A Sparse Bayesian Position Weighted Bio-Kernel Network

David C. Trudgian and Zheng Rong Yang

School of Engineering, Computing, and Mathematics,
University of Exeter, Exeter, EX4 4QF, UK.
{d.c.trudgian,z.r.yang}@ex.ac.uk

Abstract. The Bio-Basis Function Neural Network (BBFNN) is a successful neural network architecture for peptide classification. However, the selection of a subset of peptides for a parsimonious network structure is always a difficult process. We present a Sparse Bayesian Bio-Kernel Network in which a minimal set of representative peptides can be selected automatically. We also introduce per-residue weighting to the Bio-Kernel to improve accuracy and identify patterns for biological activity. The new network is shown to outperform the original BBFNN on various datasets, covering different biological activities such as as enzymatic and post-translational-modification, and generates simple, interpretable models.

1 Introduction

The Bio-Basis Function Neural Network (BBFNN) [1] is a novel neural architecture that accepts peptide data as input, without requiring the peptides to be numerically encoded. A number of bio-basis functions, which make use of sequence similarity scoring, are used to transform the non-numerical, non-linear input space, with a linear classification stage then being used to make predictions. The BBFNN can be used in both regression and classification modes. Variants of the network are available for use with fixed length peptides, or variable length input sequences, through the use of differing basis functions. This paper is concerned with the classification of fixed length peptides, a common bioinformatics problem.

In the context of peptide classification the BBFNN has been applied to the prediction of cleavage sites, post translational modifications, and aspects of protein structure. Whilst it has been able to produce high quality results for these problems, some aspects of the network merit investigation and improvement. Previous work has investigated the selection of Substitution Matrices used in the Bio-Basis Function, and the production of problem-specific matrices [2]. This paper is concerned with two topics, the selection of the most suitable support peptides from the training data, and the lack of position weightings.

1.1 Selecting Support Peptides

In the traditional BBFNN model a user must specify the number of basis neurons to be used. Each of these neurons requires a support peptide taken from the training data. Input peptides are compared to the support peptides using the bio-basis function, and the outputs of the function used in the linear classification step of the network. It makes intuitive sense that optimum classification performance would be achieved by using a set of support peptides from the training data which are highly representative of their respective classes. However, the standard BBFNN offers no automatic means of obtaining this set.

The user of the original BBFNN software may manually select a number of support peptides, use a given number of randomly selected peptides, or choose to use all peptides as support peptides. The first case requires careful analysis of the data, and one must be careful not to over-fit the support peptides to the data in hand. The second case is the most frequently used, but can result in relatively large variations in performance depending on the mix of basis peptides selected; selecting the best number of basis neurons to use is also a problem. The final case results in a network that is likely to be over-fitted, and will take a large amount of time to train due to the high complexity. Within all of these methods it has also been common to vary the balance of positive and negative support peptides to try and address class imbalance in the dataset.

It is clear that a method of selecting a set of support peptides that gives good accuracy, is resistant to over-fitting, and results in a parsimonious network that is quick to use, would be of great benefit. We propose using sparse Bayesian learning to accomplish this goal [3].

1.2 Position Weighting

It is reasonable to expect that, in most cases, certain residues will be more important than others in determining the class of a peptide, due to the differing effects on protein structure from each position. Since biological experience shows that all residue positions are not equal in their ability to determine the class of a peptide (e.g. cleavable / non-cleavable), it is reasonable to expect that peptide classification methods that consider specific positions will produce better results than those which do not. Where predictions are made by concentrating on the most informative residues, they are likely to be less affected by noise from unimportant variations at other residue positions. It is therefore reasonable to expect that a position specific method would offer more robust results. However, this is subject to the method not having been over-fitted to certain residues.

Position specific methods in machine learning seek to accommodate this situation, and examine not just the overall composition of a peptide, but consider the amino acids in each position separately. The standard BBFNN is not a position specific method, since the bio-basis function sums the similarity scores for all residue positions without weightings. Methods which use the sparse orthonormal encoding [4] are position specific, each residue has corresponding input nodes and weights. We propose introducing a per-residue weighting to the basis function address this issue.

