Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Theoretical Biology 234 (2005) 351–361

Journal of Theoretical Biology

www.elsevier.com/locate/yjtbi

Support vector machines for learning to identify the critical positions of a protein

Anshul Dubey, Matthew J. Realff*, Jay H. Lee, Andreas S. Bommarius

School of Chemical & Biomolecular Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0100, USA

Accepted 23 November 2004
Available online 22 January 2005

Abstract

A method for identifying the positions in the amino acid sequence, which are critical for the catalytic activity of a protein using support vector machines (SVMs) is introduced and analysed. SVMs are supported by an efficient learning algorithm and can utilize some prior knowledge about the structure of the problem. The amino acid sequences of the variants of a protein, created by inducing mutations, along with their fitness are required as input data by the method to predict its critical positions. To investigate the performance of this algorithm, variants of the β-lactamase enzyme were created in silico using simulations of both mutagenesis and recombination protocols. Results from literature on β-lactamase were used to test the accuracy of this method. It was also compared with the results from a simple search algorithm. The algorithm was also shown to be able to predict critical positions that can tolerate two different amino acids and retain function.
© 2005 Elsevier Ltd. All rights reserved.

Keywords: Support vector machines; Identifying critical positions; Machine learning; Amino-acid sequence; β-Lactamase; Directed evolution

1. Introduction

The map of the protein's catalytic activity or "fitness" to its sequence is one of the most significant and well-studied problems to which a satisfactory solution has not been found. This reflects the complexity of the intermediate mappings from the sequence to secondary structure and the tertiary structure and other factors such as the dynamics of conformation changes. Progress on this prediction problem will enable us to tailor these proteins by systematically substituting one amino acid for another to have a higher activity for a reaction of interest or to alter the environments of their optimum activity. Directed evolution (Chen and Arnold, 1993; Stemmer, 1994) where repeated mutations are made to the protein sequence almost randomly in the hope of obtaining the desired change in activity has emerged as a technique to improve protein performance. With the average length of a protein sequence being around 300 and the fact that each position can potentially take 19 amino acid substitutions, the number of possible mutants is astronomical. A huge fraction of this number will be inactive proteins, with no catalytic activity. However, only a very small fraction of these possible mutants can be feasibly generated and screened for activity in a laboratory, even after considerable advances in these methods (Daugherty et al., 2000; Petrounia and Arnold, 2000; Lin and Cornish, 2002; Neylon, 2004). Under these limitations, the probability of finding a superior mutant can be made much higher if we can direct our efforts to generating mutants with a better chance of being active.

It has been found that there are certain residues in a protein's amino acid sequence, that are essential for its function (Terwilliger et al., 1994; Huang et al., 1996). The active site residues, for example, are absolutely essential for catalytic activity towards a specific substrate. We can term these *critical residues* and the

*Corresponding author. Tel.: +1 404 894 1834; fax: +1 404 894 2866.
E-mail address: matthew.realff@chbe.gatech.edu (M.J. Realff).

positions where they occur in the protein sequence can be called *critical positions*. Thus, even if one such *critical position* is mutated, the variant will be inactive. The other positions either have a non-fatal effect or no effect at all on the activity. It has been shown that mutations in such position preferentially increase our chance of finding a mutant with increased activity (Voigt et al., 2001).

Presently, there is no definite or proven predictive theoretical method that can indicate the critical or important amino acid residues in a protein. The present method is to do a sequence alignment between the enzymes of the same family (Siezen et al., 1991), which gives a rough approximation of such positions by looking at the conserved residues. However, it is very likely to mislabel a residue to be critical when it can be mutated. This results in conservatively probing the sequence space around an existing variant, in almost all cases one that is known to be active. Thus, our probability of finding the mutant with the optimum activity is lowered. This motivates the possible application of a learning approach where an algorithm can be used to learn this correlation using experimental data from directed evolution experiments as examples.

Recently, algorithms have been proposed that determine the crossover regions for recombination that are like to yield an active mutant for given parent sequences. These algorithms are based on preserving secondary structure blocks (Meyer et al., 2003) or preserving residues close to each other in the tertiary structure (Saraf et al., 2004). These results are backed up by some initial experimental investigation. Owing to the requirement of knowing the secondary as well as tertiary structure of the parent proteins for recombination, these algorithms are limited in their applicability. Prediction suffix trees have been tried for classification of sequences (Largeron-Leteno, 2003) and were shown to predict biological promoter activity. Still, the residues important for this function were not predicted. Neural networks coupled with spatial clustering have also been used to predict specifically the location of the active sites in different proteins using the sequence and structure information (Gutteridge et al., 2003). The energetics of protein structure have also been used to make predictions on functionally important residues (Elcock, 2001).

Support vector machines (SVMs) (Vapnik, 1995) were introduced in the past decade and have proved to be a very successful learning technique. SVMs are derived using an empirical induction algorithm that constructs a description of a concept from positive and negative examples. They classify the data based on the similarity between the examples measured by the similarity function or *kernel*. This function can be chosen according the problem at hand and thus make the algorithm extremely flexible in handling a wide variety of problems. At the same time, one can try kernels with different structures and functions to probe which one works the best for a particular problem.

SVMs have been successfully employed in a wide range of areas such as automated face recognition (Deniz et al., 2003), and text categorization (Joachims, 1998). In the field of biochemistry, SVMs have found application in the analysis of micro-array gene expression data. They were used to functionally classify the genes by using the data from DNA micro-array hybridization experiments (Brown et al., 2000). SVMs have also been widely used in secondary structure prediction (Kim and Park, 2003; Ward et al., 2003).

The use of SVM for extracting features important for classification has recently been studied using what the authors called "perturbations" (To and Lim, 2004). This method involves inhibiting one feature at a time from the training data set and observing its effect on the margin of classification. Higher sensitivity of the margin on a feature indicates a critical feature. This method was applied in an image processing example to extract pixels of images that are important. Another method called the discriminative function pruning analysis (DFPA) has also been used in feature subset selection for SVM for the main purpose of reducing functional costs (Mao, 2004).

