A Review of Support Vector Machines in Computational Biology

Brett Naul

June 6, 2009

1 Introduction

In the biological sciences, arguably moreso than in any other discipline, the amount of data is available to researchers is exploding exponentially. Making this information available in a consistent, accessible format is itself a non-trivial task, and categorizing or classifying the data in meaningful ways is especially daunting. Laboratory experiment and human review will likely continue to represent the gold standard for reliability in these classification tasks, but these processes are time-consuming and expensive. Consequently, the development of automated replacements for these is an extremely important and widely-studied problem. Due to the scope and complexity of the data in question, statistical and machine learning algorithms are a natural choice for these types of analyses. Support vector machines are a specific type of machine learning algorithm that are among the most widely-used for many statistical learning problems, such as spam filtering, text classification, handwriting analysis, face and object recognition, and countless others. Support vector machines have also come into widespread use in practically every area of bioinformatics within the last ten years, and their area of influence continues to expand today. This paper will present a brief description of support vector machines themselves, followed by a comprehensive study of how SVMs have been and are being used in computational biology, and finally a brief discussion of some novel variations on SVM classification as well as the future of SVMs in bioinformatics.

2 Support Vector Machines and Supervised Learning

Most of the problems discussed in this paper will involve "classification" of data, i.e. placing items into categories based on some set of features which describe the data. Support vector machines in particular were designed as a tool to solve supervised learning classification problems. Supervised learning is an area of machine learning in which we are given some "training set" of data for which we know *a priori* the appropriate classifications. Using these labeled items, a statistical model is developed to distinguish between positive and negative examples, after which any new data point can be classified according to this model.

The support vector machine algorithm classifies input using a fundamentally geometric idea. The ultimate goal of the support vector machine algorithm is to express the data as elements of some vector space, and then construct a hyperplane (or hyperplanes) that appropriately separates the data. For a binary classification problem, this corresponds to finding a hyperplane such that on side contains all the examples from one category (let y^i denote the label of the ith data point, and call these points $\{x^i | y^i = 1\}$), and the other side contains all the points for which $y^i = 0$. Typically, when there are more than two categories, the problem is solved by reduction to a number of simpler binary problems. Once this hyperplane has been constructed, the algorithm makes predictions by checking which side of the hyperplane a test vector lines on; if it lies on the "1" side, it is put in that category. Notice that there could be infinitely many such separating hyperplanes; in this case, it chooses the optimal one based on the idea of maximizing the "margin" of the data, i.e. the distance from the decision boundary. In general, learning algorithms are evaluated based on their generalization error, or the percentage of a previously unseen data set that is misclassified. Maximizing the classification margin provides the best chance that new points will lie on the appropriate side of the boundary.

Since our desired decision boundary is linear, we can think of predictions as taking on the form

$$h(x) = \text{sign}(w^T x + b),$$

where w is some vector of "weights," and b is an intercept term. Clearly, the linear equation $w^T x + b = 0$ defines a hyperplane in the vector space inhabited by our feature vectors x^i. Thus, by examining whether $w^T x + b$ is greater or less than zero at a point x, we are able to determine mathematically whether a point in our high-dimensional feature space is "above" or "below" the separating hyperplane. Recall that we are interested in maximizing the margin between the data and the boundary: there are many intuitive ways to express this as an optimization problem, but using some algebra, the problem can be written as

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 \\ &\text{s.t. } y^i(w^T x^i + b) \geq 1 \ \forall \ i. \end{aligned}$$

This is a convex quadratic program, and can be solved by any number of commercial optimization software packages. However, it is often easier to solve an equivalent problem

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^i y^j \alpha_i \alpha_j (x^i)^T x^j \\ &\text{s.t } \alpha_i \geq 0 \ \forall \ i, \\ &\quad \sum_{i=1}^{m} \alpha_i y^i = 0, \end{aligned}$$

which can be derived using some ideas from optimization theory (Lagrange duality, the Karush-Kuhn-Tucker conditions, etc.). It is also straightforward to change the optimization problem to allow for cases in which the data cannot be perfectly separated, or in which certain outliers change the classifier more than is desired. These modifications do significantly change the overall complexity of the implementation or the computation.

The above constrained optimization is a problem in m variables, where m is the size of the training set. Anyone familiar with mathematical programming or any other large-scale computation should also be familiar with the curse of dimensionality and its implications for this algorithm. In order to obtain high accuracy and robustness, it would be beneficial to consider as many training examples as possible. However, the resulting increase in the dimensionality of the problem will greatly slow the convergence of the above optimization; in general, the time required for such a computation is in most cases $\Omega(m^2)$, although for some specialized inputs the required time can be as low as $\Theta(m^2)$.

One additional advantage of support vector machines is their compatibility with the so-called "kernel trick," which involves replacing the vector products from the original formulation with some new (non-linear) kernel function. The result is that points are compared in some higher-dimensional space, and so the model (and resulting decision boundary) will appear non-linear in the input space. This results in a greatly increased variety of possible classifiers, as well as far more potential for complexity and accuracy. In general, the most popular choices of kernel functions are the polynomial kernel $k(x, x') = (x \cdot x')^d$ and radial or Gaussian kernel $k(x, x') = \exp(-c\|x - x'\|^2)$. The choice of the optimal kernel for a given problem is a nontrivial decision in itself, and uses of the kernel trick specific to computational biology will be discussed in subsequent sections.

