Query-Adaptive Ranking with Support Vector Machines for Protein Homology Prediction*

Yan Fu[1], Rong Pan[2], Qiang Yang[3], and Wen Gao[4]

[1] Institute of Computing Technology and Key Lab of Intelligent Information Processing, Chinese Academy of Sciences, Beijing 100190, China
[2] School of Information Science and Technology, Sun Yat-sen University, Guangzhou 510275, China
[3] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China
[4] Institute of Digital Media, Peking University, Beijing 100871, China
yfu@ict.ac.cn, panr@mail.sysu.edu.cn, qyang@cse.ust.hk, wgao@pku.edu.cn

Abstract. Protein homology prediction is a crucial step in template-based protein structure prediction. The functions that rank the proteins in a database according to their homologies to a query protein is the key to the success of protein structure prediction. In terms of information retrieval, such functions are called ranking functions, and are often constructed by machine learning approaches. Different from traditional machine learning problems, the feature vectors in the ranking-function learning problem are not identically and independently distributed, since they are calculated with regard to queries and may vary greatly in statistical characteristics from query to query. At present, few existing algorithms make use of the query-dependence to improve ranking performance. This paper proposes a query-adaptive ranking-function learning algorithm for protein homology prediction. Experiments with the support vector machine (SVM) used as the benchmark learner demonstrate that the proposed algorithm can significantly improve the ranking performance of SVMs in the protein homology prediction task.

Keywords: Protein homology prediction, information retrieval, ranking function, machine learning, support vector machine.

1 Introduction

A good ranking function is crucial for a successful information retrieval system [1]. A ranking function is based on the measurement of the relevance of database items to a query. Usually, there are multiple ways to measure the relevance of

* This work was supported by the Research Initiation Funds for President Scholarship Winners of Chinese Academy of Sciences (CAS), the National Natural Science Foundation of China (30900262, 61003140 and 61033010), the CAS Knowledge Innovation Program (KGGX1-YW-13), and the Fundamental Research Funds for the Central Universities (09lgpy62).

J. Chen, J. Wang, and A. Zelikovsky (Eds.): ISBRA 2011, LNBI 6674, pp. 320–331, 2011.

database items. An important issue is how to automatically and intelligently combine these relevance measures into a powerful single function using machine learning technologies [2,3,4,5].

Protein structures play an important role in biological functions of proteins. Experimental approach to protein structure determination is both slow and expensive. Since homologous proteins (evolved from the same ancestor) usually share similar structures, predicting protein structures based protein homologies has been one of the most important problems in bioinformatics [6,7,8,9]. Protein homology prediction is a key step of protein structure prediction and is a typical ranking problem[10]. In this problem, the database items are protein sequences with known three-dimensional structures, and the query is a protein sequence with unknown structure. The objective is to find those proteins in the database that are homologous to the query protein so that the homologous proteins can be used as structural templates.

The homology between two proteins can be captured from multiple views, such as sequence alignment, sequence profile and threading [11]. In this paper, we will not focus on these homology measures or features, but on the machine learning algorithms that integrate these features into a single score, i.e., a ranking function, in order to rank the proteins in a database. Since the proposed algorithm is, in principle, applicable to general ranking-function learning tasks, we will discuss the problem and describe the algorithm in a somewhat general manner. For example, when we say a 'query' or 'database item', it corresponds to a 'protein' in our protein homology prediction problem, and 'relevant'/'relevance' means 'homologous'/'homology'.

In ranking-function learning, the items (proteins in our case) in a database are represented as vectors of query-dependent features, and the objective is to learn out a function that can rank the database items in order of their relevances to the query. Each query-dependent feature vector corresponds to a query-item pair. Training data also consist of relevance (homology in our case) judgments for query-item pairs, which can be either absolute (e.g., item A is relevant, item B is not, while item C is moderate, etc.) or relative (e.g., item A is more relevant than item B). The relevance judgments can be acquired from the manual annotation of domain experts.

