An Improved Fully Connected Hidden Markov Model for Rational Vaccine Design

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Departmentof Computer Science
University of Saskatchewan
Saskatoon

By
Chenhong Zhang

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
University of Saskatchewan
Saskatoon, Saskatchewan S7N 5A9

Abstract

Large-scale, *in vitro* vaccine screening is an expensive and slow process, while rational vaccine design is faster and cheaper. As opposed to the emperical ways to design vaccines in biology laboratories, rational vaccine design models the structure of vaccines with computational approaches. Building an effective predictive computer model requires extensive knowledge of the process or phenomenon being modelled. Given current knowledge about the steps involved in immune system responses, computer models are currently focused on one or two of the most important and best known steps; for example: presentation of antigens by major histo-compatibility complex (MHC) molecules. In this step, the MHC molecule selectively binds to some peptides derived from antigens and then presents them to the T-cell. One current focus in rational vaccine design is prediction of peptides that can be bound by MHC.

Theoretically, predicting which peptides bind to a particular MHC molecule involves discovering patterns in known MHC-binding peptides and then searching for peptides which conform to these patterns in some new antigenic protein sequences. According to some previous work, Hidden Markov models (HMMs), a machine learning technique, is one of the most effective approaches for this task. Unfortunately, for computer models like HMMs, the number of the parameters to be determined is larger than the number which can be estimated from available training data. Thus, heuristic approaches have to be developed to determine the parameters. In this research, two heuristic approaches are proposed. The first initializes the HMM transition and emission probability matrices by assigning biological meanings to the states. The second approach tailors the structure of a fully connected HMM (fcHMM) to increase specificity. The effectiveness of these two approaches is tested on two human leukocyte antigens(HLA) alleles, HLA-A*0201 and HLA-B*3501. The results indicate that these approaches can improve predictive accuracy. Further, the HMM implementation incorporating the above heuristics can outperform a popular profile HMM (pHMM) program, HMMER, in terms of predictive accuracy.

Acknowledgements

I would like to thank my supervisors, Dr. Tony Kusalik, for his guidance and patience during this work and Dr. Lorne Babiuk for the great financial support. I would also like to thank Dr. Mik Bickis, who contributed a lot time in discussing problems and providing his expertise. My gratitude extends to Dr. Mark Daley, Dr. Shawn Babiuk, Dr. Michael Horsh and Dr. Eric Neufeld who gave me a lot of good advice and generous help on running this project.

I am grateful to have met all people in the bioinformatics group, especially Matthew Bainbridge and Fangxiang Wu. I would like to thank Fangxiang for spending a lot time discussing problems and I would like to thank Matthew for his great editting and entertaining efforts and friendship.

My family and my husband, Hao Chen, gave me consistent support through the years I spent in Saskatoon. This thesis would not be possible without them.

to my parents, brother and husband

Contents

List of Tables

List of Figures

List of Abbreviations

3-D - 3 -dimensional
ANN - Artificial Neural Network
ANNPred - ANN Prediction
APC - Antigen Presenting Cell
AROC - area under the ROC curve
BoLA - Bovine Lymphocyte Antigen
BW - Baum-Welch
ComPred - Combined (an ANN and a matrix-based method) Prediction
CPU - Central Processing Unit
DNA - DeoxyriboNucleic Acid
e.g. - Latin phrase *exempli grati*, means "for example"
ER - Endoplasmic Reticulum
etc - *et cetera*, means "and others"
fcHMM - fully connected Hidden Markov Model
HLA - Human Leukocyte Antigens
hr - hour
MHC - Major Histo-compatibility Complex
HMM - Hidden Markov Model
HPV - Human Papilloma Virus
i.e. - Latin phrase *id est*, means "that is"
ITERALIGN - a symmetry-ITERated ALIGNment of protein sequences
LTR - Left Transcription Region
lab - laboratory
MHCPEP - a database of peptides known to bind to MHC molecules
MEME - Multiple EM (Expectation Maximum) for Motif Elicitation
MSA - Multiple Sequence Alignment
PIII - Pentium III
PIV - Pentium IV
pcHMM - partially connected Hidden Markov Model
PDB - Protein Data Bank
pHMM - profile Hidden Markov Model
PIMA - Pattern-Induced Multiple sequence Alignment
RAM - Random Access Memory
RPEDEP - PREDict EPitope
ROC - Receiver Operating Characteristic
RTR - Right Transcription Region
SAM - Sequence Alignment and Modeling system
sec - second
TCR - T-Cell Receptor
VIDO - Vaccine and Infectious Disease Organization

Chapter 1

Introduction

Technological breakthroughs in molecular biology in the past decade have led to the generation of immense amounts of data and research at the genomic or proteomic level. This has given rise to an increase in the popularity of computational research methods, which are often faster than biological experiments. Rational vaccine design, for example, uses computers for large scale screening of potential vaccines based solely on genomic and proteomic information.

Rational vaccine design has only been developed in the past decade, but in that time it has become a routine research method in some immunology laboratories [2] and a key goal of immunoinformatics [3]. As in many other computer-aided research areas, computational methods for vaccine design sometimes generate results that are disproved by biological experiments [2]. Thus predictions of rational vaccine design should always be confirmed by biological experiments before any conclusions are drawn. Fortunately, the predictive accuracies of these prediction methods increase as more data and knowledge about the mechanisms of immune response become available. Therefore, rational vaccine design can be expected to become a common approach in immunology laboratories in the future. Another limitation of rational vaccine design is that it models only short biological sequences, which often does not work so effectively as vaccine prepared from an integrated organism.

An immune response is the result of a series of biochemical reactions whose end goal is to protect the body from foreign material. These reactions act, in part, to produce and select particular epitopes from antigenic material. An epitope is a peptide that can be recognized by a T-cell and elicit an immune response against the foreign body [4, 5, 6]. Immunological experiments show that only peptides which bind to MHC with high affinity are recognized as T-cell epitopes and thus can elicit an immune response [7]. In other words, T-cell epitopes are the subset of peptides that bind to MHC and thus are potential vaccines[8]. Experiments also show that usually only 5% of the peptides from a protein are able to bind to a MHC. This means that MHC-binding peptide prediction can significantly reduce the number of peptides to be tested for vaccine development. In this sense, MHC-binding peptide prediction makes it possible to do large scale screening for vaccines, and thus it is a popular topic in computational vaccine design [9].

A broad spectrum of MHC-binding peptide prediction methods is currently available, e.g. motif-

based methods, matrix-based methods, molecular modeling, and machine learning techniques [9]. The performance of these computational methods relies heavily on quality and quantity of known MHC-binding peptides. Generally, as the number of known MHC-binding peptides (training data) increases, the predictive performance of all the methods increases. However, as the number of training data increases further, the performance of sophisticated methods, like the machine learning techniques, will eventually outperform the simple methods, e.g. matrix-based and motif-based methods [10]. Therefore sophisticated methods have enormous potential for future vaccine development.

MHC-binding prediction is primarily complicated by the amount of training data available. Since MHC molecules are highly polymorphic, i.e. many different forms of the molecules exist, a separate computer model must be developed for each MHC. Further the number of peptides known to bind to a particular MHC molecule varies from none to hundreds. For species other than humans and mice, e.g. cattle, the number of known binding peptides for an MHC is never more than 30. Consequently, complicated computer models can only be studied with data from humans and mice. Theoretically, the computer models should be adaptable to data from other species.

A machine learning model can be so complicated that even hundreds of examples of training data are not enough to determine all the parameters of that model. In such a case the only solution is to use heuristic approaches. This thesis is designed to investigate two approaches to improve the performance of a Hidden Markov model (HMM). One approach aims to improve the initialization of emission and transition matrices using knowledge of biological sequences. The other approach is to reduce the model's connectivities so that the model parameters can be better estimated with the small amount of the training data available.

