

Proceedings

Open Access

PepDist: A New Framework for Protein-Peptide Binding Prediction based on Learning Peptide Distance Functions

Tomer Hertz*^{1,2} and Chen Yanover*¹

Address: ¹School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel, 91904 and ²The Center for Neural Computation, The Hebrew University of Jerusalem, Jerusalem, Israel, 91904

Email: Tomer Hertz* - tomboy@cs.huji.ac.il; Chen Yanover* - chen@cs.huji.ac.il

* Corresponding authors

from NIPS workshop on New Problems and Methods in Computational Biology
Whistler, Canada. 18 December 2004

Published: 20 March 2006

BMC Bioinformatics 2006, 7(Suppl 1):S3 doi:10.1186/1471-2105-7-S1-S3

Abstract

Background: Many different aspects of cellular signalling, trafficking and targeting mechanisms are mediated by interactions between proteins and peptides. Representative examples are MHC-peptide complexes in the immune system. Developing computational methods for protein-peptide binding prediction is therefore an important task with applications to vaccine and drug design.

Methods: Previous learning approaches address the binding prediction problem using traditional margin based binary classifiers. In this paper we propose *PepDist*: a novel approach for predicting binding affinity. Our approach is based on learning peptide-peptide distance functions. Moreover, we suggest to 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.

Results: 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, our method's performance gain, when compared to other computational methods, is even more pronounced. We have recently uploaded the *PepDist* webserver which provides binding prediction of peptides to 35 different MHC class I alleles. The webserver which can be found at <http://www.pepdist.cs.huji.ac.il> is powered by a prediction engine which was trained using the framework presented in this paper.

Conclusion: The results obtained suggest that learning a *single* distance function over an entire family of proteins achieves higher prediction accuracy than learning a set of binary classifiers for each of the proteins separately. We also show the importance of obtaining information on experimentally determined non-binders. Learning with real non-binders generalizes better than learning with randomly generated peptides that are assumed to be non-binders. This suggests that information about non-binding peptides should also be published and made publicly available.

Background

Many different aspects of cellular signalling, trafficking and targeting mechanisms are mediated by interactions between proteins and peptides. In the immune system, for example, the major task of recognizing foreign pathogen proteins is mediated by interactions between Major Histocompatibility Complex (MHC) molecules and short pathogen-derived peptides (see Fig. 1). T-cells recognize these peptides only when they are bound to MHC molecules. Understanding the underlying principles of MHC-peptide interactions is therefore a problem of fundamental importance, with applications to vaccine and drug design [3]. MHC binding is a challenging problem because MHC molecules exhibit high specificity – it is estimated that each molecule only binds to 1% of all existing peptides [4]. Additionally, MHC molecules are highly polymorphic and polygenic – there are hundreds of different alleles in the entire population while each individual carries a few alleles only (up to 6 MHC class I alleles and up to 12 MHC class II alleles) [5].

Biochemical assays, which empirically test protein-peptide binding affinity, can nowadays provide a rather high throughput rate [6]. However, note that there are 20^L peptides of length L (for 9 amino-acid long peptides as in the MHC proteins this amounts to 10^{12} peptides) and a great number of proteins that need to be considered. Therefore, in recent years, there has been a growing interest in developing computational methods for protein-peptide binding prediction [7-13]. Formally, the protein-peptide binding prediction problem can be stated as follows: given a protein and a peptide, predict the binding affinity of the interaction between the two. Stated this way, the protein-peptide binding prediction is essentially a simplified version of the more general protein docking problem.

What should we expect from a "good" binding prediction algorithm? [8].

1. Classification: A good binding prediction algorithm should first and foremost correctly predict whether a query peptide (which was not provided during the training stage) binds or does not bind to the given protein.

2. Ranking: An even stronger requirement is that the algorithm could also obtain a relative binding score for each peptide that can be used to rank different peptides according to their specificity.

3. Affinity prediction: Ultimately, the algorithm's score would predict the precise binding affinity values as determined experimentally.

Clearly, current state-of-the-art prediction methods obtain promising *classification* results (for a recent com-

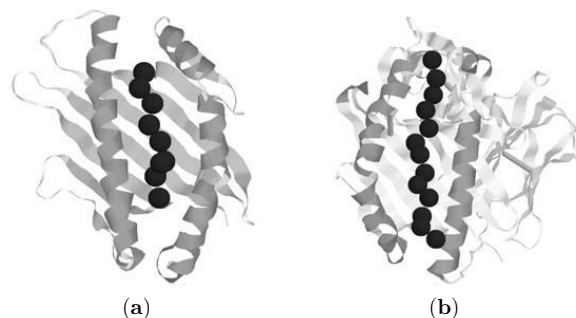


Figure 1

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

parison between several methods see [14]). Many of these methods also compute binding scores for each peptide, but these scores are in most cases not even compared to the empirically known affinity values, and have even been shown to have poor correspondences in some cases [15] (an interesting exception is a recent work on the PDZ domain [16]).

Most prediction algorithms formalize the protein-peptide binding prediction problem as a binary classification problem: For each protein (i.e. MHC molecule) a classifier is trained to distinguish binding peptides from non-binding peptides. After an initial training stage, the classifier is tested on a set of peptides, which were not presented during the training stage. The training data consists of experimentally determined binders and randomly generated peptides which are assumed to be non-binders. Interestingly enough, only rarely are experimentally determined non-binders used, mainly because a small number of these non-binders have been made publicly available.

In this paper we suggest a novel formulation of the protein-peptide binding prediction problem. Our approach is driven by the following two important observations:

Observation 1 *Peptides that bind to the same protein are "similar" to one another, and different from non-binding peptides.*

This observation underlies most, if not all, computational prediction methods. Motif based methods [8,13] for example, search for a binding motif that captures the similarity of a set of known binding peptides. Prediction is then based on the similarity of a query peptide to the motif, which implicitly measures the similarity of the

query peptide to the peptides in the training set. This observation suggests that given a novel peptide, one can predict its binding affinity to a specific protein, by explicitly measuring the average distance (or similarity) of the novel peptide to a list of known binding peptides. Intuitively speaking, if the peptide is close to the known binders, we would classify it as a binder, and if it is far – we would classify it as a non-binder.