2. Support Vector Machines (SVMs)

SVMs are primarily applied in binary classification problems where the data belong to one of two classes. Each data point is represented in the form of a vector x and the class that it belongs to is called its *label*, represented by y. SVMs need to be trained using data points that have known labels as examples. Based on these examples or the training data, SVMs learn to classify the data points whose labels are unknown. The dimensions of the input vector, x, constitute the input space, which is decided based on the problem. The two different classes of data are termed positive and negative and the labels of the training examples are usually assigned a value of 1 or -1, respectively. As an example, if we want to functionally classify the genes based on gene expression data in different conditions, the input vector for each gene can be the expression ratios obtained from the micro-array (Brown et al., 2000) under those conditions. To train the SVM, the micro-array data of genes that belong to a particular class (label $y = 1$) and ones that do not (label $y = -1$), is required. The trained SVM can then use the micro-array data for an unknown gene in similar environments and predict whether it belongs to that class or not.

The theory behind SVMs is based on a transformation of data from the input space to a *feature space* (Christianini and Shawe-Taylor, 2000). The feature space usually has a larger number of dimensions compared the original input space and should be able to *linearly* separate the data (Fig. 1). Once the transformation has been selected based on prior knowledge, a linear classification algorithm is used to find the hyperplane in the feature space that is going to classify the data as shown in Fig. 1. The dual form of the perceptron algorithm (Christianini and Shawe-Taylor, 2000), which uses only the inner product of the input vectors to find and represent the equation of the hyperplane, is used for learning the linear classification in SVMs. Since, the inner product of any two vectors is always a scalar, it implies that the solution is independent of the dimension of the problem. This aspect is critical for the application of SVMs because it implies that we do not lose accuracy by increasing the dimensions while transforming our data to the *feature space*.

If Φ is the transformation from the input space to the feature space, then for any vector x in the input space the corresponding vector in the feature space is $\Phi(x)$. To find the equation of the hyperplane, the linear perceptron formulates a minimization problem. Using the Lagrangian theorem, this problem can be converted into its following dual form (Christianini and Shawe-Taylor, 2000):

$$\max W(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2}\sum_{i=1}^{\ell}\sum_{j=1}^{\ell} \alpha_i\alpha_j y_i y_j K(\vec{x}_i, \vec{x}_j), \tag{1}$$

$$\text{st} \quad \sum_{i=1}^{\ell} \alpha_i y_i = 0$$
$$0 \leqslant \alpha_i \leqslant C, i = 1, \ldots, \ell, \tag{2}$$

where α_i are the Lagrangian multipliers, ℓ is the size of the training set with each point as a vector, $\vec{x}_i = [x_i^1, x_i^2, \ldots, x_i^m]^{\mathrm{T}}$ for an m dimensional input space, and $y_i \in \{1, -1\}$ as its fitness. The value of C can be varied to give a maximal margin or a soft margin classification (Christianini and Shawe-Taylor, 2000). Having a soft margin can enable us to take into account errors in the training data. This is a quadratic programming problem for which efficient solvers exist (Nocedal and Wright, 1999). The kernel, $K(\vec{x}_i, \vec{x}_2)$, is defined as the inner product of two vectors in the feature space:

$$K(\vec{x}_1, \vec{x}_2) = \langle \Phi(\vec{x}_1)\Phi(\vec{x}_2) \rangle. \tag{3}$$

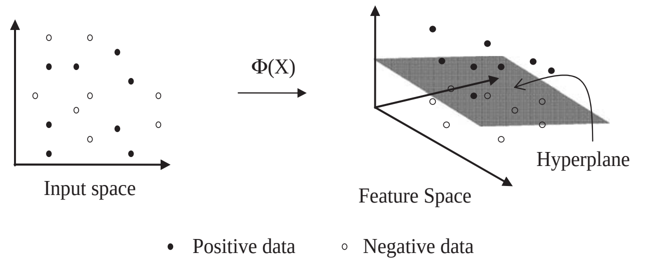

Fig. 1. An illustration of the transformation from the input space to the feature space.

Once the optimum values of the Lagrangian multipliers, α_i are obtained from Eq. (1), the classifying function is written as

$$h(\vec{x}) = \operatorname{sgn}\left(\sum_i \alpha_i y_i K(\vec{x}, \vec{x}_i) + b\right). \tag{4}$$

Thus, if $h(x)$ is positive, the vector x with an unknown label belongs to the positive class and vice versa.

To evaluate the kernel as defined in Eq. (3), the transformation and subsequently the feature space, has to be defined. It is usually not feasible to obtain a feature space due to the lack of understanding of the structure of the problem. However, since only the inner product is used in the formulation instead of the *feature vector* (a vector in the feature space) itself, it is not necessary to define a feature space explicitly. Any function that can be used as the kernel for the problem will define the feature space implicitly. The only requirement for that function is to have the same properties as an inner product like the Cauchy–Schwartz inequality and symmetry. Most importantly, the kernel matrix, as defined in Eq. (5), should satisfy the Mercer's theorem (Christianini and Shawe-Taylor, 2000). Many common functions can be used as a kernel and have been shown to work well with different kinds of problems such as the polynomial kernel, radial basis function (RBF) kernel or the Gaussian kernel (Christianini and Shawe-Taylor, 2000). Note that if the feature space is explicitly defined and the kernel function follows from it, it will satisfy all the required properties:

$$K = \begin{bmatrix} K(\vec{x}_1, \vec{x}_1) & K(\vec{x}_1, \vec{x}_2) & \cdots & K(\vec{x}_1, \vec{x}_\ell) \\ K(\vec{x}_2, \vec{x}_1) & K(\vec{x}_2, \vec{x}_2) & \cdots & K(\vec{x}_2, \vec{x}_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ K(\vec{x}_\ell, \vec{x}_1) & K(\vec{x}_\ell, \vec{x}_2) & \cdots & K(\vec{x}_\ell, \vec{x}_\ell) \end{bmatrix}. \tag{5}$$

The ability to use any suitable function as a kernel without having to define the elaborate feature space enables the application of SVMs to a wide range of problems. Using a kernel function instead of evaluating the inner product of vectors also saves on computation time. However, when possible, defining the feature space explicitly can make the results more accurate.