3 Computational Biology Applications

3.1 Protein Similarities and Homologies

Support vector machines are a robust and powerful classification method, but only within the last ten years or so have they started to see widespread use in the biological sciences. One of the areas in which SVMs first gained significant traction was the identification of similar protein sequences and homologies. Algorithms for these tasks have existed for dozens of years, some of which first compute optimal sequence alignments (e.g. the Smith-Waterman dynamic programming algorithm), and others that use heuristic approximations (BLAST, FASTA, etc.). More complicated methods include profiles, PSI-BLAST and hidden Markov models, all of which construct some model based on aggregate statistics from a number of related sequences, and use the result as a reference against which other proteins can be compared. These algorithms provide broader and more accurate results, returning more true positives and a smaller fraction of false positives, but they still miss many more distant (but still relevant) relationships and homologies.

A major breakthrough was provided by Jaakkola *et al.* in 1999 with their introduction of the Fisher kernel method of detecting remote protein homologies[46]. This method expands on the standard hidden Markov model-based search by incorporating a support vector machine classifier into the identification process. Using a HMM representation of a given protein family, we can produce some probability estimate that a test protein belongs to that family. However, such a predictor can easily fail if the underlying model is not perfectly accurate, or if the set of training data is insufficiently large. On top of these probabilities, the Fisher kernel method makes use of a quantity called the Fisher score, given by

$$U_x = \nabla_\theta \log P(x|H_1, \theta),$$

i.e. the gradient of the log-likelihood of the probability that X is in the protein family with respect to some parameter θ. These Fisher scores are themselves fixed-length vectors which can be used as input to a standard SVM classifier. The SVM classifier makes use of the standard Gaussian kernel

$$K(x, x') = \exp(-\frac{1}{2\sigma^2}\|U_x - U_{x'}\|^2),$$

but in the context of using Fisher scores, rather than some literal sequence encoding, it is known as the Fisher kernel. The Fisher-SVM improved in many ways over previous state-of-the-art remote homology identification schemes, and served as a springboard for a great deal of related SVM research.

Another SVM algorithm, Liao and Noble's SVM-pairwise, uses the results of a pairwise sequence comparison algorithm (e.g. Smith-Waterman) in the same way that Fisher-SVM uses a hidden Markov model as its underlying statistical basis[25]. The input vectors for the algorithm are constructed using the probabliity sores from the comparison search, and a standard (Gaussian, polynomial, etc.) kernel is used to compare two such vectors. The SVM-pairwise implementation benefits from the great deal of work that has gone into fine-tuning Smith-Waterman and other algorithms, but at a computational cost: computing the kernel matrix requires computing *a priori* the similarity scores for each of the training data points. Using a heuristic algorithm such as BLAST can decrease the computational burden of this step, but at a cost of decreased accuracy. Its run-time notwithstanding, SVM-pairwise is a very effective algorithm for remote homology detection: according to Liao and Noble, it improves greatly on both PSI-BLAST and SVM-Fisher in classifying previously-unseen families.

Just as there are many different kernels that one must consider in implementing a classification algorithm, there are also countless ways to represent an alphabetical or otherwise non-numeric item as a vector quantity to which a Support Vector Machine can be applied. One such technique is the so-called "composition method," which represents a protein sequence by a vector of twenty amino acid frequencies (or a DNA sequence by a vector of four nucleic acid frequencies). Ding and Dubchak showed that, using a variety of kernels (in particular polynomial and Gaussian), SVMs using this method of feature representation performed well in protein fold recognition experiments, and greatly ourperformed comparable neural networks[14]. Ding and Dubchak also suggested novel methods of partitioning a multi-class classification into several binary classification problems. The standard way of doing so, described in their paper as the one-versus-others method, involves setting some class to be one category and all other classes as the other, and making a binary prediction for each such partitioning. However, the intricate relationships between categories can lead to high numbers of false positives, which suggests other alternative schemes. In the unique one-versus-others method, all of the positive results are then compared to one another, resulting in one unambiguous prediction for the protein. In the all-versus-all method, the algorithm trains a classifier for each pair of categories, and returns the category with the most "victories" as the prediction. The unique one-versus-others method resulted in 13.6% improvement over the standard one-versus-others predictions, and all-versus-all exhibited 24% improvement, according to their experiments. However, these methods also come with a cost of increased computation: unique one-versus-others requires the computation of K classifiers (where K is the number of potential categories), and all-versus-all uses a daunting $K^2 - K$ classifiers (each of which itself has quadratic complexity).

Although the simplicity of the composition method is an attractive feature from an implementation persepective, in terms of results it is less desirable. The reduction of a protein sequence of hundreds or thousands of amino acids to a vector of twenty frequencies disregards a huge quantity of information, including any trace of the proximities of various amino acids within the chain. A more complex means of representing and comparing proteins is given by motif-based methods. In 2001, Logan *et al.* proposed a high-dimensional protein representation consisting of similarity scores drawn from the BLOCKS motif database[4]. These feature vectors are then compared and classified using normal SVM techniques (Logan *et al.* make use of the Gaussian kernel).