Observation 2 *Peptides binding to different proteins within the same "family" resemble each other*

Proteins from the same family (e.g. MHC class I) are known to have structural and sequential similarity. Therefore they bind to peptides that share common characteristics. Additionally, in MHC class I and MHC class II, many proteins are grouped into supertypes [17] (such as the HLA-A2 MHC class I supertype). A supertype is a collection of proteins whose binding peptide sets are overlapping. This observation implies that we may benefit from *simultaneously* learning a single binding prediction function over an entire family of proteins, instead of independently learning a *single* classifier for each of the proteins within the protein family. At a first glance it might appear that one can recast a set of binary classification problems using a single multi-class classification problem. However, a closer look reveals that the protein-peptide binding problem is *not* a multi-class classification problem due to the following inherent facts: (1) Some peptides bind to several proteins within a family (indeed this information is used to define MHC supertypes). (2) A peptide that does not bind to a specific protein within a family, does not necessarily bind to a different protein within the family.

Our novel approach is based on the two observations described above. We propose to address the protein-peptide binding prediction problem by learning peptide-peptide distance functions. We do not require that the triangle inequality holds, and thus our distance functions are *not* necessarily metrics. Moreover, based on *observation 2*, we suggest to pool together information from an entire protein family and to learn a *single* peptide-peptide distance function (instead of learning a different distance function for every protein independently). Our peptide-peptide distance function is then used to compute protein-peptide binding affinity – the affinity of a query peptide to a given protein is inversely proportional to its average distance from all of the peptides known to bind to that protein. Our proposed learning scheme is summarized in Fig. 2 and elaborated in the following section.

Learning peptide distance functions

As mentioned above, we propose to address the protein-peptide binding affinity prediction problem by learning a

Input:

A dataset of binding and non binding peptides from an entire protein family.

1. For each protein: Extract “positive” and “negative” equivalence constraints using its known binding and non-binding peptides, respectively.
2. Learn a single peptide-peptide distance function over this dataset using the equivalence constraints extracted in step 1.
3. Define a protein-peptide affinity function using the peptide-peptide distance function from step 2.

Output:

A *single* protein-peptide affinity function over the entire protein family.

Figure 2
The *PepDist* framework.

peptide-peptide distance function over an *entire* family of proteins. A distance function \mathcal{D} assigns a non-negative value for each pair of points. Most algorithms that learn distance functions make use of equivalence constraints [1,2,18-22]. 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 this setting the goal of the algorithm is to learn a distance function that attempts to comply with the equivalence constraints provided as input.

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. Therefore, for each protein, our training data consists of a list of binding and non-binding peptides, and the set of equivalence constraints that they induce.

We collect these sets of peptides and equivalence constraints from several proteins within a protein family into a single dataset. We then use this dataset to learn a peptide-peptide distance function (see Fig. 3 left plots). Using this distance function, we can 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. 3 right plots). More formally, let us denote by $\mathcal{D}(Peptide_i, Peptide_k)$ the distance between $Peptide_i$ and $Peptide_k$ and by B_j the group of peptides known to bind to $Protein_j$. We define the affinity between $Peptide_i$ and $Protein_j$ to be:

$$Affinity(Peptide_i, Protein_j) \equiv \exp\left(-\frac{1}{|B_j|} \sum_{k \in B_j} \mathcal{D}(Peptide_i, Peptide_k)\right) \quad (1)$$

In order to learn peptide-peptide distance functions, we use the *DistBoost* algorithm [1,2], which learns distance functions using data and some equivalence constraints (see Methods for the algorithm's description). *DistBoost* requires that the data be represented in some continuous

vector feature space. We therefore represent each amino-acid using a 5-dimensional feature vector as suggested by [23], and each peptide by concatenating its amino-acid feature vectors (for further details see the Data representation section). 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 the improve-

