Prediction of MHC class I binding peptides using probability distribution functions

Sudhir Singh Soam[1], Feroz Khan[2], Bharat Bhasker[3], Bhartendu Nath Mishra[1]

[1]Institute of Engineering & Technology, (A Constituent College of Uttar Pradesh Technical University, Lucknow) Lucknow, India; [2]Bioinformatics & In Silico Biology Division, Central Institute of Medicinal & Aromatic Plants (CSIR), Lucknow, India, [3] Indian Institute of Management, Prabandh Nagar, Lucknow India. * Corresponding Author: profbmishra@rediffmail.com

Received January 19, 2009; revised March 31, 2009; accepted April 19, 2009; published June 28, 2009

Abstract:
Binding of peptides to specific Major Histo-compatibility Complex (MHC) molecule is important for understanding immunity and has applications to vaccine discovery and design of immunotherapy. Artificial neural networks (ANN) are widely used by predictions tools to classify the peptides as binders or non-binders (BNB). However, the number of known binders to a specific MHC molecule is limited in many cases, which poses a computational challenge for prediction of BNB and hence, needs improvement in learning of ANN. Here, we describe, the application of probability distribution functions to initialize the weights and biases of the artificial neural network in order to predict HLA-A*0201 binders and non-binders. The 10-fold cross validation has been used to validate the results. It is evident from the results that the A_{ROC} for 90% of test cases for Weibull, Uniform and Rayleigh distributions is in the range 0.90-1.0. Further, the standard deviation for A_{ROC} was minimum for Weibull distribution, and may be used to train the artificial neural network for HLA-A*0201 MHC Class-I binders and non-binders prediction.

Keywords: T-cell Epitope, ANN, Probability distribution, MHC binder/non-binder.

Background:

Major Histocompatibility Complex (MHC) plays a central role in the development of both humoral and cell-mediated immune responses. While antibodies may react with antigens alone, most T cells recognize antigens only when it is combined with an MHC molecule; thus, MHC molecules play a critical role in antigen recognition by T cells. T cell do not recognize soluble native antigen but rather recognize antigen that has been processed into antigenic peptides, which are presented in combination with MHC molecules. The T cell epitope must be viewed in terms of their ability to interact with both T-cell receptor and MHC molecule. The antigen binding cleft on an MHC molecule interacts with various oligomeric peptides that functions as T-Cell epitope. The antigen binding cleft on an MHC molecule determines the nature and the size of the peptide(s) that MHC molecule can bind and consequently the maximal size of the T cell epitope. It has been observed that peptides of nine amino residues (9-mers) bind most strongly; peptides of 8-11 residues also bind but generally with lower affinity than nonamers. Binding of a peptide to a MHC molecule is a prerequisite for recognition by T cells and hence is fundamental to understand the basis of immunity and also for the development of potential vaccines **[1, 2]**.

Three type of models that incorporate biological knowledge have been used for prediction of MHC binding peptides: (i) binding motif **[3]**, which represent the anchoring patterns and the amino acids commonly observed at anchor positions, (ii) Quantitative matrices **[4]**, that provide coefficients that quantify contribution of each amino acid at each position within a peptide to MHC/peptide binding, and (iii) Artificial Neural Networks (ANN) **[5, 6]** an arbitrary level of complexity can be encoded by varying the number of nodes in hidden layer and the number of hidden layers. Artificial Neural Networks **[7]** are connectionist models commonly used for classification. ANN is widely used for classification of MHC binder and non-binder. For prediction of T-cell epitope ANN has been used with the HMM (Hidden Markov model) **[8]**, GA (Genetic Algorithms) **[9]**, Evolutionary Algorithm **[10]**. SVM (Support Vector Machine) has also been used to predict the binding peptides **[11]**. Combined GA–ANN model has also been used to find the optimal conditions **[12]**. The work for the present paper has been motivated from the GA-ANN model. Here, in this paper a new approach of using the probability distribution functions to initialize the random weights for artificial neural network training has been demonstrated.