The motif-based representation approach was improved upon by Ben-Hur and Brutlag in 2003[4]. Their implementation instead used the eBLOCKS motif database, which includes on the order of half a million motifs. This is far more than Logan's implementation made use of, and would make for far too high-dimensional a problem if used in the same way. However, using a vector representation would also be extremely wasteful, since a sequence typically matches only a handful of motifs. Here the kernel trick once again comes into play: for the computation of the model, and for making classifications, it is only necessary to compute the kernel function of two input vectors; the vectors themselves need never be manipulated or even stored. Using a trie structure, Ben-Hur and Brutlag were able to reduce the computation time of the motif content of a sequence to $O(m)$, where m is the length of the sequence. The resulting motif-based classifier provides significant improvement over previously-described methods.

String kernels are a type of kernel which has been used historically for text classification but has also seen use in homology detection. Fundamentally, string kernels represent the data using features consisting of substrings from the text or protein sequence in question. The most comprehensive such representations would include every substring as a feature, but this is in almost all cases computationally infeasible. Thus, some manageable portion of this power set is instead included as possible features, and the kernel function compares the relative occurence of these subtrings within the sequences. Eskin *et al.* proposed the so-called "spectrum kernel" as one solution to this problem: for the feature vector consists of all possible $k^{|\mathcal{A}|}$ subsets of length k for some pre-determined k, drawn from the alphabet $\mathcal{A}$[8]. In the case of $k = 1$, this scheme is identical to the composition method. As in the case of Ben-Hur and Brutlag's motif kernel, storing the feature vectors for all the data is almost certainly impractical, but computing the kernel function of two inputs can be done quickly using tries.

The basic idea of the string kernel is an appealing simple one: in some sense we are comparing the sequences as directly as possible by examining their string representations, and in doing so we discard as little proximity data and other information as possible. There have been many other attempts made to implement variations on this simple string kernel idea. The mismatch kernel was presented by Leslie *et al.* as an extension on the spectrum kernel: it uses the same structure for its features, but also counts as matches any substrings which are dislike in at most M places, for some threshold M[9]. Leslie and Kuang also published another extension which builds on the idea of the mismatch kernel and introduces the gappy kernel, the substitution kernel, and wildcard kernels. The functions of these kernels are rather self-explanatory: the gappy kernel allows matches between strings with up to α gaps inserted (for some tolerance α); the substitution kernel allows mismatches, but only between certain subsets of amino acids; and the wildcard allows for up to M occurences of the character $*$ in an alignment, and counts two strings as matching as long as no two unlike non-$*$ characters are aligned[22]. All of these methods, along with the mismatch kernel, are computable in linear time, allowing for an enormous variety of features.

One potentially major paradigm shift is exemplified by the cluster kernel method. Clustering is an unsupervised learning technique that arranges unlabeled data into k distinct subsets; the optimal value of k can be determined through various cross-validation procedures. Using clustering techniques, Weston *et al.* (among many other groups) were able to incorporate unlabeled protein sequences into their underlying statistical model, and thereby greatly increase the scope of the resulting matches. The resulting algorithm can no longer be classified as supervised learning because of the inclusion of unlabeled data, so it is instead commonly referred to as "semi-supervised learning." This type of algorithm can be applied across many areas of computational biology, and is not unique to homology detection. The clustering process requires a great deal of computation and matrix manipulation, but recent implementations (including those by Weston *et al.*) have brought this down to practical levels, and their algorithm is able to achieve accuracy on par with other state-of-the-art methods[47].

In addition to the aforementioned (and other) basic techniques of classification, there are countless possible hybrid approaches which combine aspects from multiple methodologies. One such recent algorithm was developed by Kuang *et al.*: the basic idea is the same as that of the string kernel, in that similarity is computed by measuring the number of matches between subsets of fixed length k[38]. However, this algorithm also makes use of an underlying probabilistic profile structure, generated e.g. by PSI-BLAST. Using these probabilities allows for more precise string-based comparisons (compared to the spectrum kernel method and its derivatives): rather than accepting any or no mismatches, we can weight the mismatches according to the corresponding transition probabilities from the PSI-BLAST model.

One more hybrid method was published by Damoulas and Girolami in 2008[12]. It, too, uses substrings and string kernels as its representation and comparison methods, but it also draws from Ding and Dubchak's insight regarding the effectiveness of all-versus-all training. In Ding and Dubhack's paper, they required the computation of $K * (K - 1)$ distinct classifiers, one for each pair of possible categories. Damoulas and Girolami manage to bypass this step by embedding each object description into a kernel space, and using the resulting similarity metrics to combine this information into one composite kernel space. Using this new kernel space, only one classifier must be trained in order to have all the predictive power of all-versus-all training, and thus the computational burden is greatly reduced.

3.2 Prediction of Protein Structure and Function

A related but non-identical problem to homology detection is that of predicting structural and functional characteristics for a previously unseen protein sequence. Unsurprisingly, SVMs have been utilized with great success in this area as well, and many of the same types of models are used to solve both problems. One example of a nearly identical implementation is that of Pavlidis *et al.*, which uses the Fisher kernel in order to classify genes according to promoter sequences[32]. Just as in Jaakkola's original implementation, the process is begun by identifying the promoter regions from some target set of genes that can be labeled in advance. Next, a (motif-based) hidden Markov model is trained using this set of sequence data, and the likelihood gradients with respect to this model are used as the feature vectors to represent each target sequence. The training data is then analyzed and a support vector machine trained using the standard kernels (Gaussian, polynomial, etc.), and this model is then used to make predictions for the new data. According to the authors, construction of reliably pre-classified training and tests sets for this problem is challenging (compared to the homology detection problem), but nonetheless their results demonstrated the potential for the expansion of SVM techniques into this area of research.

