Slide 1

Good afternoon, and welcome to the third lecture on the Machine Learning course. Today we will introduce an alternative approach to machine learning. This approach consists in applying logical rules and conditions when solving the problem of machine learning. Unlike the previous approach, which can be described as “show me an example and I will answer your question”, this approach can be briefly described as “show me an object and according to its characteristics I will prove that the correct answer is this”.

Slide 2

Here is the plan for today's lesson. We begin by discussing what a logical rule is and how it can be used in machine learning. We will talk about how logical rules can be described in terms of machine learning, how to choose rules from among many rules, and how to use them to solve a practical problem. Then we move on to a model called the Decision Tree. This model largely follows the ideas of applying logical rules in machine learning. In fact, the decision tree is a way to describe logical rules in a human-readable way. Decision trees are a very convenient and powerful approach to machine learning, but it is not without drawbacks. In particular, decision trees suffer from statistical inaccuracy of data in tree nodes and from overfitting. We will talk about how to get rid of overfitting in decision trees using the example of method of trees pruning and random forest. We will also talk briefly about how to turn a classifier based on a decision tree into a regressor. It turns out that this is as simple as in the case of algorithms based on k-nearest neighbors.

Slide 3

Let me remind you that in machine learning we are faced with the task of finding some rule by which a certain answer is assigned to each object of the training sample. In particular, we are interested in the search for the function y (x). This is similar to how people try to find the law of nature in physics - we have many examples or experiments and answers to them, and we need to find some rule that connects these two things. Okay, but how do we find such a function? Most machine learning tasks are somehow related to tasks that were always considered intellectual and only a human could handle them. So, if we try to somehow repeat the process of solving a problem by a human, then we can come closer to solving the problem of machine learning. One of the ways a human answers the question of a problem is logical reasoning. Let's try to recreate the process of human reasoning and thus solve the problem of machine learning.

Slide 4

So, what is a logical rule in terms of machine learning? From a mathematical point of view, a logical rule is a function that takes a description of an object in the form of a feature vector and returns a response of one or zero, true or false. We want the rules that are obtained in solving the machine learning problem to have a number of characteristics. Any rule R must be interpreted. This means that the rule should be written in the form of a phrase in natural language and it should consist of a finite set of logical conditions. Usually this is not more than one to seven conditions per rule. Also, the rule should be informative. This means that, thanks to this rule, we must most effectively separate objects into different classes. That is, if the rule takes the value true and this corresponds to the fact that the object should be classified as class “c”, then as many objects of class “c” as possible should be highlighted by this rule. At the same time, the smallest possible number of objects of other classes should be accepted by this rule.

Slide 5

On the slide you can see a schematic representation of such rules. The first rule perfectly fulfills its task. This rule highlights quite a few objects of the green class and does not highlight red dots at all. The gray line shows the ideal rule to strive for. The first rule is also called a consistent rule or a pure rule. The second rule cuts off all green objects at once, but at the same time takes a few red dots. This rule is also useful because it is wrong, but not by much. Such a rule is called an informative rule. Finally, the third rule does not separate the red and green dots at all. After we apply this rule, the ratio of red and green dots will not change. This rule does not bring any benefit.

Slide 6

Here are some examples of the rules we need.

* If patient's age \_ 60 and patient su\_ered a heart attack, then we don't do an operation, risk of death = 60%.
* If a potential borrower wrote his/her home phone and his/her salary \_ 2000$ and loan amount \_ 5000$ then the loan could be approved, default risk = 5%.
* If the book author is Arthur Clarke or Liu Cixin and text of the book contains words "spaceship", "planet" and "alien" then the book corresponds to the science fiction genre.
* If the email contains words "sale", "buy", "discount" and the email's sender sent more than one email last two days, then the email is spam with the level of con\_dence = 80%.

