KDD-Cup 2004: Protein Homology Task

Winner's Report: RKL Measure

Christophe Foussette
AI Unit
Dep. of Computer Science
University of Dortmund
44221 Dortmund, Germany
foussett@ls8.cs.uni-dortmund.de

Daniel Hakenjos
AI Unit
Dep. of Computer Science
University of Dortmund
44221 Dortmund, Germany
daniel.hakenjos@cs.uni-dortmund.de

Martin Scholz
AI Unit
Dep. of Computer Science
University of Dortmund
44221 Dortmund, Germany
scholz@ls8.cs.uni-dortmund.de

ABSTRACT

In this paper we describe the winning model for the performance measure "lowest ranked homologous sequence" (RKL). This was a subtask of the Protein Homology Prediction task of the KDD Cup 2004. The goal was to predict protein homology for different performance metrics. The given data was organized in blocks, each of which corresponds to a specific native sequence. The two metrics average precision (APR) and RKL explicitly make use of this block structure. Our solution consists of two parts. The first one is a global classification SVM not aware of the block structure. The second part is a *k-NearestNeighbor* scheme for block similarity, used to train ranking SVMs on the fly. Furthermore, we sketch our approach to optimize the root-mean-squared-error and report some alternative solutions that turned out to be suboptimal.

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 a native protein sequence and a 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 inhomologous.

Described here is the winning solution to minimize the average rank of the lowest ranked homologous sequence (RKL). For the average prediction metric (APR) we submitted the same predictions as for RKL, and were ranked fourth.

Both metrics have in common that they score rankings. This means they do not depend on predicted values, but only on the relative order of the matches within each block. Each block is evaluated separately. The overall scores for RKL and APR are computed by averaging.

For each example of the test set a real-valued prediction should be made, which can be thought of as a confidence score or probability of examples to be homologous. For each block these confidence-rated predictions induce an ordering. The RKL-value per block is the lowest rank (highest number) of an homologous example in the ordered list of examples, so this measure basically punishes missed homologous examples.

The average precision metric is based on the precision of different subsets. Starting with the empty set, in each iteration the next remaining example most confidently predicted to be homologous is added to the subset, until all examples are covered. The precision is computed each time and averaged to receive the average precision per block.

We submitted another model for the root-mean-squared-error (RMS), also briefly described in this paper. This metric measures the squared difference between a predicted value in the interval $[0, 1]$ and the true label from the set $\{0, 1\}$.

2. METHODOLOGY

We mainly used the SVM as the basic learning algorithm, a classification SVM and a ranking SVM. They are described in the next two subsections. The last subsection introduces to the *BlockNearestNeighbor* method, a *k-NearestNeighbor* approach considering the dissimilar behaviour of the blocks.

2.1 Classification SVM

In [2] predicting protein homology is mentioned as an application for a classification SVM with RBF kernel[1]. This led us to train a global SVM classifier using the RBF kernel on the normalised data. Since we faced a ranking problem, we chose function values of the SVM 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.

2.2 Ranking SVM

The performance measures APR and RKL depend on the ordering of the samples. The ranking SVM [4] addresses di-

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

rectly a ranking problem by learning - in the linear case - a weight vector which ranks the samples best according to Kendall's τ, a common measure from statistics for the similarity of two rankings. The ranking SVM performed well on the given ranking problem but performance improved using it as base learner in the *BlockNearestNeighbor* described in the next section.

2.3 BlockNearestNeighbor

BlockNearestNeighbor is a straight-forward way to couple a nearest neighbor approach with an arbitrary base learner. The underlying assumption is, that blocks sharing common statistical properties are more likely to behave similarly when it comes to prediction. As common in nearest neighbor approaches the similarity between blocks is defined by a distance function. The goal is to estimate the similarity between blocks, so unlike *k-NearestNeighbors* the distance function is not estimated over single examples, but over information describing blocks. In our experiments we used the Euclidian distance over simple block-wise aggregates like the median of specific attributes.

Given a previously unseen block only the k most similar blocks according to the distance function are used as a training set. The resulting model is used to predict the label of the new block.

3. APPROACH

This section describes our approach for the APR and RKL measure and for the RMS measure respectively. First the preprocessing of data is explained. Then the final models are specified followed by a section addressing the prevention of overfitting. Finally we describe some alternative approaches we evaluated.

3.1 Preprocessing

We discovered that normalization[2] of the attributes improved performance according to cross-validation experiments. Further enhancements were achieved by normalizing the attributes within each block instead of global normalization.

