Research Article

Journal of Molecular Recognition

Received: 5 December 2007, Revised: 24 March 2008, Accepted: 29 March 2008, Published online in Wiley InterScience: 2008

(www.interscience.wiley.com) DOI:10.1002/jmr.893

Predicting linear B-cell epitopes using string kernels

Yasser EL-Manzalawy[a,b,d,e]*, Drena Dobbs[c,d,e] and Vasant Honavar[a,b,c,e]

The identification and characterization of B-cell epitopes play an important role in vaccine design, immunodiagnostic tests, and antibody production. Therefore, computational tools for reliably predicting linear B-cell epitopes are highly desirable. We evaluated Support Vector Machine (SVM) classifiers trained utilizing five different kernel methods using fivefold cross-validation on a *homology-reduced* data set of 701 linear B-cell epitopes, extracted from Bcipep database, and 701 non-epitopes, randomly extracted from SwissProt sequences. Based on the results of our computational experiments, we propose BCPred, a novel method for predicting linear B-cell epitopes using the subsequence kernel. We show that the predictive performance of BCPred (AUC = 0.758) outperforms 11 SVM-based classifiers developed and evaluated in our experiments as well as our implementation of AAP (AUC = 0.7), a recently proposed method for predicting linear B-cell epitopes using amino acid pair antigenicity. Furthermore, we compared BCPred with AAP and ABCPred, a method that uses recurrent neural networks, using two data sets of *unique* B-cell epitopes that had been previously used to evaluate ABCPred. Analysis of the data sets used and the results of this comparison show that conclusions about the relative performance of different B-cell epitope prediction methods drawn on the basis of experiments using data sets of *unique* B-cell epitopes are likely to yield overly optimistic estimates of performance of evaluated methods. This argues for the use of carefully *homology-reduced* data sets in comparing B-cell epitope prediction methods to avoid misleading conclusions about how different methods compare to each other. Our *homology-reduced* data set and implementations of BCPred as well as the APP method are publicly available through our web-based server, BCPREDS, at: http://ailab.cs.iastate.edu/bcpreds/.

Keywords: linear B-cell epitope; epitope mapping; epitope prediction

INTRODUCTION

B-cell epitopes are antigenic determinants that are recognized and bound by receptors (membrane-bound antibodies) on the surface of B lymphocytes (Pier *et al*., 2004). There are many different types of B-cell receptors, but each B-cell produces only one type. When a B-cell receptor binds its cognate antigen, the B-cell is stimulated to undergo proliferation. This involves the generation of two types of cells, effector or plasma B-cells, which produce and secrete soluble antibodies, and memory B-cells, which remain in the organism and can proliferate rapidly if re-exposed to antigen. Hence, understanding the sequence and structural features of B-cell epitopes is critical both for the design of effective vaccines and for the development of sensitive diagnostic tests.

B-cell epitopes can be classified into two types: linear (continuous) epitopes and conformational (discontinuous) epitopes. Linear epitopes are short peptides, corresponding to a contiguous amino acid sequence fragment of a protein (Barlow *et al*., 1986; Langeveld *et al*., 2001). In contrast, conformational epitopes are composed of amino acids that are not contiguous in primary sequence, but are brought into close proximity within the folded protein structure. Although it is believed that a large majority of B-cell epitopes are discontinuous (Walter, 1986), experimental epitope identification has focused primarily on linear B-cell epitopes (Flower, 2007). Even in the case of linear B-cell epitopes, however, antibody–antigen interactions are often conformation-dependent. The conformation-dependent aspect of antibody binding complicates the problem of B-cell epitope prediction, making it less tractable than T-cell epitope prediction. Therefore, the development of reliable computational methods for predicting linear B-cell epitopes is an important challenge in bioinformatics and computational biology (Greenbaum *et al*., 2007).

Several studies have reported correlations between certain physicochemical properties of amino acids and the locations of linear B-cell epitopes within protein sequences (Emini *et al*., 1985; Karplus and Schulz, 1985; Parker *et al*., 1986; Pellequer *et al*., 1991; Pellequer *et al*., 1993), and several epitope prediction methods based on physicochemical properties of amino acids have been proposed. For example, hydrophilicity, flexibility, turns, or solvent accessibility propensity scales were used in the methods of Parker

* *Correspondence to: Y. EL-Manzalawy, 226 Atanasoff Hall, Iowa State University, Ames, IA 50010, USA.*
E-mail: yasser@iastate.edu

a *Y. EL-Manzalawy, V. Honavar*
Artificial Intelligence Laboratory, Iowa State University, Ames, IA 50010, USA

b *Y. EL-Manzalawy, V. Honavar*
Department of Computer Science, Iowa State University, Ames, IA 50010, USA

c *D. Dobbs*
Department of Genetics, Development and Cell Biology, Iowa State University, Ames, IA 50010, USA

d *Y. EL-Manzalawy, D. Dobbs, V. Honavar*
Bioinformatics and Computational Biology Graduate Program, Iowa State University, Ames, IA 50010, USA

e *Y. EL-Manzalawy, D. Dobbs, V. Honavar*
Center for Computational Intelligence, Learning and Discovery, Iowa State University, Ames, IA 50010, USA

et al., (1986), Karplus and Schulz (1985), Pellequer *et al.* (1993) and Emini *et al.* (1985), respectively. PREDITOP (Pellequer and Westhof, 1993), PEOPLE (Alix, 1999), BEPITOPE (Odorico and Pellequer, 2003), and BcePred (Saha and Raghava, 2004) predict linear B-cell epitopes based on groups of physicochemical properties instead of a single property.

Recently, Blythe and Flower (2005) performed an exhaustive assessment of 484 amino acid propensity scales, combined with ranges of profile parameters, to examine the correlation between propensity scale-based profiles and the location of linear B-cell epitopes in a set of 50 proteins. They reported that for predicting B-cell epitopes based on amino acid sequence information, even the best combinations of amino acid propensities performed only marginally better than random. They concluded that the reported performance of such methods in the literature is likely to have been overly optimistic, in part due to the small size of the data sets on which the methods had been evaluated.

Motivated by Blythe and Flower (2005) results and the increasing availability of experimentally identified linear B-cell epitopes, several studies have attempted to improve the accuracy of linear B-cell epitope prediction methods using machine learning approaches. BepiPred (Larsen *et al*., 2006) combines two amino acid propensity scales and a Hidden Markov Model (HMM) trained on linear epitopes to yield a slight improvement in prediction accuracy relative to techniques that rely on analysis of amino acid physicochemical properties. ABCPred (Saha and Raghava, 2006b) uses artificial neural networks for predicting linear B-cell epitopes. Both feed-forward and recurrent neural networks were evaluated on a *non-redundant* data set of 700 B-cell epitopes and 700 non-epitope peptides, using fivefold cross-validation tests. Input sequence windows ranging from 10 to 20 amino acids, were tested and the best performance, 66% accuracy, was obtained using a recurrent neural network trained on peptides 16 amino acids in length. In the method of Söllner and Mayer (2006), each epitope is represented using a set of 1487 features extracted from a variety of propensity scales, neighborhood matrices, and respective probability and likelihood values. Of two machine learning methods tested, decision trees and a nearest-neighbor method combined with feature selection, the latter was reported to attain an accuracy of 72% on a data set of 1211 B-cell epitopes and 1211 non-epitopes, using a fivefold cross-validation test (Söllner and Mayer, 2006). Chen *et al.* (2007) observed that certain amino acid pairs (AAPs) tend to occur more frequently in B-cell epitopes than in non-epitope peptides. Using an AAP propensity scale based on this observation, in combination with a support vector machine (SVM) classifier, they reported prediction accuracy of 71% on a data set of 872 B-cell epitopes and 872 non-B-cell epitopes, estimated using fivefold cross-validation. In addition, Chen *et al.* (2007) demonstrated an improvement in the prediction accuracy, 72.5%, when the APP propensity scale is combined with turns, accessibility, antigenicity, hydrophilicity, and flexibility propensity scales.

In this report, we present BCPred, a method for predicting linear B-cell epitopes using an SVM machine learning method. Although the performance of SVM-based classifiers largely depends on the selection of the kernel function, there are no theoretical foundations for choosing good kernel functions in a data-dependent way. Therefore, one objective of this study was to explore a class of kernel methods, namely string kernels, in addition to the widely used radial bias function (RBF) kernel. Our choice of string kernels was motivated by their successful application in a number of bioinformatics classification tasks, including protein remote homology detection (Leslie *et al*., 2002, 2004; Zaki *et al*., 2005), protein structure prediction (Rangwala *et al*., 2006), protein binding site prediction (Wu *et al*., 2006), and major histocompatibility complex (MHC) binding peptide prediction (Salomon and Flower, 2006). In addition, we introduce the subsequence kernel (SSK), which has been successfully used in text classification (Lodhi *et al*., 2002), but has been under-explored in macromolecular sequence classification applications. Our empirical results demonstrate superior performance of SSK over other string kernels and the RBF kernel. Hence, we employed the SSK in building SVM classifiers for our proposed linear B-cell epitope prediction method, BCPred.

A second goal of this study was to determine how existing methods for linear B-cell epitope prediction compare with each other and with BCPred. At present, little is known about the relative performance of different methods, due to the lack of published direct comparisons using standard benchmark data sets. Unfortunately, neither the data set used by Söllner and Mayer (2006) nor the code used for generating and selecting the features used to represent epitope peptides as input to the classifiers is publicly available. The code for the AAP method (Chen *et al*., 2007) is also not publicly available; however, in contrast to the other methods, it is relatively straightforward for implementation. Fortunately, although the code used to train the neural network classifier used in ABCPred is not publicly available, Saha and Raghava (2006b) have made available the data set used for developing and evaluating the ABCPred server, as well as a blind test set (Saha and Raghava, 2006a). Thus, although we are unable to include direct comparisons with results of Söllner and Mayer (Söllner and Mayer, 2006), in this paper we report direct comparisons of the ABCPred method (Saha and Raghava, 2006b), our implementation of the AAP method of Chen *et al.* (2007), and our proposed BCPred method, using the ABCPred data sets made publicly available by Saha and Raghava (2006a).

