Slide 1

Good afternoon and welcome to the seventh lecture of the machine learning. Today we will continue to study probabilistic models to solve the classification problem. Today's lecture is perhaps the most difficult in the whole course and for this reason it is rather optional. Questions in the exam for this material will be at extra points. Moreover, today is the final theoretical lecture. The next lesson will be devoted to combining all that we have learned into a single material. So, today we continue to talk about the Bayesian theory of classification.

Slide 2

Today we recall the main material of the previous lecture so that it is not so difficult to navigate the new material. We recall the parametric approach to reconstructing distributions and examine how these models are related to linear classification. We will again return to the logistic regression model and see that this model combines the best of two worlds - linear and probabilistic classification. We will also go further and find out how the third approach to train probability densities works, which is based on the idea of ​​a mixture of distributions. We will prove some useful formulas and see how this model can be considered with respect to the Gaussian density of distributions. So let's get started.

Slide 3

Let me remind you of the main points that we covered in the previous lecture. We are dealing with Bayesian theory of classification. This means that we are trying to present the problem of machine learning from the point of view of probability theory. Let me remind you that from the point of view of machine learning, any task comes down to finding a decision function in the space of objects X and answers Y. Considering this problem as a problem of probability theory, we imagine that there is some probability distribution in the space X-Y and our task is to use Bayes theorem to find a classification algorithm that would minimize the probability of error. At the same time, at the last lecture, we talked about the Bayesian classifier being optimal from the point of view of the probability of error, but, unfortunately, only from a theoretical point of view. Nevertheless, probabilistic models are very useful in solving machine learning problems because they allow not only to find algorithms that solve the problem, but also to determine class probabilities, which is very useful in many machine learning problems. For example, in a task like credit scoring, it is very important for us to know with what probability a person will not repay a loan.

Slide 4

Returning to the topic of risk assessment, we can recall that from the point of view of the probabilistic statement of the problem, the risk is the numerical value of the expected loss when using the classifier. If we know the probabilities with which the classifier is mistaken on many classes, then knowing the price of the classification error, we can estimate the expected losses. This value allows you to compare classifiers with each other and choose the best, as well as understand what to expect from such a classifier when used in production. And as I said, there is a theorem that says that subject to certain restrictions, the Bayesian classifier is optimal in terms of minimizing this error.

Slide 5

Let me remind you that when using probability models, we are faced with the task of reconstructing two types of probability distributions. The first is the a priori distribution of the answers in the problem. The second is the likelihood function for training sample objects. The a priori distribution of classes is nothing more than the frequency of a given class, which is very easily calculated from the sample. The hardest part is the likelihood function evaluation, which allows you to determine how much the passed object is typical of class “y”.

Slide 6

In the last lesson, we examined two of the approaches presented on the slide. Let me remind you that in the general case there are three approaches to assessing the likelihood function by the training set. The first approach is a parametric approach. According to this approach, we model the distribution of objects of the training set in the class using some well-known probability density model. For example, we can use the multidimensional normal distribution, as we did in the last lesson, representing many objects of the same class as a cloud centered in vector space. The nonparametric approach to estimating the probability density was already familiar to us by the method of nearest neighbors. In the framework of this approach, we estimate the probabilities by counting the number of neighbors of the classified object. A feature of this approach is that it, in fact, does not require any parameters and we can restore the distribution density using only the training sample. Finally, a mix of distributions approach is a probability distribution as a weighted sum of simpler distributions. Weight in such an amount can be interpreted as the probability that the object “X” belongs to the corresponding component of the mixture. Today we’ll talk about how to train such models using the EM algorithm, one of the most powerful approaches in machine learning.

Slide 7

Let's go back briefly to parametric distribution models. Let me remind you that in the framework of the parametric approach, we considered a multidimensional normal distribution. This is a probability distribution in which a generalization of the Gaussian to the multidimensional case is used. Instead of the numerical value of the mean, a vector is used here that determines the center of the normal distribution in multidimensional space. Instead of variance, a covariance matrix is ​​used here, which shows how features of objects correlate with each other. Now we will see how it is possible to turn such a probability distribution model into a linear model.

