Sequence Based T Cell Epitope Prediction: Verification and Iprovements

Alexander Crowell, Jack Holland, and Vineetha Paruchuri

***Abstract*—Successful preduction of T cell epitopes is very useful in vaccine design and immunotherapy. Obviously, computational prediction is a lot cheaper than experimental validation for T cell epitope prediction. Researchers have previously predicted the T cell epitopes with varying degrees of success. In this paper, we use Support Vector Machine, a machine learning algorithm, to do the same, and we compare our results to other approaches of T cell epitope prediction, like NetMHCII, NetMHCIIpan, and the MultiRTA methods. Our analysis showed that ¡insert data¿**

Keywords*—*T Cell Epitopes, Epitope Mapping, Epitope Prediction, NetMHC, MultiRTA, Neural Networks, Support Vector Machines, Machine Learning.

I. Introduction

TCELL epitopes are the part of an antigen, usually a specific peptide sequence, which is bound by the Major Histocompatibility Complex (MHC). There are two types of MHC proteins, MHC I, which is present on all cells, and MHC II, which is only present on antigen-presenting cells. Due to differences in the structure of their binding pockets, T cell peptides bound by MHC class I molecules are between 8 and 11 amino acids in length, whereas those presented by MHC class II are 13-17 amino acids in length.

The high degree of polymorphism exhibited by MHC II alleles means that experimental validation of epitope regions is cost intensive. Hence, the computational methods for accurate prediction of binding peptides to class II molecules is a topic of interest. Researchers have previously tried to computationally predict T cell epitopes, with various degrees of success, using Artificial Neural Networks, such as in NetMHCII [**?**] and NetMHCIIpan [**?**], or Position Specific Scoring Matric [2] such as in TEPITOPEpan [**?**], or Position Weight Matrix, such as in PREDIVAC [**?**].

In this project, we use the machine learning technique of Support Vector Machines to predict T cell epitopes, and compare the performance of our approach to allele-specific and pan-specific approaches like NetMHC, NetMHCIIpan, and MultiRTA [**?**]. The results show that ¡insert something about our performance¿

II. NetMHCII

NetMHCII is a neural network based approach for MHC class II predictions. It uses a novel artificial neural network-based method, *NN-align*, which allows for simultaneous identification of the MHC class II binding core and binding affinity. As mentioned in [**?**], the NN-align method includes "explicit encoding of the peptide flanking residues in terms of amino acid composition and length, as well as a novel scheme for neural network training that deals with the data redundancy inherent in the peptide data due to multiple examples of identical binding cores". The NN-align method is trained on a large data set of more than 14,000 quantitative peptide MHC binding values covering 14 HLA-DR alleles.

A. NetMHCII Implementation

The implementation of NN-align was done as an artificial neural network. It was done as a two-step procedure that simultaneously estimated the optimal peptide binding register, and the network weight configuration (where the network weights were updated using gradient descent back propogation). The score of a 9 mer peptide was calculated using the conventional feed-forward algorithm. The incorporation of information from residues flanking the peptide binding core seemed to considerably improve the predictive performance of the method.

B. NetMHCII Performance, and Metrics Used

In NetMHCII, all statistical comparisons were made using one-tailed binomial tests. For each comparison, it was calculated how often one method outperforms the other (excluding ties). One-tailed p-values were calculated based on these numbers. P-values ¡ 0.05 were taken to be significant. Performance across different methods was calculated using AUC.

C. Inferences from NetMHCII

Our critique goes here. I'll update once we discuss what we wanna say about it. Maybe something like it is comprehensive, and tested for all these things x y z, which is why we think it is a good method for MHCII binding predictions despite ¡cite limitations¿, so we want to compare ours to this etc.?

III. Support Vector Machines

Support Vector Machine is a supervised learning algorithm used for classification and regression analysis - they identify patterns in the data. Once we train the SVM on some data (called the training data set), it classifies new examples into one of two classes, thereby making it a non-probablistic binary linear classifier. An SVM can also perform non-linear classification using *kernel trick*, which just means that it doesn't need to compute the coordinates of data in a high-dimensional space, but rather just the inner products between images of all pairs of data, thereby making it computationally cheaper.