METHODS

Data sets

Homology-reduced data sets

Bcipep database (Saha *et al*., 2005) contains 1230 unique linear B-cell epitopes. We retrieved a set of 947 unique epitopes with each epitope satisfying one of the following two conditions: (i) the epitope length is at least 20 amino acids; or (ii) the epitope is less than 20 amino acids in length and the accession number of the source antigen is provided.

A set of 20-mer peptides was derived from the 947 unique epitopes by: (i) truncating epitopes longer than 20 residues by removing amino acids from both ends to yield a 20-mer from the middle, and (ii) extending epitopes shorter than 20 residues by adding amino acids on both ends, based on the corresponding complete antigen sequences retrieved from SwissProt (Bairoch and Apweiler, 2000). Because the resulting data set of 947 20-mer peptides was no longer *non-redundant*, we removed duplicated and highly homologous peptides by filtering the data set based on an 80% sequence identity cutoff using the CD-HIT program (Li *et al*., 2002) to obtain a *homology-reduced* data set of 701 peptides (positive instances of B-cell epitopes). A total of 701 non-epitope peptides were generated by randomly extracting 20-mer peptides from sequences in SwissProt database (Bairoch and Apweiler, 2000) while ensuring that none of the negative instances so obtained also occur in the positive instances.

Because there is no evidence that 20 amino acids is the optimal length for B-cell epitopes, we decided to experiment with different epitope lengths. Variants of the 20-mer data set were generated by repeating the above procedure for peptide lengths of 18,16, 14, or 12 residues. For the sake of brevity, we will refer to these data sets by BCP*nn*, where *nn* is a two digit number representing the length of the peptides in the data set (e.g., BCP16 refers to the *homology-reduced* data set where each peptide is composed of 16 residues).

It should be noted that deriving a data set of shorter peptides from the 20-mer data set by trimming amino acids from both termini of each peptide is not guaranteed to produce a data set with <80% sequence identity because such trimming could increase the similarity between two peptides in the data set. Therefore, to ensure that the resulting data sets are *homology-reduced*, we reapplied the 80% sequence identity cutoff filter in generating each data set of epitopes less than 20 residues in length. The resulting *homology-reduced* data sets (BCP20, BCP18, BCP16, BCP14, and BCP12) are available at http://ailab.cs.iastate.edu/bcpreds/.

ABCPred data set

Saha and Raghava (2006a) have made available the data sets used to train and evaluate ABCPred. Because the best reported performance of ABCPred was obtained using a 16-mer peptide data set (ABCP16), we chose this data set for directly comparing ABCPred with BCPred and AAP (Chen *et al*., 2007) using fivefold cross-validation.

Blind test set

Saha and Raghava (2006a) have made available a blind test set comprising 187 epitopes, none of which were used in training the ABCPred method, and a set of 200 16-mer non-epitope peptides extracted from the non-allergen data set of Björklund *et al*. (2005). B-cell epitopes less than 16 amino acids in length were extended to 16-mer peptides by adding an equal number of residues to both ends based on the protein sequence of the source antigen. In the remaining text, we will use the abbreviation (SBT16) to refer to Saha 16-mer blind test set.

Support vector machines and kernel methods

Support vector machines (SVMs) (Vapnik, 2000) are a class of supervised machine learning methods used for classification and regression. Given a set of labeled training data (x_i,y_i), where $x_i \in R^d$ and $y_i \in \{+1, -1\}$, training an SVM classifier involves finding a hyperplane that maximizes the geometric margin between positive and negative training data samples. The hyperplane is described as $f(x) = \langle w, x\rangle + b$, where w is a normal vector and b is a bias term. A test instance, x, is assigned a positive label if $f(x) > 0$, and a negative label otherwise. When the training data are not linearly separable, a kernel function is used to map nonlinearly separable data from the input space into a feature space. Given any two data samples x_i and x_j in an input space $X \in R^d$, the kernel function K returns $K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j)\rangle$ where Φ is a nonlinear map from the input space X to the corresponding feature space. The kernel function K has the property that $K(x_i,x_j)$ can be computed without explicitly mapping x_i and x_j into the feature space, but instead, using their dot product $\langle x_i, x_j\rangle$ in the input space. Therefore, the kernel trick allows us to train a linear classifier, e.g., SVM, in a high-dimensional feature space where the data are assumed to be linearly separable without explicitly mapping each training example from the input space into the feature space. This approach relies implicitly on the selection of a feature space in which the training data are likely to be linearly separable (or nearly so) and explicitly on the selection of the kernel function to achieve such separability. Unfortunately, there is no single kernel that is guaranteed to perform well on every data set. Consequently, the SVM approach requires some care in selecting a suitable kernel and tuning the kernel parameters (if any).

String kernels

String kernels (Haussler, 1999; Lodhi *et al*., 2002; Leslie *et al*., 2002, 2004; Saigo *et al*., 2004) are a class of kernel methods that have been successfully used in many sequence classification tasks (Leslie *et al*., 2002, 2004; Saigo *et al*., 2004; Zaki *et al*., 2005; Rangwala *et al*., 2006; Wu *et al*., 2006). In these applications, a protein sequence is viewed as a string defined on a finite alphabet of 20 amino acids. In this work, we explore four string kernels: spectrum (Leslie *et al*., 2002), mismatch (Leslie *et al*., 2004), local alignment (Saigo *et al*., 2004), and subsequence (Lodhi *et al*., 2002), in predicting linear B-cell epitopes. The subsequence kernel (Lodhi *et al*., 2002) has proven useful in text classification (Lodhi *et al*., 2002) and natural language processing (Clark *et al*., 2006). However, to the best of our knowledge, this kernel has not been previously explored in the context of macromolecular sequence classification problems. A brief description of the four kernels follows.

Spectrum kernel

Let A denote a finite alphabet, e.g., 20 amino acids. x and y denote two strings defined on the alphabet A. For $k \geq 1$, the k-spectrum is defined as (Leslie *et al*., 2002):

$$\Phi_k = (\phi_\alpha(x))_{\alpha \in A^k} \tag{1}$$

where ϕ_α is the number of occurrences of the k-length substring α in the sequence x. The k-spectrum kernel of the two sequences x and y is obtained by taking the dot product of the corresponding k spectra:

$$K_k^{spct}(x,y) = \langle \Phi_k(x), \Phi_k(y)\rangle \tag{2}$$

Intuitively, this kernel captures a simple notion of string similarity: two strings are deemed similar (i.e., have a high k-spectrum kernel value) if they share many of the same k-length substrings.

Mismatch kernel

The mismatch kernel (Leslie *et al*., 2004) is a variant of the spectrum kernel in which inexact matching is allowed. Specifically, the (k, m)-mismatch kernel allows up to $m \leq k$ mismatches to occur when comparing two k-length substrings. Let α be a k-length substring, the (k, m)-mismatch feature map is defined on α as:

$$\Phi_{(k,m)}(\alpha) = (\phi_\beta(\alpha))_{\beta \in A^k} \tag{3}$$

where $\phi_\beta(\alpha) = 1$ if $\beta \in N_{(k,m)(\alpha)}$, where β is the set of k-mer substrings that differs from α by at most m mismatches. Then, the feature map of an input sequence x is the sum of the feature

vectors for k-mer substrings in x:

$$\Phi_{(k,m)}(x) = \sum_{k-\text{mers } \alpha \text{ in x}} \Phi_{(k,m)}(\alpha) \tag{4}$$

The (k, m)-mismatch kernel is defined as the dot product of the corresponding feature maps in the feature space:

$$K^{\text{msmtch}}_{(k,m)}(x, y) = \langle \Phi_{(k,m)}(x), \Phi_{(k,m)}(y) \rangle \tag{5}$$

It should be noted that the $(k, 0)$-mismatch kernel results in a feature space that is identical to that of the k-spectrum kernel. An efficient data structure for computing the spectrum and mismatch kernels in $O(|x|+|y|)$ and $O(k^{m+1}|A|^m(|x| + |y|))$, respectively, has been provided by Leslie *et al*. (2004).

Local alignment kernel

Local alignment (LA) kernel (Saigo *et al*., 2004) is a string kernel adapted for biological sequences. The LA kernel measures the similarity between two sequences by summing up scores obtained from gapped local alignments of the sequences. This kernel has several parameters: the gap opening and extension penalty parameters, d and e, the amino acid mutation matrix s, and the factor β, which controls the influence of suboptimal alignments on the kernel value. Detailed formulation of the LA kernel and a dynamic programming implementation of the kernel with running time complexity in $O(|x||y|)$ have been provided by Saigo *et al*. (2004).

