Comparison of One-Class SVM and Two-Class SVM for Fold Recognition

Alexander Senf, Xue-wen Chen, and Anne Zhang

The University of Kansas, Lawrence KS 66045 USA,
ajsenf@ku.edu, xwchen@ku.edu, yazhang@eecs.ku.edu

Abstract. The best protein structure prediction results today are achieved by incorporating initial structural prediction using alignments to known protein structures. The performance of these algorithms directly depends on the quality and significance of the alignment results. Support Vector Machines (SVMs) have shown great potential in providing good alignment results in cases where very low similarities to known proteins exist. In this paper we propose the use of a one-class SVM to reduce the computational resources required to perform SVM learning and classification. Experimental results show its efficiency compared to two-class SVM algorithms while producing results of similar accuracy.

1. Introduction

Functional protein annotation is generally dependent on knowledge of the three-dimensional structure of a protein. However, our knowledge about protein structures grows at a much slower rate than the discovery of new protein sequences. Protein sequences can be recovered with relative ease from DNA sequences, but to determine the accurate structure of a protein with a given sequence still requires time-consuming experiments, such as X-Ray Crystallography or NMR. Determining the structure of a protein from its sequence computationally is among the most important problems in bioinformatics today.

Many methods have been developed over the past 25 years to generate structural information for unknown proteins by comparing their sequences to the sequences of proteins with known structures. A high degree of sequence similarity has been shown to entail a high degree of structural similarity, which also entails functional similarities. Databases such as the Structural Classification of Proteins (SCOP) database [11] have been devised to organize proteins according to various levels of structural and functional similarities.

One of the methods used to detect protein structures from sequences is called fold recognition. With fold recognition the unknown protein sequence is aligned to known proteins, and the statistical significances of the alignments are estimated. The sequence and location of secondary structural elements can then be determined using the information from the most significant matches in the database. This method is

often combined with comparative modeling approaches to build a complete three-dimensional protein structure from the fragments predicted using fold recognition.

While fold recognition works well for proteins with high degrees of sequence similarity to known proteins, this method fails to produce satisfactory results if the closest known proteins exhibit less than 20% similarity. In these cases the best results are currently produced using molecular dynamics approaches, which physically simulate the folding process of the unknown protein. These methods, however, are computationally very intensive and are not yet able to produce sufficiently accurate results that could be used for functional annotation.

A better solution is to find known proteins that may be structurally and functionally related to the unknown sequence, even in the absence of any significant sequence similarities. The existence of such distant relationships has become apparent as the number of proteins with known structure increased in recent years, and an increasing number of proteins with similar structure and function were observed to have very low primary sequence similarities.

Among the methods developed to detect more distant similarity relationships are the use of sequence profiles as in PSI_BLAST [1], or profile Hidden Markov Models (HMM) [7]. Profiles extend the sequence similarity search from individual sequences to sequences families by incorporating statistical information from multiple sequence alignments of highly related proteins. Some methods have been developed to include structural information along with sequential information to increase the probability of finding remote structural relationships. Among these approaches are GenTHREADER [6] and 3D-PSSM [8].

A more recent promising attempt has been to combine the alignment of sequence profiles with support vector machine (SVM) classifiers [5,9,14]. Approaches such as in [14], which focus on the kernel function used with the SVM, or in [9], which combine pairwise sequence alignments with SVM classification indicate a significant improvement over conventional methods in the ability to detect remotely related proteins. The work of Han et al in [5] extends this approach by combining profile alignments with SVM classifiers.

In this paper we introduce a one-class SVM for the fold recognition problem. One-class SVMs offer significant savings in terms of space and speed over two-class SVMs, because only positive examples are needed to train the SVM. Previous applications of one-class SVMs in the area of Bioinformatics include [16] and [20]. One-class SVMs find their primary use in the field of novelty detection [17] or in document and image retrieval systems [4], [12], and have been shown to be capable of producing similar results as two-class SVMs [18].