This thesis is composed of six chapters and an appendix. Chapter one is the "Introduction". Chapter two, "Background", covers some immunological knowledge and various computational methods used for rational vaccine design. Chapter three, "Research goal", describes the goal to achieve and the problems to solve. Chapter four, "Data and Methodology ", proposes solutions to the problems raised and describes data used to do model training and testing. Chapter five "Results", shows the performance (predictive accuracy and computational time) of the proposed solutions. And last chapter, chapter six "Discussion and future work" discusses how effective the proposed solutions worked, based on the results shown in Chapter five and then proposes some future work. The appendix shows a single-letter amino acid code, the training data (peptides which bind to allele HLA-A*0201 and peptides that do not bind to HLA-A*0201, the same for HLA-B*3501) used to train models and the emission and transition matrices developed from the various heuristic approaches.

This thesis assumes the reader has some basic knowledge of amino acids, proteins, immune response, etc. and knowledge of some bioinformatics concepts, e.g. substitution matrix, position

specific scoring matrix and multiple sequence alignment. HMM is the method being investigated in this work but readers are not expected to already know how an HMM works, as it is explained in some details in chapter 2.

This work uses specificity when evaluating the predictive accuracy of computational programs. It is defined as the ratio between true negatives and the sum of true negatives and false positives [11]. This is what most of the researchers use, but note that some researchers defined it as true positives over the sum of true positives and false positives [12].

Chapter 2

Background

2.1 Immune response

Animals have various defense mechanisms against pathogens. For example, for humans, the first line of defense is the skin and mucosa. When an invader breaches these outer defenses, it is the immune system which is ultimately responsible for disposing of the pathogen. To fulfill its role, the immune system must be sophisticated, with many components, processes, and interactions.

An immune response is the reaction of the body to the presence of antigens (foreign molecules). An immune response is a sequence of reactions which involve many biological components. These reactions are continuous, but for convenience they can be divided into four major stages: antigen detection, antigenic protein processing, antigenic peptide presentation to T-cells, and protection mechanism activation.

The immune response begins with an Antigen Presenting Cell (APC) engulfing an antigen and cleaving it into small pieces (peptides) using proteasomes. This is done in the cytosol of the APC. Of the resulting peptides, those which are 3-40 amino acids long are translocated by transporters into the endoplasmic reticulum (ER) where they interact with major histo-compatibility complexes (MHCs). MHCs are surface glycoproteins responsible for presenting peptides to T-cells. There are various kinds of MHC, differing in the set of peptides they prefer to bind. Typically, one peptide is bound (recognized) by only a subset of the MHCs, or none at all. An MHC-peptide complex may then be recognized by a T-cell if the peptide matches the T-cell receptor [4]. These complexes are also called T-cell epitopes. T-cell recognition is required for subsequent cell lyses/antibody induction to occur. Thus, to elicit an immune response a peptide has to go through several major filtrations: proteasome slicing, intra-cellular transportation, and recognition by MHC and then by a T-cell receptor.

A peptide which successfully elicits an immune response can be artificially synthesized and used to immunize an animal. Such a peptide is a type of vaccine. Currently, research in rational vaccine design focuses on investigation of MHC recognition of peptides. This is because MHCs are highly selective and the properties of MHC-peptide complexes are easier to study than the other two filtering steps in the formation of an immune response [8, 10].

	HLA I	# of alleles	HLA II	# of alleles
classical	HLA-A	291	HLA-DP	126
	HLA-B	555	HLA-DQ	81
	HLA-C	140	HLA-DR	435

Table 2.1: Polymorphism of Human MHC

2.2 Major histo-compatibility complex (MHC)

2.2.1 Polymorphism and functional diversity of MHCs

There are three classes of MHCs: MHC I, MHC II, and MHC III. MHC I and MHC II are more important in the immune response and thus have been investigated more than MHC III [13]. In the following context, "MHCs" will refer to MHC I and MHC II only. There are three[1] gene loci for MHC I and three for MHC II. A diploid organism will therefore have a maximum of 6 different MHC I alleles and 6 different MHC II alleles. However, in a population of animals the number of possible alleles for each loci is in the scope of hundreds. The following example illustrates the polymorphic nature of MHCs in humans. Human MHC molecules are referred to as "HLA" (human leukocyte antigens). Table 2.1 shows the number of genes encoding HLA class I (HLA I) and HLA class II (HLA II) alleles. Data in this table was collected from the IMGT/HLA sequence database [14].

The term "allele" generally refers to the variations of a single gene locus; for instance, HLA-A1 and HLA-A2 are HLA-A alleles. However, for convenience and from this point onward, "alleles" also refers to variations across gene loci within one type of MHC; e.g. HLA-A1 and HLA-B7 are both HLA I alleles, but HLA-A1 and HLA-DR3 are not because one is a HLA I gene and the other is a HLA II gene.

MHCs have been investigated in other animals besides humans and mice, including cattle, cats, dogs, wolves, apes and monkeys. Because cattle are important as a food resource for humans, research work on bovine MHCs is worthwhile. The special term for MHC in cattle (*Bos taurus*) is "BoLA", which stands for bovine lymphocyte antigen. To date, 28 BoLA class I (BoLA I) alleles, 103 BoLA class II (BoLA II) DR alleles, and 97 BoLA class II DQ alleles have been identified [15]. Compared to HLA, BoLA and its interaction with antigenic peptides are not as clearly understood.

Each MHC I molecule is distinct in its structure and binding affinity to different antigenic peptides. Nevertheless, some peptides bind to multiple MHC I molecules and are called "promiscuous

[1] There are actually three "classical" and three "non-classical" MHC I loci. For MHC II, there are three "classical" and two "non-classical" loci. In this paper, only the classical loci are considered.

peptides". These properties of MHC I also hold for MHC II. The polymorphism and functional diversity of MHCs make some individuals less vulnerable than others to a certain pathogenic antigen. As a consequence, members of a population can mount better or worse immune responses to some pathogen but it also makes MHCs hard to study.

2.2.2 MHC-peptide binding complex

Despite various functional differences between MHC I and MHC II, they bind antigenic peptides in a similar fashion to form a complex in the endoplasmic reticulum. A peptide binding to an MHC molecule does so in a groove in the molecule. In Figure 2.1, the grooves are the spaces where the peptides (colored red in images a and b) are located. Whether the groove is open or closed at the ends determines the length of the binding peptides. The groove in MHC I has closed ends, so peptides binding to MHC I are usually fixed in length, about 7-12 amino acids (or "mers") long [16] (Figure 2.1a, 2.1c). Alternately, the groove in MHC II has open ends, so this molecule binds peptides of variable lengths, 10-30 mers [17] (Figure 2.1b, 2.1d). Most of the functional allelic variations in MHC are located in the domains that form the peptide-binding groove. The PDB protein database [18] has crystal structures for a modest number of HLA (or HLA complexes with a binding peptide).

The binding affinity between a peptide and an MHC can be evaluated in various ways, including half-time disassociation rate and a pair-wise potential table (a way of calculating the binding energy).

MHC pocket structures A groove is usually composed of 6 "pockets", labelled A to F (Figure 2.2). Each pocket is formed by several amino acids in close physical proximity in the three-dimensional structure of the MHC. Each pocket is responsible for interacting with one or several amino acids on the binding peptide. The interaction is affected not only by many physical and chemical effects in the environment – for instance, ion concentration, temperature and pH – but also affected by adjacent amino acids or chemical groups on the peptides or in MHC groove.

