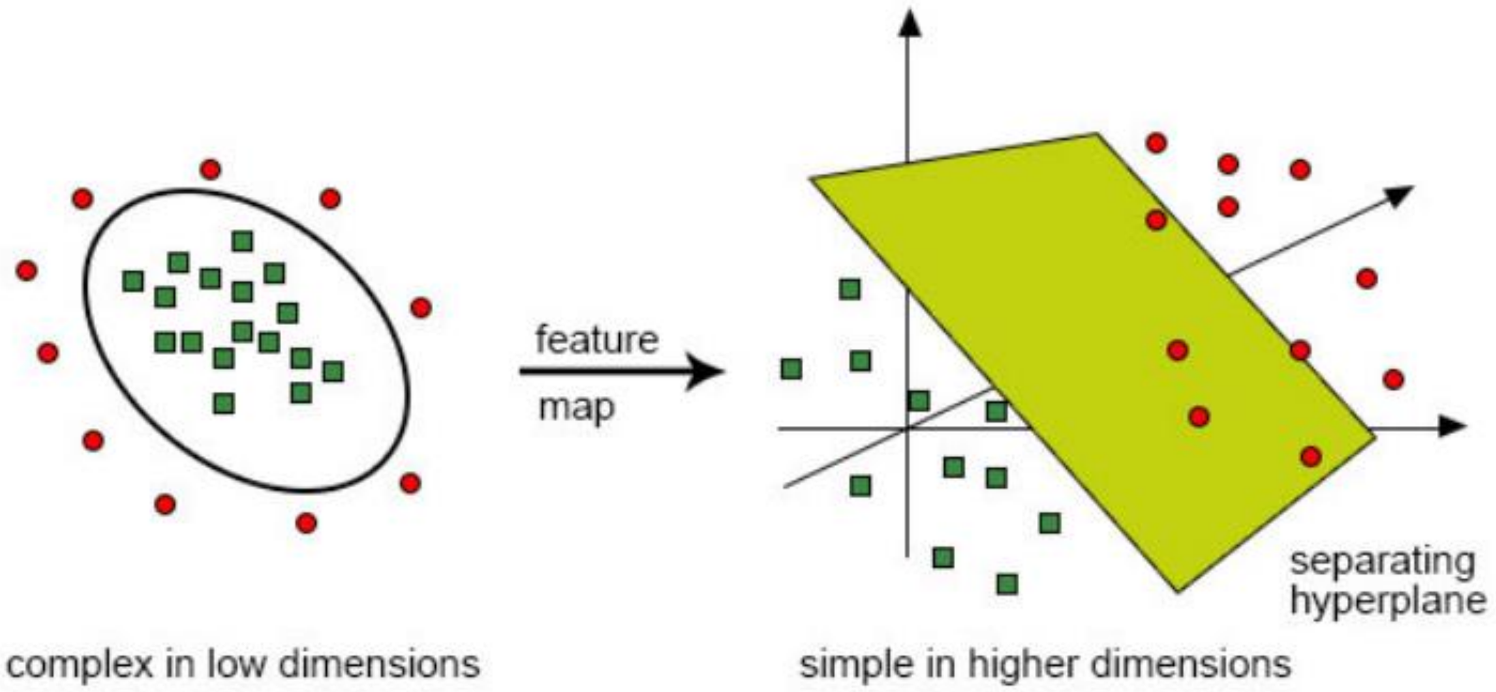
Determining the possibility of Kernel dominance

Contents:

Kernels, also known as kernel techniques or kernel functions, are a collection of distinct forms of pattern analysis algorithms, using a linear classifier, they solve an existing non-linear problem. SVM (Support Vector Machines) uses Kernels Methods in ML to solve classification and regression issues. The SVM (Support Vector Machine) employs “Kernel Trick” where data is processed, and an optimal boundary for the various outputs is determined.

In other words, a kernel is a term used to describe applying linear classifiers to non-linear problems by mapping non-linear data onto a higher-dimensional space without having to visit or understand that higher-dimensional region.

Separation may be easier in higher dimensions



Sourcre: eduCBA

What are the types of Kernel methods in SVM models?

Support vector machines use various kinds of kernel methods. Here are a few of them:

1. Linear Kernel

If there are two kernels named x1 and x2, the linear kernel can be defined by the dot product of the two vectors:

K(x1, x2) = x1 . x2

2. Polynomial Kernel

We can define a polynomial kernel with this equation:

K(x1, x2) = (x1 . x2 + 1)d

Here, x1 and x2 are vectors and d represents the degree of the polynomial.