Slide 8

But before that, let us remind ourselves what a multidimensional normal distribution looks like on the example of three-dimensional vector space. Let me remind you that if the set of objects was obtained from the normal distribution, then such a set of objects can be represented as an ellipsoid with a center that coincides with the mean vector, and the diameters of the ellipse and its orientation in space is determined by the covariance matrix.

Slide 9

Before moving on to the linear model, let's discuss the difference between the two basic visions of the machine learning problem. The picture on the right shows two sets of objects. Perhaps both sets were obtained from the normal distribution. That is why we can assume a solution that corresponds to the right side of the figure. We can model objects of different classes as a result of observing different probability distributions, for example, normal distributions. Models that explicitly model the properties of classes of objects are called generative. But the classification problem can be approached also since linear models are suitable for this. We can not try to describe the properties of classes, we can try to find a dividing surface that will separate two groups of classes from each other and, thus, solve our problem. Such models are called discriminative models. It is interesting that between the two worlds there is something in common, and we can see this through the example of a normal distribution.

Slide 10

It turns out that for probabilistic models there is the concept of a dividing surface. What it is? In fact, this is the set of all objects for which the probabilities of two classes turn out to be the same from the point of view of our model. For a Bayesian binary classifier, such a set of Hp points, taking into account the cost of the error, is indicated on the slide. And this is interesting, if we construct such a surface for a normal distribution, then it turns out that this is a simple second-order surface. An example of such a surface for a two-dimensional problem is parabola. That is, the Bayesian classifier using the normal distribution can be considered as a discriminative model with a parabolic separating plane. This classifier has such a surface in the general case when the covariance matrices are not equal. But what happens if the covariance matrices are equal? If the covariance matrices are equal, then the ellipses of the normal distributions will have the same orientation, but then a simple plane can be drawn between them and then such a model becomes a linear classifier. Let's try to prove it.

Slide 11

Imagine that all covariance matrices are equal. Let's explicitly write out a classifier formula that uses argmax. Since we use argmax, we can change the function inside it as we wish, if this does not change the signs of the expression. Then we can apply the logarithm, which does not affect the classification result in any way, since the logarithm is a monotonic function. Once we have applied the logarithm, we can explicitly substitute the multidimensional normal density formula or the Gaussian model. To do this, take the formula from the previous slide and knowing that the logarithm of the exponent is just a power at the exponent, we get the following large expression. If you look closely at this expression, you can select several components in it. The first component is alpha, which is a multiplier for the vector of the “x” object. The second component is beta, which is independent of the “x” object and defines the properties of the “y” class. When replacing the formula with alpha and beta, we can very quickly notice that we have come to a simple expression that linearly depends on “x”. That is, it turns out that with equal covariance matrices, such a classifier is a linear model! This model is called Fisher’s linear discriminant.

Slide 12

The obvious advantage of such a classification model over other linear models is that its parameters are very easy to obtain from the training sample. To do this, simply calculate the mean and the covariance matrix. Such a model is much simpler and therefore more stable than a model with different covariance matrices. But, nevertheless, the disadvantages of the previous probabilistic model still remain, and if the covariance matrix turns out to be degenerate, it will be necessary to somehow simplify it or apply regularization to it.

Slide 13

But what if we move on? What if we use not only the normal distribution, but any other exponential distribution while maintaining the condition that the variance or other similar parameter will be the same. The picture shows some two complex distributions. It is obvious that a linear classifier can also be found for them. It turns out that if we apply the family of exponential probability distributions, then such a classifier can be found. An exponential distribution is a probability distribution that can be represented as a formula on a slide. This is some exponential function, which has as its parameters theta parameter, which is an analog of the mean vector and delta parameter, which is a generalization of the covariance matrix or any other parameter similar to variance. It turns out that if we can rewrite the probability distribution in this form, then we get a linear classifier. Let's try to prove it.

Slide 14

First, we write out the binary classification problem. Please note that for binary classification using probabilistic models, the argmax expression can be rewritten using the sign function, which checks to return the sign of the expression contained in it. By slightly changing the expression inside we can see that if the expression inside the signum of the function is linear, then such a probabilistic classifier is linear, which means that all classifiers that can be described by exponential distributions are linear classifiers. Let us prove this.