The quadratic optimization problem (Eq. (1)) was solved using the subroutine *quadprog* in Matlab as well as using the Sequential Minimal Optimization (SMO) algorithm (Platt, 1998) developed specifically for SVM. The results from both the routines were similar; however, *quadprog* was predominantly used because of the lower computation time in our case.

The main objective of this work is to obtain the positions in a protein sequence that are crucial for it to be catalytically active. For the purpose of directed evolution, these positions can be defined as the ones which, when mutated, render the protein catalytically inactive towards a particular substrate. Using a data set, SVM can learn how to determine the fitness of an unknown sequence. However, to extract the features that are important using SVMs, a novel feature selection method has been developed.

From the classification function, as shown in Eq. (4), it can be concluded that the equation of the hyperplane that separates the data in the feature space is

$$\sum_i (\alpha_i y_i K(x, x_i)) + b = 0. \tag{6}$$

For a Euclidian space with finite dimension N, the kernel can be written as

$$K(x_1, x_2) = \sum_{i=1,N} \varphi^i(x_1)\varphi^i(x_2), \tag{7}$$

where $\varphi^i(x)$ is the ith element of the feature vector $\Phi(x)$, or the features, corresponding to x. So, Eq. (6) can now be written in terms of the features as

$$\sum_{j=1,N} \left(\varphi^j(x) \left(\sum_i \alpha_i y_i \varphi^j(x_i) \right) \right) + b = 0 \tag{8}$$

$$\Rightarrow \sum_{j=1,N} (\varphi^j(x) a^j) + b = 0. \tag{9}$$

Thus, the slope of the hyperplane with respect to a feature, in the feature space, is derived to be

$$\frac{\partial\left(\sum_{j=1,N}(\varphi^j(x)a^j) + b\right)}{\partial \varphi^j(x)} = a^j. \tag{10}$$

Note that the term $a^j = \sum_i \alpha_i y_i \varphi^j(x_i)$ is independent of the variable x and is constant for a specific training set. This implies that if the slope, a^j is zero for a particular feature, then the equation of the hyperplane and consequently, the classification function in Eq. (4) is not influenced by that feature. Any feature with zero a^j can be concluded to be an unimportant feature for classification and vice versa. To be able to calculate this slope for feature selection, the feature space needs to be explicitly defined. Once the critical features are found in the feature space, the knowledge of the mapping can be used to determine the significant dimensions of the input space, which, in our problem, are the critical positions of the protein sequence.

3. Motivation

Directed evolution experiments involve generating variants by either introducing random mutations in the gene or by recombining parts of the gene coding for a protein. These variants are then subjected either to a selection process testing, for instance, the ability to allow a cell to grow or to a screening process based on an assay indicating the desired trait such as improved activity. Experiments often have to go through many rounds of mutations and screening before we can find, if at all, a variant with the desired properties. Our objective is to make this process more efficient so that successful mutants can be found with a higher probability and with reduced experimental effort.

The obvious room for improvement in a directed evolution process lies in reducing the randomness involved in changing the gene sequence. The ability to point out the exact positions in the sequence that should be mutated to improve the protein requires a complete understanding of the complex protein sequence to function mapping. However, if the positions that should not be mutated to preserve activity are determined, significant improvements can be observed by conserving them. This will increase the number of active variants we generate and thus reduce the experimental effort that is spent on inactive or unstable variants. To illustrate this fact, let us assume a protein sequence of 300 amino acids, of which 50 are *critical residues*. Now, if we make 6 mutations to this sequence, we can calculate the probability of generating an active mutant when these mutations are completely random as opposed to when 50% of the critical positions are known and thus can be avoided:

$$P_{rand} = \frac{{}^{250}C_6}{{}^{300}C_6} = 0.331, \quad P_{50\%} = \frac{{}^{250}C_6}{{}^{275}C_6} = 0.561.$$

Thus, a significant improvement can be observed by avoiding just half of the critical positions. The sequences of the variants obtained in the first round of mutations can be used to locate a fraction of the critical positions and improve the subsequent rounds of evolution. The result can be progressively improved by using the variants created in the subsequent rounds as we find more critical positions.

The ability to bias mutations or crossovers in the directed evolution protocols has been explored by several researchers. Mathematical models are being developed for recombination protocols such as DNA shuffling (Moore and Maranas, 2000; Moore et al., 2000; Maheshri and Schaffer, 2003). These models can enable us to direct mutations by manipulating certain experimental conditions such as annealing temperatures of gene fragments.

4. Formulation

The first step in formulating the SVM learning problem is to determine the input space or the input vector. Since it is known that the protein activity depends directly on its amino acid sequence, this can be used as the input vector. This implies that each dimension of the *input space* corresponds to a position in the amino acid sequence. So the protein sequences of the variants generated during directed evolution can be used as input vectors. Additionally, the labels or the fitness of each variant required for training can be obtained from experimental results or more specifically by screening them for activity.

The 20 naturally occurring amino acids in a protein were numbered from 1 to 20 after arranging them according to their chemical classification as neutral, polar or charged (Voet and Voet, 1995). This ensures that a drastic mutation and a conservative mutation are distinguished by a higher and lower change in magnitude, respectively.

The second step is to define a feature space. The feature space should be chosen such that the positive and the negative examples can be linearly separated. Consider a position that renders a protein catalytically inactive or unfit, when mutated. In the wild type sequence, if that position contains the amino acid represented by number 13, which is mutated to amino acid number 5 in a variant, the fitness of the variant is -1. Now, if the same position is mutated to amino acid number 17 for a variant, the fitness is again -1. So, for that particular dimension, any value above *or below* 13 is unfit. This indicates that in the input space the positive and negative examples are not linearly separable.

