## **L2-regularization - Theory**

Let`s discuss the most popular topic of machine learning – regularization. We will look at it from several points of view so that you fully understand the reasons for using regularization and its theoretical consequences.

First of all, let us consider the problem of retraining. Let’s return to our Gaussian clouds, one of which is concentrated at a point with coordinates (2; 2), and the second is focused at a point with coordinates (-2; -2). This is the same problem for which we have found the solution of the Bayes classifier. It is easy to depict graphically – as a rule, these are two circles, separated by a straight line. As you know, the solution of the Bayes classifier, which we have found, is w = [0, 4, 4], where 0 is the bias term.

Which line corresponds to this solution? As we know from the course of mathematics, the equation of the straight line has the form

Note that in this equation, y represents the ordinate on the coordinate plane, and not the output result of logistic regression.

We can rewrite our equation in the form:

*y = mx + b.*

Consequently, the slope of the straight line is -1, and the free term, which characterizes the intersection with the y-axis, is zero.

*0 + 4x + 4y = 0, hereof y = -x.*

This can make you think why the solution for the Bayes classifier equals (4, 4). Why isn`t it (1, 1) or (10, 10)? Indeed, because all these solutions characterize the same straight line! This is the first clue why we might need regularization.

Consider the target function:

J = - [t * ;log (y) + (1 - t); * log (1 - y)], 

Note that y exactly means the output result of logistic regression.

We take a point (x1; x2) with coordinates (1; 1). We know that it belongs to the same set of points that are concentrated at the point with coordinates (2; 2). So, it belongs to class 1. If we substitute the value of our point in the formula of logistic regression with the given values of the weighting coefficients w = [0, 4, 4], we get the value of the sigmoid

*σ*(0 + 4 + 4) = *σ*(8) = 0,99966.

Only the result that equals one can be better than this one. Thus, for y = σ (8), the target function is approximate -0,0003.

And what happens if the weighting coefficients are (0, 1, 1)? In this case, the target function will have a value of about -0.12. But this is not a good indicator. If we set the weighting coefficients equal to (0, 10, 10), then the value of the target function will be approximate -2.0 \* 10-9.

The moral of the whole story is that the best weighting coefficients for the model are (0, infinity, infinity). Of course, the computer cannot process such numbers and will give an error.

People often understand the regularization in terms of regression from my previous course of linear regression. But now we are not examining regression, so this idea is inapplicable. In fact, even if our data occupy the entire range of possible values ​​of the input variables, no retraining occurs even if our model is very complicated. This is the reason why we should always have as much data as possible. Retraining happens when the model attempts to “guess” the result in the area where there have been no data before. If your training set covers the entire amount of data and includes all possible values, then you have the opportunity to train the model well. In other words, if your training dataset looks exactly like the test dataset, and the model shows a good result on the training dataset, then it will show the same good result on the test one.

Unlike the previous case, now even if our data are well balanced and include all possible variants of input variables, the logistic regression still tries to give a solution that the computer cannot calculate.

The solution to this problem is regularization. Regularization imposes a penalty on very large weighting factors. So, if we have the original function of the cross-entropy error

J = - [t * ;log (y) + (1 - t); * log (1 - y)], 

then an additional term can be added to the formula, which will increase the value of the error function if the weighting coefficients are too large:

J_{reg}= J + (frac {lambda} {2}) || w||^2 = J + (frac {lambda} {2}) w^T w.

This forces the weighting coefficients to tend to zero, and now we do not get weighting factors like (0, 10, 10) since in this case, the value of the error function will be very large.

The term λ is called the smoothing parameter. It balances the cross-entropy error function and the regularization penalty. If the value of λ is large, the weighting coefficients will tend to zero, if the value of λ is small or zero, then the weighting coefficients will merely tend to minimize the cross-entropy error function. As a rule, the value of the parameter λ is set to 0.1 or 1, or between these values, but mostly its value depends on the specific data. You need to try different values ​​and observe the behavior of the cost function and the final result. There is no universal method for determining the value of the parameter λ. Or rather, it may exist, but it is too complicated for our classes.

So, how does this affect the weighting coefficients?

Do not forget that with the gradient descent, all we need to do is to find the gradient of the objective function and move in its direction. Since adding a new member does not affect the calculation of the gradient descent, all the gradients can be found separately.

Let us consider the gradient with regularization penalties in more detail. On the one hand, it can be considered in a scalar form. In this case, the penalty is

reg;cost = (frac {lambda} {2}) (w^2_0 + w^2_1 + w^2_2 +;...

In this case, the derivative for any given wi will be equal to

frac {d(reg;cost)} {dw_i} = lambda w_i.

In the vector form, the new gradient will be equal to

frac {dJ_{reg}} {dw} = X^T (Y - T) + lambda w.

Let us consider another interpretation of regularization. As you remember, we also explain our model in terms of probability theory, and finding the minimum of the cross-entropy error function is also finding the maximum of the likelihood function. Let’s return to our target function as the one which maximum we want to find. Then

J = + [t * log; y + (1 - t) * log (1 - y)] - (lambda/2) ;* ;parallel wparallel^2. 

We now take the exponent:

expJ = y^t (1 - y)^{(1 - t)} * exp (frac {- lambda parallel wparallel^2} {2}). 

The first exponentiated part is the binomial distribution for the likelihood function. The second exponentiated quadratic term with a minus sign is merely a Gaussian distribution. All this is an introduction to the Bayes perspective of machine learning.

The Gaussian distribution of the weighting coefficient values is called the prior. This means that it represents our a priori expectations of the weighting coefficient values. In particular, they should be small and centered around zero, and the dispersion of this Gaussian distribution is 1 / λ. The parameter λ is also called the accuracy of the model; it is the inverse of the dispersion. We will meet it throughout Bayes machine learning again and again.

The rule of Bayes says: the posterior is proportional to likelihood multiplied by the prior.

p (B|A) = frac {p(A|B) p (B)} {p(A)} = frac {p (A,B)} {p(A)}.

The consequence of this rule is that

sum_{B}{p (A,B)} = sum_{B} p(A|B) p (B).

Thus, we are looking for the maximum of the likelihood function without regularization. Now we do not maximize the likelihood function – now we maximize the posterior. This method is called calculating the maximum of posterior or it is short for computing of MAP.

## **L2-regularization - The Code**

As you have guessed, we have proceeded to practice, and you will find out how to apply regularization in the code.

We use the code from previous lessons because it is very similar. We just add the regularization component. We simply add the constant λ, set its value equal to 0,1, multiplied by the value of the weighting coefficients.

learning\_rate = 0.1

for i in *xrange*(100)

if i % 10 == 0:

print cross\_entropy(T, Y)

w += learning\_rate \*(np.dot((T-Y).T, Xb) – 0.1\*w)

Y = sigmoid(Xb.dot(w))

Run the program. As you can see, now the value of the weighting coefficients is much smaller. This gives regularization. Since we assume that the data are typically distributed around zero, we get the weighting coefficient values much closer to zero, and they do not tend to grow to infinity.