BIOINFORMATICS ORIGINAL PAPER Vol. 24 no. 3 2008, pages 358–366
doi:10.1093/bioinformatics/btm611

Sequence analysis

Efficient peptide–MHC-I binding prediction for alleles with few known binders

Laurent Jacob* and Jean-Philippe Vert

Centre for Computational Biology, École des Mines de Paris, 35 rue Saint Honoré, 77305 Fontainebleau Cedex, France

Received on July 23, 2007; revised on November 6, 2007; accepted on December 7, 2007

Advance Access publication December 14, 2007

Associate Editor: Thomas Lengauer

ABSTRACT

Motivation: *In silico* methods for the prediction of antigenic peptides binding to MHC class I molecules play an increasingly important role in the identification of T-cell epitopes. Statistical and machine learning methods in particular are widely used to score candidate binders based on their similarity with known binders and non-binders. The genes coding for the MHC molecules, however, are highly polymorphic, and statistical methods have difficulties building models for alleles with few known binders. In this context, recent work has demonstrated the utility of leveraging information across alleles to improve the performance of the prediction.

Results: We design a support vector machine algorithm that is able to learn peptide–MHC-I binding models for many alleles simultaneously, by sharing binding information across alleles. The sharing of information is controlled by a user-defined measure of similarity between alleles. We show that this similarity can be defined in terms of supertypes, or more directly by comparing key residues known to play a role in the peptide–MHC binding. We illustrate the potential of this approach on various benchmark experiments where it outperforms other state-of-the-art methods.

Availability: The method is implemented on a web server: http://cbio.ensmp.fr/kiss. All data and codes are freely and publicly available from the authors.

Contact: laurent.jacob@ensmp.fr

Supplementary information: Supplementary data are available at *Bioinformatics* online.

1 INTRODUCTION

In silico computational methods for vaccine design are crucial tools to alleviate the cost and time required by the difficult tasks involved in the development of a vaccine. Recent reviews (Korber *et al*., 2006; Davies and Flower, 2007) highlight the growing importance of these approaches, ranging from epitope mapping to reverse vaccinology and related problems such as allergen and adjuvant discovery. Whereas in reverse vaccinology whole pathogen genomes are scanned in search of potential extracellular antigens (that can then be tested as vaccine subunits), epitope mapping attempts to predict directly which peptides can trigger an immune response. It can be applied to B-cell and T-cell epitopes, which both involve the recognition of peptides from the pathogen by some components of the immune system. In the case of T-cells with CD8+ receptors which are potential tools for the development of peptide vaccines, in particular for AIDS vaccines (McMichael and Hanke, 2002), and for the diagnosis and treatment of cancer (Wang, 1999; Sette *et al*., 2001), the immunogenicity of a peptide is contingent on its binding to MHC class I molecules. Not all peptides of a pathogen can bind to MHC-I molecules to be presented to T-cells: it is estimated that only 1 in 100 or 200 peptides actually binds to a particular MHC-I molecule (Yewdell and Bennink, 1999). Although binding is not a sufficient condition for a peptide to be an epitope, building a binding predictor is a good step towards predicting immunogenicity.

Numerous methods have been developed to address this binding prediction problem, as surveyed in several recent reviews (Davies and Flower, 2007; Korber *et al*., 2006). Structural approaches, on the one hand, try to evaluate how well a candidate binder fits in the binding groove of a MHC molecule, by various threading or docking approaches (Bui *et al*., 2006; Rosenfeld *et al*., 1995). In particular, QSAR models try to find peptides that optimize *in silico* the interaction between the peptide and the target molecule (Doytchinova *et al*., 2005). Sequence-based approaches, on the other hand, estimate predictive models for MHC-I binders by analyzing and learning from sets of known binders and non-binders. Models can be based on motifs (Rammensee *et al*., 1995), profiles (Rammensee *et al*., 1999) or machine learning methods like artificial neural networks (Nielsen *et al*., 2003; Zhang *et al*., 2005), hidden Markov models (Mamitsuka, 1998), support vector machines (Dönnes and Elofsson, 2002; Salomon and Flower, 2006), boosted metric learning (Hertz and Yanover, 2006) or logistic regression (Heckerman *et al*., 2006). Finally, some authors have recently proposed to combine structural and sequence-based approaches (Antes *et al*., 2006; Jojic *et al*., 2006). Although comparison is difficult, sequence-based approaches that learn a model from the analysis of known binders benefit from the accumulation of experimentally validated binders and will certainly continue to improve as more data become available.

The binding affinity of a peptide depends on the MHC molecule's 3D structure and physicochemical properties, which in turn vary between MHC alleles. This forces any prediction

*To whom correspondence should be addressed.

method to be allele-specific: indeed, the fact that a peptide can bind to an allele is neither sufficient nor necessary for it to be able to bind to another allele. Since MHC genes are highly polymorphic, some alleles have few if any known binders and non-binders. For such alleles predictive models for peptide binding can hardly be estimated with statistical machine learning approaches, because of the limited size of the training set. Thus, though achieving good precision in general, classical statistical and machine learning-based MHC–peptide-binding prediction methods fail to efficiently predict binding for these alleles.

Some alleles, however, can share binding properties. In particular, experimental work (Sette and Sidney, 1998) shows that different alleles can have overlapping peptide repertoires. This fact, together with the more recent observation of structural similarities among the alleles sharing their repertoires allows the definition of HLA allele supertypes, which are families of alleles exhibiting the same behavior in terms of peptide binding. This suggests that, although predictive models should remain allele-specific, sharing information about known binders across different but similar alleles has the potential to improve predictive models by increasing the quantity of data used to establish the model. For example, Zhu *et al*. (2006) show that simply pooling together known binders for different alleles of a given supertype to train a model can improve the accuracy of the model. Hertz and Yanover (2006) pool together binding data for all alleles simultaneously to learn a metric between peptides, which is then used to build predictive models for each allele. Finally, Heckerman *et al*. (2006) show that leveraging the information across MHC alleles and supertypes considerably improves individual allele prediction accuracy.

In this article, we go one step further in this direction and propose a new method to predict peptide binding to MHC class I molecules, even for alleles with few known binders. Following the idea of Heckerman *et al*. (2006), our method estimates different predictive models for different alleles, but uses training data available for 'similar' alleles to tune each individual model. The notion of allele similarity is user-defined and should typically include prior biological knowledge about which allele features are related to their peptide binding specificities. For example, we propose several measures of allele similarity based on supertype information, like in the work of Heckerman *et al*. (2006), or on which amino-acids are present in the binding site of the MHC-I molecule.