A quantitative representation of the interaction of all 20 amino acid residues with a particular pocket forms a "pocket profile". MHC alleles are believed to share pockets and there exist databases of pocket profiles [19]. This makes it possible to characterize the binding of new MHC alleles.

Position and neighboring effects of the peptide Although all positions on a binding peptide are involved in the MHC-peptide interaction, there is also a positional effect to consider. It is well established that a few – about 2 to 4 – positions in a peptide are critical in mediating its binding to a specific MHC. These positions are termed "anchor" positions and the amino acids that occur more frequently than others at these anchor positions are called "anchor residues". Anchor residues are

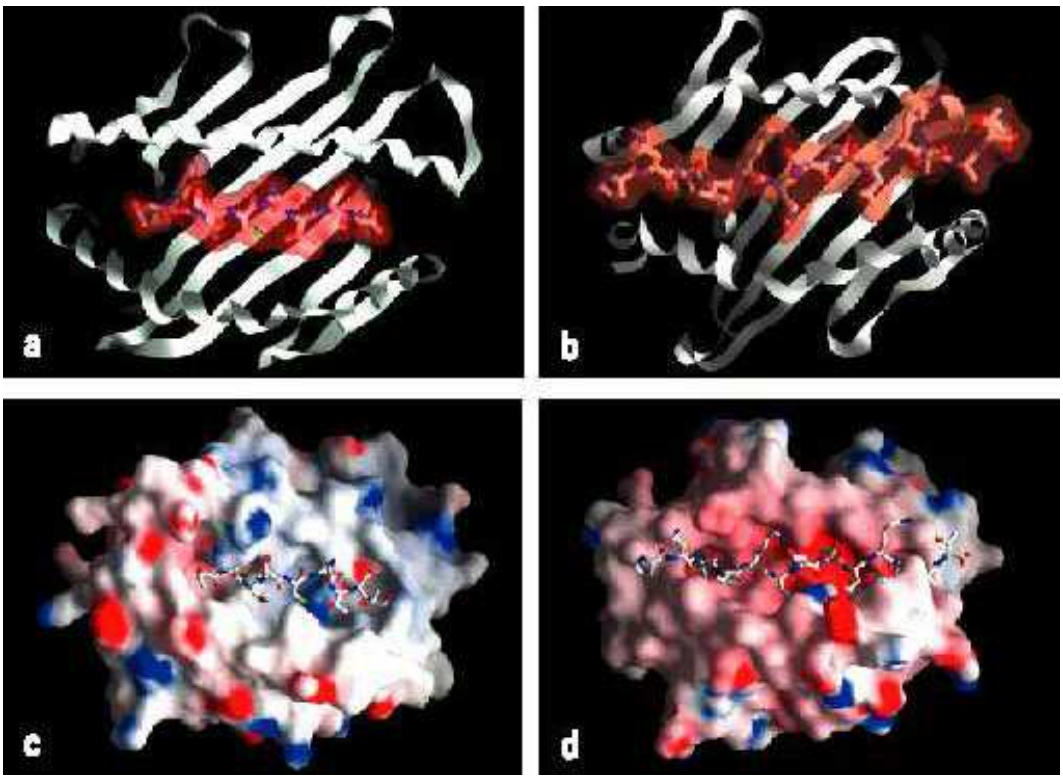

Figure 2.1: Structures of an MHC I-peptide complex (a and c) and an MHC II-peptide complex (b and d)

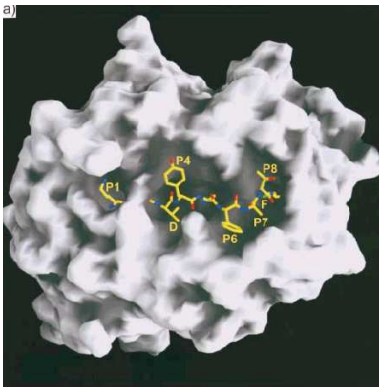

Figure 2.2: Pockets of an MHC and amino acids (P1-P9) that protrude into the pockets

	P1	P2	P3	P4	P5	P6	P7	P8	P9
Peptide 1	H	**K**	G	A	G	K	R	Y	**V**
Peptide 2	N	**K**	Y	M	C	V	S	T	**V**
Peptide 3	M	**K**	C	W	S	G	A	F	**V**

Table 2.2: Binding motifs in three HLA-A*0201-binding peptides

responsible for fixing the peptide to the groove. As mentioned earlier, the MHC-peptide complex waits to be recognized by a receptor on a T-cell. The rest of the positions on the peptide are recognized and bound by the T-cell receptor (TCR).

The anchor positions and anchor residues form a pattern called a "binding motif". Anchor positions are further divided into primary anchor positions and secondary anchor positions, according to their contributions to the binding affinity. Different MHCs have different patterns of primary and secondary anchor positions. Information about anchor positions and anchor residues for various MHC alleles are derived from the alignment of multiple peptides known to bind to each allele. For example, the peptides binding to the human MHC molecule HLA-A*0201 are often 9 amino acids long and frequently have two anchor residues, a Lysine (K) at position 2 and a Valine (V) in position 9 (see Table 2.2) [20]. P1 to P9 the table refer to the 9 positions of the binding peptide.

2.2.3 MHC-binding peptide databases

Sequences of peptides that bind to specific MHCs have been collected in MHC-binding peptide databases. MHCPEP [21] is by far the largest and most widely referenced database. As of today, it contains 13424 entries, among which are 10,012 entries for human, 3359 entries for mouse and 53 for four other mammals. Each entry contains information about source protein (if known), an estimated binding affinity, anchor residues (if identified), and references. The large number of

database entries in MHCPEP presents the possibility of automated data processing. However, this possibility is limited to research on the human and mouse.

To use data from human and mouse in the context of other organisms, researchers have been searching for evidence of commonalities between humans and other mammals. Evolutionary study of MHC molecules indicates that many orthologous loci of MHC II are shared by many mammals. Takahashi [22] provides a phylogenetic map that shows orthologous relationships among MHC II genes in several mammals. With this knowledge, data from human MHC can be adapted to other mammals. However, non-mammalian class II genes are not orthologous to mammalian class II genes [22], which means it is not valid to adapt data from human to these species. Also, MHC I loci undergo evolution faster than MHC II loci and thus it is difficult to establish orthologous relationships among MHC I in different mammals [23, 24, 25, 22]. Thus, borrowing data from human MHC may be helpful only in limited situations.

2.3 Vaccine and vaccine design strategies

Generally speaking, a vaccine is an antigen that primes the immune system against some pathogen without causing severe symptoms. Thus, vaccines can take various forms: an organism (bacteria or virus), a protein (or peptide), or nucleic acid sequence. In the context of this work, a vaccine is an 8-22mer peptide. Such a small piece of protein is not capable of acting as a pathogen, but is still sufficient to elicit an immune response.

The traditional vaccine design approach is to derive all the possible vaccines from a protein sequence and test each of them for an immune response experimentally. Unfortunately, this brute-force approach is also very expensive and time-consuming.

Luckily, the brute-force approach can be simulated easily by computers. Once a computational model "learns" how to identify a vaccine, a job that could take months or years of manual labor, could be finished in days or hours by computer. However, in order to do this a computer has to learn to identity the characteristics of peptides that have already been confirmed to elicit an immune response. This means that the performance of some of the computational methods is limited by the availability of such peptides. Vaccine identification is a kind of pattern recognition and the computational methods used for pattern recognition are applicable here.

