Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. TR-111

Machine learning approaches to protein ranking: discriminative, semi-supervised, scalable algorithms

Jason Weston,[1] Christina Leslie,[2] Andre Elisseff,[1] William Stafford Noble[3]

June 2003

[1] Department Schölkopf, Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany. weston@tuebingen.mpg.de,andre@tuebingen.mpg.de
[2] Department of Computer Science, Columbia University, USA. cleslie@cs.columbia.edu
[3] Department of Genome Sciences, University of Washington, USA. noble@gs.washington.edu

This report is available in PDF–format via anonymous ftp at ftp://ftp.kyb.tuebingen.mpg.de/pub/mpi-memos/pdf/TRtransprot03.pdf.
The complete series of Technical Reports is documented at: http://www.kyb.tuebingen.mpg.de/techreports.html

Abstract

A key tool in protein function discovery is the ability to rank databases of proteins given a query amino acid sequence. The most successful method so far is a web-based tool called PSI-BLAST which uses heuristic alignment of a profile built using the large unlabeled database. It has been shown that such use of global information via an unlabeled data improves over a local measure derived from a basic pairwise alignment such as performed by PSI-BLAST's predecessor, BLAST. In this article we look at ways of leveraging techniques from the field of machine learning for the problem of ranking. We show how clustering and semi-supervised learning techniques, which aim to capture global structure in data, can significantly improve over PSI-BLAST.

1 Introduction

Pairwise sequence comparison is the killer app of bioinformatics. Algorithms like BLAST [1] and PSI-BLAST [2] allow a user to search a database of DNA or protein sequences using a single-sequence query. The output of the search is a list of database sequences (called *targets*) that are ranked according to their similarity to the query. The similarities discovered by the algorithm may allow the user to infer functional properties of either the query or target sequences; for example, a query sequence of unknown function that retrieves from the database a large collection of kinases is likely itself to be a kinase. This straightforward bioinformatics application is familiar to essentially every biologist. The web engine of the most popular pairwise sequence comparison algorithm, the BLAST server at the NCBI, runs 50,000 searches per day.

The first algorithms for detecting subtle sequence similarities were designed explicitly with respect to a simple model of molecular evolution. However, the most successful solutions to this problem involve learning from data. These learning approaches can be divided into two types. Profiles, profile hidden Markov models (HMMs) [10] and support vector machine-based methods [8, 11, 12] do their learning before any query is issued. This type of learning is family-based, building one model for each of a collection of families of sequences that are known *a priori*. The Pfam database of profile HMMs is probably the best known example of this approach. By contrast, the second tye of learning algorithm performs all of its learning on the fly, after a query is issued. In this case, an HMM or SVM is constructed, but the training set for the model is accumulated iteratively from the target database, rather than being specified *a priori*. SAM-T98 is an example of an iterative profile HMM method, and PSI-BLAST is essentially a fast approximation of this approach.

For sequence similarity detection, learning on the fly has so far yielded more useful algorithms than learning in advance. The utility of the on-the-fly approach stems from its ability to find relationships between the query sequence and any target sequence. In the learning-in-advance approach, by contrast, only target sequences that can be placed into one of the *a priori* sequence groups are represented in the learned library of models. Hence, the learning-in-advance approach only effectively searches a subset of the target database.

In this work, we propose two methods for learning in advance that solve this problem, allowing the query to search the entire target database. Rather than building a library of models for different groups of sequences, the method learns a model of the entire space of known protein sequences. The resulting model can then be efficiently queried, yielding a ranked list of similar proteins. Many possible techniques for building a model of protein space could be applied in this framework. We focus on two simples model, clustering and leveraging semi-supervised learning techniques. The former consists of using a collection of clusters, discovered by the k-means clustering algorithm. The "distances" used by k-means are derived from E-values that are computed by PSI-BLAST. The clustering algorithm is scalable to large databases, such as the NCBI non-redundant database, and can be re-trained in an online fashion in response to incremental updates. The query algorithm is a small constant factor more expensive to compute than PSI-BLAST. The latter casts the ranking problem into a classification framework: one views proteins as belonging to a class which the query belongs to, all other types of proteins are bundled into a second class. One views the database as unlabeled data, which can help solve this classification task, to do this we employ an algorithm called label propagation [18].