From a technical point of view, the method is based on the support vector machine (SVM) algorithm, a state-of-the-art machine learning algorithm which has already been successfully applied to the problem of peptide–MHC I binding prediction for individual alleles (Dönnes and Elofsson, 2002; Salomon and Flower, 2006). We show how this algorithm can be modified to allow the estimation of predictive models for all alleles simultaneously, by sharing binding information across different alleles. The modification boils down to adding to the usual SVM formulation an additional user-defined measure of similarity between alleles. When the allele similarity is set to 0 between different alleles, then we recover the classical setting of learning individual models for each allele as in Dönnes and Elofsson (2002); Salomon and Flower (2006), while a non-zero similarity measure allows sharing of information across alleles. Compared to Heckerman *et al*. (2006), our method allows more flexibility in the way allele similarity is defined in a computational efficient framework, and we show that it results in more accurate predictive models.

We validate our method on several benchmark datasets, where it outperforms a number of other state-of-the-art methods. We show that, as expected, the performance improvement is particularly noticeable for alleles with few known binding peptides. The resulting models are implemented on a freely and publicly available companion web server available at http://cbio.ensmp.fr/kiss.

2 METHODS

In this section, we describe our method to share information across different alleles when training allele-specific models with SVM. We begin with a non-technical overview of the method, before showing more formally how it extends naturally the classical framework of learning with SVM.

2.1 Overview of the method

The aim of our method is to build, for each known MHC class I allele, a model to predict whether or not candidate peptides can bind to it. For that purpose we assume that a list of peptides already known to bind or not to bind some alleles is available. Such a list, which we call the training set below, can be obtained from various databases dedicated to immunoinformatics. Our method must then learn the binding predictive models from the training set.

A classical approach to build such models is to consider the different alleles separately, and to build one model for each allele based on the binding data available for this allele only. Many machine learning algorithms can be used in this context, including for example SVM or artificial neural networks (ANN). In essence, machine learning methods attempt to find a discriminative rule that separates binding from non-binding peptides in the training set. In the case of ANN or SVM, this rule is a linear or non-linear function of a set of features that are chosen to describe various properties of the peptides, such as which amino-acid it contains at each position.

While machine learning methods tend to produce very accurate models when enough training data are available, they can also lead to very poor models when the training set is too small. In our case, this means that this approach is not relevant for alleles with few known binders and non-binders. For example, Dönnes and Elofsson (2002) noticed that in the case of SVM, the accuracy of the model significantly decreases when $\leq$50 known peptides are used as training set, and they suggest not even to try building models for alleles with $\leq$20 known peptides. As a result, predictive models for alleles with few or no known binding peptide are rarely available for methods that follow this strategy.

The lack of a large amount of known peptides for some alleles can however be balanced, to some extent, by the fact that the sets of binding peptides sometimes largely overlap between different alleles. Typically, if an allele with no or few known binding peptides is structurally very similar to another one with many known binding peptides, then the known binding peptides of the second allele are likely also to bind the first one. Therefore these peptides may also be used as positive examples to fit the model for the first allele, but should be given less 'weight' than the peptides known for sure to bind the first allele. More generally, the predictive model for an allele could be tuned by fitting its parameters not only to the data available for this allele, but also from similar alleles, giving more importance to the data coming from the alleles that are more similar.

To transform this line of thoughts into a working algorithm we follow the approach pioneered by Heckerman *et al.* (2006) who proposed a framework to estimate allele-specific models which nonetheless share common properties across alleles. In their work, each individual allele model decomposes as a sum of three partial models: one specific to each allele, one shared across all alleles in a supertype and one shared across all alleles. Typically, the model shared across all alleles is meant to capture general binding properties of peptides, while the supertype- or allele-specific models are meant to capture the specific features that make a peptide able to bind all alleles within a supertype, or only a particular allele. Interestingly, Heckerman *et al.* (2006) show that all models can be estimated simultaneously, using classical machine learning algorithms. Indeed, they show that if each MHC allele is represented by a binary vector of features to indicate, in particular, to which supertype it belongs, then one just needs to represent each training example of the form 'peptide p binds (respectively does not bind) to allele a' by a vector to represent the *peptide/allele pair* (p, a), whose features are *products* of features of the peptide p and of the allele a, before applying a machine learning algorithms. By applying this strategy, the model of a given allele is tuned using only this allele's data for the allele-specific partial model, the data corresponding to the alleles in the same supertype for the supertype-specific partial model, and the data of all alleles for the non-specific partial models. This amounts to a form of data sharing across alleles, where binding data for the allele of interest are given more weight than binding data for different alleles in the same supertype, which are themselves given more weights than binding data for alleles in different supertypes.

While extremely appealing, this approach has several limitations, in particular when we want to extend the notion of allele similarity beyond the supertype information only:

- It is not easy to figure out systematic strategies to exploit prior information about alleles in the construction of their descriptors, needed in this approach. For example, including more prior knowledge about when two alleles should share more information, e.g. based on structural similarity between alleles, is not an obvious task.
- Practically, a training point of the form 'peptide p binds allele a' is represented by all possible products of features of p by a feature of a. If we increase the number of features of a beyond the simple supertype information, then the dimension of the vectors to be manipulated becomes large which makes statistical estimation, storage, manipulation and optimization tasks much harder.

In order to allow the inclusion of richer prior knowledge, such as sequence or structure similarity, to describe when two alleles are likely to bind similar peptides, we reformulate below the approach of Heckerman *et al.* (2006) in a different but equivalent mathematical framework. The reformulation is based on the observation that certain machine learning algorithms, such as SVM, only manipulate data through pairwise inner products, and that the inner products between the vectors we consider when we want to share information across alleles can be decomposed into an inner product between allele features, on the one hand, and an inner product between peptide features, on the other hand (Section 2.2). This apparently technical fact has important practical consequences. First, it means that the choices of peptide description and allele description can be completely uncoupled, and that any relevant peptide description can be combined with any relevant allele description. Second, from a practical point of view, the size of the data to be stored and of the problems to be solved does not scale as the product of the peptide and allele dimensions, as in Heckerman *et al.* (2006), but is in fact independent of these dimensions as long as the inner products between alleles and peptides can be easily computed. This paves the way to the use of rich descriptions of peptides and alleles, which we discuss in Sections 2.3 and 2.4, respectively.