Subsequence kernel

The subsequence kernel (Lodhi *et al*., 2002) generalizes the k-spectrum kernel by considering a feature space generated by the set of all (contiguous and non-contiguous) k-mer subsequences. For example, if we consider the two strings "*act*" and "*acctct*", the value returned by the spectrum kernel with $k = 3$ is 0. On the other hand, the (3, 1)-mismatch kernel will return 3 because the 3-mer substrings "*acc*", "*cct*", and "*tct*" have at most one mismatch when compared with "*act*". The subsequence kernel considers the set ("*ac-t*", "*a-ct*", "*ac—t*", "*a-c–t*", "*a—ct*") of non-contiguous substrings and returns a similarity score that is weighted by the length of each non-contiguous substring. Specifically, it uses a decay factor, $\lambda \leq 1$, to penalize non-contiguous substring matches. Therefore, the subsequence kernel with $k = 3$ will return $2\lambda^4 + 3\lambda^6$ when applied to "*act*" and "*acctct*" strings. More precisely, the feature map Φ_k of a string x is given by

$$\Phi_{(k,\lambda)}(x) = \left(\sum_{i:u=x[i]} \lambda^{l(i)} \right)_{u \in A^k} \tag{6}$$

where $u = x(i)$ denotes a substring in x where $1 \leq i_1 < \cdots < i_{|u|} \leq |x|$ such that $u_j = s_{ij}$, for $j = 1,\ldots, |u|$ and $l(i) = i_{|u|} - i_1 + \text{l}$ is the length of the subsequence in x. The subsequence kernel for two strings x and y is determined as the dot product of the corresponding feature maps:

$$\begin{aligned} K(x, y)^{\text{sub}}_{(k,\lambda)} &= \langle \Phi_{(k,\lambda)}(x), \Phi_{(k,\lambda)}(y) \rangle \\ &= \sum_{u \in A^k} \sum_{i:u=x[i]} \lambda^{l(i)} \sum_{j:u=y[j]} \lambda^{l(j)} \\ &= \sum_{u \in A^k} \sum_{i:u=x[i]} \sum_{j:u=y[j]} \lambda^{l(j)+l(j)} \end{aligned} \tag{7}$$

This kernel can be computed using a recursive algorithm based on dynamic programming in $O(k|x||y|)$ time and space. The running time and memory requirements can be further reduced using techniques described by Seewald and Kleedorfer (2005).

Amino acid pairs propensity scale

Amino acid pairs (AAPs) are obtained by decomposing a protein/peptide sequence into its 2-mer subsequences. Chen *et al.* (2007) observed that some particular AAPs tend to occur more frequently in B-cell epitopes than in non-epitope peptides. Based on this observation, they developed an AAP propensity scale defined by:

$$\theta(\alpha) = \log\left(\frac{f^+_\alpha}{f^-_\alpha}\right) \tag{8}$$

where f^+_α and f^-_α are the occurrence frequencies of AAP α in the epitope and non-epitope peptide sequences, respectively. These frequencies have been derived from Bcipep (Saha *et al*., 2005) and Swissprot (Bairoch and Apweiler, 2000) databases, respectively. To avoid the dominance of an individual AAP propensity value, the scale in equation (8) has been normalized to a $(-1, +1)$ interval through the following conversion:

$$\theta(\alpha) = 2\left(\frac{\theta(\alpha) - \min}{\max - \min}\right) - 1 \tag{9}$$

where max and min are the maximum and minimum values of the propensity scale before the normalization.

Chen *et al.* (2007) explored SVMs using two kernels: a dot product kernel applied to the average of the AAP scale values for all the AAPs in a peptide and an RBF kernel defined in a 400-dimensional feature space as follows:

$$\Phi_{\text{AAP}}(x) = (\phi_\alpha(x) \cdot \theta(\alpha))_{\alpha \in A^2} \tag{10}$$

where $\phi_\alpha(x)$ is the number of occurrences of the 2-mer α in the peptide x. The optimal performance was obtained using the RBF kernel and a window of 20 amino acids (Chen *et al*., 2007).

Fivefold cross-validation

In our experiments, we used stratified fivefold cross-validation tests in which the data set is randomly partitioned into five equal subsets such that the relative proportion of epitopes to non-epitopes in each subset is 1:1. Four of the five subsets are used for training the classifier and the fifth subset is used for testing the classifier. This procedure is repeated five times, each time choosing different subsets of the data for training and testing. The estimated performance of the classifier corresponds to an average of the results from the five cross-validation runs.

Implementation and SVM parameter optimization

We used Weka (Witten and Frank, 2005) machine learning workbench for implementing the spectrum, mismatch, and LA kernels (RBF and the subsequence kernel are already implemented in Weka). We evaluated the k-spectrum kernel, K^{spct}_k, for $k = 1$, 2, and 3. The (k, m)-mismatch kernel was evaluated at (k, m) equals (3, 1), (4, 1), (5, 1), and (5, 2). The subsequence kernel, $K^{\text{sub}}_{(k,\lambda)}$, was evaluated at $k = 2$, 3, and 4 and the default value for λ, 0.5. The LA kernel was evaluated using the BLOSUM62 substitution matrix, gap opening and extension parameters equal to 10 and 1, respectively, and $\beta = 0.5$. For the SVM classifier, we used the Weka implementation of the SMO (Platt, 1998) algorithm. For the string

kernels, the default value of the C parameter, $C = 1$, was used for the SMO classifier. For methods that use the RBF kernel, we found that tuning the SMO cost parameter C and the RBF kernel parameter γ is necessary to obtain satisfactory performance. We tuned these parameters using a two-dimensional grid search over the range $C = 2^{-5}, 2^{-3}, \ldots, 2^{3}$, $\gamma = 2^{-15}, 2^{-13}, \ldots, 2^{3}$. *It should be noted that the parameter optimization was performed using only the training data.*

Performance evaluation

The prediction accuracy (ACC), sensitivity (S_n), specificity (S_p), and correlation coefficient (CC) are often used to evaluate prediction algorithms (Baldi *et al*., 2000). The CC measure has a value in the range from -1 to $+1$, and the closer the value to $+1$, the better the predictor. ACC, S_n, S_p, and CC are defined as follows:

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \tag{11}$$

$$S_n = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \text{ and } S_p = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{12}$$

$$\mathrm{CC} = \frac{\mathrm{TP} \times \mathrm{TN} - \mathrm{FP} \times \mathrm{FN}}{\sqrt{(\mathrm{TN} + \mathrm{FN})(\mathrm{TN} + \mathrm{FP})(\mathrm{TP} + \mathrm{FN})(\mathrm{TP} + \mathrm{FP})}} \tag{13}$$

where TP, FP, TN, and FN are the numbers of true positives, false positives, true negatives, and false negatives, respectively.

Although these metrics are widely used to assess the performance of machine learning methods, they all suffer from the important limitation of being threshold-dependent. Threshold-dependent metrics describe the classifier performance at a specific threshold value. It is often possible to increase the number of true positives (equivalently, sensitivity) of the classifier at the expense of an increase in false positives (equivalently, false alarm rate). The receiver operating characteristic (ROC) curve describes the performance of the classifier over all possible thresholds. The ROC curve is obtained by plotting the true positive rate as a function of the false positive rate or, equivalently, sensitivity versus (1-specificity) as the discrimination threshold of the binary classifier is varied. Each point on the ROC curve describes the classifier at a certain threshold value, i.e., at a particular choice of tradeoff between true positive rate and false positive rate. The area under ROC curve (AUC) is a useful summary statistic for comparing two ROC curves. AUC is defined as the probability that a randomly chosen positive example will be ranked higher than a randomly chosen negative example. An ideal classifier will have an AUC = 1, while a classifier assigning labels at random will have an AUC = 0.5, any classifier performing better than random will have an AUC value that lies between these two extremes.

RESULTS

SVM using the subsequence kernel outperforms other kernel methods and the AAP method

In the first set of experiments, we used our *homology-reduced* data sets to evaluate SVMs trained using the spectrum kernel at $k = 1$, 2, and 3, the (k, m)-mismatch kernel at (k, m) = (3, 1), (4, 1), (5, 1), and (5, 2), the LA kernel, and the subsequence kernel at $k = 2$, 3, and 4. We compared the performance of the four string kernels to that of the RBF kernel trained using a binary representation of the data in which each amino acid is represented by a 20-bit binary string. In addition, we evaluated our implementation of the AAP method (Chen *et al*., 2007) on our data sets. For all methods, the performance was evaluated using fivefold cross-validation. Because it is not feasible to include the complete set of results in this paper, we report only the results on the 20-mer peptides data set, BCP20, and provide the results on data sets BCP18, BCP16, BCP14, and BCP12 in the Supplementary Materials.

Table 1 compares the performance of different kernel-based SVM classifiers on BCP20 data set. The subsequence kernel has the best overall performance, in terms of AUC. The (5, 1)-mismatch kernel performs slightly better than the k-spectrum kernel, and the performance of k-spectrum kernel with $k = 1$ and $k = 3$ is much better than its performance with $k = 2$. The performance of both the k-spectrum and (k, m)-mismatch kernels appears to be very sensitive to the choice of k and m parameters, because for some choices of k and m, the classifier performance deteriorates to that expected for random assignment of labels to test instances. In contrast, the performance of the subsequence kernel appears to be much less sensitive to the choice of parameter k.

Our implementation of the AAP method (Chen *et al*., 2007) has the second best overall performance and demonstrates the highest specificity. The LA kernel is very competitive in performance with AAP. Interestingly, the AAP significantly outperforms the RBF kernel trained using data in its binary representation. The AAP method is essentially an RBF kernel trained on the same data but using a different representation in which each peptide is represented by a vector of 400 numeric values computed based on the AAP propensity scale. The significant difference observed in performance of these two RBF-based methods highlights the importance of the data representation in kernel methods. All of these observations hold not only for the BCP20 data set but also for the *homology-reduced* data sets of peptides with different lengths (see Supplementary Materials). Most of the methods have their best performance on BCP20 data set and show slight decreases in performance on data sets with decreasing peptide length.