Algorithms for ranking-function learning mainly differ in the form of training data (e.g., absolute or relative relevance judgments), the type of ranking function (e.g., linear or nonlinear), and the way to optimize coefficients. In early years, various regression models were used to infer probability of relevance from binary judgments, e.g., the polynomial regression [3] and the logistic regression [12,13]. Information retrieval can also be viewed as a binary classification problem: given a query, classify all database items into two classes - relevant or irrelevant[2,14]. An advantage of viewing retrieval as a binary classification problem is that powerful discriminative models in machine learning, e.g., SVM, can be directly applied and the resultant ranking function is discriminative. Between regression and classification is the ordinal regression. Ordinal regression differs from conventional regression in that the targets are not continuous

but finite and differs from classification in that the finite targets are not nominal but ordered [15,16]. Methods were also proposed to directly learn to rank things instead of learning the concept of relevance. For example, Joachims addressed the ranking-function learning problem in the framework of large margin criterion, resulting in the Ranking SVM algorithm [5]. Learning to rank has drawn more and more attention from the machine learning field in recent years (e.g., [17,18]).

A major characteristic of ranking-function learning is that the feature vector of each database item is computed with regard to a query. Therefore, all feature vectors are partitioned into groups by queries (each group of data associated with a query is called a *block* in this paper). Unlike traditional learning tasks, e.g., classification and regression, in which data are assumed to be independently and identically distributed, the ranking data belonging to the same block are correlated via the same query. We have observed that the data distributions may vary greatly from block to block [19]. The same value of a feature may indicate relevance in one block but irrelevance in another block. In the pure ranking algorithms that take preference judgments as input and do not aim to estimate relevance, e.g., Ranking SVM [5], training is performed so that rankings are only consistent within training queries. In this case, the difference between queries does not pose an obstacle to learning a pure ranking function. However, few efforts have so far been devoted to explicitly making use of the difference between queries to *improve* the generalization performance of learned ranking functions. Since queries in practice differ in various ways, no single ranking function performs well for all queries. A possible way to improve ranking performance is to use different ranking functions for different queries [20].

In this paper, we describe a query-adaptive ranking-function learning algorithm for the protein homology prediction task. Our approach, called K-Nearest-Block Ensemble Ranking, is motivated by the intuitive idea of learning a ranking function for a query using its similar queries in training data instead of using all available training data. Note that by similar we do not mean sequence similarity, but rather similarity in distributions of query-dependent data. To avoid online training, we employ an ensemble method. On each data block (corresponding to a query protein) in training data, an individual ranking model is trained offline in advance. Given the data block derived from a new query, the k ranking models trained on the k most similar blocks to the given block are applied separately to the given block and the k groups of ranks are aggregated into a single rank. In this way, incremental learning is also supported. As support vector machines (SVMs) [21] have been extensively studied in recently years for ranking-function learning and have been shown to have excellent performance (see, e.g., [5,15,14,17]), we use the SVM as the benchmark learner in this work. Experiments on a public dataset of protein homology prediction show that the proposed algorithm performs excellently in terms of both ranking accuracy and training speed, significantly improving the ranking performance of SVMs.

2 Algorithm

In this section, we describe our K-Nearest-Block (KNB) approach to ranking. Below is the terminology used in this paper.

Query-Dependent Feature Vector. Given a query (a protein sequence here), a database item (a protein sequence with known structure) is represented as a vector of (query-dependent) features that measure the relevance (homology) of the database item to the query. Each query-dependent feature vector corresponds to a query-item pair.

Block. A block B is a group of instances of query-dependent feature vectors associated with the same query. Each block corresponds to a query. A block usually includes several hundreds or thousands of feature vectors, which are computed from the most homologous proteins in a database according to some coarse scoring function.

Training Block. A training block is a block with relevance judgements for all of the feature vectors in it.

Test Block. A test block is a block in which the relevance judgments are unavailable and are to be made.

Block Distance. A block distance $D(B_i, B_j)$ is a mapping from two blocks B_i and B_j to a real value that measures the dissimilarity between the two blocks.

k **Nearest Blocks.** Just as the name implies, the k nearest blocks to a block are the k training blocks that are most similar to this blocks according to a block distance definition.