3. Gaussian Kernel

The Gaussian kernel is an example of a radial basis function kernel. It can be represented with this equation:

k(xi, xj) = exp(-γ||xi - xj||2)

The given sigma has a vital role in the performance of the Gaussian kernel. It should be carefully tuned according to the problem, neither overestimated and nor underestimated.

4. Exponential Kernel

Exponential kernels are closely related to Gaussian kernels. These are also radial basis kernel functions. The difference between these two types of kernels is that the square of the norm is removed in Exponential kernels.

The function of an exponential function is:

k(x, y) =exp(-||x -y||22)

5. Laplacian Kernel

A Laplacian kernel is less prone to changes. It is equal to an exponential kernel.

The equation of a Laplacian kernel is

$k(x, y) = \exp(- ||x - y||)$

6. Hyperbolic or the Sigmoid Kernel

Hyperbolic or Sigmoid kernels are used in neural networks. These kernels use a bipolar sigmoid activation function.

The hyperbolic kernel can be represented with this equation:

$k(x, y) = \tanh(xTy + c)$

7. Anova radial basis kernel

This is another type of radial basis kernel function. Anova radial basis kernels work rather well in multidimensional regression problems.

An Anova radial basis kernel can be represented with this equation:

$k(x, y) = k=1n\exp(-(xk -yk)2)d$

Kernel-Density Estimation

A first problem occurs when estimating the probability distribution of some variable x. To obtain a first impression of the distribution, three histograms with different bin widths are plotted (Figure 1).

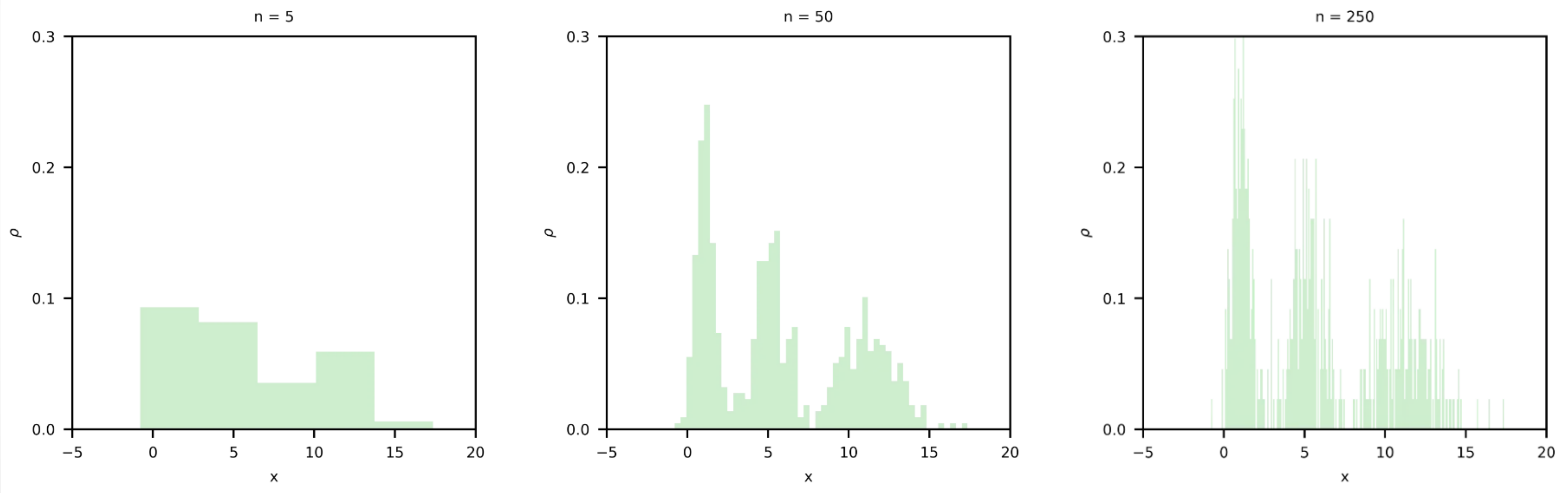
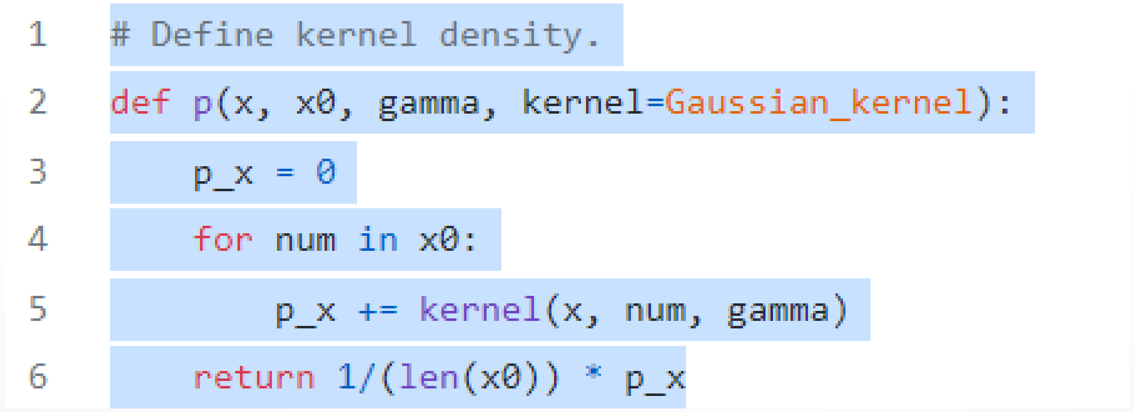


