BMC Bioinformatics

Research article



# Improving peptide-MHC class I binding prediction for unbalanced datasets

Ana Paula Sales[1,2], Georgia D Tomaras[4] and Thomas B Kepler*[1,3]

Address: [1]Center for Computational Immunology, Duke University, Durham, NC, 27705, USA , [2]Computational Biology and Bioinformatics PhD Program, Institute for Genome Sciences & Policy, Duke University, Durham, NC, 27705, USA , [3]Department of Biostatistics and Bioinformatics and Department of Immunology, Duke University, Durham, NC, 27705,USA and [4]Duke Human Vaccine Institute and Departments of Molecular Genetics and Microbiology, Immunology, and Surgery, Duke University, Durham, NC, 27705, USA

Email: Ana Paula Sales - ad44@duke.edu; Georgia D Tomaras - gdt@duke.edu; Thomas B Kepler* - kepler@duke.edu

* Corresponding author

Published: 19 September 2008

*BMC Bioinformatics* 2008, **9**:385 doi:10.1186/1471-2105-9-385

Received: 15 April 2008
Accepted: 19 September 2008

This article is available from: http://www.biomedcentral.com/1471-2105/9/385



## Abstract

**Background:** Establishment of peptide binding to Major Histocompatibility Complex class I (MHCI) is a crucial step in the development of subunit vaccines and prediction of such binding could greatly reduce costs and accelerate the experimental process of identifying immunogenic peptides. Many methods have been applied to the prediction of peptide-MHCI binding, with some achieving outstanding performance. Because of the experimental methods used to measure binding or affinity between peptides and MHCI molecules, however, available datasets are enriched for nonbinders, and thus highly unbalanced. Although there is no consensus on the ideal class distribution for training sets, extremely unbalanced datasets can be detrimental to the performance of prediction algorithms.

**Results:** We have developed a decision-theoretic framework to construct cost-sensitive trees to predict peptide-MHCI binding and have used them to 1) Assess the impact of the training data's class distribution on classifier accuracy, and 2) Compare resampling and cost-sensitive methods as approaches to compensate for training data imbalance. Our results confirm that highly unbalanced training sets can reduce the accuracy of classifier predictions and show that, in the peptide-MHCI binding context, resampling methods do not improve the classifier performance. In contrast, cost-sensitive methods significantly improve accuracy of decision trees. Finally, we propose the use of a training scheme that, when the training set is enriched for nonbinders, consistently improves the overall classifier accuracy compared to cost-insensitive classifiers and, in particular, increases the sensitivity of the classifiers. This method minimizes the expected classification cost for large datasets.

**Conclusion:** Our method consistently improves the performance of decision trees in predicting peptide-MHC class I binding by using cost-balancing techniques to compensate for the imbalance in the training dataset.

## Background

Determination of binding between peptide and Major Histocompatibility Complex class I (MHCI) is a crucial step in the development of subunit vaccines. The peptide-MHCI complexes are required for T cell activation and thus for the initiation of the adaptative immune response. Although MHCI binding does not alone determine the immunogenicity of peptides, it plays an important part, being a major bottleneck that separates immunogenic peptides from non-immunogenic ones. Hence, the ability to predict the binding between peptides and MHCI molecules would greatly reduce costs and accelerate the experimental process of identifying immunogenic peptides, which can then be used in the development of vaccines and therapies against neoplastic, infectious, and autoimmune diseases.

Our primary goal is to guide experimental research in identifying potential vaccine epitopes. In a given microbial genome, there are tens of thousands of peptides and the experimental assessment of the affinity between each peptide and an MHCI molecule represents a significant cost in terms of time and resources. The investigator has to consider the benefits of identifying binders versus the cost associated with experimentally testing nonbinders in order to decide which and how many peptides will be tested in the laboratory. This type of concern can be best addressed by the use of decision-theoretic approaches. Here we formalize such an approach to training decision trees to differentiate binders from nonbinders and show how costs that reflect this experimental tradeoff can be incorporated into the training of classifiers to increase their utility.

Myriad approaches have been applied to the prediction of peptide-MHCI binding. These methods can be divided into two broad categories: 1) MHCI structure-based methods, which use crystallized structures of MHCI molecules to develop computational models of the interaction between MHCI and peptides [1]; and 2) peptide sequence-based methods, which infer the physico-chemical preferences of a particular MHCI allele by analyzing the amino acid sequence of peptides with known affinity to it, where peptides with IC50 lower than a certain threshold, typically 500 nM [2], are classified as binders, and otherwise as nonbinders. Earlier prediction methods used the amino acid frequencies in each position of MHCI-eluted peptides to derive binding motifs and position specific scoring matrices (PSSMs). Methods of this type include SYFPEITHI [3] and BIMAS [4], which have been publicly available and used extensively by the experimental community. As the number of peptides in the MHCI databases increased, so did the number of different machine learning methods that were applied to this problem, which include artificial neural networks [5], support vector machines [6], hidden Markov models [7], Gibbs sampling [8], and classification trees [9-11].

While some of these methods achieve outstanding performance in predicting binding between peptides and certain MHCI alleles, all of them suffer from the fact that the available data for training is heavily biased towards one class of peptides (either binders or nonbinders). There is a vast literature on the impact of class distribution of training sets on the performance of the prediction algorithms (for further readings see Chawla *et al.*, 2004 [12]), and although there is not a straightforward answer to the question of what the ideal class distribution of training datasets is, it has been suggested that a balanced distribution or the estimated distribution of the target population should be used. Moreover, it is a well known phenomenon that highly unbalanced datasets are detrimental to classifier performance. The imbalance in the peptide-MHCI binding data depends on the experimental methods used to produce them: either elution assays, in which case the dataset consists purely of binders; or binding assays in which peptides are tested for binding or affinity to a particular MHCI allele, leading to datasets consisting mostly of nonbinders. The reason for this imbalance towards nonbinders is that binders are extremely rare in nature: It has been estimated that the proportion of peptides in a protein that will bind to a given MHC allele varies between 0.001 and 0.05 [13]. Datasets generated in different laboratories using different assays and conditions are often inconsistent with each other and thus combination of datasets can be very difficult.

Here we investigate how best to use unbalanced datasets to train algorithms for the prediction of peptide-MHCI binding. Although there is no universally agreed upon method for dealing with unbalanced data, several techniques have been proposed to deal with this issue and have been demonstrated to improve prediction accuracy depending on the context in which they are used [12]. Elkan [14] showed how to make a standard learning algorithm yield cost-sensitive results when trained with an unbalanced dataset. Another successful strategy is referred to as cost-sensitive methods, in which weights are used to compensate for the imbalance in the ratio of the two classes. Other methods pre-process the data to achieve a balanced class distribution. In particular two resampling methods stand out: 1) Undersampling, where random cases of the majority class are deleted until both classes have the same number of cases; and 2) Oversampling, where random cases of the minority class are duplicated until both classes have the same number of cases. Our primary goal is to determine whether or not the accuracy of peptide-MHCI binding prediction can be improved by the use of methods that compensate for the training data imbalance, such as resampling and cost-sensitive meth-

ods. The results presented herein suggest that resampling procedures, such as undersampling and oversampling, do not consistently improve the utility of classifiers used in the context of peptide-MHCI binding. The cost-sensitive method, however, significantly improves prediction accuracy when the training data is biased towards nonbinders. These results are derived from analysis using decision trees. The underlying mathematical treatment is, however, quite general, and can be applied to any classifier capable of cost-sensitive learning, including most of the classifiers used in peptide-MHCI binding prediction.

