USING POSITION-SPECIFIC-VALUE METHOD FOR REMOTE PROTEIN CLASSIFICATION

YuGang Li   Zhi Yong Liu   XiangZhen Qiao
lyg@ncic.ac.cn   zliu@mail.nsfc.gov.cn   qiao@ncic.ac.cn

Institute of Computing Technology,
Chinese Academy of Sciences, Beijing, China 100080,
Graduate School of the Chinese Academy of Sciences, Beijing, China 100039

ABSTRACT

An important research topic in Bioinformatics is to understand the meaning and function of each protein encoded in the genome. One of the most successful approaches to this problem is via sequence similarity with one or more proteins whose functions are known. There are two different kinds of methods used for the protein classification: the generative method and the discriminative one. Till now, many approaches have been presented for the two kinds of methods. The methods for the first kind include those based on pair-wise sequence similarity [1, 2, 3, 4, 5, 6], profiles for protein family [7], and hidden Markov Models [8, 9, 10]. Meanwhile, many discriminative methods have been developed in the literature [11, 12, 13, 14, 15, 16, 17]. Among the most successful ones of them are those based on the Support Vector Machines [12, 13, 14, 15, 16]. The Fisher Kernel for the SVM is used in [12, 13], while the spectrum Kernel is used [15, 16], and [14] uses a pair-wise sequence similarity algorithm, such as Smith-Waterman, BLAST, etc., as the kernel.

The methodology of the generative method involves building a model for a single protein family and then evaluating the sequence to be judged to see the degree it fits the model. If the degree exceeds a threshold, then the sequence is regarded as a member sequence of the family; it is not, otherwise. Discriminative methodology differs from the generative one in that it classifies the protein sequences into positive and negative ones according to whether they belong to the family concerned and then uses a learning algorithm to learn the distinction between the different classes. Because the generative method uses only the positive sequences, while the discriminative ones take into consideration both the positive and the negative examples, a discriminative method will in general be more accurate than a generative one.

The algorithms can be divided into four classes in

1. Introduction

One key element in understanding the molecular machinery of the cell is to understand the meaning and function of each protein encoded in the genome. A very successful approach to this problem is via sequence similarity with one or more proteins whose functions are
accordance with their rank of accuracy [14]. The dynamic programming algorithm [1, 2, 3] is the most sensitive among all the pair-wise based methods, while FASTA [5] and BLAST [4] are faster and less sensitive than the Smith-Waterman algorithm because of their heuristic features. The second class includes the Profiles methods. Two typical representatives of it are the Hidden Markov Model [9, 10] and the Profiles [7]. They aggregate statistics from a set of similar sequence and compare the resulting statistics to a single sequence of interest. As to the third, information contained in large databases of unlabeled protein sequences is taken into account. SAM-T98 [8] and PSI-BLAST [6] are their typical representatives. The fourth uses supervised machine learning algorithms, for example, the Bayesian neural network (BNN) [11, 17]. They explicitly model the positive and negative examples and try to use it to judge some unlabelled sequences. When applying these algorithms, the variant length sequence space should be mapped to some fixed length vector spaces. Many methods are applied for the translating from sequence space to vector spaces [12, 13, 14, 15, 16, 17]. [12, 13] begin by training a generative hidden Markov model (HMM) for a protein family and use the trained model for each input sequence. [15] uses a string spectrum. [16] uses mismatch strings spectrum. [14] uses the family based pair-wise kernel, while the kernel adopted by [17] incorporates the 2-gram and the 6-letter exchange group. Among them, [15, 16] are general for any sequence based classification problem. The kernel [12, 13] uses only positive sequences when the model is being trained, while [14] use both the positive and negative sequences in not only the classification step, but also the map step. This is probably the reason that it outperforms the other methods.