Figure 1: Empirical distribution of x with different bin widths. (source: author)

Depending on the bin width, the conclusion about the underlying distribution is completely different; it could be either uniform or multi-modal with three or even more modes. Therefore, it not clear which probability distribution to use to estimate the corresponding parameters (like mean and variance). Note that every observations adds some “amount” of ρ to the bin in which it lies. Consequently, the plot above can be constructed by a sum of the individual contributions at different levels of x. Let’s assume that the samples are independent and identically distributed. Thus, what one can do is to express the probability distribution as a superposition of “mini-densities” around data points x_i as a so-called kernel density (normalized by the amount of observations). Therefore, $p(x) = 1/n \Sigma K(x, x_i)$, with $K(x, y)$ being the kernel.



```python
# Define kernel density.
def p(x, x0, gamma, kernel=Gaussian_kernel):
    p_x = 0
    for num in x0:
        p_x += kernel(x, num, gamma)
    return 1/(len(x0)) * p_x
```

One possible kernel is the Gaussian kernel; the formula for the Gaussian kernel is given below. It is essentially a probability distribution centered around a point x_i.

```python
# Define Gaussian kernel.
def Gaussian_kernel(x, x0, gamma):
        return (1/np.sqrt(2*np.pi*gamma)) * np.exp(-(x - x0)**2/(2*gamma))
```

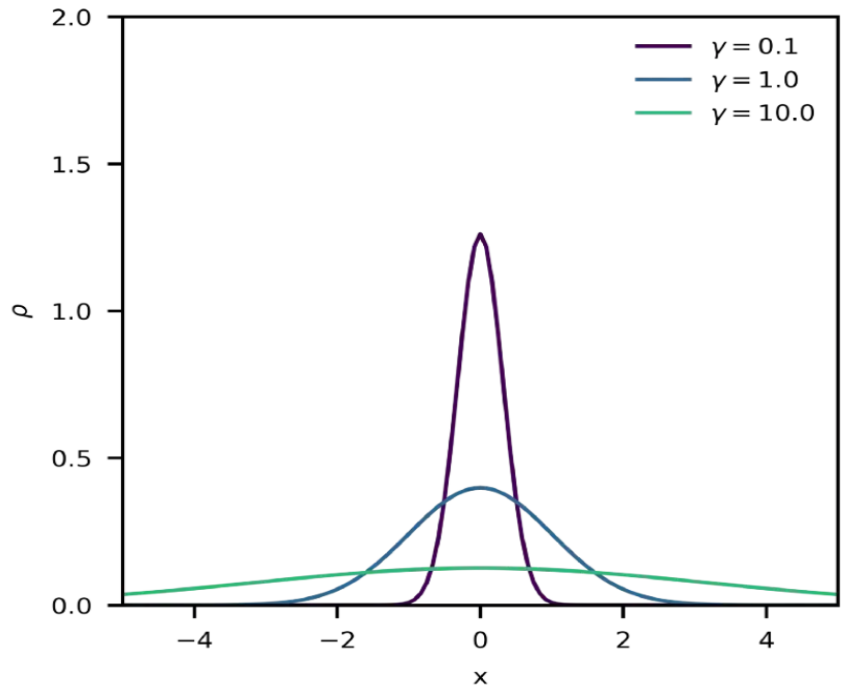


Figure 2: Gaussian kernel with different values for γ. (source: author)

