Predicting Protein-Peptide Binding Affinity by Learning Peptide-Peptide Distance Functions

Chen Yanover[1⋆] and Tomer Hertz[1,2⋆]

[1] School of Computer Science and Engineering
[2] The Center for Neural Computation,
The Hebrew University of Jerusalem, Jerusalem, Israel, 91904
cheny@cs.huji.ac.il, tomboy@cs.huji.ac.il

Abstract. Many important cellular response mechanisms are activated when a peptide binds to an appropriate receptor. In the immune system, the recognition of pathogen peptides begins when they bind to cell membrane Major Histocompatibility Complexes (MHCs). MHC proteins then carry these peptides to the cell surface in order to allow the activation of cytotoxic T-cells. The MHC binding cleft is highly polymorphic and therefore protein-peptide binding is highly specific. Developing computational methods for predicting protein-peptide binding is important for vaccine design and treatment of diseases like cancer.
Previous learning approaches address the binding prediction problem using traditional margin based binary classifiers. In this paper we propose a novel approach for predicting binding affinity. Our approach is based on learning a peptide-peptide distance function. Moreover, we learn a **single** peptide-peptide distance function over an **entire** family of proteins (e.g MHC class I). This distance function can be used to compute the affinity of a novel peptide to any of the proteins in the given family. In order to learn these peptide-peptide distance functions, we formalize the problem as a semi-supervised learning problem with partial information in the form of equivalence constraints. Specifically we propose to use *DistBoost* [1, 2], which is a semi-supervised distance learning algorithm.
We compare our method to various state-of-the-art binding prediction algorithms on MHC class I and MHC class II datasets. In almost all cases, our method outperforms all of its competitors. One of the major advantages of our novel approach is that it can also learn an affinity function over proteins for which only small amounts of labeled peptides exist. In these cases, *DistBoost*'s performance gain, when compared to other computational methods, is even more pronounced.

1 Introduction

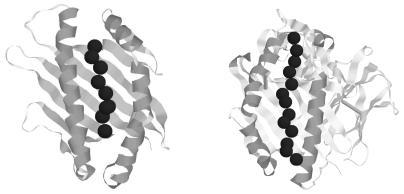
Understanding the underlying principles of protein-peptide interactions is a problem of fundamental importance in biology, with application to medicinal chemistry and drug design. Many cellular responses and activation mechanisms

⋆ Both authors contributed equally

are triggered when a peptide binds to an appropriate receptor which leads to a cascade of downstream events. This communication mechanism is widely used in eukaryotic cells by cytokins, hormones and growth factors. In the immune system, the activation of cytotoxic T-cells is mediated by the binding of foreign antigen peptides to cell membrane Major Histocompatibility Complexes (MHCs). Antigen presenting MHC molecules are usually divided into two major classes: class I and class II (see Fig. 1). The function of both class I and class II molecules is to carry short peptides to the cell surface for presentation to circulating T-cells. The recognition of these pathogen peptides as non-self by T-cells elicits a cascade of immune responses. MHC binding peptides, therefore, play an important role in diagnosis and treatment of cancer [3].

As a result of two decades of extensive experimental research, there exists empirical evidence on peptides that bind to a specific MHC molecule and peptides that do not bind to it. In some cases, binding peptides are classified as either high-affinity, moderate-affinity or low-affinity binders. Empirical evidence reveals that only 1 out of 100 to 200 peptides actually binds to a particular MHC molecule [4]. Since biochemical assays, which empirically test protein-peptide binding affinity, are not amenable to high throughput analysis, protein-peptide computational prediction methods come into play. Many different computational approaches have been suggested for predicting protein-peptide binding including motif based methods [5, 6], structural methods [7] and machine learning algorithms [8–10]. While all of these methods obtain promising results, the problem seems far from being solved.

Many machine learning prediction methods [8–10] are implicitly based on the observation that peptides that bind to a specific protein are similar in some sense. These learning algorithms formalize the problem as a binary (margin based) classification task — binding predictions are provided using a classifier which is trained to separate the binding and non-binding peptides for each protein independently.



(**a**) (**b**)

Fig. 1. Schematized drawing of a peptide in the binding groove of MHC class I (**a**) and class II (**b**) molecules. The peptide backbone is shown as a string of balls, each of which represents a residue.

In this paper we propose a novel approach for predicting protein-peptide binding affinity. Our approach is based on learning a peptide-peptide distance (or similarity) function[1]. This peptide-peptide distance function can then be used to compute a protein-peptide affinity score. We further propose to pool together information about binding and non-binding peptides from a number of related proteins ("protein family", e.g MHC class I). Our algorithm uses this data to learn a **single** peptide-peptide distance function for an entire protein family. Intuitively, a "good" learnt distance function should assign relatively small values (distances) to pairs of peptides that

[1] We do not require that the triangle inequality holds, and thus our distance functions are *not* necessarily metrics.

bind to a specific protein. Thus, given a novel binding peptide, we would expect its average distance to all known binders to be relatively small (as opposed to the average distance of a novel non-binding peptide to the same known binding peptides). We therefore propose the following learning scheme:

1. Compile a dataset of binding and non binding peptides from an entire protein family (e.g MHC class I or MHC class II).
2. Use the *DistBoost* algorithm [1, 2] to learn a single peptide-peptide distance function over this dataset using the information provided about binding and non-binding peptides.
3. Use this learnt distance function to compute the affinity of novel peptides to any of the proteins in the protein family.

We compare our method to various protein-peptide affinity prediction methods on several datasets of proteins from MHC class I and MHC class II. The results show that our method significantly outperforms all other methods. We also show that on proteins for which small amounts of binding peptides are available our improvement in performance is even more pronounced. This demonstrates one of the important advantages of learning a single peptide distance function on an entire protein family.

