A Block-Based Support Vector Machine Approach to the Protein Homology Prediction Task in KDD Cup 2004

Yan Fu[1,2], Ruixiang Sun[1], Qiang Yang[3], Simin He[1],
Chunli Wang[1], Haipeng Wang[1], Shiguang Shan[1], Junfa Liu[1], Wen Gao[1]

[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080, China

{yfu, rxsun, smhe, clwang, hpwang, sgshan, jfliu, wgao}@jdl.ac.cn

[2]Graduate School of Chinese Academy of Sciences, Beijing 100039, China

[3]Department of Computer Science, Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong, China

qyang@cs.ust.hk

ABSTRACT

This paper describes our solution for the protein homology prediction task in KDD Cup 2004 competition. This task is modeled as a supervised learning problem with multiple performance metrics. Several key characteristics make the problem both novel and challenging, including the concept of data blocks and the presence of large-scale and imbalanced training data. These features make a naive application of the traditional classification algorithms infeasible. Our approach focuses on making full use of the abundant information within the blocks, and developing a new technique for reducing and balancing training data to make the support vector machine applicable to this kind of large-scale and imbalanced learning tasks.

Keywords

Data mining, Bioinformatics, Support vector machine, Data reduction, KDD Cup

1. INTRODUCTION

This paper introduces our support vector machine (SVM) approach to the protein homology prediction task in the KDD Cup 2004 competition. This task is modeled as a supervised learning problem with multiple performance metrics, namely, Top 1 (maximize), average last rank (RKL, minimize), root mean squared error (RMSE, minimize) and average precision (APR, maximize). The detailed description of this task can be found in a companion paper written by the organizers in the same issue. In this section, we highlight several novel and challenging characteristics of the task, which motivated our approach to the task.

After carefully examining the training data, we have found that this dataset is different from the traditional ones used for training data-mining models. First, the data are grouped into blocks with each one corresponding to a hidden native protein sequence. Each block includes approximately a thousand examples. The training and test data consist of 153 and 150 blocks, respectively. Compared to traditional classification problems, the partition of the training data into blocks is a novel concept. How to make use of this additional information to improve the predictive ability of the classifier has become the central issue for us. We have tried a large number of methods, and finally settled on the technique of intra-block data normalization, which we will describe later.

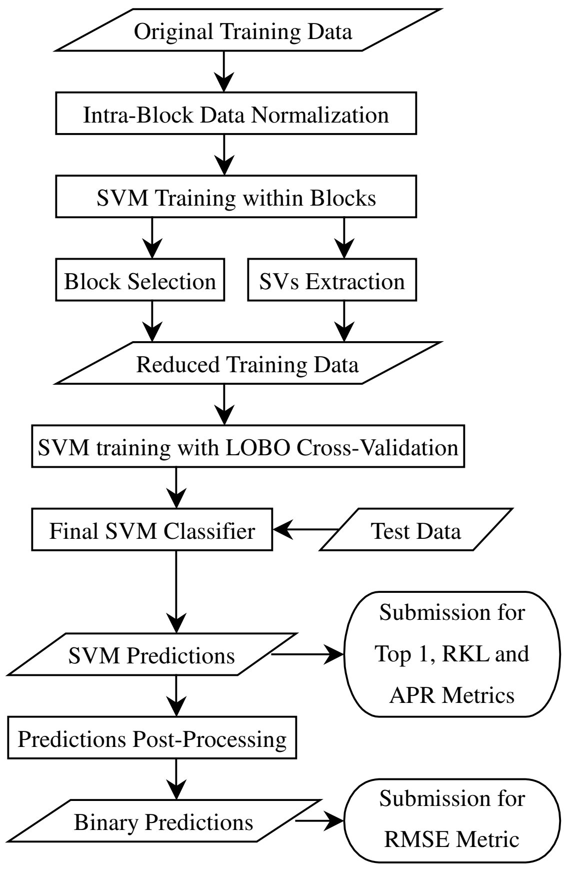
Original Training Data
Intra-Block Data Normalization
SVM Training within Blocks
Block Selection
SVs Extraction
Reduced Training Data
SVM training with LOBO Cross-Validation
Final SVM Classifier
Test Data
SVM Predictions
Submission for Top 1, RKL and APR Metrics
Predictions Post-Processing
Binary Predictions
Submission for RMSE Metric