2.4 Pattern recognition in sequential data analysis

According to Kiran [26], a pattern in biological sequence analysis is any type of sub-sequence that fits a certain regular expression, which is a string that describes or matches a set of strings, according to certain syntax rules. For example, M-[PST]-X(1,3) is a sequence pattern matching any sequence containing a sub-string starting with M, followed by either a P or a S or a T, followed by one,

two or three arbitrary (X means "any") symbols. This example is a deterministic pattern. If the occurrence of a character is associated with a probability, then the pattern turns into a probabilistic pattern. Pattern can be used to characterize protein families. Motifs are short patterns that occur frequently within a given set of protein sequences. Motifs usually have some biochemical functions which represent information conserved through evolution. Thus a motif is a pattern associated with some biological function, while a pattern is more of a statistical presence than a biological one. Also, all motifs are patterns but not all patterns are motifs.

The detection of underlying patterns in a collection of peptides is called pattern recognition. Pattern recognition is a basic task in many applications, e.g. internet searching for document classification, printed circuit board inspection in industrial automation, personal identification in biometric recognition, and biological sequence analysis. No matter what application uses pattern recognition, there needs to be a substantial amount of known data (information for which the presence or absence of patterns is known) available. When such data is available, a computer is a good tool for performing the pattern recognition.

Jain [27] classified the pattern recognition techniques used today into four major categories. These categories are: template matching approach, statistical approach, syntactic approach, and neural networks. Techniques from each of these categories have been applied to biological sequence analysis. To explore them, it is necessary to first describe the languages of biological sequences.

2.4.1 Language of biological sequences

Like human language, biology employs its own sets of language rules. Proteins and nucleic acids (made from nucleotides) are the two most important components of an organism. Nucleic acids are carriers of genetic information. Proteins are not only the building blocks of the organism but also important in catalyzing metabolic reactions. Protein and nucleic acid sequences follow certain rules like the grammars of human language which is the system of rules by which words are formed and put together to make sentences. For example, a gene is usually composed of a promoter, a transcription-start site, an LTR (left transcription region), a translation start site, several exons intermixed with several introns, a RTR (right transcription region), and a transcription-stopping signal. With such a structure a gene is laid out like a sentence, which has a subject, a verb and some objects. The structures of genes exhibit patterns.

2.5 Pattern recognition methods for MHC-binding peptide prediction

MHC-binding peptide prediction methods are usually categorized into five groups. Although methods of some groups are theoretically more sophisticated than those of other groups, no single group

of methods dominates the area. In the following subsections, details about methods in each category are discussed, followed by a comparison of these methods.

2.5.1 Motif-based method

In the late 1980s, x-ray crystallography allowed researchers to visualize the structure of MHC-peptide complexes. They found that MHCs had clefts on their surfaces that could hold peptides. The earliest MHC-binding peptide prediction methods came independently from the work of Rammensee and of Sette [2]. First, both groups compiled a database of sequences of natural epitopes (peptides recognized by TCRs) by washing the peptides off of different MHC class I and class II proteins. Using the sequences of the peptides, their models extracted (learned) the binding motifs. Thus the computer models Rammensee and Sette developed belong to the motif-based class of methods.

As mentioned earlier, there are 2 - 4 anchor positions in a peptide binding to MHC. The pattern of amino acids at these positions is called a "motif". Since peptides binding to MHC I are uniform in length, in some cases motif identification is possible even "by eye". Motif identification for MHC II-binding peptides is more difficult because their lengths vary. In this case, motif identification software is needed. Many general purpose motif-finding packages are available – e.g. BLOCK MAKER, ITERALIGN, MATCH-BOX, PIMA, MEME and SAM [28] – and can be used for this application.

More recent motif identification approaches use the concept of a profile to evaluate the binding affinity of a peptide. A profile records a value for each amino acid at each position of the peptide. A preferred amino acid at an anchor position receives a higher score than a preferred amino acid at other positions, and amino acids regarded as unfavorable are assigned negative scores. The binding affinity of a peptide is calculated as the sum of scores of amino acids at each position of the peptide. A characteristic of this method is that the pattern being sought is very specific, i.e. the motif is either present or not. Because of this, this method can work with a small set of training data, as few as ten peptides. SYFPEITHI [29] is an example of a binding motif-based application that uses profiles and is still in common use today.

However, very few applications besides SYFPEITHI use this technique. This is likely because motif-based methods do not perform well when true binding peptides do not conform to the pattern or motif structure. This shortcoming can be addressed by many other prediction methods, as described in the next subsection.

2.5.2 Matrix-based methods

Matrix-based methods work by assigning a probability score to each of the 20 possible amino acids at each position of the peptide. This score is based on either half-time disassociation rate data [10]

or an MSA (multiple sequence alignment) of known binders.

Matrix-based methods evaluate the positional effect of amino acids, i.e. the same amino acid contributes differently to various positions in the peptide. However, they do not capture the neighborhood effect of amino acids at adjacent positions. Instead, it assumes that each amino acid contributes independently to the whole peptide's binding affinity. In the biological world, neighborhood effects usually exist and may be important in some cases.

Different matrices result from using different underlying algorithms to calculate the score for each amino acid at each position of a peptide. For example, YK0201 matrix developed by Yu [10], was generated using logarithmic equations based on the frequency of amino acids at specific positions of the peptides in the training set using the following two formulae [10]:

$$S_{AK} = \log_2(\frac{F_{AKb}-F_{AKn}}{F_{AKb}+F_{AKn}} + 1) \qquad if F_{AKb} > F_{AKn}$$

$$S_{AK} = -\log_2(\frac{F_{AKn}-F_{AKb}}{F_{AKb}+F_{AKn}} + 1) \qquad if F_{AKb} < F_{AKn}$$

where S_{AK} is the matrix coefficient for amino acid A at position K of the peptide; F_{AKb} refers to the proportion of amino acid A at the observed position K within binders, and F_{AKn} refers to the proportion of amino acid A at the observed position K within non-binders. In a similar fashion to the motif-based method, the binding affinity of a peptide is calculated as the sum of position scores from the YK0201 matrix. That is, given a peptide P, let $P[n]$ be the nth amino acid in P and $|P|$ be the length of P, then the affinity score of P, $S(P) = \sum_{k=1}^{|P|} S_{P[k]k}$, where $S_{P[k]k}$ is the matrix score for amino acid $P[k]$ at position k.

There are two differences between this method and methods based on binding motifs. The matrix-based techniques do not explicitly identify anchor positions nor do they require specific amino acid at certain anchor positions. Thus matrix-based methods predict some binding peptides that do not conform to the rules of binding motifs. Ten or more peptides are usually necessary to form the matrix. Of course, the more data used, the more accurate the result can be. Interestingly, for matrix-based methods, after a certain amount of training data, any further increase in the amount of data has little effect on the accuracy of the method [10].

The matrix-based methods are more popular than other MHC-binding prediction methods. Several software tools employing this method are RANKPEP [30], BIMAS [31], EpiMatrix [32], TEPITOPE [33] and Propred [34].

2.5.3 Hidden Markov models (HMMs)

An HMM is a statistical model developed in the 1970s for speech recognition. Since then, it has been used for many other problems with a simple grammar. Some examples of these problems

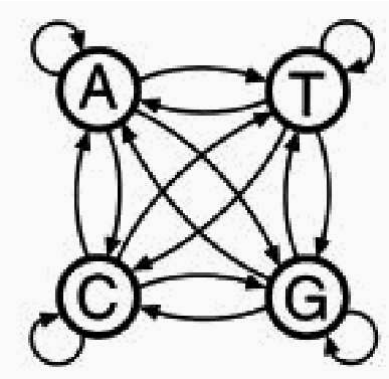

Figure 2.3: A graphical representation of a Markov chain for DNA sequences (cited from [1])

are: handwriting recognition, gene finding and protein modeling, gesture recognition, and facial recognition [27]. A good biological example of using an HMM is to form a " profile HMM" [1] which is developed from profiling a family of proteins. The resulting model can then be used to search against databases to discover other members of the family.