2 Related work

Many different computational approaches have been suggested for the protein-peptide binding prediction problem (see [11] for a recent review). These methods can be roughly divided into three categories:

Motif based methods: Binding *motifs* represent important requirements needed for binding, such as the presence and proper spacing of certain amino acids within the peptide sequence. Prediction of protein-peptide binding is usually performed as motif searches [5]. The position specific scoring matrix (PSSM) approach is a statistical extension of the motif search methods, where a matrix represents the frequency of every amino acid in every position along the peptide. Peptide candidates can be assigned scores by summing up the position specific weights. The *RANKPEP* resource [6] uses this approach to predict peptide binding to MHC class I and class II molecules.

Structure based methods: These methods predict binding affinity by evaluating the binding energy of the protein-peptide complex [7]. Note that these methods can be applied only when the three-dimensional structure of the protein-peptide complex is known or when reliable molecular models can be obtained.

Machine learning methods: Most of the methods in this category formalize the problem of predicting protein-peptide binding as a binary classification problem — for each specific protein, a classifier is trained to separate binding and non-binding peptides. Many different algorithms have been proposed. Among these are artificial neural networks (NetMHC) [10], hidden Markov models (HMM's) [8] and support vector machines (SVMHC) [9]. These methods require sufficient amounts of training data (i.e peptides which are known to be binders or non-binders) for each of the proteins.

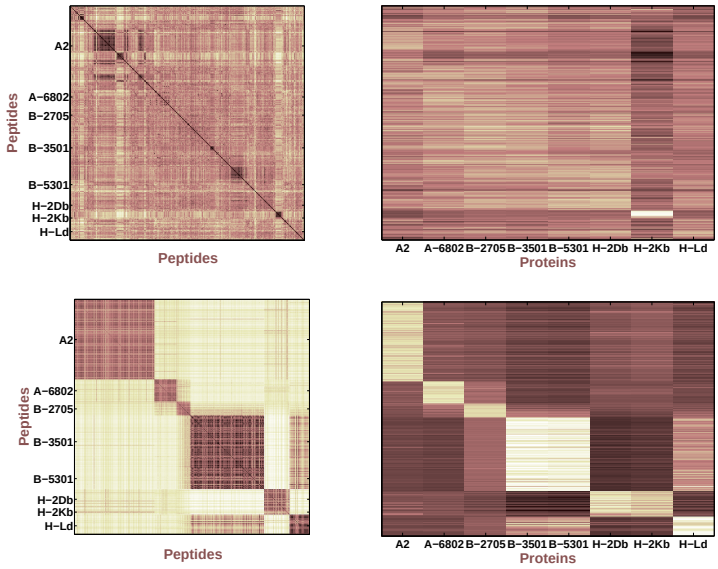


Fig. 2. Left: peptide-peptide distance matrices on MHC class I binding peptides, collected from the MHCBN dataset. Peptides that bind to each of the proteins were grouped together and labeled accordingly. A "good" distance matrix should therefore be block diagonal. Top left: The Euclidean peptide-peptide distance matrix in $\mathbb{R}^{45}$ (see Section 5 for details). Bottom left: The peptide-peptide distance matrix computed using the *DistBoost* algorithm. **Right**: protein-peptide affinity matrices. The affinity between a peptide and a specific protein is computed by measuring the average distance of the peptide to all peptides known to bind to that protein (see eq. 1). Top right: the Euclidean affinity matrix. Bottom right: the *DistBoost* affinity matrix. *DistBoost* was trained on binding peptides from all of the molecules **simultaneously**.

3 From peptide distance functions to binding prediction

As mentioned above, we propose to address the protein-peptide binding affinity prediction problem by learning a peptide-peptide distance function. Intuitively, we would expect that peptides that bind to a specific protein would be "similar" (or close) to each other, and "different" (far) from peptides that do not bind to this protein. Following this intuition, our goal is to learn a distance function, which assigns relatively small distances to pairs of peptides that bind to a specific protein and relatively large distances to pairs, consisting of a binder and a non binder (see Fig. 2). We can then predict the binding affinity of a novel peptide to a specific protein, by measuring its average distance to all of the peptides which are known to bind to that protein (see Fig. 2). More formally, we define the affinity between peptide i and protein j to be:

$$\mathcal{A}ffinity(Peptide_i, Protein_j) \equiv e^{\left(-\frac{1}{|Binding_j|}\sum_{k\in Binding_j}\mathcal{D}(Peptide_i,Peptide_k)\right)} \quad (1)$$

where $\mathcal{D}(Peptide_i, Peptide_k)$ is the distance between $Peptide_i$ and $Peptide_k$ and $Binding_j = \{k|Peptide_k \text{ is known to bind to } Protein_j\}$.

For each protein, our training data consists of a list of binding peptides (and, possibly, their binding affinities) and a list of non-binding peptides. This form of partial information can be formally defined as equivalence constraints. Equivalence constraints are relations between pairs of data points, which indicate whether the points in the pair belong to the same category or not. We term a constraint *positive* when the points are known to be from the same class, and *negative* in the opposite case. In our setting, each protein defines a class. Each pair of peptides (data points) which are known to bind to a specific protein (that is, belong to the same class) defines a positive constraint, while each pair of peptides in which one binds to the protein and the other does not — defines a negative constraint.

In this work we propose to use the *DistBoost* algorithm for learning peptide-peptide distance functions. Moreover, we propose to learn a single peptide distance function using partial information about binding and non binding peptides on several proteins from the same "protein family". When the different classes share common characteristics, learning a single distance function for all classes might benefit from a larger and more diverse training set. Our suggested approach has, therefore, the following potential advantages over the above mentioned computational approaches: (1) it can be used on proteins for which only a small amount of training data is available, (2) it can also be used to predict peptide binding affinity to novel proteins and (3) it can compute the relative binding affinities of a peptide to several proteins from the same protein family.