Since we implemented a *k-NearestNeighbor* approach we needed a distance measure between blocks. We used statistical measures, actually mean, median, minimum and maximum, for each attribute in a block. With 74 original attributes this led to a 296 dimensional vector. Having 153 blocks in the training set, we wanted to reduce the dimension of the vector. We trained J48 decision tree classifiers [6] on each block and counted the occurences of a specific attribute in these trees. Then we selected those attributes occuring at least in two different trees. With this method we reduced the number of attributes to 46, hence the dimension of the vector to 184. Our distance measure was the Euclidian distance between these 184 dimensional vectors representing a block.

3.2 Model for RKL and APR

Our final model for the performance measures APR and RKL is schematically shown in figure 1. Consider a previously unseen block b. On the normalized training data we learned a global model, a classification SVM with RBF kernel. This model can directly be applied to block b without further training.

[2] $x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$

On the other hand the *BlockNearestNeighbor* approach selects the set of blocks most similar to block b. A ranking SVM is trained on this set. Afterwards the resulting model is also applicable to block b.

Both models assign a real number to each example, thus define an ordering over b. In order to combine them properly we normalized these two rankings. As a method for combining, simply adding them did the trick, according to cross-validation experiments.

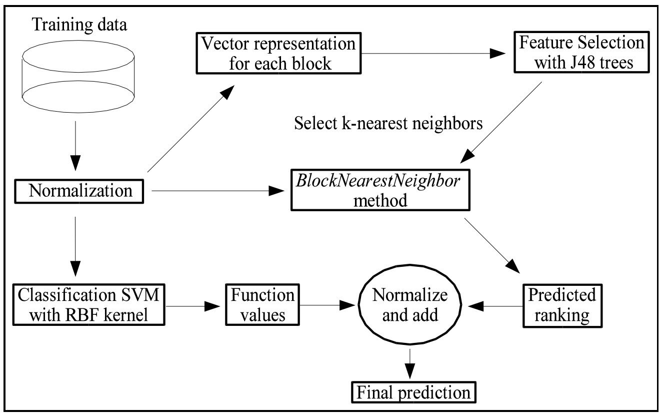
Training data
Vector representation for each block
Feature Selection with J48 trees
Select k-nearest neighbors
Normalization
BlockNearestNeighbor method
Classification SVM with RBF kernel
Function values
Normalize and add
Predicted ranking
Final prediction

Figure 1: Schematic representation of our approach.

3.3 The RMS model

Cross-validation experiments indicated, that it pays off to train a separate model for the RMS metric. With less than 1% positives the data was highly skewed. The fine-tuned SVM classifier for RKL and APR served as a good starting point, just suffering from a low recall of about 60%. In contrast, boosted decision trees showed a little bit higher accuracy, a much higher recall of about 75%, but a significantly lower RMS value.

An analysis of those cases in which boosting and SVM models disagreed encouraged to combine them by a simple disjunction. In the resulting ensemble an example is predicted homologous, if at least one of these two models considers it to be homologous.

The boosting algorithm used in our experiments was standard AdaBoost [7], the decision tree induction algorithm was J48. The ensemble of the SVM classifier with boosted trees was ranked 14th.

An experiment that finished shortly after submission deadline indicated that AdaBoost was not the optimal choice. The *BayesianBoosting* algorithm works by directly estimating probabilities. One of its properties is to keep the priors of class labels, when changing example weights during training. It is not yet published, but available as part of the free learning environment *YALE*[3]. Compared to the corresponding AdaBoost models, the models of this operator achieved a significantly higher recall (close to 80%) while not losing accuracy.

The upload facility of the KDD Cup website re-opened after the contest. We were able to evaluate the alternative model, a combination of an SVM classifier with BayesianBoosting on top of J48 decision tree classifiers. This model would have been ranked fourth.

[3] http://yale.cs.uni-dortmund.de

3.4 Prevention of overfitting

The classification SVM with RBF kernel has two parameters which can be optimized. One parameter, called j in SVM^{light} [3], is the factor by which false-negatives are penalized higher than false-positives during training. We expected this to be greater than 1, because positively labeled instances were outnumbered by the negative instances. The other optimizable parameter is γ, the bandwith of the RBF kernel. We conducted tenfold cross-validation in order to optimize these parameters. From these experiments we got an estimation for the RKL measure of 44.34, using the classification SVM stand alone, and 37.13 when combining it with the *BlockNearestNeighbor* method. The performance on the test set was 45.62.

To estimate the predictive performance of *BlockNearestNeighbor* we trained a model for each block of the training set. Because we left out the block itself the result was a leave-one-out estimate, where "one" relates to blocks rather than examples. This estimate was used to optimize the parameters, namely k, the number of neighbors to be selected for training, and j, a parameter of the ranking SVM.

The estimate for the combination of the two models, however, was too optimistic. The estimated RKL value due to a tenfold cross-validation was 19% lower than the final evaluation on the validation set, for APR the value was 1.8% too low.

3.5 Alternative approaches

Two promising approaches to tackle the problem that turned out to be inferior to the solution finally selected are shortly described in this subsection. We basically focussed on ways to overcome the striking differences between blocks.