The feature space dimensions should represent a specific amino acid in a specific position. So, for each position in the amino acid sequence, we assign 20 dimensions corresponding to the 20 amino acids, thus, making the dimensions of the feature space 20 times the length of the sequence. Note that for SVMs, the increased dimensions of the feature space do no affect the computational accuracy. Each sequence can be represented as a unique vector in this space. Dimensions are assigned sequentially in the feature space, so dimensions $20 \times (n-1)+1$ to $20 \times n$ are assigned to the nth position of the amino acid sequence. For example, if the amino acid represented by number 10 appears in position number 50 in the original amino acid sequence, then out of the dimensions 1001–1020 in this space, which are assigned to position 50, we assign a value m to the dimension 1010. The other dimensions between 1001 and 1020 are assigned a value 0. Now if a position in the original amino acid sequence is critical and thus accepts only a particular amino acid, then all the positive sequences will have the value m for this dimension and the negative sequences will have either m or 0 for the same dimension. Thus, they are linearly separable at the value m.

Apart from these *critical positions* where any mutation is deleterious, positions exist that can accept only two different amino acids, which we should be able to predict. These positions can be called *tolerable positions* and can be considered equally important while trying to bias a DE protocol. Since each dimension in the above-mentioned space is specific to a particular amino acid in a particular position, a data set with such positions is also linearly separable. So, this space satisfies the criteria that we were looking for in a feature space.

The degree of freedom in this formulation is the value of m. The obvious choice for this value is 1, which will ensure that each sequence will have a unique vector in the feature space. However, this makes the kernel matrix degenerate and making it difficult to solve the optimization problem for learning. To demonstrate this, consider two sequences, x_1 & x_2. If these sequences are exactly the same, then their projection in the feature space is for example

$$x_1 = [0 \quad 0 \quad 1 \quad \dots \quad 0 \quad 1 \quad 0 \quad \dots \quad 1 \quad 0]^{\mathrm{T}}, \tag{11}$$

$$x_2 = [0 \quad 0 \quad 1 \quad \dots \quad 0 \quad 1 \quad 0 \quad \dots \quad 1 \quad 0]^{\mathrm{T}}. \tag{12}$$

So, their inner product, which is, $\sum x_1^i x_2^i$ will simply be equal to N, which is the length of the sequence. However, if they differ in only one position, their projection in the feature space is, for example,

$$x_1 = [0 \quad 0 \quad 1 \quad \dots \quad 1 \quad 0 \quad 0 \quad \dots \quad 1 \quad 0]^{\mathrm{T}}, \tag{13}$$

$$x_2 = [0 \quad 0 \quad 1 \quad \dots \quad 0 \quad 0 \quad 1 \quad \dots \quad 1 \quad 0]^{\mathrm{T}}. \tag{14}$$

This is because each dimension in the feature space corresponds to one particular amino acid in a particular position. Now, their inner product will be equal to $N-1$, irrespective of the amino acid and the position that they differ in. Thus, the kernel functions of different variants will be similar making the kernel matrix degenerate. To solve this problem, a value, which has a specificity of the position as well as amino acid should be assigned. The m value that was found to work best is given by

$$m(x,n) = \sqrt{x(21.5-x)\left(1+\frac{n}{100}\right)}, \tag{15}$$

where x is the amino acid in the position n. The term $x(21.5-x)$ is used because it does not differ significantly in magnitude for the different amino acids, thus, keeping the results relatively unbiased to the number representation of the amino acids. So, if an amino acid represented by x occurs in a position n of a particular sequence, then the dimension $20 \times (n-1) + x$ of the vector corresponding to that sequence will have the value $m(x,n)$. The kernel function, which is the inner product of any two vectors in the feature space can be written in terms of the original sequences as

$$K(X,Y) = \sum_{i=1,L} \phi(x_i, y_i), \tag{16}$$

where

$$\phi(x_i, y_i) = \begin{cases} x(21.5-x)\left(1+\frac{n}{100}\right) & \text{if } x_i = y_i, \\ 0 & \text{if } x_i \neq y_i. \end{cases} \tag{17}$$

where x_i and y_i denote the amino acid in the ith position of the original protein sequences X and Y.

Our main objective for this formulation is to extract information on the positions that are critical for a protein's function. The slopes of the resulting hyperplane for classification with respect to the features are calculated and mapped back to the sequence positions. However, since classification is not our primary objective, we have to verify if there exist techniques, involving much less computation effort that can give the same results when applied to the pool of examples we are using for SVM. One simple but effective approach is to have an algorithm that searches through the different sequences and eliminates positions based on certain set rules about their mutations in the positive and negative examples. We term this algorithm the *Greedy-And* algorithm as it aims to include all the positions that feature even the smallest likelihood of being important.

The logic used for *Greedy-And* is to identify the mutations in each position in the positive and negative examples. If a particular position is not mutated in the positive sequences but is mutated in the negative ones, it is taken as a critical position. Apart from these, if a position is mutated to only one particular amino acid in the positive sequences and is mutated to some other amino acid in the negative sequences, it is assumed to be *tolerable* to one amino acid substitution and is also included as a *tolerable position*. The shortcomings of such an algorithm can be easily seen. It will mislabel a lot of *non-critical* positions as critical. This occurs because a non-critical position, which has no mutations among the positive sequences, but happens to be mutated along with a critical position, will show up mutated among the negative sequences and be labelled as critical. Similarly *non-critical* positions can also appear as tolerable positions.