It should be emphasized that one cannot employ standard multi-class learning techniques in this scenario, since peptides do not have a well defined label. The partial information we have access to cannot be regarded as labeled data for three reasons: (1) if a peptide does not bind to a specific protein, it will not necessarily bind to a different protein from the same protein family, (2) peptides that bind to a specific protein can also bind to other proteins in its family and (3) most peptide pairs are unlabeled, that is we do not know whether they both bind to some specific protein, or not.

4 The *DistBoost* Algorithm

Let us denote by $\{x_i\}_{i=1}^n$ the set of input data points which belong to some vector space $\mathcal{X}$. The space of all pairs of points in $\mathcal{X}$ is called the "product space" and is denoted by $\mathcal{X} \times \mathcal{X}$. An equivalence constraint is denoted by (x_{i_1}, x_{i_2}, y_i), where $y_i = 1$ if points (x_{i_1}, x_{i_2}) belong to the same class (positive constraint) and $y_i = -1$ if these points belong to different classes (negative constraint). The *DistBoost* algorithm is a semi-supervised algorithm that uses unlabeled data points in $\mathcal{X}$ and equivalence constraints to learn a bounded distance function, $\mathcal{D} : \mathcal{X} \times \mathcal{X} \to [0, 1]$, that maps each pair of points to a real number in $[0, 1]$.

The algorithm makes use of the observation that equivalence constraints on points in $\mathcal{X}$ are binary labels in the product space, $\mathcal{X} \times \mathcal{X}$. By posing the problem in product space we obtain a classical binary classification problem: an optimal classifier should assign $+1$ to all pairs of points that come from

Algorithm 1 The *DistBoost* Algorithm

Input:

Data points: $(x_1, ..., x_n)$, $x_k \in \mathcal{X}$

A set of equivalence constraints: (x_{i_1}, x_{i_2}, y_i), where $y_i \in \{-1, 1\}$

Unlabeled pairs of points: $(x_{i_1}, x_{i_2}, y_i = *)$, implicitly defined by all unconstrained pairs of points

- Initialize $W^1_{i_1 i_2} = 1/(n^2)$ $i_1, i_2 = 1, \ldots, n$ (weights over pairs of points)
  $w_k = 1/n$ $k = 1, \ldots, n$ (weights over data points)
- For $t = 1, .., T$
  1. Fit a constrained GMM (weak learner) on weighted data points in $\mathcal{X}$ using the equivalence constraints.
  2. Generate a weak hypothesis $\tilde{h}_t : \mathcal{X} \times \mathcal{X} \rightarrow [-1, 1]$ and define a weak distance function as $h_t(x_i, x_j) = \frac{1}{2}\left(1 - \tilde{h}_t(x_i, x_j)\right) \in [0, 1]$
  3. Compute $r_t = \sum_{(x_{i_1}, x_{i_2}, y_i = \pm 1)} W^t_{i_1 i_2} y_i h_t(x_{i_1}, x_{i_2})$, only over **labeled** pairs. Accept the current hypothesis only if $r_t > 0$.
  4. Choose the hypothesis weight $\alpha_t = \frac{1}{2}\ln(\frac{1+r_t}{1-r_t})$
  5. Update the weights of **all** points in $\mathcal{X} \times \mathcal{X}$ as follows:

$$W^{t+1}_{i_1 i_2} = \begin{cases} W^t_{i_1 i_2} \exp(-\alpha_t y_i \tilde{h}_t(x_{i_1}, x_{i_2})) & y_i \in \{-1, 1\} \\ W^t_{i_1 i_2} \exp(-\alpha_t) & y_i = * \end{cases}$$

  6. Normalize: $W^{t+1}_{i_1 i_2} = \frac{W^{t+1}_{i_1 i_2}}{\sum_{i_1, i_2 = 1}^{n} W^{t+1}_{i_1 i_2}}$
  7. Translate the weights from $\mathcal{X} \times \mathcal{X}$ to $\mathcal{X}$: $w^{t+1}_k = \sum_j W^{t+1}_{kj}$

Output: A final distance function $\mathcal{D}(x_i, x_j) = \sum_{t=1}^{T} \alpha_t h_t(x_i, x_j)$

the same class, and -1 to all pairs of points that come from different classes. This binary classification problem can be solved using traditional margin based classification techniques. Note, however, that in many real world problems, we are only provided with a sparse set of equivalence constraints and therefore the margin based binary classification problem is semi-supervised.

DistBoost learns a distance function using a well known machine learning technique, called *Boosting* [12]. In Boosting, a set of "weak" learners are iteratively trained and then linearly combined to produce a "strong" learner. Specifically, *DistBoost*'s weak learner is based on the constrained Expectation Maximization (cEM) algorithm [13]. The cEM algorithm is used to generate a "weak" distance function. The final ("strong") distance function is a weighted sum of a set of such "weak" distance functions. The algorithm is illustrated in Fig. 3, and presented in Alg. 1.

In order to make use of unlabeled data points, *DistBoost*'s weak learner is trained in the original space, $\mathcal{X}$, and is then used to generate a "weak distance function" on the product space. *DistBoost* uses an augmentation of the 'Ad-

aboost with confidence intervals' algorithm [12] to incorporate unlabeled data into the boosting process. Our semi-supervised boosting scheme computes the weighted loss only on **labeled** pairs of points but updates the weights over **all** pairs of points (see steps (3-6)). The weights of the unlabeled pairs decay at least as fast as the weight of any labeled pair. The translation scheme from product space to the original space and vice-versa is presented in steps (2,7) of the algorithm.