## Methods

### *Approach*

The development of subunit vaccines is a multi-step process; at each stage, the investigators must decide whether a particular peptide warrants further investment or should be omitted from further experimentation. These decisions must be informed, either explicitly or implicitly, by consideration of the costs incurred in continuing the experiments and of the potential reward for a positive discovery. One must also estimate the probability that a given decision will be erroneous, either as a false positive (continuing to invest in a peptide that will prove to be unsuitable) or a false negative (discontinuing tests on a peptide that would have worked). Let the cost of misclassifying a binder be denoted $\kappa_2$ (for type 2 error) and that for misclassifying a non-binder, $\kappa_1$. We refer to $\kappa_2$ as the "real-world" cost, as it can be interpreted as the number of nonbinders an investigator is willing to test in the laboratory in order to find one binder. Finally, suppose that we can parameterize a family of classifiers with the continuous vector $\theta$. Then the cost, $K$, incurred in making a decision on a peptide $\varphi$ using the classifier $T(\theta)$ is

$$K(\varphi|\theta) = \tau_+(\varphi)\kappa_2 c_-(\varphi|\theta) + \tau_-(\varphi)\kappa_1 c_+(\varphi|\theta), \quad (1)$$

where $\tau_+$ and $\tau_-$ are indicators of true class and $c_+(\cdot|\theta)$ and $c_-(\cdot|\theta)$ are indicators of the classification induced by $T(\theta)$. The expected decision cost over all peptides is

$$EK(q) = pk_{2}\,\epsilon_{2}(q) + (1-p)k_{1}\,\epsilon_{1}(q) \quad (2)$$

where $\pi$ is the proportion of binders in the population and $_i$ is the expected rate of type $i$ errors. We would like to find the classifier $T^*$ that minimizes this expected cost. In the training context, we use the "training cost function", $K_T(\varphi|\theta)$, which has the same form as the decision cost described above, but differs from it in the fact that both the false positive $\lambda_2$ and false negative $\lambda_1$ costs are now tunable parameters:

$$K_T(\varphi|\theta) = \tau_+(\varphi)c_-(\varphi)\lambda_2 + \tau_-(\varphi)c_+(\varphi)\lambda_1. \quad (3)$$

Because we only ever have access to finite datasets to train classifiers, the training cost from using such classifiers can be decomposed into two parts: the decision-making cost described in Eq. 1 and the residual, due to the deviation of the training set $D$ from the whole population:

$$\begin{aligned} K_T(D|\theta) &= n\{p\lambda_2\dagger_2(\theta) + (1-p)\lambda_1\dagger_1(\theta)\} \\ &+ \sum_{\phi\in D}\{\tau_+(\phi)\lambda_2\delta c_-(\phi|\theta) + \tau_-(\phi)\delta c_+(\phi|\theta)\} \end{aligned} \quad (4)$$

where $n$ is the size of the training dataset, $p$ is its proportion of positives and the classification error is defined on positive (negative) peptides as

$$\delta c_{-(+)}(\varphi|\theta) \equiv c_{-(+)}(\varphi|\theta) - {}_{2(1)}(\theta) \quad (5)$$

We may further abbreviate this expression to

$$K_T(D|\theta) = n\,\{p\lambda_{22}(\theta) + (1-p)\lambda_{11}(\theta)\} + R(D;\theta,\lambda). \quad (6)$$

The expected decision cost described in Eq. 2 is minimized at $\hat{\theta}$ where

$$0 = \frac{\partial}{\partial\theta}EK(\theta) = \pi\kappa_2\frac{\partial\dagger_2}{\partial\theta}(\theta) + (1-\pi)\kappa_1\frac{\partial\dagger_1}{\partial\theta}(\theta). \quad (7)$$

Similarly, the training cost function is minimized at $\theta^*$ where

$$0 = \frac{\partial}{\partial\theta}L(D|\theta^*) = p\lambda_2\frac{\partial\dagger_2}{\partial\theta}(\theta^*) + (1-p)\lambda_1\frac{\partial\dagger_1}{\partial\theta}(\theta^*) + \frac{\partial R}{\partial\theta}(D;\theta^*,\lambda). \quad (8)$$

Denote the value of $\theta$ that minimizes the expected decision cost by $\hat{\theta}$, and that that minimizes the training cost function by $\theta^* = \hat{\theta} + 1/n\,\delta\theta$. Now differentiation and Taylor expansion yield the sufficient condition for the minimum of the training cost function to approach $\hat{\theta}$ as $R(\theta)/n \to 0$:

$$\lambda_2^B = \frac{\pi}{1-\pi}\frac{1-p}{p}\frac{\kappa_2}{\kappa_1}\lambda_1. \quad (9)$$

This expression defines what we refer to as the "balancing cost", $\lambda_2^B$. The basic intuition behind the balancing cost is that its use results in both classes having equal importance in the training of the classifier. It is helpful to note that: 1) As the real-world false negative cost $\kappa_2$ increases, so does the balancing cost; and 2) As the proportion of

positives $p$ in the training set increases in relation to the population positive frequency $\pi$, the balancing cost decreases (see figure 1). Finally, we have

$$\delta\theta = -\frac{\partial R}{\partial \theta}\left[ p\lambda_2 \frac{\partial^2 \dagger_2}{\partial \theta^2} + (1-p)\lambda_1 \frac{\partial^2 \dagger_1}{\partial \theta^2} \right]^{-1}, \quad (10)$$

with the right-hand side evaluated at $\hat{\theta}$, which provides a first-order correction for finite datasets. Figure 1 displays the relation between population and training sample positive proportions and costs as described in Eq. 9 and can serve as a guideline of what weights to assign to peptides of different classes given the class distribution in the training set and the relative importance of positives versus negatives in the real-world application.

### *Datasets*

The peptide binding data used to train and test the decision trees were obtained from a publicly available database published by Peters *et al.* [2], where the peptide affinity to a particular MHCI molecule is measured by one of two assays, and classified as binder when its IC50 is less or equal to 500 nM, and nonbinder otherwise. Decision trees were constructed for each one of 35 alleles in the dataset. The cost-sensitive and resampling experiments (described below) were performed for five alleles: A0203, A1101, A3101, B0702 and B1501. The numbers of peptides in the datasets for these five alleles are shown in table 1.