Profile and weight matrix methods are functionally similar to HMMs in terms of scoring alignments, but HMMs treat gaps in the pattern in a systematic way, while the other two methods do not. To explain hidden Markov models, it is necessary to discuss Markov chains first.

Markov chains

A Markov chain model generates sequences in which the probability of a symbol depends on the previous n symbols, where $n >= 1$. The value n is called the order of the model. A Markov chain has some basic elements, are "states", "symbols" and transitions between states. Each state transition is associated with a probability. A state can emit a symbol with some probability and this is called "symbol emission rate". All transitions going out of a state adds up to 1. The probabilities of emitting all symbols by a particular state also add up to 1. The number of transitions, the number of states, the transition rates and the emission rates are parameters of a Markov chain model.

A Markov chain can be represented as a directed graph, where the nodes correspond to states in the model and edges correspond to state transitions. On arrival at a destination state, one symbol is generated (or consumed). Figure 2.3 is a graphical representation of a Markov chain for DNA sequences. Each circle is a state and the letters "A", "C", "G", and "T" are the symbols emitted (or generated) by each state, respectively. These letters represent the four nucleotides that make up DNA. For some sequence x, the transition probability from state s to state t is a_{st} where $a_{st} = P(x_i = t|x_{i-1} = s)$. a_{st} is estimated by the following equation in which c_{st} is the count of transitions from s to t in the training data.

$$a_{st} = \frac{c_{st}}{\sum_{t}^{'} c_{st'}} \tag{1}$$

In a first order Markov chain, the probability of generating a sequence x of length L is,

$$\begin{aligned} P(x) &= P(x_L, x_{l-1}, ...x_1) \\ &= P(x_L|x_{L-1})P(x_{L-1}|x_{L-2})...P(x_2|x_1)P(x_1) \end{aligned} \tag{2}$$

With equation 1, the formula for calculating $P(x)$ (equation 2) becomes

$$P(x) = P(x_1) * \prod a_{x_{i-1}x_i} \tag{3}$$

The order of the model reflects the amount of interdependency between the adjacent positions of a sequence. Hence, in second order Markov chains, the probability of generating a state is determined by its previous two states.

In a Markov chain, a particular state of it can generate a number of symbols, each with certain probability. However, a particular symbol can only be generated by one state. The following section introduces a more complicated Markov model which allows a symbol to be generated by multiple states.

A general HMM for sequence analysis

Some biological sequence analysis problems involve identifying a characteristic subsequence within a long sequence, e.g. finding CpG islands [1]. A CpG island is a short piece of DNA in which the frequency of the CG sequence is higher than in other regions. Most parts of a sequence are non-islands. If the length of the island is known, a Markov chain can be used. However, the length is usually not known. To effectively model such a situation, two states are necessary: one for being in "islands" and one for being in "non-islands". Each state has its own set of probabilities for generating each symbol (also called emission probabilities). Given a particular subsequence, it is unknown what path through the states led to this sequence. This is one type of problem which can be modeled by a hidden Markov model (HMM). Here "hidden" refers to the unknown identity of the state which generates the subsequence. The problem of finding CpG islands turns into a problem of finding the underlying state sequences that generate the character sequence, i.e. finding a path through the states of the HMM. An algorithm to do this is the Viterbi algorithm.

Viterbi algorithm The Viterbi algorithm is an algorithm to determine the most probable underlying state sequence that generates a given character sequence. Let π^* be the most probable path (state sequence) and x be the character sequence. Then we have

$$\pi^* = \underset{\pi}{argmax} P(x, \pi)$$

Let $v_t(i)$ be the probability of the most probable path ending in state i at time t, with observations up to x_t. If $v_t(i)$ is known for all the states before and including i, we can calculate the probability of observation $v_{t+1}(j)$ by $v_{t+1}(j) = e_j(x_{t+1}) max_i(v_t(i)a_{ij})$ where a_{ij} is the transition probability from state i to state j and $e_j(x_{t+1})$ is the symbol emission probability of state j to emit symbol x_{t+1}. Initially, $v_0(0) = 1, v_0(i) = 0$ for $i > 0$, where 0 is the starting state which does not emit any symbol, and the ending condition is $P(x, \pi^*) = max_i(v_L(i)a_{L0})$, where L is the length of the symbol sequence.

After this calculation runs through all observations in the sequence, the underlying state sequence can be found by backtracking. The state sequence given by the Viterbi algorithm is the best one in mathematical sense, but may not be the actual path that generates the observed symbol sequence in a biological system.

Baum-Welch (BW) algorithm Transition and emission probabilities are needed for the Viterbi algorithm. They are obtained from previous experience or a set of training sequences. Another algorithm to obtain HMM parameters from a set of training sequences is called the forward-backward algorithm or Baum-Welch (BW) algorithm [35]. Here is a brief description of the BW algorithm:

Let the set of training sequences be O, the s_{th} training sequence be O^s, and the t_{th} symbol of O^s be O^s_t. The length of O^s is l_s. Let the model be H. In model H, let the number of states be N, the number of symbols be M, the state transition probability from state i to state j be a_{ij} ($1 <= i <= N$, $1 <= j <= N$), and the symbol emission probability of symbol c at state i be $e_i(c)$ ($1 <= c <= M$). H has two extra states, a starting state which labels the start of a state sequence and an ending state which labels the end of a state sequence. These two states do not generate any symbols. The transition probability a and emission probability e must satisfy the following constraints:

$$\sum_{j=1}^{N} a_{ij} = 1, \sum_{c=1}^{M} e_j(c) = 1, \qquad \text{where } a_{ij} >= 0 \text{ , } e_j(c) >= 0$$

Let $f_t(j)$ be the probability of generating a partial sequence $O^s_1 ... \ O^s_t$ such that the state at time t is j, then we have

$f_0(j) = 1$, if j is the starting state, and

$f_0(j) = 0$, if j is not the starting state,

$f_t(j) = \sum_{i=1}^{N} a_{ij} e_j(O_t^s) f_{t-1}(i), \qquad (t = 1, ..., l_s),$

$f_{l_s+1}(j) = \sum_{i=1}^{N} a_{ij} f_{l_s}(i)$

Alternatively, if we start from the other end of the sequence and call $b_t(i)$ the probability of generating a partial sequence O_{t+1}^s ... $O_{l_s}^s$. Then we have

$b_{l_s+1}(i) = 1$, if i is the ending state, and

$b_{l_s+1}(i) = 0$, if j is not the ending state,

$b_t(i) = \sum_{j=1}^{N} a_{ij} e_j(O_{t+1}^s) b_{t+1}(j), \qquad (t = l_s - 1, ..., 0),$

$b_{l_s}(i) = \sum_{j=1}^{N} a_{ij} b_{l_s+1}(j)$

Thus, $\sum_{i=1}^{N} f_{l_s+1}(i) b_{l_s+1}(i)$ or $\sum_{i=1}^{N} f_0(i) b_0(i)$ is the probability that the given training sequence O^s is generated by H, i.e. $P(O^s|H)$. Let $\gamma_t(i)$ be the conditional probability of state i at time t, given the sequence. We have

$\gamma_t(i) = \frac{f_t(i) b_t(i)}{P(O^s|H)}$

Further, let $\xi_t(i,j)$ be the conditional probability of state i at time t and state j at time $t+1$. Then we have

$\xi_t(i,j) = \frac{f_t(i) a_{ij} b_j(O_{t+1}^s) b_{t+1}(j)}{P(O^s|H)}, \qquad (t = 0, ..., l_s - 1)$ and