In this paper, we present a new kernel, called the Position Specific Value (PSV) kernel, to use with an SVM for the remote protein homology detection. The Position Specific Value kernel is focused on a feature mapped to a vector space indexed by the position of the sequence. The value of an entity of the mapped vector is the Edit Distance (or the evolution distance) [18] of the k-length subsequence starting at certain position of the sequence to the k-length subsequence of the family’s latest common ancestor, starting from the same place.

Compared with SVM-pairwise, one of the most successful methods published, our algorithm is significant better than the SVM-pairwise in the time complexity. During the vectorization of the training step, the running time complexity of the SVM-pairwise is $O(N^2mp)$ while that of the PSV is $O(Nmk^2)$, where $m$ and $p$ are the length of the two sequences concerned, $N$ is the number of sequences in the training set, $k^2<<p$. Moreover, the edit distance of every pair of the k-length subsequences can be work out in advance, then the time will be reduced by a factor of $k^2$. Thus our algorithm will be $O(N^2p)$ times faster than the SVM-pairwise. They both use $O(N^2)$ running time in the optimization period.

When a sequence is to be classified, the SVM-pairwise needs running time $O(Nmp)$ to map it to the vector space, while the PSV needs only $O(m)$.

In Section 2 the PSV method is presented and vectorization of the PSV is described in more detail. Section 3 describes the experiments for the comparison of several protein homology detection methods. In section 4, the results of the experiments are analyzed. Concluding remarks are presented in section 5.

2. The PSV method

The idea of the Position Specific Value kernel is that: each family of protein sequences have evolved from some latest common protein ancestor and different families are evolved from different latest common protein ancestors. Given a protein sequence family, the evolution from the latest common ancestor to a member sequence will lead to less “inserts” and “deletes” than that to a nonmember sequence.

And, as far as a member sequence is concerned, for most of its k-length subsequences, the edit-distance from it to the k-length subsequence starting from the same position of the latest common ancestor, will be smaller than when a nonmember sequence.

SVMs need a fixed length vector space, while sequences are variant-length. Thus we need to translate variant-length sequences into fixed-length vectors in
order to use SVM.

The protein sequences are represented as vectors in a high-dimensional feature space via a Position Specific Value feature map, then a support vector machine (SVM) is being trained, which is a large margin linear classifier on the feature vectors representing our training sequences. Since SVMs are kernel based learning algorithms, we don’t calculate the feature vectors explicitly but compute their inner products instead.

The PSV kernel is based on a feature map from the space of all finite sequences to \( M \)-dimensional vector space indexed by the position where certain spectrum starts. Different sequences of a family may have different lengths. Currently we use the length of the largest sequence as the length of a family (shorter ones will be processed with some spaces appended). Let \( n \) be the length of the sequences of a family. Given a number \( k > 1 \), we define \( M = n - k + 1 \). Suppose that the family’s sequences are \( S_1, S_2, \ldots, S_n \), their latest common ancestor is \( X \), \( S_{i,j} \) be the \( j \)-th \( k \)-length subsequence of \( S_i \) (starting at position \( j \) ), \( X_j \) be the \( j \)-th \( k \)-length subsequence of \( X \), and \( D_i \) be the vector that will be mapped from sequence \( S_i \) and \( d_{i,j} \), the \( j \)-th entry of \( D_i \) is the edit distance from \( S_{i,j} \) to \( X_j \). Edit distance is a method to formalize the notion of distance between sequences. It focuses on editing one sequence into another via a series of edit operations on individual letters. The legal operations include insertion, deletion, and substitution of one character with another. Given costs or penalties for the insertion, deletion and substitution, the edit distance between two sequences is the minimal cost to translate one sequence into another.