1.3 Biological Interpretation

The two proposed improvements to the BBFNN are not motivated only by a wish to increase performance, but also to increase the ease with which trained models can be interpreted. Whilst neural network models have been shown to be able to make accurate predictions on biological problems, they are often criticised for being a black box, from which it is hard to extract knowledge. The original BBFNN has a simpler structure than that of multi layer perceptrons, having a single weight layer, and using support peptides and similarity values which can be examined easily. However, by selecting an arbitrary number of support peptides we risk over complicating the model, or excluding potentially interesting support peptides. A method which allows a small model with a set of highly relevant support peptides selected will allow for easier analysis.

By introducing per-residue weights to the basis function we hope additional information useful for biological interpretation will be obtained. On problems where certain positions are known to be more important we hope that the residue weights will allow greater prediction accuracy. On problems where there are no general motifs, we hope that the residue weight information will be useful when interpreting the model, allowing more distinct rules to be identified.

2 Method

Sparse Bayesian Learning, as discussed in [3], is a method to find sparse solutions to models with linearly combined parameters. The BBFNN takes this linear form, where N is the number of basis neurons, w_i is the network weight associated with basis n, y is the model output, and $\Phi(\mathbf{x}, \mathbf{z_i})$ is the value of the basis function applied to input vector $\mathbf{x}$, using support peptide $\mathbf{z_i}$.

$$y = \sum_{i=1}^{N} w_i \Phi(\mathbf{x}, \mathbf{z_i})$$

Since there is a linear combination of our parameters w_n, we can use the method to find a sparse model, where the majority of network weights are close to zero, and can be zeroed, and therefore few basis neurons are required. To carry out sparse Bayesian learning we begin with a network consisting of all possible basis functions, i.e. N is the size of our training dataset. During the learning process network weights, and basis neurons, will be removed. This approach is equivalent to a relevance vector machine (RVM) [3], using the bio-basis function as its kernel function.

We will use the Bernoulli likelihood function, and apply a sigmoid function to the model output y, as is appropriate for two class problems. If $\mathbf{t}$ is the target vector, and l is the number of data points, then the negative log likelihood function of the model is:

$$\mathcal{L} = -\sum_{n=1}^{\ell} \{t_n \log y_n + (1 - t_n) \log(1 - y_n)\}$$

As we will be introducing residue weights which will be learnt using the Bayesian method, we choose to use a kernel function which does not include the normalisation terms, or the exponential operator. The original Bio-Basis Function, given in [1], is such that introducing sparse Bayesian position weighting would generate complicated second derivatives which are inconvenient to work with, and cause the Hessian matrix to be extremely costly to compute. The simplified kernel function, including residue weighting is:

$$\Phi_{ni} = \sum_{d=1}^{D} \theta_d \mathcal{M}_{nid}$$

where θ_d is the weighting for the d^{th} residue in a peptide, D is the number of residues in the peptide, and $\mathcal{M}_{nid}$ is the similarity matrix score between the d^{th} residues of the n^{th} support peptide and i^{th} input peptide. Note that $\theta_d = 1$ when residue weights are not in use.

2.1 Optimisation of Network Weights

For the network weights, a standard Gaussian prior $w_n \sim \mathcal{G}(0, \alpha_n^{w-1})$ is used. The Laplace approximation procedure in [5] is also applied. A Newton-Raphson method optimisation is performed to find the most probable weight vector $\hat{\boldsymbol{w}}$ for the current hyper parameter values $\boldsymbol{\alpha}^w$.

The Hessian matrix, used in the Newton's method optimiser is:

$$H^w = \mathbf{A} + \mathbf{\Phi} \mathbf{B} \mathbf{\Phi}^T$$

where $\mathbf{A}$ is a diagonal matrix of the hyper-parameters $\boldsymbol{\alpha}_w$, and B is an $l \times l$ diagonal matrix with values $y_n(1 - y_n)$.

This Hessian can be negated and inverted to give the covariance matrix:

$$\Sigma^w = (\mathbf{A} + \mathbf{\Phi} \mathbf{B} \mathbf{\Phi}^T)^{-1}$$

for a Gaussian approximation to the hyper-parameter posterior, with mean $\hat{\boldsymbol{w}}$.