5. Case evaluated

Extensive studies have been done on the TEM-1 β-lactamase enzyme (Palzkill and Botstein, 1992; Raquet et al., 1995; Cantu et al., 1997; Axe, 2000), which catalyses the undesired hydrolysis of β-lactam ring in the β-lactam antibiotics such as penicillins and cephalosporins. Bacterial resistance against β-lactam antibiotics is conferred by the action of β-lactamases. The TEM-1 β-lactamase sequence consists of 286 amino acid residues. A recent experimental effort was aimed at determining the amino acid residues that are accepted at each position of the enzyme sequence (Huang et al., 1996). The *bla*$_{\mathrm{TEM-1}}$ gene was mutated exhaustively by random replacement mutagenesis and insertion mutants (three codons at a time) and such that each position was individually mutated to the 19 non-wild-type amino acid residues positioning the variants. The variants were

Table 1
The residues in β-lactamase that either do not tolerate a substitution or can be replaced by just one other residue

Leu28	Ala132	Trp227	Trp286
Tyr44	Arg162	Ala230	Ala,Glu35
Phe64	Glu164	Asp231	Gly,Ala52
Pro65	Asp174	Lys232	Val,Leu72
Ser68	Asp177	Ser233	Tyr,His103
Thr69	Thr178	Gly234	Ala,Thr133
Lys71	Thr179	Gly239	Pro,Ala181
Leu74	Trp208	Arg241	Leu,Val223
Pro105	Asp212	Gly242	Ala,Gly235
Leu120	Val214	Leu247	Tyr,Phe260
Ala123	Ala215	Gly248	Ile,Val275
Ser128	Leu218	Met268	
Asp129	Arg220	Asn272	
Asn130	Pro224	Ile278	

subjected to a selection scheme probing for the ability to confer ampicillin resistance to *Escherichia coli*. The successful mutants were then gene-sequenced to observe tolerated amino acid substitutions. The results showed that 43 positions in the sequence do not tolerate any kind of substitutions while some other positions were found to accept two different amino acids, and so on (Table 1). Note that though the results of this study are comprehensive, the strategy deals with only one substitution at a time and does not investigate the effect of simultaneous substitutions in different positions.

These findings provide us with the necessary information to test the effectiveness of the SVM approach. Although we expect to obtain the same results as found by Huang et al. we need to show that the effort or the amount of data required could be reduced and that intermediate results with fewer examples can provide focusing for directed evolution. Instead of producing variants experimentally, we use simulations to create them in silico. To obtain the fitness of these variants, which are required for training, we use the critical positions found by Huang et al. We look for mutations in the critical positions, among the variant sequences. If one or more critical positions have been mutated to an amino acid not accepted there, the variant is labeled as negative and vice versa. The critical positions that are found by using the SVM trained on this data are then compared to the critical positions with which we started.

The in silico variants of the parent gene sequence of β-lactamase, the *bla*$_{\mathrm{TEM-1}}$ gene were obtained by simulating one of the directed evolution protocols used in the laboratory. The simplest such protocol is random mutagenesis (Moore and Arnold, 1996; Miyazaki et al., 2000), a procedure in which nucleotides in random positions in the gene are mutated to any of the other three possible nucleotides among the possible set of A, T, C, and G. We assume all mutations to occur with equal probability, thus ignoring any possible bias in the susceptibility of A, T, C, and G towards mutation as well as bias in the protocols themselves. A total of eight mutations (Zaccolo and Gherardi, 1999) are made in the parent gene sequence to generate each example in the data set. As many single mutations are silent, such a procedure generates less than eight changes in resulting amino acid sequence of the variant protein. Only full-length amino acid sequences are considered for analysis and lower length sequences arising due to mutations to stop codons are neglected.

Random mutagenesis is easy to simulate and allows us to explore variation of amino acid sequence without inherent bias of mutagenesis protocols. However, it is not possible to use the knowledge that we can gain from our strategy to bias the mutations away from the critical positions using random mutagenesis. The other most popular technique is recombination, as embodied by the DNA shuffling protocol (Stemmer, 1994). Recently developed models for this procedure, potentially allow us to manipulate experimental conditions to direct crossovers (equivalent to mutations in recombination protocols) (Moore and Maranas, 2000; Moore et al., 2000; Maheshri and Schaffer, 2003; Ostermeier, 2003). To demonstrate the utility of the SVM, we created a data set by simulating a shuffling procedure using a very crude model. We started with four parent sequences, which we obtained from mutagenesis. For generating each mutant, we considered random number of crossovers up to a maximum of 4 (Joern et al., 2002). Any two crossovers had a minimum of 50 nucleotide bases between them. Since, out-of-frame alignment was not considered in shuffling for the data set, only the critical positions that had mutation in the parent sequences will change in the mutant library Table 1.

6. Results

Different training data sets were generated, each with a different number of sequences. Sequence fitness was approximated using the results from Huang et al. as described earlier. The SVM was trained with each data set and the slope of the hyperplane with respect of the features was evaluated. We define a parameter, $\boldsymbol{d}_i$ which is the ratio of slope of the hyperplane with respect to a certain feature i and the maximum slope among all the features. A ***Cutoff*** value can be fixed, which $\boldsymbol{d}_i$ must exceed to consider the ith position critical. The critical features are then mapped from the feature space back to the input space to obtain the critical residues in the protein sequence. Once this set of critical positions is obtained, it is compared with the critical positions determined by Huang et al. which were used to predict the fitness of the in silico generated sequences. This provides us the positions that the algorithm was able to identify correctly as being critical and the ones that were missed.

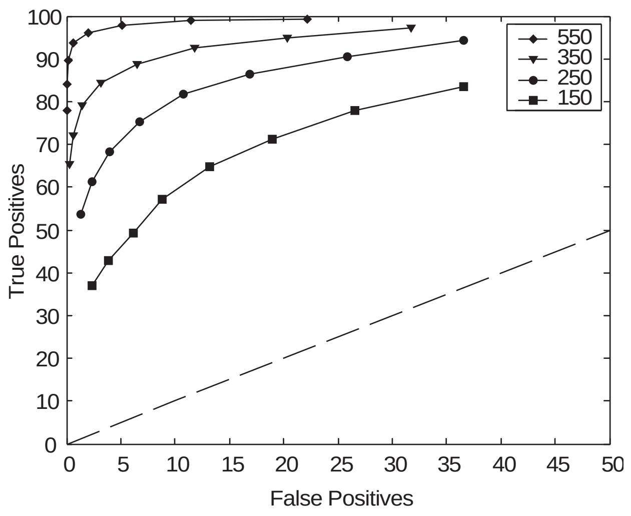

Fig. 2. ROC Analysis for data sets with different number of variant sequences, as indicated in the legend. The numbers are averaged over 30 data sets of each size generated by random mutagenesis. The broken line represents the 45° line.