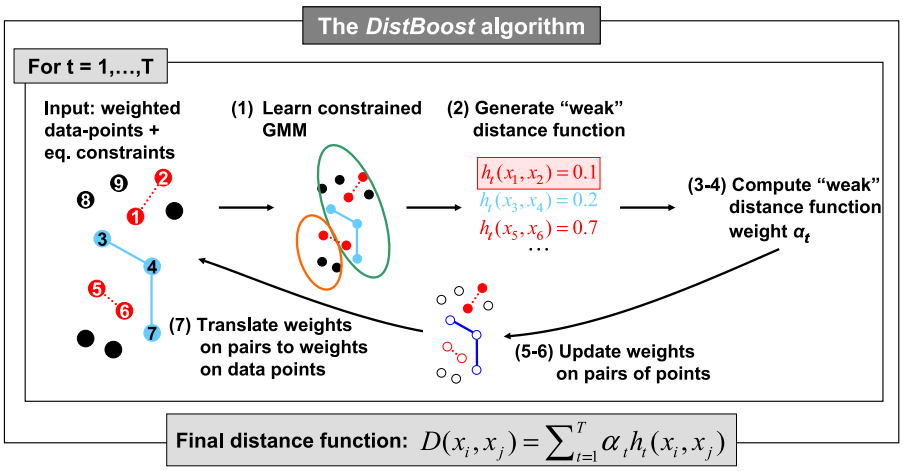


Fig. 3. An illustration of the *DistBoost* algorithm. At each boosting round t the weak learner is trained using weighted input points and some equivalence constraints. In the example above, points $1,2$ and $5,6$ are negatively constrained (belong to different classes) and points $3,4$ and $4,7$ are positively constrained (belong to the same class). All other pairs of points (e.g $8,9$ and $1,4$) are unconstrained. The constrained EM algorithm is used to learn a GMM (step (1)). This GMM is then used to generate a "weak" distance function (step (2)) that assigns a value in $[0,1]$ to each pair of points. The distance function is assigned a hypothesis weight (steps (3-4)) which corresponds to its success in satisfying the current weighted constraints. The weights of the equivalence constraints are updated (steps (5-6)) – increasing the weights of constraints that were unsatisfied by the current weak learner. Finally, the weights on pairs are translated to weights on data points (step (7)). In the example above, the distance between the negatively constrained points $1,2$ is small (0.1) and therefore the weight of this constraint will be enhanced.

***DistBoost*'s weak learner:** *DistBoost*'s weak learner is based on the constrained Expectation Maximization (cEM) algorithm [13]. The algorithm uses unlabeled data points and a set of equivalence constraints to find a Gaussian Mixture Model (GMM) that complies with these constraints. A GMM is a parametric statistical model which is given by $p(x|\Theta) = \Sigma_{l=1}^{M}\pi_l p(x|\theta_l)$, where π_l denotes the weight of each Gaussian, θ_l its parameters, and M denotes the number of Gaussian sources in the GMM. Under this model, each data sample originates independently from a weighted sum of several Gaussian sources. Estimating the parameters (Θ) of a GMM is usually done using the well known EM algorithm. The cEM algorithm introduces equivalence constraints by modifying

the 'E' (Expectation) step of the algorithm: instead of summing over *all* possible assignments of data points to sources, the expectation is taken only over assignments which comply with the given equivalence constraints.

The algorithm's input is a set of unlabeled points $X = \{x_i\}_{i=1}^n$, and a set of pairwise constraints, Ω, over these points. Denote positive constraints by $\{(p_j^1, p_j^2)\}_{j=1}^{N_p}$ and negative constraints by $\{(n_k^1, n_k^2)\}_{k=1}^{N_n}$. Let $H = \{h_i\}_{i=1}^n$ denote the hidden assignment of each data point x_i to one of the Gaussian sources ($h_i \in \{1, \ldots, M\}$). The constrained EM algorithm assumes the following joint distribution of the observables X and the hiddens H:

$$p(X, H|\Theta, \Omega) = \frac{1}{Z} \prod_{i=1}^{n} \pi_{h_i} p(x_i|\theta_{h_i}) \prod_{j=1}^{N_p} \delta_{h_{p_j^1} h_{p_j^2}} \prod_{k=1}^{N_n} (1 - \delta_{h_{n_k^1} h_{n_k^2}}) \qquad (2)$$

where Z is the normalizing factor and δ_{ij} is Kronecker's delta. The algorithm seeks to maximize the data likelihood, which is the marginal distribution of (2) with respect to H. For a more detailed description of this weak learner see [13].

In order to use the algorithm as a weak learner in our boosting scheme, we modified the algorithm to incorporate weights over the data samples. These weights are provided by the boosting process in each round (see Alg. 1 step 7).

Generating a weak distance function using a GMM: The weak learners' task is to provide a weak distance function $h_t(x_i, x_j)$ over the product space $\mathcal{X} \times \mathcal{X}$. Denote by $p^{MAP}(x_i)$ the probability of the Maximum A-Posteriori (MAP) assignment of point x_i ($p^{MAP}(x_i) = \max_m p(h_i = m|x_i, \Theta)$). We partition the data into M groups using the MAP assignment of the points and define

$$\tilde{h}_t(x_i, x_j) \equiv \begin{cases} +p^{MAP}(x_i) \cdot p^{MAP}(x_j) & \text{if} \quad \text{MAP}(x_i) = \text{MAP}(x_j) \\ -p^{MAP}(x_i) \cdot p^{MAP}(x_j) & \text{if} \quad \text{MAP}(x_i) \neq \text{MAP}(x_j) \end{cases}$$

The weak distance function is given by $h_t(x_i, x_j) = \frac{1}{2}\left(1 - \tilde{h}_t(x_i, x_j)\right) \in [0, 1]$. It is easy to see that if the MAP assignment of two points is identical their distance will be in $[0, 0.5]$ and if their MAP assignment is different their distance will be in $[0.5, 1]$.