We will consider our hyper-parameters to be uniformly distributed and therefore only the marginal likelihood must be maximised in order to find $\boldsymbol{\alpha}_w$. We integrate over the negative log of the marginal likelihood with respect to $\mathbf{w}$, and then differentiate with respect to $\boldsymbol{\alpha}_w$ as in [5] to give the update equation:

$$\alpha_i^w = \frac{1 - \alpha_i^w \Sigma_{ii}^w}{\hat{w}_i^2}$$

The training process beings with $\mathbf{w} = 0$, and $\boldsymbol{\alpha}_w = 0.1$. An inner loop implements the Newton method search for the most probable weights given α_w. Once this search has converged, the hyper-parameters are updated in the outer loop. Due to the nature of the sparse Bayesian process, a large number of hyper-parameters will tend to infinity, and their corresponding network weights tend to zero. We prune a weight w_i and its corresponding basis function from the network when $\alpha_i^w > 1 \times 10^{10}$. We continue to update $\mathbf{w}$ and $\boldsymbol{\alpha}_w$ until convergence.

2.2 Optimising Residue Weights

To introduce residue weights we add an additional layer, outside of the hyper-parameter loop to update $\boldsymbol{\theta}$. Note that $\boldsymbol{\theta}$ is a kernel parameter, whereas $\mathbf{w}$ is a network parameter. Again we use the sparse Bayesian approach, resulting in inner and outer loops for optimisation of $\boldsymbol{\theta}$, and the hyper-parameter vector $\boldsymbol{\alpha}^\theta$ respectively. These loops sit outside of those for optimisation of the network weights.

Within the inner loop, the most probable residue weights $\hat{\boldsymbol{\theta}}$ given $\boldsymbol{\alpha}^\theta$ are again obtained using a Newton's method optimiser. The first derivative of the marginal likelihood $p(\theta|\mathcal{D})$ with respect to θ_d is $-\mathbf{Ze}+\mathbf{A}\boldsymbol{\theta}$, where $\mathbf{A} = \mathrm{diag}(\boldsymbol{\alpha}^\theta)$, $\mathbf{Z} = (\mathbf{M}_1\mathbf{w}, \mathbf{M}_2\mathbf{w}, \cdots, \mathbf{M}_\ell\mathbf{w})$ and

$$\mathbf{M}_n = \begin{pmatrix} \mathcal{M}_{n11} & \mathcal{M}_{n12} & \cdots & \mathcal{M}_{n1\ell} \\ \mathcal{M}_{n21} & \mathcal{M}_{n22} & \cdots & \mathcal{M}_{n2\ell} \\ \vdots & \vdots & \vdots & \vdots \\ \mathcal{M}_{nD1} & \mathcal{M}_{nD2} & \cdots & \mathcal{M}_{nD\ell} \end{pmatrix}$$

The Hessian is then:

$$H^\theta = \mathbf{A} + \mathbf{ZBZ}^T$$

$\Delta\theta$ for the Newton-Raphson optimisation in the inner loop is:

$$\Delta\theta = -(\mathbf{Z}^T\mathbf{BZ} + \mathbf{A})^{-1}(\mathbf{A}\theta - \mathbf{Ze})$$

The covariance matrix for the Gaussian approximation is:

$$\Sigma^\theta = (\mathbf{A} + \mathbf{ZBZ}^T)^{-1}$$

The marginal likelihood maximisation procedure is applied as previously, to give the hyper-parameter update equation:

$$\alpha_\theta = \frac{1 - \alpha_\theta \Sigma^\theta_{ii}}{\hat{\theta}_i^{\,2}}$$

Initially $\boldsymbol{\theta} = 1$, with $\boldsymbol{\alpha}^\theta = 0.1$. At each iteration of the $\boldsymbol{\theta}$ loop, the Φ matrix is recalculated, as with different residue weights the kernel function scores are altered. The network weights $\mathbf{w}$, and weight hyper-parameters $\boldsymbol{\alpha}^w$ are updated for each change in $\boldsymbol{\theta}$. No pruning of residue weights is implemented. In practice the hyper-parameters only approach values that would result in pruning on rare occasions.

3 Results & Discussion

3.1 Datasets

We will use three datasets to compare the performance of the new models with the original BBFNN:

GAL - Glycoprotein Linkage Sites. Glycoproteins are an important subset of proteins with a high level of potential pharmacological significance. Carbohydrate groups attached to glycoproteins affect the solubility and thermal stability and are implicated in important biological functions such as controlling uptake of glycoproteins into cells. Chou et al. [6] presented a dataset of 302 9-residue peptides, of which 190 are linkage sites and 112 non-linkage sites.