To analyse the results, we define two parameters, *True Positives* and *False Positives*, as

$$\text{True Positives} = \frac{\text{Number of critical positions identified correctly}}{\text{Total number of critical positions}} \times 100 \tag{18}$$

$$\text{False Positives} = \frac{\text{Number of non-critical positions identified as correctly critical}}{\text{Total number of non-critical positions}} \times 100. \tag{19}$$

By varying the value of the ***Cutoff***, different values of *True Positives* and the corresponding *False Positives* can be obtained for each data set. These results are plotted in Fig. 2 for data sets of different sizes as shown by the legend. The results were averaged over 30 randomly generated training sets of each size. Each point on the curve thus represents the average *True Positives* and the *False Positives* for a particular value of ***Cutoff***. Such plots are called the receiver operating characteristic (ROC) graphs (Swets and Pickett, 1982; Metz, 1986), which have long been used in signal detection theory and radiology to depict trade-offs between hit rate (Sensitivity or *True Positives*) and false alarm rate (1–Specificity or 1–*False Positives*). For a good classifier, these curves are expected to be convex in shape, featuring high sensitivity along with a high specificity. As can be seen from the figure, the ROC curve for even a small-size data set is highly convex (the broken line is at 45° to the axes). It can also be observed that the accuracy of identification varies greatly with the number of sequences in the data set as well as the ***Cutoff***.

The inaccuracies observed in the identification can be explained by the decreasing probability of mutations in a given position as the number of sequences in the training data decreases. For the algorithm to identify any position as critical, it is imperative that the particular position is mutated frequently so that the effect of changing that position can be seen. In very small sized data sets, many positions exist that are not mutated at all and thus are ignored by the algorithm. At the same time, there is a high probability that certain positions, though non-critical, are always mutated along with some other critical position, which can account for the increasing false identification. As observed, data sets with more than 500 sequences give almost 100% accuracy because all the positions are mutated with sufficient frequency to enable us to capture their effect.

When it is infeasible to obtain large number of sequences for training, a fraction of the critical positions can still be determined. The improvement that can be made by avoiding mutations even in a small fraction of *critical positions* has already been shown. Data from the subsequent rounds can be then used to determine the remaining critical positions. While using this strategy, false positives should be kept at the minimum. If a position is wrongly labeled to be a critical position and is not mutated in the subsequent rounds, the error will not be realized. Thus, a higher value of cutoff resulting in smaller number of identified positions should be used in order to keep the false positives low. For instance, as shown by the results, for a small data set of size 150, the total number of predicted critical positions should be lower than what we can accept for a data set of size 350.

In the above results, only the *critical positions* that do not tolerate any mutations are considered and the fitness of the sequences is approximated, but as mentioned earlier, our aim is also to be able to identify the positions that can accept up to two amino acids. These positions have also been determined by the work of Huang et al. along the amino acids that are accepted in them. We selected 10 such positions and incorporated them into the fitness evaluation, which implies that to evaluate the fitness of the generated sequences we also check these positions for the accepted amino acids in addition to the original 43 critical positions. Table 2 shows the results from learning using one such data set. The results shown are again averaged over 30 separate sets, each randomly generated and of the same size. As shown, we are able to predict these positions with accuracy comparable to the critical positions. Though, not shown, but the identification of these positions was consistent even with data of smaller size.

A peculiar situation arises because of the possibility of errors in the training data. Since the fitness of a sequence is found experimentally using an assay with a certain level of inaccuracy, we should make sure that this does not disrupt our results drastically. By restricting the bound on the Langrangian multipliers α (Eq. (2)), SVMs incorporate a soft margin classification which is able to ignore points that tend to skew the classification margin, typically caused by errors in the training data. To test this, we created a data set and induced an error of 10% in the fitness approximation. This implies that the fitness of 10% of the sequences in the data, which were randomly picked, was deliberately reversed by switching their signs. Table 3 presents the results obtained, which are again averaged over 30 different sets generated randomly. The standard deviation of the data was of the same order of magnitude as shown in the previous data. As seen from the results the identification from error-prone data still retains high degree of accuracy.

As suggested earlier, the *Greedy-And* algorithm can also be used to extract critical positions from the data

Table 3
The results obtained with and without errors in the fitness estimation of the variant sequences

Cutoff	Without error in fitness		With 10% error in fitness	
	True Positives	False Positives	True Positives	False Positives
0.2	90.5	15.0	83.1	22.3
0.3	81.1	5.0	72.1	11.0
0.4	67.8	1.4	59.6	4.7

The results are averaged over 30 input sets generated by random mutagenesis each with 300 variant sequences.

Table 2
The average results obtained for 30 independently generated input data sets each having 450 variant sequences

Cutoff	Predicted		True Positives		False Positives		Tolerable missed	
	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.	Mean	St. dev.
0.15	107.8	13.1	97.5	2.2	23.9	5.4	0.8	0.9
0.25	67.0	8.7	91.9	3.6	7.8	3.2	2.1	1.3
0.35	48.0	6.2	80.8	5.9	2.2	1.7	3.9	1.5
0.45	36.3	4.4	66.6	6.7	0.4	0.5	5.8	1.4

The critical positions include 10 positions that can tolerate two different amino acids. The variants were generated in silico using random mutagenesis.