First of all we tried to perform clustering as a step of preprocessing. The goal is to find a few disjoint sets of similar blocks, each of which could then be assigned a model of its own. Candidate clustering algorithms were *k-NearestNeighbor* and Diez clustering[8]. The Diez clustering algorithm is an agglomerative clustering algorithm using the performance of SVMs as its joining criterion. It starts by selecting the most trivial clusters. In our case each block forms a cluster. On each cluster a separate SVM is trained. The pair of most similar SVMs according to a fixed distance function[4] is the first candidate for being joined. If the estimated performance of the SVM trained on the temporarily formed cluster is better than the performance of the single models before, the candidates are joined and become a new cluster. Otherwise the next two similar candidates are tested. The algorithm stops when no more canditates remain. Although this approach exploits the similarity of blocks, just like the winning solution, in our experiments it performed worse than just training a ranking SVM on the original data set.

The second approach adds blocks to the training set sequentially. Starting with an initial set of a few blocks a first ranking SVM model is trained. Then the model is applied to the remaining blocks, to find the subset of blocks for which the model is least appropriate. These blocks are added to the training set of the next iteration. Another ranking SVM is trained on this larger example set, and so on. The number of iterations and the number of blocks to add are parameters that can be optimized empirically. The advantage of this approach over the winning solution is that the output is a single global model, rather than a lazy learner, postponing calculations to the time when a request for predictions arises. Iteratively augmenting the training set yielded the most accurate *global* model in our experiments. It is not part of the submitted solution, because it was clearly outperformed by the *BlockNearestNeighbor* approach.

[4]In [8] it is the cosine between the weight vectors.

4. CONCLUSION

In this paper we outlined our approaches to the protein homology task of KDD Cup 2004. The common goal of all metrics associated to that tasks was to predict homology of a protein sequence in terms of a real-valued confidence. The focus of this paper is on our model for the two ranking metrics, namely lowest ranked homologous sequence (RKL) and average prediction (APR). For the RKL metric our submission was ranked first, for APR the same predictions were ranked fourth.

An early analysis of the data revealed, that statistical properties varied drastically from block to block. With less than 1% of the data being homologous sequences the data was highly skewed. Although under these conditions one would prefer block-wise modeling, it was possible to fit a global classifier surprisingly well. A classification SVM with RBF kernel gave best results in our cross-validation experiments. Still block-related modeling gave significantly better results. The ranking SVM is directly tailored towards optimization of ranking measures. Starting with a small subset of blocks and adding new blocks for which predictions are yet poor, iteratively, was a successful way to train accurate block-related models. Our final model is still different, because it directly accounts for the statistical differences of blocks by applying a *k-NearestNeighbor* algorithm at the block level. To predict new blocks it is necessary to identify the k most *similar* blocks. Only for these blocks a ranking SVM is trained, which is then applied to the new block. This method worked surprisingly well, keeping in mind the simple similarity measure we used. The disadvantage of this method compared to the iteratively trained ranking SVM is that for each block to be classified a new model needs to be trained on the fly.

For all metrics we made the experience that without combining models the results are not compatible. But combining models seems easy since even simple approaches in this direction gave a boost in accuracy. The final model for RKL and APR is a result of simply adding the confidence values of the global classification SVM and the more local *BlockNearestNeighbor* ranking SVM. Similarly, a simple disjunction of the global SVM model and boosted decision trees was still ranked 14th, after more careful selection of the boosting algorithm even ranked 4th. Considering more sophisticated methods of model combination would probably further increase performace.

Finally we want to point out that in our experiments the performance of different learners and parameter settings was quite metric dependent. As a result performance could almost always be increased by optimizing parameters for each metric separately. The main reason for us to still submit a single model for RKL and APR was lack of time.

APPENDIX

A. ADDITIONAL AUTHORS

We participated at KDD Cup 2004 in the framework of a "project group", a course over two semesters for 12 graduate students. The course was organized by Prof. Katharina Morik and Martin Scholz, AI Unit, Dep. of Computer Science, University of Dortmund, Germany.
The full list of participating students and co-authors: Dirk Dach, Holger Flick, Christophe Foussette, Marcel Gaspar, Daniel Hakenjos, Felix Jungermann, Christian Kullmann, Anna Litvina, Lars Michele, Siehyun Strobel, Marc Twiehaus, and Nazif Veliu.

B. 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. Schoelkopf 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 - Adaptivitaet, 2003.

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

[7] Y. Freund and R. Schapire. *A decision–theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences, 55(1): 119 – 139, 1997.

[8] J. Diez, J.J. del Coz, O. Luaces, and A. Bahamonde. *A Clustering Algorithm to Find Groups With Homogeneous Preferences.*