Figure 1. An overall flowchart of our approach

A second challenge that we recognized is the large amount of training data; specifically, the 153 blocks of training data contain 145,751 examples in total. Although the technique of the SVM [1] has been well established in theory [2] and has been successfully applied to many practical classification problems [2, 3, 4] in the past, the large-scale SVM training is still a challenging problem in the fields of data mining and machine learning. To make the SVM suitable for this task, we tried two ways to reduce the data size. One is block selection, in which only the blocks ranked higher in terms of predictive ability or "usefulness" in prediction are selected for later processing. The other is an approximate method for identifying support vectors, in which an SVM is trained for each block respectively and the support vectors in all blocks are extracted as the new training data.

The last characteristic of the dataset is the imbalance between positive and negative examples; this is typical of some previous KDD Cup datasets. Among the 145,751 training examples, only 1,296 (0.89%) are positives. Fortunately, this problem can be overcome as the data size is reduced, as we will describe below.

An overall flowchart of our approach is presented in Figure 1. In the rest of the paper, we first describe our methods in the stages of data preprocessing, data reduction, predictions post-processing, and cross-validation. Then, we give the final prediction results on both training and test datasets. Finally, we conclude the paper with some discussions.

2. INTRA-BLOCK NORMALIZATION

In the data given in this task, the features in a block are not the traditional physical measurements but are the attributes that measure the matches between all returned sequences and the native sequence corresponding to that particular block. The values of a feature in one block are related to the corresponding native sequence and thus may be different in distribution from those of the same feature in blocks related to other native sequences. For example, it is possible that the values of a feature in a block tend to be larger or smaller values than those in another block, as demonstrated in Figure 2.

This implies that the i.i.d. (independent and identical distribution) condition, on which most machine learning algorithms are established, may not be satisfied across blocks. This problem can result in poor generalization ability of a classifier if the block information is ignored at all.

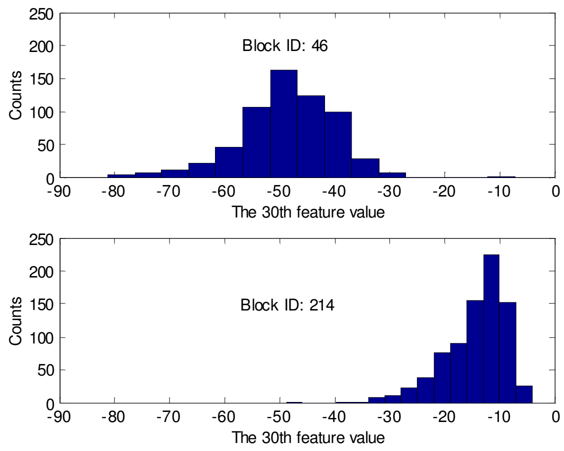

Figure 2. Different distributions of values of the 30th feature in two blocks: ID 46 and 214. The values of this feature in the Block 46 are mostly below –30 whereas those in the Block 214 are mostly above -30.

Our solution for this issue was to normalize the data in each block by taking into account the distribution differences among blocks. Instead of using a global normalization over all the examples in all blocks, we used the intra-block data normalization, in which features were normalized within each block such that the means were zero and the standard deviations were one within the block. Compared to the global normalization, the intra-block normalization remarkably improved the prediction performance. Table 1 shows the experimental comparison of these two kinds of normalization methods, both using the linear SVM as the classifiers.