Please note that these rules are a set of conditions that are combined according to the logical "AND". Also, it is worth paying attention to the fact that it is very convenient to introduce logical rules on features that correspond to certain numerical values ​​- you need to enter some threshold value by which the rule applies. Another important characteristic of the rule is its ability to show the degree of confidence in the answer. For most machine learning tasks this is very important.

Slide 7

I would also like to show some examples of the Bongard test. These tests were developed by Bongard to evaluate the quality of image recognition systems and logical rules derivation. According to this test, the algorithm needs to derive some rule that will allow you to separate the images to the left of the images to the right according to the characteristics that should be found in these pictures. I suggest you think a little about what rules should be invented for a series of pictures.

So… what’s kind of rule could fit to this picture?

So, the rule for this test: if the figure is black, then it refers to the right pictures, if white, then to the left.

Slide 8

What will be the answer for this task? Think about it…

The rule for this test: if the figure is smooth, then this is the right class of figures, otherwise it is the left class of figures. Note that now the task for the algorithm is very complicated. The algorithm should somehow derive the concept of smoothness of a figure. This is not an easy task.

Slide 9

And what is the answer for this task? What do you think? ...

In this test, the algorithm must understand that there are closed and open figures. Closed figures refer to the left side of the image, and open figures refer to the right side of the image. As you can see, the algorithm must somehow discover the human intuition in geometry.

Slide 10

Finally, another interesting task ... what is the answer for it?

It seems that here the algorithm should deduce the concept of horizontality. If the figure is horizontal, then it belongs to the class on the right, otherwise the figure refers to the class on the left. If you are involved in image processing, then Bongard tests are a great way to check how your algorithm works on artificial data.

Slide 11

Now let's move on to the questions that arise when developing a rule generation system. How to choose attributes for rules? Unfortunately, this is the only question that we cannot give a definite answer, because the invention of features is a creative process that is highly curled from the problem you are solving. Perhaps this is the only moment when a hunan really plays a big role in solving the problem of machine learning. The second question is what kind of rules do we need? As we have already said, we want the rules that are generated by our algorithm to be interpreted and easily perceived by humans. How to mathematically write such rules? Third question. How to choose from the set of rules those rules that are most useful to us? Let me remind you that before that we formulated utility, but how to express it in the form of the function that we want to optimize? How to combine both criteria of information content? And the fourth question. How to infer logical rules from the set of objects in the traning set? How can we unite a lot of features to some rules? Finally, how can these rules be used to solve the classification or regression problem?

Let's try to answer all these questions.

Slide 12

So, we are only interested in such rules, which can be written in the form of some formula. This will not only improve understanding of such rules, but also simplify the process of programming them. The first way that you can set the rules we have already seen. This is a form of rules under which there is a set of conditions and restrictions on features of objects and these conditions are combined in a rule using the logical operation “AND”. The slide shows an example of such a rule, where I highlighted the conditions and the combination of conditions. Conditions can specify both left and right boundaries for the value of the attribute and only one of the boundaries.

Slide 13

The next type of rules is very familiar to doctors. How does a doctor work? The doctor will examine the symptoms of the patient’s illness, perform a series of diagnostic tests, and ask the patient several questions. In this case, the doctor may not do absolutely all the measurements, and some of the symptoms may simply be absent. Despite this, the doctor can make a diagnosis with a set of positive tests or symptoms. This set is called a syndrome. Rules of this kind are very useful not only in medicine, but also in the task of credit scoring, for example. The parameter of this rule is the number d. Note that if d is equal to one, then this rule turns into a disjunction. If d is equal to the number of conditions, then the logical rule turns into a conjunction, that is, into the rule from the previous slide. The syndrome allows you to find a middle point between a strict conjunction and a soft disjunction. Functions of this kind are called linear Boolean equations.

Note also that threshold values ​​can be set not only for features that take a numerical value. Boolean and nominal features may also be limited by such conditions. The slide shows an example of a constraint for a Boolean attribute. By analogy, one can also develop restrictions for a nominal attribute.

