key advantages of the algorithm is its ability to estimate the probability of observing a particular sequence of characters given a model. The algorithm can be used to decode hidden state sequences based on the observed sequence. This is important in tasks related to the recognition, decoding or classification of sequences. The forward-backward algorithm can be used to train HMM parameters based on observed data. It can help find optimal parameter values that maximize the likelihood of observing data.

The next step is decoding. Where it is required to determine the most probable sequence of hidden states. For this, the Viterbi algorithm is used - this is an efficient method for decoding hidden states in HMM. It has several advantages that make it an important tool in the analysis of data sequences. The Viterbi algorithm finds the optimal hidden state sequence that most likely resulted in the observed data sequence. This makes it indispensable in tasks where it is required to choose the best explanation for the observed data. It has linear complexity with respect to the length of the observed sequence and the number of states in the model. This allows fast and efficient decoding of large

amounts of data. The Viterbi algorithm can be used for various types of hidden Markov models, including discrete and continuous state models, as well as mixed models. The implementation of the Viterbi algorithm is relatively simple and can be done with a relatively small amount of code. This makes it available for practical implementation in various applications. The Viterbi algorithm works in a local context, choosing the optimal state based on local transition probabilities. This reduces the amount of computation and memory required for decoding.

Finally, a genetic algorithm can be used to optimize model parameters. There are various optimization methods that can be used to solve problems, including HMM learning problems.

Methods based on mathematical calculations, which use mathematical calculations to find the optimal solution. They can be directional (e.g. gradient descent) or non-directional (e.g. coordinate descent). Directional methods involve determining the direction of the function's steepest decay and then moving in that direction to find a local extremum. For example, the gradient descent method uses the gradient (derivative) of a function to determine the descending direction.

Enumeration methods are based on enumeration of all possible solutions in order to find the optimal one. These methods can be quite slow and inefficient for large search spaces, but they are guaranteed to find the optimal solution. An example would be the brute force method, when all possible combinations of parameters are checked.

Methods that use the element of randomness include a random element in the optimization process. An example would be simulated annealing, which starts with a large random variance and then reduces it gradually, allowing the system to "cool" and find an optimum over a narrower range.

When considering the problem of HMM learning, methods based on mathematical calculations can be problematic due to the complex structure of loss functions and differentiability constraints. Enumeration methods can be inefficient due to the high dimensionality of the search space. Methods that use randomness can be helpful in avoiding getting stuck in local optima.

The genetic algorithm can be used to optimize the number of hidden states in the SMM, which can improve the accuracy of speech recognition. The HMM training problem is a global optimization problem, since it is required to find the optimal model parameters that provide the best fit to the observed data. Genetic algorithms are capable of performing global searches and can help avoid getting stuck in local optima. The HMM training task may involve tuning a large number of parameters, such as initial state probabilities, transition matrices, and emission functions. Genetic algorithms are well suited for working with high-dimensional problems and combinatorial spaces. Genetic algorithms do not require differentiability of the objective function, which is an advantage, since the likelihood functions are distorted in the HMM learning problem and do not always have analytical derivatives. Genetic algorithms can be applied to various types of HMMs such as Hidden Markov Models with continuous or discrete states and various emission functions. Genetic algorithms can be easily adapted for parallel computing, which makes it possible to speed up the HMM learning process, and can be adapted to work with constraints on model parameters, which is important in the HMM learning task with limited resources. HMM training objectives can have a variety of structures

and requirements. Genetic algorithms can be applied to solve complex, unstructured problems where other optimization methods may be less effective.

The resulting GA solution: Trained Transition Matrix:
[[0, 96 0,04 … 0]
[0 0,92 … 0]
……..
[0 0 … 1]]

The disadvantages of existing HMM learning methods are their strong dependence on initial conditions and the inability to find a global extremum. There is no single best method for teaching HMM, and different approaches can be used for this task. One of such approaches considered in the work is genetic algorithms (GA). Genetic algorithms are optimization methods based on the principles of natural evolution. They are a promising and actively developed direction in the field of optimization of multicriteria problems.