Table 1. The prediction results of linear SVMs trained on the globally normalized data and the intra-block normalized data

	Top 1	**RKL**	**APR**	**RSME**
Global normalization	0.8497	54.7843	0.8033	0.0373
Intra-block normalization	0.8758	51.9412	0.8305	0.0367

Note that in this and other experiments below, the SVM-Light package[1] [5] was used. The prediction performance on training data was evaluated by the leave-one-block-out cross-validation, which we will introduce in Section 5.

3. BLOCK SELECTION

In supervised learning, the outliers in the training data can misguide the classifier and thus do harm to the generalization ability of the learned model. We wish to remove this effect in our task. Generally, the detection of outlying samples is difficult in machine learning. However, as the given dataset is grouped into blocks, the detection of outlying blocks is relatively easier than the general case. Each block as a group of samples can be used to train a classifier. Then each block-based classifier can be used to predict the samples in other blocks. This produces a score for each block according to its prediction performance on other blocks. In this way, the blocks with inferior predictive abilities can be identified and removed.

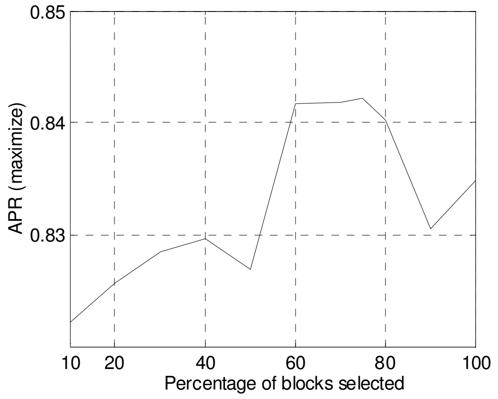

Figure 3. APR performance versus percentage of blocks selected as training data according to their abilities to predict other blocks under the Top 1 metric

In our practice, we trained an SVM on each block and used the learned model to predict all other blocks in the training data. The

[1] http://svmlight.joachims.org

predictive ability of each block could then be assessed by any one of the four performance measures given in this task. By selecting those blocks with superior predictive abilities as the final training data, the computational efforts for training was significantly reduced and meanwhile the performance was remarkably improved. Figure 3 plots the relationship between the APR performance and the percentage of blocks selected as training data according to the Top 1 metric. In this experiment, the data from each selected block were also reduced with the strategy introduced in the next section. The Radial Basis Function (RBF) kernel was used for SVM training.

It is shown in Figure 3 that when between 60 and 80 percent of blocks were used for training, the APR performance was significantly (0.7%) improved compared to the case when all blocks were used.

4. SUPPORT-VECTOR DATA REDUCTION

Although the training data are reduced to some extent via block selection, the data size is still too large to train an SVM classifier efficiently, especially when nonlinear kernels are used. In training an SVM, one needs to resolve a quadratic optimization problem with size *m*, the number of the training examples. Off-the-shelf optimization packages typically work on the entire $m \times m$ Gram matrix, and thus can quickly become intractable as a huge amount of training examples are involved in a learning task. Some techniques [5, 6, 7] have been proposed to decrease the complexity of the training algorithm. However, training an SVM on one hundred thousand examples in our case still poses a severe problem. Experiments showed that training the SVM-Light [5] on tens of thousands of examples was extremely slow when nonlinear kernels were used. What if we used the linear SVM? Our experiments showed that the linear SVM trained on the entire data had only limited prediction performance.

To take full advantage of the SVM, the amount of training data must be reduced. Our observation is that not all data need be used. In fact, among the 145,751 samples in training data, the negative examples take up 99.11% due to the imbalanced nature of the data. We decided to remove the redundant negative examples for double benefits: to reduce the amount of training data in order to improve efficiency, and to control the effect of imbalanced data. Although only a small proportion of training examples, i.e. support vectors (SVs), contribute to the construction of the final SVM classifier, there is no efficient method that can precisely identify the SVs in general. We used a simple method to approximately identify the SVs from the large number of training examples.