Slide 14

Another way of specifying a feature is to use a hyperplane that separates two groups of objects among themselves. Such a function corresponds to a convolution of characteristic values ​​with some weights. If the sum of these values ​​exceeds a certain threshold, then the rule will be true, otherwise the rule will return false. Rules of this kind correspond to the so-called linear classifiers and this will be the topic of our next lecture.

Another example of a rule is the use of a ball condition. Note that this condition matches exactly the nearest neighbor method that we examined in the previous lesson. Such a rule is as follows: “If the classified object is located at a distance from the prototype no further than w\_0, then return true, otherwise false.” Ball-type rules are very useful for doctors because they clearly provide an example of a patient who fits this rule.

I also remind you that the notation “condition in square brackets” is Iverson's notation, which corresponds to a function that returns one when the condition is met in brackets.

Slide 15

The next question: how do we formalize the information content criterion? Let me remind you that we want some rule to efficiently define objects of the same class. This means that the number of objects of the target class should be recognized by the rule as large as possible, and the objects of all other classes should be as small as possible. We must develop a criterion that links both characteristics into one. Several options immediately come to mind: precision, accuracy, relative accuracy and linear cost accuracy. Such measures of informativeness really combine both criteria, but functions of this kind have a number of disadvantages.

Slide 16

Take a look at the table on the slide. Imagine that we are solving the binary classification problem and the training set contains 200 objects with positive class and 100 with negative class label. Now let's assign different values ​​for p and n - the results of the assessment for each information content criterion. Note that in the first two cases, accuracy does not distinguish between a good rule at number one and a bad rule at number two. Why is the first rule better? Indeed, it allocates fewer positive-class objects, but it does not emit negative-objects at all. The second rule in this sense is worse. Linear cost accuracy may make a different kind of error. Note that in the fourth row, this feature found the rule as useful as the rule in the third row. At the same time, the fourth rule identified only 5 objects of the training sample. This is a statistically insignificant rule. It turns out that Linear cost accuracy is insensitive to overfitting. And so on. For all the information content criteria from the previous list, one can easily select marginal cases in which such criteria do not work.

Slide 17

Fortunately, there are other criteria that solve this problem. On the slide you can see two Information Gain criteria and the Fisher’s statistical test. The first criterion reflects the amount of entropy before applying the rule to the training sample and after. The second criterion seeks to assess how much the application of the rule is unlike random mixing of elements of the training set. In any case, both criteria evaluate how the rule being evaluated mixes objects of different classes. In other words, these criteria evaluate the degree of uncertainty in our knowledge about sample objects before and after applying the decision rule.

Slide 18

A useful graph for analyzing rules is the so-called (p, n) -plane. This is a graph that connects the number of erroneous classifications n and the number of correct classifications p with informational criteria. The slide shows a schematic representation of this graph. The gray diagonal line shows the border of completely useless rules. If a rule falls to a point above this line, then it is useless. The pink area shows the rules that find some statistical patterns. That is, these are rules that do not emit strict features of dividing objects into different classes, but at the same time can bring additional useful information. The most useful rules are those in green, yellow, and red. Such rules are called logical rules and the color of the area shows the severity of one of the listed informational criteria. The more to the right and lower the point on the graph, the more informative the rule is, the more we are confident in the logical regularity that is determined by this rule.

Slide 19

Finally, let’s move on to the process of generating new rules. The slide shows the pseudo-code of this procedure. There are a large number of methods for obtaining new rules. Some of them are listed on the slide. You can use genetic programming to generate rules. Its essence is that a set of some rules is generated according to a given algorithm, then each rule is evaluated from the point of view of information criterion. After that, the most informative rules are selected and the algorithm for crossing and mutating rules is performed for them. This procedure consists in combining parts of successful rules into new ones, as well as randomly modifying existing ones. An example of a modification may be the displacement of the boundary value for a characteristic or the selection of another characteristic. After mutating the rules, the procedure is repeated - an assessment is performed, informative rules are selected, and so on. The rule generation procedure stops when the rules stop changing dramatically. The method of stochastic local search is to select a rule and generate a set of random modifications to the rule. From the new rules, those that are more informative are selected. You can find more information about these methods on the Internet.