The block structure of data is a unique feature of the ranking problem. We believe that the differences among blocks, if appropriately used, can be very valuable information for training more accurate ranking models. One straightforward idea is that given the test block corresponding to a new query, all n training blocks should not be used to learn a ranking model, but only the $k(\ll n)$ most similar training blocks to the test block should be used. This is the block-level version of the traditional K-Nearest Neighbors method for classification or regression and thus can be called the K-Nearest Block (KNB) approach to ranking.

Three important sub-problems in the KNB approach are:

1. How to find the k nearest training blocks?
2. How to learn a ranking model using the k nearest blocks?
3. How to choose the value of k?

Different resolutions to the above three sub-problems lead to different implementations of the KNB method for ranking. High speed is a most crucial factor for a real-world retrieval system. Generating a new ranking model for each new query seems to apparently conflict with the above criterion. Therefore, the second sub-problem is especially important and needs to be carefully addressed.

2.1 K Nearest Blocks (KNB)

To find the k nearest blocks to a given block, a distance between blocks must be defined in advance. In general, the block distance can be defined in various ways, either domain-dependent or independent.

The most intuitive way is to represent each block as a vector of block features. Block features are variables that characterize a block from some views. For example, block features can be the data distribution statistics of a block. Given a vector representation of blocks, any vector-based distance can serve as a block distance, such as Euclidean distance or Mahalanobis' distance.

For simplicity and generality, we employ a domain-independent vector presentation of blocks and use Euclidean distance in this paper. Each block is represented by the statistics of each feature in the block, that is,

$$\Phi(B_i) = \langle \mu_{i1}, \sigma_{i1}, \mu_{i2}, \sigma_{i2}, \cdots, \mu_{id}, \sigma_{id} \rangle, \tag{1}$$

where μ_{ik} and σ_{ik} are the mean and the standard deviation of the *k-th* feature in block B_i, respectively. The distance between two blocks B_i and B_j then is

$$D(B_i, B_j) = \sqrt{\parallel \Phi(B_i) - \Phi(B_j) \parallel^2}. \tag{2}$$

2.2 KNB Ensemble Ranking

Given the k nearest blocks, the next step is to learn a ranking model from the selected training data. The most simple resolution is to train a **global model** on the union of all selected blocks. However, as we pointed out previously, a fatal drawback of doing this is that a training process has to be conducted online for each new query while the training time to be needed is totally unknown. This is unacceptable for a practical retrieval system.

To overcome the difficulty of online training, we propose to use an **ensemble model**. First, on each training block, a ranking model (called a local model) is trained offline. All local models are saved for future use. When a new query comes, a test block is generated and is compared to all training blocks. The k nearest training blocks to the test block are identified and the k corresponding local models are separately applied to the test block. Then, the k groups of relevance predictions are aggregated together to generate the final single rank for the query. Figure 1 gives the flowchart of the KNB Ensemble Ranking method.

In principle, any ranking model can serve as the local model in the KNB Ensemble Ranking method. Since SVMs have recently been extensively explored for ranking, we choose the classification SVM as the base learner and compare the resulted KNB ensemble SVM with other SVM-based ranking methods. Another reason for using SVMs is that previous best results on the data set used in this paper (see next section for detail) were mostly obtained with SVMs [19,22,23,24].

For aggregation, we simply sum over all the predictions made by selected local models with block distances as weights; that is,

$$Relevance(B^*) = \sum_{i \in I^*} \frac{1}{D(B^*, B_i)} Model_i(B^*), \tag{3}$$