By learning about all available protein sequences and protein classes simultaneously, these approaches leverage information that is unavailable during an isolated PSI-BLAST query, improving upon the recognition

performance. Experiments described here demonstrate that, when tested on its ability to recognize remote homologs in the SCOP database of protein domain structures, both new approaches perform significantly better than PSI-BLAST.

2 Methods

2.1 Ranking by clustering

In this section we show how clustering algorithms can be used to help solve the ranking problem. Our descrition focuses on the k-means clustering algorithm, however one could equally use another fast clustering algorithm (e.g. hierarchical clustering such as average linkage).

The k-means algorithm is an unsupervised clustering algorithm that represents structure in the data using a collection of inferred cluster means. The algorithm begins with a random assignment of each data point to one of k clusters, where k is a user-specified parameter. The algorithm then computes the mean of each cluster and re-assigns each data point to the cluster whose mean is nearest. The procedure of re-computing means and re-assigning points iterates until the cluster assignments stop changing. Convergence typically happens quickly; in the experiments reported here, a typical number of iterations is 10. From this clustering, the ranking with respect to a given query is produced by assigning a higher rank to all proteins in the cluster closest to the query than all proteins from other clusters. Such a ranking can be expected to outperform the ranking with the original distance measure if there is cluster structure in the data captured by the clustering algorithm.

The k-means algorithm requires that the data points being clustered have some notion of distance associated with them, so that it is possible to find the mean of a set of sequences and to compute the distance from that mean to a single sequence. In typical applications of k-means, this distance can be calculated explicitly from the fixed-length input vectors; however, for our problem the input data are variable-length sequences from a discrete alphabet. We thus define the pairwise "distance" between two proteins A and B as the PSI-BLAST E-value computed with A as query and B as target. PSI-BLAST runs for a maximum of 6 iterations and reports E-values up to a maximum of 10000. Database entries that are not assigned an E-value are also given this maximal score. Using these pairwise distances, the distance from a protein to a cluster center can then be computed via the mean distance to all the points belonging to that cluster. This simple generalization of the k-means algorithm is sometimes called *kernel k-means.*[1] We assign each cluster center to be equal to the position of a single protein from the cluster. These central proteins are randomly selected such that no two cluster centers have distance less than 0.005 (the default threshold E-value of PSI-BLAST). To perform the final ranking we use the following method. Let x_i, $i = 1, \ldots, m$ be the database of proteins, $C(x_i) \in \{1, \ldots, k\}$ denotes the cluster that the i^{th} example belongs to, and $psi(x_i, x_j)$ denote the E-value given by PSI-BLAST for example x_j given the query x_i. Then, given a query Q, Protein x_i from the database is ranked higher than protein x_j by k-means ranking if $d(Q, x_i) < d(Q, x_j)$, where $d(x^*, q) = \frac{1}{\#\{C(x^*)=C(x_i):i=1,\ldots,m\}} \sum_{i:C(x_i)=C(x^*)} psi(Q, x_i)$. If $d(Q, x_i) = d(Q, x_j)$ then x_i is ranked higher than x_j if $psi(Q, x_i) < psi(Q, x_j)$.

In general, we do not know *a priori* the optimal number k of clusters to create from a given collection of protein sequence data. Therefore, we learn this parameter from the data in a supervised fashion, using the known (annotated) proteins in our database. We could choose the k which gives the smallest mean score over the labeled superfamilies, where the score for each superfamily is defined as the mean ROC-50 score (see below for an explanation) over the members of the superfamily. (One could also average over superfamilies in this way (rather than individual queries) to remove bias in the sizes of the superfamilies, which do not accurately reflect the likely distribution of queries requested by a user.)

The time complexity of the standard k-means algorithm [5] is $O(rkmd)$ where m is the number of data points, d is their dimensionality, and r is the number of iterations required. Empirically, this running time grows sublinearly with k, m and d. However, in our case we have data for which we only have pairwise distances available. Assuming that a distance matrix is already computed, the worst case complexity is

[1] Note that k-means usually minimizes the Euclidean distortion $\sum_{i=1}^{m}(x_i - C(x_i))^2$ given training data $x_1, \ldots, x_m$ where $C(x_i) = \arg\min_{c_j \in \{c_1,\ldots,c_n\}}(x_i - c_j)^2$ are the k centers. In our experiments, we observed convergence of the algorithm even though the pseudo-distance derived from PSI-BLAST is not symmetric and does not satisfy the triangle inequality.