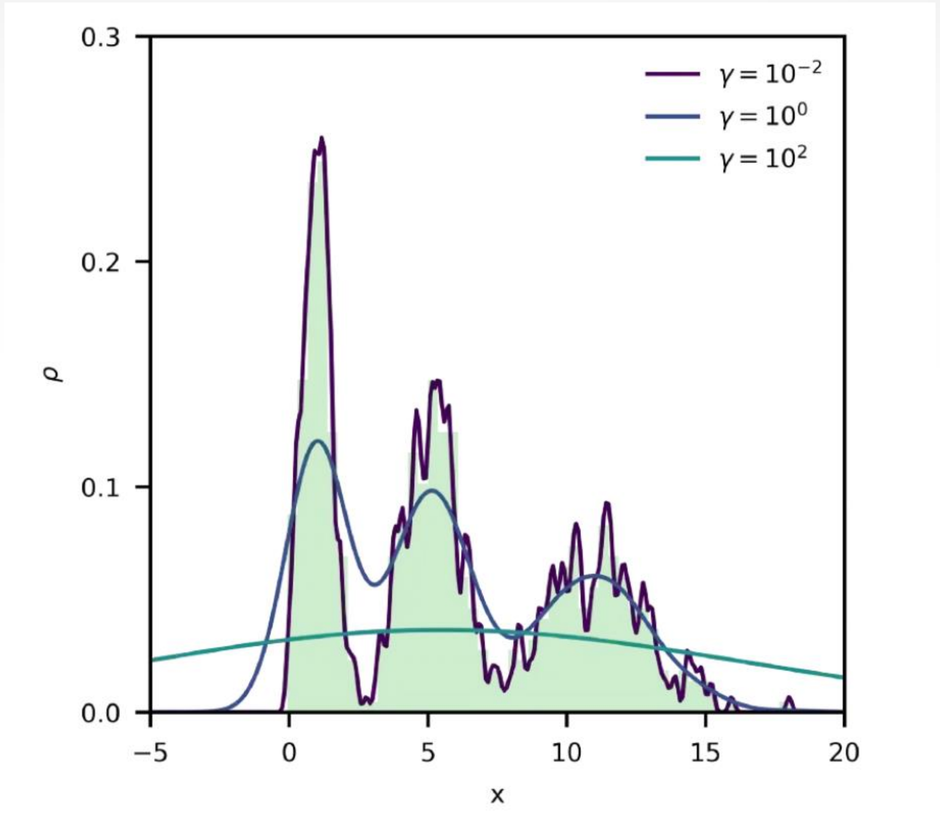
The Gaussian kernel is parametrized by a hyperparameter γ (Figure 2), which acts as variance and fixes the “concentration” of the density around a point x_i. There are also other kernels available (uniform, parabolic, cosine, etc.), but for now it is sufficient to work with this one.

Formally, a kernel is a non-negative function with two inputs x and y. The requirements for a kernel K(x, y) are that

it is symmetric, i.e., K(x, y) = K(y, x),

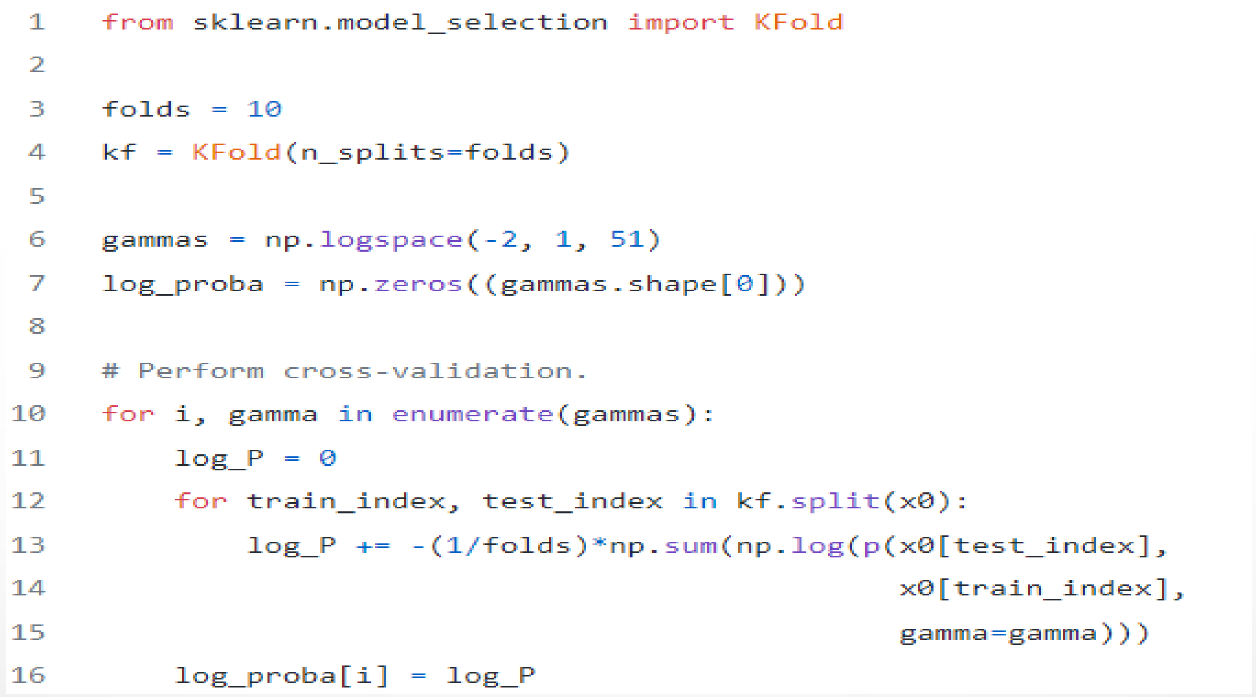
positive-semidefinite, i.e., finite matrices with pairwise evaluations should be free from negative eigenvalues,