### *Cost Adjustments*

Seven training sets for each allele studied were generated, such that all training sets for a given allele had the same number of observations but varying proportions $p$ of positives, namely 5%, 10%, 25%, 50%, 75%, 90% and 95%. These training sets were created as follows. First, 25% of the binders and 25% of the nonbinders were randomly

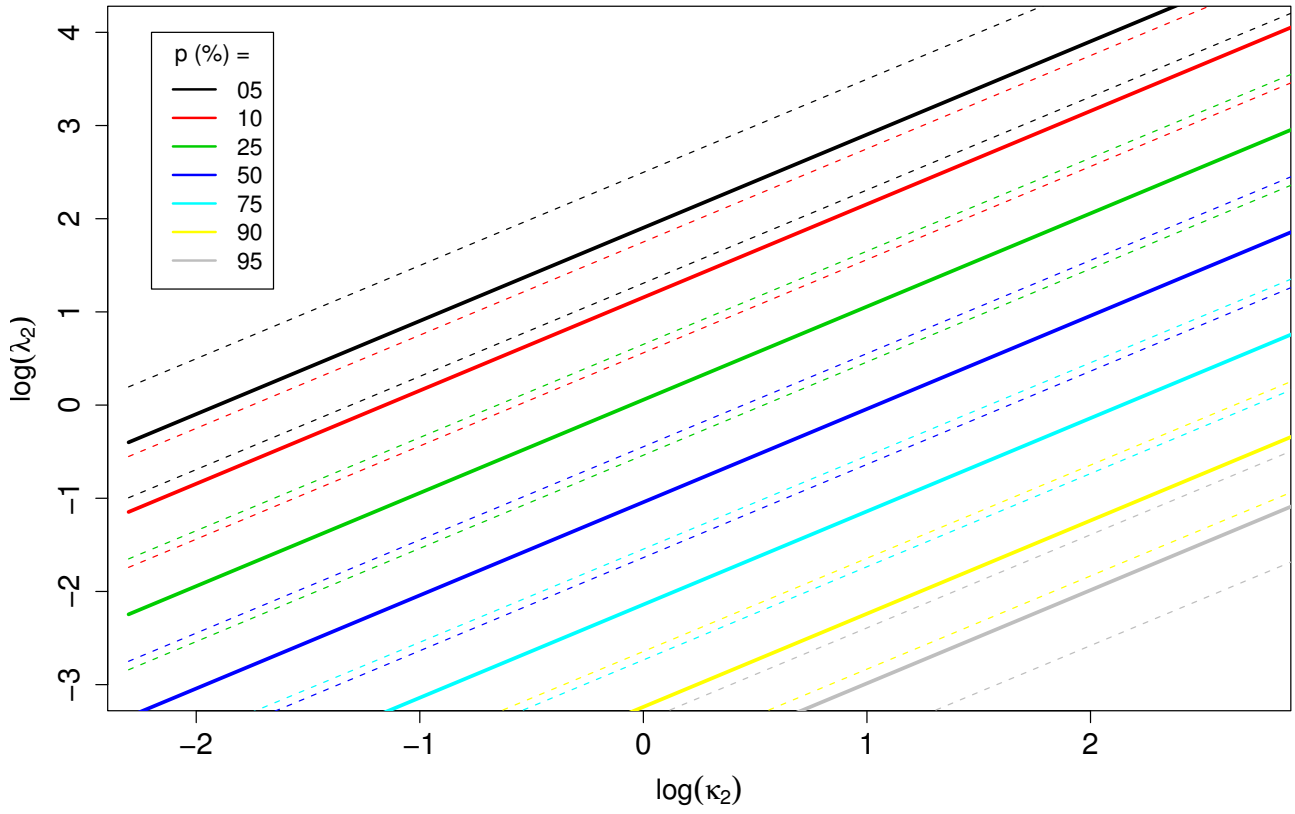


**Figure 1**
**Theoretical relation between $\lambda_2$ and $EK(\theta)$.** Theoretical relationship between the training false negative cost ($\lambda_2$) that minimizes the expected cost of a classifier ($EK(\theta)$) for a given type 2 error cost ($\kappa_2$). The dotted lines represent one standard deviation from the mean. Here $\kappa_1 = 1$, $\lambda_1 = 1$ and $\pi = 0.5$.

**Table 1: Number of binders and nonbinders in Peters *et al.* [2] datasets for 5 alleles.**

| allele | binders | nonbinders |
|---|---|---|
| A0203 | 639 | 804 |
| A1101 | 695 | 1290 |
| A3101 | 427 | 1442 |
| B0702 | 210 | 1052 |
| B1501 | 179 | 799 |

selected and set aside as a testing set. The remaining 75% of the binders and of the nonbinders formed the "training superset", from which the peptides for the various training sets were sampled. The total number of peptides in each of the seven training sets was fixed and equal to the number of peptides of the minority class in the training superset. The minority class was the positive for all 5 alleles that we tested. Finally, the training sets were formed by randomly sampling without replacement positive and negative peptides from the training superset such that the described class distribution was reached. The numbers of binders and nonbinders in the resulting training sets are shown in table 2.

The goal of this set of experiments was two-fold: 1) To investigate the relationship between class distribution and classifier performance, and 2) To learn how can misclassification costs be used to improve prediction accuracy for a given class distribution of the training set. We emphasize that our goal is not to improve upon existing computational methods, but rather to show that the performance of a single classifier can be improved with the use of cost-sensitive techniques. Misclassification costs were used as weights with the purpose of artificially changing the class distribution of the training dataset. The false negative cost ($\lambda_2$) can be interpreted as the weight given to the peptides in the positive class, and similarly false positive cost ($\lambda_1$) is the weight given to the negative class. The overall scale of the training cost function (Eq. 6) is arbitrary, so we have fixed $\lambda_1 = 1$ and varied $\lambda_2$ between 1/20 and 20 in order to investigate the relationship between costs and class distribution.

**Table 2: Training sets used in the cost-sensitive experiment**

| | A0203 | | A1101 | | A3101 | | B0702 | | B1501 | |
|---|---|---|---|---|---|---|---|---|---|---|
| %pos | B | NB | B | NB | B | NB | B | NB | B | NB |
| 05 | 24 | 456 | 26 | 495 | 16 | 304 | 7 | 150 | 6 | 128 |
| 10 | 48 | 432 | 52 | 469 | 32 | 288 | 15 | 142 | 13 | 121 |
| 25 | 120 | 360 | 130 | 391 | 80 | 240 | 39 | 118 | 33 | 101 |
| 50 | 240 | 240 | 261 | 261 | 160 | 160 | 79 | 79 | 67 | 67 |
| 75 | 360 | 120 | 391 | 130 | 240 | 80 | 118 | 39 | 101 | 33 |
| 90 | 432 | 47 | 469 | 52 | 288 | 32 | 142 | 15 | 121 | 13 |
| 95 | 456 | 24 | 495 | 26 | 304 | 16 | 150 | 7 | 128 | 6 |

Number of binders (B) and nonbinders (NB) in training sets.

Previous works (e.g., [15]) have suggested that among the best class distributions for learning is the balanced distribution, one in which all classes are equally represented. We assume that given an unbalanced training set, a balancing misclassification cost can be used to achieve an artificially nearly-balanced class distribution. The balancing cost, $\lambda_2^B$, defined in Eq. 9 can be interpreted to be the $\lambda_2$ that weights the positive peptides to be the same number as the negatives and therefore compensates for the imbalance ratio of the two classes. Consider the simplest scenario, where $\lambda_1 = 1$, $\kappa_1 = 1$, $\kappa_2 = 1$ and $\pi = 0.5$, then the balancing cost reduces to

$$\lambda_2^B = \frac{(1-p)}{p}$$

We are particularly interested in how classifiers trained with this simplified balancing cost perform compared to the best classifiers for a given allele, as well as compared with classifiers trained with unit costs ($\lambda_1 = 1$ and $\lambda_2 = 1$).