While this framework is computationally efficient in terms of richness of representation for peptides and alleles, it should be pointed out, however, that the resulting algorithm is an SVM, which in its basic form is known to scale poorly with respect to the number of training points. This potential issue is however likely to benefit from recent and future developments of large-scale SVM implementations, which can already process up to millions of examples in a reasonable time (Bottou *et al.*, 2007).

2.2 Kernel formulation

In order to learn models for all alleles simultaneously by sharing binding information across similar alleles, we follow the approach proposed by Heckerman *et al.* (2006) which consists in four steps: (i) represent each peptide p by a vector of descriptors $\Phi_{\text{pep}}(p)$; (ii) represent each allele a by a vector of descriptors $\Phi_{\text{all}}(a)$ in such a way that alleles with similar descriptors are likely to share similar binding peptides; (iii) for each training example of the form 'peptide p binds (respectively does not bind) allele a', form the joint vector $\Phi(p, a)$ obtained by taking all products of a descriptor of a by a descriptor of p and assign it the class 'bind' (respectively 'does not bind'); (iv) tune a machine learning algorithm on the training set to separate binding pairs from non-binding pairs.

In mathematical terms, the vector $\Phi(p, a)$ built in step (iii) is called the *tensor product* and denoted:

$$\Phi(p, a) = \Phi_{\text{pep}}(p) \otimes \Phi_{\text{all}}(a).$$

If the peptide description vector $\Phi_{\text{pep}}(p)$ is made of d_p descriptors, and the allele description vector $\Phi_{\text{all}}(a)$ is made of d_a descriptors, then the vector $\Phi(p, a)$ that describes the pair is made of $d_p \times d_a$ descriptors. This number can be prohibitively large for practical applications: for example, using as features all co-occurrences of any two amino-acids at each of the nine positions in a 9mer peptide leads to $d_p = (9 \times 20)^2 = 32\,761$, and doing the same at 35 key residues in the MHC molecule results in $d_a = (35 \times 20)^2 = 490\,000$. In that case the features vector for a peptide/allele pair would have of the order of 10^{10} descriptors, making it intractable for methods like the logistic regression used in Heckerman *et al.* (2006).

However, if the machine learning algorithm used in step (iv) is a SVM, or more generally belongs to the family of algorithms called kernel methods (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004), then the explicit computation of $\Phi(p, a)$ in step (iii) is not required. Instead, what is needed is a fast way to compute the inner product between the vectors $\Phi(p, a)$ and $\Phi(p', a')$ for any two peptide/allele pairs in the training set. If this inner product can be computed, then there is no need to compute, manipulate or store the vectors $\Phi(p, a)$, a property often referred to as the *kernel trick*. Fortunately, a classical and easy to check property of tensor products allows to write the inner product between two tensor product vectors as a product of inner products as follows (where for any two vectors u and v we denote by $u^\top v$ their inner product):

$$\begin{aligned}\Phi(p, a)^\top \Phi(p', a') &= \left(\Phi_{\text{pep}}(p) \otimes \Phi_{\text{all}}(a)\right)^\top \left(\Phi_{\text{pep}}(p') \otimes \Phi_{\text{all}}(a')\right) \\ &= \Phi_{\text{pep}}(p)^\top \Phi_{\text{pep}}(p') \times \Phi_{\text{all}}(a)^\top \Phi_{\text{all}}(a').\end{aligned}$$

In other words, the inner product between peptide/allele pairs can be decomposed as a product between the inner product between peptides, on the one hand, and the inner product between alleles, on the other hand. This property dramatically reduces the effective dimension of the problem: instead of manipulating vectors of dimensions $d_p \times d_a$, we can just manipulate peptide and allele vectors, of respective dimensions d_p and d_a.

This complexity can even be further reduced when the inner products between peptides and those between alleles are themselves fast to compute. This is the case in particular when one uses a particular

kernel function to define these inner products, benefiting from a variety of recent work in bioinformatics (Schölkopf *et al.*, 2004). Kernel functions generalize inner products in the sense that they are pairwise comparison function between objects, which under some simple conditions (Aronszajn, 1950) are guaranteed to correspond to inner products between given descriptions of the objects. These descriptions can be very complex, in high or infinite dimension, but do not need to be explicited since only the inner product (given by the kernel function) is needed. Using kernels between peptides and between alleles can be useful for various reasons, in particular:

- d_p and d_a can be very large, even infinite, yet there may be an efficient way to compute the inner products.
- One can want to use existing kernels and simply apply them on the peptides and the(e.g. Schöelkopf *et al.,* 2004). For example, the fact that non-linear kernels such as Gaussian or polynomial kernels for peptides give good results for SVMs trained on individual alleles suggests that they are natural candidates for the peptide part of the product kernel.

By simply rewriting

$$K_{\text{pep}}(p, p') = \Phi_{\text{pep}}(p)^\top \Phi_{\text{pep}}(p'),$$
$$K_{\text{all}}(a, a') = \Phi_{\text{all}}(a)^\top \Phi_{\text{all}}(a'),$$

we obtain the inner product between tensor products by:

$$K((p, a), (p', a')) = K_{\text{pep}}(p, p') \times K_{\text{all}}(a, a'), \quad (1)$$

in which any existing kernel or any fast way to compute inner product between peptide and allele representation can be plugged.

To summarize, when expressed in these terms, the problem of sharing training data across the alleles in a systematic and computationally efficient way boils down to choosing a description (explicit or by a pairwise similarity kernel) for the peptides, another one for the alleles, and apply any classical learning machinery such as SVM to peptide-allele pairs using the product kernel (1). We can now discuss in more details the choice of the peptide and allele kernels in Sections 2.3 and 2.4, respectively.