HIV - HIV-1 Protease Cleavage Sites. During the life-cycle of HIV, precursor polyproteins are cleaved by HIV protease. Disruption of cleavage ability causes non-infectious, imperfect virus replication and is therefore a promising target for anti AIDS drugs. The dataset presented in [7] consists of 8-residue peptides from HIV protease marked as cleavable or non-cleavable. There are 362 peptides of which 114 are positive, cleavage sites and 248 are negative, non cleavable sites.

TCL - T-Cell Epitopes. T-cells are a critical part of the immune response to viral infection. Epitopes are sites on viral proteins that are recognised and bound by the T-cell receptor. The TCL dataset consists of 202 10-residue peptides of which 36 are positive T-cell epitope peptides, the remaining 167 are non-epitope peptides. This data was presented in [8].

3.2 Results

A 5-fold cross validation experiment was used to compare the Sparse Bayesian Bio-kernel Network (SBBKN) with, and without residue weighting, to the original BBFNN. Data was randomised and split into five equal folds. Four folds are used to train a classifier, with the remaining fold then being used to test the model. Each fold is used once for testing, and four times as part of the training set. The entire procedure was repeated twenty times to allow means and standard deviations to be taken for the test statistics. Experiments using the original BBFNN were carried out using 20 bio-basis neurons, with the basis functions randomly selected. The basis functions were not manually selected or balanced as comparison is being made with a new method which does not require manual intervention.

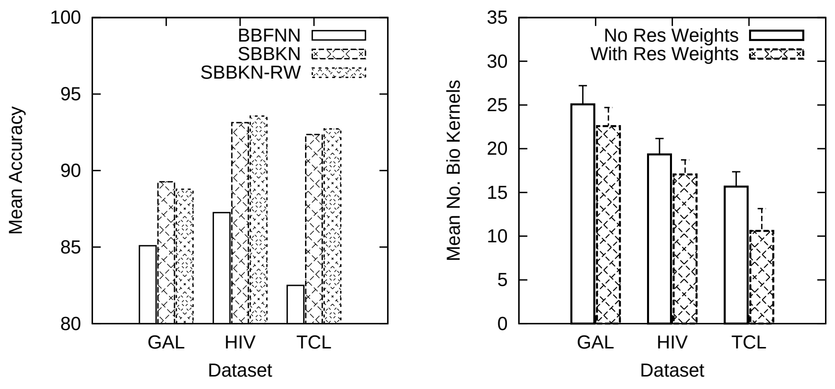
Table 3.2 gives the mean values, and standard deviations in italics, for the test statistics. ACC is the total accuracy, MCC is the value of the Matthew's correlation co-efficient, and AUR is the area under the ROC curve for the model [9], calculated using a 1000 step trapezium method numerical integration.

It can be seen from the ACC column in the table, and more clearly from figure 1(a), that the new techniques outperform the original BBFNN in terms of prediction accuracy on all of the datasets. Standard deviations are slightly larger with the new techniques. The new techniques also outperform the original BBFNN in terms of the Matthews Correlation Coefficient, and mean area under the ROC curves. In the GAL and HIV cases the true negative and positive fractions were both higher than for the original BBFNN. However, on the TCL dataset, which has few positive cases, large improvements in total accuracy and the true negative fraction are accompanied by a smaller decrease in the true positive fraction.

Data	**Method**	**ACC**	**MCC**	**AUR**
GAL	BBFNN	85.09 *(5.76)*	0.69 *(0.12)*	0.92 *(0.05)*
	SBBKN	89.27 *(8.08)*	0.77 *(0.17)*	0.94 *(0.05)*
	SBBKN-RW	88.78 *(8.27)*	0.76 *(0.18)*	0.94 *(0.06)*
HIV	BBFNN	87.25 *(4.39)*	0.70 *(0.10)*	0.92 *(0.03)*
	SBBKN	93.13 *(5.45)*	0.84 *(0.13)*	0.97 *(0.03)*
	SBBKN-RW	93.56 *(5.46)*	0.85 *(0.13)*	0.97 *(0.03)*
TCL	BBFNN	82.50 *(6.22)*	0.54 *(0.15)*	0.90 *(0.05)*
	SBBKN	92.36 *(6.28)*	0.74 *(0.21)*	0.95 *(0.06)*
	SBBKN-RW	92.72 *(5.11)*	0.75 *(0.17)*	0.96 *(0.05)*

Table 1. Cross Validation Results - Mean & *(Standard Deviation)*

The SBBKN with residue weightings slightly outperforms that without weightings on the HIV and T-Cell data, but falls behind on the glycoprotein linkage data. The small differences suggest that there is little to choose between the methods, except the increased computational cost of residue weightings. However, analysis of the models shows a difference in the number of kernels used to achieve the same performance. Also, the residue weight information may be of biological significance, and therefore useful in further interpretation.