### *Resampling*

#### *Undersampling*

The undersampling method consists of randomly eliminating peptides of the majority class from the training set until both classes have the same number of examples. The training sets were constructed in a similar manner to the cost-modifying experiment. First, we set aside 25% of binders and nonbinders into the testing set. The remaining binders were put into the training set together with the same number of nonbinders, which were randomly sampled without replacement from the nonbinders training superset. One of the issues concerning undersampling is the loss of information that results from the process, which can be aggravated when particularly important elements are removed from the training set. To get around this problem, we used 10-fold crossvalidation and the results presented here are the average of the 10 experiments.

#### *Oversampling*

The oversampling method consists of randomly replicating peptides of the minority class into the training set until both classes have the same number of examples. The training sets were constructed as follows. First, we set aside 25% of binders and nonbinders into the testing set. All remaining peptides were put into the training set together with $d$ peptides from the minority class which were sampled with replacement, where $d$ is the difference between the number of peptides in the the training set belonging

to the majority and minority classes. Hence, each peptide of the minority class is represented at least once and possibly multiple times in the training set. Similarly to the undersampling procedure, we used 10-fold crossvalidation and the results presented here are the average of the 10 experiments.

### *Decision trees*

The present study applies tree-based models to the peptide-MHCI binding prediction problem. We have chosen to use decision trees for the simplicity in their interpretation and also because they have not been thoroughly explored in the context of peptide-MHCI binding. Moreover, decision and classification trees have become the canonical method for comparison of techniques used to deal with unbalanced datasets in the machine learning community. Finally, there seems to be a natural correspondence between the importance of the different residue positions in a peptide and the hierarchical way in which decision trees are constructed.

#### *Tree generation*

Breiman *et al*. [16] provides an excellent and detailed description of classification and regression trees. Briefly, given a dataset in which each object, $\phi$, is represented by a $(\tau(\phi), \mathbf{x}(\phi))$ pair, where $\mathbf{x}(\phi)$ is a vector containing attributes of the object and $\tau$ is an indicator function of the class of the object, a tree-based classifier recursively partitions the data's attribute space into sub-regions, called *nodes*, in which the response variable is increasingly more homogeneous. These trees are created in two steps: (1) induction of a large tree; and (2) pruning of the large tree into gradually smaller subtrees (here we use the cost-complexity pruning [16]). Finally, one subtree must be chosen from the sequence of subtrees generated by the pruning process. In the present study, we chose the tree that minimizes the training cost function (Eq. 6) when applied to the test set.

The construction of a tree requires (1) a set of *splits*, which are binary questions with mutually exclusive and exhaustive outcomes used to partition the data, where the questions are coined in terms of the attributes of the objects in the dataset; and (2) a *split function* used to quantify the goodness of a split, by measuring the change in the homogeneity of the response variable in the tree due to splitting a node into two subsets based on the given split.

#### *Splits and split function*

In the problem at hand, the training dataset consists of peptides $\varphi$, where $\tau(\varphi)$ is the class of the peptide (either binder or nonbinder) and $\mathbf{x}(\varphi)$ is the linear sequence of amino acids of the peptide, with $x_j$ being the $j^{th}$ amino acid from the amino terminal end of the peptide. The binary questions about the sequence of peptides can be phrased in several distinct ways, and each one of them generates a different class of splits, called *motifs*, that can be used in the construction of trees. We used motifs based on the anchor positions, which are represented by a single amino acid with a fixed position in the peptide, such that every amino acid is represented in every position of the peptide. The amino acids, in turn, can be represented in one of two ways: 1) by the traditional amino acid single-letter code. For example, alanine is represented by "A", arginine by "R" and so forth; and 2) by their physico-chemical properties, namely molecular weight, hydropathicity, volume, isoelectric point, polarity, ability to form hydrogen bonds and chain type (aliphatic, aromatic) as previously shown [17].

The split function used was the training cost described in Eq. 3.

## Results

### *Cost Adjustments*

The first goal of this set of experiments was to investigate the relationship between class distribution and classifier performance. Our results suggests that for a fixed training set size, decision trees perform best when trained with datasets of nearly balanced class distribution. Figure 2 shows the performance of classifiers trained with datasets of the same size but different class distributions and training costs for alleles A1101 and B0702 (see Additional file 1 for the results for the other three alleles). Note that as the proportion of positives in the training set increases, the false negative rate decreases and the false positive rate increases as can be seen by the subtle shift in the curves from left to right.

Our second goal was to determine whether or not prediction accuracy of a given classifier can be improved by the use of cost-sensitive techniques and, if so, to establish the relationship between classifier performance and training costs. Our results demonstrate that misclassification costs can be used to improve prediction accuracy. In fact, for each one of the alleles we tested there was a cost $\lambda_2$ that performed significantly better than the unit cost, as can be seen by the increase in AUC shown in table 3. Although our goal is not to improve upon the performance of existing methods, we also show in table 3 the AUC for 4 other methods as described in [2] for purposes of comparison.

Note in figure 2 that for the training sets with majority of nonbinders, $\lambda_2^B$ consistently reduced the total error rate as compared to the unit cost ($\lambda_2 = 1$). The impact of $\lambda_2^B$ on the performance of classifiers trained with binders-enriched datasets was not consistent, being better than unit cost for some classifiers and worse for others. In addi-

