**Учебный материал 2**

**Регрессия**

With linear regression we aim to fit a line (or hyperplane in higher dimensions) to a scattering of data. In this section we describe the fundamental concepts underlying this procedure.

Data for regression problems comes in the form of a training set of *P* input/output observation pairs:

$$\{\left(x\_{1},y\_{1}\right),\left(x\_{2},y\_{2}\right),…, (x\_{p},y\_{p}) \}$$

In many instances of regression, the input to regression problems is scalar-valued (the output will always be considered scalar-valued here) and hence the linear regression problem is geometrically speaking one of fitting a line to the associated scatter of data points in 2-dimensional space. In general, however, each input **x***p* may be a column vector of length *N*.

in which case the linear regression problem is analogously one of fitting a hyperplane to a scatter of points in *N* + 1 dimensional space. In the case of scalar input, fitting a line to the data requires we determine a slope *w* and bias (or “y-intercept”) *b* so that the approximate linear relationship holds between the input/output data, 

Note that we have used the *approximately equal* sign in because we cannot be sure that all data lies completely on a single line. More generally, when the input dimension is *N* ≥ 1, then we have a bias and *N* associated weights,

to tune properly in order to fit a hyperplane. Likewise, the linear relationship is then more generally given as



The elements of an input vector **x***p* are referred to as *input features* to a regression problem. For instance, the student debt data described has only one feature: *year*. Conversely in the GDP growth rate data the first element of the input feature vector might contain the feature *unemployment rate*.



**The Least Squares cost function for linear regression**

To find the parameters of the hyperplane which best fits a regression dataset, it is common practice to first form the *Least Squares cost function*. For a given set of parameters *(b*,**w***)* this cost function computes the total squared error between the associated hyperplane and the data, giving a good measure of how well the particular linear model fits the dataset. Naturally then the best fitting hyperplane is the one whose parameters minimize this error.

Because we aim to have the system of equations hold as well as possible, to form the desired cost we simply square the difference (or error) between the linear model $b+x\_{p}^{T}\*w$ and the corresponding output *yp* over the entire dataset. This gives the Least Squares cost function.



A simulated 2-dimensional training dataset along with a line (in magenta) fit to the data using the

Least Squares framework, which aims at recovering the line that minimizes the total squared length of the dashed error bars.



We of course want to find a parameter pair *(b*,**w***)* that provides a small value for *g (b*,**w***)* since the larger this value the larger the squared error between the corresponding linearmodel and the data, and hence the poorer we represent the given data. Therefore, we aimto *minimize g* over the bias and weight vector in order to recover the best pair *(b*,**w***)*,which is written formally as

**

**Minimization of the Least Squares cost function**

To perform calculations, it will first be convenient to use the following more compact notation:



With this notation we can rewrite the cost function shown in terms of the single vector $\tilde{w}$ of parameters as



To compute the gradient of this cost we simply apply the chain rule from calculus, which gives



Using this we can perform gradient descent to minimize the cost. However, in this (rare) instance we can actually solve the first order system directly in order to recover a global minimum. Setting the gradient above to zero and solving for $\tilde{w}$gives the system of linear equations



**Predicting the value of new input data**

With optimal parameters $(b^{\*},w^{\*})$, found by minimizing the Least Squares cost, we can predict the output *y* new of a new input feature **x** new by simply plugging the new input into the tuned linear model and estimating the associated output as



This is illustrated pictorially on a toy dataset for the case when *N* = 1





A simulated regression dataset where the relationship between the input feature *x* and the output *y* is not linear. However, because we can visualize this dataset, we can see that there is clearly a structured nonlinear relationship between its input and output. Our knowledge in this instance, based on our ability to visualize the data, allows us to design a new feature for the data and formulate a corresponding function (shown here in dashed black) that appears to be generating the data.