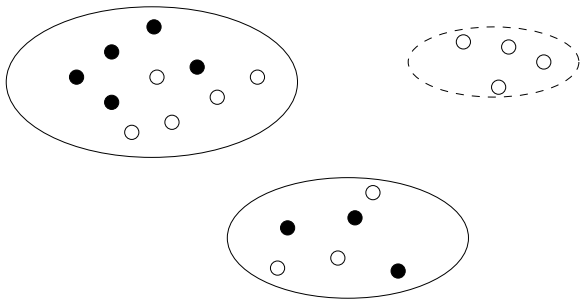

Figure 1: **Missing labels and missing classes.** The diagram schematically represents a portion of protein sequence space. Each point represents a protein; filled points are labeled, and unfilled are unlabeled. Superfamilies are represented as ellipses, and the dotted ellipse represents a superfamily that exists in the data but is not known a priori. Constrained clustering can make use of both known and missing classes for ranking.

$O(rm^2)$, because all pairwise distances have to be summed to compute distances to all cluster centers in each iteration, independent of k. However, this complexity can be improved upon by caching the sum of distances to each cluster center, and only updating the sums with respect to the points which change clusters. Moreover, note that the distance matrix provided by PSI-BLAST is sparse, because not all proteins are ranked with an E-value. This sparseness also makes the computations more efficient. Finally, note also that computing the distance matrix via PSI-BLAST can be parallelized, and that there exist parallel versions of k-means [4].

In the testing stage, given a query one has simply to compute the distance to all proteins in the database via a single run of PSI-BLAST and then to compute the distance to each cluster center via averaging over the points in the database that belong to that cluster, making the algorithm only a small constant factor more expensive to compute than PSI-BLAST. Note that we symmetrized the matrix of E-values (by taking the maximum) for the fixed database before performing the clustering. We did this as the assymetry could confuse the k-means algorithm (however, we did not symmetrize the query itself, as this would be computationally infeasible in a real-world system.)[2]

Constrained k-means If some labels are already known, it would be nice to also take these into account when performing the clustering, see e.g. Figure 1. For example, one may know protein X and protein Y are in the same superfamily, and that X and Z are not, such information are called *label constraints*, and can be given e.g. via the SCOP database. Such constraints can be easily incorporated into k-means by using constrained k-means [16]. The algorithm is almost identical except one forbids placing points in the same cluster if a label constraint is violated. If a superfamily is known, it has already been shown that a simple average distance to the superfamily is better than using PSI-BLAST to determine protein classicication, this method is called family pairwise search (FPS) [7]. Such constraints will make our method perform like FPS for known superfamilies, and will behave similarly to FPS when the clustering accurately predicts superfamilies for unknown classes.[3]

2.2 Ranking by label propagation

The ranking problem can be seen as an extreme semi-supervised problem. The field of semi-supervised learning aims to improve classification accuracy given unlabeled data, usually there is much more unlabeled data than labeled data, e.g. in text classification [9] or web page classification [3]. Ranking can be seen as a two-class problem (examples are in the same class as the query or are not) where one has a large amount

[2]Interestingly, in our experiments we noticed that also symmetrizing the query can give a further improvement in accuracy even for PSI-BLAST without clustering at all (results not shown).

[3]Note in the experiments using constrained k-means we symmetrized both the database distance matrix *and* the queries.

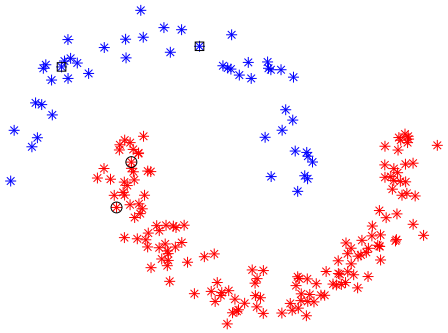

Figure 2: **The two moons problem**: A two class problem with clear clustering structure: all data (red and blue) is unlabeled apart from the black squares and circles, purely supervised classification methods ignoring the unlabeled data will not perform optimally on such a problem. Methods such as label propagation, however, can find the manifold structure in such data.

of unlabeled data, and only one labeled point: the query itself. In this extreme view one can then try to leverage methods used in semi-supervised learning, such as the label propagation algorithm [18].