and it should integrate to one.



One problem that arises is the choice of γ, whose value has to be fixed. Depending on the value of γ, the density can look quite different (Figure 3), just as with the bin width in Figure 1. For too large values, the model is too simplistic and important information goes missing (underfitting), while the opposite is true for too small values (overfitting). The main problem here is the “small” sample size, as in general the estimator is consistent and converges to the true density in the limit of infinitely many samples. While there are empirical formulas to choose γ, the easiest way is to perform cross-validation to find its optimal value.

One problem that arises is the choice of γ, whose value has to be fixed. Depending on the value of γ, the density can look quite different (Figure 3), just as with the bin width in Figure 1. For too large values, the model is too simplistic and important information goes missing (underfitting), while the opposite is true for too small values (overfitting). The main problem here is the “small” sample size, as in general the estimator is consistent and converges to the true density in the limit of infinitely many samples. While there are empirical formulas to choose γ, the easiest way is to perform cross-validation to find its optimal value.



```python
from sklearn.model_selection import KFold

folds = 10
kf = KFold(n_splits=folds)

gammas = np.logspace(-2, 1, 51)
log_proba = np.zeros((gammas.shape[0]))

# Perform cross-validation.
for i, gamma in enumerate(gammas):
    log_P = 0
    for train_index, test_index in kf.split(x0):
        log_P += -(1/folds)*np.sum(np.log(p(x0[test_index],
                                            x0[train_index],
                                            gamma=gamma)))
    log_proba[i] = log_P
```

More precisely, the value of γ that maximizes the probability of the data in the validation set is considered optimal. Figure 4 illustrates the negative logarithmic likelihood as a function of γ as well as the density with the optimal parameter, which seems to fit quite well.