$\xi_{l_s}(i,j) = \frac{f_{l_s}(i) a_{ij} b_{l_s+1}(j)}{P(O^s|H)}$

The goal of the BW algorithm is to maximize the likelihood, i.e. the probability of the observed sequence O^s when given the model H, i.e. $\prod_s P(O^s|H)$. This is done by reestimating the transition and emission probabilities iteratively until the difference of $\prod_s P(O^s|H)$ from two consecutive cycles is small enough (i.e. smaller than a threshold). The equations to perform the parameter reestimations are

$\hat{a}_{ij} = \frac{\sum_s \sum_{t=0}^{l_s} \xi_t(i,j)}{\sum_s \sum_{t=0}^{l_s} \gamma_t(i)}$

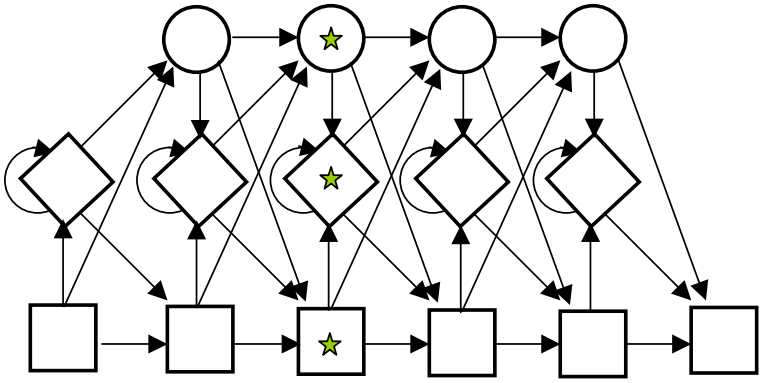

Figure 2.4: The structure of a pHMM

$$\hat{e}_i(c) = \frac{\sum_s \sum_{t=1, O_t^s = c}^{l_s} \gamma_t(i)}{\sum_s \sum_{t=1}^{l_s} \gamma_t(i)}$$

HMM in vaccine design

Vaccine design is a pattern recognition problem. Determining if a sequence is a potential vaccine is equivalent to calculating the probability that this sequence should belong to the set of (known) vaccines. An HMM is one approach that can give the answer. Previous attempts of using HMMs for vaccine design can be categorized into two groups according to the model's topology.

Krogh et al. [36] used a structure called an "alignment HMM" or a "profile HMM" (pHMM). pHMMs were originally developed for multiple sequence alignment (MSA), specifically for the task of finding motifs or functional units of a protein. Figure 2.4 shows the structure of a pHMM. The rectangles, diamonds and the circles represent three different states in the model. A rectangular box is a match state that models the distribution of letters in the corresponding column of an alignment. A diamond models insertions of symbols between two alignment positions. A circle models a deletion, which is a gap in an alignment. As evident from the figure, neighboring states are connected directly by directional lines. Each line represents a transition from one state to another according to the indicated direction. Each transition has a probability. The states in one column in the figure correspond to one position of a biological sequence. One such set is indicated by stars in Figure 2.4.

To perform a MHC-binding peptide prediction, training sequences have to be aligned into a MSA, from which the parameter set of a pHMM is obtained. Then the pHMM can align an unknown peptide to the MSA to give this peptide an alignment score which is proportional to the peptide's potential to bind to an MHC. More specifically, a pHMM can be trained from unaligned, known MHC-binding peptides (training data/sequences) or it can be built from prealigned MHC-

binding peptides using some MSA techniques like ClustalW [37] or SAM [38]. For the latter case, the state sequence of each peptide in the MSA can be extracted from the alignment. From these state sequences and the aligned peptide sequences, the parameter set of the pHMM can be determined by converting observed counts of state transitions and symbol emissions into probabilities. Alternatively, if a pHMM is trained from unaligned known MHC-binding peptides, the training process is analogous to doing an MSA of these peptides. The standard training algorithm is the BW algorithm. From the training, the parameter set of the model is obtained. With the parameter set, the Viterbi algorithm can be used to find the best state sequence for each training sequence, which is then aligned into a MSA. At this point, no matter if the prediction starts with prealigned sequences or unaligned sequences, an MSA of all training sequences and the parameter set of the pHMM are both available for further analysis. When a new peptide is considered, the Viterbi algorithm is used again to find the best state sequence for this peptide and then this new peptide is aligned to the MSA of all training sequences. From this alignment, a score can be calculated indicating how likely it is that the new peptide is of the same class as the training data. In turn this membership implies the likelihood of this peptide being able to bind to the MHC.

One problem associated with a pHMM is that the method cannot predict multiple patterns in the training sequences because the pHMM will combine overlapping patterns into a single one and thus only the most common pattern in the training data will be discovered. Another problem of pHMMs is that the MSA used/given by the model is only one alignment for a set of sequences. When the sequences are of very different lengths, as in the case of MHC II-binding peptides, there may be several MSAs that have close alignment scores. The one used/given by the model may not be the one that detects the patterns the best.

In response to this problem, Mamitsuka [39, 40, 41] proposed an alternative HMM topology for the MHC-binding peptide prediction problem. His model is a fully connected HMM (fcHMM). In his model, except for two special states, every state is connected to every other state. The two special states are the starting state and the ending state. Neither of these two special states emit any symbols, rather they represent the start of a binding-peptide and the end of the binding peptide. Also, a number of transitions do not exist in this model's topology: any state to the starting state, the starting state to the ending state, and the ending state to any other state. The structure of an fcHMM is the most general kind of topology and allows for every kind of state path through the model. Since the structure of an fcHMM is so general, the model suffers from being computationally intensive (because you have to compute many transition probabilities) and is highly sensitive to noise in training data. However, unlike pHMMs, fcHMMs can capture multiple patterns in a set of training sequences.

One advantage of HMMs over simpler prediction methods discussed in Section 2.5.1 and 2.5.2 are their ability to capture more complex, non-linear features of the peptide-MHC interaction.

Here is an example of a non-linear feature: assume the individual contribution to binding affinity by an amino acid V at position 2 is 5 and that for an amino acid P at position 9 is 10, then the total contribution of binding affinity made by V and P is not the sum of 5 and 10. In contrast, HMMs require more training data than simpler methods. This is because in order to capture the non-linear features, a larger number of parameters are necessary. Not surprisingly, a comparison between machine learning techniques (HMM and ANN, which will be discussed below) and simple prediction methods shows that the accuracy of machine learning techniques is much lower than simpler methods with a small amount of training data [10]. However, there is a point where the accuracy of machine learning techniques increases dramatically with small increases in the size of the training data [10]. This study also reported that HMMs needs about 100 training sequences to achieve a good performance.

There are very few HMM implementations online that are specifically targeted to MHC-binding prediction. NetMHC [42] and PREDICT [43] are two such packages. NetMHC has a graphical interface that takes user's input (a protein sequence) and outputs potential MHC-binding peptides in the protein sequence. NetMHC has pre-built models (pHMMs) for certain alleles, and does not take users' training data to build new models. PREDICT is only available in a demonstration version which, like NetMHC, does not allow the user to input new training data.

2.5.4 Artificial neural network (ANN)

An ANN [44] is a machine learning technique that is used for classification and pattern recognition. It is by nature a form of multiprocessor computational system which simulates the parallel architecture of animal brains. An ANN is orders of magnitude simpler than a network of biological neurons, but the principles they employ are similar. Figure 2.5 is a structural diagram of a simple neural network. An ANN is usually composed of an input layer, several hidden layers and an output layer of neural nodes (circles in Figure2.5). There can be multiple nodes in each layer, but the output layer usually has only one node. There are interconnections between layers and within layers, however the direction of interconnection between layers is from input layer to hidden layers and finally to output layer. Each interconnection is assigned a weight. The number of hidden layers, the density of interconnection and weights are parameters that can be used to adjust the performance of the network.