## 3 Conclusions

The use of HMM for speech recognition is based on the following assumptions:

All speech can be represented as a sequence of sounds or phonemes. The probability of each phonemic state depends only on the previous state, not on earlier states. The sound wave of speech contains some level of noise and ambiguity, and HMM allows you to model this noise in order to improve recognition accuracy. HMMs use statistical methods to estimate model parameters based on a large amount of training data and determine the most likely sequence of states for a given speech wave. Building a model for speech recognition requires large amounts of computational resources and time for training, as well as parameter optimization to ensure maximum recognition accuracy.

The advantages of genetic algorithms in solving the HMM learning problem are that genetic algorithms work with several points in the search space at the same time. This avoids getting stuck in local optima and improves the probability of finding globally optimal HMM parameters. Genetic algorithms operate on coded representations of parameters, which can be chosen to better fit the characteristics of the problem. This gives greater flexibility and adaptability of the algorithm to a specific task and work with parameter codes regardless of their interpretation. This allows the algorithm to be applied to different types of parameters and tasks. Multipoint search reduces the chance of getting stuck in local optima. Genetic algorithms contribute to the exploration of the solution space, which is especially important for problems with many local extrema and work based only on information about the objective function at an arbitrary point and the range of acceptable parameter values. This simplifies the process of tuning the algorithm and its application to various problems.

## References

1. Ognev, I.V.: Preliminary processing of a speech signal for building a database of pronunciations of single words. In: Ognev, I.V., Paramonov, P.A. (eds.) Information tools and Technologies: tr. XX International Science-Technical Conference, pp. 53–58. MPEI, Moscow (2012)

