Machine Learning 2013: Project 1 - Regression Report

Diego Ballesteros (diegob@student.ethz.ch)
Jonas Nick (jnick@student.ethz.ch)
Rastislav Starkov (starkovr@student.ethz.ch)

October 31, 2013

Experimental Protocol

1. Observe the data and analyze how the input features interact with the response variable (e.g. plotting individual curves) and identify outliers.
2. Scale the data to fulfill the preconditions of some algorithms and obtain more stable numerical performance.
3. Select the different tools and algorithms, e.g. libsvm, Matlab, R, mboost.
4. Try the algorithms by using greedy backward feature selection with linear features and use some intuition and trial and error to identify possible transformations or important features. E.g. from linear regression we could identify some dominant features to use in the GAM regressor. Measure the performance using cross-validation. In addition use hypothesis testing to get rid of superfluous variables.
5. Identify the best tool and optimize its performance through cross-validation, use this as the final predictor.

1 Tools

For the final solution, the language used was R with the following packages: stats, caret, party and mboost. RStudio was the IDE used during the development, testing and prediction.

2 Algorithm

For the final submission the method used was conditional inference trees with boosting. Conditional inference trees are a regression tool that allows for non-linear regression based on statistical significance of the input features and the response variable, this method is presented in [3]. The main idea is to perform statistical tests on every split of the tree until the null hypothesis of independence between the response and the most influential variable can not be rejected, and continue the splitting by choosing the most influential variable on the response (e.g. determined by its p-value).

Even though the conditional inference trees provide a good result, there was room for improvement and this was done by using gradient boosting on the trees. Boosting is a useful learning technique, introduced by Schapire in [4], which allows "weak" learners to produce a strong learner by aggregating them after training each one on weighted versions of the original training data. In our case, the "weak" or base learner is the conditional inference tree.

For the implementation we used the blackboost function from the mboost package [1]. Additionally, we scaled the input and output data to improve the numerical performance of the algorithms.

3 Features

For the final solution we used all the features because the conditional inference trees and boosting provide feature selection, the first one does it based on statistical significance of the covariates with the response and the second one by weighting the base learners based on the error on the training set.

4 Parameters

We performed cross validation on the number of boosting iterations to prevent overfitting. This was conveniently implemented in the cvrisk function from the mboost package in R. The cross-validation performed was 10-fold and the range of the boosting iterations was from 1 to 1000, the cost in this case was the RMSE.

For the conditional inference tree we used the default parameters from the package because the boosting algorithm works better with weak learners [2] so it is not recommendable to optimize the base learner through cross-validation.

5 Lessons Learned

We tried several other methods before trying the conditional inference trees, these were:

- Ridge regression: Without input transformation or with the input variables log transformed, this performed poorly with CV(RMSE) of the order of 0.5 in 10-fold cross-validation in the training data. The results were barely above the easy baseline on the validation set.
- ϵ-SVR: With a Gaussian kernel, for this learner we performed cross-validation to obtain optimal C and γ, for this the CV(RMSE) in 10-fold cross-validation was 0.36 which was also the cost in the validation set. This was discarded since it only achieved the easy baseline.
- Generalized linear models with boosting: This did not perform much better than the simple ridge regression, in this case boosting did not significantly improve the performance of the linear base learners.
- Generalized additive models: This was the second best learning method during our tests, in this case we did some analysis of how the input features influenced the response by looking at the information from the previous learners and determined a additive model as follows

$Delay \leftarrow s(RFSize) + BranchesAllowed + te(Depth, fL2Ucache)$[1] the cost during 10-fold cross-validation was 0.2 and the cost in validation was 0.17.

References

[1] Bühlmann, P., and Hothorn, T. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science 22*, 4 (2007), 477–505. with discussion.

[2] Freund, Y., and Schapire, R. E. Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)* (1996), L. Saitta, Ed., Morgan Kaufmann, pp. 148–156.

[3] Hothorn, T., Hornik, K., and Zeileis, A. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics 15*, 3 (2006), 651–674.

[4] Schapire, R. E. The strength of weak learnability. *Mach. Learn. 5*, 2 (July 1990), 197–227.

[1]Where s denote splines, a sum of basis functions estimated by the GAM. The te term specifies a non-linear interaction between two variables (a tensor product)