Methodology:

Data Collection

The data sets used for training and testing for binders and non-binders (BNB) were obtained from IEDB Beta 2.0 database [www.immuneepitope.org] for HLA-A*0201 MHC Class I allele. The 1609 peptides with $0 <= IC_{50} <= 500$ have been retrieved as binders and 397 peptides with $IC_{50} > 5000$ have been retrieved as non-binders. After removing the duplicates, 800 9-mer binders and 256 9-mer non-binders have been used for training and prediction as shown in **Table 5**. Since the ratio of binders and non-binders have to be kept nearly 1:1 in order to reduce the biasness in learning, the additional 544 9-mer non-binders have been generated through ExPASy server. Further, the common peptides among binders and newly generated 9-mer non-binders have been deleted. At last 800 nonamer binders and 790 nonamer non-binders have been used for training and prediction.

```
While terminating condition not satisfied {
    for each training sample X in samples {
        // propagate the inputs forward:
    for each hidden or output layer unit j {
    $I_j = \sum_i w_{ij} O_i + \theta_j$; // compute the net input of unit j w.r.t. the previous layer, i
    $O_j = 1 / (1 + \exp(-I_j))$} // compute the output of each unit.
// Back propagate the errors.
for each unit j in the output layer
    $Err_j = O_j (1 - O_j) (T_j - O_j)$; // compute error.
for each unit j in the hidden layers, from the last to the first hidden layer
    $Err_j = O_j (1 - O_j) \sum_k Err_k w_{jk}$ // compute error w.r.t. the next higher layer, k.
for each weight wij in network {
    $\Delta w_{ij} = (L) Err_j O_i$; // weight increment
    $w_{ij} = w_{ij} + \Delta w_{ij}$; } // weight update.
for each bias $\theta_j$ in the network {
    $\theta\Delta_j = (L) Err_j$; // bias increment
        $\theta_j = \theta_j + \theta\Delta_j$; }// bias update
}}
```

Figure 1: The Back propagation algorithm.

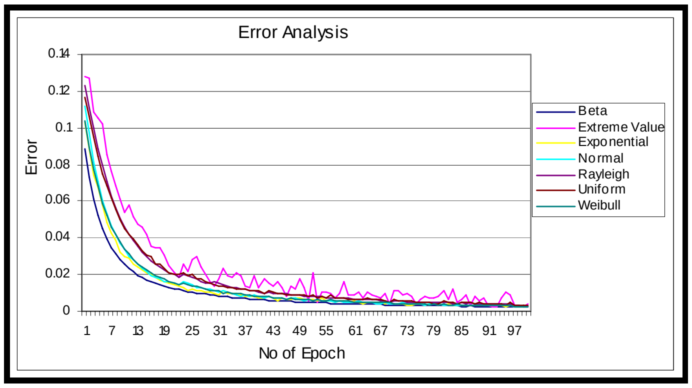

Figure 2: The error analysis for small number of epoch (to make convergence clear)

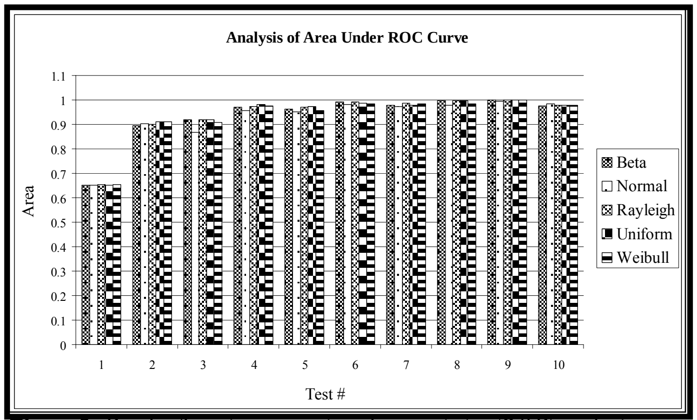

Figure 3: Graph of receiver operating characteristics (ROC) analysis.