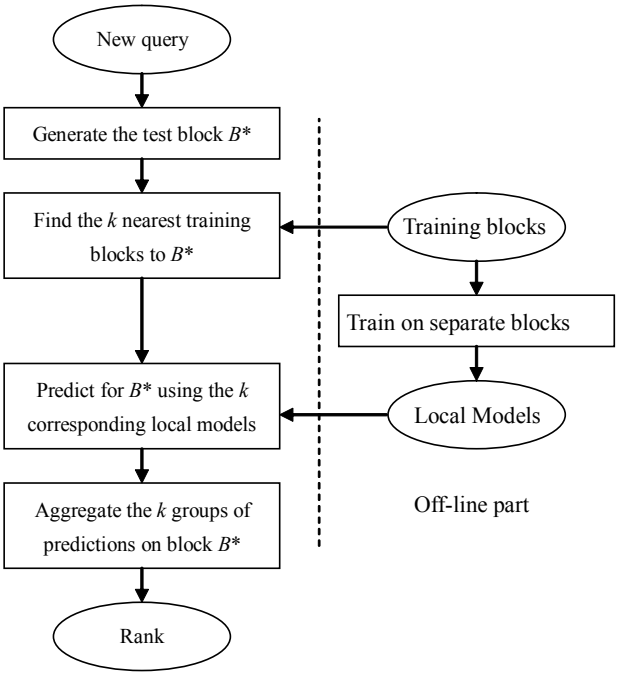

Fig. 1. Flowchart of the KNB Ensemble Ranking method

where I^* is the index of the k nearest training blocks to the test block B^* and $Model_i(B^*)$ denotes the predictions made by the local model trained on block B_i.

Compared to the global model, the ensemble model used in the KNB ranking method has several advantages:

- Firstly, training is fast. The time needed for training a learner (e.g. SVM) often increases nonlinearly with the number of training examples. Training on separate blocks is a divide-and-conquer strategy and thus is faster.
- Secondly, test is fast. KNB-based global model must be trained online for each new query, while local models can be trained offline in advance.
- Thirdly, incremental learning is supported. When a new training block comes, a new local model can be trained on it and be added into the repository of local models.
- Fourthly, a training block can be easily weighted according to its distance from the test block.
- Finally, an ensemble model often outperforms a global model in complex learning problems.

3 Experiments

In this section, we apply the KNB ensemble ranking algorithm to the protein homology prediction problem and demonstrate that the algorithm is superior to other SVM-based ranking methods in both ranking accuracy and training speed.

Table 1. Statistics of the protein homology data set

	#Queries	Block size	#Examples	#Features
Training data	153	~ 1000	145,751	74
Test data	150		139,658	

3.1 Data Set

Protein homology search is a routine task in current biology and bioinformatics researches. The task of protein homology prediction is to rank/predict the homologies of database proteins to a query protein. In this paper, we use the KDDCUP2004 data set of protein homology prediction [25]. In this data set, each database protein is characterized by 74 features measuring its homology to the query protein. The data are generated by the program LOOPP (Learning Observing and Outputting Protein Patterns), a protein fold recognition program [26]. The homology features include length of alignment, percentage of sequence identity, z-score for global sequence alignment, etc. [11]. On this data set, we have obtained the Tied for First Place Overall Award in the KDDCUP2004 competition. In the original winning solution, we successfully developed and used the intra-block data normalization and support-vector data sampling technologies for ranking-function learning [19].

The statistics of the data are summarized in Table 1. They include 153 training queries and 150 training queries. For each query, the examples (candidate homologous proteins) were obtained from a preliminary scoring/ranking function. The labels (homology judgments) are in binary form (0 for homology and 1 for non-homology) and are available for training data. Labels for test data are not published. The predictions for test data can be evaluated online at the competition web site. This provides a relatively fair manner for researchers to test and compare their methods.

For computing the block distance, we globally normalize the query-dependent features so that the mean is zero and the variance is one in the whole training data. For training local SVMs, we locally normalize the features so that the mean is zero and the variance is one within each block. We found that this kind of intra-block normalization resulted in improved prediction performance compared to the global normalization [19].

3.2 Performance Evaluation

Four metrics are used to evaluate the performance of a ranking method. They are TOP1, RKL (average rank of the last relevant item), APR (mean average precision), and RMS (root mean squared error). Each of the four metrics is first computed on individual blocks, and then averaged over all blocks.

TOP1. (maximize) TOP1 is defined as the fraction of blocks with a relevant item ranked highest. It measures how frequently a search engine returns a relevant item to the user at the top position.

RKL. (minimize) RKL is defined as the average rank of the last relevant item. It measures how many returned items have to be examined sequentially in average so that all relevant items can be found. If the purpose is to find out all relevant items, then RKL is a more suitable metric than TOP1.

