Machine Learning 2015: Project 1 - Regression Report

Ivaylo Toskov
itoskov@student.ethz.ch

Maximilian Wurm
mwurm@student.ethz.ch

Martina Rivizzigno
rimartin@student.ethz.ch

October 31, 2015

Experimental Protocol

This is the protocol of the group "I don't care".

1 Tools

For our approach we used Python in combination with numpy and the machine learning library "scikit-learn". Like this, Python offers an API comparable to Matlab.

2 Algorithm

After trying out several algorithms we finally stick with the Random Forest Estimator of scikit-learn. We used Random Forest Regressor which is a perturb-and-combine technique. It operates by constructing multiples decision trees in the training process and outputs the mean regression of the individual trees. The deeper the tree becomes, the more complex the decision process will be, and the better the data will be fitted. By using this approach, heavy preprocessing of the data is not mandatory.

3 Features

Features are preprocessed performing MinMax Normalization. As far as features transformation is concerned, we tried to add dimensions to perform polynomial regression and to study the correlation between pairs or triplets of features and the output but we were not able to improve our model. Furthermore we noticed that several dimensions have values which are a power of two. A continuous value space can be achieved by applying the logarithm to the basis of two. This approach gave us the best output. Given fourteen dimensions, it is reasonable to assume that some of them are not relevant or redundant. Furthermore features selection enhances the generalization properties of the model by reducing over fitting. The function SelectKBest estimates the influence of the single dimensions. Then it drops all, but the k leading ones. Like this we got the best results.

4 Parameters

The parameter estimation is of course an important part for the random forest algorithm. Based on decision trees, the results can be unstable because small changes can end up in a completely different decision tree. To get a better overview, we first tried the method with a small set of parameters. We quickly recognized that the sequential

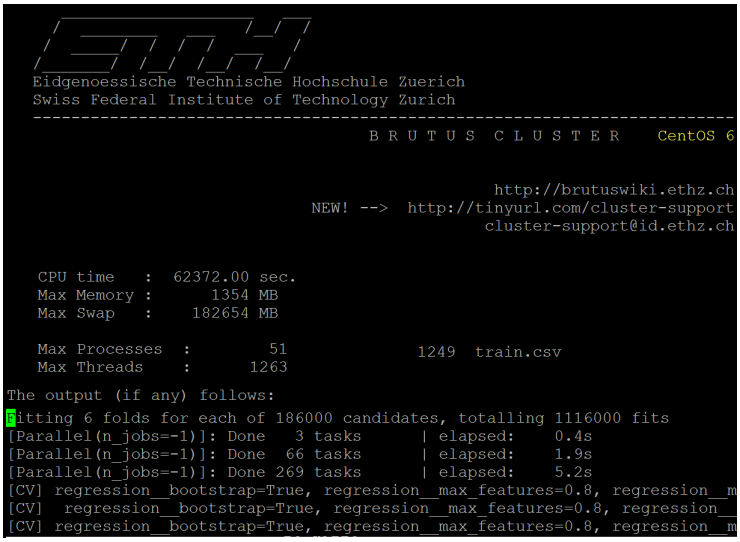

Figure 1: BRUTUS cluster

recompilation and execution of the test, would take an enormous amount of time. Therefore the program has been optimized to enable a higher performance. Scikit-learn GridSearchCV function runs a cross validation based, fitting-performance test over all combinations of possible parameters, which are given in a separate list. The cross validation makes it possible to immediately estimate the precision of a certain configuration. The whole process from Feature Selection, over transformation to the fitting of the regressors can be combined within a single "Pipeline". Like this, the GridSearch can not only be applied to the selection of regressor parameters, but also to the selection of the amount of interesting features. To make it possible to search a huge grid, performance is an important issue. Due to the fact, that all grid executions are independent, parallelization is fortunately easily possible (and also supported by the scikit implementation). To take the most out of this fact, the easily accessible BRUTUS cluster of ETH 1, was a welcome possibility to execute the script. The best result has been then choosen as submission.

5 Lessons Learned

In the beginning of the project we implement the different linear regressors mentioned in the lectures without using libraries. Then we switch to library functions and the RandomForestSelector gave the best results. Nevertheless we think, that in the end, linear methods could outperform the decision tree approach if the inherent structure of the data would be much more involved into the fitting process. Preprocessing of the given data has a paramount importance. Since the set of possible values of each feature is limited it would be possible to extract multiple dimensions for each attribute having only 0 or 1 values. For example feature 1 comprises the following values:

$$x_1 = [2, 4, 6, 8]$$

. It would be transformed in four dimensions:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Unfortunately, there is not enough time to implement this but we think it would definitely improve the final result.