5 Results

MHC datasets The first two datasets we compiled (MHCclass1 and MHCclass2) were the same as those described in [6]. Sequences of peptides, that bind to MHC class I or class II molecules, were collected from the MHCPEP dataset [14]. Each entry in the MHCPEP dataset contains the peptide sequence, its MHC specificity and, where available, observed activity and binding affinity. Peptides, that are classified as low binders or contain undetermined residues (denoted by the letter code X), were excluded. We then grouped all 9 amino acid long peptides (9-mers), that bind to MHC class I molecules, to a dataset, called *MHCclass1*. This dataset consists of binding peptides for 25 different MHC class I molecules.

Unlike MHC class I binding peptides, peptides binding to MHC class II molecules display a great variability in length, although only a peptide core of 9 residues fits into the binding groove. Following [6], we first used the MEME program [15] to align the binding peptides for each molecule, based on a single 9 residues motif. We finally filtered out redundant peptides and obtained the *MHCclass2* dataset. This dataset consists of binding peptides for 24 different MHC class II molecules.

Since all peptides in the MHCPEP dataset are binders, we added randomly generated peptides as non-binders to both MHCclass1 and MHCclass2 datasets (amino acid frequencies as in the Swiss-Prot database). The number of non-binders used in any test set was twice the number of the binding peptides. During the train phase, the number of non-binders was the same as the number of binders.

In order to assess the performance of the prediction algorithms on experimentally determined non-binders, we compiled a third dataset, called *MHCclass1BN*. This dataset consists of binding and non-binding peptides, for 8 different MHC class I molecules, based on the *MHCBN 3.1* website [16] (see Fig. 8 (b)).

Data representation *DistBoost* requires that the data be represented in some continuous vector feature space. Following [17] each amino acid was encoded using a 5-dimensional property vector (and, thus, each peptide in the MHC datasets is a point in $\mathbb{R}^{45}$). The property vectors for each of the 20 amino acids are based on multidimensional scaling of 237 physical-chemical properties. Venkatarajan and Braun's analysis [17] showed that these 5 properties correlate well with hydrophobicity, size, α-helix preference, number of degenerate triplet codons and the frequency of occurrence of amino acid residues in β-strands. They also showed that the distances between pairs of amino-acids in the 5-dimensional property space are highly correlated with corresponding scores from similarity matrices derived from sequence and 3D structure comparisons.

Evaluation methods In order to evaluate the algorithms' performance, we measured the affinity of all test peptides to each of the proteins. We present the prediction accuracy (that is how well binders are distinguished from non-binders) of the various algorithms as ROC (Receiver Operating Characteristic) curves. The fraction of the area under the curve (AUC) is indicative of the distinguishing power of the algorithm and is used as its prediction accuracy.

5.1 MHC binding prediction on the MHCPEP dataset

We compared our method to the recently enhanced RANKPEP method [6]. We replicated the exact experimental setup described in [6]: (1) We used the exact same MHC class I and class II datasets. (2) Training was performed using 50% of the known binders for each of the MHC molecules. (3) The remaining binding peptides were used as test data to evaluate the algorithm's performance. These binders were tested against randomly generated peptides as described above.

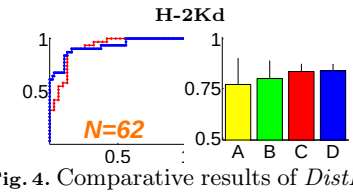


Fig. 4. Comparative results of *DistBoost* and RANKPEP on the H-2Kd MHC class I molecule. The left plot presents ROC curves of the best train score obtained when training on 50% of the entire data (red - using only positive constraints and blue - using both types of constraints). The right plot presents average AUC scores on test data. We compare the two PSSM methods used by RANKPEP (A - PROFILEWEIGHT, B - BLK2PSSM) to *DistBoost* when trained using only positive constraints (C) and when trained using both positive and randomly generated negative constraints (D). The averages were taken over 10 different runs on randomly selected train and test sets. N denotes the total number of binding peptides (of which 50% were used in the training phase and the remaining 50% were used in the test phase).

We trained *DistBoost* in two distinct scenarios: (1) Training using only binding peptides (using only positive constraints). (2) Training using both binding and (randomly generated) non-binding peptides (using both positive and negative constraints). In both scenarios *DistBoost* was trained **simultaneously** on all of the MHC molecules in each class. Fig. 4 presents a comparison of *DistBoost* to both of the PSSM's used in [6] on the H-2Kd MHC class I molecule. Comparative results on the entire MHC class I and class II datasets are presented in Figures 5 and 6. In all these comparisons, the PSSM AUC scores are as reported in [6].

On the MHC class I molecules, our method significantly outperforms both PSSM's used by RANKPEP. On 21 out of the 25 molecules *DistBoost*'s average AUC score, when trained using only positive constraints, is higher than both PSSM methods. The improvement in performance is more pronounced on molecules with relatively small amounts of known binders (e.g. HLA-B27(B*2704) - 10 binders, HLA-A2(A*0205) - 22 binders and HLA-A33(A*3301) - 23 binders). One possible explanation of these results is that the information provided by other proteins within the protein family is used to enhance prediction accuracy, especially in these cases where only small amounts of known binders exist. Additionally, it may be seen that using both positive and negative constraints on this dataset, usually improves the algorithm's performance. Another important advantage of *DistBoost* can be seen when comparing standard deviations (std) of the AUC scores. On 16 out of the 25 molecules the algorithm's std is lower then the std of both PSSM's, implying that our method is more robust.

When tested on the MHC class II molecules, our method obtained similar improvements (see Fig. 6): On 18 out of the 24 molecules *DistBoost*'s average AUC score when trained using only positive constraints is higher then both PSSM methods. In general, it appears that the performance of all of the compared methods is lower than on the MHC class I dataset. It is known that predicting binding affinity on MHC class II is more challenging, partially due to the fact that peptides that bind to class II molecules are extremely variable in length and share very limited sequence similarity [18]. On this dataset, the use of both positive and negative constraints only slightly improved *DistBoost*'s performance (13 out of 24 molecules) over the use of positive constraints only.