2.3 Peptide kernels

We consider in this article peptides made of 9 amino-acids, although extensions to variable-length peptides poses no difficulty in principle (Salomon and Flower, 2006). The classical way to represent these 9mers as fixed length vectors is to encode the letter at each position by a 20-dimensional binary vector indicating which amino acid is present, resulting in a 180-dimensional vector representations. In terms of kernel, the inner product between two peptides in this representation is simply the number of letters they have in common at the same positions, which we take as our baseline kernel:

$$K_{\text{linseq}}(x, x') = \sum_{i=1}^{l} \delta(x[i], x'[i]),$$

where l is the length of the peptides (9 in our case), $x[i]$ is the i-th residue in x and $\delta(\text{x}[i], x'[i])$ is 1 if $x[i] = x'[i]$, 0 otherwise.

Alternatively, several authors have noted that non-linear variants of the linear kernel can improve the performance of SVM for epitope prediction (Bhasin and Raghava, 2004; Dönnes and Elofsson, 2002; Zhao *et al.*, 2003). In particular, using a polynomial kernel of degree p over the baseline kernel is equivalent, in terms of feature space, to encoding p-order interactions between amino-acids at different positions. In order to assess the relevance of such non-linear extensions we tested this polynomial kernel,

$$K_{\text{poly}}(x, x') = \left(K_{\text{linseq}}(x, x') + 1\right)^p.$$

The degree $p = 2$ was selected by internal cross validation on the training data of our first benchmark dataset. We kept this value for the other benchmarks.

In order to limit the risk of overfitting on the benchmark data we restrict ourselves to the evaluation of the baseline linear kernel and its non-linear (polynomial) extension. Designing a specific peptide kernel for binding prediction, e.g. by weighting differently the positions known to be critical in the MHC–peptide complex, is however an interesting research topic that could bring further improvements in the future.

2.4 Allele kernels

Although the question of kernel design for peptides has been raised in previous studies involving SVM for epitope prediction (Bhasin and Raghava, 2004; Dönnes and Elofsson, 2002; Salomon and Flower, 2006; Zhao *et al.*, 2003), the question of kernel design for alleles is new to our knowledge. We tested several approaches:

- The *Dirac* kernel is:

  $$K_{\text{Dirac}}(a, a') = \begin{cases} 1 & \text{if } a = a', \\ 0 & \text{otherwise.} \end{cases}$$

  for the Dirac kernel, no information is shared across alleles and the SVM learns one model for each allele independently from the others. Therefore this corresponds to the classical setting of learning binding prediction models independently for each allele with SVM. This is easily seen if we consider that the decision function learned by the SVM from n $(peptide, allele)$ pairs $\{(p_1, a_1) \ldots (p_n, a_n)\}$ has the form:

  $$f(p, a) = \sum_{i=1}^{n} \alpha_i K_{\text{pep}}(p, p_i) K_{\text{all}}(a, a_i), \alpha_i \in \mathbb{R},$$

  which implies that only pairs with $a_i = a$ are taken into account to classify a pair involving allele a (the other terms of the sum are 0 because of the *Dirac* kernel). From the feature space point of view, using this *Dirac* kernel amounts to describing the pairs in a space such that pairs involving different alleles be in orthogonal subspaces. Therefore, it is not possible for a pair involving an allele a to participate in the discriminative hyperplane of allele $a' \neq a$.
- The *uniform* kernel is:

  $$K_{\text{uniform}}(a, a') = 1 \text{ for all } a, a'.$$

  With this kernel all alleles are considered the same, and a unique model is created by pooling together the data available for all alleles. This is easily seen from the feature space associated to this kernel being simply an identity mapping, i.e. $\forall\, a, a'$, $\Phi(x, a) = \Phi(x, a')$, so discriminating (peptide,allele) pairs in this space amounts to discriminating pooled peptides.
- The *multitask* kernel is:

  $$K_{\text{multitask}}(a, a') = K_{\text{dirac}}(a, a') + K_{\text{uniform}}(a, a').$$

  As explained in the previous section and in Evgeniou *et al.* (2005) this is the simplest way to train different but related models. The SVM learns one model for each allele, using known binders and non-binders for the allele, but using also known binders and non-binders for all other alleles with a smaller contribution. The training peptides are shared uniformly across different alleles.
- The *supertype* kernel is

  $$K_{\text{supertype}}(a, a') = K_{\text{multitask}} + \delta_s(a, a'),$$

  where $\delta_s(a, a')$ is 1 if a and a' are in the same supertype, 0 otherwise. As explained in the previous section this scheme trains a specific model for each allele using training peptides from different alleles, but here the training peptides are more shared across alleles within

Table 1. Residue positions involved in the binding site for the three loci, according to Doytchinova *et al.* 2004

Locus	Positions
HLA-A	5, 7, 9, 24, 25, 34, 45, 59, 63, 66, 67, 70, 74, 77, 80, 81, 84, 97, 99, 113, 114, 116, 123, 133, 143, 146, 147, 152, 155, 156, 159, 160, 163, 167, 171
HLA-B	5, 7, 8, 9, 24, 45, 59, 62, 63, 65, 66, 67, 70, 73, 74, 76, 77, 80, 81, 84, 95, 97, 99, 114, 116, 123, 143, 146, 147, 152, 155, 156, 159, 160, 163, 167, 171
HLA-C	5, 7, 9, 22, 59, 62, 64, 66, 67, 69, 70, 73, 74, 77, 80, 81, 84, 95, 97, 99, 116, 123, 124, 143, 146, 147, 156, 159, 163, 164, 167, 171

a supertype than across alleles in different supertypes. This is used by Heckerman *et al.* (2006), without the kernel formulation, to train a logistic regression model.

Heckerman *et al.* (2006) show that the supertype kernel generally improves the performance of logistic regression models compared to the uniform or Dirac kernel. Intuitively it seems to be an interesting way to include prior knowledge about alleles. However, one should be careful since the definition of supertypes is based on the comparison of binders of different alleles, which suggests that the supertype information might be based on some information used to assess the performance of the method in the benchmark experiment. In order to overcome this issue, and illustrate the possibilities offered by our formulation, we also tested a kernel between alleles which tries to quantify the similarity of alleles without using known binders information. For that purpose we reasoned that alleles with similar residues at the positions involved in the peptide binding were more likely to have similar binders, and decided to make a kernel between alleles based on this information. For each locus we gathered from Doytchinova *et al.* (2004) the list of positions involved in the binding site of the peptide (Table 1). Taking the union of these sets of positions we then represented each allele by the list of residues at these positions, and used the same polynomial kernel used for the peptides to compare two sequences of residues associated to two alleles, i.e.,

$$K_{\text{poly}}(a, a') = \left(K_{\text{linseq}}(a, a') + 1\right)^p.$$

where we extend K_{linseq} to the allele space by:

$$K_{\text{linseq}}(a, a') = \sum_{i \in \text{bsite}} \delta(a[i]a'[i]),$$

where bsite is the set of residues known to be in the binding site for one of the three allele groups HLA-A, B, C, $a[i]$ is the i-th residue in a and $\delta(a[i]a'[i])$ is 1 if $a[i] = a'[i]$, 0 otherwise. Here again, a degree of $p = 7$ was selected by internal cross validation on the first benchmark and kept for the other experiments.