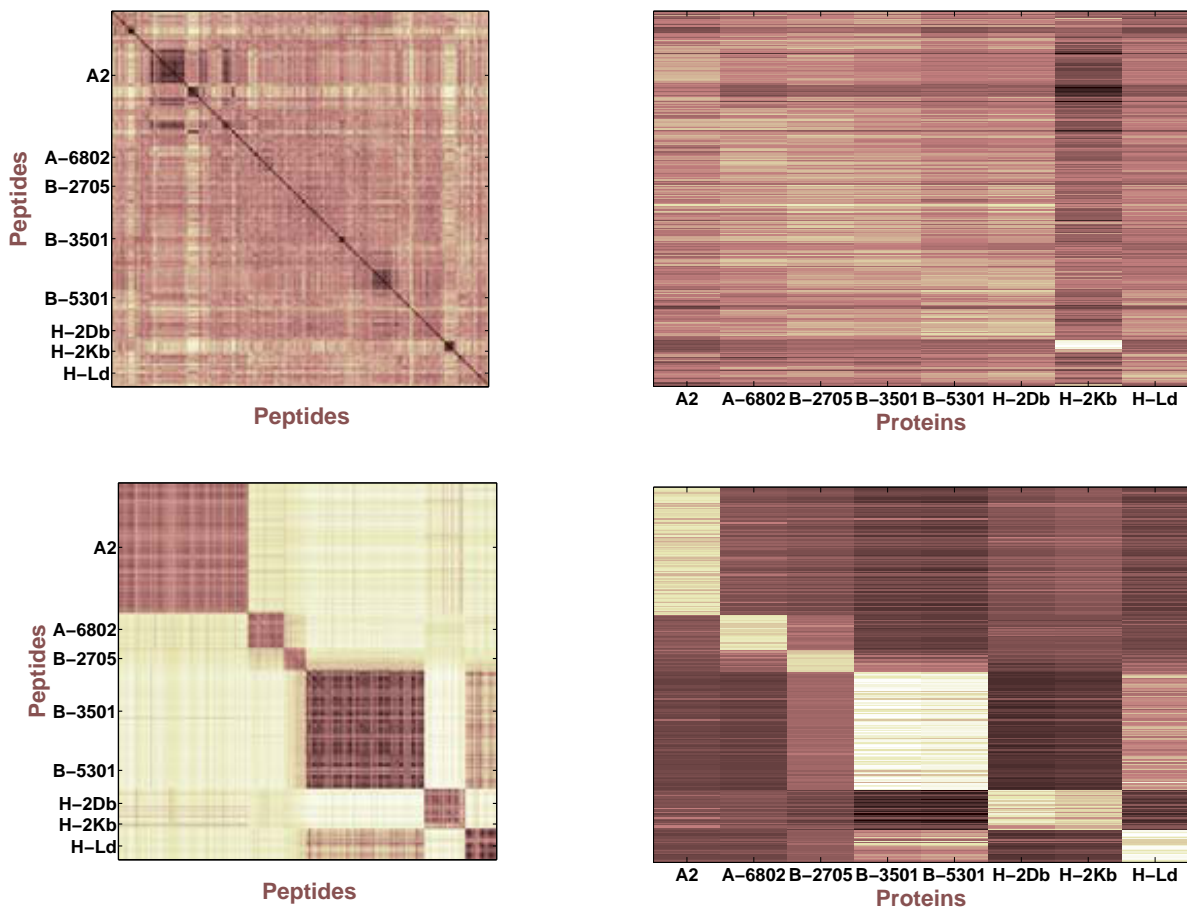


Figure 3

Left: peptide-peptide distance matrices of MHC class I binding peptides, collected from the MHCBN dataset. Peptides that bind to each of the proteins were grouped together and labeled accordingly. Following *Observation 1*, a "good" distance matrix should therefore be block diagonal. Top left: The Euclidean peptide-peptide distance matrix in \mathbb{R}^{45} (see Methods 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 proteins **simultaneously**.

ment 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.

Related work

Many different computational approaches have been suggested for the protein-peptide binding prediction problem (see [24] 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 [8,15]. 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 [13] 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 [9]. 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

Many different learning algorithms have been suggested for binding prediction. Among these are artificial neural networks (NetMHC) [12], Hidden Markov Models (HMM's) [10] and support vector machines (SVMHC) [11]. To the best of our knowledge all of these methods are trained separately for each protein (or supertype). Therefore, these methods work well when sufficient amounts of training data (i.e peptides which are known to be binders or non-binders for a given protein) is provided.

Results

We evaluated the performance of our method on several MHC class I and MHC class II datasets, and compared it to various other prediction methods (see Methods for details about these datasets). We begin with a thorough comparison of our method to the recently enhanced *RANKPEP* method [13] on MHC class I and class II datasets. In order to assess the importance of using experimentally determined non-binders, we tested our method on another MHC class I dataset collected from the MHCBN repository. On this dataset we also compare our method to various other MHC binding prediction methods.

MHC binding prediction on the MHCPEP dataset

We compared our method to the recently enhanced *RANKPEP* method [13]. We replicated the exact experimental setup described in [13]: (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.

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 [13]. 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, respectively. In all these comparisons, the PSSM AUC scores (See Methods for details) are as reported in [13].

On the MHC class I molecules, our method significantly outperforms both PSSM's used by *RANKPEP*. On 21 out

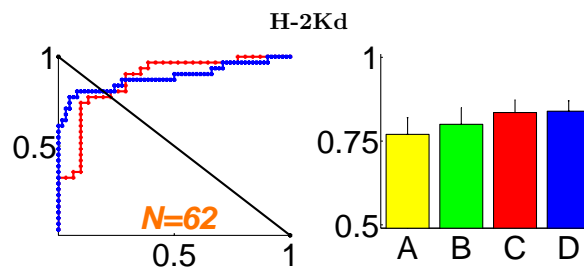


Figure 4

Comparative results of *DistBoost* and *RANKPEP* on the H-2Kd MHC class I molecule. The left plot presents ROC (see Evaluation methods section for details) curves of the best test score obtained when training on 50% of the entire data (red: using only positive constraints; blue: using both types of constraints). The intersection between the curves and the diagonal line marks the equal error-rate statistic. The right plot presents average AUC scores on test data. We compare the two PSSM methods used by *RANKPEP* (A: PROFILE-WEIGHT, B: BLK2PSSM) to *DistBoost* when trained using only positive constraints (C) and when trained using both positive and 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). For a detailed comparison see Figs. 5-6.