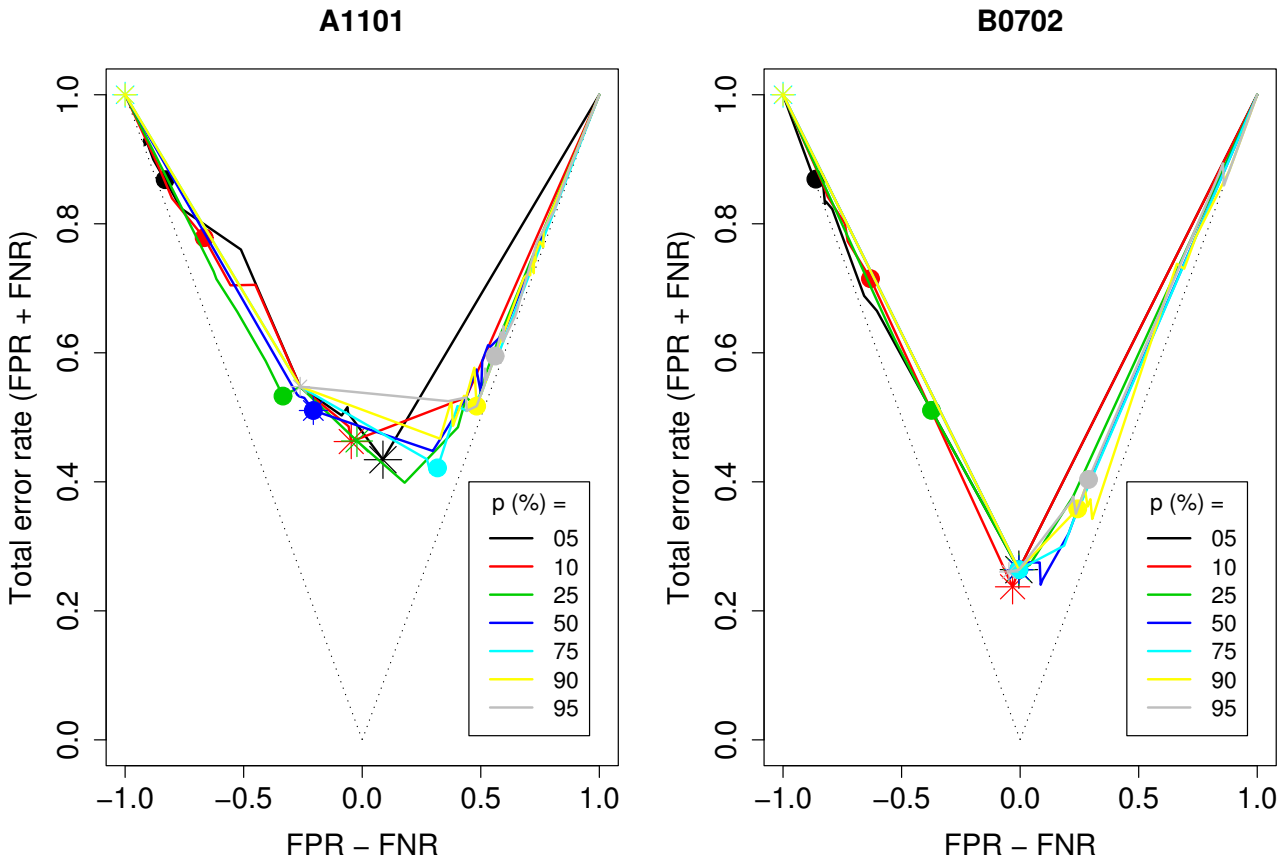


**Figure 2**
**Classifier performance vs. class distribution**. Comparison of the performance of classifiers built with training sets of same size but different proportions of positives for alleles A1101 (left panel) and B0702 (right panel). Each point in a curve represents a classifier constructed with a different false negative training cost. The classifier constructed with the unit cost ($\lambda_2 = 1$) in each curve is marked with a solid circle and that constructed with the balancing cost is marked with a star. The curve for the perfect classifier would lie on the dotted line. The y-axis shows the total error rate of a classifier, which is the same as the classifier cost (*K*) when the type 1 and type 2 misclassification costs are identical ($\kappa_1 = \kappa_2 = 1$). FNR: false negative rate. FPR: false positive rate.

tion to representing an improvement over the unit cost, in a few cases $\lambda_2^B$ coincided with the minimizing cost, that is, the most accurate classifier for a given allele and training set was the one trained with $\lambda_2^B$. However, in most cases, the balancing cost over-compensated for the imbalance in the class distribution, such that it was larger than the minimizing cost (see figure 3)

We then compared the performance of trees trained with the complete dataset using either the unit cost or $\lambda_2^B$ (the red and green ROC curves in figure 4, respectively). The use of $\lambda_2^B$ resulted in AUC at least as large as those for unit cost, such that $\lambda_2^B$ improved the ROC curves as compared to the unit cost in the majority of the cases. One interesting feature of the use of $\lambda_2^B$ is that it consistently shifts the ROC curve toward increasing sensitivity at the price of decreasing specificity, which is a desirable tradeoff when binders are rare. Thus, even in the cases when the increase in AUC is not substantial, the use of $\lambda_2^B$ can still represent an improvement over unit cost due to the shift it causes to the ROC curve.

### *Resampling*

The results obtained using the balanced undersampled and oversampled training sets did not represent an improvement over those using the complete unbalanced

**Table 3: Comparison of classifiers performance as measured by AUC**

| allele | Trees, unit $\lambda_2$ | Trees, best $\lambda_2$ | ARB* | SMM* | ANN* | External Tool* |
|---|---|---|---|---|---|---|
| A0101 | .903 | .951 | .964 | .980 | .982 | .955 |
| A0201 | .796 | .842 | .934 | .952 | .957 | .922 |
| A0202 | .770 | .770 | .875 | .899 | .900 | .793 |
| A0203 | .746 | .781 | .884 | .916 | .921 | .788 |
| A0206 | .732 | .747 | .872 | .914 | .927 | .735 |
| A0301 | .763 | .763 | .908 | .940 | .937 | .851 |
| A1101 | .841 | .859 | .918 | .948 | .951 | .869 |
| A2301 | .742 | .782 | NA | NA | NA | NA |
| A2402 | .685 | .748 | .718 | .780 | .825 | .770 |
| A2403 | .673 | .846 | NA | NA | NA | NA |
| A2601 | .606 | .811 | .907 | .931 | .956 | .736 |
| A2902 | .783 | .847 | NA | NA | NA | NA |
| A3001 | .741 | .861 | NA | NA | NA | NA |
| A3002 | .777 | .810 | NA | NA | NA | NA |
| A3101 | .825 | .833 | .909 | .930 | .928 | .829 |
| A3301 | .636 | .827 | .892 | .925 | .915 | .807 |
| A6801 | .756 | .761 | .840 | .885 | .883 | .772 |
| A6802 | .699 | .714 | .865 | .898 | .899 | .643 |
| A6901 | .614 | .813 | NA | NA | NA | NA |
| B0702 | .887 | .911 | .952 | .964 | .965 | .942 |
| B0801 | .547 | .835 | .936 | .943 | .955 | .766 |
| B1501 | .759 | .823 | .900 | .952 | .941 | .816 |
| B1801 | .745 | .833 | .573 | .853 | .838 | .779 |
| B2705 | .753 | .892 | .915 | .940 | .938 | .926 |
| B3501 | .712 | .771 | .851 | .889 | .875 | .792 |
| B4001 | .587 | .897 | NA | NA | NA | NA |
| B4002 | .718 | .778 | .541 | .842 | .754 | .775 |
| B4402 | .588 | .762 | .533 | .740 | .778 | .783 |
| B4403 | .647 | .804 | .461 | .770 | .763 | .698 |
| B4501 | .679 | .824 | NA | NA | NA | NA |
| B5101 | .664 | .792 | .822 | .868 | .866 | .820 |
| B5301 | .795 | .819 | .871 | .882 | .899 | .861 |
| B5401 | .654 | .796 | .847 | .921 | .903 | .799 |
| B5701 | .756 | .936 | .428 | .871 | .826 | .767 |
| B5801 | .815 | .864 | .899 | .964 | .961 | .899 |

The second and third columns correspond to the decision trees described in the present work. Note the improvement in the performance of the trees with the use of training costs. *Values extracted from table 2 of Peters *et al.*, 2006.

training sets (see figure 4 and Additional file 2). For alleles A0203, A1101 and B0702, the ROC curves for the trees trained with the entire dataset and those trained with the re-sampled dataset were indistinguishable from one another, whereas for alleles A3101 and B1501, the use of undersampling severely damaged the accuracy of the trees.

### *Real-world costs versus training costs*

We built decision trees with the training data described in table 1 using different values of false negative cost ($\lambda_2$), and evaluated them on a test set using the "real-world" cost $\kappa_2$. We call $\hat{\lambda}_2$ the training cost that minimizes the total cost of a classifier on the test set for a given $\kappa_2$. Figure 5 shows the relationship between $\hat{\lambda}_2$ and $\kappa_2$. Note that although the results are relatively noisy, in general the same trend shown in theory can be observed from this empirical data (see figure 1). The $\hat{\lambda}_2$ increases with $\kappa_2$ and as the proportion of positives in the training set increases, the line shifts to the right, indicating that for a particular value of $\kappa_2$, the suggested $\hat{\lambda}_2$ decreases.