Slide 15

To do this, we need to explicitly use the generalized exponential distribution formula that we introduced before. Pay attention to the signs of the classes in the formula - they will be very useful to us further. We substitute this model into the Bayesian classifier formula expressed through the signum function. Here we can again use the property of the logarithm - if we use the logarithm, then the value of the signum function will not change. Then, as was the case with the Fisher’s linear discriminant example, using the properties of the logarithm, we can simplify the initial expression and get the difference between the model parameters for the class +1 and -1. Let me remind you that the logarithm of a product is the sum of the logarithms. The logarithm of division is the difference of the logarithms, and the logarithm of the exponent is its degree. Then we get the expression of the difference in the distribution parameters presented on the slide. If we carefully group the members of this expression, knowing that “c (d) of class -1” is equal to “c (d) of class +1”, then we easily notice that the resulting expression is a linear function. Let me remind you that the triangular brackets denote the scalar product of vectors.

Slide 16

Well, if the logarithm of the ratio of two probabilities is equal to the scalar product of vectors plus some constant that does not depend on the object “x”, then the ratio of probabilities is equal to the exponent in the degree of the scalar product. Then, recalling the formula of complete probability for classes, we very quickly come to the expression in paragraph 2 of this slide. Thus, we have come to the conclusion that the probability that the object “x” belongs to the class “y” can be expressed through the use of a sigmoid function. That is, we have just been able to derive the logistic regression formula, which we have already seen in the topic about linear classifiers. Indeed, logistic regression is a classifier that is based on the idea that objects of different classes were obtained from an exponential family of probability distributions. Such a classifier constructs a linear dividing hyperplane and is a linear classifier. Moreover, such a classifier allows one to evaluate the probabilities of an object belonging to a class

Slide 17

Let's go back to linear classifiers and recall that the main idea is an attempt to approximate an undifferentiated stepwise loss function. There are many different approximations, and each allows us to obtain a new classifier with some properties. For us, the most interesting approximation is L (M), the logarithmic loss function. Why?

Slide 18

Because it is exactly logistic regression. How to check it? Very simple. Let's substitute our expression for evaluating the probability of a class into the maximum likelihood principle. Using the properties of the logarithm again, we quickly come to the conclusion that maximizing this function is equivalent to minimizing the quality functional Q (w) presented on the slide. If you look closely at the degree of the exponent, it is quite obvious that we have before us an expression that is already familiar to us for calculating margin. Thus, having explicitly written out the quality function through margin, we obtain an expression for the quality functional through the logarithmic loss function. It turns out we were able to find a connection between the world of linear classifiers and Bayesian classifiers. And this connection has been with us since the fourth lecture.

Slide 19

How to train such a model? Now we do not need to think about the form of probability distributions. We explicitly wrote that our model is a simple linear classifier, which completely depends on the vector of weights. Then we can use the gradient descent method, which is already familiar to us, to obtain a vector of model parameters. The slide shows the expression for iteratively calculating the vector of weights, which was obtained by substituting the quality function Q (w) in the method of stochastic gradient descent.

Slide 20

Let's move on to the topic of a mixture of probability distributions. Recall that according to this model, the likelihood of the object “x” is expressed through the weighted sum of simpler probabilistic models, where weight means the probability that the object “x” was obtained from the corresponding simple model. That is, the normalization condition must be fulfilled on these scales and they are positive. Then such a model can be a probability distribution model. Well, in this case, we can immediately formulate two tasks. The first is how to find out the parameters of simple probabilistic models that are in the mix. Second - how to determine “k” - the number of components of the mixture?