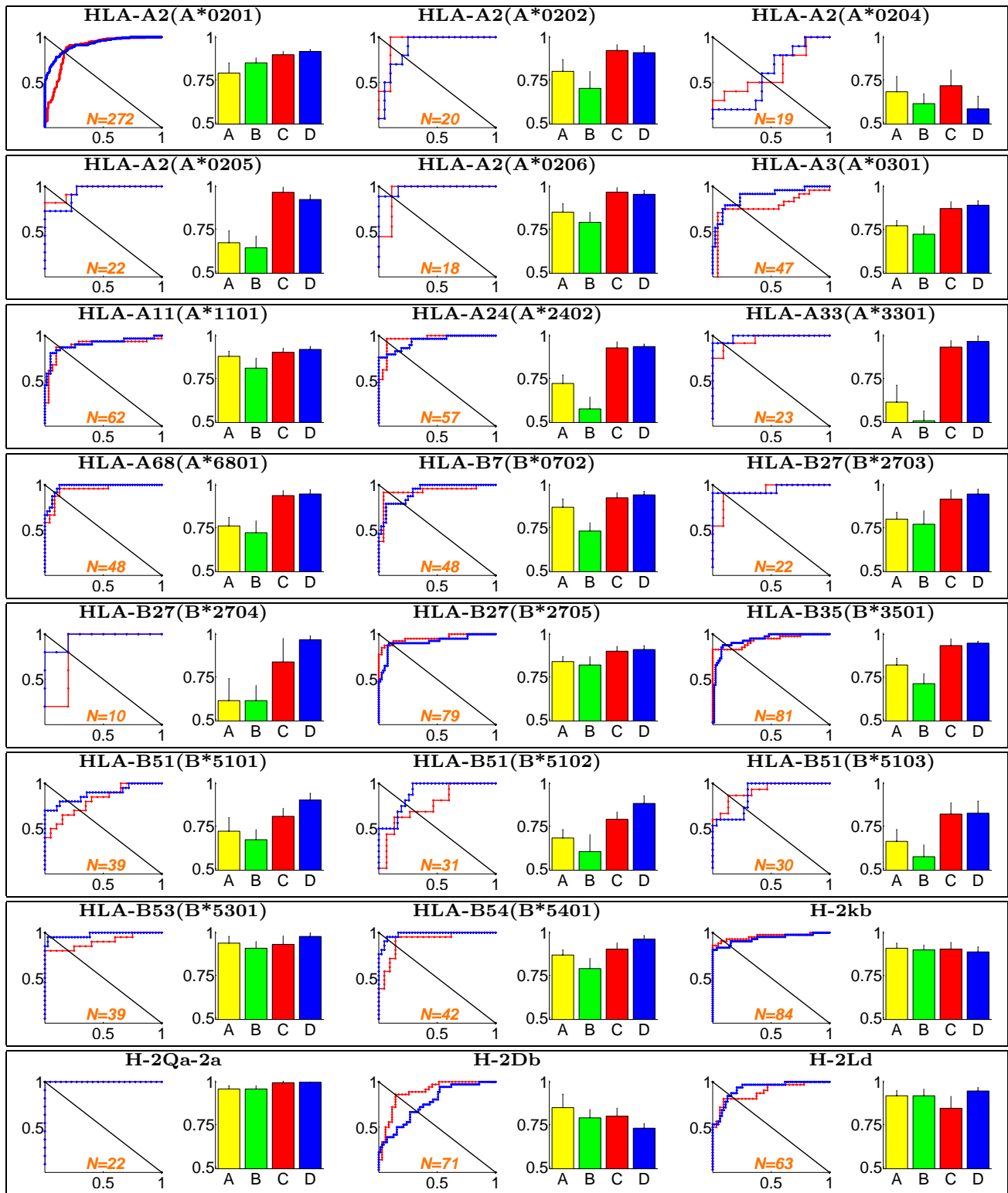


Figure 5

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. For numerical comparison, see additional file 1: [Pepdist_SupplementaryMaterials.ps](#), Table 1.

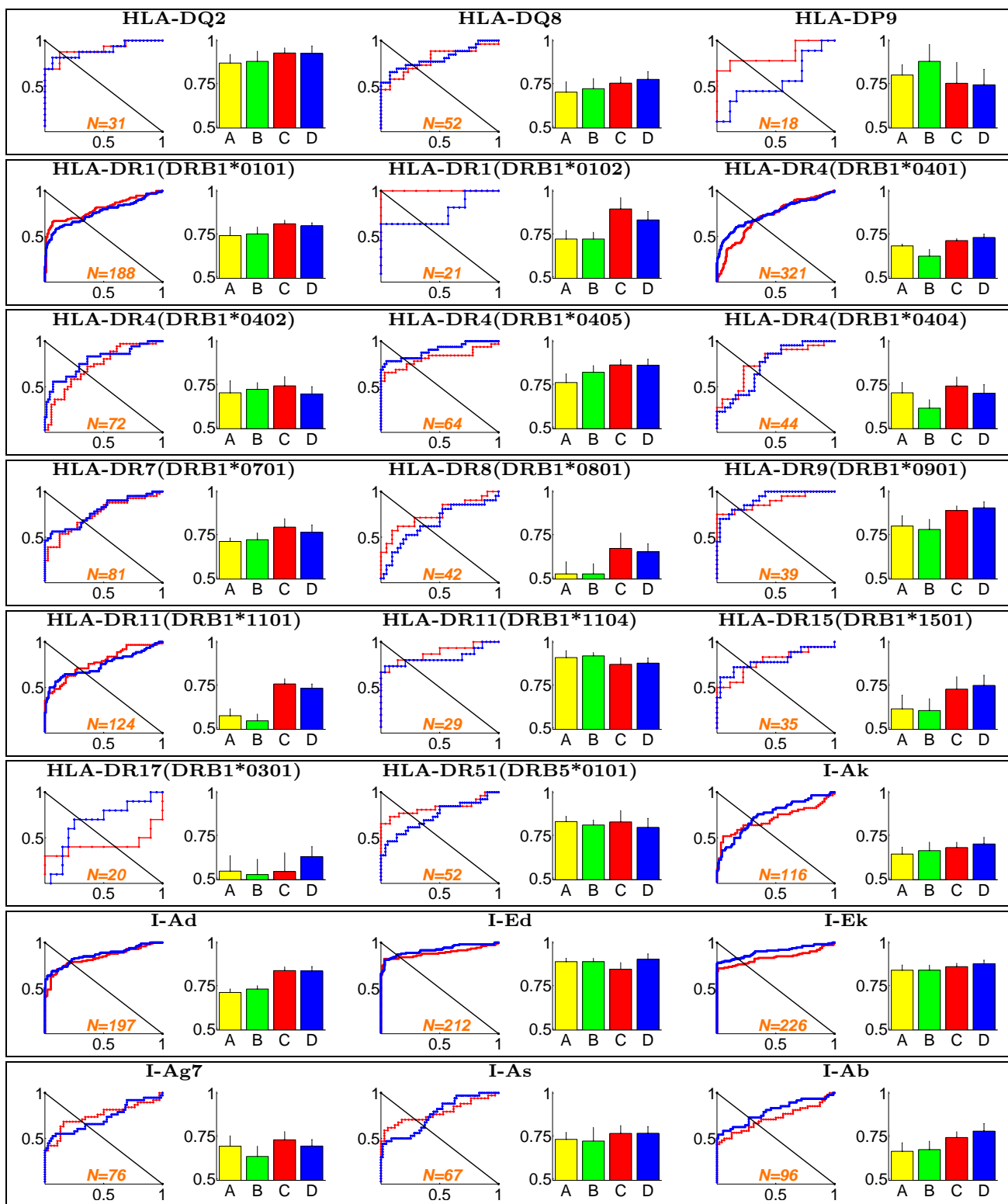


Figure 6

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 19 out of the 24 molecules, *DistBoost* outperforms both PSSM methods. On this dataset the use of negative constraints only slightly improves performance. For numerical comparison, see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 2.