The next section of this paper presents an overview of the theory behind two-class SVMs, and the adaptations for one-class SVMs. Section 3 describes the experimental setup and data used. Section 4 reports the results obtained performing the algorithm presented. The last section summarizes what can be learned from this experiment and gives a brief outlook on future research directions.

2. Support Vector Machines

2.1 Theory

An SVM is a machine learning technique based on Vapnik's Statistical Learning Theory [21]. Two-class support vector machines learn to distinguish between two classes in a given data set by fitting a hyperplane that maximally divides both classes.

This works well for data sets that are linearly separable. In cases where the data is not linearly separable, a linear SVM may still be used when it allows for a certain amount of errors. This is achieved by introducing a slack variable ξ and an upper bound C for the number of errors. The formula to be minimized then takes on the commonly used form:

$$J(\bar{w}, b, \xi) = \tfrac{1}{2}(\bar{w} \cdot \bar{w}) + C\sum_{i=1}^{n} \xi_i \tag{1}$$

subject to $y_i\left[\bar{w} \cdot \bar{x}_i + b\right] \geq 1 - \xi_i$ and $\xi_i \geq 0$.

If the data points are not easily separable even with the provision for a certain amount of errors, the data can be projected into a higher-dimensional feature space using kernels. A kernel is a function that takes the original data points and several parameters, and increases their dimensionality. A good choice of kernel function and corresponding parameters will allow the data to then be separable by a hyperplane, using the same function described in (1). Examples of popular kernel functions are:

Polynomial,

$$\kappa(\bar{x}, \bar{y}) = (\bar{x} \cdot \bar{y})^d \tag{2}$$

Radial Basis Function (RBF),

$$\kappa\left(\bar{x}, \bar{y}\right) = \exp\left(\frac{-\left|\bar{x} - \bar{y}\right|^2}{(2\sigma)^2}\right) \tag{3}$$

and Sigmoid

$$\kappa\left(\bar{x}, \bar{y}\right) = \tanh\left(\kappa\left(\bar{x} \cdot \bar{y}\right) + \Theta\right) \tag{4}$$

with gain κ and offset Θ.

2.2 One-Class SVM

One-class SVMs were first proposed by Schölkopf in [15]. One-class SVMs are an extension of the original two-class SVM learning algorithm to enable the training of a

classifier in the absence of any negative example data. Training can be achieved by treating a certain number of data points of the positive class as if they belong to the negative class. The idea is to define a boundary between the majority of the positive data points and outliers (or atypical data points). One-class SVMs use the parameter ν (Nu) to define the trade-off between the percentage of data points treated as the positive class and the negative class. Two approaches to generate this separating boundary are typically available:

The first approach to train a one-class SVM is to describe a classification function that conforms to a hypersphere boundary between the positive class and the outliers, based on a density distribution function. The parameter ν determines the shape of the boundary.

The second approach fits a hyperplane between the origin (of the coordinate system) and the data points, separating a certain percentage of outliers from the rest of the data points. This approach has been shown to be equivalent to the decision hypersphere and is used by many one-class SVM implementations due to its simpler implementation. The LIBSVM package uses this approach.

These requirements for the separation boundary can be formulated mathematically by providing a measure $f(z)$ of the distance $d(z)$ to the positive class, or of the probability $p(z)$ of belonging to the positive class, and a threshold θ to distinguish between the positive class and the outliers [18]:

$$f(z) = I(d(z) < \Theta_d) \tag{5}$$

or

$$f(z) = I(p(z) > \Theta_d) \tag{6}$$

where I is the function indicating positive or negative class membership. One-class classifiers learn by optimizing the function $d(z)$ or $p(z)$. Some implementations go on to optimize the parameter θ while some use an a-priori defined threshold, usually provided by the parameter ν.