A widely-used type of ANN, a back-propagation neural network, is used to illustrate the process of training. A supervised learning process, in which the true value of the output is known for a certain input, is followed. The training starts with initializing the parameters of the network with random values. Each training datum is run through the network in a forward pass, generating an output. This output is compared with the true value for this datum so that an error in the output can be calculated. A backward pass is then performed, during which the error is propagated

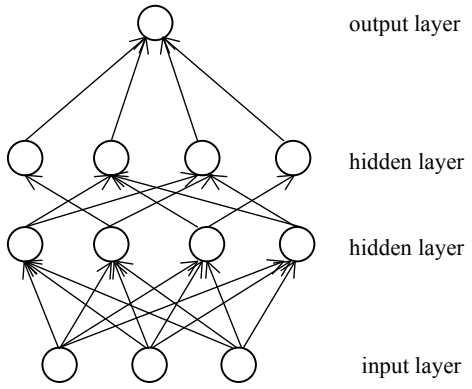

Figure 2.5: Structure Diagram of a Simple Artificial Neural Network

through the network. For each node, weights are changed by an amount proportional to the error. Once the weights have been adjusted, the network generates a new output for the training datum and the adjustment process is repeated. It continues until the error is less than a threshold, or some limit (e.g. on number of iterations) is exceeded. After an ANN is trained, an unknown input can be given to the network and the network will generate an output which tells how likely the input has the property being studied or which class (or a number of classes) the input belongs to.

For vaccine design, an ANN is trained with a large number of known MHC-binding and non-binding peptides. Each of the 20 amino acids is represented as a 20-bit binary string with a single 1 bit and nineteen 0 bits. The bit position which has value 1 is specific for each amino acid. Thus, each known 9-mer peptide is converted into a binary string of $9 * 20$ bits, with each of the bits fed into one input node of the ANN. By adjusting the number of hidden layers, the density of interconnections between nodes, and the weights of the interconnections, the output of the network can be made so that it correctly predicts whether a peptide is MHC-binding or not. Brusic developed an ANN for doing MHC-binding peptide prediction [45]. Raghava not only developed a program using an ANN, called ANNPred [46], but also developed a hybrid system that combined matrix-based and ANN technologies, called ComPred [47]. Both ANNPred and ComPred are free and available online. NetMHC [42], which is developed by researchers in Denmark, can use both ANN and HMMs to predict MHC-binding peptides. The previously mentioned program PREDICT (section 2.5.3) demonstrates MHC-binding peptide prediction using an ANN.

HMMs and ANNs are machine learning techniques, which are considered sophisticated methods that capture non-linear features of MHC-binding activity. Thus, neither HMMs nor ANNs work well with small input data sets. Unfortunately, researchers do not agree how large the data set must be before good results are obtained. Donnes and Elofsson reported this number to be 20 peptides [48]. Raghava and his colleagues used 40 peptides for ANNPred [46]. Yu reported that an HMM needs about 100 peptides in the training set and to have accurate results an ANN needs even more

[10]. Yu did not report if machine learning techniques have a upper threshold of accuracy, a point where more training data makes little difference. This might be due to the fact that no MHC allele has a training data set large enough to test this effect.

Another machine learning technique uses support vector machines (called SVMHC). It has been also been applied to MHC-binding peptide prediction [48]. A discussion of this technique is omitted, however, because it is less commonly used.

2.5.5 Molecular modeling

Molecular modeling tackles the same problem using a very different tactic. It models the interactions between MHC and the binding peptide using their structural and chemical properties [49, 50, 51]. In other words, this method relies heavily on the structural information of MHC-binding peptide complexes.

The entities directly involved in the binding activity are the pockets and the amino acids of the peptide that protrude into the pockets. The structural features of the entities are the size and shape, and the chemical features include hydrophobicity, charge, aromatic nature, etc. If all these properties are known for a MHC-peptide complex, the peptide in the MHC groove can be used as a template upon which unknown peptides are threaded. The binding energy of the unknown peptides can then be calculated based on a pair-wise potential table [52]. Unfortunately, the energy calculation is a computationally intensive process. At the end, if a peptide is energetically favored then it is predicted as a binder. Otherwise, the peptide is taken as a non-binder. A representative software tool for molecular modeling is PREDEP [53].

One advantage of molecular modeling is that a single known complex alone might be sufficient to create a prediction model for a MHC allele [48]. However, it is easy to see why this method is not popular, even though it models the binding event more closely to the real situation than any other methods. One reason is because available structural information about MHC-peptide complexes is too sparse, partially because it is technically very hard to get a crystal structure of the complex. Additionally, the energy calculation is not a simple one. In contrast, because of its characteristics, molecular modeling has great potential to become popular in the future as more structural information becomes available and a simpler way to do the energy calculation is found.

2.5.6 Predictive accuracy comparison between methods

When there are multiple methods to solve the same problem, comparison between methods is a good way to show how well each works. The performance of a program is affected by factors such as the training and testing data used. Thus the same program may be reported with different predictive accuracy across comparisons. Note that for molecular modeling, the data required is

Parameters:

TP (true positive), TN (true negative), FP (false positive)

FN (false negative), TPR (true positive rate), FPR (false positive rate)

Formulae:

sensitivity (TPR)= $\frac{TP}{TP+FN}$,	specificity = $\frac{TN}{TN+FP}$
positive predictive value = $\frac{TP}{TP+FP}$,	negative predictive value = $\frac{TN}{TN+FN}$
predictive accuracy = $\frac{TP+TN}{TP+FP+TN+FN}$,	false positive rate (FPR) = $\frac{FP}{FP+TN}$

Figure 2.6: Definition of Predictive Parameters

very different to those used by other methods. Moreover, the amount of molecular modeling data is too sparse. Hence, no comparison involving molecular modeling has been reported.

The main issue in comparing methods is the predictive accuracy. The four basic measuring parameters to analyze a prediction are true positives, true negatives, false positives and false negatives, as listed in Figure 2.6. Five predictive parameters (also in Figure 2.6) can be derived from these basic measuring parameters. These definitions are widely accepted; however some researchers defined specificity as true positives over the sum of true positives and false positives [12].

Prediction results are usually in the form of numerical values which need to be translated into binary values indicating binding identity. The translation is done by choosing a threshold which separate the values into two grouping, binders and non-binders. Hence, when the threshold is chosen differently, the numbers of the binders and non-binders vary. An objective way to present the predictive performance is to derive an AROC curve, which gives an overall view of how the performance changes with the threshold. AROC stands for the " area under the receiver operating characteristic curve [54]". The curve is developed from plotting the true positive rate (TPR) versus false positive rate (FPR) over a series of threshold values. The area under the curve represents the performance of a program, with a value of .5 representing no predictive power and 1.0 perfect predictive power [55]. Empirically, the *AROC* value between .5 and .7 is considered less predictive, between .7 and .9 moderately predictive and over .9 highly predictive. Various methods have beed developed to find an optimal decision threshold based on the prevalence and the costs of incorrect classification [56]. Generally, a higher decision threshold is preferred if the prevalence is low, or if the cost of treating a false positive is greater than that of a false negative. A lower decision threshold is preferred if the condition is reversed. Some researchers use "Matthew's correlation coefficient" to determine the prediction accuracy, which is $\frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TN+FN)(TP+FN)(FP+TN)}}$ [57].