APR. (maximize) APR is defined as the average of a kind of average ranking precision on each block. The average precision on single block is quite similar to the AUC (Area Under presision/recall Curve) metric. APR provides an overall measurement of the ranking quality.

RMS. (minimize) RMS is the average root mean square error. It evaluates how accurate the prediction values are if they are used as estimates of relevance (1 for absolute relevance and 0 for absolute irrelevance).

The first three metrics exclusively depend on the relative ranking of the items in a block while RMS needs relevance estimates. Since the target values are binary, we found that to a large extent RMS seems to rely on a good normalization of the prediction values more than on a good ranking or classification. Therefore, we place emphasis on the first three metrics, although we have obtained the best result of RMS on test data.

To evaluate a learning algorithm and perform model selection, cross validation is the most widely used strategy. Since the performance measures for ranking are calculated based on blocks, it becomes natural to divide the training data by blocks for cross validation. We extend the traditional cross validation method Leave-One-Out (LOO) to a block-level version which we call Leave-One-Block-Out (LOBO). Given a training data set, the LOBO cross validation puts one block aside as the validation set at a time and uses other blocks for training. After all blocks have their turns as the validation set, an averaged performance measure is computed at the end. The LOBO cross validation is different from both the traditional LOO and the n-fold cross validation. It is a graceful combination of these two common cross validation strategies, taking advantage of the block structure of ranking data.

3.3 Results

Four algorithms are compared, including the standard classification SVM, the Ranking SVM, the KNB global SVM, and the KNB ensemble SVM. For Ranking SVM, relative relevance judgments are derived from binary labels. In all experiments, the SVM^{light} package [27] is used. In most cases, the linear kernel is used for SVM training, based on the following considerations:

- High speed is crucial for a real-word retrieval system and linear ranking function are more efficient than nonlinear ones. Especially, a linear SVM can be represented by a weight vector while a nonlinear SVM has to be represented by a group of support vectors, the number of which is in general uncontrollable.
- Nonlinear kernels introduce additional parameters, thus increasing the difficulty of model selection. In our case, experiments have shown that training with nonlinear kernels, e.g., RBF kernel, on the entire data set we used is extremely slow and does not show better results.

Table 2. Cross-validation performance on training data set (the results of Standard SVM and Ranking SVM were obtained after intra-block data normalization, a method we previously proposed [19]; otherwise they would perform much worse. It is the same with Table 3).

Method	TOP1 (maximize)	RKL (minimize)	APR (maximize)	RMS (minimize)
Standard SVM	0.8758	51.94	0.8305	0.0367
Ranking SVM	0.8562	**36.83**	0.8257	N/A
KNB global SVM	**0.8889**	45.18	**0.8560**	**0.0354**
KNB ensemble SVM	**0.8889**	39.50	0.8538	0.0357

Table 3. Performance on test data set

Method	TOP1 (maximize)	RKL (minimize)	APR (maximize)	RMS (minimize)
Standard SVM	**0.9133**	59.21	0.8338	**0.0357**
Ranking SVM	0.9000	45.80	0.8369	N/A
KNB global SVM	0.9067	45.90	0.8475	0.0379
KNB ensemble SVM	0.9067	**40.50**	**0.8476**	0.0364

- In the KNB SVM ensemble approach, the linear local SVMs can be easily combined into a single linear function before prediction, thus decreasing the online computational burden. Moreover, it has been shown that in ensemble machine learning, a good generalization ability is often achieved using weak (e.g., linear) base learners rather than strong (e.g., nonlinear) ones.
- The use of linear kernel is fair for all the SVM-based methods compared.

The only parameter in the linear SVM is the parameter C, the tradeoff between training error and learner complexity. Another parameter in the KNB-based methods is k, the number of selected local models. To do model selection and performance evaluation on the training data, we use the LOBO cross-validation as described above. Table 2 gives the best results obtained using various methods. It shows that on the TOP1, APR and RMS metrics, KNB-based methods are superior to the other two methods. On the RKL metric, the Ranking SVM obtains the best result. On all metrics, the KNB ensemble SVM is comparable or superior to the KNB global SVM.