As a rule, algorithms that make predictions regarding protein structure and/or function seem to be more specialized than similar algorithms for distant homology detection. Whereas the algorithms from the previous section are all able to handle practically any protein sequence, techniques from this section will tend to focus on predicting one specific attribute. While these implementations may seem to be lacking in robustness, they certainly match the accuracy of previously-discussed SVM methods. One such algorithm is the SVM approach for predicting subcellular localization of proteins from amino acid compositions, due to Hua and Sun[38]. Before Hua and Sun's paper, the state of the art prediction technique used neural networks based on individual sorting signal predictions. Although this method provides good results in the case where the gene 5'-regions are accurately aligned, this often cannot be taken for granted, and thus the results of the neural network predictors cannot be considered totally reliable. The SVM classifier, on the other hand, also produces accurate results in the easy case when trustworthy alignments are available, but is also far more robust to errors in the gene 5'-region annotation. The kernel function used by the algorithm is a simple composition kernel, i.e. a comparison of the twenty-element frequency vectors correspondiing to each protein sequence. A similar attempt by Park and Kanehisa using a slightly more complicated kernel, which took into account amino acid pair and gapped pair compositions, performed even better on the same data[34].

Some SVM methods have been able to make predictions that were previously impractical or impossible. As of March 2009, Nassif *et al.* claimed to have developed the first and only glucose-binding site classifier algorithm[28]. In their model, binding cites are represented as spheres centered at the ligand. The components of the feature vectors considered by the algorithm include atom type, functional group, residue, charge, hydrophobicity, etc. Like most of the implementations seen so far, Nassif *et al.* use the standard Gaussian kernel to evaluate similarity between vectors. If anything, this implementation is an especially straightforward application of SVMs: the feature vectors are simply vectors of characteristics, and no unusual kernel trick is applied, in contrast with sequence-based methods that had to deconstruct a length sequence into a compact form in order to properly compare them. According to their paper, Nassir *et al.* achieved 89.66% sensitivity and 93.33% specificity over their chosen data set of glucose binding (positives) and non-glucose binding (negative) sites, which seems to be a rather promising preliminary result.

There are dozens, if not hundreds, of similarly innovative results that one could enumerate, so the focus here will be on newer results, e.g. from within the past year. Cheng *et al.* proposed a new method, RNAProB, in December 2008 for predicting RNA-binding sites of proteins using SVMs[10]. Their approach also incorporates evolutionary information derived from PSSM profiles; on their data set, their implementation improved on state-of-the-art methods by about 5% in terms of both overall accuracy and sensitivity. In May 2009, Nugent and Jones presented an “SVM-based transmembrane protein topology predictor that integrates both signal peptide and re-entrant helix prediction”[31]. By their benchmarks, this method achieves an unprecedented level of both sensitivity and specificity, and they conclude that their algorithm is ideally suited to identifying alpha-helical TM proteins throughout entire genomes. In April 2009, Gao *et al.* developed a new method for predicting protein function changes associated with single amino acid substitutions, which they call “substitution-matrix-based kernel SVM”[16]. Their kernel function is derived from entries from the substitution score matrix BLOSUM62, which gave results comparable to those of SNAP, a neural-network based algorithm designed for the same purpose. However, the SVM method is easier to implement and learns more quickly than the neural network approach. Qiu *et al.* published a new method for differentiating between enzyme and non-enzyme proteins in November 2008; their approach combines support vector machines with the so-called discrete wavelet transform, resulting in improved predictive performance[19].

Intuitively, if SVMs are well-suited to protein classification, then they should also be effective tools for gene classification. A paper by Ma *et al.* from March 2009 lent some credence to this intuition: they attempted to classify genes based on the relative frequency of various codons, which composed their 59-dimensional feature vector[20] Their technique proved to be more robust than the standard methods, in particular for sequences of different lengths. In attempting to classify Human Leukocyte Antigen (HLA) sequences into two major groups, HLA-I and HLA-II. Their SVM model achieved 99.3% accuracy within this data set, and produced results consistent with the known biological functions and molecular structures of the molecules. Another gene classification algorithm, GISMO, by Krause *et al.*, is a novel SVM-based method of discovery and classification for prokaryotic genome sequences[23]. Although this problem already has many effective known solutions, the authors feel that this algorithm’s effectiveness, especially with regard to hard-to-classify short gene sequences, improves sufficiently on existing methods that it warrants discussion as a replacement for those well-established techniques.