2.5 SVM

We learned binding models with SVM, a state-of-the-art algorithm for pattern recognition (Schölkopf and Smola, 2002; Schölkopf *et al.*, 2004; Vapnik, 1998). We used the PyML library (http://pyml.sourceforge.org), and its SVM implementation with a custom kernel to account for the various kernels we tested. All kernels were normalized to one on the diagonal. Besides the kernel, SVM depends on one parameter usually called C. For each experiment, we selected the best C among the values $2^i, \in \{-15, -14, \ldots, 9, 10\}$ by selecting the value leading to the largest area under the ROC curve estimated by cross-validation on the training set only. The performance of each method was then tested on each experiment by evaluating the AUC over the test data.

3 DATA

In order to evaluate both the performance of our method and the impact of using various kernels for the peptides or the alleles, we tested our method on three different benchmark datasets that were recently compiled. The first two datasets were used in Heckerman *et al.* (2006) to evaluate the performance of leveraging logistic regression on epitope prediction. Since this method already improved prediction accuracy with respect to the best published results, we use these benchmarks to compare our method with state-of-the-art methods. We note that the dataset was created with the goal of predicting epitopes and not MHC-I binding, but our method can be applied without any change in this slightly different context. The third dataset, on the other hand, was specifically designed as a reference to compare MHC-I binding prediction methods.

The first dataset, called SYFPEITHI+LANL, combines experimentally confirmed positive epitopes from the SYFPEITHI database (see Rammensee *et al.*, 1999, available at http://www.syfpeithi.de) and from the Los Alamos HIV database (http://www.hiv.lanl.gov). Negative example were randomly drawn from the HLA and amino-acid distribution in the positive examples, for a total of 3152 data points. Since this dataset is quite small and was already used as a benchmark, we use it as a first performance evaluation, and to compare our kernels.

The second dataset of Heckerman *et al.* (2006) contains 160 085 peptides including those from SYFPEITHI+LANL and others from the MHCBN data repository (see Bhasin *et al.*, 2003, available at http://www.imtech.res.in/raghava/mhcbn/index.html. In the following, it will be referred to as MHCBN+SYFPEITHI+LANL. This corresponds to 1585 experimentally validated epitopes, and 158 500 randomly generated non-binders (100 for each positive). In the interest of time, we only kept 50 negative for each positive for the training step, but tested on all the original testing points. We assumed that this would not deteriorate too much the performance of our algorithm. In the worst case, it is only a handicap for our methods.

Finally, we assess the performance of our method on the MHC–peptide binding benchmark recently proposed by Peters *et al.* (2006) who gathered quantitative peptide-binding affinity measurements for various species, MHC class I alleles and peptide lengths, which makes it an excellent tool to compare MHC–peptide binding learning methods. We focused on the 9mer peptides for the 35 human alleles and thresholded at $\text{IC50} = 500$, a classical choice to separate binders ($\text{IC50} < 500$) and non-binders ($\text{IC50} \geq 500$), see Sette *et al.* (1994). Nevertheless, the application of our method to other species or peptide lengths would be straightforward, and generalization to quantitative prediction should not be too problematic either. The benchmark contains 29 336 9-mers.

Five-fold cross validation was used on the first dataset, 10-fold on the second, except for the HIV (LANL) data which is split up into folds only for testing, and never for training. Five-fold cross validation was also used on the third dataset. We used the same folds as Heckerman *et al.* (2006), available

Table 2. AUC results for an SVM trained on the SYFPEITHI+LANL with various kernels and estimated error on the 5 folds

$K_{all}\backslash K_{pep}$	Linseq	Poly
uniform	0.826 ± 0.010	0.866 ± 0.010
Dirac	0.891 ± 0.014	0.893 ± 0.017
multitask	0.910 ± 0.008	0.929 ± 0.011
supertype	0.924 ± 0.011	0.930 ± 0.010
poly	0.920 ± 0.011	0.930 ± 0.010

at ftp://ftp.research.microsoft.com/users/heckerma/recomb06 for the first two datasets and the same folds as Peters *et al*. (2006) available at http://mhcbindingpredictions.immuneepitope.org/ for the third one.

Molecule-based allele kernels require the amino-acid sequences corresponding to each allele. These sequences are available in various databases, including http://www.anthonynolan.org.uk/ and Robinson *et al*. (2000). We used the peptide-sequence alignment for HLA-A, HLA-B and HLA-C loci. Each sequence was restricted to residues at positions involved in the binding site of one of the three loci, see Table 1. Preliminary experiments showed that using this restriction instead of whole sequences did not change the performance significantly, but indeed speeds up the calculation of the kernel.

We were not able to find the sequence of a few molecules of the two datasets of Heckerman *et al*. (2006), so in the experiments involving these datasets and a molecule-based allele kernel, we used $K_{poly}(a, a') + K_{multitask}(a, a')$ instead of simply using $K_{poly}(a, a')$, with a value of $K_{poly}(a, a') = \delta(a, a')$ in the cases where the sequence was unknown. This is the sum of two kernels, so still a positive definite kernel and actually exactly the same thing as $K_{supertype}$ but using K_{poly} instead of δ_s.

4 RESULTS

In the following, $K_{pep} \times K_{all}$ indicates the use of SVM with a product kernel of K_{pep} for the peptides and K_{all} for the alleles.

We first use K_{linseq} and K_{poly} for the peptides and $K_{uniform}$ (one SVM for all the alleles), K_{Dirac} (one SVM for each allele), $K_{multitask}$, $K_{supertype}$ and K_{poly} for the alleles on the small SYFPEITHI+LANL dataset. Using combinations of molecule-based and non-molecule-based kernels for K_{all} did not improve the prediction, generally the result was as good as or slightly worse than the result obtained with the best of the two combined kernels. Results are displayed in Table 2, and ROC curves for $K_{linseq} \times K_{Dirac}$, $K_{linseq} \times K_{supertype}$, $K_{poly} \times K_{supertype}$ and $K_{poly} \times K_{poly}$ in Figure 1.