Slide 21

Let us, as for all probabilistic models, explicitly write the principle of maximum likelihood logarithm. What will we get then? The result is the sum of the logarithms of the sums of the components of the mixture. This is a very unpleasant model, because the logarithm of the sum cannot be converted in any way. Moreover, the model also has limitations in the form of inequalities. The problem is that such a problem in general does not have an analytical solution. No matter how hard we try, nothing can be done with the logarithm of the sum. Then you need to look for some approximate methods for solving this problem. One such method is called the expectation-maximization method or EM-algorithm. What is the main idea or philosophy of the method? This method assumes that the probabilistic model has some hidden parameters. These are parameters that affect its values, but are not explicitly written in the model formula. The EM algorithm involves an iterative process of refining model parameters using hidden parameters. The whole process in the algorithm is divided into two parts. The “expectation” part calculates the values ​​of hidden variables through the visible parameters of the model and the training set. For us, such visible parameters may be, for example, covariance matrices for a mixture of normal distributions. The “maximization” part provides the calculation of model parameters through hidden variables and the previous state of the model. It turns out a closed process, which through itself expresses new values ​​of its own parameters. It is interesting that such a process makes sense and the EM-algorithm allows you to effectively restore the parameters of probabilistic models.

Slide 22

Before we get to the derivation of formulas for a mixture of distributions, let's immediately look at the answer. The hidden variables for the mixture of probability distributions are g-ij. Please note that they are calculated only by the model parameters and the training sample. These variables were said to be computed at the E-step. At the M-step, we seek to maximize the likelihood based on the values ​​of the hidden variable. Namely, we strive to calculate the distribution parameters in the mixture and the values ​​of their weights. Please note that here the calculation is performed only using hidden variables and the previous state of the model. But what exactly are these hidden variables?

Slide 23

It turns out that hidden variables have a probabilistic meaning for us. The hidden variable g-ij is the conditional probability that the random object of the training sample x-i was obtained from the mixture component with the number “j”. Note that this probability can be expressed through the Bayesian theorem. Then we need to know the probability of the mixture component and the likelihood of the x-i object for each component of the mixture. But we know these parameters! The probability of a component is its weight in the mixture. The total likelihood of an x-i object is a mixture of likelihoods for each component based on their weights. And the likelihood for one component is nothing more than the value of the probability distribution function of an individual component with number j. Thus, despite the fact that the g-ij variables are hidden and seem artificial, these variables have a completely understandable interpretation. It turns out g-ij is a posteriori information about the component of the mixture. By the way, based on the normalization in the denominator according to the Bayes formula, it becomes obvious that the sum of all g-ij by components, regardless of the selected element x-i, is 1.

Slide 24

Let's now begin to derive the above formulas in the EM algorithm. In order to solve this problem, we need to familiarize ourselves with the Karush-Kun-Tucker theorem. This is a theorem of the mathematical theory of optimization, which allows us to answer the question of how to minimize or maximize some function with a set of restrictions. This is exactly what we need. Let me remind you that in our problem, the objective function is the principle of maximum likelihood, and the constraints are a set of conditions on the weight of the mixture of distributions. The Karush-Kun-Takker theorem tells us that if a system of equations and inequalities satisfies a number of constraints that we will not consider here, then it can be solved through another system. This system is written below. Let's describe her. A new objective function L (x) appears in this system - this is the sum of the original objective function f (x) with a set of constraints taken from the original system with some lambda weights. In this case, the lambda weights must be greater than or equal to zero. The solution to the system is a vector X which, when substituted into a new function L (X), will translate the gradient of the function L (X) to zero. It is this ratio that allows us to find the answer to our task. Let's try to see what happens if we substitute the restrictions from the original problem into the system according to this theorem.

Slide 25

To begin, let's explicitly write down a new function L from the original principle of maximum likelihood and restrictions on the weight of the mixture of probability distributions. Then we can take the partial derivative for each parameter. Let me remind you that the gradient is a vector of partial derivatives and if it is equal to zero, then each partial derivative is also equal to zero. What happens if we calculate the partial derivative by weight of the mixture? To begin with, we note that we get an expression that does not depend on the weight coefficient at all. If this expression is equal to zero, then multiplying it again by the same weight w\_j we will not change anything. Now pay attention to the fact that now the expression under the sum is exactly equal to g-ij - a hidden variable. If we now transfer the lambda part to the right side of the equation and summarize all such expressions for each w\_j, then, based on the fact that the sum g-ij is equal to one, and the sum of units over the entire sample is simply equal to the size of the sample, we get an important equality. Namely, the lambda is equal to the size of the sample for training.