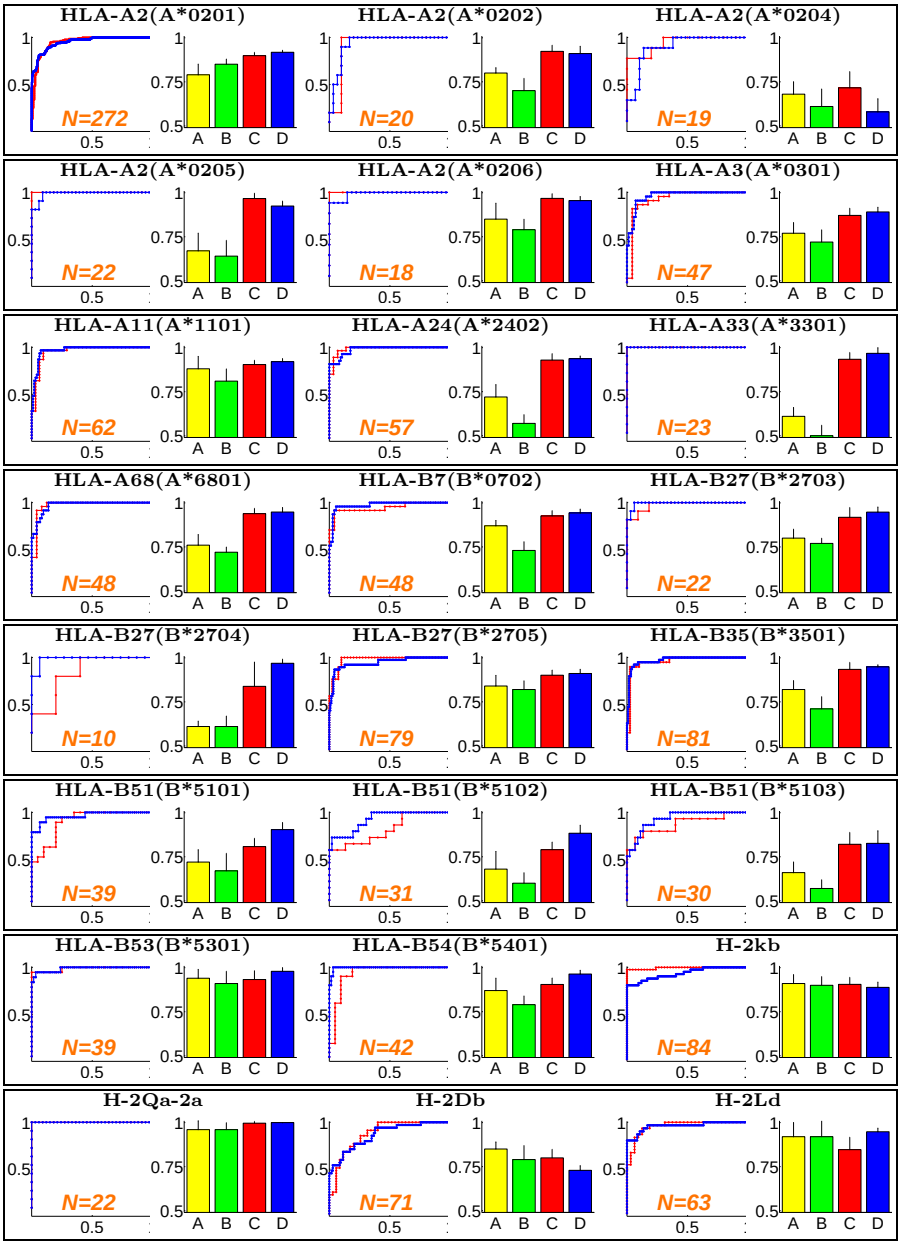


Fig. 5. Comparative results of *DistBoost* (left plot and bars C-D) and RANKPEP (bars A-B) on 24 MHC class I molecules. Plot legends are identical to Fig 4. On 21 out of the 25 molecules (including Fig. 4), *DistBoost* outperforms both PSSM methods. On this data the use of negative constraints also improves performance.