Algorithm used for the prediction of MHC binding peptides:

Probability distribution based weights and biases initialization

A probability model does not allow to predict the result of any individual experiment but the probability that a given outcome will fall inside a specific range of values cab be determined by using the model. Since the weights of the ANN are small numbers and the variation among them should be small, so continuous probability distributions have been used for initialization of weights and biases for artificial neural network. Beta, Exponential, Extreme Value, Gamma, Lognormal, Normal, Rayleigh, Uniform and Weibull continuous distributions have been examined in the studied research work. Following steps have been used to generate the small random numbers using MATLAB [www.mathworks.com]: (1) Use the functions given in second column of **Table 1** (see supplementary material) to generate a vector of small random numbers; (2) The functions given in the third column of the **Table 1 (**see supplementary material**)** have been used to estimate the parameters and confidence interval for a given distribution; (3) Repeat the steps 1 and 2 till the parameters correspond to 95% confidence intervals.

Back propagation method for learning of artificial neural network

There are 20 amino acids found in all kinds of proteins. To code each amino acid a 20 bit binary code is used. For each binary code it will have value 1 according to its position and rest of the values is zeros. Since the binder and non binders sequences are 9-mer, hence a binder sequence will be represented by a vector of 180 (20x9) binary values. The model is used for only predicting the binder or non binder for a given 9-mer sequence, hence one output node and two hidden nodes are used. Therefore, 180 input nodes 2 nodes in a single hidden layer and 1 output node have been used to model. If the value at the output for a given epitope is less then the given threshold it is classified as non-binder otherwise the epitope is predicted as binder. The back propagation method has been used for learning ANN. For each training sample the weights have been modified so as to minimize the mean squared error between the network's prediction and the actual prediction. This error has been propagated backwards by updating the weights and biases to reflect the error of the network's prediction. The algorithm is shown in **Figure 1**.

Evaluation Parameters

The predictive performance for Beta, Normal, Rayleigh, Uniform, and Weibull distributions was accessed using receiver operating characteristics (ROC) analysis. The area under the ROC curve (A_{ROC}) provides a measure of overall prediction accuracy, $A_{ROC} < 70$ % for poor, $A_{ROC} > 80$ % for good, $A_{ROC} > 90$ % for excellent prediction **[13]**. The ROC curve is generated by plotting sensitivity (SN) as a function of 1-specificity (SP). The sensitivity, SN=(TP/(TP+FN))*100 and SP=(TN/(TN+FP))*100, gives percentage of correctly predicted binders and non-binders respectively. The PPV = ((TP)/(TP+FP))*100 and NPV=((TN)/(FN+TN))*100 gives the positive probability value i.e. the probability that a predicted binder will actually be a binder, and negative probability value i.e. the probability that a predicted non-binder will actually be a non-binder. The terms are defined in **Table 2 (**see supplementary material**)**. 10-fold cross validation has been used for training and prediction of the artificial neural network with various probability distribution functions. 10 data sets of BNB have been designed. The training has been done for 9 test data set (*i.e.* 1st test data to test data 9th) and the 10th data set has been used for prediction and the results have been recorded. Then the 2nd test data to 10th test data have been used for training and the 1st has been used for prediction. Similarly when the prediction has been done for the ith test data the remaining 9 test data except for ith have been used for training.

Implementation:

The programs for training and classification have been implemented using C on Windows environment. The initial weights and biases matrix using various probability distributions functions have been created by MATLAB.

Results:

The continuous (data) probability distributions (Beta, Exponential, Extreme value, Gama, Lognormal, Normal, Rayleigh, Uniform, Weibull) have been used for initialization the weights. Gama and Lognormal continuous distributions have been discarded because the variations among the random initial values were too high, and hence not found suitable for modeling. The probability distribution functions and the estimated values of parameters using MLE (Maximum Likelihood Estimation) have been shown in **Table 3 (**see supplementary material**)** except for Gama and Lognormal. The probability distributions except Gama and Lognormal have been used for learning the ANN. Exponential and Extreme value distributions have been discarded because the error convergence curve is not smooth which might lead to wrong predictions as it is evident from the error graph shown in **Figure 2**.