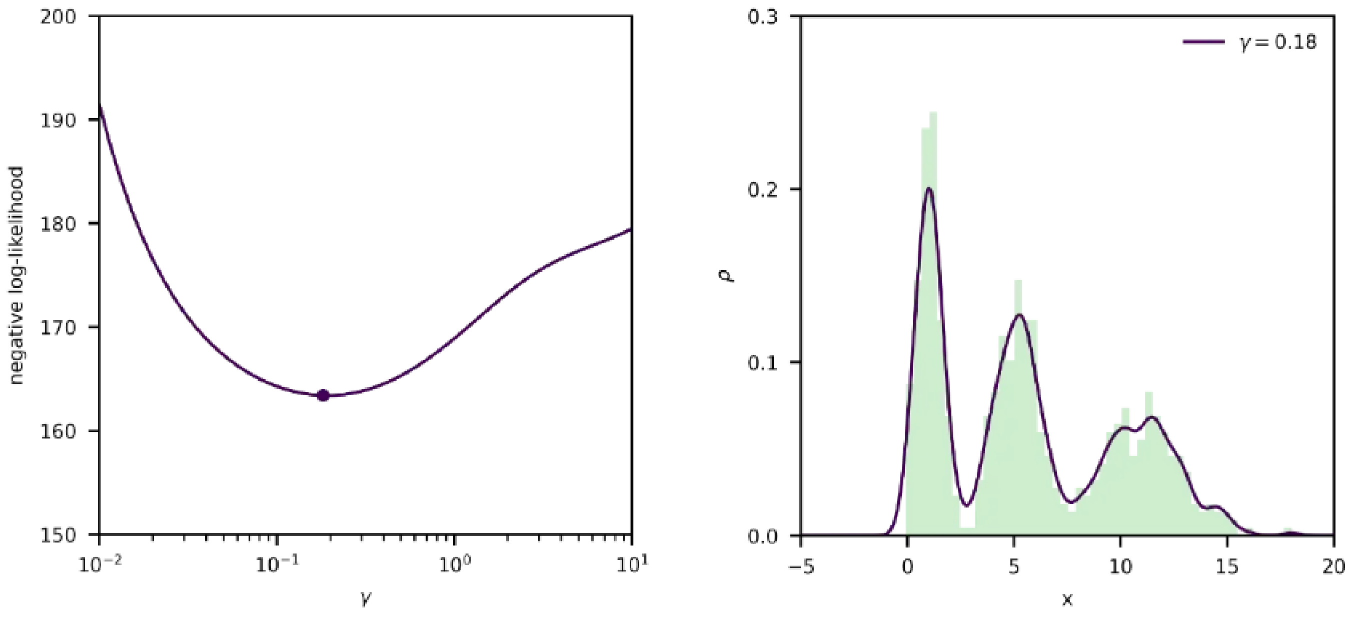
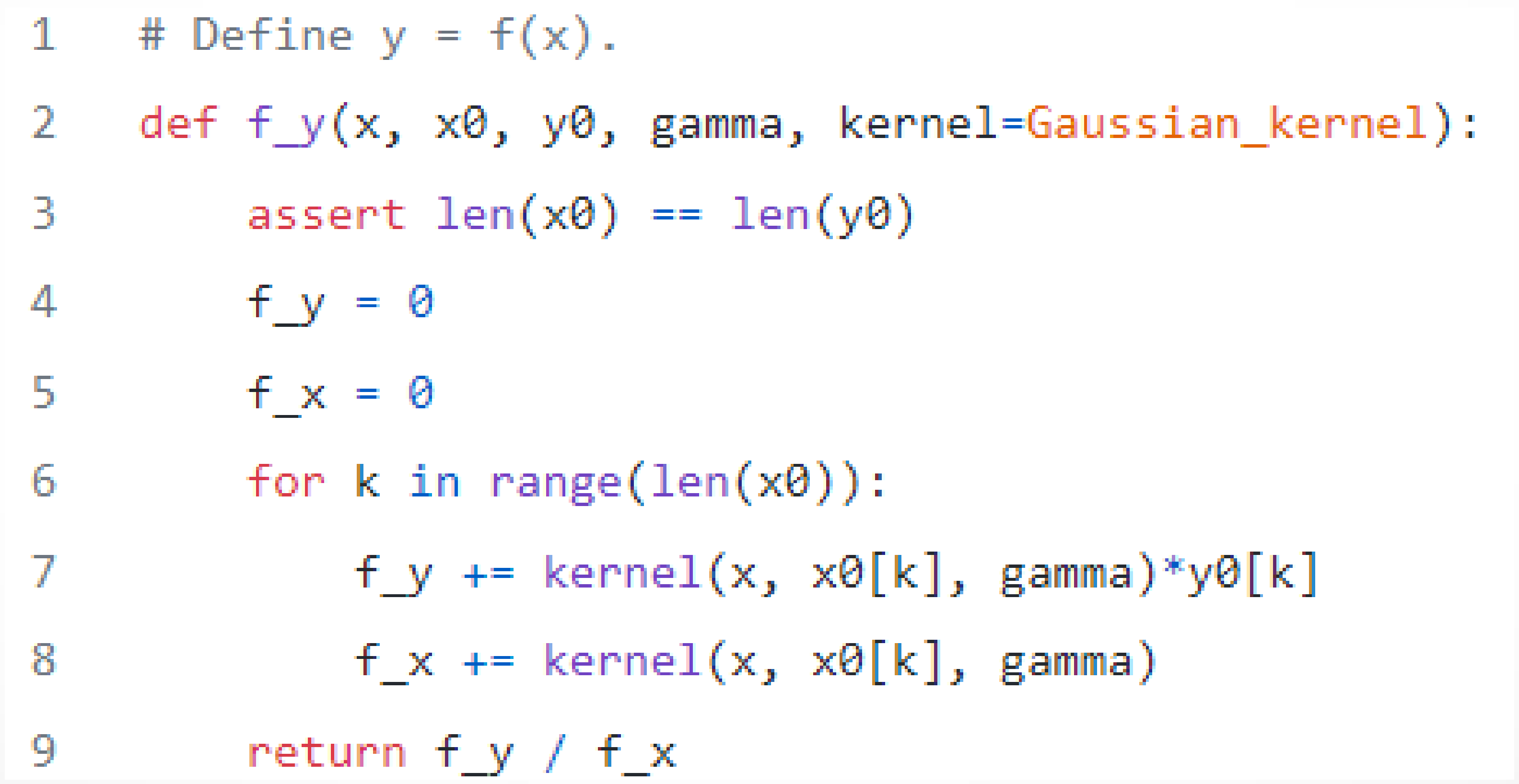