Label propagation is a simple method to use unlabeled data by propagating labels through dense regions, this implements the cluster assumption commonly used in semi-supervised learning, which says that the label does not change in high density regions[15]. The algorithm works as follows. Let K be a matrix of transition probabilities, typically built using $K_{ij} = \exp(-d(x_i, x_j)/2\sigma^2)$ where $d(\cdot, \cdot)$ is a distance and is subsequently pre-processed with $K \leftarrow D^{-1}K$ where D is a diagonal matrix with $D_{ii} = \sum_p K_{ip}$. Let Y be an $m \times 2$ matrix of label probabilities, where m is the number of examples. Set $Y_{i1} = 0.5, Y_{i2} = 0.5$ for unlabeled data, and $Y_{i1} = 1, Y_{i2} = 0$ or $Y_{i1} = 0, Y_{i2} = 1$ when labeled, depending on the class. Then one iteratively performs the following steps:

1 Propagate $Y \leftarrow KY$

2 Row normalize Y

3 Clamp (reset) known labels, repeat from step 1 until convergence.

The algorithm is proved to converge to a fixed point [18] (in fact, the solution can also be computed in closed form). It is simple and relatively efficient (compared to other semi-supervised learning techniques) and has few parameters.

We will use this technique for ranking, we do this by using PSI-BLAST to assign some initial "pseudo"-labels. PSI-BLAST returns a confidence in prediction, it is extremely reliable for high confidence hits: our idea is to use PSI-BLAST to assign pseudo-labels to these hits, we clamp those labels in the algorithm above. That is, we choose for all prtoeins with a PSI-BLAST E-value < 0.05 compared to the query we assign them as positive examples. We define all proteins as negative examples that have a maximum E-value (unassigned in PSI-BLASTs ranking), as long as they have no neighbor with an assigned E-value (where we define two proteins as neighbors if they have an assigned E-value). Note that labeled data is still unused, we can use it for model selection. As for k-means, such a method could be expected to outperform ranking with the original distance measure if there is cluster structure in the data that relates to the true classes of proteins, and the algorithm captures these clusters in its decision rule. Label propagation can find manifold structure such as in the two-class classification problem shown in Figure 2.

Scalability issues can be resolved somewhat by only computing a few iterations: in our experiments we only perform 20 iterations. Secondly, PSI-BLAST only assigns a ranking score to a few proteins in the database, thus the kernel matrix generated will very sparse.

Finally it is possible to generalize label propagation to not have to deal with two-classes but with only one unlabeled database for the ranking problem directly, a preliminary study can be found in [17].

2.3 Experimental design

In order to measure the performance of the methods, we use the SCOP database of protein structural domains as a gold standard [13]. SCOP is a human-curated database of known 3D structures of protein domains. The database is organized hierarchically into classes, folds, superfamilies and families. For the purposes of this experiment, two domains that come from the same superfamily are assumed to be homologous, and two domains from different folds are assumed to unrelated. For pairs of proteins in the same fold but different superfamilies, their relationship is uncertain, and so these pairs are not used in evaluating the algorithm. This labelling scheme has been used in several previous studies of remote homology detection algorithms [].

The algorithms are provided a combined collection of labelled and unlabelled domain sequences. The sequences are 7329 SCOP domains from version 1.59 of the database, purged using `astral.stanford.edu` so that no pair of sequences share more than 95% identity, and a collection of 94,074 proteins from the Swiss-Prot (SPROT) database. The latter consists of all sequences from version 40 of Swiss-Prot, discarding those labeled as fragments. The SCOP domains were split into two portions: 554 superfamilies (4246 proteins) for training and 516 (3083 proteins) for testing. Note that training and testing sequences never come from the same superfamily.