3.3 Microarray Data Analysis

Around the same time that SVMs began to be applied to protein homology searches, they also started to be used in the analysis of microarray data. Microarray experiments are used to measure changes in expression levels between different types or populations of genes. The primary advantage of microarrays is that they can be used to generate large quantities of data quickly and cheaply; their primary disadvantage is that the resulting data is often quite noisy and hard to analyze. As with any difficult problem with large amounts of disposable data, statistical learning methods are a natural recourse, and SVMs are among the best-performing microarray analysis tools to date. Brown *et al.*, in 1999, were perhaps the first to publish an SVM-based method for directly classifying genes based on microarray data[26]. The result of a single microarray experiment is a list of n ratios of expression levels between samples 1 and 2, where n is the number of genes on the chip; the result of a series of m experiments is then an $n \times m$ matrix, and each row of length m can be thought of as a single data point. Brown *et al.* used these features along with both the polynomial and Gaussian kernels. They trained their SVMs to distinguish between six functional classes, and compared their performance with that of four other standard algorithms: Parzen windows, Fisher's linear discriminant, and C4.5 and MOC1 decision trees. The accuracy of both kernels, in particular the Gaussian kernel, surpassed those of all four alternative machine learning techniques in terms of overall error rate.

Another early application of SVMs to microarray data was that of tissue classification, one that remains popular today. Numerous studies released around the same time all achieved similarly encouraging results: Golub *et al.* in 1999 provided the first tissue classification algorithm using SVMs, applied to the problem of differentiating between two types of leukemia[42]. Their chosen feature set was based on the signal-to-noise ratio

$$P(j) = \left| \frac{\mu_A(j) - \mu_{-1}(j)}{\sigma_A(j) - \sigma_{-1}(j)} \right|,$$

where $u_i(j)$ is the mean expression level for class i and $\sigma_i(j)$ the per-class standard deviation for gene j. The same feature set was used by Furey *et al.* in 2000 in their classification of ovarian cancer tissues[26]. Both implementations outperformed Naive Bayes and other standard machine learning techniques that were typically used for these tasks.

As with the already-discussed protein sequence analysis methods, applications of SVMs to microarray data continue to develop and achieve higher accuracy and robustness. One particularly intriguing innovation is due to Fotiadis *et al.* from 2008. As previously described, the primary obstacle to overcome in interpreting microarray data is its typically noisy, random quality. Fotiadis *et al.* used SVMs to classify the pixels themselves, either into two groups (foreground and background) or three (signal, background and artefact)[18]. This type of partitioning is typically done using clustering and other unsupervised machine learning algorithms, but their implementation manages to achieve extremely high accuracy. The pixels themselves are represented by vectors of eleven distinct features, which measure the intensity of the pixel, the intensity of its neighbors, and the variation within its neighbors, among others. The classifiers are trained on already-classified data, and are tested on real new microarrays and simulated microarrays, along with various types of preprocessing filters. In every case, the sensitivity of the classifiers exceeds 98%; the accuracy and specificity exceed 99.8%.

Another interesting and recent development is due to Zhu *et al.* from 2009[52]. They approach the same standard binary classification problem as previous researchers, but incorporate network-based information into their training programs. More specifically, an underlying model of statistically significant subnetworks is constructed by searching various subnetworks and assigning scores based on each subnetwork's gene expression level; the algorithm then identifies subnetworks that are capable of discriminating effectively between categories. Using this information, a penalty term is constructed which penalizes contradictions between some classification and the corresponding subnetwork model(s). This penalty term is added to the optimization program itself, rather than incorporated into the feature space or the kernel function, but the effect is still that the resulting classifiers are biased towards the underlying subnetwork models. According to the authors, this technique improves the consistency of the SVM classifier, and also allows for the extraction of higher-level biological data which is available in other databases and formats.

Recently, another type of classifier has been gaining traction in microarray classification, namely random forests. Random forests are another type of machine learning algorithm which consist of many randomly-generated (by a bootstrap-like process) decision trees. The output of the classifier for a given input is the most popular result among all the random trees. In July 2008, Statnikov *et al.* released a comparison of random forests and SVMs for classifying cancer tissue based on microarray data[44]. According to the authors, random forests, despite their increasing popularity, are still not as accurate as SVMs for typical microarray classification problems. Their metric for the comparison was the area under the respective ROC curves, as well as the relative classifier information (RCI), an entropy-based measure that can be applied to multi-class decision problems. By these measures, SVMs outformed random forests on nine of eleven and eleven of eleven tasks, respectively.

4 Pharmacology

On a different note, a less theoretical, more imminently practical application of machine learning techniques is the analysis of pharmaceutical data and prediction of interactions between drugs and proteins. Since SVMs are widely used and successful in predicting interactions and relationships between proteins and genes, it is no surprise that similar techniques can be used in the design of drugs which are designed to minimic or prevent such interactions. Burbidge *et al.* concluded that SVMs are "a robust and highly accurate intelligent classification technique, likely to be well suited to SAR [structural activity relationship] analysis," even for large numbers of compounds[36]. The data used by the group was the inhibition of dihydrofolate reductase by pyrimidines; the activity was measured as the log of the equilibrium constant for the association of the drug to dihydrofolate reductase. Using this measure, the resulting classification problem is to determine, for all drugs m, n, the structure of the function $F(d_n, d_m)$, where F is an ordering function which equals 1 if the activity of drug n is higher than that of drug m, and -1 if it is lower. Their results showed that

SVMs were more accurate than every other algorithm studied with the exception of a certain neural network, which had a prohibitively long training time.

A related problem was addressed by Warmuth *et al.*: given a large collection of compounds, discover which bind to a molecule as quickly as possible[47]. Specifically, their algorithm employs "active learning," which involves selecting a batch of unlabeled components at each iteration to be test. This process mimics the typical drug discovery process, which is itself an iterative procedure in which the current data determines the drugs to be tested during the next round. For their feature set, the researchers chose the (rather ambiguous) "DuPont Pharmaceuticals...shape feature and pharmacophore descriptors." By the eighth round of testing, the SVM classifier had correctly identified 93% of the active compounds and nearly 100% of the inactives, which outperformed (to varying extents) the voted perceptron, Bayes point machine, and k-nearest neighbor algorithms.

