Slide 1

Hello everyone and welcome to the fourth lecture of the machine learning course. My name is Aleksei Platonov and today we’ll talk about linear models of machine learning for solving regression and classification problems. On the one hand, linear models are very simple models, but in reality they are fraught with great flexibility.

Slide 2

Here is the plan for today's lesson. We will start by understanding what is the basis of linear models and what is the basic idea of a linear model. Then we move on to the most powerful machine learning tool called gradient descent optimization. We will also talk about how linear models solve the classification problems and the regression problems. Finally, we will see how linear models and the gradient descent method can be improved to increase the flexibility and quality of the model.

Slide 3

So, at first glance, linear models are the simplest mathematical models for solving some very simple problems. But the apparent simplicity contains a large number of possibilities. We examined two basic approaches to machine learning. The first is the k-nearest neighbor method. It is based on the idea that we can make a decision on how to classify an object or calculate a function based on which objects are closest to the feature vector with an unknown answer. The intuition in this model is simple and clear, everyone can agree with the case-law logic of decision making. For example, lawyers and doctors work like this. This is what we often do in life. The second machine learning paradigm we examined is the use of logical rules. It seems to us that a person thinks using logic in his reasoning and in solving problems, which means we can try to find logical rules and combine them in some way so as to simulate the process of human reasoning. One of the methods of greedy search for such rules is decision trees. But what about linear models? Linear models are based on the idea that to solve a problem, you can choose a mathematical model, a formula that will solve the problem. Once the formula is obtained, we can apply to it the full power of mathematical analysis to solve the problem. In this sense, linear models certainly outperform heuristic models. But nevertheless, our intuition works worse when trying to understand such a model. Let's move on to the linear model for regression and classification.

Slide 4

So let me remind you that when solving the regression problem, we have a learning set. The training set consists of pairs of objects and answers. Each object is described by a vector of values ​​of the feature functions. In the problem of regression learning, the answer is some numerical value, in the general case it is continuous. We need to find an approximation of the function that connects the feature vectors with the answers for them. The slide shows a linear regression model. As you can see, this model is nothing more than a generalization of the plane equation to multidimensional spaces. The main idea of ​​this model is a simple summation of all the features of the model, each of which is multiplied by some weight. An additional coefficient w0 is also introduced, which allows shifting the model response. It is for this reason that the model is called linear. Such a model cannot describe anything but the equation of a straight line in multidimensional space.

Slide 5

On the slide you can see an example of the result of training a linear regression model. The original data source is the equation of the line with the addition of a little noise. The result of training a linear model in two-dimensional space is the equation of the line. In solving this problem, the algorithm strives to draw a straight line so that the points of the training set are as far away as possible from the straight line. But what happens if we try to train a linear regression model on data that does not fall on a straight line?

Slide 6

It turns out nonsense. Indeed, if you try to explicitly use a simple linear model on such complex data, then nothing will work out and the model will not be able to predict anything useful. In this sense, linear models are really quite primitive. But later we will see how to solve such a problem.

Slide 7

Okay, it seems that to solve the regression problem, albeit in a primitive form, the linear model fits well. But how is the classification problem solved in such models? It turns out that we can apply the same formula as for linear regression, but we need to slightly modify it. Note that now the result of summing characteristics with weights is processed by the sign (x) function. This function returns 1 if the number passed to it is greater than zero and -1 otherwise. By the way, pay attention to the fact that summing a feature vector with weights is nothing more than a scalar product of vectors. If, at the same time, we add the attribute f0, which is always equal to -1, then the weight w0 can be taken into account in the vector of weights and then writing the linear model becomes much simpler. But how does it work? Why does the sign function make a classifier out of regression? Let's look at an example below.

Slide 8

In this example, the training set consists of two classes, the elements of which are grouped into two sets of points. What does a linear classification model do? It turns out that using the sign function allows you to determine whether the point X is above or below the line defined by the equation of the line in multidimensional space. In the example above, all points above the red line will receive a +1 response. And all the points under the red line will receive a -1 answer. It turns out that the linear classification model builds a line that divides several sets of points among themselves. That is what we see in the image. But then again, the line cannot pass between the three classes or between several sets of points of different classes.

Slide 9

And this we can see on this slide. But before moving on to improving the linear model, let's see how to train the simplest version of the linear model in the training set.