The 10-fold cross validation has been used to validate the results. In 10-fold cross-validation, the data has been divided into 10 subsets of (approximately) equal size. The ANN has been trained 10 times, each time leaving out one of the subsets from training, but using only the omitted subset for prediction results. The 800 binders and 790 non binders have been divided in 10 sets of 80 and 79 respectively for prediction. The remaining binders and non-binders have been used for training. The ANN has been trained for 10 times for every probability distribution function leaving one out one of the subset from training and uses that for the prediction of BNB. Web based tool have been used to calculate the area under the ROC curve [www.rad.jhmi.edu/jeng/javarad/roc/JROCFITi.html].
Area under the fitted ROC curve for BNB sequences have been shown in **Table 4 (**see supplementary material**)** and the analysis of are under the ROC curve having been shown in **Figure 3**. The mean and standard deviation have been calculated for various probability distributions.

Discussion:

We assembled a data set of binders and non-binders for HLA-A*0201 MHC Class I to study the impact of the probability distribution function for initialization of weights and biases of artificial neural network, motivated by the GA-ANN model where the GA have been used to initialize the weights and biases of artificial neural network. The high binding affinity peptides with $0<=IC_{50}<=500$ have been retrieved as binders and low binding affinity peptides with $IC_{50}>5000$ have been retrieved as non-binders from IEDB Beta 2.0 database. The total number of binders and non-binders was 1609 and 397 respectively. A set of 800 9-mer binders and 256 9-mer non-binders have been prepared after eliminating the duplicates. The ratio of binders and non-binders have to be kept nearly 1:1 in order to reduce the biasness in learning, hence, additional 544 9-mer non-binders have been generated from a EBI-Expasy protein database and added to the non-binder set. Finally 800 9-mer binders and 790 9-mer non-binders have been used for training and prediction after further removing the duplicates caused by newly generated non-binders. The 10 sets of binders and non-binders of nearly equal size have been made for 10-fold cross validation.

The results have been shown in **Table 4 (**see supplementary material**)** for all the probability distribution functions for all the test sets. The mean values of area under ROC curve for Beta, Normal, Rayleigh, Uniform and Weibull is 0.934, 0.924, 0.9367, 0.937 and 0.9337 respectively. All the distributions have performed well. The standard deviation for each has also calculated which shows that the standard deviation is minimum for Weibull probability distribution. The threshold parameter has been varied from 0.5 to 0.95. Further the values for Sensitivity, Specificity, PPV, NPV and accuracy for Beta, Normal, Rayleigh, Uniform, and Weibull distributions for all sets have been shown in **Table 6, 7, 8, 9,** and **10 (**see supplementary material**)**, respectively.

From the above results it is evident that the weight initialization may have an impact on the performance of artificial neural network. This is basically adding some prior knowledge to the artificial neural network. The MHC class-I 9-mer binders and non-binders may have any combination of 20 amino acids. The amino acids at the position 1 to 9 may follow a probability distribution or close to any probability distribution. As the results have shown that in case of HLA-A*0201 allele the performance was better in case when the weights for artificial neural network have been initialized using Weibull probability distribution. The modules for the training, classification, and results have been implemented in C using pointers, in order to improve the efficiency of training and classification. Overall this study shows that the quality of the prediction of binders and non-binders can be substantially improved by using the probability distributions for initialization of the weights for artificial neural network.

Acknowledgements:
The authors are grateful to Dr D S Yadav, Dept of CSE, I. E. T. Lucknow, Sri S. P. Singh Amity University Lucknow, for kind cooperation. Sri Rajeev Kumar, Department of Mechanical Eng. for providing MATLAB.

References:

[1] A. S. De Groot *et al., Immunology and Cell Biology,* (2002) **80:** 255-269.
[2] S. Buss *et al.*, *Tissue Antigens*, (2003) **62**, 378-384.
[3] H.G. Rammensee, *et al.,* Immunogenetics, (1999) **50**:213-219.
[4] M. Bhasin, G. P. S. Ragahava, *Vaccine*, (2004) **22**, 3195-3204.
[5] M. Bhasin, G. P. S. Raghava, *J. Biosciences*, (2007) **32**(1), 31-42.
[6] V. Brussic *et al.,* et al., *Methods,* (2004) 34: 436-443
[7] G. Armano *et al., BMC Bioinformatics*, (2005), **6**(Suppl 4).
[8] G. L. Zhang *et al., Nucleic Acid Research,* (2005) **33**:172-179.
[9] A. Prugel-Bennett, J.L. Shapiro, *Physical Review Letters*, (1994), 72(9):1305-09.
[10] V. Brusic *et al., Bioinformatics,* (1998) **14**(2):121-30.
[11] H. Riedesel *et al., Genome Informatics,* (2004) **15**(1): 198-212.
[12] M. Anijdan S.H. *et al., Science Direct*, (2006) **27**(7): 605-609.
[13] G. L. Zhang *et al., Immunome Research*, (2006) **2**:3.