## Discussion

Prediction of peptide-MHCI binding has great potential to accelerate and reduce the cost of subunit vaccine development. One of the issues concerning the prediction of MHC-peptide binding is that binders are much less abundant than nonbinders, and thus much harder to find experimentally. This circumstance typically leads to highly unbalanced training sets, which can hinder the performance of algorithms trained with them. In fact, such

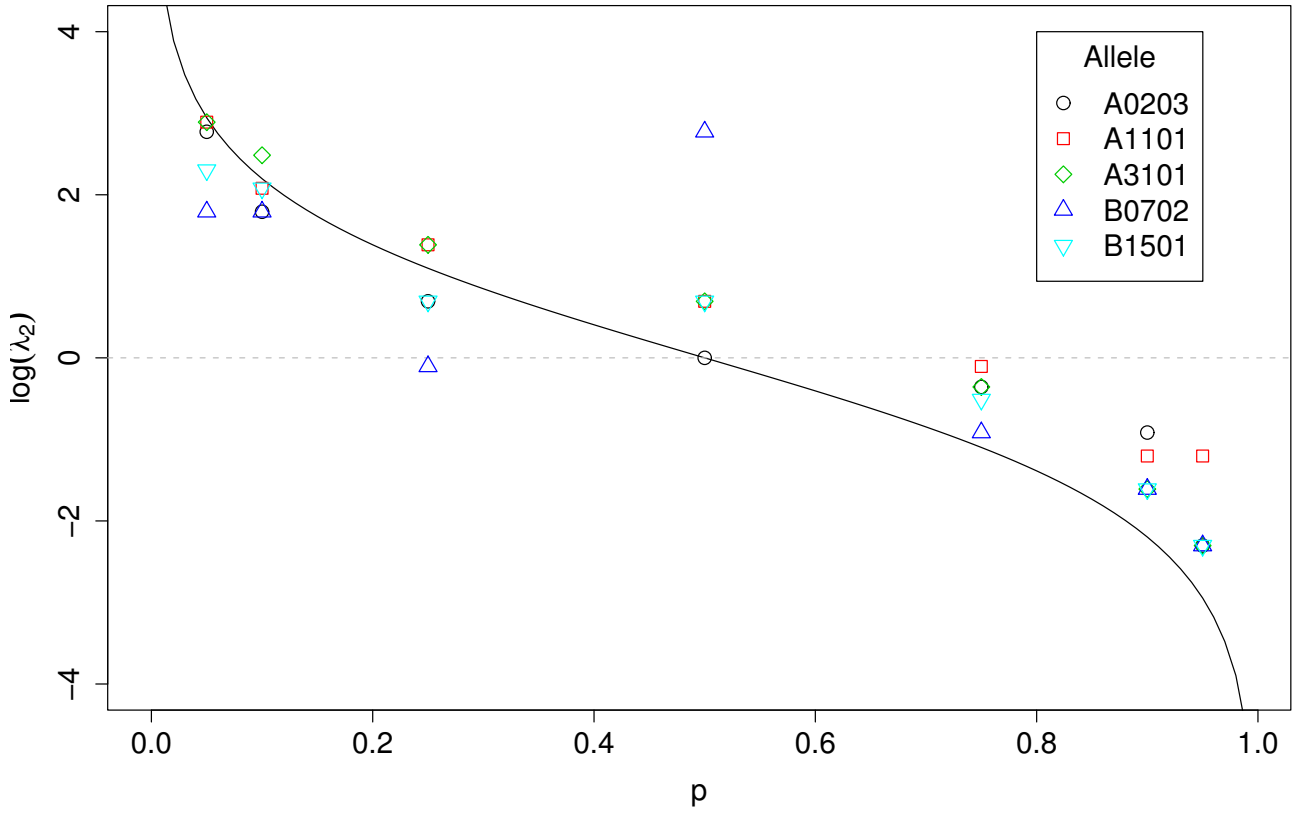


**Figure 3**
**Balancing cost vs. minimizing cost**. Comparison of balancing cost (solid black line) and the minimizing costs (symbols) for each one of the five alleles.

training sets lead to a significant increase of type 2 errors and thus make it more difficult still to find binders.

Our results show that highly unbalanced training sets do indeed reduce the accuracy of predictions made with decision trees and that these predictions improve as the training sets become more balanced. We have examined three approaches that aim at improving classifier accuracy by compensating for the imbalance in the class distribution of the training sets: undersampling, oversampling and a cost-sensitive method. Overall, resampling did not improve the performance of the decision trees. In fact, in several cases classifiers trained with undersampled training sets performed much worse than those trained with the full dataset. This could have been caused by the loss of information relevant to the training process. For this reason, undersampling methods may only be appropriately used with datasets in which the majority class contains a lot of redundancy, in which circumstance undersampling has been shown to outperform other random resampling methods in four distinct datasets [18]. Another potential drawback of undersampling, and in broader terms of random resampling methods, is that they may yield noisy results due to the variability introduced in the process by the randomness of the sampling procedure.

In contrast to undersampling, using misclassification costs as a means to artificially counterbalance data bias led to significant improvements in the performance of the decision trees in the majority of the cases. Although cost-sensitive procedures do not add any extra information to the training set, they seem to be more advantageous than random resampling techniques because they do not cause loss of information as does undersampling and do not have the extra variability introduced by the random sampling process. Several other studies have shown cost-modifying methods to be advantageous. For example, Japkowicz and Stephen [19] performed a systematic comparison of these methods in both artificially-generated and real-world domains, showing that cost-modifying methods yield better results than resampling techniques. Fundamentally, the cost-sensitive method described here can be straightforwardly applied to any classifier that is trained using datasets that include both classes of pep-