For example, let the cost for insertion, deletion and substitution be 1, then the edit distance between “vintner” and “writers” are 5, because there are 1 substitution, 2 deletions and 2 insertion are operated on “vintner” to translate it to “writers” as shown in the following.

\[
\text{v - i n t n e r -} \\
\text{w r i - t - e r s}
\]

Now, we will map a sequence, say \( S_n \), from the sequence space into a vector, say \( D_n \), in the \( M \)-dimentional vector space.

\[
\phi_k(S_{i,j}) = \text{edit_distance}(S_{i,j}, X_{i,j}) \quad (1)
\]

For the sequence \( S_n \), we represent the map of \( S_{i,j} \) as:

\[
\Phi_k(S_n) = (\phi_k(S_{i,1}), \phi_k(S_{i,2}), \ldots, \phi_k(S_{i,M})) \quad (2)
\]

The PSV kernel is the inner product in the feature space of feature vectors:

\[
K_k(S_i, S_j) = \langle \Phi_k(S_i), \Phi_k(S_j) \rangle \quad (3)
\]

However, because a family’s latest common ancestor sequence is not known, something must be done to “find” it. Here, we use the sequence that will be mapped to the center of the vectors mapped from the family sequence. So we get a variant of the feature map function for simplicity:

\[
d_{i,j} = \frac{1}{N} \sum_{k=1}^{N} \text{edit_distance}(S_{i,j}, S_{k,j})
\]

\[
\Phi(S_i) = (d_{i,1}, d_{i,2}, \ldots, d_{i,M}) \quad (4)
\]

\[
K(S_i, S_j) = \langle \Phi(S_i), \Phi(S_j) \rangle
\]

Note that we adopt a simple method to make the sequences the same length by appending ‘-‘ to the right end of the short sequences.

3. Description of the Experiments

In the experiments, we compare the performance of nine different algorithms used for protein homology detection: SVM-pairwise, SVM-pairwise+, FPS, SVM-Fisher, SAM, PSI-BLAST, SVM-pairwise_BLAST, PSI-BLAST, and our SVM-PSV algorithm.

We evaluate the recognition performance of each algorithm through testing the ability to classify protein domains into super-families in the Structural Classification of Proteins (SCOP)[19]. Sequences were selected using the Astral database (astral.standford.edu [20]), removing similar sequences using an E-value threshold of \( 10^{-25} \). This resulted in 4352 distinct sequences, grouped into families and super-families. For each family, the protein domains belonging to it are considered positive test samples, and the protein domains outside it but within the same super-family are regarded as positive training examples. Thus, we obtain 54
families, each having at least 10 member sequences (positive test) and 5 super-family members out of it (positive train). Negative examples are taken from outside the positive sequences’ fold, and are randomly split into train and test sets in accordance with the same ratio as positive examples [14]. This is the same as that used by [14].

We use the Gist Support vector machine free software as our SVM implementation [21] (http://microarray.cpmc.columbia.edu/gist/). The kernel is the key element of an SVM, which produces the similarity score between pairs of input vectors.

Because a subsequence may appear many times in some sequences, this leads to one pair of subsequences being calculated many times. In the experiment k=3 is used, so the distance of all the pairs of subsequences can be hold in memory if a memory larger than 512M is available. The edit distance of every pair of k-length spectrums is calculated in advance. Both it and the vectorization of the SVM-pairwise use the Smith-Waterman algorithm, and the default parameter, gap opening penalty and extension penalties of 11 and 1, respectively, and the BLOSUM 62 matrix. Hidden Markov models are trained using the Hmmer software [22] (predict.sanger.ac.uk/mirrors/hmm/hmm.html) and the Clustalw software [23] (bimas.dcr.t.nih.gov/clustalw/clustalw.html). The SVM-fisher uses the same trained HMMs during the vectorization step. For the kernel function, PSV adopts the inner-product kernel, while the other methods based on the SVM use a radial basis as in [14]. Other methods are the same as those in [14].

Each of the above nine methods worked out as output a ranking of the test set sequence. We adopt the receiver operating characteristic (ROC) scores and the median rate of false positive (RFP) to judge the performance of the methods. The ROC scores is the normalized area under a curve that plots true positives as a function of false positives for differential classification thresholds [24]. A perfect classifier that puts all positives at the top of the ranked list will gain an ROC score near 1, while the ROC score of a random classifier will near 0. The median RFP score is the fraction of negative test sequence that score higher or better than the median score of positive sequence.