On the test data, predictions, made by models trained with the parameter values optimized for each metric, were evaluated on online. Table 3 gives the test results. The KNB ensemble SVM obtains the best results on the RKL and APR metrics among the four methods. In fact, they are the **best** known results on these two metrics (up to the time that this paper is submitted). On the other two metrics, KNB ensemble SVM does not perform best. However, the differences are very small and once again we argue that the RMS metric is very sensitive to the normalization of prediction values. On average, the solution of KNB ensemble SVM is the best result among all the original and subsequent submissions to the competition (http://kodiak.cs.cornell.edu/cgi-bin/newtable.pl?prob=bio).

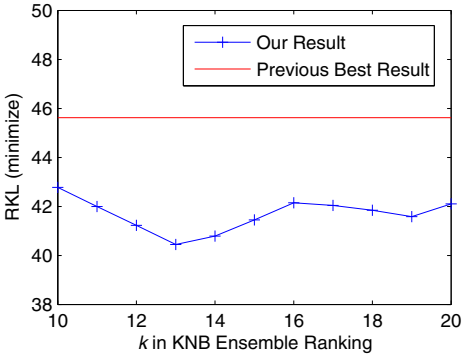

Fig. 2. RKL performance on test data set vs. k in KNB ensemble SVM in comparison with previous best result

Table 4. Training speed comparison

Method	Training mode	Training time (seconds)
Standard SVM	Offline	95
Ranking SVM	Offline	32255
KNB global SVM	Online	dependent on k
KNB ensemble SVM	Offline	**9**

It is also found that the KNB ensemble SVM is not sensitive to k, the number of selected nearest blocks. Figure 2 shows the RKL result of KNB ensemble SVM on the test data with the k as a variable. It can be seen that between a large range of the value of k (from 10 to 20), the RKL is rather stable and is considerably better than the previous best result obtained by Foussette et al. [22].

Besides the ranking accuracy, the training speed is another important factor. Table 4 lists the training mode and training time of the four methods. Standard classification SVM and Ranking SVM are trained on all available blocks. KNB global SVM is trained *online* on selected k nearest blocks, and the training time is dependent on k and is unpredictable in practice. For KNB ensemble SVM, local SVMs are trained *offline* on separate blocks. We can see that the offline training of KNB ensemble SVM only costs 9 seconds, 3600 times faster than ranking SVM. These experiments were performed on a Solaris/SPARC Server with 8 Sun Microsystems Ultra-SPARC III 900Mhz CPUs and 8GB RAM.

4 Conclusion and Future Work

In this paper, we have proposed a K-Nearest-Blocks (KNB) ensemble ranking algorithm with SVMs used as the base learners, and applied it to the protein homology prediction problem. Experiments show that compared to several other SVM-based ranking algorithms, the proposed one is significantly better on most performance evaluation metrics and meanwhile is extremely fast in training

speed. Here, we have used a public data set. It is possible to develop more measures of protein homology to improve the accuracy of protein homology prediction. On the other hand, since the method is domain-independent, it is in principle applicable to general ranking problems. A potential problem with the KNB approach is that when the number of training blocks becomes very large, finding the k nearest blocks may be computationally expensive. However, all methods for expediting the traditional K-Nearest Neighbor method can also be used for the KNB method. In addition, we used a very simply definition of block distance in this paper. In fact, it can be improved in various ways, for example, refinement of block features, feature selection, distance learning, etc. We will try to address some of these aspects in our future work.