Our motivation is based on the assumption that the SVs obtained by training SVMs on the subsets of the training data should cover most of the SVs obtained by training a single SVM on the entire training data. The number of SVs, which are extracted from subsets of the original training data, can be easily controlled by the parameters used to train SVMs on the subsets. For example, a larger gamma value in the RBF kernel corresponds to a smaller radius in the radial basis function and will generally lead to more SVs. Figure 4 plots the relationship between the number of SVs extracted from subsets and the value of gamma in the RBF kernel.

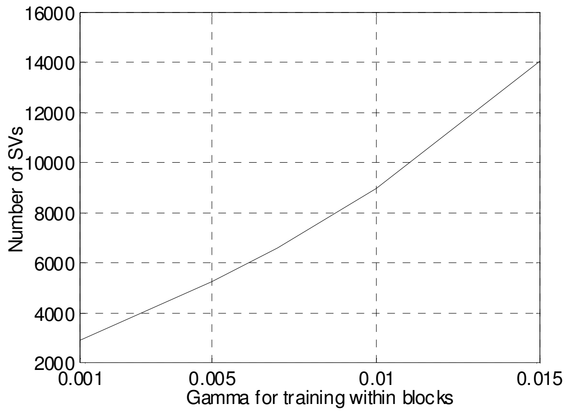

Figure 4. Relationship between the number of SVs extracted from subsets of original training data and the value of gamma in the RBF kernel used for SVM training

In our implementation, we trained an SVM classifier on each block respectively and extracted the SVs in all blocks. The negative SVs and all positive examples were used as the final training data. In this way, the training data were further reduced from about one hundred thousands to several thousands, which at the same time solved the class imbalance problem. Therefore, the SVMs on this synthesized training data could be successfully trained at a high speed with nonlinear kernels. As a result, such parameters as those in kernels and the percentage of selected blocks could be finely tuned for better performance.

Table 2 compares the prediction results obtained by training linear and nonlinear SVMs on the entire training data, selected blocks, and negative SVs plus all positives in selected blocks, respectively.

Table 2. Prediction results obtained on original training dataset and on two reduced training datasets

Training data	Kernel	Top 1	RKL	APR	RSME
All (145,751 examples)	Linear	0.8758	51.9412	0.8305	0.0367
	RBF	Training failed			
Selected blocks (109,547 examples)	Linear	0.8758	53.2941	0.8380	0.0360
	RBF	Training failed			
Negative SVs plus all positives in selected blocks (7,178 examples)	Linear	0.8758	52.5359	0.8314	0.0359
	RBF (gamma = 0.0004)	0.8954	50.5948	0.8426	0.0353

As illustrated in Table 2, the reduction of training data did not significantly influence the prediction performance of linear SVMs. However, training linear or nonlinear SVMs on the greatly reduced data was much faster than training linear SVMs on the entire training data. Most importantly, the nonlinear SVM trained on the final reduced dataset remarkably improved the prediction performance compared to linear SVMs.

5. PREDICTIONS POST-PROCESSING

Although we could use four different classifiers for each of the four metrics in our prediction, the testing on the training set has shown the possibility to use only a single classifier for good performance on all four metrics. We paid special attention to the RMSE measure because unlike the other three measures (Top 1, RKL, and APR), it depends on the predicted values. In contrast, the other three measures can be determined based on the ranking of samples within each block. Therefore, the predicted values produced by an SVM should be post-processed so as to be more suitable for the RMSE metric.

We tried three approaches for this task. The first was most straightforward, in which predicted values were normalized into the interval between 0 and 1. In the second approach, we applied the sigmoid function to the predicted values so that the majority of the predicted values approached to 1 or 0. In the third approach, we discretized the predicted values into binary values 0 or 1 by thresholding. In our training experiments, given a set of predictions the thresholding approach always led to the best result. Continuing in this direction, we have also tried different values for the threshold, and found that the value of zero was the best.