A different approach was taken by Bao and Sun, who used microarray data to measure the expression recorded by a 1217×50 drug matrix and a 4864×50 gene matrix for some set of hypothetically anticancer compounds[1]. As one would expect after reading the previous section, SVMs performed admirably, exhibiting both high specificity and sensitivity. However, the researchers also pointed out some drawbacks of this method of classification: most notably, this method can identify only relationships which are present at the transcription level, which is only a fraction of all the possible causal relationships. Furthermore, this supervised learning approach requires previous (labeled) knowledge of drug mechanisms for the training data, which is available for only a small fraction of potential compounds and genes. The former issue is specific to the methodology of this experiment; the latter is symptomatic of a general problem with support vector machines and supervised learning, namely that we require reliably labeled data to produce reliably label data. In this instance, it is possible that some of the semi-supervised learning techniques from previous sections, e.g. cluster kernels, could help to alleviate this issue.

5 Implementations

The descriptions from previous sections allude occasionally to arguably the biggest drawback of support vector machines, namely that training them can be time- and space-intensive, even for moderate input sets. Consequently, the optimization algorithm which underlies the training procedure is itself an area of active research, and faster and faster convergence is constantly being achieved.

Some techniques for improving the convergence of the algorithm make heuristic compromises and sacrifice some classification accuracy for more reasonable training time. One recent such implementation, published by Kramer *et al.* in May 2009, employs bit reduction in order to reduce the complexity of the input data[22]. More specifically, the data is first normalized by multiplying by some Z large enough that a sufficient number of digits from each input value are contained in the integer part of $Z * v$. We then perform bit reduction on each value of $I(v) = \text{int}(Z * v)$ until the desired compression factor is reached. As nearby training examples are merged into the same bin, it is important to keep track of how many belong in each category. Intuitively, if we had very many close values and a few outliers, then merging the close values without somehow re-weighting them would greatly bias the resulting classifier towards those outliers. The resulting optimization problem is very similar to the standard SVM mathematical program, but its dimensionality is greatly reduced, and thus convergence is greatly facilitated. Using the sample problem of image recognition (specifically, identifying plankton), the classifier went from 89.6% accuracy and 24.3s training time with no compression to 84.5% accuracy and 4s training time. For a small drop in accuracy, we receive an enormous reduction in training time, and it is quite possible that the same compression schemes could be used in conjunction with the algorithms from previous sections in order to greatly expand the variety of training examples.

Other schemes, such as those of Collobert *et al.* and Huang *et al.*, attempt to partition the training process into a number of smaller cases, which reduces the dimensionality and lends itself to paralellization[12][16]. Collobert *et al.* produce N distinct classifiers, each based on m/N training examples, and solving a related optimization problem that can be thought of as minimizing the sum of the squared errors of each of the N classifiers. Huang *et al.*, by comparison, solve for each of the N classifiers normally, and then use a gate network to form a weighted average of the predictions of each classifier, resulting in one final prediction. These two papers have been cited by numerous computational biology researchers, which seems to indicate that these methods are as applicable to the biological sciences as they are to other classic machine learning problems.

One final SVM innovation worth mentioning is the SVM-Fold algorithm, released in 2007 by Melvin *et al.*[27]. This algorithm was constructed with protein classification in mind, and is designed to avoid the inaccuracies and inefficiencies inherent in the standard "one-versus-all" classification method. In particular, the training is done using an "adaptive multi-class codes algorithm," which uses a weighted combination of one-versus-all classifiers to take advantage of the dependence between the classifier results and leverage it towards improving the predictions of the ultimate aggregate classifier. The kernel itself is a string kernel based on PSI-BLAST profiles, and the resulting predictor outperforms both PSI-BLAST and standard one-versus-all SVM classifiers. More importantly, SVM-Fold has an easy-to-use public web interface available at http://svm-fold.c2b2.columbia.edu, which could help speed adoption of SVM methods among the bioinformatics community (hence its placement here, rather than in Section 3.1).

6 Conclusion

The previous examples, along with many others, make a strong case for the importance and effectiveness of support vector machine classifiers in computational biology. SVMs have a number of distinct advantages over other statistical classification models, including low bias, high customizability due to numerous possible choices of kernels and feature models, and generally low generalization error. They also have obvious drawbacks, most notably the influence of the curse of dimensionality and the generally slow training speed. Recent developments have demonstrated that there is some hope of conquering these obstacles, even without unforseen advances in computing power. Other related algorithms which may gain popularity in the future include support vector regression (see [28]), which has been employed sparingly but with impressive results in computational biology, and semi-supervised classification algorithms such as cluster kernels, which are able to make use of the increasingly massive quantity of unclassified data available to researchers. But even as things stand today, support vector machines represent state-of-the-art methods for solving many of computational biology's largest and most complex problems.

7 References

1. Bao, L., and Sun, Z. R., Identifying genes related to drug anticancer mechanisms using support vector machine. FEBS Lett. 521:109-114, 2002.