Edited by P. Kangueane
Citation: Soam *et al***,** Bioinformation 3(9): 403-408 (2009)

Supplementary Material

S. No.	Name of Distribution	Functions for generating Random Numbers	Parameter Estimation Function
1.	Beta	R=betarnd (a,b,m,n)	[phat, pci] = betafit(data)
2.	Exponential	R=exprnd(μ,m,n)	[parmhat, parmci] = expfit(data)
3.	Extreme value	R=evrnd(μ,Σ, m, n)	[parmhat, parmci] = evfit(data)
4.	Gama	R=gamrnd(A,B,m,n)	[p,ci] = gamfit(data)
5.	Normal	R=normrnd(μ,Σ, m, n)	[mu,sigma,muci,sigmaci] = normfit(data)
6.	Rayleigh	R=raylrnd(B,m,n)	[phat, pci] = raylfit(data, alpha)
7.	Uniform	R=unifrnd(A,B,m,n)	[ahat,bhat,aci,bci] = unifit(r)
8.	Weibull	R=wblrnd(A,B,m,n)	[p,ci] = wblfit(strength)

Table 1: The functions for random number generation and parameter estimation

S. No.	Threshold	Binders	Non-binders
1.	Score at least threshold T	TP	FP
2.	Score under threshold T	FN	TN

Table 2: Explanation of the terms TP (true positive), FP (false positive), TN (true negative) & FN (false negative) related to threshold T

S. No.	Name of Distribution	Functions for generating Random Numbers	Parameter Estimation Function
1.	Beta	R=betarnd (a,b,m,n)	a=5, & b=0.2
2.	Exponential	R=exprnd(μ,m,n)	μ=0.1
3.	Extreme value	R=evrnd(μ,Σ, m, n)	μ=0.05, & Σ=0.2
4.	Normal	R=normrnd(μ,Σ, m, n)	μ=.1, & Σ=0.05
5.	Rayleigh	R=raylrnd(B,m,n)	b=0.01
6.	Uniform	R=unifrnd(A,B,m,n)	a=-0.1, & b=0.1
7.	Weibull	R=wblrnd(A,B,m,n)	a=0.1 & b=2

Table 3: Function and value of respective parameters. Here 'R' refers (m X n) matrix

Test Set #	Beta	Normal	Rayleigh	Uniform	Weibull
Test 1	0.651	0.651	0.653	0.652	0.655
Test 2	0.894	0.902	0.900	0.911	0.911
Test 3	0.919	0.867	0.920	0.918	0.909
Test 4	0.969	0.958	0.973	0.980	0.975
Test 5	0.963	0.951	0.969	0.973	0.956
Test 6	0.993	0.980	0.993	0.986	0.985
Test 7	0.978	0.974	0.986	0.978	0.983
Test 8	0.996	0.979	0.997	0.998	0.985
Test 9	1.000	0.994	1.000	1.000	1.000
Test 10	0.977	0.984	0.978	0.978	0.978
Mean	0.934	0.924	0.937	0.937	0.934
Std. Dev.	0.105	0.104	0.105	0.105	0.103

Table 4: Area under the ROC curve for various distributions along with mean and standard deviation

S. No.	Binders/Non-binders (BNB)	Total Records Retrieved	Criteria	9-mer after Removing Duplication
1.	Binders	1609	$0<=IC_{50}<=500$	800
2.	Non-Binders	397	$IC_{50}>5000$	256