The error rate of a one-class SVM is estimated as the fraction of the positive (target) class f_{T+} versus the fraction of outliers that is rejected f_{O-}. The density distribution of f_{O-} needs to be estimated to calculate this error measure. Error E_I then is the group of positive examples rejected by the classifier, and E_{II} is the group of negative examples accepted by the classifier.

Various methods are available to estimate a distribution for Z: density methods, boundary methods, and reconstruction methods. The most simple of these is a Gaussian density model with a probability distribution as:

$$p_N(z;\mu,\Sigma) = \frac{1}{(2\pi)^{d/2}\left|\Sigma^{1/2}\right|}\exp\left\{-\frac{1}{2}(z-\mu)^T\Sigma^{-1}(z-\mu)\right\} \tag{7}$$

where μ is the mean, Σ the covariance matrix, and d the dimension of an object. More complicated probability and density functions have been studied as well. A notable difference with boundary methods is that these methods allow for multiple boundaries to cover a given positive class, but the parameters for the number of distinct boundaries need to be supplied.

Kernel functions can be applied to one-class SVM data points in the same way as for two-class SVMs, allowing more complicated data sets to be used with one-class SVMs.

2.2 Discussion: SVM and One-class SVM

The primary difference between a two-class SVM and a one-class SVM is the use of negative data points in the training of a classification functions. The one-class SVM approach has the advantage of being able to use a very small training set to learn a classification function.

This feature has been used successfully in [20] to study small-size genomes. When dealing with genes or proteins there already exists a very large amount of known data. It is not always desirable to use the entire dataset available to train classification functions for certain features or sequences. Additionally, it is also not easy to decide which of the data is relevant and which data can be left out when designing a classifier. The one-class SVM approach allows for a solution, as it only requires the data of the class to be discovered to learn a decision function. This allows for substantial savings in computation time and memory space, while maintaining a comparable level of accuracy.

3. Data Set and Feature Extraction

The algorithm presented in this paper attempts to determine structurally and functionally related protein domains at three different levels in the SCOP hierarchy: family, superfamily, and fold. The data for the experiments is taken from SCOP version 1.65. Using the subset of domains with <40% pairwise similarity from the ASTRAL compendium [2], all folds with at least 40 members are initially selected. This results in a data set with 1870 domains, comprised of 5 classes and 21 different folds. Each selected fold contains, on average, more than 10 superfamilies. This set is randomly divided into two-thirds training data and one-third testing data, resulting in 1247 training sequences and 623 testing sequences, representing members of all folds in each set.

3.1 SVM Training

This algorithm initially creates sequence profiles running six iterations of PSI-BLAST, generating the Position Specific Scoring Matrices (PSSM) and Position Frequency Matrices (PFM) for each training and testing sequence. Each profile is generated using PSI-BLAST default settings, except for the number of iterations, which is specified to be six (j=6). The PSSM and PFM output is stored to disk using parameter Q. Next, an all-against-all alignment of profiles in the training data is created. The alignment matrix m_{ij} of a profile q and a profile t is given by

$$m_{ij} = \sum_{k=1}^{20} \left[f_{ij}^{q} S_{jk}^{t} + S_{ij}^{q} f_{jk}^{t} \right] \quad (8)$$

where f, S are the PFM scores and PSSM scores of amino acid k at position i of profile q or at position j of profile t, respectively [5].
These 1246 m_{ij} alignment matrices for profile i (i=1, 2, …, 1247) of length n_i are used to extract (n+1) dimensional feature vectors s=(sa^1, sa^2, …, sa^n, total_score), where total_score is the total score of the alignment. The alignment feature vectors are then smoothed using

$$sa^{i} = m^{i-2} + 2m^{i-1} + 3m^{i} + 2m^{i+1} + m^{i+2} \quad (9)$$

where m^i is the profile alignment score at position i [19]. To produce comparable feature vectors, the individual scores are scaled down to values between 0 and 1. Feature vectors of alignments between profile i and profiles of the same fold as i are assigned the class label 1 (positive class), all other feature vectors are assigned class label 0 (negative class).