Table 1. Performance of different methods on our BCP20 *homology-reduced* data set using fivefold cross-validation. BCPred method denotes $K^{\mathrm{sub}}_{(4,0.5)}$

Method	ACC(%)	S_n(%)	S_p(%)	CC	AUC
K^{spct}_{1}	62.62	60.63	64.62	0.253	0.681
K^{spct}_{2}	58.56	59.49	57.63	0.171	0.614
K^{spct}_{3}	64.12	63.2	65.05	0.283	0.660
$K^{\mathrm{msmtch}}_{(3,1)}$	48.86	50.5	47.22	−0.023	0.468
$K^{\mathrm{msmtch}}_{(4,1)}$	55.35	54.64	56.06	0.107	0.593
$K^{\mathrm{msmtch}}_{(5,1)}$	64.91	62.05	67.76	0.299	0.683
$K^{\mathrm{msmtch}}_{(5,2)}$	55.85	55.35	56.35	0.117	0.584
LA	64.76	61.63	67.9	0.296	0.696
$K^{\mathrm{sub}}_{(2,0.5)}$	62.62	62.34	62.91	0.253	0.664
$K^{\mathrm{sub}}_{(3,0.5)}$	65.83	67.48	64.19	0.317	0.722
BCPred	67.90	72.61	63.2	0.360	0.758
RBF	57.28	57.49	57.06	0.146	0.617
AAP	64.05	52.92	75.18	0.288	0.700

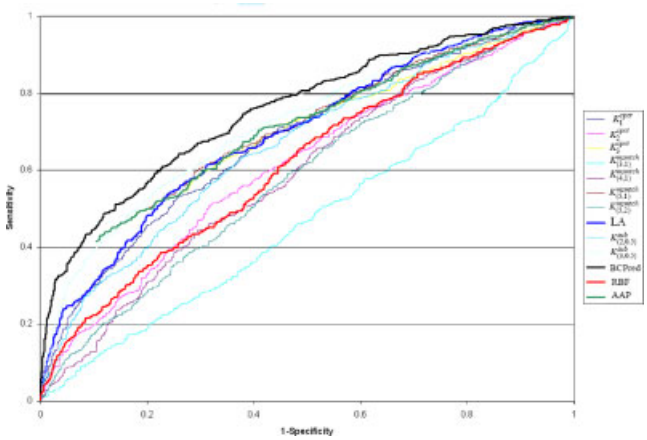

Figure 1. ROC curves for different prediction methods on BCP20 homology-reduced data set. BCPred method denotes $K^{sub}_{(4,0.5)}$. The BCPred ROC curve dominates all other ROC curves for any user-selected threshold corresponding to specificity in the range of 100 to 20%.

Figure 1 shows the ROC curves for all methods evaluated in this experiment. The ROC curve for the subsequence kernel, $K^{sub}_{(4,0.5)}$, dominates the other ROC curves over a broad range of choices for the tradeoff between true positive and false positive rates. For any user-selected threshold corresponding to specificity in the range 100 to 20%, $K^{sub}_{(4,0.5)}$ has the best corresponding sensitivity. We conclude that BCPred, SVM-based classifier trained using the subsequence kernel $K^{sub}_{(4,0.5)}$, outperforms all other methods tested in predicting linear B-cell epitopes.

Statistical analysis

We summarize statistical analysis of the results and conclusions presented in the preceding subsection. Specifically, we attempt to answer, from a statistical prospective, the following questions: is the performance of BCPred significantly different from those of other methods? Or more generally, how do the different B-cell epitope prediction methods compare with each other?

To answer these questions, we utilized multiple hypothesis comparisons (Fisher, 1973; Friedman, 1940) for comparing a set of classifiers on multiple data sets. We chose to use the AUC as the performance metric in these tests. Table 2 shows the AUC values of 13 classifiers on the five *homology-reduced* data sets.

One approach for performing multiple hypothesis comparisons over the results in Table 2, is to perform paired *t*-tests between each pair of classifiers at *p*-value equal to 0.05. However, when the number of classifiers being compared is large

Table 2. AUC values for different methods evaluated on *homology-reduced* data sets. For each data set, the rank of each classifier is shown in parentheses

Method	BCP20	BCP18	BCP16	BCP14	BCP12	Avg
K^{spct}_{1}	0.681(6)	0.588(11)	0.652(7)	0.582(11)	0.591(10)	0.619(9)
K^{spct}_{2}	0.614(10)	0.636(8)	0.612(9)	0.597(9)	0.606(8)	0.613(8.8)
K^{spct}_{3}	0.660(8)	0.675(6)	0.645(8)	0.675(3)	0.636(6)	0.658(6.2)
$K^{msmtch}_{(3,1)}$	0.468(13)	0.465(13)	0.460(13)	0.506(13)	0.450(13)	0.470(13)
$K^{msmtch}_{(4,1)}$	0.593(11)	0.599(10)	0.569(11)	0.596(10)	0.548(11)	0.581(10.6)
$K^{msmtch}_{(5,1)}$	0.683(5)	0.691(4.5)	0.667(6)	0.649(6)	0.594(9)	0.657(6.1)
$K^{msmtch}_{(5,2)}$	0.584(12)	0.568(12)	0.563(12)	0.574(12)	0.535(12)	0.565(12)
LA	0.696(4)	0.691(4.5)	0.686(4)	0.671(4)	0.662(4)	0.681(4.1)
$K^{sub}_{(2,0.5)}$	0.664(7)	0.668(7)	0.681(5)	0.647(7)	0.643(5)	0.661(6.2)
$K^{sub}_{(3,0.5)}$	0.722(2)	0.726(2)	0.718(2)	0.697(2)	0.687(2)	0.710(2)
BCPred	0.758(1)	0.751(1)	0.730(1)	0.733(1)	0.709(1)	0.736(1)
RBF	0.617(9)	0.601(9)	0.594(10)	0.603(8)	0.620(7)	0.607(8.6)
AAP	0.700(3)	0.699(3)	0.689(3)	0.665(5)	0.663(3)	0.683(3.4)

compared to the number of datasets, paired *t*-tests are susceptible to type I error, i.e., falsely concluding that the two methods significantly differ from each other in terms of performance when in fact they do not. To reduce the chance of type I errors, we used Bonferroni adjustments (Neter *et al*., 1985) in performing multiple comparisons. Specifically, two classifiers are considered different at 0.05 significance level, if the null hypothesis (that they are not different) is rejected by a paired *t*-test at $0.05/12 = 0.0042$ confidence level (12 denotes the number of comparisons). Table 3 summarizes the results of Bonferroni-corrected tests comparing the performance of the classifiers. Significantly different pairs of classifiers are indicated with a $\times$. The results in Table 3 show that the reported performance of BCPred is significantly different from the performance of other classifiers. On the other hand, the differences between the performance of K_3^{spct}, $K_{(5,1)}^{msmtch}$, LA, and $K_{(3,0.5)}^{sub}$ classifiers and the performance of AAP are not statistically significant.

A second approach for performing multiple hypothesis comparisons over the results in Table 2 is to use non-parametric tests. Demšar (Demšar, 2006) has suggested that non-parametric tests should be preferred over parametric tests for comparing machine learning algorithms because the non-parametric tests, unlike parametric tests, do not assume normal distribution of the samples (e.g., the data sets). Demšar suggested a three-step procedure for performing multiple hypothesis comparisons using non-parametric tests. First, the classifiers being compared are ranked on the basis of their observed performance on each data set (see Table 2). Second, Friedman test is applied to determine whether the measured average ranks are significantly different from the mean rank under the null hypothesis. Third, if the null hypothesis can be rejected at 0.05 significance level, the Nemenyi test is used to determine whether significant differences exist between any given pair of classifiers. Unfortunately, this procedure requires the number of data sets to be greater than 10 and the number of methods to be greater than 5 (Demšar, 2006). Because we have 13 classifiers to compare and only 5 data sets, we cannot use this procedure. However, as noted by Demšar (Demšar, 2006), the average ranks by themselves provide a reasonably fair comparison of classifiers. Hence, we use average ranks to compare BCPred with the other methods. As shown in Table 2, BCPred and $K_{(3,0.5)}^{sub}$ have average ranks 1 and 2, respectively, followed by AAP and LA kernel with average ranks 3.4 and 4.1.

In summary, the results reported in Table 2 along with the statistical analysis of the results lend support to the conclusion summarized in the preceding subsection that the performance of BCPred is superior to that of the other 12 methods.

Effect of epitope length on BCPred performance

Our choice of an epitope length of 20 amino acids in the experiments summarized above was motivated by the previous works (Saha and Raghava, 2006b; Chen *et al*., 2007). Figure 2 shows the distribution of unique epitope lengths in Bcipep database (Saha *et al*., 2005). The Bcipep database contains 1230 unique B-cell epitopes with 99.4% of the epitopes having lengths ranging from 3 to 38 amino acids. It turns out that 86.7% of the unique B-cell epitopes are at most 20 amino acids in length. However, it is natural to ask as to how the performance of BCPred varies with the choice of epitope length. We now proceed to examine the effect of epitope length on the performance of BCPred.