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 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. When *DistBoost* was trained using only positive constraints, on 13 out of the 25 molecules the algorithm's std was lower than the std of both PSSM's. When *DistBoost* was trained using both positive and negative constraints, on 20 out of the 25 molecules the algorithm's std was lower than the std of both PSSM's. These results imply that our method is more robust.

When tested on the MHC class II molecules, our method obtained similar improvements (see Fig. 6): On 19 out of the 24 molecules *DistBoost*'s average AUC score when trained using only positive constraints is higher than 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 [25]. On this dataset, the use of both positive and negative constraints improved *DistBoost*'s performance on only 11 out of 24 molecules.

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).

We compared *DistBoost* to various other computational prediction methods on peptides that bind to the HLA-A2 supertype, collected from the MHCBN repository. Specifically, we compared the performance of the following methods: (1) The *DistBoost* algorithm. (2) The SVMHC web server [11]. (3) The *NetMHC* web server [12]. (4) The RANKPEP resource [13] (5) The Euclidean distance metric in \mathbb{R}^{45} . Despite the fact that methods (2–4) are protein

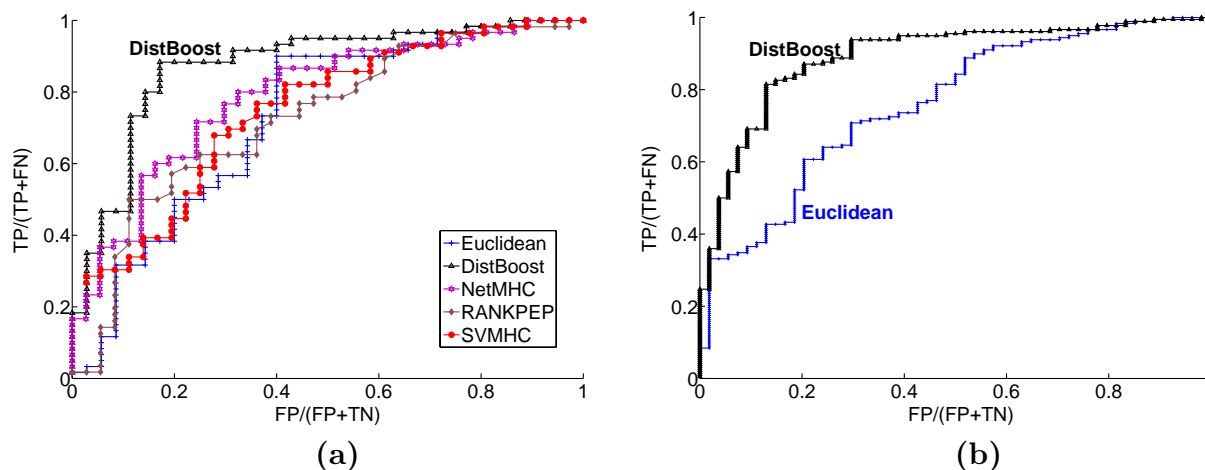
specific, they also provide predictions on various MHC supertypes including the HLA-A2 supertype.

We note that it is unclear whether the peptides collected from the MHCBN repository are the HLA-A2 supertype binders, or HLA-A*0201 binders which was named HLA-A2 in the older HLA nomenclature. When we compared our predictions to those of the SVMHC and NetMHC methods on the HLA-A*0201, similar results were obtained.

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 SVMHC: 0.947, NetMHC: 0.93 and RANKPEP: 0.928. When *DistBoost* was trained and tested using randomly generated non-binders it achieved an AUC score of 0.976. Interestingly, when *DistBoost* was trained using real non-binders and tested on randomly generated non-binders it obtained an AUC score of 0.923. These results seem to imply that learning using random non-binders does **not** generalize well to experimentally determined non-binders. On the other hand, 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. 3. 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 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

**Figure 7**

(a) ROC curves on test data from the HLA-A2 supertype. *DistBoost* is compared to the following algorithms: the *SVMHC* web server [11], the *NetMHC* web server [12], the RANKPEP resource [13] and the Euclidean distance metric in \mathbb{R}^{45} . (b) *DistBoost* and the Euclidean affinity ROC curves on test data from the **entire** *MHCclass I/BN* 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%. Results are best seen in color.

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.

The *PepDist* Webserver

Our proposed method is now publicly available through the *PepDist* webserver, which can be found at <http://www.pepdist.cs.huji.ac.il>. The current version provides binding predictions of 9-mer peptides to 35 different MHC class I alleles. The engine also supports multiple peptide queries. We hope to enhance the webserver in the near future to provide predictions for more MHC class I alleles and also for MHC class II alleles.

Discussion and Conclusion

In this paper we proposed *PepDist*: a novel formulation of the protein-peptide binding prediction problem that has two foundations. The first is to predict binding affinity by learning peptide-peptide distance functions. The second is to learn a *single* distance function over an entire family of proteins. Our formulation has several advantages over existing computational approaches:

1. Our method also works well on proteins for which small amounts of known binders are currently available.

2. Unlike standard binary classifiers, our method can be trained on an entire protein family using only information about binding peptides (i.e. without using real/randomly generated non-binders).

3. Our method can compute the relative binding affinities of a peptide to several proteins from the same protein family.

In order to learn such distance functions we casted the problem as a semi-supervised learning problem in which equivalence constraints can be naturally obtained from empirical data. Specifically, we used the *DistBoost* algorithm, that learns distance functions using positive and negative equivalence constraints. Our experiments suggest that binding prediction based on such learned distance functions exhibits excellent performance. It should be noted that our proposed learning scheme can be also implemented using other distance learning algorithms and in our future work we also plan to further investigate this idea. We also hope that the *PepDist* formulation will allow addressing the more challenging task of peptide ranking. One way of doing this is by incorporating information about relative binding values into the distance learning algorithm.

Our approach may also be useful for predicting some protein-protein interactions such as PDZ-protein complexes.