Figure 4: Negative log-likelihood and kernel density with optimal value for γ. (source: author)

Non-Parametric Regression

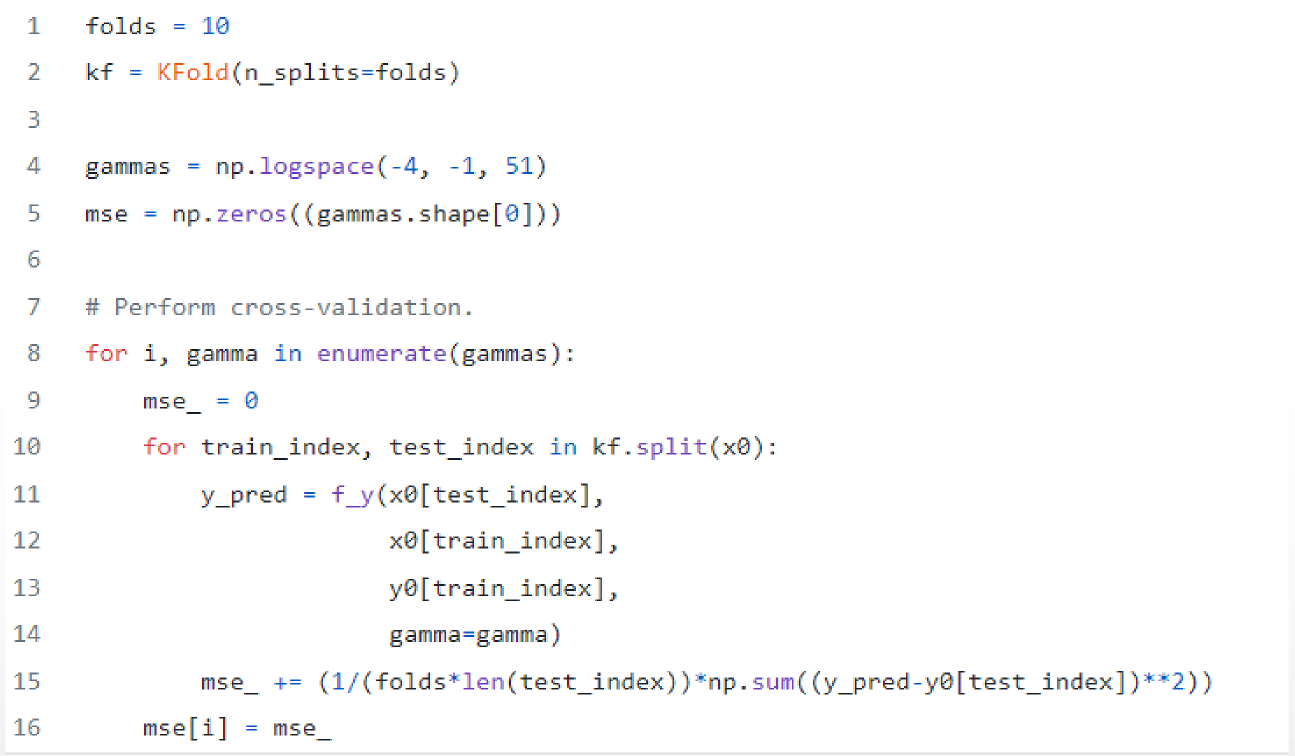
Because regression and classification problems can be derived from probability distributions, kernels can also be used to perform regression and classification. A regression line is the expectation E of the conditional distribution $p(y|x)$, i.e., $f(x) = E[p(y|x)]$. Moreover, it holds that $p(y|x) = p(x, y)/p(x)$. Hence, $E[p(y|x)] = \int y[p(x, y)/p(x)]dy$.