Table 2 demonstrates the benefits of sharing information across alleles. The *Dirac* allele kernel being the baseline kernel corresponding to independent training of SVMs on different alleles, we observe an improvement of at least 2% when information is shared across alleles during training (with the *multitask, supertype* or *poly* strategies). It should be noted, however, that the *uniform* strategies which amount to training a single model for all alleles perform considerably worse than the *Dirac* strategies, justifying the fact that it is still better to build individual models than a single pooling model for all alleles. On this benchmark, we did not observe any significant difference between the three strategies to share information when using a non-linear kernel for the peptides, yet the use of biological knowledge like supertype or allele sequence improves the performance with respect to the naive multitask approach when using a linear kernel for the peptides. However, one should keep in mind that there is a risk of bias in the performance of the *supertype* kernel, because some peptides in the test sets might have contributed to the definition of the allele supertypes. Since the *poly* kernel, which shares more information between alleles that have similar residues at key positions, performs equally well, it can be advantageously used to include biological knowledge in the learning process.

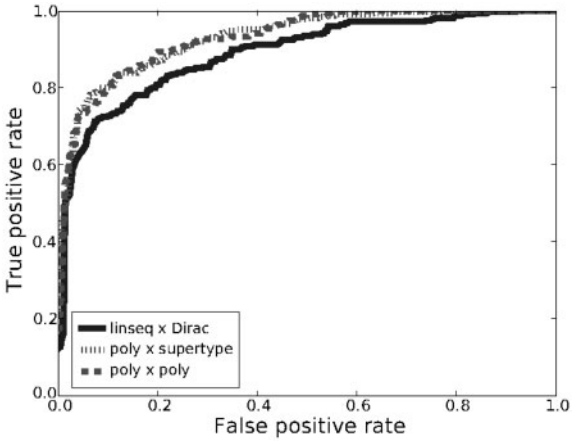

Fig. 1. ROC curves on the 5-folds of the SYFPEITHI+LANL benchmark.

Finally, we observe that for all allele kernels, the non-linear *poly* peptide kernel outperforms the baseline *linseq* kernel, which confirms that linear models based on position-specific score matrices might be too restrictive a set of models to predict accurately MHC–peptide binding.

In terms of performance, all three allele kernels that share information across alleles combined with the non-linear *poly* peptide kernel outperform the leveraged logistic regression of Heckerman *et al*. (2006) (AUC $= 0.906 \pm 0.016$ against 0.930 ± 0.010 for the same supertype kernel associated to a non-linear peptide kernel) and the boosted distance metric learning algorithm of Hertz and Yanover (2006) (AUC $= 0.819 \pm 0.055$). As the boosted distance metric learning approach was shown to be superior to a variety of state-of-the-art methods by Hertz and Yanover (2006), this suggest that our approach can compete if not overcome the best methods in terms of accuracy.

From Table 2, we see that two factors are involved in the improvement over the method of Heckerman *et al*. (2006):

- The use of an SVM instead of a logistic regression, since this is the only difference between the leveraged logistic regression and our SVM with a $K_{linseq} \times K_{supertype}$ kernel. This, however, may not be intrinsic to the algorithms, but caused by optimization issues for the logistic regression in high dimension.

Table 3. AUC results for an SVM trained on the MHCBN+SYFPEITHI+LANL benchmark with various kernels and estimated error on the 10-folds

Method	AUC
Leveraged LR (from Heckerman *et al.* (2006))	0.906
$K_{\text{linseq}} \times K_{\text{stype}}$	0.911 ± 0.010
$K_{\text{poly}} \times K_{\text{dirac}}$	0.876 ± 0.010
$K_{\text{poly}} \times K_{\text{multitask}}$	0.938 ± 0.007
$K_{\text{poly}} \times K_{\text{supertype}}$	0.934 ± 0.007
$K_{\text{poly}} \times K_{\text{poly}}$	0.938 ± 0.007

- The use of a non-linear kernel for the peptide, as we observe a clear improvement in the case of SVM (this improvement might therefore also appear if the logistic regression was replaced by a kernel logistic regression model with the adequate kernel).

Figure 1 illustrates the improvement underlined by this experiment: from the individual SVM ($K_{\text{linseq}} \times K_{\text{Dirac}}$), to $K_{\text{poly}} \times K_{\text{supertype}}$ and $K_{\text{poly}} \times K_{\text{poly}}$ SVM that both give better performances than $K_{\text{linseq}} \times K_{\text{Dirac}}$ SVM because they use multitask strategies and a non-linear kernel to compare the peptides.

These results are confirmed by the MHCBN+SYFPEITHI+LANL benchmark, for which the results are displayed in Table 3. Again, the use of SVM with our product kernels improves the performance with respect to Heckerman *et al.* (2006) (from 0.906 to 0.938). Moreover, we observe that learning a leveraged predictor using the data from all the alleles strongly improves the global performance, hence the important step between Dirac (0.876) and all the multitask-based methods, including the simplest multitask kernel (0.938). It is worth reminding that the multitask kernel is nothing but the sum of the Dirac and uniform kernels, i.e. that it contains no additional biological information: the improvement is caused by the mere fact of using roughly (with a weighting of 0.5) the points of all the other alleles to learn the predictor of one allele. Figure 4 in the Supplementary Material shows the ROC curves for SVM with $K_{\text{poly}} \times K_{\text{Dirac}}$, $K_{\text{poly}} \times K_{\text{supertype}}$ and $K_{\text{poly}} \times K_{\text{poly}}$ kernels on this benchmark. Again, we see the clear improvement between leveraged and non-leveraged strategies. The difference between the $K_{\text{poly}} \times K_{\text{Dirac}}$ and the two others is only caused by leveraging, since in the three case the same non-linear strategy was used for the peptide part. On the other hand, the figure illustrates that our three strategies for leveraging across alleles, combined with non-linear kernels for the peptides, give roughly the same result on this benchmark.

Finally, Table 4 presents the performance on the IEDB benchmark proposed in Peters *et al.* (2006). The indicated performance corresponds, for each method, to the average on the AUC for each of the 35 alleles, which gives an indication of the global performances of each methods.