While the one-class SVM used in this approach only requires feature vectors belonging to the positive class, the full data set is computed to enable a performance comparison with the two-class SVM. Two feature vector files are produced: one containing only the positive class for each profile, and one containing positive and negative examples to be used with the two-class SVM.

The SVMs are then trained using the radial basis function (RBF) kernel. The training algorithm performs some basic parameter optimization for each of the 1247 two-class SVMs to be trained, and produces a trained model for each profile. For the one-class SVM, the parameter ν is set to the default value of ν =0.5, which indicates that 50% of the positive class feature vectors are treated as outliers. An optimization algorithm is used to determine the RBF kernel parameter gamma.

The freely available software LIBSVM [3] is used for all SVM training and testing. LIBSVM is available for download at http://www.csie.ntu.edu.tw/~cjlin/libsvm/. All programs written for this algorithm are implemented in Java, incorporating some of the LIBSVM Java source code. A task distribution system written in Fortran is used to distribute individual tasks over multiple CPUs. The algorithm was executed on a parallel machine comprised of dual and quad Intel Xeon 3200 EMT64 nodes communicating with MPICH2 under Linux.

3.2 SVM Testing

PSSM and PFM matrices are generated for the testing data in the same way as for the training data. Aligning each test profile with all training profiles generates a set of 1247 feature vectors for every test profile. These feature vectors are then evaluated with the corresponding trained SVM models. A score is produced for each result by summation of all positive evaluations. The training sequences with the highest-scoring results are taken as candidate targets for the test sequence.

3.3 Algorithm Flowchart

Start with ASTRAL subset of domains with <40% similarity.

1. Select folds with at least 40 members.
2. Use PSI-BLAST to generate profiles matrices (PSSMs and PFMs).
3. Divide data set: 2/3 training data, 1/3 testing data.
4. Training Data: Perform all-to-all profile-profile alignments. For the one-class SVM, only align within each fold group. For the two-class SVM, align all training profiles.
5. Training Data: Extract feature vectors from alignments, and scale and smooth feature vectors.
6. Training Data: Train one-class SVM and two-class SVM for each set of feature vectors.
7. Testing Data: Perform one-to-one profile-profile alignment between each test profile and all training profiles, producing 1247 feature vectors for each testing sequence (same for one-class SVM and two-class SVM).
8. Testing Data: Extract feature vectors from alignments, and scale and smooth feature vectors.
9. Testing Data: Evaluate feature vectors of each testing profile with all appropriate SVM models.
10. Testing Data: Sum results of each evaluation. Rank groups of results for each testing profile to get highest-scoring results.

This flowchart outlines the essential flow of data through our algorithm. Steps 4, 5, 6, 7 and 8 can be performed in parallel. Step 9 requires step 6 to be completed, as it needs the output of step 6 as input.

3.4 Performance Assessment

The results obtained in this paper compare the performance of a two-class SVM with the performance of a one-class SVM approach. The algorithm presented builds upon the work of Han et al in [5]. Han et al. used the same data set from the ASTRAL compendium version 1.65, selecting all folds with at least 20 domains, resulting in 62 folds and 2,854 domains. This data was pre-processed using the same algorithm as this paper. Two-class SVMs were trained using an RBF kernel, and the results were post-processed to produce statistically comparable results between all SVM classifications.

4. Experimental Results

Parameter selection is very important with SVM learning algorithms. This algorithm performs a simple grid-searching parameter optimization routine with 8-fold cross validation to select (LIBSVM-) SVM parameters *c* (upper bound in the number of errors allowed to occur) and RBF kernel parameter *gamma* for the two-class SVM learning process. The one-class SVM learning routine uses a basic algorithm to

optimize the RBF kernel parameter *gamma*. The SVM parameter *g*, which determines the fraction of training data points to include in the positive class, is left at the LIBSVM default value of 0.5.