Slide 10

Before writing an expression, optimizing which you can get the values ​​of the parameters of the linear model, let's recall the concept of margin for the classifier. For the first time, we came across the concept of margin when considering the k-nearest-neighbor model. It turns out that the concept of margin applies to linear models. Moreover, in linear models this concept is introduced in a much more natural way. On the slide you can see the formula for calculating margin. Note that if class labels can only be -1 and +1, then the margin formula becomes very simple. We simply multiply the result of the scalar product of the vector of weights and attributes by the class label. If the class label and the sign of the result of the scalar product coincide, then the margin will be greater than zero. If the margin value is negative, then this means that the classifier was wrong. The greater the positive margin, the better and vice versa. Small margin values ​​may indicate that the classifier is unsure of the answer and may be mistaken in the neighborhood of this point.

Slide 11

The introduced concept of margin on the previous slide allows you to write an expression for optimization, the error function. In the basic version, the error function is just a counter of the number of negative indents. But there is a drawback to this feature. The indentation counter is not a continuous function and operations such as differentiation cannot be performed with it. This operation will be useful to us further. What to do? We must try to replace the counter function with something simpler and at the same time allowing differentiation to be performed. In the very first lesson, we talked about loss functions. Today we will finally use it explicitly! Loss function is a non-increasing function, which should be proportional to the error counter. Also, this is very important, the loss function must be differentiable. If the loss function is greater than or equal in value to the counter, then by minimizing the error function with the loss function, we will minimize the original error function with the counter.

Slide 12

What can the loss functions be? On the slide you can see examples of loss functions that meet the set requirements. All of them can be differentiated. In most cases, all functions decrease. If the margin value is negative, the functions take large values, which indicates a larger error. Interestingly, each loss allows you to create a new classification or regression algorithm. Generally speaking, regression from classification in a linear model differs only in the loss function. Later we will see how this works for regression, and now we continue to study classifiers based on a linear model.

It turns out that the linear classifier is fraught with not only the idea of ​​the simplest model possible. Linear classifiers came to us from research in the field of neurophysiology.

Slide 13

In the middle of the last century, biologists and mathematicians have just begun to study the structure of the human brain. They quickly found out that the brain was made up of neurons. A neuron is a cell of nerve tissue that connects to other cells with the help of a dendride, has its own nucleus, and an exit in the form of a long bond - an axon. The axon of such a cell can connect to another cell in the brain or human body. Scientists have learned that the brain internally transmits information using electrical impulses. The impulse through the dendrides enters the nucleus, accumulates, and if it exceeds a certain threshold value, the neuron generates its impulse and transfers it to the axon. In this case, impulses can be amplified or attenuated by the cell using synapses. Does this description not resemble anything?

Slide 14

When neuroscientists developed a mathematical model of a neuron, they got something similar to the figure on the slide. A neuron is an adder that has several inputs and one output. In this case, the value of the inputs are summed with some weight. To simulate the reaction of a neuron on the magnitude of the charge, a threshold function of neuron activation is introduced. This is a non-linear function that works as the Heaviside function, that is, on a certain interval of values, it returns values ​​close to zero, and on another interval returns values ​​close to one. But this is the linear model that we just looked at! It turns out that a linear model of classification and regression came to us from brain research. Thus, we get acquainted with the third concept of machine learning. The concept can be stated like this: "If we want to solve complex intellectual problems on a computer, then we should try to imitate the work of the human brain." A linear model is a neuron model that underlies neural networks. You will pass this type of models in the course “Artificial Intelligence”.

When scientists studied the process of learning the model, they quickly realized that if a neuron is mistaken in generating a pulse, then part of its weights - synapses is weakened. If the model generates an impulse at the right moment, then the synaptic connections are amplified. When mathematicians saw this study, they quickly realized that it was nothing more than a gradient method of numerical optimization.