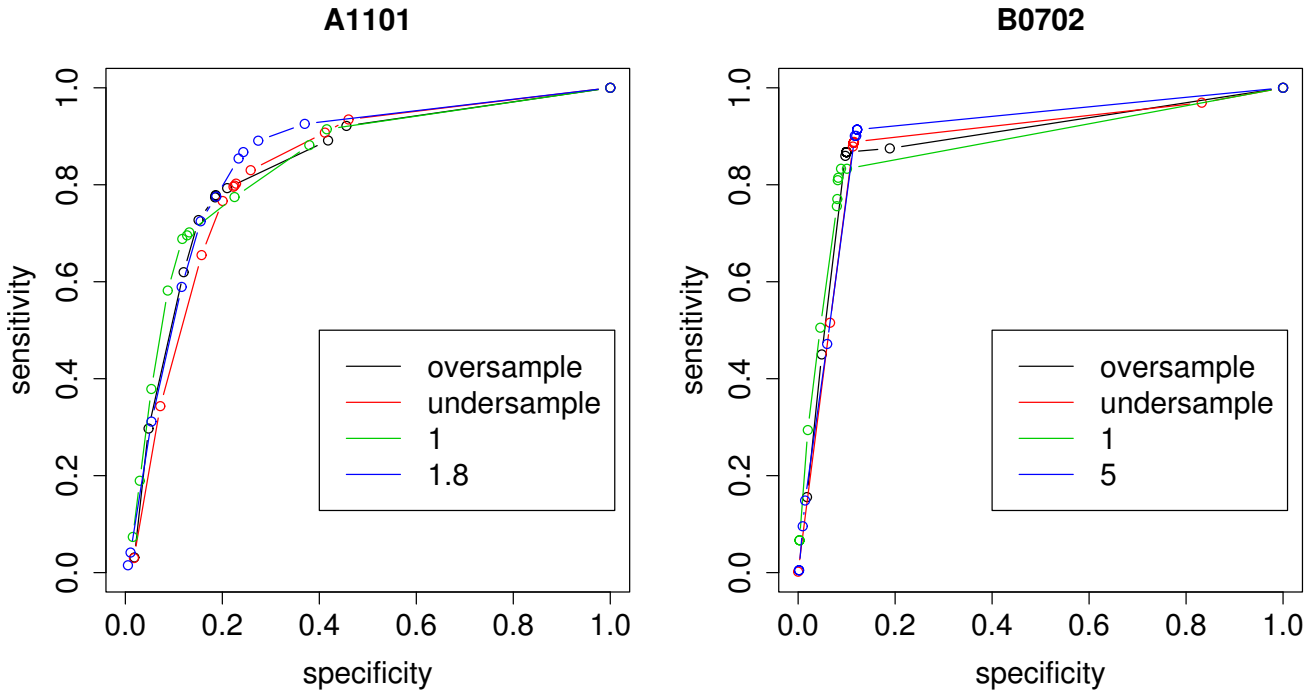


**Figure 4**
**Comparison of unit cost, balancing cost, undersampling and oversampling**. ROC curves for alleles A1101 (left panel) and B0702 (right panel) comparing the results of trees constructed with the oversampled training set (black curve), the undersampled training set (red curve), and the full training set without training costs, that is, $\lambda_1 = \lambda_2 = 1$ (green curve) and with the balancing training cost, that is, $\lambda_1 = 1$ and $\lambda_2 = (1 - p)/p$ (blue curve). The ROC curves were constructed by varying the threshold used to label a node from 0 to 1 and evaluating its sensitivity and specificity at each threshold.

tides, binders and nonbinders. For instance, the individual weights of a weight matrix can be derived by minimizing the cost function (Eq. 3) over these weights. The indicator function $c_{-(+)}(\varphi)$ can be defined by the score function's being above or below a given threshold, where the scoring function is typically the sum of the scores of each amino acid in each position of a peptide. Similarly, this cost function can be incorporated into a neural network by differentially weighting the output depending on the class of the training example and allowing it to be used in the learning process by the backpropagation procedure [20,21]. Likewise, for support vector machines, the cost function can be implemented through the definition of the "soft margin" [22], allowing the SVM to misclassify more examples of one class than examples of the other class.

In addition to showing that peptide-MHCI binding predictions can be improved by the use of cost-sensitive decision trees, we have investigated the use of the balancing cost, $\lambda_2^B$, as a rule-of-thumb to train classifiers. We have shown that although $\lambda_2^B$ is not always the $\lambda_2$ that minimizes the total cost of the classifier, it consistently outperforms the unit cost ($\lambda_2 = \lambda_1$) when the training set is enriched for nonbinders.

Moreover, we have showed that the use of $\lambda_2^B$ shifts the ROC curves towards areas of higher sensitivity in relation to ROC curves generated with unit cost, which can be highly desirable in situations such as epitope discovery projects.

Thus, although the relationship between training costs and class imbalance is relatively noisy, and further studies should be conducted before a complete guideline of what training costs should be used for a particular peptide-MHCI binding dataset, our results allow us to suggest that a balancing cost should be used for datasets enriched for nonbinders, and the unit cost should be used for binders-enriched training sets.

## Conclusion

The vaccine development process is costly and time-consuming, requiring decisions to be made at each step and it

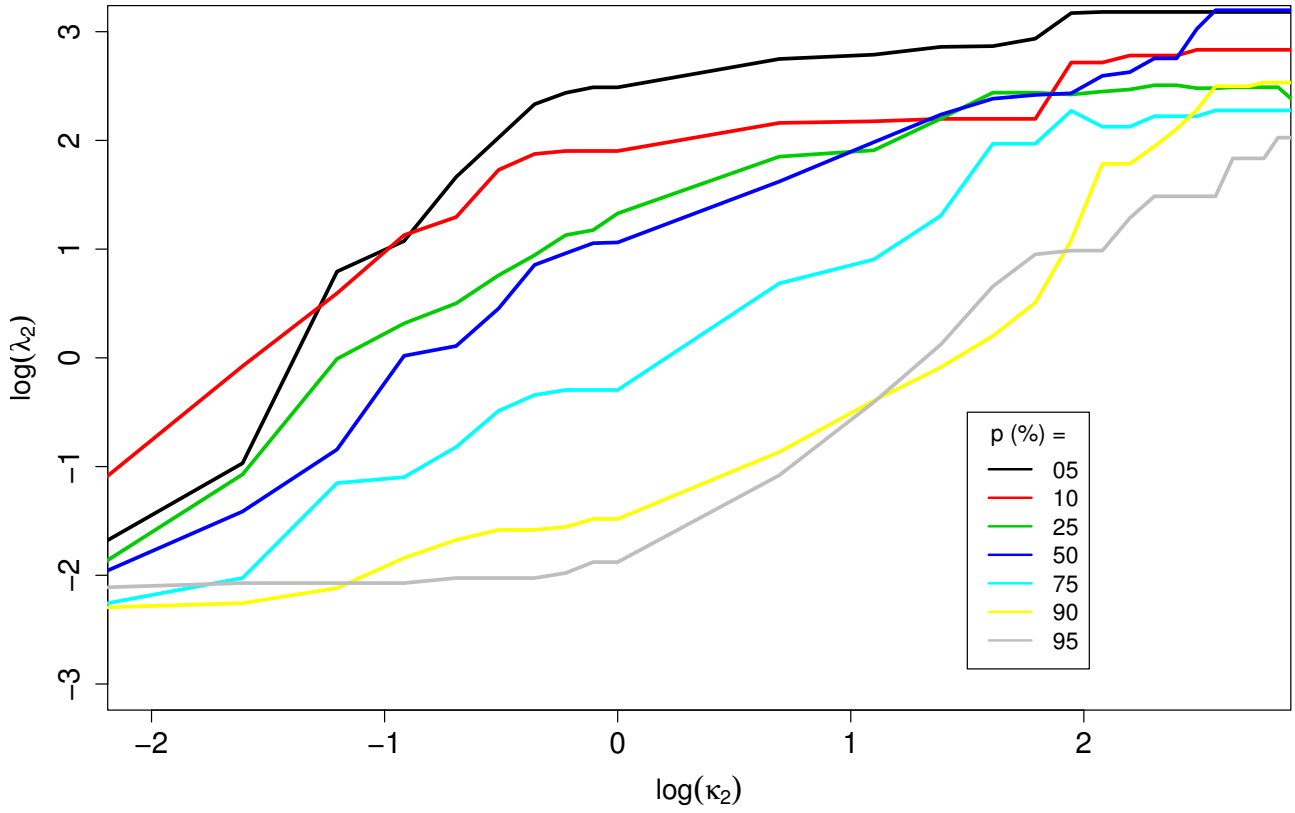


**Figure 5**

**Empirical relation between $\hat{\lambda}_2$ and *EK*($\theta$)**. Optimal false negative training cost ( $\hat{\lambda}_2$ ) as a function of type 2 error cost ($\kappa_2$). Classifiers were trained at multiple values of $\hat{\lambda}_2$ and tested at $\kappa_2$ (compare with figure 1). This was done for each of the five alleles and the $\lambda_2$ shown in the curves are the average of the minimizing $\lambda_2$ for each allele.

lends itself nicely to a decision-theoretic approach, which we have described here. In particular, at the epitope discovery stage, there are real costs associated with the risk of missing a positive and with the experimental verification of nonbinders. Here we have described a decision-theoretic framework for the prediction of peptide-MHCI binding and have provided a guideline on how to incorporate real-world costs together with misclassification costs at the training level in order to maximize prediction accuracy and push it in the desired direction.