Running this algorithm with the two-class SVM, and counting only the top-three scoring results as candidate solution produces an accuracy rate of 58.9%, which means that 58.9% of tested sequences contain the correct fold among the top-three scoring results. Running the same algorithm using the one-class SVM instead produced an accuracy rate of 54.9%. It is notable that very little parameter optimization has yet been attempted for the one-class SVM.

Table 1. Classification performance

	Two-Class SVM	One-class SVM
3-Best Scores	58.9%	54.9%

One-class SVMs require less time and storage space to run compared to two-class SVMs. For this algorithm, computational savings for the one-class SVM occur when generating the feature vectors, during parameter optimization, and SVM training. Significant space and time savings of the one-class SVM algorithm over two-class SVMs can be expected in steps 4, 5, and 6 of the algorithm described above. Comparing the space required of the algorithm, the two-class SVM requires 368.29MB to store the training feature vector files, compared to 25.17MB for the one-class SVM algorithm. This is a savings of 1 order of magnitude, and is due to the fact that the one-class SVM does not require any negative examples to be generated and stored. The evaluation of the test sequences is identical between the one-class SVM and two-class SVM approaches.

Table 2. Space requirements for training feature vector files

	Two-Class SVM	One-class SVM
Space Requirements	368.29 MB	25.17 MB

The advantages regarding time are due partly to the sufficiency of positive examples, which speeds up the feature vector generation phase, and partly because the training step (step 6) for the one-class SVM is completed faster than for the two-class SVM. This second component of savings varies with the kernel function used, and is much more pronounced using a linear kernel as compared with the RBF kernel. Utilizing 20 CPUs, the one-class SVM training step was completed in 3 hours. By comparison, the two-class SVM training step, utilizing 40 CPUs, took 70 hours.

Table 3. CPU time requirements for SVM learning

	Two-Class SVM	One-class SVM
	40 CPUs	20 CPUs
Time Requirements	70 hours	3 hours

These improvements regarding space and time requirements enable the one-class SVM based algorithm to scale much easier to larger data sets. Adding more sequences

to the training data only affects the subset of SVMs for the same fold as the new sequences. The two-class algorithm requires feature vector generation, and re-training, for all training SVMs.

5. Conclusions and Future Work

In this paper we introduced a one-class SVM approach to detect remote relationships between protein sequences with very low sequence similarities. The algorithm begins by generating sequence profiles for all training sequences using PSI-BLAST. In the next step profile-profile alignments between each training sequence and all other training sequences are generated. For the one-class SVM algorithm, only alignments between sequences belonging to the same fold are required. The two-class SVM algorithm assigns all sequences belonging to the same fold to the positive class, and all remaining sequences to the negative class. This results in a set of feature vectors for each training sequence. Individual SVMs are then trained for each training sequence, using either a two-class SVM or a one-class SVM and using the Radial Basis Function kernel.

The advantages of the one-class SVM approach become apparent when comparing the reduced time and space requirements, which show a significant improvement over the two-class SVM approach.

The algorithm presented in this paper produces initial protein sequence profiles by running six iterations of PSI-BLAST, and using the resulting PSSM and PFM matrices to perform profile-profile alignments. A different approach to profile-profile alignments has been given in the recent literature by Söding in [16], where alignments are produced using profile HMMs instead of PSSMs and PFMs. This approach may well be capable of improving the performance of this algorithm.

A second focus of improvement lies in the kernel function used for SVM learning and classification. Several papers in the recent literature have presented novel kernel functions aimed at improving the ability to detect remote sequence relationships, as for example in [13]. This offers a second promising direction of future research to improve the performance of this algorithm.