(a) Mean Accuracies (b) Model Size

Fig. 1. Accuracy comparison & model size.

The computational cost of the new methods is an increase on that of the standard BBFNN, but is not excessive. On a PC containing an Intel Core 2 Duo processor at 1.7Ghz, the SBBKN without residue weighting takes, on average, 96 seconds to train on 288 peptides from the HIV dataset. With residue weighting the average is 145 seconds. Residue weighting adds relatively little extra computation time. The majority of kernels will be pruned in the first update cycle, leaving only a small network for the majority of the residue weight update calculations.

3.3 Model Analysis

Having considered raw performance statistics we will examine the models that are produced by the training process. Although it was shown that the performance of the weighted and non-weighted SBBKNs was similar, the weighted SBBKN uses fewer kernels on each problem. Figure 1(b) shows the mean, and standard deviation, of the number of bio-kernels used for each dataset. It can be seen that around three fewer kernels are used on the GAL and HIV sets, with five fewer on the T-Cell data. For the T-Cell data this represents a 32% decrease in the size of the network. Whilst there is increased complexity due to the addition of per-residue weights, this is partially offset by a reduction in the number of bio-kernels required to achieve similar performance.

Further analysis work has been carried out on the residue-weighted models produced for the HIV-1 cleavage site prediction problem. Whilst there have been some questions regarding the use of neural networks on the dataset [10], the availability of motifs in the literature will allow comparison with the residue weights and support peptides used in our trained models. In future we will examine the models for datasets without known strong motifs.

The intuitive first step in the analysis to calculate the mean values of the residue weights, to identify the most important positions in the peptides. However, initial inspection suggested that the weights were highly skewed, precluding the sensible use of the mean and standard deviation in analysis work. Histograms were plotted which confirmed that the distribution is heavily positively skewed. Further investigation showed that in general, where any given θ_d value is high, all other position weights will also be high. When examining the relative importance of residues we are interested in the difference between θ_d values, not their absolute value. With this in mind, we have chosen to scale the values, relative to the highest position weight in each model: $\theta'_d = \theta_d/\theta_{MAX}$. This will allow for meaningful averages to be taken, and would not be necessary if examining a single model. Figure 2 shows the minimum, mean, and median values for the scaled weights.

Whilst there is a large spread of values there appears to be a trend that is consistant between the statistics. Values are high in the P1, P2, P1', and P2' positions. The P2' site is recognised as important for cleavage predictions, with Glutamate (E) or Glutamine (Q) identified as amino acids which indicative of a possible cleavage. At P2, Valine (V) or Alanine (A) are associated with an increased likelihood of the peptide being cleavable [10, 11]. The P1 and P1' positions, directly to each side of the cleavage site, are also of importance and similarly mentioned in motifs. The high value of the P4' position was unexpected, since attention is usually paid to the P1-P1' positions [12, 7]. However, Lysine

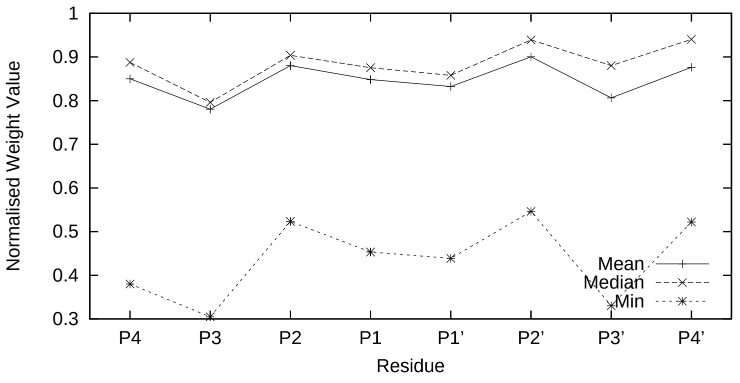


Fig. 2. HIV scaled residue weights.