We compared average linkage, k-means and label propagation with three BLAST-based methods: (1) BLAST, (2) PSI-BLAST searching the entire SCOP database, and (3) PSI-BLAST searching a database comprised of both the SCOP and SPROT databases. This comparison demonstrates the degree of improvement that can be gained via the extra (unlabeled) data. k-means and average linkage were trained on a 7329 by 7329 matrix of "distances" between SCOP proteins, where "distance" is the PSI-BLAST (negative log) E-value derived from a search of the complete database (SCOP and SPROT). Note that this experiment can be viewed as a first test of the usefulness of these algorithms. In practice, the clustering could be improved further by using the entire 101,403 by 101,403 (sparse) matrix. In the current experiment, the clustering algorithms only use the unlabeled SPROT data indirectly via the distances induced by PSI-BLAST.

Each algorithm is tested on its ability to recognize a complete superfamily given a single query sequence from that superfamily. Each method is given the above database. The prediction algorithm then produces a rank ordering of the database sequences. From this ordering, the test set SCOP domain sequences are extracted. This ranked list is the basis for the performance comparison of the various algorithms.

For k-means and average linkage we report the optimum parameter choice for the number of clusters having looked at the test error, however in principle one could use model selection to choose this parameter. The value of $2\sigma^2 = 100$ was chosen for label propagation as it was the only power of ten that gave a non-degenerate matrix (not all zeros or all ones), however this could in principle have also been chosen using the training set. We ran 20 iterations of label propagation for each query, some tests of subsets of the data indicate the result may improve by running more iterations (results not shown).

The performance of the algorithms is measured using a modified version of the receiver operating characteristic (ROC) score [6]. The ROC score is based upon a plot of false positives versus true positives for varying classification thresholds. The score is simply the normalized area under this curve. The ROC score thus measures the quality of the entire ranking produced by the algorithm. In practice, only the top of this ranking is important. Therefore, we compute the ROC_{50} score, which is the area under the receiver operatorating characteristic up to the first 50 false positives. A value of 1 implies that the algorithm successfully assigns all the true relationships higher scores than the false relationships. For a random ranking of this data, the expected ROC_{50} score is close to 0.

3 Results

A plot of the total number of queries for which a given method exceeds an ROC_{50} score threshold is given in Figure 3 for average linkage and label propagation, comparing to the BLAST and PSI-BLAST variants. A scatter plot of the ROC-50 scores of individual queries comparing average linkage and label propagation with PSI-BLAST run on the entire database (SCOP+SPROT) is also given in Figure 4. A Wilcoxon signed rank test of equality of medians shows that both k-means and label propagation are significantly better than the other methods with a level of significance level of $\alpha = 0.05$, returning a p-value less than $1e-89$. The steep slope in the plots at around 0.9 ROC-50 correspond to queries mostly from the largest superfamily in

