# Machine Learning

6 września 2022

## 1 Linear Regression

### 1.1 Method of Least Squares

We have some measurements (data) concerning single objects - single measurements consists of several variable. Precisely we have a set of numbers of form

$$M_j = (y, x_1, ..., x_n)$$

Our goal is to predict value of $y$ knowing all $x_i$ values. We look for function $f$ such that

$$y = f(x_1, ..., x_n)$$

Simplest case: we assume that f is linear, that is to say:

$$y = a_0 + a_1x_1 + ... + a_nx_n$$

We want to find coefficients $a_i$ which will make the predictions most precise. To do that, we shall try to minimize squared error of a model on train data. Train data consists of $p$ observations of form $M_j$. Squared error of single $M_j$ is equal to

$$\varepsilon_j = \left| y - a_0 - \sum_{i=1}^{n} x_i a_i \right|^2$$

Obviously we're going to find minimum error of sum of $\varepsilon_j$ from 1 to $p$. To do that one needs to find $a_i$ where derivative $\frac{d}{da_i} \sum \varepsilon_j$ is equal to zero for every $a_i$.

Let $X$ be a matrix that in $k$-th row and $m$-th column has $x_m$ from observation $M_k$. Actually that's a lie. Let $X$ be that matrix **plus** we add column of ones on the left. I turns out that coefficients $a_i$ that minimize sum of errors, are well and uniquely defined.

$$(a_0, a_1, ..., a_n) = \left(X^T X\right)^{-1} X(y_1, ..., y_p)$$

**Problem:** Its linear.

We can improve accurateness by adding more variables. Instead of looking just for coefficients that stay with $x_i$ we may add hundreds variables which are actually functions of $x_i$ such as $x_i^2$ or $\sin(x_i)$. Math remains the same.

**Problem:** With hundreds or thousands variables we can fit training data almost perfectly. Or to be precise: overfit.

## 1.2 LASSO Method, Ridge Regression and Elastic Net

Once again: in linear regression we're minimizing residual sum of squares (RSS) of form:

$$RSS = \sum_{j=i}^{p} \left| y - a_0 - \sum_{i=1}^{n} x_i a_i \right|^2$$

With the use of following methods we are minimizing similar expression, but with penalty correlated with number and value of coefficients.

**LASSO**
We minimize:

$$RSS + \lambda \left| \sum_{i=1}^{n} a_i \right|$$

**Ridge Regression** We minimize:

$$RSS + \lambda \left( \sum_{i=1}^{n} a_i \right)^2$$

**Elastic net**
We minimize:

$$RSS + \lambda_1 \left| \sum_{i=1}^{n} a_i \right| + \lambda_2 \left( \sum_{i=1}^{n} a_i \right)^2$$

How to finds lambdas? About that later. Firstly little overview.
**Features of LASSO**:

- It zeroes some coefficients.
- When to coefficients are highly correlated LASSO will probably zero one of them.
- Not bad when little nonzero coefficients.

**Features of Ridge Regression**:

- It does not zero coefficients - one must to add cutting level.
- Not bad when many nonzero coefficients.

**Features of Elastic Net**:

- It combines good features of both LASSO and Ridge Regression.

OK, cool. So how to find lambdas?

## 2 Cross Validation

Each choice of lambdas leads directly to some model. So instead of asking which lambdas to choose, we can ask which model to choose.

**First idea:** We divide data in two parts. We teach multiple models on first part and then test their quality on second part. Problem: we lose data.
**Second idea:** We divide data in $k$ parts (we call them *folds*). We choose one fold and teach model on all the others. Then we test it on that one and calculate the square error. Then we choose second fold and again the same. After repeating this procedure $k$ times we take a mean of all $k$ square error - that's validation error. And we simply choose the model, which has lowest validation error. That's cross validation. Usually we choose $k = 5$ or $k = 10$. Drawbacks? Time.

## 3 Logistic Regression

In linear regression we predicted values a continuous variable. In logistic regression we deal with a discrete variable of only two values (typically 0 and 1). Example: we have numerical data about people and we want to know, if they pay back the credit (1) or not (0).

Let's consider similar data frame as previously: $p$ observations of $n$ variables. Firstly we create an linear model of form:

$$y = a_0 + a_1 x_1 + ... + a_n x_n,$$

where $y$ in train data is equal to 0 or 1. By now $y$ may be arbitrarily large or small. To fix it we set

$$P = \frac{e^{a_0 + a_1 x_1 + ... + a_n x_n}}{1 + e^{a_0 + a_1 x_1 + ... + a_n x_n}}.$$

Now we have $P \in (0, 1)$ - nice! We can interpret $P$ as probability of an accident (like paying back credit) to happen. If we have already found $a_i$, we can predict, if an event happens or not. To do that we must arbitrarily set a barrier: say, if $P > 0.5$ then we assume that it'll happen, in other case we assume the opposite. Of course, using cross-validation we can test other barrier that 0.5.

How to find $a_i$? We achieve by maximizing on teach data likelihood function given by:

$$L = \prod_{y_i=1} P_i \cdot \prod_{y_j=0} (1 - P_j)$$

Unfortunately there is rather no analytical way to find $a_i$. Numerical methods must be used.

## 4 Principal Components Analysis (PCA)

The main point of PCA is to reduce number of variables while keeping as many "information" as possible. It is done by creating new variables equal to linear combinations of the old ones. New variables are chosen to have maximum possible variance. We standardize all the variables to have mean equal to 0 and variance equal to 1.

Let $X$ be matrix of $n$ rows (observations) and $p$ columns (variables). Now we take matrix of covariance $nX^T X$ - on the diagonal of this matrix we have variance of each variable. If we change variables to some other orthonormal variables, then on the diagonal of $X^T X$ once again we have variance of those variables. But there exists a theorem which says, that the greatest element of diagonal is at most equal to the greatest eigenvalue. So, to find variable of greatest variance, we need to find eigenvalues of $X^T X$ and later corresponding eigenvectors. Eigenvector with greatest eigenvalue is the very first principal component. Eigenvector with second greatest eigenvalue is second principal component. Etc.