Here again, the quantitative jump between individual and leveraged strategies is illustrated by the ROC curves of Figure 5 in Supplementary Material that shows the performances of $K_{\text{poly}} \times K_{\text{Dirac}}$, and $K_{\text{poly}} \times K_{\text{poly}}$ strategies on the benchmark.

Table 4. AUC results for an SVM trained on the IEDB benchmark with various methods

Method	AUC
SVM with $K_{\text{poly}} \times K_{\text{Dirac}}$ kernel	0.879
SVM with $K_{\text{poly}} \times K_{\text{supertype}}$ kernel	0.893
SVM with $K_{\text{poly}} \times K_{\text{poly}}$ kernel	0.903
ADT	0.874
ANN	0.897

The ANN column of Table 4 corresponds to the tool proposed in Peters *et al.* (2006) web server with the best results on the 9mer dataset. This is an artificial neural network proposed in Nielsen *et al.* (2003). The ADT field refers to the adaptive double threading approach recently proposed in Jojic *et al.* (2006) and tested on the same benchmark.

These tools were compared to and significantly outperformed other tools in the comprehensive study of Peters *et al.* (2006), specifically Peters and Sette (2005) and Bui *et al.* (2005), that are both scoring-matrix-based. Our approach gives slightly better results in terms of global performances than Nielsen *et al.* (2003) (0.903 against 0.898), and therefore outperforms the other internal methods. To our knowledge, no other method had reached this performance on this dataset (actually no external method has been tested on all the alleles).

It is actually difficult to compare precisely our method with the ANN since the parameters of the latter are adjusted to the data. During the ANN evaluation procedure, for each fold various models were trained on the training set and the one that gave the best results on the testing set was selected, whereas all the parameters of our model were selected by internal cross validation on the training set. The fact that the multitask kernel approach is more accurate even in this setting is a strong illustration of its good performance with respect to state-of-the-art methods on this benchmark.

Table 5 presents the performances of the $K_{\text{poly}} \times K_{\text{poly}}$ strategy on the 10 alleles with $\leq$200 training points, together with the performances of the best internal tool, the ANN of Nielsen *et al.* (2003), and the adaptive double threading model that gave good prediction performances on the alleles with few training data. Except for two cases, our SVM outperforms both models, with an average improvement of 3.4% of AUC with respect to the ANN method and 7.6% with respect to the ADT approach, again without touching the test set when fitting our parameters since they were selected by internal cross validation. As we said in introduction, our original concern was to improve binding prediction for alleles with few training points, and for which it is hard to generalize. This was the main point of using a multitask learning approach. The results on this last benchmark suggest that the leveraging approaches succeed in improving prediction performances when few training points are available.

5 DISCUSSION AND CONCLUDING REMARKS

In this article, we introduce a general framework to share efficiently the binding information available for various alleles

Table 5. Detail of the IEDB benchmark for the 10 alleles with ≤200 training points (9mer data)

Allele	Peptide number	$K_{poly} \times K_{poly}$	ADT	ANN
A2301	104	**0.881** ± 0.028	0.804	0.852
A2402	197	0.801 ± 0.009	0.785	**0.825**
A2902	160	**0.942** ± 0.018	0.887	0.935
A3002	92	**0.826** ± 0.034	0.763	0.744
B1801	118	**0.909** ± 0.025	0.869	0.838
B4002	118	0.806 ± 0.015	**0.819**	0.754
B4402	119	**0.797** ± 0.056	0.677	0.778
B4403	119	**0.831** ± 0.037	0.624	0.763
B4501	114	**0.895** ± 0.024	0.801	0.862
B5701	59	**0.931** ± 0.045	0.832	0.926
Mean		0.862	0.786	0.828

by simply defining a kernel for the peptides, and another one for the alleles. The result is a model for MHC–peptide binding prediction that uses information from the whole dataset to make specific prediction for any of the alleles. Our approach is simple, general and both easy to adapt to a specific problem by using more adequate kernels, and to implement, by running any SVM implementation with these kernels. Everything is performed in low dimensions and with no need for feature selection.

We presented performances on three benchmarks. On the three benchmarks, our approach performed better than state-of-the-art results, which illustrates its generally good behavior in terms of prediction accuracy. These experiments confirmed the interest of leveraging the information across the alleles, especially when trying to predict peptide binding for an allele with few training data available. They also illustrated the interest of using non-linear kernels in terms of performance accuracy. We believe highly accurate prediction models are crucial for vaccine design, and complementary to more analysis-oriented models.

The method we describe has been implemented and is freely available through a KISS (Kernel-based Inter-allele peptide binding prediction SyStem) web server: http://cbio.ensmp.fr/kiss. For this implementation, we trained an SVM using $K_{poly} \times K_{poly}$ kernel on the concatenation of the MHCBN+SYFPEITHI+LANL and IEDB databases, removing all duplicates. This tool is intended to predict MHC-I binding, but could as well be used directly as an efficient approximation for epitope prediction.

Although the kernels we used already gave good performances, there is still room for improvement. A first way to improve the performances would be to use more adequate kernels to compare the peptides and, probably more importantly, to compare the alleles. In other words we would be answering the question: what does it mean in the context of MHC–peptide-binding prediction for two alleles to be similar? Possible answers should probably involve better kernels for the allele sequences, and structural information which could be crucial to predict binding and, as we said in introduction, is already used in some models. Finally, it could be useful to incorporate the quantitative IC50 information when available, instead of simply thresholding as we did for the last benchmark.

Using the binding affinity information, it is possible to apply our general framework to predict quantitative values, using regression models with the same type of kernels. More generally, it would be interesting, and more useful in terms of vaccine design, to try to predict immunogenicity of the peptides instead of the mere MHC class I binding, as it has been proposed, for example, in Tung and Ho (2007). This could be done by adding to the peptide kernel information relevant to the other major steps of the MHC-I pathway, i.e. proteasome cleavage and TAP transporter affinity.

The multitask kernels could also be used for a lot of similar problems involving binding, like MHC-II-peptide binding where sequences can have variable length and the alignment of epitopes, usually performed as pre-processing, can be ambiguous. Salomon and Flower (2006) has already proposed a kernel for this case. Another interesting application for the general approach of predicting affinity between small molecules and families of related proteins would be drug design, for example protein-kinase-inhibitor or GPCR-binding prediction (Jacob and Vert, 2007).

