KDD-Cup 2004: Protein Homology Task

Winner's Report: RKL Measure

Dirk Dach Holger Flick Christophe Foussette Marcel Gaspar Daniel Hakenjos
Felix Jungermann Christian Kullmann Anna Litvina Lars Michele
Katharina Morik Martin Scholz Siehyun Strobel Marc Twiehaus Nazif Veliu

1.1 Question

Two datasets were provided, one for training and another for testing. The training set contained 145,751 and the test set consisted of 139,658 examples. Each example could be identified by an unique example id and was mainly made up of 74 numerical feature values. These values described the match between the native protein sequence and the sequence that is tested for homology. Furthermore, every example could be associated with this native protein sequence using the so-called block id. Each block embodied about 1,000 examples. Both datasets did not contain any incomplete information. The main goal was to predict which proteins were homologous to a native sequence. For this purpose the test set included a label which declared each example as homologous or inhomologues. Described here is the winning solution to minimize the average rank of the lowest ranked homologous sequence (RKL). Thus, this was not a classification task, but a ranking task. RKL does not depend on the predicted values, only on the relative order of the matches within each block.

1.2 Approach

BlockNearestNeighbor

The data inspection mainly focused on the analysis of the 153 blocks that were given in the training dataset. From this we learned that there were a lot of blocks having only a little number of positive examples. In addition, just some blocks contained up to 50 positive examples. Due to this, we decided to analyse the numerical feature values for every block in the training set. For this we used statistical measures[1] and discovered that these were different [1].

One possibilty to consider these discrepancies between the blocks is to quantify them. A nearest neighbor method is able to achieve this using distance measurement. Thus, each block had to be transformed into a vector representation. For this we used the statistical measures. Our *BlockNearestNeighbor* method can be characterized with the following steps:

1. In order to predict all examples in a block, it locates the k-nearest blocks with their vector representation.

2. Learn the ranking model on the examples of this k-nearest blocks using the Ranking SVM [4].

3. Predict all examples in the block applying the ranking model.

Before the whole learning process the datasets have been normalised[2]. To assess the importance of the numerical attributes, J4.8 decision trees were learned with Yale using Weka [5, 6] for all blocks and the frequency of each attribute being a node in a tree was calculated. Using these frequencies a feature selection of the attributes for the vector representation was carried out. We reached our best results only including the attributes that occured at least twice. It was of utmost importance to find an appropriate value for k. A value of $k = 15$ yielded the best results. In addition to this, the learning option j of the SVM^{light} [3], which does denote the cost factor by which training errors on positive examples outweight errors on negative examples, had to be optimized.

SVM Classification

In [2] predicting protein homology is mentioned as an application for a classification SVM with RBF kernel[3]. This led us to train a global SVM classifier using the RBF kernel on the normalised data. Further, we conducted cross-validation experiments in order to optimize the parameters γ and j of the SVM^{light}. Since we faced a ranking problem, we chose function values of the SVM

[1]Minimum, maximum, mean and median.

[2]Every normalisation in this work was made according to: $x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$.

[3]The RBF kernel is defined as $K(\mathbf{x}, \mathbf{y}) = e^{-\|\mathbf{x}-\mathbf{y}\|^2/2\gamma^2}$.

as the predicted ranking instead of the class assignment. During training the SVM penalizes misclassified instances proportionally to their distance to the separating hyperplane. This justifies the assumption that examples far away from the separating hyperplane are less frequently misclassified. The function values are real numbers, directly implying an order on the examples.

Final Model

Our final model combined the two methods described above. These output rankings given in real numbers. In order to combine them properly we normalised these two rankings. As a method for combining, simply adding them did the trick, according to cross-validation experiments.

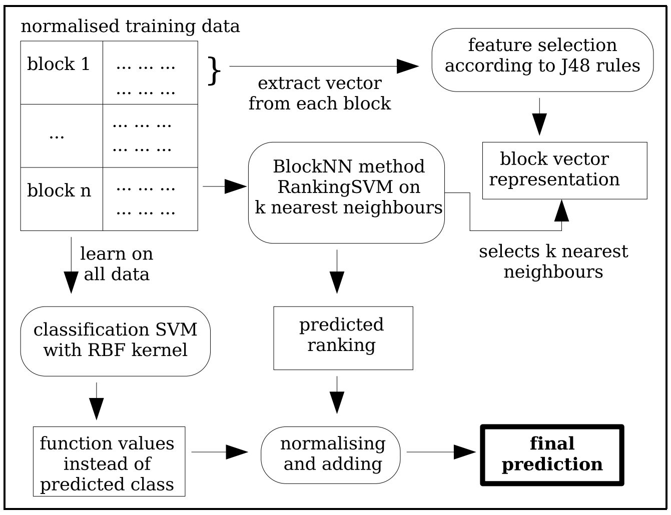
normalised training data
block 1
... ...
...
block n
extract vector from each block
feature selection according to J48 rules
block vector representation
BlockNN method RankingSVM on k nearest neighbours
selects k nearest neighbours
learn on all data
classification SVM with RBF kernel
predicted ranking
function values instead of predicted class
normalising and adding
final prediction

Figure 1: Schematic representation of our approach.

Conclusion

We implemented an ensemble learning approach by combining two SVM predictions. On the one hand the ranking SVM delivered directly a ranking, on the other hand the classification SVM implied a ranking taking advantage of the function values. We think that our method of combination (simple adding) could be improved when using other weighting techniques.

References

[1] Richard A. Becker, John M. Chambers, and Allan R. Wilks. The New S Language. Chapman & Hall, London, 1988.

[2] Cristianini, N. and Shawe-Taylor, J., An Introduction to Support Vector Machines and other kernel-based learning methods, Cambridge Press, 2000.

[3] T. Joachims, Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning, B. Schlkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[4] T. Joachims, Optimizing Search Engines Using Clickthrough Data, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD), ACM, 2002.

[5] Mierswa, Ingo and Klinkberg, Ralf and Fischer, Simon and Ritthoff, Oliver. A Flexible Platform for Knowledge Discovery Experiments: YALE – Yet Another Learning Environment. In LLWA 03 - Tagungsband der GI-Workshop-Woche Lernen - Lehren - Wissen - Adaptivitt, 2003.

[6] Ian H. Witten and Eibe Frank, Data Mining: Practical machine learning tools with Java implementations, Morgan Kaufmann, San Francisco, 2000