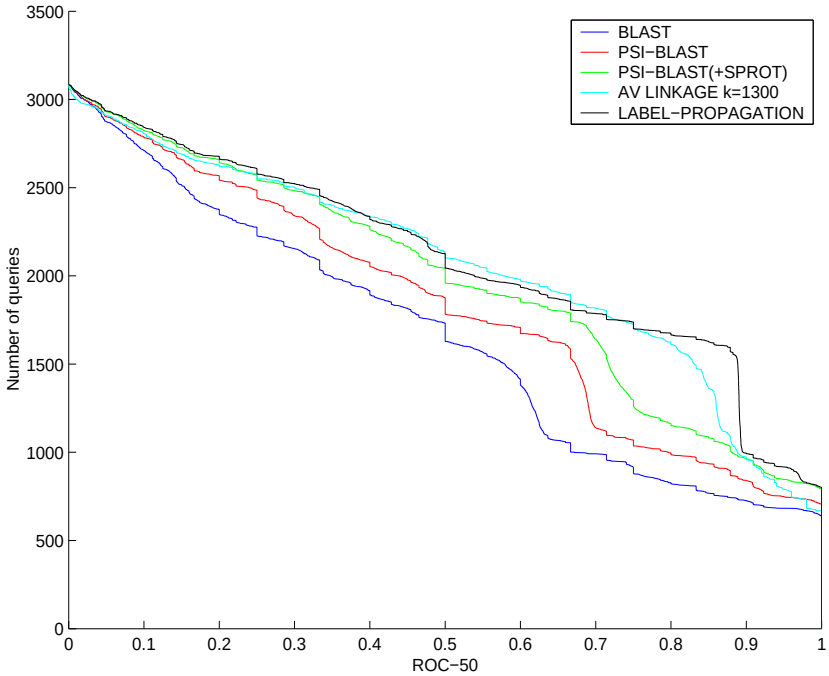

Figure 3: **Number of queries versus ROC-50 threshold** The graph plots the total number of queries for which a given method exceeds an ROC-50 score threshold. Label propagation and average linkage ranking outperform variants of BLAST and PSI-BLAST.

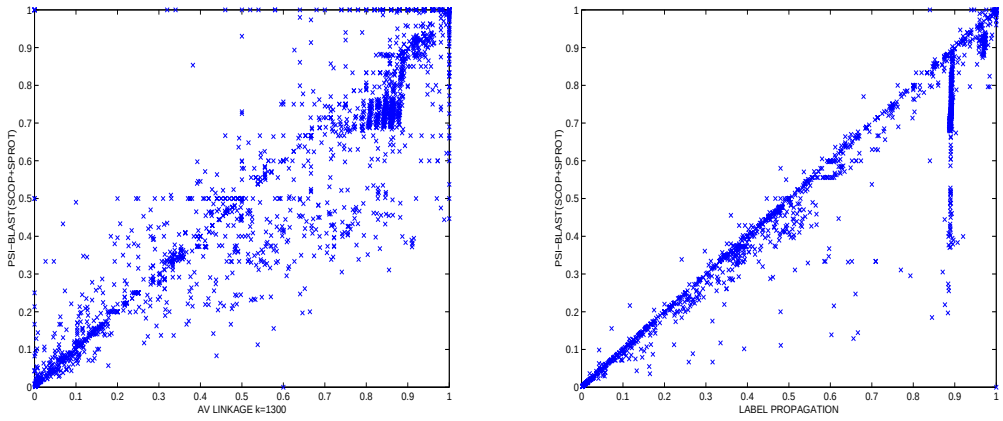

Figure 4: Scatter Plots of ROC-50 Scores, comparing the two new methods k-means (left) and label propagation (right) with PSI-BLAST.

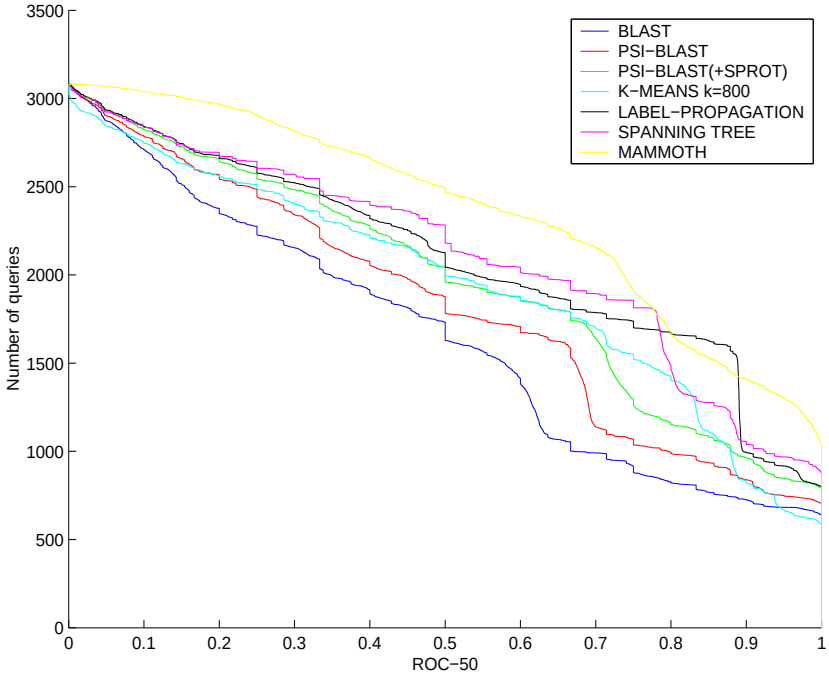

Figure 5: ***k*-means and spanning tree ranking** The graph plots the total number of queries for which a given method exceeds an ROC-50 score threshold for variants of our algorithms.