(K) at P4' was noted as a contributor to positive predictions in 29% of cases in a model given in [10].

The low importance of the P3 and P3' positions were also noted in [10], where it was found that they could be excluded from models without changing the separability of the data. It is perhaps disappointing that the median value for these residue positions is still fairly high. However, whilst there are only fairly small variations in the statistics between each residue, the trends fit with what might be expected. The majority of residue weight updates take place after pruning of support peptides and network weights has taken place, once the network weights are tuned for high performance on with a small set of support peptides. It may be the case that allowing for a cycle of residue weight updates before any pruning of network weights takes place would allow the values to move further; with less tuned network weights, changes to the residue weights would likely cause larger changes in error, in turn causing larger updates. Further investigation would be useful.

One aspect of the motivation for the use of the sparse Bayesian approach was to identify a parsimonious model, with a minimal number of support peptides. For the HIV case there is a mean of 17.07 support peptides per model, of which 53% are positive, i.e. actual cleavage sites. Within the 5 fold, 20 repeat cross validation procedure, each peptide occurs in the training data for 80 models. Therefore the most common support peptide, `DAINTEFK`, observed in 68 models, occurred in 85% of possible cases. The 30 most common support peptides account for 1027 out of the 1707 selections across 100 models, i.e. 60% of support peptides are selected from 8% of the training data. The fact that this small number of peptides are commonly selected across randomly populated training folds seems to indicate that they are representative across all of the data, and that over-fitting to training sets has been eliminated by the sparse Bayesian method.

4 Conclusions

We have introduced a sparse Bayesian bio-kernel network, based on the Bio-Basis Function Neural Network. The new method produces models which have been shown to make accurate predictions on three datasets, using only a small number of support peptides. In addition we have introduced a variant that includes per-residue weights in the basis function. With the added residue weights, slightly better classification accuracies are achieved using fewer support peptides. Analysis of the models produced for HIV-1 protease cleavage site prediction identifies patterns which agree with previous literature.

References

1. R. Thomson, T. C. Hodgman, Z. R. Yang and A. K. Doyle, "Characterising proteolytic cleavage site activity using bio-basis function neural networks", *Bioinformatics*, vol. 19, pp. 1741-1747, 2003.
2. D. C. Trudgian and Z. R. Yang, "Substitution Matrix Optimisation for Peptide Classification", *Lecture Notes In Computer Science*, vol. 4447, pp. 291-300, 2007.
3. M. E. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine", *J. Machine Learning Res.*, vol. 1, pp. 211-244, 2001.
4. N. Qian, and T. J. Sejnowski, "Predicting the secondary structure of globular proteins using neural network models", *J. Mol. Biol.*, vol. 202, pp. 865-884, 1988.
5. D. J. C. MacKay, "Bayesian Interpolation", *Neural Computation*, vol. 4(3), pp 415-447, 1992.
6. K. Chou and C. Zhang and F. J. Kezdy and R. A. Poorman, "A Vector Projection Method for Predicting the Specificity of GalNAc-Transferase", *Proteins*, vol. 21, pp. 118-126, 1995.
7. Y. Cai and K. Chou, "Artificial neural network model for predicting HIV protease cleavage sites in protein", *Adv. in Eng. Software*, vol. 29(2), pp. 119-128, 1998.
8. Y. Zhao, C. Pinilla, D. Valmori, R. Martin and R. Simon, "Application of support vector machines for T-cell epitopes prediction", *Bioinformatics*, vol. 19(15), pp. 1978-1984, 2003.
9. J. A. Hanley B. J. McNeil, "The meaning and use of the area under a receiver operating characteristic (ROC) curve", *Radiology*, vol. 143(1), pp. 29-36, 1982.
10. T. Rognvaldsson and L. You, "Why neural networks should not be used for HIV-1 protease cleavage site prediction", *Bioinformatics*, vol. 20(11), pp 1702-1709, 2004.
11. L. You, D. Garwicz and T. Rognvaldsson, "Comprehensive Bioinformatic Analysis of the Specificity of Human Immunodeficiency Virus Type 1 Protease", *J. Virology*, vol. 79(19), pp. 12477-12486, 2005.
12. A. Narayanan, X. Wu and Z. Yang, "Mining viral protease data to extract cleavage knowledge", *Bioinformatics*, vol. 18, pp. S5-S13, 2002.