4. Experimental Results and Analysis

The results of the experiments are shown in Figure 1 and Figure 2. They rank the nine protein sequence classification methods according to the ROC and median RFP scores respectively. In every graph, a higher curve means more accurate homology detection performance. When the ROC score is used, as shown in Figure 1, the PSV is much higher than the other seven methods and as high as, if not higher than, SVM-pairwise method. When the median RFP score is adopted, as shown in Figure 2, the PSV is a little higher than the SVM-pairwise method and they both are much higher than all other seven methods.

Moreover, the recognition performance of PSV may be improved significantly. The reason is that the model adopted at present is still preliminary. First, currently the center of the sequences’ mapped vectors is used to stand for the latest common ancestor of the family. Second, we simply padding ‘-‘ at the shorter sequence to make the sequences having the same length. Finally, some family having too few member sequences, this exemplifies the error. Thus the model may be fine tuned to increase its recognition performance.

With respect to the computational efficiency, the PSV-SVM method is significantly better than the SVM-pairwise. In the step of training the SVM, both the algorithms include an SVM optimization, which is roughly O(N^2). The vectorization step of SVM-pairwise involves N^2 computing pair-wise scores. Using Smith-Waterman, each computation is O(mp), where m and p are the length of the two sequences concerned. Thus the total running time complexity is O(N^2mp). In contrast, the vectorization step of the PSV-SVM involves computing M edit distance of k-length subsequences, each has time complexity O(k^2), yielding a total running time of O(KNMc^2) where K is the number of positive training sequences. Thinking that the k used in real application is usually less than 10 (3 or 5 in general) while p is about hundreds, thousands or tens of thousands,
Figure 1: Relative performance of the nine homology detection methods. This graph plots the total number of families for which a given method exceeds a ROC score threshold.

Figure 2: Relative performance of the nine homology detection methods. This graph plots the total number of families for which a given method exceeds a median RFP score threshold.
we have \(k^2 \ll p\), \(N / K \approx 100\) and \(m \approx p \approx M\). For simplicity, we let \(m = p\), then the PSV is \(O\left(\frac{Nm}{Kk^2}\right)\) or \(O(m)\) times faster than the SVM-pairwise. Moreover, the edit distance of every pair of the \(k\)-length spectrums can be work out in advance, thus we will reduce the running time dramatically because we need not apply the Smith-Waterman algorithm to calculate the edit-distance of two \(k\)-length spectrums. Getting together, we can obtain a significant speedup over the SVM-pairwise. Certainly, there are many ways to speed up the SVM-pairwise vectorization. For example, it is possible to carry out the vectorization using a linear time approximation of the Smith-Waterman, such as BLAST. This modification will remove a factor of \(m\) from the running time, but the change will also decrease the accuracy of the algorithm.

In the step of testing, our SVM-PSV method can discover an unlabelled sequence using time of \(O(KMk^2)\), while the SVM-pairwise will spend \(O(Nm^2)\), we get a speedup of \(O\left(\frac{Nm}{Kk^2}\right)\) or \(O(m)\).

5. Conclusions

In this paper, we present a new method to solve the problem of protein sequence classification. The idea of it is that the member sequences will lie nearer their latest common ancestor than those that do not belong to the family (super-family). It is as accurate as the SVM-pairwise, one of the most accurate methods. But, with respect to the computational complexity, it is significantly better than the SVM-pairwise.

Experiments were carried out in order to test the performance of our proposed algorithm. The experiments are based on a benchmark data set on the SCOP [19]. The result shows that the PSV-SVM is as accurate as the SVM-pairwise algorithm, and much more accurate than the other exiting algorithms in the experiments. However, it is more time efficient than the SVM-pairwise algorithm.

Future work to improve the recognition performance of the SVM-PSV is being investigated.

Acknowledgements: we thank Professor William Standford Noble for providing access to detailed results from their prior work.

REFERENCES