REFERENCES

Antes,I. *et al.* (2006) DynaPred: a structure and sequence based method for the prediction of MHC class I binding peptide sequences and conformations. *Bioinformatics*, **22**, e16–e24.

Aronszajn,N. (1950) Theory of reproducing kernels. *Trans. Am. Math. Soc*, **68**, 337–404.

Bhasin,M. and Raghava,G.P.S. (2004) Prediction of CTL epitopes using QM, SVM and ANN techniques. *Vaccine*, **22**, 3195–3204.

Bhasin,M. *et al.* (2003) MHCBN: a comprehensive database of MHC binding and non-binding peptides. *Bioinformatics*, **19**, 665–666.

Bottou,L. *et al.* (eds.) (2007) *Large-scale kernel machines*. MIT Press.

Bui,H.-H. *et al.* (2005) Automated generation and evaluation of specific MHC binding predictive tools: Arb matrix applications. *Immunogenetics*, **57**, 304–314.

Bui,H.-H. *et al.* (2006) Structural prediction of peptides binding to MHC class I molecules. *Proteins*, **63**, 43–52.

Davies,M.N. and Flower,D.R. (2007) Harnessing bioinformatics to discover new vaccines. *Drug Discov. Today*, **12**, 389–395.

Dönnes,P. and Elofsson,A. (2002) Prediction of MHC class I binding peptides, using SVMHC. *BMC Bioinformatics*, **3**, 25.

Doytchinova,I.A. *et al.* (2004) Identifying human MHC supertypes using bioinformatic methods. *J. Immunol.*, **172**, 4314–4323.

Doytchinova,I.A. *et al.* (2005) Towards the chemometric dissection of peptide–hla-a*0201 binding affinity: comparison of local and global qsar models. *J. Comput. Aided Mol. Des.*, **19**, 203–212.

Evgeniou,T. *et al.* (2005) Learning multiple tasks with kernel methods. *J. Mach. Learn. Res*, **6**, 615–637.

Heckerman,D. *et al.* (2006) Leveraging information across HLA alleles/supertypes improves HLA-specific epitope prediction.

Hertz,T. and Yanover,C. (2006) PepDist: a new framework for protein-peptide binding prediction based on learning peptide distance functions. *BMC Bioinformatics*, **7** (**Suppl. 1**), S3.

Jacob,L. and Vert,J.-P. (2007) Kernel methods for *in silico* chemogenomics. Technical Report 0709.3931v1, arXiv.

Jojic,N. *et al.* (2006) Learning MHC I–peptide binding. *Bioinformatics*, **22**, e227–e235.

Korber,B. *et al.* (2006) Immunoinformatics comes of age. *PLoS Comput. Biol.*, **2**, e71.

Mamitsuka,H. (1998) Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models. *Proteins*, **33**, 460–474.

McMichael,A. and Hanke,T. (2002) The quest for an AIDS vaccine: is the CD8+ T-cell approach feasible? *Nat. Rev. Immunol.*, **2**, 283–291.

Nielsen,M. *et al*. (2003) Reliable prediction of T-cell epitopes using neural networks with novel sequence representations. *Protein Sci*., **12**, 1007–1017.

Peters,B. and Sette,A. (2005) Generating quantitative models describing the sequence specificity of biological processes with the stabilized matrix method. *BMC Bioinformatics*, **6**, 132.

Peters,B. *et al*. (2006) A community resource benchmarking predictions of peptide binding to MHC-I molecules. *PLoS Comput. Biol*., **2**, e65.

Rammensee,H. *et al*. (1999) Syfpeithi: database for MHC ligands and peptide motifs. *Immunogenetics*, **50**, 213–219.

Rammensee,H.G. *et al*. (1995) MHC ligands and peptide motifs: first listing. *Immunogenetics*, **41**, 178–228.

Robinson,J. *et al*. (2000) IMGT/HLA database–a sequence database for the human major histocompatibility complex. *Tissue Antigens*, **55**, 280–287.

Rosenfeld,R. *et al*. (1995) Flexible docking of peptides to class I major-histocompatibility-complex receptors. *Genet. Anal*., **12**, 1–21.

Salomon,J. and Flower,D.R. (2006) Predicting Class II MHC-Peptide binding: a kernel based approach using similarity scores. *BMC Bioinformatics*, **7**, 501.

Schölkopf,B. and Smola,A.J. (2002) *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA.

Schölkopf,B. *et al*. (2004) *Kernel Methods in Computational Biology*. MIT Press.

Sette,A. and Sidney,J. (1998) HLA supertypes and supermotifs: a functional perspective on HLA polymorphism. *Curr. Opin. Immunol*., **10**, 478–482.

Sette,A. *et al*. (1994) The relationship between class I binding affinity and immunogenicity of potential cytotoxic T cell epitopes. *J. Immunol*., **153**, 5586–5592.

Sette,A. *et al*. (2001) HLA expression in cancer: implications for T cell-based immunotherapy. *Immunogenetics*, **53**, 255–263.

Shawe-Taylor,J. and Cristianini,N. (2004) *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Tung,C.-W. and Ho,S.-Y. (2007) Popi: predicting immunogenicity of MHC class I binding peptides by mining informative physicochemical properties. *Bioinformatics*, **23**, 942–949.

Vapnik,V.N. (1998) *Statistical Learning Theory* Wiley, New-York.

Wang,R.F. (1999) Human tumor antigens: implications for cancer vaccine development. *J. Mol. Med*, **77**, 640–655.

Yewdell,J.W. and Bennink,J.R. (1999) Immunodominance in major histocompatibility complex class I-restricted T lymphocyte responses. *Annu. Rev. Immunol*., **17**, 51–88.

Zhang,G.L. *et al*. (2005) MULTIPRED: a computational system for prediction of promiscuous HLA binding peptides. *Nucleic Acids Res*., **33** (Web Server issue), W172–W179.

Zhao,Y. *et al*. (2003) Application of support vector machines for T-cell epitopes prediction. *Bioinformatics*, **19**, 1978–1984.

Zhu,S. *et al*. (2006) Improving MHC binding peptide prediction by incorporating binding data of auxiliary MHC molecules. *Bioinformatics*, **22**, 1648–1655.