2. Ben-Hur A, Ong CS, Sonnenburg S, Schlkopf B, Rtsch G, 2008. Support Vector Machines and Kernels for Computational Biology. PLoS Comput Biol 4(10).

3. Ben-Hur, A. and Brutlag, D. (2003). Remote homology detection: a motif based approach. Bioinformatics, 19.

4. Logan, P. Moreno, B. Suzek, Z. Weng, and S. Kasif. A study of remote homology detection. Technical report, Cambridge Research Laboratory, June 2001. http://www.hpl.hp.com/techreports/Compaq-DEC/CRL-2001-5.html.

5. Ben-Hur, A., Horn, D., Siegelmann, H. T., and Vapnik, V. 2002. Support vector clustering. J. Mach. Learn. Res. 2 (Mar. 2002), 125-137.

6. Bing-Yu Sun; De-Shuang Huang, "Support vector clustering for multiclass classification problems," Evolutionary Computation, 2003. CEC '03. The 2003 Congress on , vol.2, no., pp. 1480-1485 Vol.2, 8-12 Dec. 2003.

7. Byvatov, E. and Schneider, G. (2003). Support vector machine applications in bioinformatics. Appl Bioinformatics, 2(2):67-77.

8. Byvatov, E., Fechner, U., Sadowski, J., and Schneider, G.. Comparison of Support Vector Machine and Artificial Neural Network Systems for Drug/Nondrug Classification. Journal of Chemical Information and Computer Sciences 2003 43 (6), 1882-1889.

9. C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for SVM protein classification. Proceedings of the Pacific Symposium on Biocomputing, 2002.

10. C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch string kernels for SVM protein classification. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, Advances in Neural Information Processing Systems. MIT Press, 2003b.

11. Cheng, C. W., Su, E., Hwang, J. K., Sung, T. Y., and Hsu, W. L. (2008). Predicting rna-binding sites of proteins using support vector machines and evolutionary information. BMC Bioinformatics, 9(Suppl 12).

12. Collobert, R., Bengio, S., and Bengio, Y. 2002. A parallel mixture of SVMs for very large scale problems. Neural Comput. 14, 5 (May. 2002), 1105-1114.

13. Damoulas, T. and Girolami, M. A. 2008. Probabilistic multi-class multi-kernel learning. Bioinformatics 24, 10 (May. 2008), 1264-1270.

14. Ding, C. and Dubchak, I. Multi-class protein fold recognition using support vector machines and neural networks. Bioinformatics 2001, 17: 349–358.

15. E J. Moler, M. L. Chow, and I. S. Mian. Analysis of molecular profile data using generative and discriminative methods. Physiol Genomics, 4:109126, 2000.

16. G.B. Huang, K.Z. Mao, C.K. Siew, D.-S. Huang, Fast modular network implementation for support vector machines, IEEE Trans. Neural Networks, 2006.

17. Gao S., Zhang N., Duan GY, Yang Z., Ruan JS, Zhang T. Prediction of function changes associated with single-point protein mutations using support vector machines (SVMs). Human Mutation, 2009. To appear.

18. Giannakeas, Nikolaos, Karvelis, Petros S., and Fotiadis, Dimitrios I. A classification-based segmentation of cDNA microarray images using Support Vector machines. Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE 20-25 Aug. 2008 Page(s):875 - 878.

19. Jia, J., Liang, Y., Zi-zhang, Z. EHPred: an SVM-based method for epoxide hydrolases recognition and classification. J Zhejiang Univ SCIENCE B, 2006, 7(1):1-6.

20. Jian-Ding Qiu, San-Hua Luo, Jian-Hua Huang, Ru-Ping Liang. Using support vector machines to distinguish enzymes: Approached by incorporating wavelet transform, Journal of Theoretical Biology, Volume 256, Issue 4, 21 February 2009, Pages 625-631.

21. Jianmin Ma, Minh N. Nguyen, Jagath C. Rajapakse, ”Gene Classification Using Codon Usage and Support Vector Machines,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 6, no. 1, pp. 134-143, January-March, 2009.

22. K. Kramer, L. Hall, D. Goldgof, Fast Support Vector Machines for Continuous Data.” IEEE Transactions on Systems, Man and Cybernetics, 2009. To appear.

23. Krause, L., McHardy, A. C., Nattkemper, T. W., Phler, A., Stoye, J., and Meyer, F. (2007). Gismo-gene identification using a support vector machine for orf classification. Nucleic Acids Res, 35(2):540-549.

24. Leslie, C. and Kuang, R. 2004. Fast String Kernels using Inexact Matching for Protein Sequences. J. Mach. Learn. Res. 5 (Dec. 2004), 1435-1455.

25. Liao, L. and Noble, W. S. 2002. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In Proceedings of the Sixth Annual international Conference on Computational Biology (Washington, DC, USA, April 18 - 21, 2002). G. Myers, S. Hannenhalli, D. Sankoff, S.

Istrail, P. Pevzner, and M. Waterman, Eds. RECOMB '02. ACM, New York, NY, 225-232.

26. M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. S. Furey, Jr. M. Ares, and D. Haussler. Knowledge-based analysis of microarray gene expression data using support vector machines. Proceedings of the National Academy of Sciences of the United States of America, 97(1):262267, 2000.

