

# EMC Israel Data Science Challenge

## Team ULjubljana's Solution

Members of team ULjubljana  
`jure.zbontar@fri.uni-lj.si`

September 20, 2012

## 1 Introduction

We solved the challenge by training a set of base learners and combining them with stacking. Our final score was 0.19812 which was enough to claim second place in the competition.

## 2 Data Preparation

We found data preparation to be a very important component in our solution. At the end, we settled on using *tf-idf*. Smoothing (similar to Laplace smoothing<sup>1</sup>) and sublinear term frequency scaling (take the logarithm of every term count) were also applied. The exact method that we used can be found in the function `tfidf` in the file `main.py`.

Prior to training, we used basic feature selection, where we removed the features with document frequency less than `n` (`n` was a parameter of feature selection). We tried different values of `n` for different models.

## 3 Methods

### 3.1 L2-Regularized Logistic Regression

We used our own implementation of L2-regularized logistic regression, implemented in Python. We removed features with document frequency less than 10 and added an artificial constant feature with value 1. In order to extend logistic regression to the multi-class case, the binary relevance method was used, i.e., we trained a separate model for each of the 97 classes, performing one-vs-all classification on each run. Logistic regression took around 15 minutes to run<sup>2</sup>.

---

<sup>1</sup>[http://en.wikipedia.org/wiki/Additive\\_smoothing](http://en.wikipedia.org/wiki/Additive_smoothing)

<sup>2</sup> All timings were performed on an Intel i7-3770 3.4 GHz processor, with 16 GB of RAM.

## 3.2 K-Nearest Neighbour

We implemented our own k-nearest neighbour (KNN) algorithm using the cosine similarity measure. A slight modification of KNN was used, which is best described on an example. Assume that the three ( $k = 3$ ) closest documents belong to classes 1, 3, and 1 and that their similarity with the query document is 0.9, 0.5, and 0.4, respectively. The predicted probabilities in this case are  $[0, 0.9 + 0.4, 0, 0.5, 0, 0, \dots, 0] + 0.001$ <sup>3</sup>.  $eps = 0.001$  is a smoothing parameter. The bigger the  $eps$ , the more each probability is pulled towards  $1/97$ . The vector of predictions was normalized to sum to 1. As with logistic regression, we remove features with document frequency less than 10. The number of neighbours was set to 17. KNN took 25 minutes to run.

## 3.3 Softmax Regression

Softmax regression is a generalization of logistic regression to the multi-class classification setting and was implemented following the description on [http://ufldl.stanford.edu/wiki/index.php/Softmax\\_Regression](http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression). As always, we removed features with document frequency less than 10. Softmax regression is memory intensive. It needs around 14GB of RAM and 30 minutes to run.

## 4 Stacking

We combined the prediction of the described methods with stacking. 5-fold cross-validation was used to obtain out-of-sample predictions for each of the three base learners on all the training samples. These predictions were subsequently used as training data for an artificial neural network, which produced a blend of the predictions of the base learners. We used our own implementation of artificial neural networks. We noticed that stacking improved if we added 100 components of latent semantic indexing<sup>4</sup>. Latent semantic indexing was implemented with the help of the SVDLIBC library<sup>5</sup>. Stacking was the most time consuming component. It needed 7 hours to run.

## 5 Results

Table 1 presents the scores of our submissions, from our earliest attempts with the score of 0.29837, to our final submission with score 0.19812.

---

<sup>3</sup>Note that the rightmost plus sign is overloaded and should be interpreted as adding a constant to every element of the vector.

<sup>4</sup>Latent semantic indexing is nothing more than principal component analysis without the preprocessing step of mean centering the data.

<sup>5</sup><http://tedlab.mit.edu/~dr/SVDLIBC/>

Method	Score
Logistic Regression	0.29837
Stack: Logistic Regression	0.26025
Stack: Logistic Regression, KNN	0.22721
Stack: Logistic Regression, KNN, Softmax	0.20912
Stack: Logistic Regression, KNN, Softmax, LSI	0.19812

Table 1: This table presents scores of our submissions. The word *stack* means that the outputs were passed to a neural network for stacking as described in this report. Note that it also makes sense to send the output of a single learner through the stacking procedure, as can be seen from the logistic regression example in the table.

## 6 How To Generate the Solution

The source code can be downloaded from an online repository at <https://bitbucket.org/jzbontar/emc>. First, unpack the training and test sets into the directory `data_orig`. The directory `data_orig` should contain three file:

- `test_data.csv`
- `train_data.csv`
- `train_labels.csv`

To train the models and generate the solution run the command `run.sh`. The predictions will be stored in a file named `submission.csv.gz`.

### 6.1 Conclusion

In this report we presented our approach to the EMC Israel Data Science Challenge. The method required 8 hours to run and was placed second in the competition.

From Table 1 we conclude that stacking has an enormous impact on score on this particular dataset. The best score obtained by any single method was 0.29837. With stacking, we were able to improve to our final score of 0.19812.