Given the fact that it is a binary linear classifier that also performs non-linear classification, we felt that, as a first step, using SVM as the choice of machine learning algorithm made sense.

A. Formal Definition

From Wikipedia,
Given some training data D, a set of n points of the form

$$D = \{(x_i, y_i) \mid x_i \in R^p, y_i \in \{-1, 1\}\}_{i=1}^{n}$$

where the y_i is either 1 or 1, indicating the class to which the point x_i belongs. Each x_i is a p-dimensional real vector. We want to find the maximum-margin hyperplane that divides the points having $y_i = 1$ from those having $y_i = -1$. Any hyperplane can be written as the set of points x satisfying

$$\mathbf{w}.x - b = 0,$$

where $\cdot$ denotes the dot product and w the (not necessarily normalized) normal vector to the hyperplane. The parameter $\frac{b}{\|\mathbf{w}\|}$ determines the offset of the hyperplane from the origin along the normal vector $\mathbf{w}$.

If the training data is linearly separable, we can separate the examples under two hyperplanes such that there are no points between them. We can then try to maximize their distance. The region bound by these hyperplanes is called "the margin", and ideally, we want to maximize it so that there is no overlap.

These hyperplanes can be described by the equations

$$\mathbf{w}.\mathbf{x} - b = 1 \quad \text{and} \quad \mathbf{w}.\mathbf{x} - b = -1$$

We then find the geometric distance between these two hyperplanes is $\frac{2}{\|\mathbf{w}\|}$, so we want to minimize $\|\mathbf{w}\|$. We also add the below constraint because we want to prevent the data points from falling under the margin:

For each i either

$$\mathbf{w} \cdot \mathbf{x}_i - b \geq 1 \qquad \text{for } \mathbf{x}_i$$

of the first class, or

$$\mathbf{w} \cdot \mathbf{x}_i - b \leq -1 \qquad \text{for } \mathbf{x}_i$$

of the second.

Which can be written as:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i - b) \geq 1, \quad \text{for all } 1 \leq i \leq n. \tag{1}$$

We can express the above as an optimization problem:
Minimize $(in \quad \mathbf{w}, b)$

$$\|\mathbf{w}\|$$

subject to $(for \ \ any \ \ i = 1, \ldots, n)$

$$y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1.$$

The optimization problem presented above can be converted into a quadratic programming optimization problem, which gives the primal and dual forms of the SVM. In this project, we used the dual form of the SVM, "which reveals that the maximum-margin hyperplane, and therefore the classification task is only a function of the support vectors" [**?**].

IV. Methods

This section provides a brief overview of how we implemented our algorithm and how we validated it.

A. Datasets

We took the peptides from IEDB, and subdivided them according to allele type. We also tried out other data repositories, like the Dana-Farber [**?**] repository, but since all entries in it have a positive binding value, we didn't find it suitable for our problem. We also ran our algorithm on the MultiRTA [**?**] dataset. Data selection and sanitization is a crucial step here, because any irregularities in data can affect the learning of the SVM in unpredictable ways.

B. Implementation of the SVM and Metrics

We implemented SVM in python using scikit-learn [**?**], and used scikit-learn functions such as roc_auc_score etc. for metrics.

C. Encoding Peptides

D. AUC as a Comparison Metric

Area Under the Curve (AUC) is used as a metric throughout, because other approaches used the same metric or comparable metrics. Also, some of our data is skewed, and since AUC accounts for both true and false positives, it ensures that any good predictions we get isn't due to the fact that we have more positives in our data.

V. Results

A. Individual Alleles

B. Combined Alleles

C. Results Compared to RTA

D. Results Compared to NetMHCII

E. Results Compared to NetMHCIIpan

Can describe what results we got in cross validaton, what we compared to, how it scaled up etc.

VI. Conclusion

Our inferences in how this performed and what it means etc.

Appendix A
Division of Work

A. Alexander Crowell

B. Jack Holland

C. Vineetha Paruchuri

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.

[2] Sturniolo T, Bono E, Ding J, Raddrizzani L, Tuereci O, et al. *Generation of tissue-specific and promiscuous HLA ligand database using DNA microarrays and virtual HLA class II matrices*, 7th ed. Nat Biotechnol, 1999, pp.555561