6. LEAVE-ONE-BLOCK-OUT (LOBO) CROSS-VALIDATION

In order to compare all the methods we tried in various stages and adjusting the parameters involved, an assessment on the generalization abilities of the trained classifiers is necessary. We extend the traditional cross-validation method "leave-one-out" to a form that we call Leave-One-Block-Out (LOBO).

Since the performance measures are applied to each block and then averaged over all blocks, it becomes natural to partition the original training data by blocks for cross-validation. Given a training dataset, the LOBO cross-validation puts the examples in a block aside as a validation set at a time, and uses examples in all other blocks for training. After all blocks have their turns, an averaged measure is computed at the end. Note that our LOBO cross-validation is different from the traditional leave-one-out cross-validation, in which every example constitutes a validation set. It is thus akin to a special kind of n-fold cross-validation resulting from the special concept of data block. Experiments on the test data later showed that the LOBO cross-validation successfully prevented overfitting.

The penalty parameter *C* in setting the SVM is a very important parameter, because it controls the tradeoff between the training error and the margin. The SVM-Light package [5] did an excellent job at setting the default value for this parameter. We observed that other manually determined values of this parameter in fact led to worse performance of the SVM when compared to the default one.

7. RESULTS

The percentage of selected blocks, the types of the kernels used for extracting SVs within blocks and training the final SVM classifier, as well as the parameters in kernels were experimentally optimized by the LOBO cross-validation.

Table 3 presents the prediction results of our final SVM classifier on the training set and the test set. The performance on the training set was assessed by the LOBO cross-validation. Our results on the test data occupied the first place for the APR and RMSE metrics, second for the Top 1 metric, fourteenth for the RKL metric. We were tied with other two teams for first place overall, among all 59 participants.

Table 3. Prediction performance of the final SVM classifier

	Top 1	**RKL**	**APR**	**RSME**
On training set	0.8954	50.5948	0.8426	0.0353
On test set	0.9133	54.0867	0.8412	0.0350

It can be seen from Table 3 that overfitting has been successfully prevented except for the RKL on test data, which result is slightly worse than that on the training data. Among the four performance measures, RKL is more sensitive to the ranks of individual examples. For a low RKL, all positive examples in a block must be ranked as high as possible. In this sense, the RKL is relatively unstable. For example, a decrease of 300 places in the predicted rank of the last positive example in a test block will result in a decrease of two in the RKL for this block, whereas the influence on the APR or RMSE could be very little if there are many other positive examples in this block.

8. DISCUSSIONS

Data block is a unique and novel concept in this classification problem. Our success on this problem is derived from our efforts to dig out the extensive information contained in blocks using data normalization, data reduction and LOBO cross-validation. However, the road to our final methods was not straightforward. From the beginning to the end, we have made a lot of attempts to make sure that using the block information is beneficial to the predictive ability of a classifier. Below, we briefly mention several failed but interesting attempts.

The first idea that came into our minds was to predict the unknown block using its similar blocks. To do this, we designed a distribution-overlap-based similarity measure between blocks. A second attempt was to use intra-block statistics such as the means and variances of the features in each block to augment the feature vectors. Given that the training data were readily grouped into blocks, a bagging approach was tried in a third attempt, in which SVMs were trained separately on each block and then were combined together with some strategies. In another interesting attempt, we used a genetic algorithm to optimize the linear discriminate functions with each of the four performance metrics directly as the fitness functions. Although these attempts were all aborted after experiments showed that they suffered one way or another, we consider them to be good directions to pursue

Training SVMs on a large-scale and imbalanced dataset is both challenging and novel in data mining and machine learning research. In our task, by extracting support vectors from subsets of the training data, the data were reduced and balanced at the same time. To the best of our knowledge, this is a novel approach to making the SVM training tractable on large-scale as well as imbalanced data. In the general case when the training data are not given in subsets, such as blocks in our task, some re-sampling techniques can be tried to generate subsets. The general framework of reducing training data for the SVM can be described as the following steps:

1) Generate subsets of the original training dataset using re-sampling techniques, such as random partition or bootstrap.
2) On each subset, train an SVM and extract the support vectors. Adjust the kernel parameter, e.g. gamma in the RBF kernel, to control the number of extracted support vectors.
3) Combine all the support vectors obtained in step 2) as the new training data.
4) Train the final SVM classifier on the reduced training data obtained in step 3).