In order to study the effect of epitope length we compared the performance of BCPred and other methods trained and tested on data sets with epitope lengths of 20, 18, 16, 14, and 12 amino acids. Our results show that BCPred and five other methods reach their best performance (in terms of AUC) on data set BCP20 (corresponding to epitope length of 20) (see Table 2). This observation raises an obvious question: can we improve the predictive performance of BCPred if we increase the epitope length to beyond 20? To explore this question, we generated five additional *homology-reduced* data sets, BCP22, BCP24, BCB26, BCP28, and BCP30 (corresponding to epitope lengths of 22, 24, 26, 28, and 30, respectively) and compared the performance of BCPred on the resulting data sets using fivefold cross-validation. The performance of BCPred on the five data sets is summarized in Table 4. It is interesting to note that the measured AUC on BCP22

Table 3. Results of Bonferroni adjustments using *p*-value = 0.0042. "$\times$" indicates that the corresponding pair of methods is significantly different

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12
K_1^{spct} (M1)												
K_2^{spct} (M2)	0											
K_3^{spct} (M3)	0	0										
$K_{(3,1)}^{msmtch}$ (M4)	0	×	×									
$K_{(4,1)}^{msmtch}$ (M5)	0	×	×	×								
$K_{(5,1)}^{msmtch}$ (M6)	0	0	0	×	×							
$K_{(5,2)}^{msmtch}$ (M7)	0	×	×	×	×	×						
LA (M8)	×	×	0	×	×	0	×					
$K_{(2,0.5)}^{sub}$ (M9)	0	×	0	×	×	0	×	×				
$K_{(3,0.5)}^{sub}$ (M10)	×	×	×	×	×	×	×	×	×			
BCPred (M11)	×	×	×	×	×	×	×	×	×	×		
RBF (M12)	0	0	0	×	0	0	×	×	0	×	×	
AAP (M13)	×	×	0	×	×	0	×	0	×	0	×	×

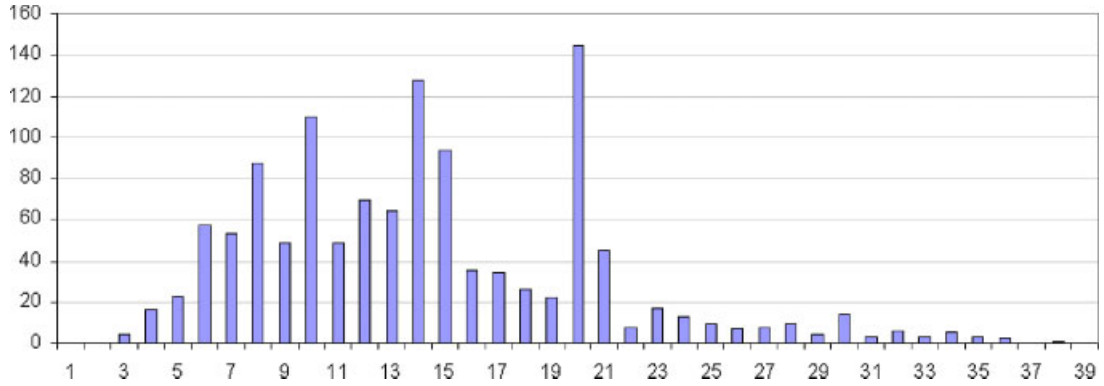

Figure 2. Length distribution of unique linear B-cell epitopes in Bcipep database (Saha *et al.*, 2005). 86.7% of the epitopes are at most 20 amino acids in length.

is 0.788 compared to 0.758 on BCP20. A slightly better AUC, 0.804, was observed on BCP28.

Why does BCPred have a better performance on BCP28 compared with its performance on BCP20? There are at least three possible explanations: (i) BCP28 includes longer segments of epitope sequences of length greater than 20 amino acids in the data set than BCP20; (ii) Some of the hard-to-predict epitopes in BCP20 are eliminated from BCP28 because these epitopes are located very close to the ends of the antigen sequence and so extending these epitopes to 28 amino acids in length by adding an equal number of amino acids from both ends is not possible; (iii) amino acid neighbors of the epitopes carry some useful signal that helps the classifier to better discriminate epitopes from non-epitopes.

To test these hypotheses, we constructed a modified version of BCP20 data set, MBCP20. MBCP20 was derived from BCP28 by trimming 4 amino acids from both ends of each peptide in BCP28. Therefore, BCP28 and MBCP20 can be viewed as two different representations of the same set of epitope/non-epitope data. However, the sequence similarity between any pair of epitopes in BCP28 is guaranteed to be less than 80% but this is not necessarily the case for epitopes in MBCP20. The performance of BCPred on MBCP20 data set is shown in Table 4. The results show that the performance of BCPred on MBCP20, the trimmed version of BCP28, is worse than that on BCP28. This observation provides some evidence against the second of the three possible explanations for observed improvements in performance with epitope length chosen to construct the data sets used to train and test BCPred. It also lends some credence to the suggestion that the amino acid neighbors of the B-cell epitopes may help the classifier to better discriminate between epitopes and non-epitope sequences. As noted earlier, another possibility is that increasing the length results in covering a larger fraction of epitope sequence in the data set in the case of epitope sequences that are longer than 20 amino acid in length (about 13% of the epitopes).

Table 4. Performance of BCPred on *homology-reduced* data sets containing longer epitopes (22–30 residues) and modified BCP20, MBCP20, data set

Data set	ACC(%)	S_n(%)	S_p(%)	CC	AUC
BCP20	67.90	72.61	63.2	0.360	0.758
BCP22	70.91	73.20	68.63	0.419	0.788
BCP24	67.70	79.81	55.59	0.365	0.783
BCP26	70.66	74.18	67.14	0.414	0.796
BCP28	72.06	72.37	71.74	0.441	0.804
BCP30	70.62	68.95	72.29	0.413	0.788
MBCP20	68.45	67.97	68.92	0.369	0.758

Comparing BCPred with existing linear B-cell epitope prediction methods

Although a number of machine learning based methods for predicting linear B-cell epitopes have been proposed (Saha and Raghava, 2006b; Söllner and Mayer, 2006; Chen *et al.*, 2007), little is known about how these methods directly compare with one another due to the lack of published comparisons using standard benchmark data sets. Unfortunately, because the code and precise parameters used to train several of these methods are not available, we were unable to make direct comparisons of these methods using the *homology-reduced* data sets we used in our first set of experiments (summarized in Tables 1 and 2). However, we were able to compare BCPred with our implementation of APP and ABCPred, using the publicly available benchmark data sets (Saha and Raghava, 2006a) that were used to evaluate ABCPred. Because the best reported performance of ABCPred was obtained using a data set of 16-mer peptides, comprising 700 epitopes and 700 non-epitopes peptides, we used the same data set, ABCP16, to compare ABCPred with BCPred and AAP. In addition, a blind test set, SBT16, consisting of 187 epitopes and 200 16-mer non-epitopes, also made available by Saha and Raghava (2006a), was used to compare the three methods.

Table 5 compares the performance of BCPred, AAP, and ABCPred on ABCP16 data set (Saha and Raghava, 2006a), using fivefold cross-validation. In terms of overall accuracy, both BCPred

Table 5. Performance of BCPred, AAP, and ABCpred evaluated on ABCP16 data set using fivefold cross-validation. "—" denotes unavailable information

Method	ACC(%)	S_n(%)	S_p(%)	CC	AUC
BCPred	74.57	70.14	79.00	0.493	0.801
AAP	73.14	50.17	95.57	0.518	0.782
ABCPred	65.93	67.14	64.71	0.319	—

Journal of Molecular Recognition

Table 6. Performance comparison of BCPred, AAP, and ABCPred. The three classifiers were trained using ABCP16 data set and evaluated using SBT16 blind test set

Method	ACC(%)	S_n(%)	S_p(%)	CC	AUC
BCPred	65.89	66.31	65.50	0.318	0.699
AAP	64.60	64.17	65.00	0.292	0.689
ABCPred	66.41	71.66	61.50	-	-

and AAP outperformed ABCPred on this data set, with BCPred showing the best performance (74.57%). Interestingly, the performance of BCPred and AAP on ABCP16 data set was better than their performance on the *homology-reduced* data set used in the first set of experiments described above. The performance of the three classifiers trained on ABCP16 data set, but tested on the blind test set SBT16 is summarized in Table 6. In this case, the performance of ABCPred was slightly better than that of BCPred and APP.

What explains the discrepancy between the performance estimated on ABCP16 data set and the performance on SBT16 blind test set?

Based on the empirical results summarized above, it is natural to ask: How can we explain the differences in relative performance of BCPred and AAP on our *homology-reduced* data sets versus the performance of these methods on ABCP16 data set (Saha and Raghava, 2006b)? How can we explain the observation that BCPred and AAP outperform ABCPred in fivefold cross validation experiments using ABCP16 data set but not on the blind test set, SBT16 data set?

Could the observed differences in relative performance be explained by differences in the two data sets, BCP16 and ABCP16? To explore this possibility, we considered the procedures used to create the data sets. Recall that Saha and Raghava (2006b) started with a data set of 20-mer peptides (after extending the length of shorter B-cell epitopes based on the corresponding antigen sequences). [As noted above, there is a possibility that the resulting data set of 20-mer peptides includes several highly similar peptides (e.g., peptides that differ from each other in only one or two amino acids). More importantly, the 16-mer data set, ABCP16, was derived from the 20-mer data set, ABCP20, by trimming two amino acids from the ends of each 20-mer peptide; as a result, two 20-mers that were not duplicates of each other might yield 16-mers that are highly similar after the ends are trimmed off. In summary, the ABCP20 data set reported by Saha and Raghava (2006b) was constructed from unique epitopes without applying any homology reduction filters. Moreover, the procedure used by Saha and Raghava (2006b) to derive ABCP16 from ABCP20 can be expected to increase the pair-wise similarity between sequences in ABCP16 relative to the pairwise sequence similarity within ABCP20.

Indeed, when we scanned the positive peptides in ABCP16 data set (Saha and Raghava, 2006a) for duplicate peptides, we found 37 cases in which a 16-mer peptide has at least one exact duplicate in the 16-mer data set and several of these have multiple copies in the 16-mer data set (see Table 7). Consequently, fivefold cross validation using ABCP16 data set is likely to yield overly optimistic performance estimates, especially for methods that rely on sequence features such as those identified by the subsequence kernel and AAP.