This gives rise to the following representation of f(x), known as Nadaraya-Watson kernel estimator.



```python
# Define y = f(x).
def f_y(x, x0, y0, gamma, kernel=Gaussian_kernel):
    assert len(x0) == len(y0)
    f_y = 0
    f_x = 0
    for k in range(len(x0)):
        f_y += kernel(x, x0[k], gamma)*y0[k]
        f_x += kernel(x, x0[k], gamma)
    return f_y / f_x
```

A prediction can be seen as a weighted average, whereas the weights are the relative positions of the samples in the training data to a new point. Consequently, the kernel serves as a similarity metric. Furthermore, the same problem as above arises, since the hyperparameter γ has to be “tuned”. This time, the metric to optimize will be the mean squared error.



```python
folds = 10
kf = KFold(n_splits=folds)

gammas = np.logspace(-4, -1, 51)
mse = np.zeros((gammas.shape[0]))

# Perform cross-validation.
for i, gamma in enumerate(gammas):
    mse_ = 0
    for train_index, test_index in kf.split(x0):
        y_pred = f_y(x0[test_index],
                     x0[train_index],
                     y0[train_index],
                     gamma=gamma)
        mse_ += (1/(folds*len(test_index))*np.sum((y_pred-y0[test_index])**2))
    mse[i] = mse_
```

The optimal value for γ is the one that minimizes the mean squared error. In Figure 5, the left-most plot shows that for high values, the function is too simplistic (underfitting), whereas for low values it becomes somewhat noisy (overfitting). The optimal value appears to be between 0.01 and 0.001, and the right-most plot shows the optimal regression line for this problem as found by cross-validation. Note that the minimum value of the mean squared error corresponds to the irreducible error (noise) in the data.

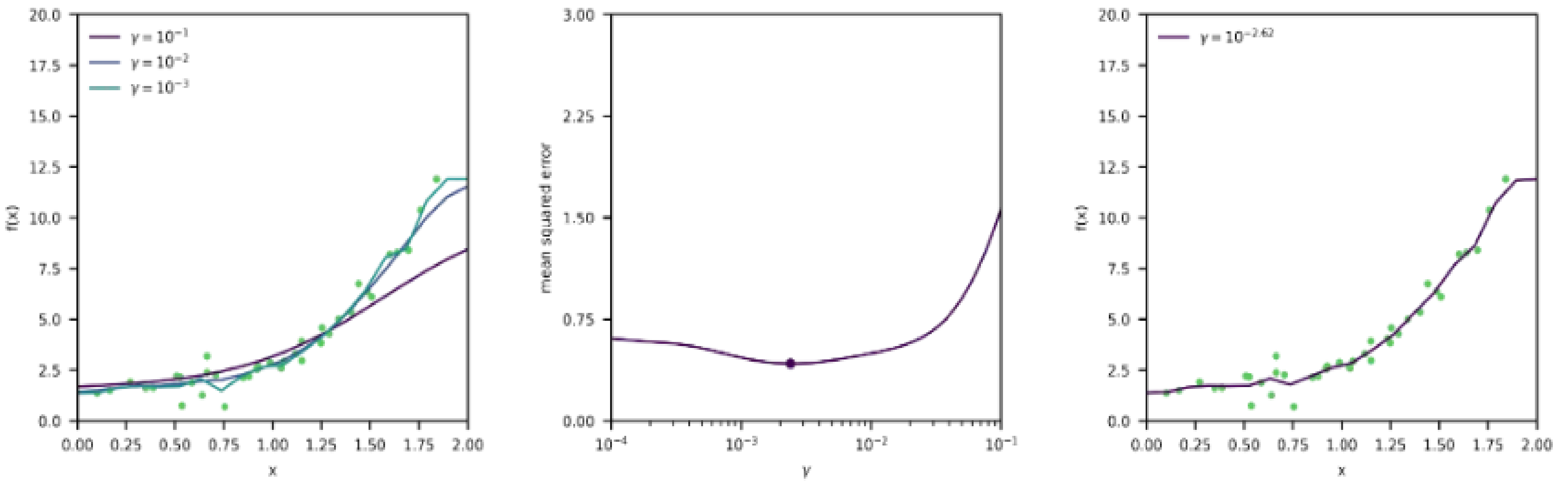


Figure 5: Non-parametric regression. (source: author)

Conclusion

Summing up, kernel methods are an alternative and easy way of estimating functions. However, such non-parametric techniques bear also some disadvantages. For instance, the computations shift from training to evaluation (lazy learning), as training is almost free (besides hyperparameter estimation of which there is only one). On the other hand, they cannot extrapolate outside the training data, which is visible in the right-most plot in Figure 5; the curve flattens for high values of x, which intuitively appears to be “wrong”. Finally, the underlying mechanisms are not discovered, i.e., the model is not really interpretable unless some model inspection technique is applied. For parameteric models, the magnitudes of the parameters indicate the relevance of a variable. Nonetheless, if the only goal is prediction, kernel methods are definitely an alternative to classical approaches.