Machine Learning 2013: Project 1 - Regression Report

anufer@student.ethz.ch
elmerl@student.ethz.ch
nivo@student.ethz.ch

November 1, 2013

Experimental Protocol

Usage:
Download the csv files to /data/....csv (... = training, testing, validation)
Run linear_regression_with_model.m
Results are in /data/....out (... = training, testing, validation) The model used is stored in /data/model/*

1 Tools

- Matlab (code is in /code directory)
- Git / Github Repository [1]
  - Final solution [2] (tag best_v3, master branch, code attached in the zip file)
  - Experimental branch [3] with more feature variation and stochastic gradient decent (algorithm is slow, branch monte_carlo_superexperimental)
  - Nonlinear regression approches [4] (e.g. with neural networks)

2 Algorithm

The algorithm we used has evolved during the project. Basically there are four lage evolution steps. Basic Algorithm Ridge regression with k-Fold cross validation with k=10. Use all features linearly and manually add some features. Problem: Manual try and error. Determination and Elimination of "bad" training rows We observed that some training data could be corrupt, and thus this would impact the linear regression negatively. Therefore the following rows were excluded from the learning dataset: 48 88 94 132 302 187. Automatic Model Selection Using Greedy Selection We define a set of possible

[1]https://github.com/lukaselmer/ethz-machine-learning
[2]https://github.com/lukaselmer/ethz-machine-learning/releases/tag/best_v3
[3]https://github.com/lukaselmer/ethz-machine-learning/tree/monte_carlo_superexperimental
[4]https://github.com/lukaselmer/ethz-machine-learning/tree/experimental

features that can be used for the model. The model itself is represented as a binary matrix where an element in the model-matrix defines if the corresponding feature is used in the model or not. The first step in the algorithm then is to create the input data using the data read from the file and adding the additional features according to the model matrix.

The greedy selection works as follows: The algorithm starts with an empty model (zero matrix). Then it finds the best model with exactly one element set to one. To find it, the algorithm simply tries every possible combination and selects the one with the best error. In the second iteration, the algorithm finds the best model with 2 elements set to one, based on the model found in the first iteration. The algorithm goes on until the desired number of features are found.

Automatic Model Selection Using Monte Carlo Selection The downside of the greedy algorithm is that it can be trapped in local minima. For example, if two features in combination would be great, the greedy algorithm can miss this combination.

Therefore, a monte carlo algorithm was developed. Even though the next feature is selected greedily, after every iteration there is a small chance that a random feature is dropped. Additionally, there is a parameter which controls how many features the model can contain at most (experiments show that 10 features is a good tradeoff for fitting and generalization). If the amount of features is reached, then some random features are dropped. Thus, many new combinations are possible.

To not loose the best found model, the best model is stored in every iteration a new lowest error is found. Additionally, to help converge to a good fit, there is a small probability that the current model is set to the best model.

3 Features

The features found by the algorithm vary with each training. The submitted solution contains following features:

$$y = w \cdot [x_5^3, x_{14}^2 + x_{14}^3, x_{10} \cdot x_5, x_{13} \cdot x_{14}, x_5/x_3, x_{10}/x_4, x_{14}/x_5, x_3/x_{10}, x_{13} \cdot x_5^2, x_{14} \cdot x_{13}^2, x_{13} \cdot x_{10}^3, x_{14} \cdot x_{13}^3]$$

4 Parameters

As described in section 2, the parameters for the model are learned. There are some other parameters tough:

- # of features in feature map: 12 (can be combined, see section 3)
- Hyper parameter for ridge regression: 0.25, evaluated by experiments
- # of for-loops it takes for learning (with monte carlo): 25, can be more, but may lead to overfitting
- Probabilities for decision based on random values / monte carlo: educated guess and experiments

5 Lessons Learned

The automatic approach saved us a lot of time of manual testing. Even the first very simple automatic selection got us a good result which encouraged us to go on with further optimizations. Further improvements The algorithm could be improved further, but unfortunately time is about to be up. We figured out that the following things would improve the results:

- Normalization: Currently, normalization is not implemented. This would improve the ridge regression a lot.
- More complex feature map / more complex transformations / parametrized transformations: Currently, only simple transformations are implemented (e.g. $log(x1 + x2^2)$ should be $log(a * x1 + b * x2^2)$). This would slow down the algorithm tough, because more parameters would have to be learned.
- Stochastic gradient decent instead of greedy gradient decent (to speed up the algorithm)
- Use simulated annealing to improve algorithm speed, avoid local minimas and make the algorithm more solid.