27. Melvin I, Ie E, Kuang R, Weston J, Noble WS, Leslie C. SVM-fold: a tool for discriminative multi-class protein fold and superfamily recognition. BMC Bioinformatics 2007, 8(Suppl 4):S2.

28. Myasnikova, E., Samsonova, A., Samsonova, M. and Reinitz, J., Support vector regression applied to the determination of the developmental age of a Drosophila embryo from its segmentation gene expression patterns. Bioinformatics. v18 iSuppl. 1. S87-S95.

29. Nassif, H. Al-Ali, H. Khuri, S. Walid, K. Prediction of protein-glucose binding sites using support vector machines. Proteins: Structure, Function, and Bioinformatics, 2009. To appear.

30. Ng, Andrew. "CS 229 Lecture Notes 3." http://www.stanford.edu/class/cs229/notes/

31. Noble, William. "Support Vector Machine Applications in Computational Biology." In B. Schoelkopf, K. Tsuda and J.-P. Vert, ed., Kernel Methods in Computational Biology. MIT Press, 2004. pp. 71-92.

32. Nugent, T. and Jones, D. (2009). Transmembrane protein topology prediction using support vector machines. BMC Bioinformatics, 10(1):159+.

33. P. Pavlidis, J. Weston, J. Cai, and W. N. Grundy. Gene functional classification from heterogeneous data. In Proceedings of the Fifth International Conference on Computational Molecular Biology, pages 242248, 2001b.

34. P. Pavlidis, T. S. Furey, M. Liberto, and W. N. Grundy. Promoter region-based classification of genes. Proceedings of the Pacific Symposium on Biocomputing, pages 151163, 2001a.

35. Park, K. and Kanehisa, M., Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. Bioinformatics. v19. 1656-1663.

36. R. Burbidge, M. Trotter, B. Buxton, S. Holden. Drug design by machine learning: support vector machines for pharmaceutical data analysis. Computers & Chemistry, Volume 26, Issue 1, December 2001, Pages 5-14.

37. Reczko, M. and Hatzigerrorgiou, A., Prediction of the subcellular localization of eukaryotic proteins using sequence signals and composition. Proteomics. v4. 1591-1596.

38. Rucker, H., Burgers, CJC, Kaufman, L., Smola, AJ. Support Vector Regression Machines, NIPS, 1996.

39. Rui Kuang; Ie, E.; Ke Wang; Kai Wang; Siddiqi, M.; Freund, Y.; Leslie, C., "Profile-based string kernels for remote homology detection and motif extraction," Computational Systems Bioinformatics Conference, 2004. CSB 2004. Proceedings. 2004 IEEE , vol., no., pp. 152-160, 16-19 Aug. 2004.

40. S. Hua and Z. Sun. A novel method of protein secondary structure prediction with high segment overlap measure: support vector machine approach. Journal of Molecular Biology, 308:397407, 2001a.

41. S. Knudsen. A Biologists Guide to Analysis of DNA Microarray Data. Wiley, New York, 2002.

42. S. Ramaswamy, P. Tamayo, R. Rifkin, S. Mukherjee, C. H. Yeang, M. Angelo, C. Ladd, M. Reich, E. Latulippe, J. P. Mesirov, T. Poggio, W. Gerald, M. Loda, E. S. Lander, and T. R. Golub. Multiclass cancer diagnosis using tumor gene expression signatures. Proceedings of the National Academy of Sciences of the United States of America, 98(26):1514954, 2001.

43. Saigo, H., Vert, J., Ueda, N., and Akutsu, T. 2004. Protein homology detection using string alignment kernels. Bioinformatics 20, 11 (Jul. 2004), 1682-1689.

44. Statnikov A, Wang L, Aliferis CF. A Comprehensive Comparison of Random Forests and Support Vector Machines for Microarray-Based Cancer Classification. BMC Bioinformatics, 2008; 9:319.

45. T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. Journal of Computational Biology, 7(1-2): 95114, 2000.

46. T. Jaakkola, M. Diekhans, and D. Haussler. Using the Fisher kernel method to detect remote protein homologies. In Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology, pages 149158, Menlo Park, CA, 1999. AAAI Press.

47. Warmuth, M.K., Liao, J., Rtsch, G., Mathieson, M., Putta, S. and Lemmen, C., Active learning with support vector machines in the drug discovery process. J. Chem. Inform. Comput. Sci. v43 i2. 667-673.

48. Weston, J., Leslie, C., Ie, E., Zhou, D., Elisseeff, A., and Noble, W. S. 2005. Semi-supervised protein classification using cluster kernels. Bioinformatics 21, 15 (Aug. 2005), 3241-3247.

49. Y.-D. Cai, X.-J. Liu, X.-B. Xu, and G.-P. Zhou, "Support vector machines for predicting protein structural class," BMC Bioinformatics, vol. 2, article 3, pp. 1-5, 2001.

50. Zhang, S., Pan, Q., Zhang, H., Zhang, Y. and Wang, H., Classification of protein quaternary structure with support vector machine. Bioinformatics. v18. 2390-2396.

51. Zheng Rong Yang. Biological applications of support vector machines. Brief Bioinform 5: 328-338.

52. Zhu, Y., Shen, X., Pan, W. Network-based support vector machine for classification of microarray samples. BMC Bioinformatics 2009, 10(Suppl 1):S21doi:10.1186/1471-2105-10-S1-S21