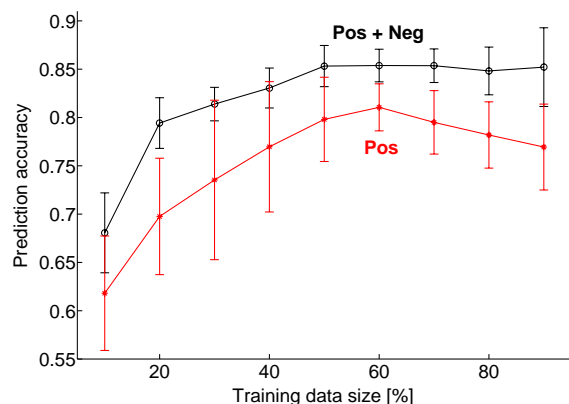


Figure 8
Learning curves of *DistBoost* trained using only positive constraints (*Pos*) and using both types of constraints (*Pos + Neg*). Prediction accuracy based on the AUC score, averaged over 20 different randomly selected training sets.

The PDZ domains are frequently occurring interaction domains involved in organizing signal transduction complexes and attaching proteins to the cytoskeleton [26]. In most cases, this is accomplished by specific recognition of the ligands' carboxyl termini (or regions "mimicking" the structure of a carboxyl terminal). Therefore, predicting whether a protein binds to a specific PDZ domain, can be cast as protein-peptide prediction problem where the "peptide" is the short linear sequence (4 – 6 amino acids long) lying at the protein's C-terminal. We are currently examining the feasibility of using the *PepDist* framework for this application.

Our novel formulation of the protein-peptide binding prediction problem and the results obtained suggest two interesting conclusions: The first is that learning a *single* distance function over an entire family of proteins achieves higher prediction accuracy than learning a set of binary classifiers for each of the proteins separately. This effect is even more pronounced on proteins for which only small amounts of binders and non-binders are currently available. The second interesting conclusion, is the importance of obtaining information on experimentally determined non-binders. These non-binders (as opposed to randomly generated non-binders) are usually somewhat similar to known binders, since they were in many cases suspected to be binders. Our results on the MHCBN dataset show that learning with real non-binders generalizes better than learning with randomly generated peptides that are assumed to be non-binders. This suggests that information about non-binding peptides should also be published and made publicly available.

Methods

The *DistBoost* Algorithm

Our peptide-peptide distance functions are learned using the *DistBoost* algorithm. *DistBoost* is a semi-supervised learning technique that learns a distance function using unlabeled data points and equivalence constraints.

Notations

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 $(x_{i1}, x_{i2}, \gamma_i)$ where $\gamma_i = 1$ if points (x_{i1}, x_{i2}) belong to the same class (positive constraint) and $\gamma_i = -1$ if these points belong to different classes (negative constraint). $(x_{i1}, x_{i2}, *)$ denotes an unlabeled pair. The *DistBoost* algorithm learns a bounded distance function, $D: \mathcal{X} \times \mathcal{X} \rightarrow [0, 1]$, that maps each pair of points to a real number in $[0, 1]$.

Algorithm description

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 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* [27,28]. 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 [29]. 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 presented in Fig. 9 and illustrated in Fig. 10.

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

Input:

Data points: (x_1, \dots, 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 $W_{i_1 i_2}^1 = 1/(n^2)$ $i_1, i_2 = 1, \dots, n$ (weights over pairs of points)
 $w_k = 1/n$ $k = 1, \dots, n$ (weights over data points)
- For $t = 1, \dots, T$
 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 $h_t(x_i, x_j) = \frac{1}{2} (1 - \tilde{h}_t(x_i, x_j)) \in [0, 1]$
 3. Compute $r_t = \sum_{(x_{i_1}, x_{i_2}, y_i = \pm 1)} W_{i_1 i_2}^t y_i \tilde{h}_t(x_{i_1}, x_{i_2})$, 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:

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

6. Normalize: $W_{i_1 i_2}^{t+1} = \frac{W_{i_1 i_2}^{t+1}}{\sum_{i_1, i_2=1}^n W_{i_1 i_2}^{t+1}}$

7. Translate the weights from $\mathcal{X} \times \mathcal{X}$ to \mathcal{X} : $w_k^{t+1} = \sum_j W_{kj}^{t+1}$

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

Figure 9
The *DistBoost* Algorithm.

'Adaboost with confidence intervals' algorithm [27] to incorporate unlabeled data into the boosting process. More specifically, given a partially labeled dataset $\{(x_{i_1}, x_{i_2}, y_i)\}_{i=1}^N$ where $y_i \in \{1, -1, *\}$, the algorithm

searches for a hypothesis $\mathcal{D}(x_{i_1}, x_{i_2}) = \sum_{t=1}^T \alpha_t h_t(x_{i_1}, x_{i_2})$ which minimizes the following loss function:

$$\sum_{\{i|y_i=1,-1\}} \exp(-y_i \mathcal{D}(x_{i_1}, x_{i_2})) \quad (2)$$

Note that this semi-supervised boosting scheme computes the weighted loss only on **labeled** pairs of points but updates the weights over **all** pairs of points (see Figure 9 steps (3–6)). The unlabeled points effectively constrain the parameter space of the weak learner, giving priority to hypotheses which both comply with the pairwise constraints and with the data's density. Since the weak learner's task becomes harder in later boosting rounds, the boosting algorithm gradually reduces the weights of the unlabeled pairs (see Figure 9 step (5)). While the weak learner accepts a distribution over the points in the original space \mathcal{X} , the boosting process described above generates a distribution over pairs of points that belong to the product space $\mathcal{X} \times \mathcal{X}$. The distribution over the product space is converted to a distribution over the sample points by simple marginalization (see Figure 9 step (7) of the algorithm). The translation from the original input space into product space is introduced in step (2) of the algorithm and is further discussed below.