Table 7. List of 37 duplicated 16-mer peptides and number of occurrence of each (N) in ABCP16 data set (Saha and Raghava, 2006a)

Peptide	N	Peptide	N
RGPGRAFVTIGKIGNM	2	RKRIHIGPGRAFYTTK	3
GPQGLAGQRGIVGLPG	1	KSIRIQRGPGRAFVTI	9
QEVGKAMYAPPISGQI	2	SIRIQRGPGRAFVTIG	3
SEGATPQDLNTMLNTV	1	RQGPKEPFRDYVDRFY	1
LGIWGCSGKLICTTAV	1	AKATYEAALKQYEADL	1
GDRADGQPAGDRADGQ	1	NYNKRKRIHIGPGRAF	2
DGVGAASRDLEKHGAI	1	NNNTRKSITKGPGRVI	1
RLIEDNEYTARQGAKF	1	LQARILAVERYLKDQQ	1
EQELLELDKWASLWNW	2	KMQTLWDEIMDINKRK	1
NVTENFDMWKNDMVEQ	2	TRKSIRIQRGPGRAFV	2
NVTENFNMWKNDMVEQ	1	DPNPQEVVLVNVTENF	1
GIWGCSGKLICTTAVP	1	YLKDQQLLGIWGCSGK	1
QLLGIWGCSGKLICTT	1	TRKSITKGPGRVIYAT	1
QVTPGRGPGRAPCSAG	1	SFNISTSIRGKVQKEY	1
IRIQRGPGRAFVTIGK	1	RPVVSTQLLLNGSLAE	1
RIQRGPGRAFVTIGKI	3	RKDPVVYPWMKKVHVN	1
KRIHIGPGRAFYTTKN	2	RNRWEWRPDFESEKVK	1
AGTVGENVPDDLYIKG	1	FLQIYKQGGFLGLSNI	1
KRKRIHIGPGRAFYTT	4		

To determine exactly how redundant are the positive peptides in ABCP16 data set, we filtered them using an 80% sequence identity cutoff. We found that applying the 80% sequence identity cutoff resulted in the number of positive peptides in the ABCP16 data being reduced from 696 to 532. Thus, 23.5% of the positive peptides in ABCP16 data set have more than 80% sequence identity. This observation leads us to conclude that the observed differences in the performance of BCPred and AAP on the *homology-reduced* data set (BCP16) relative to that on the ABCP16 data set, as well as the results of comparisons of AAP, ABCPred, and BCPred on the blind test set (SBT16), are explained by the presence of a relatively large number of highly similar peptides in ABCP16 data set.

The preceding analysis highlights an important issue in evaluating linear B-cell prediction tools, which, to our knowledge, has not been addressed in previous studies. Previously published linear B-cell epitope prediction methods (Larsen *et al*., 2006; Saha and Raghava, 2006b; Söllner and Mayer, 2006; Chen *et al*., 2007) have been evaluated using data sets of unique epitopes without considering any sequence similarities that may exist among epitopes. In reality, *unique* epitopes may share a high degree of similarity (e.g., a shorter epitope may be included within a longer one, or two epitopes may differ in only one or two amino acids). In this work, we demonstrated that cross-validation performance estimated on such data sets can be overly optimistic. Moreover, such data sets can lead to false conclusions when used to compare different prediction methods. For instance, our comparison of ABCPred, AAP, and BCPred using fivefold cross-validation on ABCP16 data set suggested that AAP and BCPred significantly outperform ABCPred. Such a conclusion may not be valid because evaluation of the three methods on a blind test set,

SBT16, suggests that the three methods are comparable to each other.

BCPREDS web server

We have implemented BCPREDS, an online web server for B-cell epitope prediction, using classifiers trained on the *homology-reduced* data sets of B-cell epitopes developed in this work. The server can be accessed at http://ailab.cs.iastate.edu/bcpreds/.

Because it is often valuable to compare predictions of multiple methods, and consensus predictions are more reliable than individual predictions, the BCPREDS server allows users to choose the method for predicting B-cell epitopes, either BCPred or AAP (and in the future, additional methods). Users provide an antigen sequence and optionally can specify desired epitope length and specificity threshold. Results are returned in several user-friendly formats. In what follows, we illustrate the use of the BCPREDS server in a representative application of B-cell epitope prediction.

Identifying B-cell epitopes in the receptor-binding domain of SARS-CoV spike protein

Since its outbreak in 2002, the development of an effective and safe vaccine against Severe Acute Respiratory Syndrome Coronavirus (SARS-CoV) has become an urgent need for preventing future worldwide outbreak of SARS, a life threatening disease (Drosten *et al*., 2003; Fouchier *et al*., 2003; Ksiazek *et al*., 2003; Peiris *et al*., 2003). Infection by SARS-CoV is initiated by the binding of its spike (S) protein to its functional receptor, angiotensin-converting enzyme 2 (ACE2), which is expressed on the surface of host cells (Dimitrov, 2003; Li *et al*., 2003). The S protein comprises 1255 amino acids and consists of two functional domains: S1 (residues 1–667) and S2 (residues 668–1255) (Wu *et al*., 2004). The S1 domain is responsible for binding to receptors on target cells (Li *et al*., 2003) and the S2 domain contributes to the subsequence fusion between viral envelope and cellular membrane (Beniac *et al*., 2006). In addition, the S2 domain contains two highly conserved heptad repeat (HR) regions, HR1 and HR2 correspond to amino acid residues 915–949 and 1150–1184, respectively (Sainz *et al*., 2005). Several studies reported that the receptor-binding domain (RBD), residues 318–510, is an attractive target for developing SARS-CoV vaccine because blocking the binding of S1 domain to cellular receptors can prevent envelope fusion and virus entry mediated by the S2 domain (Sui *et al*., 2004; Prabakaran *et al*., 2006). Based on these findings, we surveyed the literature to collect previously identified epitopes within the RBD fragment of the SARS-CoV S protein. The collected epitopes are summarized in Table 8. None of these epitopes appears in our training data sets. Because epitope SP3 is included within the epitope (434–467) (GNYNY-KYRYLKHGKLRPFERDISNVPFSPDGKPC) reported by Lien *et al.* (2007), we omitted the longer epitope.

Table 8. B-cell epitopes previously identified in the RBD of SARS-CoV S protein

Epitope	Amino acid sequence	PubMed ID
SP1 (336–352)	SVYAWERKKISNCVADY	16725238
SP2 (424–435)	NTRNIDATSTGN	17055022
SP3 (442–458)	YLKHGKLRPFERDISNV	16725238
SP4 (459–470)	PFSPDGKPCTPP	17055022
SP5 (483–494)	FYTTTGIGYQPY	16337697

We submitted 193 residues comprising the RBD region of SARS-CoV S protein (residues 318–510 according to accession AAT74874) to the BCPREDS, ABCPred, and Bepipred servers. For BCPREDS, we used the default specificity threshold (75%) and set the epitope length to 16 residues. For the other two servers, we used the default settings. Figure 3 shows the BCPred (top) and AAP (bottom) predictions returned by BCPREDS. Four of the B-cell epitopes predicted by BCPred overlap with epitopes that have been identified in the antigenic regions of RBD of SARS-CoV S

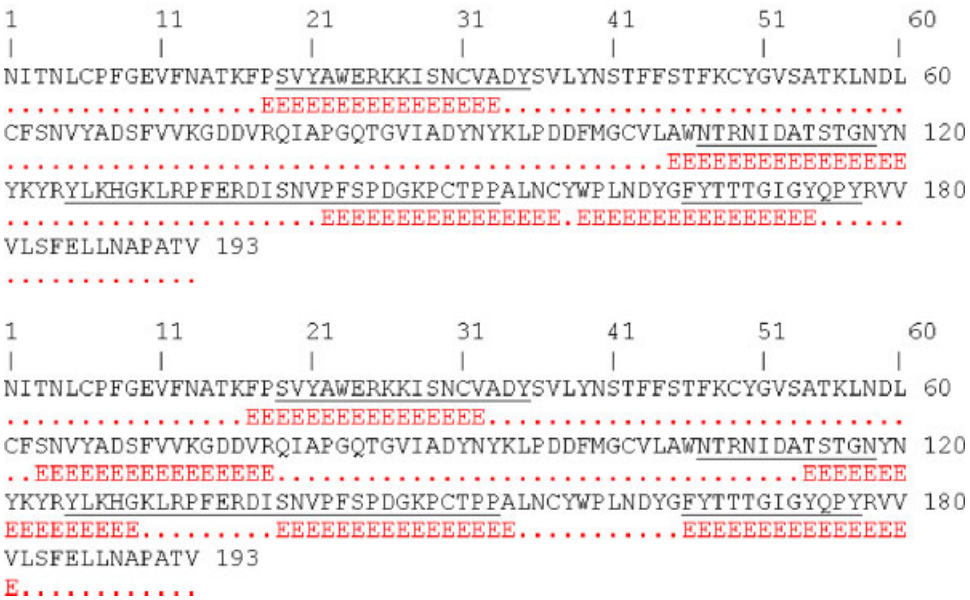

Figure 3. BCPREDS server predictions of epitopes within the RBD of SARS-CoV S protein, made using BCPred (top) and AAP (bottom). Experimentally identified epitopes are underlined. "E" indicates that the corresponding amino acid residue lies in a predicted linear B-cell epitope.

protein through experiments. Three of the five epitopes predicted by AAP have substantial overlap with the SP1, SP2, and SP5 epitopes, and the fourth partially overlaps epitopes SP2 and SP3; the fifth does not overlap with any experimentally reported epitopes. In contrast, the ABCPred server, using default parameters, returned 22 predictions covering almost the entire query sequence. BepiPred returned nine predicted variable-length epitopes, but only three of them are longer than four residues in length. Two out of these four epitopes overlap with experimentally reported epitopes. The complete ABCPred and BepiPred predictions are provided in the Supplementary Materials. In evaluating these results, it is worth noting that a high false positive rate is more problematic than an occasional false negative prediction in the B-cell epitope prediction task (Söllner and Mayer, 2006), because a major goal of B-cell epitope prediction tools is to reduce the time and expense of wet lab experiments.