Table 4
Results obtained from using the 3 different prediction strategies averaged over 30 input sets of different sizes generated by random mutagenesis

Size of training set	SVM		Greedy-And		Hybrid	
	True Positives	False Positives	True Positives	False Positives	True Positives	False Positives
400	91.1	12.7	94.7	14.8	88.7	1.6
300	81.9	14.8	95.5	19.7	85.0	3.8
200	68.8	14.5	90.8	27.3	75.2	8.4
100	47.8	14.8	73.8	29.5	55.1	12.3

Table 5
Results averaged over 30 randomly generated shuffled libraries of mutants (8 critical positions mutated) for the two different prediction strategies

Size of shuffled library	SVM			Greedy-And		
	Number predicted	True Positives	False Positives	Number predicted	True Positives	False Positives
50	20.8	77.5	7.9	54.4	100.0	19.8
100	17.6	87.8	4.5	53.5	100.0	19.4
150	13.7	94.2	2.6	52.3	100.0	18.8

set. The results obtained are compared with those obtained from SVM. It is observed that *Greedy-And* has good sensitivity since it does not miss many critical positions. It performs poorly, however, in specificity and accumulates a large number of *False Positives*. This suggests a strategy that is a combination of *Greedy-And* and SVM, which we refer to as the *hybrid* approach. *Greedy-And* can be used as an initial filter before SVM and only the positions selected by it are used for training. In other words, instead of using the whole sequence of the variants, the positions that are predicted to be non-critical by *Greedy-And* are eliminated for each variant and the shortened sequence is used to train the SVM. Since the dimensions of the input space are reduced, the accuracy of SVM increases greatly. However, when different positions are eliminated, many sequences become identical. Since these sequences do not contribute to the learning process and make the kernel matrix ill-conditioned, they are eliminated from the data set. In this example, significant improvement was observed by using the *hybrid* approach as shown in Table 4. In the results shown, the ***Cutoff*** parameter for the hybrid method as well as SVM was kept at 0.2 and the numbers are averaged over 30 different sets for each size. The standard deviation (not presented here) was again of the same order of magnitude as in the previous results. Note that the size of the training set for the *hybrid* strategy is always smaller than that for SVM and *Greedy-And* alone because of the elimination of the redundant sequences.

The advantage of the hybrid strategy can be seen from the results in the reduced number of *False Positives*. Compared to *Greedy-And* the true positives are lower for both SVM and the hybrid strategy, however, the reduction in false positives is more significant since it corresponds to a larger difference in the actual number of non-critical positions identified as critical (Eq. (19)).

As discussed earlier, we also created a mutant library using a simple shuffling simulation. The library was generated using four parents that had mutations in eight different critical positions among them, including the tolerable positions. Therefore, the maximum number of correct positions that can be predicted is eight. Hence, the *True Positives* and *False Positives* are calculated as defined earlier but based on the total number of critical positions equal to eight. The results are shown in Table 5. The data obtained from shuffling leads to a harder learning problem because the positions experience correlated changes due to crossovers between fragments. This intricacy of the problem is evident from using the Greedy-And algorithm on the data set. It fails to decouple the positions and gives an unusually large set of critical positions with many *False Positives*. However, the results obtained from SVM show a very high accuracy even for the data set as small as 100. Again, the results shown are averaged over 30 sets of libraries for each size, which were created by shuffling the same 4 parent sequences.

7. Conclusions

Support vector machines are applied for predicting critical positions required for fitness of a protein using a data set of known sequences and their fitness. The accuracy of the prediction was shown using the mutant libraries of β-lactamase created by random mutagenesis for different sizes of the data set. It was also shown that positions that are tolerable to one amino acid substitution can be predicted by the same framework. A simpler algorithm named *Greedy-And* was developed that can search through the data set and pick out positions that had a remote chance of being critical. The results obtained by this algorithm were compared with that obtained from SVM and a new strategy was developed, by combining both of these methods, which was shown to improve the performance further. SVM was also shown to give good predictions in cases where the fitness approximation of the training sequences was error-prone. Finally, a data set was created using the principle of DNA shuffling rather than random mutagenesis, which was used for training SVM and was shown to give accurate results.

Though we have demonstrated successfully the significance and the accuracy of the results we obtained, the SVM technique has some limitations. Obtaining the training data for the algorithm requires potentially resource-intensive sequencing results. However, with ever reducing costs of DNA sequencing, this option will soon become a more practical one (Meldrum, 2000a, b). A small number of sequences can provide valuable knowledge on the critical positions. The situation of interacting critical positions cannot be predicted (at least not theoretically) under the present structure of the problem. Handling of such cases should still be possible with the SVM technique but will need a different formulation.

Acknowledgements

We thank Dr. Phil Gibbs, presently the Chief Science Officer of Stheno Corporation, Atlanta, GA, for his suggestions and contributions in this work.

References

Axe, D.D., 2000. Extreme functional sensitivity to conservative amino acid changes on enzyme exteriors1. J. Mol. Biol. 301 (3), 585–595.

Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Ares, M., Haussler, D., 2000. Knowledge-based analysis of microarray gene expression data using support vector machines. Proc. Natl Acad. Sci. USA 97, 262–267.

Cantu III, C., Huang, W., Palzkill, T., 1997. Cephalosporin substrate specificity determinants of TEM-1 beta -lactamase. J. Biol. Chem. 272 (46), 29144–29150.

Chen, K., Arnold, F., 1993. Tuning the activity of an enzyme for unusual environments: sequential random mutagenesis of subtilisin E for catalysis in dimethylformamide. PNAS 90 (12), 5618–5622.

Christianini, N., Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge.

Daugherty, P.S., Chen, G., Iverson, B.L., Georgiou, G., 2000. Quantitative analysis of the effect of the mutation frequency on the affinity maturation of single chain Fv antibodies. PNAS 97 (5), 2029–2034.

Deniz, O., Castrillon, M., Hernandez, M., 2003. Face recognition using independent component analysis and support vector machines*1. Pattern Recogn. Lett. 24 (13), 2153–2157.

Elcock, A.H., 2001. Prediction of functionally important residues based solely on the computed energetics of protein structure1. J. Mol. Biol. 312 (4), 885–896.

Gutteridge, A., Bartlett, G.J., Thornton, J.M., 2003. Using a neural network and spatial clustering to predict the location of active sites in enzymes. J. Mol. Biol. 330 (4), 719–734.