Slide 15

What is a gradient optimization method? To answer this question, you first need to understand what a function gradient is. A function gradient is a vector of partial derivatives of a function with respect to all its parameters. So, for example, if a function depends on three variables, then its gradient will be a vector of three functions, where each function is a partial derivative with respect to the corresponding parameter of the function. The slide shows some examples of the results of applying the gradient. In the first case, we see a function of two arguments - x1 and x2. The gradient vector will consist of two functions. If we take the partial derivatives with respect to x1 and x2, we get a vector of functions, which you can see to the right of the function. Below is an example of a function with a large number of parameters and its gradient vector. If you want to better understand how it works or don’t remember, then it is better to recall the course of mathematical analysis. What happens if we substitute some numerical value into each function in the gradient? It turns out we get a vector that will indicate the direction of maximum growth of the function for which the gradient was calculated. The slide shows a few examples. The red arrows are the direction vectors of the growth of the function. At each point on the chart, its direction. Gradient allows you to mathematically write and calculate such vectors.

Slide 16

Thus, we can use the gradient to solve the optimization problem that we posed at the very beginning of the lecture. Let me remind you that one of the requirements for a loss function was the ability to differentiate it. This was needed just for calculating the gradient. But the original task was to minimize the error function, and the gradient shows the direction of increasing the value of the function. The anti-gradient will help us solve this problem. An anti-gradient is a vector that is pointing in the opposite direction from the gradient vector. That is, this is the same gradient vector that was taken with a minus sign. Once we have learned to read the gradient vector, we can minimize the error function. To do this, there is an algorithm called "gradient descent." In this algorithm, we gradually change the vector of the function parameters, subtracting the gradient at the current point from it with some scaling factor. The sign of the gradient with the letter "w" means that the gradient is calculated according to the parameters of the vector of weights "w". As soon as the value of the vector of the model parameters by the gradient is calculated, we immediately use it to recalculate the gradient at a new point and again update the parameter vector. And so we continue until the vector of parameters stops updating. All this allows us to build an algorithm for minimizing the error function.

Slide 17

However, calculating the full gradient can be expensive. Recall that the error function is the sum of the errors for each object of the training set. If the training set is very large, for example, several million elements, then calculating the gradient can be a very expensive operation. Fortunately, a gradient is a linear operation and the sum gradient is equal to the sum of the gradients. This means that we can calculate each component of the gradient of the error function independently of the other parts. Moreover, when the parts began to be calculated independently from each other, we may not use all the elements of the training set to calculate the full value of the gradient. We can roughly calculate the gradient with just a few examples. And we can use only one example from the training sample. This is the main idea of ​​the stochastic gradient descent method.

Slide 18

The slide shows the pseudo-code of the stochastic gradient algorithm. This algorithm uses only one element from the training set to calculate the vector of weights at the current iteration of the algorithm. At the very beginning, we must generate a vector of weights. Later we will talk about how this could be done. Next, we must calculate the value of the error per sample. After that, a cycle is started that runs until the error function stops changing dramatically. To update the vector of weights, we randomly select an object from the training set in line 7, consider the loss function for it, calculate the gradient of the loss function, and use it to update the vector of weights. And then, in order not to consider the error function again over the entire sample, we approximate the new value by the loss function calculated earlier. The gradient descent algorithm takes several parameters unknown to us. The first parameter is “etha” which is the gradient factor. This is the so-called learning rate. This value indicates how fast gradient descent should be performed. A very large value of this parameter can lead to the fact that we cannot find the minimum of the function. A very small value will cause the algorithm to look for a minimum of the error function for a very long time. The “lambda” parameter allows us to regulate how carefully we should evaluate the error function by only one object of the training set. The larger the lambda value, the stronger the algorithm relies on calculating the error only in the current object.

Slide 19

It is very important for machine learning algorithms that work using the gradient descent method, it is very important to follow the learning curves. We talked about this idea in the first lecture. Always try to monitor how the quality assessment of the algorithm changes during the training process, and if training ceases to affect quality or quality starts to deteriorate in a test sample, then this is an occasion to stop the algorithm based on gradient descent. This is a way to detect overfitting. After that, it is worth evaluating it from another sample and decide to use it, or you need to change some model training parameters.

Slide 20

Let us now consider some special cases of training linear models for specific loss functions. On the slide you can see the quadratic loss function. Such a function corresponds to the regression problem. This is one of the first mathematical models of neuron training. It is called ADALINE. Below is a function for iteratively calculating the vector of weights of a function for learning by the stochastic gradient method. Pay attention to how simple this function is. Such an iterative process is very easy to program, it does not require large computational costs and storing a large amount of data in memory.