The B-cell epitopes predicted over the entire SARS-CoV S protein using BCPred is given in Figure S3. Interestingly, the predictions of BCPred over the RBD region are identical regardless of whether the predictions are made over only the RBD sequence fragment or over the entire S protein sequence of SARS-CoV.

DISCUSSION

In this paper, we explored a family of SVM-based machine learning methods for predicting linear B-cell epitopes from primary amino acid sequence. We explored four string kernel methods and compared them to the widely used RBF kernel. Our results demonstrate the usefulness of the four string kernels in predicting linear B-cell epitopes, with the subsequence kernel showing a superior performance over other kernels. In addition, we observed that the subsequence kernel is less sensitive to the choice of the parameter k than the k-spectrum and (k,m)-mismatch kernels. Our experiments using fivefold cross-validation on a *homology-reduced* data set of 701 linear B-cell epitopes and 701 non-epitopes demonstrated that the subsequence kernel significantly outperforms other kernel methods in addition to APP method (Chen *et al*., 2007). To the best of our knowledge, the subsequence kernel (Lodhi *et al*., 2002) although previously used in text classification and natural language processing applications, have not been widely exploited in the context of macromolecular sequence classification tasks. The superior performance of the subsequence kernel on B-cell epitope prediction task suggests that it might find use in other related macromolecular sequence classification tasks, e.g., MHC binding peptide prediction (Salomon and Flower, 2006; Cui *et al*., 2006) and protein subcellular localization prediction (Bulashevska and Eils, 2006; Yu *et al*., 2006).

One of the challenges for developing reliable linear B-cell epitope predictors is how to deal with the large variability in the length of the epitopes which ranges from 3 to 30 amino acids in length. Many standard machine learning methods require training and testing the classifier using sequences of fixed length. For example, the AAP (Chen *et al*., 2007) method was evaluated on a data set where the length of the input sequences was fixed to 20 amino acids. Saha and Raghava (2006b) experimented with data sets consisting of peptide sequences of length 20 and shorter, and reported optimal performance of ABCPred classifier on a data set consisting of 16-mer peptides. In BepiPred (Larsen *et al*., 2006) and propensity scale based methods (Karplus and Schulz. 1985; Emini *et al*., 1985; Parker *et al*., 1986; Pellequer *et al*., 1991; Pellequer *et al*., 1993; Saha and Raghava, 2004), the training examples are windows of five or seven amino acids labeled according to whether the amino acid at the center of the window is included in a linear B-cell epitope or not. Here, we evaluated BCPred on several data sets consisting of fixed length peptides with lengths ranging from 12 to 30 amino acids with incremental step equal to 2. Our results suggest that amino acid neighbors of the B-cell epitope carry some useful information that can help the classifier to discriminate better between epitopes and non-epitopes. This is especially interesting in light of the observation that adding a single amino acid to a linear B-cell epitope may affect binding to the antibody.

A similar situation arises in predicting the major histocompatibility complex class II (MHC-II) binding peptides. The length of MHC-II binding peptides typically varies from 11 to 30 amino acids in length. Most of the currently available MHC-II binding peptide prediction methods focus on identifying a putative 9-mer binding core region. Therefore, classifiers are trained using 9-mer peptides instead of variable length ones. Recently, two methods (Cui *et al*., 2006; Salomon and Flower, 2006) for predicting variable length MHC-II peptides have been proposed. Both methods use the entire sequences of MHC-II binding peptides (as opposed to only the 9-mer cores) for training MHC-II binding peptide predictors. The first method (Cui *et al*., 2006) maps a variable length peptide into a fixed length feature vector obtained from sequence-derived structural and physicochemical properties of the peptide. The second method (Salomon and Flower, 2006) uses the local alignment (LA) kernel that we used in this study. It would be interesting to apply these methods to the problem of learning to identify variable length linear B-cell epitopes. Our ongoing work aims at exploring the application of string kernels for learning from flexible length linear B-cell epitopes.

In light of the significant room for improvement in performance of B-cell epitope prediction methods reported in the literature, it is important to understand the strengths and limitations of different methods through direct comparisons on standard benchmark data sets. Hence, we compared the BCPred method using the subsequence kernel-based SVM developed in this paper with two published methods: AAP (Chen *et al*., 2007) and ABCPred (Saha and Raghava, 2006b). In our experiments using the Saha and Raghava (2006b) 16-mer peptide data set (containing approximately 700 B-cell epitopes and 700 non-epitope peptides) on which ABCPred had the best reported performance of 66%, both BCPred and AAP outperformed ABCPred, based on fivefold cross-validation. However, when the classifiers were tested on a separate *blind* test set instead, no significant difference was observed in their performance. Careful examination of the ABCPred 16-mer data set revealed that the data set has a high degree of sequence redundancy among the epitope peptides, leading to overly optimistic estimates of performance in some cases.

Our demonstration that the only publicly available data set of linear B-cell epitopes (Saha and Raghava, 2006a) is, in fact, highly redundant (with almost 25% of individual 16-mer epitopes having at least one other epitope with $>80\%$ sequence identity) is significant. We showed that the redundancy in such a data set can be reflected in overly-optimistic performance estimates, especially for certain types of machine learning classifiers. Consequently, using such a data set can also lead to false conclusions when directly comparing different prediction

methods. Therefore, it is very important to evaluate and compare different linear B-cell epitope prediction methods on data sets that are truly *non-redundant* or *homology-reduced* with respect to their constituent epitope sequences, i.e., in which the level of pair-wise sequence identity shared between individual epitopes is known. Towards this goal, we have made our *homology-reduced* data set of linear B-cell epitopes (with $<$80% sequence identity) publicly available as a benchmarking data set for comparing existing and future linear B-cell epitope prediction methods.

Based on the results of this study, we developed BCPREDS, an online web server for predicting linear B-cell epitopes using either the BCPred method, which implements the subsequence kernel introduced in this paper, or the AAP method of Chen *et al.* (2007). A case study in which BCPREDS was used to predict linear B-cell epitopes in the RBD of the SARS-CoV S protein demonstrates the potential value of this server in guiding clinical investigations.

Work in progress is aimed at further development and empirical comparisons of different methods for B-cell epitope prediction, in particular, addressing the more challenging problem of predicting discontinuous or conformational B-cell epitopes.

Acknowledgements

We thank the anonymous reviewers for their comments and suggestions, Dr. Janez Demšar for discussing the applicability of non-parametric tests, Dr. Douglas Bonett for suggesting the Bonferroni adjustment test. This research was supported in part by a doctoral fellowship from the Egyptian Government to Yasser El-Manzalawy and a grant from the National Institutes of Health (GM066387) to Vasant Honavar and Drena Dobbs.

REFERENCES

Alix A. 1999. Predictive estimation of protein linear epitopes by using the program PEOPLE. *Vaccine* **18**: 311–314.

Bairoch A, Apweiler R. 2000. The SWISS-PROT protein sequence database and its supplement TrEMBL in 2000. *Nucleic Acids Res.* **28**: 45–48.

Baldi P, Brunak S, Chauvin Y, Andersen C, Nielsen H. 2000. Assessing the accuracy of prediction algorithms for classification: An overview. *Bioinformatics* **16**: 412–424.

Barlow D, Edwards M, Thornton J. 1986. Continuous and discontinuous protein antigenic determinants. *Nature* **322**: 747–748.

Beniac D, Andonov A, Grudeski E, Booth T. 2006. Architecture of the SARS coronavirus prefusion spike. *Nat. Struct. Mol. Biol.* **13**: 751–752.

Björklund Å, Soeria-Atmadja D, Zorzet A, Hammerling U, Gustafsson M. 2005. Supervised identification of allergen-representative peptides for in silico detection of potentially allergenic proteins. *Bioinformatics* **21**: 39–50.

Blythe M, Flower D. 2005. Benchmarking B cell epitope prediction: underperformance of existing methods. *Prot. Sci.* **14**: 246–248.

Bulashevska A, Eils R. 2006. Predicting protein subcellular locations using hierarchical ensemble of Bayesian classifiers based on Markov chains. *BMC Bioinform.* **7**: 298.

Chen J, Liu H, Yang J, Chou K. 2007. Prediction of linear B-cell epitopes using amino acid pair antigenicity scale. *Amino Acids* **33**: 423–428.

Clark A, Florencio C, Watkins C, Serayet M. 2006. Planar languages and learnability. International Colloquium on Grammatical Inference (ICGI06). Lihue, Hawaii.

Cui J, Han L, Lin H, Tan Z, Jiang L, Cao Z, Chen Y. 2006. MHC-BPS: MHC-binder prediction server for identifying peptides of flexible lengths from sequence-derived physicochemical properties. *Immunogenetics* **58**: 607–613.

Demšar J. 2006. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **7**: 1–30.

Dimitrov D. 2003. The secret life of ACE2 as a receptor for the SARS virus. *Cell* **115**: 652–653.

Drosten C, Gunther S, Preiser W, van der Werf S, Brodt H, Becker S, Rabenau H, Panning M, Kolesnikova L, Fouchier R, Berger A, Burguiere A, Cinatl J, Eickmann M, Escriou N, Grywna K, Kramme S, Manuguerra J, Muller S, Rickerts V, Sturmer M, Vieth S, Klenk H, Osterhaus ADME, Schmitz H, Doerr HW. 2003. Identification of a novel coronavirus in patients with severe acute respiratory syndrome. *N. Engl. J. Med. V* **348**: 1967–1976.