## Authors' contributions

APS performed the computational experiments, analyzed the data and wrote the manuscript. TBK supervised the study and wrote the manuscript. GDT supervised the pilot experimental studies and helped revise the manuscript.

## Additional material

### Additional file 1

*Classifier performance vs class distribution for alleles A0203, A3101 and B1501. Comparison of the performance of classifiers built with training sets of same size but different proportions of positives for alleles A0203, A3101 and B1501 (compare to figure 2). Each point in a curve represents a classifier constructed with a different false negative training cost. The classifier constructed with the unit cost ($\lambda_1 = 1$) in each curve is marked with a solid circle and that constructed with the balancing cost is marked with a star. The curve for the perfect classifier would lie on the dotted line. The y-axis shows the total error rate of a classifier, which is the same as the classifier cost (K) when the type 1 and type 2 misclassification costs are identical ($\kappa_1 = \kappa_2 = 1$). FNR: false negative rate. FPR: false positive rate.*

Click here for file

[http://www.biomedcentral.com/content/supplementary/1471-2105-9-385-S1.pdf]

## Additional file 2

*Comparison of unit cost, balancing cost, undersampling and oversampling for alleles A0203, A3101 and B1501. ROC curves for alleles A1101 (left panel) and B0702 (right panel) comparing the results of trees constructed with the oversampled training set (black curve), the undersampled training set (red curve), and the full training set without training costs, that is, $\lambda_1 = \lambda_2 = 1$ (green curve) and with the balancing training cost, that is, $\lambda_1 = 1$ and $\lambda_2 = (1 - p)/p$ (blue curve). Compare to figure 4. The ROC curves were constructed by varying the threshold used to label a node from 0 to 1 and evaluating its sensitivity and specificity at each threshold.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1471-2105-9-385-S2.pdf]

## Acknowledgements

We thank Cliburn Chan for insightful discussions, Yongting Cai for sharing experimental data used in our pilot studies and Kent Weinhold for his leadership and guidance. This work was supported by the Large Scale Antibody & T cell Epitope Discovery Program (Kent Weinhold, PI), under the NIH grant N01-A1-400822.

## References

1. Zhang C, Anderson A, DeLisi C: **Structural principles that govern the peptide-binding motifs of class I MHC molecules.** *J Mol Biol* 1998, **281(5):**929-47.
2. Peters B, Bui HH, Frankild S, Nielson M, Lundegaard C, Kostem E, Basch D, Lamberth K, Harndahl M, Fleri W, Wilson SS, Sidney J, Lund O, Buus S, Sette A: **A community resource benchmarking predictions of peptide binding to MHC-I molecules.** *PLoS Comput Biol* 2006, **2(6):**e65.
3. Rammensee H, Bachmann J, Emmerich NP, Bachor OA, Stevanovic S: **SYFPEITHI: database for MHC ligands and peptide motifs.** *Immunogenetics* 1999, **50(3–4):**213-9.
4. Parker KC, Bednarek MA, Coligan JE: **Scheme for ranking potential HLA-A2 binding peptides based on independent binding of individual peptide side-chains.** *J Immunol* 1994, **152:**163-75.
5. Gulukota K, Sidney J, Sette A, DeLisi C: **Two complementary methods for predicting peptides binding major histocompatibility complex molecules.** *J Mol Biol* 1997, **267(5):**1258-67.
6. Donnes P, Elofsson A: **Prediction of MHC class I binding peptides, using SVMHC.** *BMC Bioinformatics* 2002, **3:**25.
7. Yu K, Petrovsky N, Schonbach C, Koh JY, Brusic V: **Methods for prediction of peptide binding to MHC molecules: a comparative study.** *Mol Med* 2002, **8(3):**137-48.
8. Nielsen M, Lundegaard C, Worning P, Hvid CS, Lamberth K, Buus S, Brunak S, Lund O: **Improved prediction of MHC class I and class II epitopes using a novel Gibbs sampling approach.** *Bioinformatics* 2004, **20(9):**1388-97.
9. Segal MR, Cummings MP, Hubbard AE: **Relating amino acid sequence to phenotype: analysis of peptide-binding data.** *Biometrics* 2001, **57(2):**632-42.
10. Zhu S, Udaka K, Sidney J, Sette A, Aoki-Kinoshita KF, Mamitsuka H: **Improving MHC binding peptide prediction by incorporating binding data of auxiliary MHC molecules.** *Bioinformatics* 2006, **22(13):**1648-55.
11. Peters B, Tong W, Sidney J, Sette A, Weng Z: **Examining the independent binding assumption for binding of peptide epitopes to MHC-I molecules.** *Bioinformatics* 2003, **19(14):**1765-72.
12. Chawla NV, Japkowicz N, Kotcz A: **Editorial: special issue on learning from imbalanced data sets.** *SIGKDD Explor Newsl* 2004, **6:**1-6.
13. Brusic V, Zeleznikow J: **Computational binding assays of antigenic peptides.** *Letters in Peptide Science* 1999, **6:**313-324.
14. Elkan C: **The Foundations of Cost-Sensitive Learning.** *IJCAI* 2001:973-978.
15. Weiss GM, Provost FJ: **Learning When Training Data are Costly: The Effect of Class Distribution on Tree Induction.** *J Artif Intell Res (JAIR)* 2003, **19:**315-354.
16. Breiman L: *Classification and regression trees. Wadsworth statistics/probability series* New York, N.Y.: Chapman and Hall; 1993.
17. Ray S, Kepler T: **Amino acid biophysical properties in the statistical prediction of peptide-MHC class I binding.** *Immunome Research* 2007, **3:9** [http://www.immunome-research.com/content/3/1/9].
18. Drummond C, Holte R: **C4.5, Class Imbalance, and Cost-Sensitivity: Why Under-Sampling beats Over-Sampling.** *Proceedings of the International Conference on Machine Learning (ICML 2003) Workshop on Learning from Imbalanced Data Sets II* 2003.
19. Japkowicz N, Stephen S: **The class imbalance problem: A systematic study.** *Intelligent Data Analysis* 2002, **6:**429-449.
20. Kukar M, Kononenko I: **Cost-Sensitive Learning with Neural Networks.** *European Conference on Artificial Intelligence* 1998:445-449 [http://citeseer.ist.psu.edu/kukar98costsensitive.html].
21. Zhou ZH, Liu XY: **Training Cost-Sensitive Neural Networks with Methods Addressing the Class Imbalance Problem.** *IEEE Transactions on Knowledge and Data Engineering* 2006, **18:**63-77.
22. Brefeld U, Geibel P, Wysotzki F: **Support Vector Machines with Examples Dependent Costs.** *Lecture Notes in Computer Science* 2003:23-34.