DistBoost's weak learner

DistBoost's weak learner is based on the constrained Expectation Maximization (cEM) algorithm [29]. 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 $p(x|\Theta) = \sum_{l=1}^M \pi_l p(x|\theta_l)$, where π_l denotes the weight of each Gaussian, θ_l its parameters, and M denotes the number of Gaussian sources in the GMM. Estimating the parameters (Θ) of a GMM is usually done using the well known EM algorithm [30]. 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 cEM 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 $\left\{ \left(p_j^1, p_j^2 \right) \right\}_{j=1}^{N_p}$

and negative constraints by $\left\{ \left(n_k^1, n_k^2 \right) \right\}_{k=1}^{N_n}$. Let

$H = \{h_i\}_{i=1}^n$ denote the hidden assignment of each data point x_i to one of the Gaussian sources ($h_i \in \{1, \dots, M\}$). The constrained EM algorithm assumes the following joint distribution of the observables X and the hidden H :

$$p(X, H | \Theta, \Omega) = \frac{1}{Z} \prod_{i=1}^n \pi_{h_i} p(x_i | \theta_{h_i}) \prod_{j=1}^{N_p} \delta_{h_{p_j^1}, h_{p_j^2}} \prod_{k=1}^{N_n} (1 - \delta_{h_{n_k^1}, h_{n_k^2}}) \quad (3)$$

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 (3) with respect to H . For a more detailed description of this weak learner see [29].

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 Fig. 9 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 $\mathcal{X} \times \mathcal{X}$. Let us Denote by $MAP(x_i)$ the Maximum A-Posterior assignment of point x_i and by $p^{MAP}(x_i)$ the MAP probability of this point: $p^{MAP}(x_i) = \max_m p(h_i = m | x_i, \Theta)$. We partition the data into M groups using the MAP assignment of the points and define

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

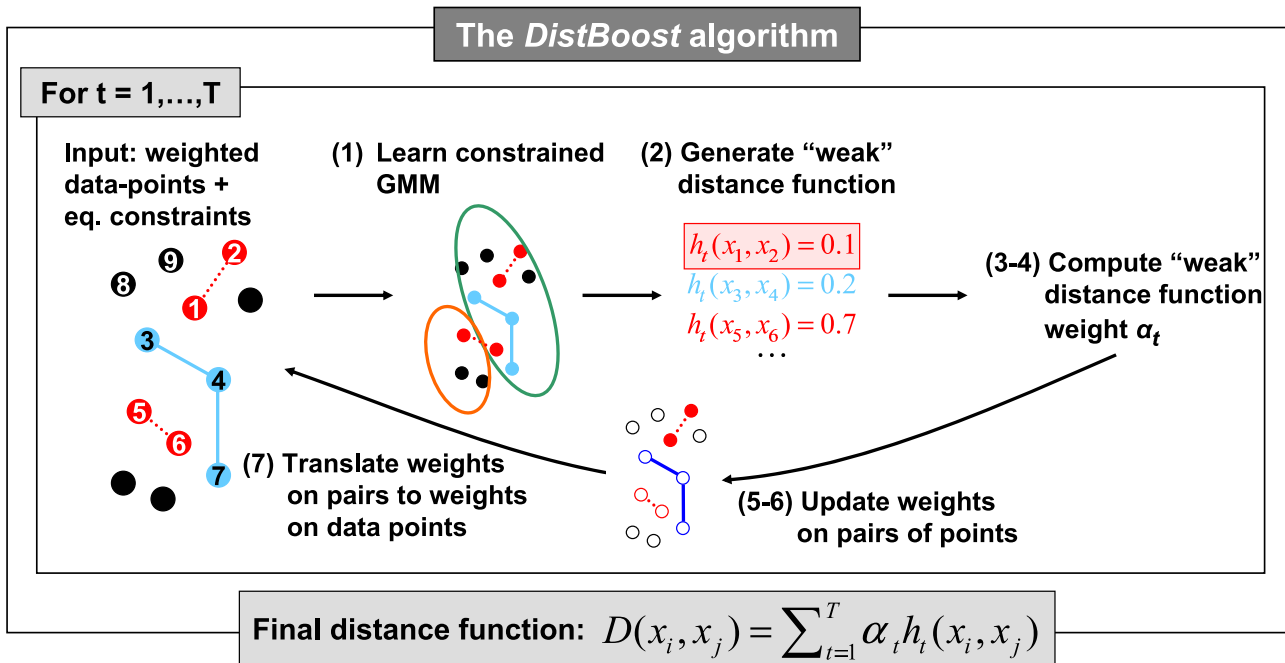
The weak distance function is given by

$$h_t(x_i, x_j) = \frac{1}{2} \left(1 - \tilde{h}_t(x_i, x_j) \right) \in [0, 1] \quad (4)$$

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 $[0.5, 1]$.

Datasets

The first two datasets we compiled (MHCclass1 and MHCclass2) were the same as those described in [13]. Fol-

**Figure 10**

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.

Following the works of [8,9,13] we considered peptides with a fixed sequence length of 9 amino acids. Sequences of peptides, that bind to MHC class I or class II molecules, were collected from the MHCPEP dataset [31]. 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 (see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 3).

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 [13], we first used the MEME program [32] 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 (see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 4).

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 training 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 [33] (see additional file 1: *Pepdist_SupplementaryMaterials.ps*, Table 5).

Data representation

DistBoost requires that the data be represented in some continuous vector feature space. Following [23] each amino acid was encoded using a 5-dimensional property vector. Therefore, 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 [23] 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 X-axis represents the percentage of "false alarms" which is $FP/(FP + TN)$ (where FP denotes False Positives, and TN denotes True Negatives). The Y-axis represents the percentage of "hits" which is $TP/(TP + FN)$ (where TP denotes True Positives and FN denotes False Negatives). 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.

Authors' contributions

Both authors contributed equally.

Additional material

Additional File 1

This file includes the following tables: 1. Table 1: Average AUC scores and standard deviations obtained by RANKPEP and DistBoost on MHC class I molecules. 2. Table 2: Average AUC scores and standard deviations obtained by RANKPEP and DistBoost on MHC class II molecules. 3. Tables 3, 4, 5, describe the 3 datasets used in the paper.

Click here for file

[<http://www.biomedcentral.com/content/supplementary/1471-2105-7-S1-S3-S1.pdf>]

Acknowledgements

We thank Daphna Weinshall for many fruitful discussions and comments. We also thank Itamar Glatzer for his help in the implementation of the *PepDist* webserver. C.Y is supported by Yeshaya Horowitz Association through the Center for Complexity Science.