2. Popov, E.V.: Communication with computers in natural language, 2nd edn. Stereotypical, 360 p. Editorial URSS, Moscow (2004)
3. Niyozmatova, N.A., Mamatov, N.S., Tulyaganova, Sh.A., Samijonov, A.N., Samijonov, B.N.: Methods for determining speech activity of uzbek speech in recognition systems. In: AIP Conference Proceedings, vol. 2789, no. 1, p. 050019 (2023). https://doi.org/10.1063/5.0145438
4. Mamatov, N.S., Niyozmatova, N.A., Yuldoshev, Y.S., Abdullaev, S.S., Samijonov, A.N.: Automatic speech recognition on the neutral network based on attention mechanism. In: Zaynidinov, H., Singh, M., Tiwary, U.S., Singh, D. (eds.) IHCI 2022. LNCS, vol. 13741, pp. 100–108. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-27199-1_11
5. Mamatov, N.S., Niyozmatova, N.A., Samijonov, A.N., Samijonov, B.N.: Construction of language models for Uzbek language. In: 2022 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, pp. 1–4 (2022). https://doi.org/10.1109/ICISCT55600.2022.10146788
6. Niyozmatova, N.A., Mamatov, N.S., Otaxonova, B.I., Samijonov, A.N., Erejepov, K.K.: Classification based on decision trees and neural networks. In: 2021 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, pp. 01–04 (2021). https://doi.org/10.1109/ICISCT52966.2021.9670345
7. Mamatov, N.S., Niyozmatova, N.A., Abdullaev, S.S., Samijonov, A.N., Erejepov, K.K.: Speech recognition based on transformer neural networks. In: 2021 International Conference on Information Science and Communications Technologies (ICISCT), Tashkent, Uzbekistan, 2021, pp. 1–5 (2021). https://doi.org/10.1109/ICISCT52966.2021.9670093
8. Mamatov, N., Niyozmatova, N., Samijonov, A.: Software for preprocessing voice signals. Int. J. Appl. Sci. Eng. **18**, 2020163 (2021). https://doi.org/10.6703/IJASE.202103_18(1).006
9. Narzillo, M., Abdurashid, S., Parakhat, N., Nilufar, N.: Automatic speaker identification by voice based on vector quantization method. Int. J. Innov. Technol. Explor. Eng. **8**(10), 2443–2445 (2019). https://doi.org/10.35940/ijitee.J9523.0881019
10. Wiedecke, B., Narzillo, M., Payazov, M., Abdurashid, S.: Acoustic signal analysis and identification. Int. J. Innov. Technol. Explor. Eng. **8**(10), 2440–2442 (2019). https://doi.org/10.35940/ijitee.J9522.0881019
11. Narzillo, M., Abdurashid, S., Parakhat, N., Nilufar, N.: Karakalpak speech recognition with CMU sphinx. Int. J. Innov. Technol. Explor. Eng. **8**(10), 2446–2448 (2019). https://doi.org/10.35940/ijitee.J9524.0881019
12. Mosleh, M.: FPGA implementation of a linear systolic array for speech recognition based on HMM. In: Mosleh, M., Setayeshi, S., Mehdi Lotfinejad, M., Mirshekari, A. (eds.) The 2nd International Conference on Computer and Automation Engineering (ICCAE), vol. 3, pp. 75–78 (2010)
13. Ikonin, S.Yu., Sarana, D.V.: SPIRIT ASP engine automatic speech recognition system. Digital Signal Processing (2003)
14. Sapunov, G.V., Trufanov, F.A.: Genetic algorithms as a method for optimizing hidden Markov models in problems of speech recognition. In: Information Technologies in Computer Systems. Issue 3. Under the general editorship of prof. Azarova V.N. MIEM, Moscow (2004)
15. Marczyk, A.: Genetic Algorithms and Evolutionary Computation (2004). http://www.talkorigins.org/faqs/genalg/genalg.html
16. Komarov, A.N.: Basic cellular ensembles of associative oscillatory environments and the possibility of their expansion. In: Komarov, A.N., Ognev, I.V., Podolin, P.B. (eds.) Computational systems and information processing technologies: Interuniversity. Sat. scientific tr. – Issue, vol. 5, no. 30, 200 p. Inf.-ed. center of PGU, Penza (2006)
17. Ognev, I.V.: Character recognition in an associative oscillatory environment. In: Ognev, I.V., Podolin, P.B. (eds.) News of Higher Educational Institutions. Volga region. Ser. Technical Science, no. 6, pp. 55–66 (2006)