Slide 21

The next slide shows another loss function. Now this function corresponds to the solution of the classification problem. Note that the loss function in the case of the classifier clearly depends on the margin, as it was before. What is logical is that the error function is zero on all margins greater than zero, that is, such a function does not fine for the correct classification at all. This function is called Hebb Rule and was also developed at a time when mathematical models of neurons were just beginning to be developed. Our approach, however, is that we introduce some loss function that shows the magnitude of the error on the object of the training set and explicitly minimize the error function through the use of the loss function. In the past, training algorithms for such models were built on the basis of empirical considerations.

Slide 22

Interestingly, a theorem was proved for a simple classifier model based on Hebb Rule. It is called the Novikov theorem. Its essence lies in the fact that if the binary classification poses the task of constructing a separating hyperplane, and this is what we do in linear models, and the data is distributed so that such a hyperplane can be drawn between them, then the simplest Hebb Rule allows us to solve this problem. Moreover, the theorem guarantees that the problem will be solved as accurately as possible and at the same time regardless of the values ​​of the learning rate, parameter lambda and the initial weight vector w0.

Do you remember the picture before where we saw two disjoint sets of objects? It is in such cases that the Hebb Rule works perfectly.

Slide 23

So let's move on to setting up the gradient descent algorithm. How to choose the initial vector w0? One possible solution is to simply select a vector that consists of zeros only. However, in some cases this is a bad decision. There is the so-called gradient paralysis problem. It is rare in simple linear models, but is a common problem, for example, for neural networks. Its essence is that at some point the gradient begins to take on very small values ​​and the weights of the function practically stop changing. As a result, the algorithm cannot learn anything. If you always initialize the weights at one point, then you can once observe such an effect. Another way to initialize the weights is a random value in the interval. To estimate the size of the interval, you can use the number n equal to the number of features in the task. In general, this will work better than the zero vector, but still it will not solve the problem of gradient paralysis. For the regression problem, you can use the third expression. A feature vector is a vector made up of all the values ​​of the feature of the feature of all objects in the training set. Let me remind you that the scalar product of vectors is written in triangular brackets. Such an approximation of the weight vector is the result of an analytical solution of the regression problem for a linear model and for most practical problems it will be an excellent approximation. The fourth way to initialize the vector of weights is to pre-train a small model on random data from the training set and then start training. Finally, the last option, the most correct one, is to select several random weight vectors at the beginning of the training algorithm and at the end to select the best point where there was no problem of gradient paralysis and where the highest quality model is observed.

Slide 24

In the basic algorithm of stochastic gradient descent, the order of selecting objects from the training set is not specified. One way is to just randomly take objects regardless of their class or the value of the objective function. But if the model is no longer mistaken in some class, then why should it present such objects? One option is to select the next object from the training set, taking into account the probability of taking this element. This probability should be proportional to the magnitude of the error. The more the classifier or regressor is wrong, the more likely it will be used in training. To evaluate this probability, it is very convenient to use margin. Finally, another option is to not present the model with examples on which it is not at all mistaken at all, and also to discard from training such examples on which the model is mistaken too much - such objects can be outliers.

Slide 25

The last question when setting up the stochastic gradient algorithm is how to set up the learning rate? If the error function were convex, then the problem would be solved very simply, because there are theorems that tell us that in this case it is enough to take the value inversely proportional to the step number in the training. The problem is that often the error function is non-convex. So you need to choose something else. One of the options is adaptive gradient descent, which at each step of solving the problem optimizes the value of the learning rate parameter. There are many methods of one-dimensional optimization, for example, the halving method or the Newton method. These are good algorithms, but they require additional computing overhead. In practice, it often turns out that you can simply randomly select the step values ​​at the beginning of the algorithm and, after the algorithm finishes, select the best solution according to the error function.