Slide 20

Let's move on to an algorithm that implicitly generates sets of rules. This algorithm is called the “Binary Decision Tree”. The main idea of ​​the algorithm is that a “binary tree” data structure is built and a certain predicate, condition, or simple logical rule from those that we saw on the previous slides is associated with the internal nodes of this tree. The leaves of this tree correspond to the class label. The process of classifying such a tree consists in sequentially traversing the tree using logical rules in nodes. If the rule is fulfilled, then the object should be classified using the right subtree. If the rule fails, then the left subtree should be used. Tree traversal stops in a leaf that assigns a class label to the current object. We introduce some notation. Beta is a predicate in a tree node. V\_inner matches the set of all internal nodes of a tree with a predicate. V\_leaf matches the set of the leaves in a tree.

Slide 21

On the slide, you can see how decision trees work on the example of the task of classifying iris flowers. As you can see, the predicates in the internal nodes build a simple dividing hyperplane parallel to all the features except one. The decision tree combines a set of such hyperplanes and thus builds a complex surface that separates objects of different classes. How is the decision tree built?

Slide 22

On this slide, you can see the pseudo-code of the decision tree construction algorithm. This is a recursive algorithm. That is its idea. The first line is an ending the recursion. If the transferred set of objects consists of objects of the same class, then we must create a leaf node with the label of this class. Otherwise, we must split the set U into two. We will generate predicates for all feature sets. Each such predicate will consist of a threshold condition for one of the possible attribute values ​​from the training set. Among the whole set of predicates, we choose the most informative one. Further, in lines 5-6, the set U is split into two. The first set corresponds to the left subtree, the second corresponds to the right subtree. If it turns out that one of the sets is empty, then we have no choice but to create a leaf node in the tree. In this case, the class label is assigned to the leaf by voting, as is done in the k-nearest neighbor algorithm. Otherwise, we create an internal node, recursively create a left and right subtree for it, and return the result.

Slide 23

How to evaluate the information criterion of conditions in a binary tree? You can use all the criteria that saw on previous slides. However, there are other criteria that were invented specifically for Decision Trees. In particular, the Gini criterion and the dual criterion D-criterion of Donskoy. Both criteria evaluate the quality of a decision tree splitting a set of objects. For example, the Gini index counts how many objects of the same class fall into the same subtree. At the moment, the most popular is the Gini index.

Slide 24

Decision tree is the first algorithm that can work with gaps in data. How it works? At the training stage, we discard objects from the sample when evaluating the information content of the criterion, if there is a missing attribute that is used by this criterion. For the remaining objects, we estimate the probability of transition to the left branch of the tree. This probability will be useful to us in classifying an object that has a missing value. How is this classification performed? We must evaluate the likelihood that the object “x” belongs to some class “y”. If “x” contains the attribute for the current condition in the tree node, then we use the first formula, that is, we ignore the probability of one of the branches and go to the lower level of the tree. If the current node is a leaf, then the probability of the class “y” is the proportion of objects in the training set that corresponds to this leaf and this class label. If the object “x” has a missing value, then the probability of class “y” is the sum of the probabilities of class “y” in the left and right branches. In this case, each of the probabilities is weighted by the value q0 - the probability of transition to the left branch of the subtree. Thus, decision trees allow not only building logical rules, but also evaluating the degree of confidence in the answer.