Emini E, Hughes J, Perlow D, Boger J. 1985. Induction of hepatitis A virus-neutralizing antibody by a virus-specific synthetic peptide. *J. Virol.* **55**: 836–839.

Fisher R. 1973. Statistical Methods and Scientific Inference. Hafner Press: New York.

Flower D. 2007. Immunoinformatics: Predicting Immunogenicity in silico, 1st Ed. Humana: Totowa NJ.

Fouchier R, Kuiken T, Schutten M, van Amerongen G, van Doornum G, van den Hoogen B, Peiris M, Lim W, Stöhr K, Osterhaus A. 2003. Koch's postulates fulfilled for SARS virus. *Nature* **423**: 240.

Friedman M. 1940. A comparison of alternative tests of significance for the problem of m rankings. *Ann. Math. Stat.* **11**: 86–92.

Greenbaum J, Andersen P, Blythe M, Bui H, Cachau R, Crowe J, Davies M, Kolaskar A, Lund O, Morrison S, Mumey B, Ofran Y, Pellequer J, Pinilla C, Ponomarenko JV, Raghava GPS, van Regenmortel MHV, Roggen EL, Sette A, Schlessinger A, Sollner J, Zand M, Peters B. 2007. Towards a consensus on datasets and evaluation metrics for developing B-cell epitope prediction tools. *J. Mol. Recognit.* **20**: 75–82.

Haussler D. 1999. Convolution kernels on discrete structures. UC Santa Cruz Technical Report UCS-CRL-99–10.

Karplus P, Schulz G. 1985. Prediction of chain flexibility in proteins: a tool for the selection of peptide antigen. *Naturwiss.* **72**: 21–213.

Ksiazek T, Erdman D, Goldsmith C, Zaki S, Peret T, Emery S, Tong S, Urbani C, Comer J, Lim W, Rollin PE, Dowell SF, Ling A, Humphrey CD, Shieh W, Guarner J, Paddock CD, Rota P, Fields B, DeRisi J, Yang J, Cox N, Hughes JM, Le Duc JW, Bellini WJ, Anderson LJ, the SARS Working Group. 2003. A novel coronavirus associated with severe acute respiratory syndrome. *N. Engl. J. Med.* **348**: 1953–1966.

Langeveld J, Martinez Torrecuadrada J, Boshuizen R, Meloen R, Ignacio C. 2001. Characterisation of a protective linear B cell epitope against feline parvoviruses. *Vaccine* **19**: 2352–2360.

Larsen J, Lund O, Nielsen M. 2006. Improved method for predicting linear B-cell epitopes. *Immun. Res.* **2**: 2.

Leslie C, Eskin E, Cohen A, Weston J, Noble W. 2004. Mismatch string kernels for discriminative protein classification. *Bioinformatics* **20**: 467–476.

Leslie C, Eskin E, Noble W. 2002. The spectrum kernel: a string kernel for SVM protein classification. *Proc. Pacific Sympos. Biocomput.* **7**: 566–575.

Li W, Jaroszewski L, Godzik A. 2002. Tolerating some redundancy significantly speeds up clustering of large protein databases. *Bioinformatics* **18**: 77–82.

Li W, Moore M, Vasilieva N, Sui J, Wong S, Berne M, Somasundaran M, Sullivan J, Luzuriaga K, Greenough TC, Choe H, Farzan M. 2003. Angiotensin-converting enzyme 2 is a functional receptor for the SARS coronavirus. *Nature* **426**: 450–454.

Lien S, Shih Y, Chen H, Tsai J, Leng C, Lin M, Lin L, Liu H, Chou A, Chang Y, Chen Y, Chong P, Liu S. 2007. Identification of synthetic vaccine candidates against SARS CoV infection. *Biochem. Biophys. Res. Commun.* **358**: 716–721.

Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins C. 2002. Text classification using string kernels. *J. Mach. Learn. Res.* **2**: 419–444.

Neter J, Wasserman J, Kutner M. 1985. Applied Linear Statistical Models, 2nd Ed. Irwin: Homewood, IL.

Odorico M, Pellequer J. 2003. BEPITOPE: predicting the location of continuous epitopes and patterns in proteins. *J. Mol. Recognit.* **16**: 20–22.

Parker JM, Guo D, Hodges RS. 1986. New hydrophilicity scale derived from high-performance liquid chromatography peptide retention data: Correlation of predicted surface residues with antigenicity and x-ray-derived accessible sites. *Biochemistry* **25**: 5425–5432.

Peiris J, Lai S, Poon L, Guan Y, Yam L, Lim W, Nicholls J, Yee W, Yan W, Cheung M, Cheng V, Chan K, Tsang D, Tung R, Ng T, Yuen K, members of the SARS study group. 2003. Coronavirus as a possible cause of severe acute respiratory syndrome. *The Lancet* **361**: 1319–1325.

Pellequer J, Westhof E. 1993. PREDITOP: a program for antigenicity prediction. *J. Mol. Graph.* **11**: 204–210.

Pellequer J, Westhof E, Van Regenmortel M. 1991. Predicting location of continuous epitopes in proteins from their primary structures. *Meth. Enzymol.* **203**: 176–201.

Pellequer J, Westhof E, Van Regenmortel M. 1993. Correlation between the location of antigenic sites and the prediction of turns in proteins. *Immunol. Lett.* **36**: 83–99.

Pier G, Lyczak J, Wetzler L. 2004. Immunology, Infection, and Immunity, 1st Ed. ASM Press: PL Washington.

Platt J. 1998. Fast Training of Support Vector Machines using Sequential Minimal Optimization. MIT Press: Cambridge, MA.

Prabakaran P, Gan J, Feng Y, Zhu Z, Choudhry V, Xiao X, Ji X, Dimitrov D. 2006. Structure of severe acute respiratory syndrome coronavirus receptor-binding domain complexed with neutralizing antibody. *J. Biol. Chem.* **281**: 15829.

Rangwala H, DeRonne K, Karypis G. 2006. Protein Structure Prediction using String Kernels. Defense Technical Information Center.

Saha S, Bhasin M, Raghava GP. 2005. Bcipep: a database of B-cell epitopes. *BMC Genomics* **6**: 79.

Saha S, Raghava GP. 2004. BcePred: prediction of continuous B-cell epitopes in antigenic sequences using physico-chemical properties. *Lect. Notes Comput. Sci.* **3239**: 197–204.

Saha S, Raghava G. 2006a. ABCPred benchmarking datasets. Available at http://www.imtech.res.in/raghava/abcpred/dataset.html

Saha S, Raghava G. 2006b. Prediction of continuous B-cell epitopes in an antigen using recurrent neural network. *Proteins* **65**: 40–48.

Saigo H, Vert J, Ueda N, Akutsu T. 2004. Protein homology detection using string alignment kernels. *Bioinformatics* **20**: 1682–1689.

Sainz JB, Rausch J, Gallaher W, Garry R, Wimley W. 2005. Identification and characterization of the putative fusion peptide of the severe acute respiratory syndrome-associated coronavirus spike protein. *Virology* **79**: 7195–7206.

Salomon J, Flower D. 2006. Predicting class II MHC-peptide binding: a kernel based approach using similarity scores. *BMC Bioinform.* **7**: 501.

Seewald A, Kleedorfer F. 2005. Lambda pruning: an approximation of the string subsequence kernel. Technical Report, Osterreichisches Forschungsinstitut fur Artificial Intelligence, Wien, TR-2005–13.

Söllner J, Mayer B. 2006. Machine learning approaches for prediction of linear B-cell epitopes on proteins. *J. Mol. Recognit.* **19**: 200–208.

Sui J, Li W, Murakami A, Tamin A, Matthews L, Wong S, Moore M, Tallarico A, Olurinde M, Choe H, Anderson LJ, Bellini WJ, Farzan M, Marasco WA. 2004. Potent neutralization of severe acute respiratory syndrome (SARS) coronavirus by a human mAb to S1 protein that blocks receptor association. *Proc. Natl Acad. Sci.* **101**: 2536–2541.

Vapnik V. 2000. The Nature of Statistical Learning Theory, 2nd Ed. Springer-Verlag New York, Inc. New York, NY, USA.

Walter G. 1986. Production and use of antibodies against synthetic peptides. *J. Immunol. Meth.* **88**: 149–161.

Witten IH, Frank E. 2005. Data Mining: Practical Machine Learning Tools and Techniques, 2nd Ed. Morgan Kaufmann. San Francisco, USA.

Wu F, Olson B, Dobbs D, Honavar V. 2006. Comparing kernels for predicting protein binding sites from amino acid sequence. International Joint Conference on Neural Networks (IJCNN06) 1612–1616.

Wu X, Shang B, Yang R, Yu H, Hai Z, Shen X, Ji Y, Lin Y, Di Wu Y, Lin G, Tian L, Gan XQ, Yang S, Jiang WH, Dai EH, Wang XY, Jiang HL, Xie YH, Zhu XL, Pei G, Li L, Wu JR, Sun B. 2004. The spike protein of severe acute respiratory syndrome (SARS) is cleaved in virus infected Vero-E6 cells. *Cell Res.* **14**: 400–406.

Yu C, Chen Y, Lu C, Hwang J. 2006. Prediction of protein subcellular localization. *Proteins* **64**: 643–651.

Zaki N, Deris S, Illias R. 2005. Application of string kernels in protein sequence classification. *Appl. Bioinform.* **4**: 45–52.