In this framework, the reduced training dataset is expected to be an approximation to the set of support vectors that would be obtained if the SVM were trained on the entire training data. However, to what extent the SVM trained in this way will approximate to the one trained on the entire training data needs both theoretical and experimental analyses. This will be our future work.

9. ACKNOWLEDGMENTS

This work was supported by the National Key Basic Research & Development Program (973) of China under Grant No. 2002CB713807. Qiang Yang is supported by a Hong Kong RGC Grant HKUST 6180/02E. We thank Rong Pan, Peng Yang, Yiqiang Chen, Hongming Zhang, Jingfen Zhang and Mingjie Zhang for valuable discussions.

10. REFERENCES

[1] Cristianini, N. and Shawe-Taylor, J. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press, 2000.

[2] Vapnik, V.N. The Nature of Statistical Learning Theory. NY: Springer, 1995.

[3] Joachims, T. Learning to Classify Text Using Support Vector Machines: Methods, Theory, and Algorithms. Kluwer, 2002.

[4] Schölkopf, B., Tsuda, K. and Vert J.-P. (eds.), Kernel Methods in Computational Biology. MIT Press, 2004.

[5] Joachims, T. Making large-Scale SVM Learning Practical. In Schölkopf, B., Burges, C. and Smola, A. (eds), Advances in Kernel Methods: Support Vector Learning. MIT Press, 1999.

[6] Osuna, E., Freund, R. and Girosi, F. An Improved Training Algorithm for Support Vector Machines. In Island, A. (ed), IEEE NNSP, 1997.

[7] Platt, J.C. Fast Training of Support Vector Machines Using Sequential Minimal Optimization. In Schölkopf, B., Burges, C. and Smola, A. (eds), Advances in Kernel Methods: Support Vector Learning. MIT Press, 1999.

About the authors:

Yan Fu is a Ph.D student in the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS). His research interests are bioinformatics, data mining and machine learning. Currently, he is mainly focusing on proteomic data mining. He is expected to obtain his Ph.D degree in computer science in 2005.

Ruixiang Sun is an associate professor in ICT, CAS. His research interests include bioinformatics, pattern recognition, and evolutionary computation.

Qiang Yang is an associate professor in the Computer Science Department of Hong Kong University of Science and Technology. His research covers planning, case-based reasoning, data mining and machine learning with applications to Web and wireless problems, etc. He acted as the major coach of this KDD Cup team.

Simin He is an associate professor in ICT, CAS. His research interests include design, analysis and application of combinatorial optimization algorithms.

Chunli Wang is a postdoctoral scholar in ICT, CAS. Her research interests are pattern recognition and machine learning.

Haipeng Wang is currently a Ph.D student in ICT, CAS. His research interests include data mining, statistical learning theory, and bioinformatics. Now he is specializing in computational problems in proteomics.

Shiguang Shan received his Ph.D degree in computer science from ICT, CAS in 2004. His research interests include pattern analysis and machine intelligence, image and vision computing, and human–computer interface. Currently, he is mainly focusing on face recognition related researches.

Junfa Liu is a Ph.D student in ICT, CAS. His research interests are speech analysis, data mining and statistical learning.

Wen Gao is a professor in ICT, CAS. His research interests are in the areas of signal processing, image and video communication, computer vision, artificial intelligence and data mining. He is another coach of the team.