References

- Hertz T, Bar-Hillel A, Weinshall D: **Learning Distance Functions for Image Retrieval**. *CVPR, Washington DC* 2004.
- Hertz T, Bar-Hillel A, Weinshall D: **Boosting Margin Based Distance Functions for Clustering**. *ICML* 2004.
- Rammensee HG, Friede T, Stevanovic S: **MHC ligands and peptide motifs: first listing**. *Immunogenetics* 1995, **41(4)**:178-228.
- Yewdell JW, Bennink JR: **Immunodominance in Major Histocompatibility Complex Class I-Restricted T-Lymphocyte Responses**. *Annual Review of Immunology* 1999, **17**:51-88.
- Janeway CA, Travers P, Walport M, Shlomchik M: *Immunobiology* 5th edition. New York and London: Garland Publishing; 2001.
- Sette A: **Tools of the Trade in Vaccine Design**. *Science* 2000, **290(5499)**:2074b-2075.
- Brusic V, Rudy G, Harrison LC: **Prediction of MHC binding peptides using artificial neural networks**. *Complexity International* 1995, **2**.
- Gulukota K, Sidney J, Sette A, DeLisi C: **Two complementary methods for predicting peptides binding major histocompatibility complex molecules**. *Journal of Molecular Biology* 1997, **267**:1258-1267.
- 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* 2000, **9(9)**:1838-1846.
- Mamitsuka H: **Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models**. *Proteins* 1998, **33(4)**:460-474.
- Donnes P, Elofsson A: **Prediction of MHC class I binding peptides, using SVMHC**. *BMC Bioinformatics* 2002, **3**:25. [http://www-bs-informatik.uni-tuebingen.de/SVMHC](http://www.bs-informatik.uni-tuebingen.de/SVMHC)
- Buus S, Laue-moller 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* 2003, **62(5)**:378-384. <http://www.cbs.dtu.dk/services/NetMHC/>
- Reche PA, Glutting JP, Zhang H, Reinher EL: **Enhancement to the RANKPEP resource for the prediction of peptide binding to MHC molecules using profiles**. *Immunogenetics* 2004, **56(6)**:405-419. <http://mif.dfci.harvard.edu/Tools/rankpep.html>
- Yu K, Petrovsky N, Schonbach C, Koh JL, Brusic V: **Methods for Prediction of Peptide Binding to MHC Molecules: A Comparative Study**. *Molecular Medicine* 2002, **8**:137-148.
- 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* 2000, **55(6)**:519-531.
- Wiedemann U, Boisguerin P, Leben R, Leitner D, Krause G, Moelling K, Volkmer-Engert R, Oschkinat H: **Quantification of PDZ Domain Specificity, Prediction of Ligand Affinity and Rational Design of Super-binding Peptides**. *Journal of Molecular Biology* 2004, **343**:703-718.
- Sette A, Sidney J: **Nine major HLA class I supertypes account for the vast preponderance of HLA-A and -B polymorphism**. *Immunogenetics* 1999, **50**:201-212.
- Bar-Hillel A, Hertz T, Shental N, Weinshall D: **Learning Distance Functions using Equivalence Relations**. *The 20th International Conference on Machine Learning* 2003.
- Xing E, Ng A, Jordan M, Russell S: **Distance Metric learnign with application to clustering with side-information**. In *Advances in Neural Information Processing Systems Volume 15*. The MIT Press; 2002.
- Wagstaff K, Cardie C, Rogers S, Schroedl S: **Constrained K-means Clustering with Background Knowledge**. In *Proc 18th International Conf on Machine Learning Morgan Kaufmann, San Francisco, CA*; 2001:577-584.
- Bilenko M, Basu S, Mooney R: **Integrating Constraints and Metric Learning in Semi-Supervised Clustering**. In *ICML Banff Canada, AAAI press*; 2004. [citeseer.ist.psu.edu/705723.html]

22. Klein D, Kamvar S, Manning C: **From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering.** 2002. [citeseer.nj.nec.com/klein02from.html]
23. Venkatarajan MS, Braun W: **New quantitative descriptors of amino acids based on multidimensional scaling of a large number of physical-chemical properties.** *Journal of Molecular Modeling* 2001, **7(12)**:445-453.
24. Flower DR: **Towards in silico prediction of immunogenic epitopes.** *TRENDS in immunology* 2003, **24**:
25. Madden DR: **The Three-Dimensional Structure of Peptide-MHC Complexes.** *Annual Review of Immunology* 1995, **13**:587-622.
26. Hung AY, Sheng M: **PDZ Domains: Structural Modules for Protein Complex Assembly.** *J Biol Chem* 2002, **277(8)**:5699-5702.
27. Schapire RE, Singer Y: **Improved Boosting Using Confidence-rated Predictions.** *Machine Learning* 1999, **37(3)**:297-336.
28. Schapire RE, Freund Y, Bartlett P, Lee WS: **Boosting the margin: a new explanation for the effectiveness of voting methods.** In *Proc 14th International Conference on Machine Learning Morgan Kaufmann*; 1997:322-330.
29. Shental N, Bar-Hilel A, Hertz T, Weinshall D: **Computing Gaussian Mixture Models with EM using Equivalence Constraints.** *NIPS* 2003.
30. Dempster AP, Laird NM, Rubin DB: **Maximum Likelihood from incomplete data via the EM algorithm.** *JRSSB* 1977, **39**:1-38.
31. Brusica V, Rudy G, Harrison LC: **MHCPEP, a database of MHC-binding peptides: update 1997.** *Nucl Acids Res* 1998, **26**:368-371.
32. Bailey T, Elkan C: **Fitting a mixture model by expectation maximization to discover motifs in biopolymers.** *ISMB* 1994, **2**:28-36.
33. Bhasin M, Singh H, Raghava GPS: **MHCBN: a comprehensive database of MHC binding and non-binding peptides.** *Bioinformatics* 2003, **19(5)**:665-666.

Publish with **BioMed Central** and every scientist can read your work free of charge

"BioMed Central will be the most significant development for disseminating the results of biomedical research in our lifetime."

Sir Paul Nurse, Cancer Research UK

Your research papers will be:

- available free of charge to the entire biomedical community
- peer reviewed and published immediately upon acceptance
- cited in PubMed and archived on PubMed Central
- yours — you keep the copyright

Submit your manuscript here:
http://www.biomedcentral.com/info/publishing_adv.asp