Slide 25

With all the ease of use and interpretation, decision trees have a number of significant drawbacks. This algorithm often suffers from overfitting. Very few positive examples often accumulate in the leaves of decision trees and this leads to the fact that the classification result is statistically unreliable. As a result of this, this algorithm is very sensitive to noise in the data. Moreover, the greedy tree-building algorithm does not guarantee that an ideal decision tree will be built. The figure above shows the task of determining the XOR rule. According to this rule, identical labels are assigned to objects along the diagonals of a square. The figure above shows the ideal tree for solving this problem. But from the point of view of all informational criteria, such a division of many objects is meaningless. Therefore, the algorithm will first select the upper green dots, then it will separate a large group of red dots and finish its work in each subgroup. Obviously, the rules found do not solve the problem well. This is due to the greed of the ID3 algorithm.

Slide 26

One way to deal with this is pruning of decision trees. The main idea of ​​this algorithm is this. Let's take some set of objects on which we did not train a tree. After that, in each node of the tree, we calculate how many objects of this set fell into it. If in any node there are no objects at all, then this means that the node is useless and can be thrown out. After that, you can look at all the nodes that contain a small number of objects of this set and try to evaluate the possibility of replacing such nodes with leaves or child subtrees.

Slide 27

The slide shows the pseudo-code of this procedure. Lines 2-6 evaluate the proportion of objects from the developing set that fall into the internal nodes of the tree. After, in lines 7-8, all nodes that have not classified any objects from the developing set are replaced with leaves.

Slide 28

Next, the remaining nodes are evaluated. In lines 3-6, the number of erroneous classifications for the current node, its child nodes, and the potential node corresponding to the leaf at this point is considered. Among all error counters, the minimum value is selected and the corresponding decision is made - leave the node as is or replace it with a child node or leaf.

Slide 29

Another example of solving the overfitting problem is the use of so-called oblivious decision trees. Such trees are much less flexible when learning. In particular, each level in the tree is associated with the same rule for all nodes at this level. It turns out that this can significantly accelerate the learning process. In combination with the RandomForest technique, which we will study later, this method of solving the classification problem is quite effective.

Slide 30

But how to solve the problem of regression? It turns out, as in the previous k-nearest neighbors method, we need minimal changes to the original algorithm. Decisive rules remain the same as they were. The tree generation algorithm does not change at all. Only two things change. The first is computing information criterion. Since we are solving the regression problem, it is important for us to evaluate how strongly the spread in the data for the regression is expressed. This is essentially an analogue of the entropy criterion for a continuous function. The smaller the range of values, the higher the confidence in the answer. In statistics, the spread of function values ​​is estimated using the Variance function. It is the variance of a random variable that is the main criterion for information content when constructing a decision tree for a regression problem. Finally, we need to change the value assignment algorithm in the leaves of the tree. It's still simpler here. We can estimate the average value of the response on a subset of the objects that hit the sheet. And that’s all. Now we can solve the regression problem using trees.

Slide 31

Finally, let's look at another important idea for improving the DecisionTrees algorithm. We talked about the fact that the decision tree is a statistically unstable classification or regression algorithm. We can use pruning, but there is another technique that builds ensembles of algorithms. There are theorems that tell us that if we randomly divide the entire training set into random subsets. And if in these subsets we take random sets of features from the original ones, then the set of trained algorithms for these sets will be statistically more stable. If we use decision trees for classification, then we need to train several trees and use the principle of voting by classifiers to form an answer. If we solve the regression problem, then it is enough to take the average of the responses of each regressor separately.

Slide 32

The slide briefly shows this idea. Algorithm 5 constructs a random forest for the regression problem. We select k random subsets in the data and train k-models on them. Algorithm 6 shows the simplest procedure of answering of the RandomForest algorithm - we simply return the average value of the responses of individual models.

Slide 33

So, today we learnt a lot interesting things. We learned about another approach in ML: the rule-based classication. We studied how to apply the rule-based approach to train decision trees. We learned how to simplify decision trees. We studied how to transform classication tree into regression tree and random forest.

In the next lecture we will meet with the new concept in Machine Learning – linear models of classification and regression.