References

1. Altschul, S. F, et al.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research Vol. 25. (1997) 3389-3402.
2. Brenner SE, Koehl P, Levitt M.: The ASTRAL compendium for sequence and structure analysis. Nucleic Acids Research. Vol. 28. (2000) 254-256.
3. Chang, Chih-Chung and Chih-Jen Lin: LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm/. (2001).
4. Chen, Yunquiang, Zhou, Xiang, and Thomas S. Huang. One-Class SVM For Learning in Image Retrieval. Proc. IEEE Int'l Conf. On Image Processing (ICIP). 7 Oct.-10 Oct. 2001. Vol. 1. (2001) 34-37.
5. Han, S., B.-c. Lee, et al.: Fold recognition by combining profile-profile alignment and support vector machine. Bioinformatics. Vol. 21(11). (2005) 2667-2673.

6. Jones, D.T.: GenTHREADER: an efficient and reliable protein fold recognition method for genomic sequences. Journal of Molecular Biology. Vol. 287. (1999) 797-815.
7. Karplus, K., et al.: Predicting protein structure using only sequence information. Proteins. Suppl 3. (1999) 121-125.
8. Kelly, L. A., et al.: Enhanced genome annotation using structural profiles in the program 3D-PSSM. Journal of Molecular Biology. Vol. 299. (2000) 499-520.
9. Liao, Li, and William Stafford Noble.: Combining Pairwise Sequence Similarity and Support Vector Machines for Detecting Remote Protein Evolutionary and Structural Relationships. Journal of Computational Biology. Vol. 10(6). (2003) 857-868.
10. Manewitz, Larry M., and Malik Yousef.: One-Class SVMs for Document Classification. Journal of Machine Learning Research. Vol. 2(3). (2001) 139-154
11. Murzin A. G., Brenner S. E., Hubbard T., Chothia C.: SCOP: a structural classification of proteins database for the investigation of sequences and structures. J. Mol. Biol. Vol. 247. (1995) 536-540.
12. Onoda, Takashi, Murata, Hiroshi, and Yamada, Seiji. One Class Support Vector Machine based Non-Relevance Feedback Document Retrieval. Proc. IEEE Int'l Joint Conf. on Neural Networks (ICJNN). 31 July-4 Aug. 2005. Vol. 1. (2005) 552-557.
13. Rangwala, Huzefa, and George Karpys. Profile-based direct kernels for remote homology detection and fold recognition. Bioinformatics. Vol. 21(23). (2005) 4239-4247.
14. Saigo, Hiroto, et al.: Protein homology detection using string alignment kernels. Bioinformatics. Vol. 20(11). (2004) 1682-1689.
15. Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., Williamson, A.: Estimating the support for a high-dimensional distribution. Microsoft Research, One Microsoft Way Redmond WA 98052, Tech. Rep. MSR-TR-99-87, 1999.
16. Söding, Johannes. Protein homology detection by HMM-HMM comparison. Bioinformatics. Vol. 21(7). (2005) 951-960.
17. Spinosa, Eduardo J. and de Carvalho, Andre C.P.L.F.: Support vector machines for novel class detection in Bioinformatics. Genet. Mol. Res. Vol. 4(3). (2005) 608-615.
18. Tax, David M. J.: One-class classification-concept learning in the absence of counter-examples. Ph.D. Dissertation, Delft University of Technology. ASCI Dissertation Series. Vol. 65. (2001) 1-190
19. Tress, M. L., et al.: Predicting reliable regions in protein alignments from sequence profiles. Journal of Molecular Biology. Vol. 330. (2003) 705-718.
20. Tsirigos, Aristotelis and Isidore Rigoutsos: A sensitive, support-vector-machine method for the detection of horizontal gene transfers in viral, archaeal and bacterial genomes. Nucleic Acids Research. Vol. 33(12). (2005) 3699-3707.
21. Vapnik, V. N. (1995) The Nature of Statistical Learning Theory. Springer-Verlag, New York.
22. Zhang, Jianguo, Ma, Kai-Kuang, Er, Meng Hwa, and Vincent Chong. Tumor Segmentation from Magnetic Resonance Imaging by Learning Via One-Class Support Vector Machine. International Workshop on Advanced Image Technology (IWAIT04). (2004) 207-211.