Huang, W., Petrosino, J., Hirsch, M., Shenkin, P.S., Palzkill, T., 1996. Amino acid sequence determinants of *b*-lactamase structure and activity. J. Mol. Biol. 258 (4), 688–703.

Joachims, 1998. Text categorization with support vector machines. Proceedings of European Conference on Machine Learning.

Joern, J.M., Meinhold, P., Arnold, F.H., 2002. Analysis of shuffled gene libraries. J. Mol. Biol. 316 (3), 643–656.

Kim, H., Park, H., 2003. Protein secondary structure prediction based on an improved support vector machines approach. Protein Eng. 16 (8), 553–560.

Largeron-Leteno, C., 2003. Prediction suffix trees for supervised classification of sequences. Pattern Recogn. Lett. 24 (16), 3153–3164.

Lin, H., Cornish, V.W., 2002. Screening and selection methods for large-scale analysis of protein function. Angew. Chem. Int. Ed. 41 (23), 4402–4425.

Maheshri, N., Schaffer, D.V., 2003. Computational and experimental analysis of DNA shuffling. PNAS 100 (6), 3071–3076.

Mao, K.Z., 2004. Feature subset selection for support vector machines through discriminative function pruning analysis. IEEE Trans. Syst. Man Cybernet., Part B, Cybernetics 34 (1), 60–67.

Meldrum, D., 2000a. Automation for genomics, part one: preparation for sequencing. Genome Res. 10 (8), 1081–1092.

Meldrum, D., 2000b. Automation for genomics, part two: sequencers, microarrays, and future trends. Genome Res. 10 (9), 1288–1303.

Metz, C.E., 1986. ROC methodology in radiologic imaging. Invest. Radiol. 21 (9), 720–733.

Meyer, M.M., Silberg, J.J., Voigt, C.A., Endelman, J.B., Mayo, S.L., Wang, Z.-G., Arnold, F.H., 2003. Library analysis of SCHEMA-guided protein recombination. Protein Sci.: A Publ. Protein Soc. 12 (8), 1686–1693.

Miyazaki, K., Wintrode, P.L., Grayling, R.A., Rubingh, D.N., Arnold, F.H., 2000. Directed evolution study of temperature adaptation in a psychrophilic enzyme. J. Mol. Biol. 297 (4), 1015–1026.

Moore, J.C., Arnold, F.H., 1996. Directed evolution of a para-nitrobenzyl esterase for aqueous-organic solvents. Nat. Biotechnol. 14 (4), 458–467.

Moore, G.L., Maranas, C.D., 2000. Modeling DNA mutation and recombination for directed evolution experiments. J. Theor. Biol. 205 (3), 483–503.

Moore, G.L., Maranas, C.D., Gutshall, K.R., Brenchley, J.E., 2000. Modeling and optimization of DNA recombination. Comput. Chem. Eng. 24 (2–7), 693–699.

Neylon, C., 2004. Chemical and biochemical strategies for the randomization of protein encoding DNA sequences: library construction methods for directed evolution. Nucl. Acids Res. 32 (4), 1448–1459.

Nocedal, J., Wright, S.J., 1999. Numerical Optimization. Springer, New York.

Ostermeier, M., 2003. Synthetic gene libraries: in search of the optimal diversity. Trends Biotechnol. 21 (6), 244–247.

Palzkill, T., Botstein, D., 1992. Identification of amino acid substitutions that alter the substrate specificity of TEM-1 beta-lactamase. J. Bacteriol. 174 (16), 5237–5243.

Petrounia, I.P., Arnold, F.H., 2000. Designed evolution of enzymatic properties. Curr. Opin. Biotechnol. 11 (4), 325–330.

Platt, J.C., 1998. Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Microsoft Research (MSR-TR-98-14).

Raquet, X., Vanhove, M., Lamotte-Brasseur, J., Goussard, S., Courvalin, P., Frere, J.M., 1995. Stability of TEM *b*-Lactamase mutants hydrolyzing third generation cephalosporins. Proteins: Struct. Funct. Genet. 23 (1), 63–72.

Saraf, M.C., Horswill, A.R., Benkovic, S.J., Maranas, C.D., 2004. FamClash: a method for ranking the activity of engineered enzymes. PNAS USA 101 (12), 4142–4147.

Siezen, R.J., de Vos, W.M., Leunissen, J.A., Dijkstra, B.W., 1991. Homology modelling and protein engineering strategy of subtilases, the family of subtilisin-like serine proteinases. Protein Eng. 4 (7), 719–737.

Stemmer, W.P.C., 1994. DNA shuffling by random fragmentation and reassembly: in vitro recombination for molecular evolution. PNAS USA 91 (22), 10747–10751.

Swets, J.A., Pickett, R.M., 1982. The Evaluation of Diagnostic Systems: Methods from Signal Detection Theory. Academic Press, New York.

Terwilliger, T.C., Zabin, H.B., Horvath, M.P., Sandberg, W.S., Schlunk, P.M., 1994. In vivo characterization of mutants of the bacteriophage f1 gene V protein isolated by saturation mutagenesis. J. Mol. Biol. 236 (2), 556–571.

To, K.N., Lim, C.C., 2004. Perturbation to enhance support vector machines for classification. J. Comput. Appl. Math. 163 (1), 233–239.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer, Berlin.

Voet, D., Voet, J.G., 1995. Biochemistry, Wiley Textbooks.

Voigt, C.A., Mayo, S.L., Arnold, F.H., Wang, Z.G., 2001. Computational method to reduce the search space for directed protein evolution. Proc. Natl Acad. Sci. USA 98 (7), 3778–3783.

Ward, J.J., McGuffin, L.J., Buxton, B.F., Jones, D.T., 2003. Secondary structure prediction with support vector machines. Bioinformatics 19 (13), 1650–1655.

Zaccolo, M., Gherardi, E., 1999. The effect of high-frequency random mutagenesis on in vitro protein evolution: a study on TEM-1 [beta]-lactamase1. J. Mol. Biol. 285 (2), 775–783.