Slide 26

What next? Now we need to substitute this equality into the original formula for the partial derivative with respect to w\_j, multiply it again by w\_j and express this value through it, as shown in the slide above. But then the expression for calculating the weights of the mixture of distributions through hidden variables that we originally presented for the EM algorithm is clearly obtained. What now? Now we also need to get an expression for the partial derivative with respect to theta\_j parameter. Very quickly you can make sure that we again get an expression that depends on hidden variables. True, since we do not know the form of the distribution function “p”, then we cannot go further. But please note that equating such a partial derivative to zero is equivalent to applying argmax to the expression for the derivative. In this case, it turns out that the new value of theta\_j is equal to the result of maximizing, by this parameter, one component of the mixture p-j.

Slide 27

Let's rewrite this whole bunch of formulas into a single algorithm. The final EM algorithm for reconstructing a mixture of distributions is presented on this slide. Let's look at it a little more carefully. The input parameters of the algorithm are the training set and the initial ones, the number of mixture components and the initial value for the parameters of the mixture components. To begin with, we need to somehow get the initial approximation of the hidden variables. Let's just take random values ​​from 0 to 1, taking into account normalization. Now we must repeat the steps of the EM algorithm until the hidden variables stop changing. At the E-step, we, according to the formulas that we proved earlier, calculate the new value of the hidden variables, while remembering the old ones. Next, perform the M-step. We update the mixture component weights and model parameters, performing optimization for these parameters.

But all this sounds very incomprehensible, because we do not know how the models of the components of the distribution mixture look. Let's look at the most popular method - using normal distribution for the EM algorithm.

Slide 28

Let me remind you that maximizing the likelihood for a normal distribution is equivalent to calculating the mean and variance values ​​from the training set. This form of multidimensional normal distribution corresponds to the naive Bayes classifier. Then, if we fix the label of the class “y”, then for each class and each component of the mixture of distributions with number j, as well as for each feature with number d, we must calculate simple statistics, namely, the mean value for the feature and the variance value for the feature, as if it were one-dimensional tasks. These expressions must be substituted into line 9 of the algorithm on the previous slide.

Slide 29

Then we can substitute the entire expression for the mixture of Gaussian distributions into the Bayesian classifier and get the expression presented on the slide. Note that the expression is written in such a way that it closely resembles the k-nearest-neighbor method. Indeed, this algorithm can be represented just like that if we use the Gaussian kernel function. Thus, we were able to connect two more worlds with each other. First, we showed how probabilistic and linear models are related, and now we can see how probabilistic models and the k-nearest-neighbor method are related. Indeed, in the world of machine learning, all methods are interconnected.

Slide 30

But we can go further and present the previous expression as a graph of the computational process, which consists of several stages. At the first stage, we calculate the likelihood of the object “x” for each component of the mixture and for each class, at the second stage for each class, we add the likelihood values ​​with the corresponding weight, and then, multiplying the obtained expressions by the a priori probabilities of the classes, select the largest value using argmax . But this is nothing more than a neural network! It turns out that we were able to see in this model another invisible layer, which is a whole family of machine learning algorithms. This is a truly fantastic observation.

Slide 31

So, having considered the device of the EM algorithm for reconstructing the parameters of a mixture of probability distributions, let's look at its advantages and disadvantages. Firstly, this algorithm converges much faster than other iterative algorithms, for example, the gradient descent method for linear models. This algorithm automatically detects the structure of classes, highlighting the individual components of the mixture. This allows you to learn new information about the problem being solved. This algorithm, like all probabilistic models, allows you to evaluate risks in the problem to be solved, because it allows you to calculate the probabilities of certain events. But the EM algorithm also has a number of significant drawbacks, namely: it is far from always clear how to choose k - the number of mixture components and the algorithm depends very much on the initial approximation, i.e. on how we initialize the parameters of the probability distribution components.

Slide 32

So, we have completed the chapter on probabilistic methods in machine learning. This was perhaps the most difficult lecture in the course, but no less interesting! We were able to interconnect several worlds - the world of linear classification models and the k-nearest-neighbor method with probabilistic models. We saw that a neural network is hidden under some probabilistic models. We met with a powerful EM algorithm. We proved that a linear classifier based on the logarithmic loss function is a Bayesian classifier. Truly many interesting discoveries for one lesson. Thank you very much for your attention and all the best!