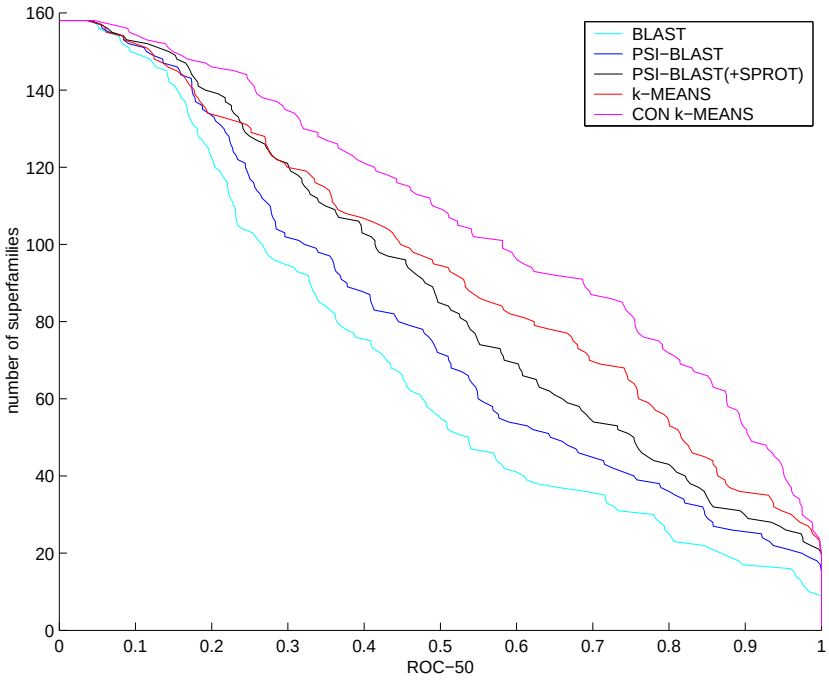

Figure 6: **Constrained k-means ranking for detecting new families from known superfamilies** Constrained k-means outperforms k-means when unknown families of known superfamilies are included in the test set. (Note: this dataset is hence different to the ones plotted in the other graphs.)

the database: superfamily b.1.1.1 (Immunoglobins) with 623 proteins. Improvement for these queries gives evidence of the usefulness of clustering in this domain, indeed as the databases get larger such clustering should help even more.

3.1 Variants of the algorithms.

k-means clustering We also compared with k-means clustering, we report the best vlaue of k ($k = 800$). The results can be seen in Figure 5.

Constrained k-means We also tested the improvement given by constrained k-means over k-means. In our original dataset, above, no superfamily appears in both training and testing sets, in this setting constrained k-means in fact gives no improvement (results not shown). However, if some label information is given for a superfamily that *does* appear in the test set, one observes an improvement. In this experiment we chose a training set of 109 superfamilies and a testing set of 157 different superfamilies, and 48 new families from the training set superfamilies. Hence one never sees proteins from the same family across training and test set, this allows to ask the question: can constrained k-means rank a new family from a known superfamily better than k-means (which wouldn't take into account that the superfamily was known). The answer is yes, as shown in Figure 6: the plot shows the average ROC-50 score averaged over each superfamily in the test set, the optimal k was chosen for each method.

Ranking by spanning tree We also tried a simple method that tries to capture some of the same manifold structure as label propagation, which is similar to computing a spanning tree: start with the query, place it in an otherwise empty set S. Then, find the example which is the minimum distance over all points in the set S: $\text{argmin}_{i:x_i \notin S} \min_{q \in S} psi(q, x_i)$. [4] This point i is then added to the set S and this is repeated until there are no points left to add, the order they were added is the chosen ranking. If there is manifold structure in the data as in Figure 2 one could expect this method to perform well. The performance of this method is shown in Figure 5, it is actually the best known method on the far right of the graph (for completely identifying superfamilies). It also has the advantage of being very efficient to compute. However, it may not perform so well for problems with multiple domains as one may find strong relations in one domain with proteins which are actually related to the query in another domain.