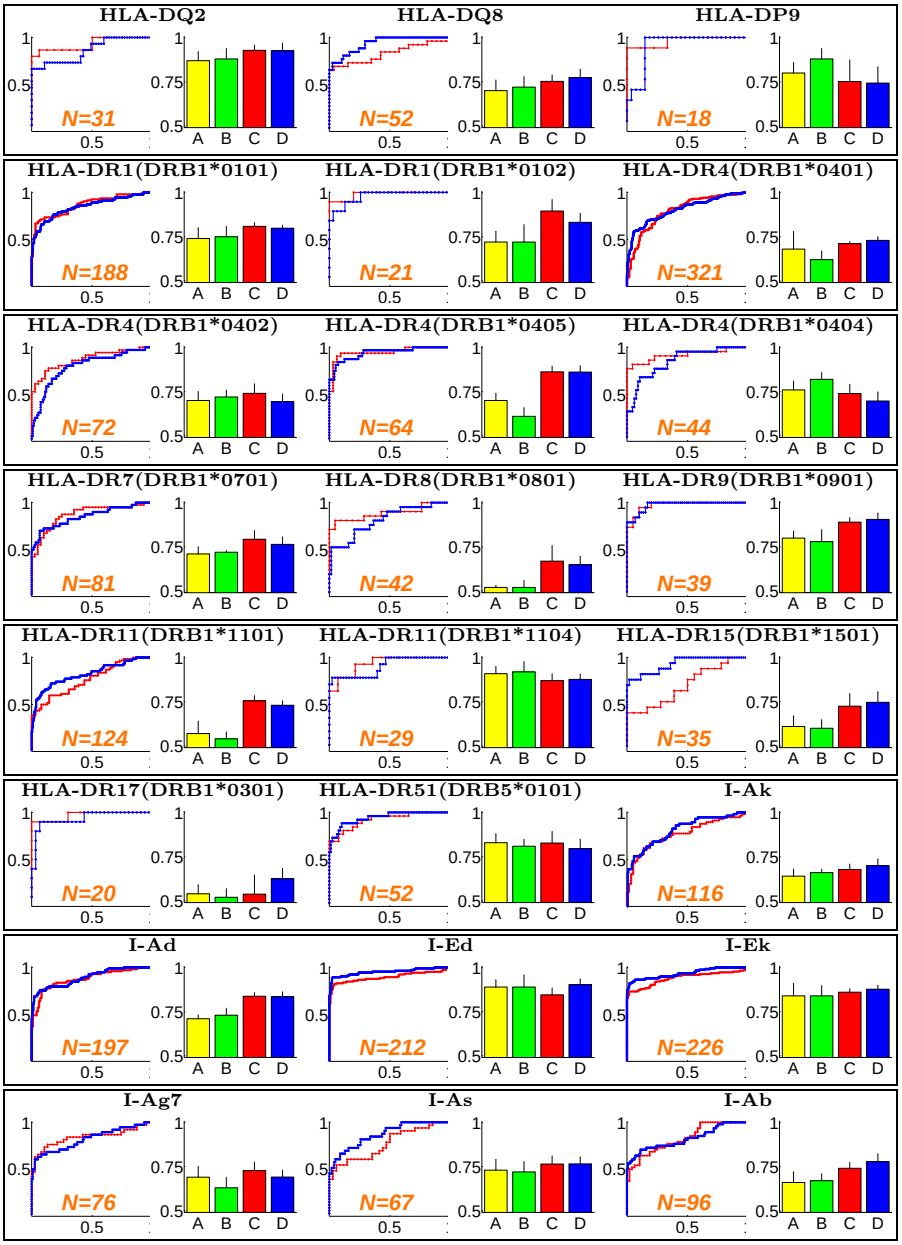


Fig. 6. Comparative results of *DistBoost* (left plot and bars C-D) and RANKPEP (bars A-B) on 24 MHC class II molecules. Plot legends are identical to Fig 4. As may be seen on 18 out of the 24 molecules, *DistBoost* outperforms both PSSM methods. On this dataset the use of negative constraints only slightly improves performance.

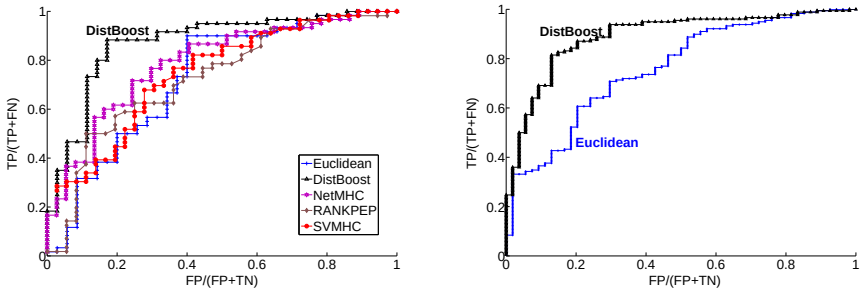


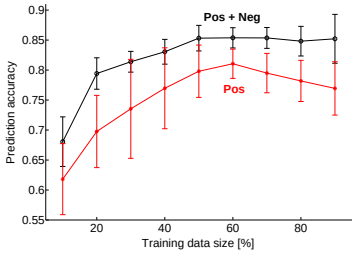
Fig. 7. (**a**) ROC curves on test data from the HLA-A2 supertype. *DistBoost* is compared to various other prediction algorithms. (**b**) *DistBoost* and the Euclidean affinity ROC curves on test data from the **entire** *MHCclass1BN* dataset. The rest of the methods are not presented since they were not trained in this multi-protein scenario. In both cases, *DistBoost* was trained on 70% of the data and tested on the remaining 30%. TP (true positives), FP (false positives), TN (true negatives), FN (false negatives). Results are best seen in color.

5.2 MHC class I binding prediction on the MHCBN dataset

The MHCPEP dataset only contains information about peptides that bind to various MHC molecules. In contrast, the MHCBN dataset also contains information about non-binding peptides for some MHC class I molecules. We used this dataset to evaluate the importance of learning using experimentally determined non-binders (as opposed to randomly generated non binders).

The MHCBN dataset also contains information about binding and non-binding peptides to supertypes which are collections of similar molecules. We compared *DistBoost* to various other computational prediction methods on the HLA-A2 supertype. Specifically we compared the performance of the following methods: **(1)** The *DistBoost* algorithm. **(2)** The *SVMHC* web server [9]. **(3)** The *NetMHC* web server [10]. **(4)** The RANKPEP resource [6] **(5)** The Euclidean distance metric in $\mathbb{R}^{45}$. Despite the fact that methods 2-4 are protein specific they also provide predictions on various MHC supertypes including the HLA-A2 supertype.

We trained *DistBoost* on 70% of the **entire** *MHCclass1BN* data (including binding and non-binding peptides) and compared its performance to all other methods on the **single** HLA-A2 supertype. The test set, therefore, consists of the remaining 30% of HLA-A2 data. The results are shown in Fig. 7 (a). As may be seen *DistBoost* outperforms all other methods, including SVMHC, NetMHC and RANKPEP, which were trained on this specific supertype. However, it is important to note, that unlike *DistBoost* all of these methods were trained using randomly generated non-binders. The performance of all of these methods when tested against random peptides is much better — AUC scores of 0.947 (SVMHC), 0.93 (NetMHC) and 0.928 (RANKPEP). These results seem to imply that learning using random non-binders does **not** generalize well to experimentally determined non-binders. Interestingly, when we tested *DistBoost* on randomly generated non-binders we obtained an AUC score of 0.923. We can



(a)

MHCclass1BN dataset				
MHC class I molecule	Binders mod.	high	?	Non binders
HLA-A2 supertype	29	42	126	116
HLA-A68 (A*6801)	10	27	17	0
HLA-B27 (B*2705)	7	17	11	7
HLA-B35 (B*3501)	31	47	48	6
HLA-B53 (B*5301)	17	16	22	0
H-2Db	18	11	17	26
H-2Kb	6	7	4	19
H-2Ld	9	16	25	0

(b)

Fig. 8. (**a**) Learning curves of *DistBoost* trained using only positive constraints (*Pos*) and using both types of constraints (*Pos+Neg*). Prediction accuracy based on the AUC score, averaged over 20 different randomly selected training sets. (**b**) The MHCclass1BN dataset.

therefore conclude that learning from "real" non-binders generalizes very well to random non-binders.