Table 5: The Binder's and Non-binder's obtained from IEDB Beta 2.0 version

Set#	Senstivity	Specificity	Accuracy	PPV	NPV
1.	79.746834	39.240505	59.493671	56.756756	65.957443
2.	92.40506	67.088608	79.746834	73.737373	89.830505
3.	93.670883	72.151901	82.911392	77.083336	91.935486
4.	94.936707	89.873421	92.40506	90.361443	94.666664
5.	87.341774	93.670883	90.506332	93.24324	88.095238
6.	93.670883	100	96.835442	100	94.047623
7.	84.810127	100	92.40506	100	86.813187
8.	94.936707	98.734177	96.835442	98.684212	95.121948
9.	100	100	100	100	100
10.	88.607597	96.20253	92.40506	95.890411	89.411766

Table 6: The values of SN (Sensitivity), SP (Specificity), PPV (Positive prediction value), NPV (Negative prediction value) and Accuracy for Beta Probability Distribution

Set#	Senstivity	Specificity	Accuracy	PPV	NPV
1.	73.417725	48.101265	60.759495	58.585857	64.406776
2.	92.40506	75.949364	84.177216	79.347824	90.909088
3.	82.278481	81.012657	81.645569	81.25	82.051285
4.	89.873421	92.40506	91.139244	92.207794	90.123459
5.	86.075951	96.20253	91.139244	95.774651	87.356323
6.	88.607597	100	94.303795	100	89.772728
7.	82.278481	100	91.139244	100	84.946236
8.	88.607597	98.734177	93.670883	98.591553	89.655174
9.	98.734177	100	99.367088	100	98.75
10.	88.607597	98.734177	93.670883	98.591553	89.655174

Table 7: The values of SN, SP, PPV, NPV and Accuracy for Normal Probability Distribution

Set#	Senstivity	Specificity	Accuracy	PPV	NPV
1.	81.012657	44.303799	62.658226	59.259258	70
2.	93.670883	68.354431	81.012657	74.747475	91.525421
3.	94.936707	70.886078	82.911392	76.530609	93.333336
4.	94.936707	89.873421	92.40506	90.361443	94.666664
5.	87.341774	93.670883	90.506332	93.24324	88.095238
6.	92.40506	100	96.20253	100	92.941177
7.	83.544304	100	91.772148	100	85.869568
8.	94.936707	97.468353	96.20253	97.402596	95.061729
9.	100	100	100	100	100
10.	94.936707	96.20253	95.569618	96.153847	95

Table 8: The values of SN, SP, PPV, NPV and Accuracy for Rayleigh Probability Distribution

Set #	SEN	SPE	ACC	PPV	NPV
1	81.012657	44.303799	62.658226	59.259258	70
2	93.670883	73.417725	83.544304	77.894737	92.063492
3	94.936707	73.417725	84.177216	78.125	93.548386
4	93.670883	89.873421	91.772148	90.243904	93.421051
5	87.341774	93.670883	90.506332	93.24324	88.095238
6	92.40506	100	96.20253	100	92.941177
7	82.278481	98.734177	90.506332	98.484848	84.782608
8	96.20253	98.734177	97.468353	98.701302	96.296295
9	100	100	100	100	100
10	94.936707	97.468353	96.20253	97.402596	95.061729

Table 9: The values of SN, SP, PPV, NPV and Accuracy for Uniform Probability Distribution

Set #	Senstivity	Specificity	Accuracy	PPV	NPV
1	78.48101	45.569622	62.025318	59.047619	67.92453
2	93.670883	72.151901	82.911392	77.083336	91.935486
3	93.670883	69.620255	81.645569	75.510201	91.666664
4	97.468353	88.607597	93.037971	89.534882	97.222221
5	87.341774	93.670883	90.506332	93.24324	88.095238
6	92.40506	97.468353	94.936707	97.333336	92.771088
7	82.278481	98.734177	90.506332	98.484848	84.782608
8	94.936707	98.734177	96.835442	98.684212	95.121948
9	100	100	100	100	100
10	94.936707	97.468353	96.20253	97.402596	95.061729

Table 10: The values of SN, SP, PPV, NPV and Accuracy for Weibull Probability Distribution