Slide 26

Let's look at several modifications of the basic idea of ​​linear models. The first one is called Logistic Regression. Please do not confuse with the basic task of regression. Logistic regression is used to solve the classification problem. That is its essence. Let me remind you that when classifying linear models in the basic form, the sign function is used. This function returns +1 if its parameter has a positive value and vice versa. But what if we try to approximate the sign function itself, instead of using margin as a way to make the error function differentiable? It turns out that this will lead us to the so-called sigmoid. This function is shown on the slide. In this case, the shape of the error function is somewhat modified. It has become somewhat more complicated, but it can still be differentiated. The gradient of this function is shown below. If we train the classifier based on logistic regression, we find an interesting fact. It turns out that if we apply the sigmoid to the scalar product of the vector of weight and features, then we can get the probability that the classified object belongs to the corresponding class. This means that we have connected linear models and probabilistic! This is a really important and useful fact, which helps a lot in many tasks in which you need to perform, for example, a risk assessment. In one of the following lectures we will talk about how probability theory is used in machine learning, but now we have a useful result.

Slide 27

The next feature that concerns linear models is that when solving the classification problem, such models can only be binary classifiers. Unlike previous machine learning paradigms, here we need to somehow solve this issue. A simple and effective solution is to apply the Ove-Vs-All principle. What does it consist of? His main idea is to train K different models for the multi-classification task with K classes. At the same time, for each model, its own copy of the training sample is compiled in which the +1 mark is assigned only to one particular class, and the rest is assigned the -1 mark. Thus, you can get a composition of classifiers, each of which works only for one of its own class. How to use this composition now? If we use logistic regression, then it is enough to only evaluate the probabilities of the corresponding class labels in each model and among them choose the label in which the probability takes the greatest value.

Slide 28

Do you remember I said that linear models can be transformed in a special way so as to restore non-linear dependencies? In order to do this, you do not need any magic. There is a standard trick consisting in adding new features to the model. Imagine that in the training set there are points that are located as shown in the graph. Gray color indicates linear regression. As you can see, a simple linear regression is not able to adapt well to the points in the training set. But we can generate new features that depend non-linearly on the original features. Imagine that we have only one attribute x. We can calculate the value of the square of this feature and add it with some weight to the model. The result is a model with three weights - one weight is w0 and weights w1 and w2 for features that depend on x. This regression is shown in red on the graph. We can go further and calculate the sine of x instead of the square of x. Then we will more accurately restore the dependence. Please note that we introduce non-linear features, but the model remains linear as well, since the weight vector as before is simply scalarly multiplied with the features. This means that we can continue to use the same gradient descent algorithm, but now we can solve the problem of nonlinear regression and classification.

Slide 29

Finally, let's talk about overfitting of linear patterns. What does it consist of? If in the training sample of features there are more than examples for training, or there is a linear relationship between the features, then you can choose a vector v such that if you use it as a vector of weights, the linear model function will always return zero. This means that we can add this vector arbitrarily many times to the vector of weights and the answer of the model will not change from this. It turns out that the vector of weights, firstly, can have a large number of values ​​that solve the problem, and secondly, and this is bad, the vector of weights can have infinitely large values ​​of weights. What does this lead to? This leads to the fact that the classifier or regressor becomes unstable in its answers, that is, small changes in the values ​​of the attributes can lead to a radical change in the response of the classifier or regressor. To determine the presence of overfitting, drawing learning curves and observing the vector of weights can help. If, after training, the vector of weights has very large values, then this indicates overfitting.

Slide 30

How to try to minimize overfitting? It is interesting that this is very simple to do, just add a dependence on the length of the vector of weights. It turns out that by minimizing the first function, we will strive to reduce the value of the vector of weights, avoiding overfitting. The tau parameter is the force with which the additional term affects. This addition to the error function, which imposes a restriction on the vector of weights, is called the regularizer. An interesting fact is that the gradient descent algorithm is practically unchanged. All you need to do is add an additional term in the gradient descent step. How to choose a tau parameter? There are several simple techniques, the first is to try a few random parameter values ​​and in the end choose the best model. The second method is to use the LeaveOneOut method, which we studied in the second lecture of the course. How to do this, because tau is a real number? You need to select a certain range of tau values ​​and divide it into a finite number of possible values ​​and iterate over these values ​​using the LeaveOneOut method. In this way, you can regularize the linear model and avoid overfitting.

Slide 31

So, in this lecture we learnt the simplest method in machine learning of solving classification and regression problems. But the apparent simplicity is fraught with many possibilities. The linear models give us pure mathematical representation of machine learning task and produce the way of imitating a human brain and a problems decision way. In the next lecture we’re going to learn how to process feature values to increase the quality of machine learning models.