Comparison with MAMMOTH. We also compared these methods with an automated structure-structure comparison algorithm called MAMMOTH [14]. The scores produced by MAMMOTH provide a distance metric that is much more accurate than a sequence-based metric such as BLAST or Smith-Waterman, but can only be computed for proteins with known strcuture. The comparison is shown in Figure 5. It seems promising that the performance of label propagation is reasonably close to the ranking provided by MAMMOTH.

4 Discussion

From a machine learning perspective, the problem of learning to recognize subtle protein sequence similarities has many interesting characteristics, some of which have not been fully exploited by previous algorithms. First, the problem offers a significant amount of prior knowledge, both in the form of biological knowledge of proteins and of molecular evolution, and in the form of algorihms and models for efficiently encoding this knowledge. Second, the problem involves a large quantity of unlabelled data, in the form of sequences with no functional annotation. Third, the problem involves learning multiple classes, where the number of classes is very large. Fourth, in addition to known protein classes, there are a large number of unknown classes, corresponding to families of proteins that have not yet been annotated. Fifth and finally, the problem does

[4] This can be generalized to take the example which is closest where distance is defined as the mean distance over the closest n points in S (choosing $n = 1$ gives the previous equation). This could be important for multidomain proteins, where their is a lack of transitivity of homology. Taking a mean instead of a mean renders a strong similarity of only a single point in S that could be for a domain other than ones in the query being ignored by the algorithm.

not fit cleanly into the standard classification, regreesion or density estimation frameworks, because the required output is a ranking of the similar protein sequences.

The solutions described above address all of these problem characteristics. We handle representation issues and the encoding of prior knowledge by building upon the powerful PSI-BLAST algorithm. Our algorithm exploits unlabelled data, and allows for multiple classes, including missing classes via clustering. The best performing algorithm so far discovered is label propagation which produces a ranking algorithm that takes into account clustering information in a very intuitive way. On the other hand, what we have lost is the original E-value output of PSI-BLAST, which is no longer valid.

Future work will include comparing with HMMs such as SAM-T02 and performing experiments on a full size database, that is performing the clustering and propagation on the Swiss PROT database. Finally, and perhaps most importantly, we need to test our methods on a multidomain protein dataset where clustering information is quite different to that in single domain datasets.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. A basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[2] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. *Nucleic Acids Research*, 25:3389–3402, 1997.

[3] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. *Proceedings of the Worshop on Computational Learning Theory*, 1998.

[4] Inderjit S. Dhillon and Dharmendra S. Modha. A data-clustering algorithm on distributed memory multiprocessors. In *Large-Scale Parallel Data Mining*, pages 245–260, 1999.

[5] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, second edition, 2001.

[6] M. Gribskov and N. L. Robinson. Use of receiver operating characteristic (roc) analysis to evaluate sequence matching. *Computers and Chemistry*, 20(1):25–33, 1996.

[7] W. Noble Grundy. Family-based homology detection via pairwise sequence comparison. *Proceedings of the Second Annual International Conference on Computational Molecular Biology (RECOMB98)*, 1998.

[8] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 2000.

[9] T. Joachims. Transductive inference for text classification using support vector machines. *Proceedings of ICML*, 1999.

[10] A. Krogh, M. Brown, I. Mian, K. Sjolander, and D. Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[11] C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. *Neural Information Processing Systems 15*, 2002.

[12] C. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. *Proceedings of RECOMB*, 2002.

[13] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. SCOP: A structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[14] A. Ortiz, C. Strauss, and O. Olmea. MAMMOTH (Matching molecular models obtained from theory): An automated method for model comparison. *Protein Science*, 11:2606:2621, 2002.

[15] M. Seeger. Learning with labeled and unlabeled data. Technical report, University of Edinburgh, 2001.

[16] Kiri Wagsta, Claire Cardie, Seth Rogers, and Stefan Schroedl. Constrained k-means clustering with background knowledge. http://citeseer.nj.nec.com/wagsta01constrained.html.

[17] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schoelkopf. Ranking on data manifolds. Technical report, Max Planck Institute for Biological Cybernetics, 2003. (in preparation).

[18] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical report, CMU, 2002.