Our proposed method is trained **simultaneously** on a number of proteins from the same family, unlike methods (2-4). However, our final predictions are protein specific. As the results reveal, we obtain high binding prediction accuracy when tested on a single protein (see Fig. 7 (a)). In order to quantify the overall protein specific binding prediction accuracy, we present ROC curves for *DistBoost* and the Euclidean affinity functions when tested on the **entire** MHCclass1BN dataset (Fig. 7 (b)). The peptide-peptide distance matrices and the protein-peptide affinity matrices of these two methods are presented in Fig. 2. On this dataset *DistBoost* obtained excellent performance.

In order to evaluate the stability and learning power of *DistBoost* we ran it on the *MHCclass1BN* dataset, while varying the percentage of training data. Fig. 8 (a), presents the algorithm's learning curves when trained using only positive constraints and when trained using both positive and negative constraints. As may be expected, on average, performance improves as the amount of training data increases. Note that *DistBoost* achieves almost perfect performance with relatively small amounts of training data. Additionally, we can see that on this dataset learning from both types of constraints dramatically improves performance.

6 Discussion

In this paper we showed how the protein-peptide binding affinity problem can be cast as a semi-supervised learning problem in which equivalence constraints can be naturally obtained from empirical data. We then proposed to learn protein peptide affinity by learning a peptide-peptide distance function. Specifically we presented *DistBoost*, an algorithm that can learn distance functions using positive and negative equivalence constraints. Our experiments suggest that binding prediction based on such learned distance functions exhibits excellent performance. We also showed the importance of using negative constraints, which

further suggests that information about non-binding peptides should also be published and made publicly available.

Acknowledgements: We thank Daphna Weinshall for many fruitful discussions and comments. C.Y is supported by Yeshaya Horowitz Association through the Center for Complexity Science.

References

1. Hertz, T., Bar-Hillel, A., Weinshall, D.: Learning distance functions for image retrieval. In: CVPR, Washington DC, June 2004. (2004)
2. Hertz, T., Bar-Hillel, A., Weinshall, D.: Boosting margin based distance functions for clustering. In: ICML. (2004)
3. Rammensee., H.G., Friede., T., Stevanoviic, S.: MHC ligands and peptide motifs: first listing. Immunogenetics **41(4)** (1995) 178–228
4. Wang., R.F., Rosenberg, S.A.: Human tumor antigens for cancer vaccine development. Immunol Rev. **170** (1999) 85–100
5. Andersen, M., Tan, L., Sondergaard, I., Zeuthen, J., Elliott, T., Haurum, J.: Poor correspondence between predicted and experimental binding of peptides to class I MHC molecules. Tissue Antigens **55** (2000) 519–531
6. Reche, P.A., Glutting, J., Zhang, H., Reinher, E.: Enhancement to the RANKPEP resource for the prediction of peptide binding to MHC molecules using profiles. Immunogenetics **26** (2004) 405–419
7. Schueler-Furman, O., Altuvia, Y., Sette, A., Margalit, H.: Structure-based prediction of binding peptides to MHC class I molecules: application to a broad range of MHC alleles. Protein Sci **9** (2000) 1838–1846
8. Mamitsuka, H.: Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models. Proteins **33** (1998) 460–474
9. Donnes, P., Elofsson, A.: Prediction of MHC class I binding. BMC Bioinformatics **3** (2002)
10. Buus, S., Lauemoller, S., Worning, P., Kesmir, C., Frimurer, T., Corbet, S., Fomsgaard, A., Hilden, J., Holm, A., Brunak, S.: Sensitive quantitative predictions of peptide-MHC binding by a 'query by committee' artificial neural network approach. Tissue Antigens **62** (2003) 378–384
11. Flower, D.R.: Towards in silico prediction of immunogenic epitopes. TRENDS in immunology **24** (2003)
12. Schapire, R.E., Singer, Y.: Improved boosting using confidence-rated predictions. Machine Learning **37** (1999) 297–336
13. Shental, N., Bar-Hilel, A., Hertz, T., Weinshall, D.: Computing Gaussian mixture models with EM using equivalence constraints. In: NIPS. (2003)
14. Brusic, V., Rudy, G., Harrison, L.: MHCPEP, a database of MHC-binding peptides: update 1997. Nucl. Acids Res. **26** (1998) 368–371
15. Bailey, T., Elkan, C.: Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In: ISMB. Volume 2. (1994) 28–36
16. Bhasin, M., Singh, H., Raghava, G.P.S.: MHCBN: a comprehensive database of MHC binding and non-binding peptides. Bioinformatics **19** (2003) 665–666 http://www.imtech.res.in/raghava/mhcbn/index.html.
17. Venkatarajan, M.S., Braun, W.: New quantitative descriptors of amino acids based on multidimensional scaling of a large number of physical-chemical properties. Journal of Molecular Modeling **7** (2001) 445–453
18. Madden, D.R.: The three-dimensional structure of peptide-MHC complexes. Annual Review of Immunology **13** (1995) 587–622