References

1. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley-Longman, Harlow (1999)
2. Robertson, S.E., Sparck Jones, K.: Relevance weighting of search terms. Journal of American Society for Information Sciences 27, 129–146 (1976)
3. Fuhr, N.: Optimal polynomial retrieval functions based on the probability ranking principle. ACM Transactions on Information Systems 7, 183–204 (1989)
4. Cohen, W., Shapire, R., Singer, Y.: Learning to order things. Journal of Artificial Intelligence Research 10, 243–270 (1999)
5. Joachims, T.: Optimizing Search Engines Using Clickthrough Data. In: 8th ACM Conference on Knowledge Discovery and Data Mining, pp. 133–142. ACM Press, New York (2002)
6. Baker, D., Sali, A.: Protein structure prediction and structural genomics. Science 294, 93–96 (2001)
7. Zhang, Y., Skolnick, J.: The protein structure prediction problem could be solved using the current PDB library. Proc. Natl. Acad. Sci. USA 102, 1029–1034 (2005)
8. Ginalski, K.: Comparative modeling for protein structure prediction. Current Opinion in Structural Biology 16, 172–177 (2006)
9. Zhang, Y.: Progress and challenges in protein structure prediction. Current Opinion in Structural Biology 18, 342–348 (2008)
10. Soding, J.: Protein homology detection by HMMCHMM comparison. Bioinformatics 2, 951–960 (2005)
11. Teodorescu, O., Galor, T., Pillardy, J., Elber, R.: Enriching the sequence substitution matrix by structural information. Proteins: Structure, Function and Bioinformatics 54, 41–48 (2004)
12. Cooper, W., Gey, F., Chen, A.: Information retrieval from the TIPSTER collection: an application of staged logistic regression. In: 1st NIST Text Retrieval Conference, pp. 73–88. National Institute for Standards and Technology, Washington, DC (1993)
13. Gey, F.: Inferring Probability of Relevance Using the Method of Logistic Regression. In: 17th Annual International ACM Conference on Research and Development in Information Retrieval, Dublin, Ireland, pp. 222–231 (1994)
14. Nallapati, R.: Discriminative Models for Information Retrieval. In: 27th Annual International ACM Conference on Research and Development in Information Retrieval, pp. 64–71. ACM Press, New York (2004)

15. Herbrich, R., Obermayer, K., Graepel, T.: Large margin rank boundaries for ordinal regression. In: Smola, A.J., Bartlett, P., Schölkopf, B., Schuurmans, C. (eds.) Advances in Large Margin Classifiers, pp. 115–132. MIT Press, Cambridge (2000)
16. Crammer, K., Singer, Y.: Pranking with ranking. In: Advances in Neural Information Processing Systems, vol. 14, pp. 641–647. MIT Press, Cambridge (2002)
17. Chapelle, O., Keerthi, S.S.: Efficient algorithms for ranking with SVMs. Information Retrieval Journal 13, 201–215 (2010)
18. McFee, B., Lanckriet, G.: Metric Learning to Rank. In: 27th International Conference on Machine Learning, Haifa, Israel (2010)
19. Fu, Y., Sun, R., Yang, Q., He, S., Wang, C., Wang, H., Shan, S., Liu, J., Gao, W.: A Block-Based Support Vector Machine Approach to the Protein Homology Prediction Task in KDD Cup 2004. SIGKDD Explorations 6, 120–124 (2004)
20. Fu, Y.: Machine Learning Based Bioinformation Retrieval. Ph.D. Thesis, Institute of Computing Technology, Chinese Academy of Sciences (2007)
21. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1995)
22. Foussette, C., Hakenjos, D., Scholz, M.: KDD-Cup 2004 - Protein Homology Task. SIGKDD Explorations 6, 128–131 (2004)
23. Pfahringer, B.: The Weka Solution to the 2004 KDD Cup. SIGKDD Explorations 6, 117–119 (2004)
24. Tang, Y., Jin, B., Zhang, Y.: Granular Support Vector Machines with Association Rules Mining for Protein Homology Prediction. Special Issue on Computational Intelligence Techniques in Bioinformatics, Artificial Intelligence in Medicine 35, 121–134 (2005)
25. Caruana, R., Joachims, T., Backstrom, L.: KDD Cup 2004: Results and Analysis. SIGKDD Explorations 6, 95–108 (2004)
26. Tobi, D., Elber, R.: Distance dependent, pair potential for protein folding: Results from linear optimization. Proteins, Structure Function and Genetics 41, 16–40 (2000)
27. Joachims, T.: Making large-Scale SVM Learning Practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods - Support Vector Learning, pp. 115–132. MIT Press, Cambridge (1999)