18. Elliott, L., Ingham, D., Kyne, A., Mera, N., Pourkashanian, M., Whittaker, S.: Efficient clustering-based genetic algorithms in chemical kinetic modelling. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3103, pp. 932–944. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24855-2_106
19. Sastry, K., O'Reilly, U.M., Goldberg, D.E.: Population sizing for genetic programming based upon decision making. IlliGAL Report No. 2004028 (2004)
20. Samijonov, A., Mamatov, N., Niyozmatova, N.A., Yuldoshev, Y., Asraev, M.: Gradient method for determining non-informative features on the basis of a homogeneous criterion with a positive degree. In: IOP Conference Series: Materials Science and Engineering, vol. 919, no. 4 (2020). https://doi.org/10.1088/1757-899X/919/4/042011
21. Mamatov, N., Niyozmatova, N.A., Samijonov, A., Juraev, S., Abdullayeva, B.: The choice of informative features based on heterogeneous functionals. In: IOP Conference Series: Materials Science and Engineering, vol. 919, no. 4 (2020). https://doi.org/10.1088/1757-899X/919/4/042009
22. Mamatov, N.S., Samijonov, A.N., Yuldoshev, Y., Khusan, R.: Selection the informative features on the basis of interrelationship of features. In: Techno-Societal 2018 - Proceedings of the 2nd International Conference on Advanced Technologies for Societal Applications, vol. 2, pp. 121–129 (2020). https://doi.org/10.1007/978-3-030-16962-6_13
23. Fazilov, S., Mamatov, N., Samijonov, A., Abdullaev, S.: Reducing the dimensionality of feature space in pattern recognition tasks. J. Phys. Conf. Ser. **1441**(1), 012139 (2020). https://doi.org/10.1088/1742-6596/1441/1/012139
24. Mamatov, N., Samijonov, A., Niyozmatova, N.: Determination of non-informative features based on the analysis of their relationships. J. Phys. Conf. Ser. **1441**(1), 012149 (2020). https://doi.org/10.1088/1742-6596/1441/1/012149
25. Niyozmatova, N.A., Mamatov, N., Samijonov, A., Mamadalieva, N., Abdullayeva, B.M.: Unconditional discrete optimization of linear-fractional function "-1"-order. In: IOP Conference Series: Materials Science and Engineering, vol. 862, no. 4, p. 042028 (2020). https://doi.org/10.1088/1757-899X/862/4/042028
26. Niyozmatova, N.A., Mamatov, N., Samijonov, A., Rahmonov, E., Juraev, S.: Method for selecting informative and non-informative features. In: IOP Conference Series: Materials Science and Engineering, vol. 919, no. 4 (2020). https://doi.org/10.1088/1757-899X/919/4/042013
27. Fazilov, S., Mamatov, N.: Formation an informative description of recognizable objects. J. Phys.: Conf. Ser. **1210**(1) (2019). https://doi.org/10.1088/1742-6596/1210/1/012043
28. Mamatov, N., Samijonov, A., Yuldashev, Z.: Selection of features based on relationships. J. Phys. Conf. Ser. **1260**(10), 102008 (2019). https://doi.org/10.1088/1742-6596/1260/10/102008
29. Shavkat, F., Narzillo, M., Abdurashid, S.: Selection of significant features of objects in the classification data processing. Int. J. Recent Technol. Eng. **8**(2 Special Issue 11), 3790–3794 (2019). https://doi.org/10.35940/ijrte.B1494.0982S1119
30. Mamatov, N., Samijonov, A., Yuldashev, Z., Niyozmatova, N.: Discrete optimization of linear fractional functionals. In: 2019 15th International Asian School-Seminar Optimization Problems of Complex Systems, OPCS 2019, pp. 96–99 ((2019)). https://doi.org/10.1109/OPCS.2019.8880208
31. Shavkat, F., Narzillo, M., Nilufar, N.: Developing methods and algorithms for forming of informative features' space on the base K-types uniform criteria. Int. J. Recent Technol. Eng. **8**(2 Special Issue 11), 3784–3786 (2019). https://doi.org/10.35940/ijrte.B1492.0982S1119
32. Nagy, P., Németh, G.: Improving HMM speech synthesis of interrogative sentences by pitch track transformations. Speech Commun. **82C**(September 2016), 97–112 (2016). https://doi.org/10.1016/j.specom.2016.06.005

33. Daridi, F., Kharma, N., Salik, J.F.N.: Parameterless genetic algorithms: review and innovation. IEEE Can. Rev. (47) (2004)
34. Aida-ZadeK, R.: Investigation of combined use of MFCC and LPC features in speech recognition systems. Aida-Zade, K.R., Ardil, C., Rustamov, S.S. (eds.) World Acadamic of Science, Engineering and Technology (2006)
35. Noisy channel model (2020). https://en.wikipedia.org/wiki/Noisy_channel_model. Accessed 12 Apr 2020
36. Watanabe, S., et al.: Hybrid CTC. Attention Archit. End-to-End **11**(8), 1240–1253 (2017)
37. Hannun "Sequence Modeling with CTC". Distill (2017). https://distill.pub/2017/ctc/
38. Graves, A., Fernandez, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: ICML (2006)
39. Chan, W., et al.: Listen, attend and spell. arXiv:1508.01211 (2015)
40. Amodei, D., et al.: Deep Speech2: end-to-end speech recognition in English and Mandarin. arXiv:1512.02595 (2016)
41. Zeghidour, N., Usunier, N., Synnaeve, G., Collobert, R., Dupoux, E.: End-to-end speech recognition from the raw waveform. arXiv:1806.07098 (2018)