Works comparing various MHC-binding prediction programs have been published and selected ones are reviewed here. Yu [10] compared MHC-binding peptide prediction methods on comprehensive sets of binding data for HLA-A*0201 and HLA-B*3501. The methods include one

HLA	Non-Binders	Binders				
		T-cell Epitopes	Naturally Processed Peptides	Poly-Ala	Other Synthetic	Total Binders
A*0201a	787	123	33	44	159	359
A*0201b	787	183	57	44	159	443
B*3501	128	24	-	-	82	106

Table 2.3: Data set for MHC-binding peptide comparison

motif-based method (SYFPEITHI[29]), three matrix-based methods (YK0201[10], YKW0201[10], and BIMAS[31]), one HMM method (HMMER [58]) and one ANN method (PlaNet[59]). Table 2.3 shows the composition of the data set for the two alleles used. Each set of binders is composed of four subsets: T-cell epitopes, naturally processed peptides, poly-A variants and synthetic peptides. The difference between the initial data set (A*0201a) and expanded data set (A*0201b) for HLA-A*0201 is that the latter includes 60 more T-cell epitopes and 24 more naturally processed peptides. The binders were retrieved from MHCPEP and the non-binders were collected from the same literature sources as the binders in MHCPEP.

For allele A*0201, the *AROC* values for the 6 methods using the initial set are in the range of 0.81-0.87, which is considered not excellent, but good. Of the six methods, ANN, HMM and YKW0201 have slightly higher accuracies than the other three. Unexpectedly, the addition of 84 binders in the expanded set resulted in improved accuracy only for ANN [10]. The accuracies of the same method vary on different subsets of the data in table 2.3. In terms of specificity, the best preformed methods on initial set are ANN and BIMAS. When the expanded set is used, the best methods remain the same, except that the ANN's performance is further improved. For allele B*3501, the *AROC* values are in the range of 0.65-0.75 across all methods, which is poor. SYFPEITHI, HMM and YKW have *AROC* values of 0.75, 0.75 and 0.72, respectively and the *AROC* values for BIMAS, ANN and YK are all below 0.70. Conclusions from Yu's comparison are the following: (a). No single method outperforms all the others. The best choice of method is depend on the quality of data, the extent of bias in the training data set, and the intended purpose of the prediction. If only the size of data is being considered, the best methods are: for small amounts of data (less than 40), motif and matrix-based methods; for moderate amounts of data (40-100), HMM; and for large amounts of data (over 100), HMM and ANN. (b). Predictive accuracy of a program may not be consistent across alleles. (c). The composition of a data set (source of data) affects the predictive accuracy.

Another group of researchers [60] compared matrix-based methods with ANNs using two custom programs HLAPred and ANNPred. The data for model training and testing are drawn from the

Software Package	Predicting Methods	Accuracy	# of alleles being tested (training data size / allele)
HLAPred	Matrix-based (M)	88.1±4.8	47 (>40 peptides for 30 alleles, 15-40 peptides for 17 alleles)
ANNPred	ANN(A)	88.3±4.8	30 (>40 peptides for all alleles)
ComPred	M+A	93.6±2.9	30 (>40 peptides for all alleles)

Table 2.4: Predictive Accuracy Results of Three MHC-binding Peptide Predicting Methods

MHCBN database [61], a database created and maintained by this group. Predictive accuracy is used to evaluate the performance of the programs. The average accuracy of the matrix-based method (HLAPred), based on tests for 47 MHC alleles, is 85.4 (mean)±8.9 (deviation)%. The average accuracy for ANNs (ANNPred) is 87.6±5.9%, based on tests for 30 MHC alleles. The reason that the latter was tested only for 30 of the 47 alleles is because there are less than 40 (15-40) binding peptides available for 17 alleles. Less data for those 17 alleles may reduce the average accuracy of the matrix-based methods. As a result, the difference between these two kinds of methods is not significant. This group found that the accuracy of the individual method varied from allele to allele and so they reasoned that if methods were combined they might improve overall accuracy across alleles. Hence, they created a hybrid approach combining the matrix-based method and ANN and implemented by a program called ComPred. Together with HLAPred and ANNPred, ComPred was tested for predictive accuracy using data generated by Parker et al. [62]. The results of comparison are shown in Table 2.4. The hybrid method resulted in about 5% improvement in mean accuracy.

Donnes [48] did a comparison of a support vector machine method (SVMHC), a motif-based method (SYFPEITHI), and a matrix -based prediction method (HLA_BIND) over six MHC alleles, by using binding peptides from the SYFPEITHI database. The non-binders are extracted randomly from the ENSEMBL database [63]; i.e. a protein sequence from ENSEMBLE is chopped to the lengths of interest and known binders are removed. The ratio of binders to non-binders in the training data is 1:2. Analyzing the results with Matthew's correlation coefficient (Mcc) showed that a minimum of 20 binders gave successful predictions with all three methods. There was a small performance improvement if the training data (binders) was increased to 50 peptides. With 20 binders, the Mcc values for SVMHC, SYFPEITHI and HLA_BIND are .85, .75 and .62 respectively. The accuracy of machine learning technique SVMHC is slightly higher than the other two techniques.

All the comparisons shown above agreed that machine learning techniques outperform simpler

methods when data size is big enough. For people developing MHC-binding predictions these more sophisticated methods are of preference, because data set size is guaranteed to grow as time goes on. Two popular machine learning techniques, HMM and ANN, have high predictive power when the data set is large. According to Yu [10], HMM needs less (100 peptides) training data than does ANN. For some HLA alleles there are more than 100 binding peptides, but for MHC in other species, 100 is still a big number. Thus, HMM seems to be a preferred method to explore at the present time. Moreover, an ANN is a very sophisticated system, which tries to include every feature into consideration. To get the network to perform well, it happens quite often that users have to experiment with even more parameters than are needed by an HMM. An HMM gives a user more controllability than an ANN does and at the same time it has comparable performance with ANN, even when the data set size is getting more than 100 peptides [10]. In this sense, HMM is also preferable to ANN. However, the topology of an HMM is very important to its functionality and most existing HMM programs (NetMHC and HMMER) implement a Profile HMM. Other types of model structure are certainly worthy of more exploration. Based on these considerations, this thesis project will investigate the HMM methods.

Chapter 3

Research goal

3.1 Specification of thesis problem

Vaccine development in cattle is economically worthwhile and is being pursued by some researchers in the Vaccine and Infectious Diseases Organization (VIDO) in Saskatoon, SK, Canda. However, because of the costs associated with traditional vaccine development, computer based methods of vaccine screening are coming into play.

As discussed previously, many software programs have been developed to conduct rational vaccine design. However, most of the existing programs were developed using human and mouse data because these two data sets are most prevalent and most vaccine design techniques require large data sets. Many of these programs are not open source, which means they only do predictions for the alleles that existed in their database. As usually only human or mouse MHCs are available in their database, it is not possible to use these programs to do prediction on alleles from other species. Thus an open source prediction program is helpful to researchers working on alleles from various species. Open source here means the source code is free and others can modify the source code.

Machine learning techniques have their advantages in doing MHC-binding peptide prediction. Also, there are some successful cases of using these techniques to predict peptides that bind to MHCs and some of the predicted peptides even elicit an immune response [2]. However, there is still room to make improvements on the predictive accuracy of these machine learning programs.

3.2 Objectives and hypothesis

The goal of this project is to develop an HMM-based, open source, accurate MHC-binding peptide predictor.

The majority of free vaccine development programs use matrix-based methods. According to Yu's method comparison, HMM and ANN outperform matrix-based methods and HMM needs less training data than ANN does, without having to comprise on the performance. Thus HMM is taken by this program to do MHC binding peptide prediction. Like many other programs, this program

will be developed on data from human and mouse. But nothing will be done to compromise its ability to analyze data from other species. Indeed, eventually the program will be used to analyze data from cattle. The source code of this program is free to the computational immunology community. Researchers with their own training data for some alleles can use this program. As to the predictive accuracy issue, this program will strive to achieve higher predictive power by using two heuristic approaches.

The hypothesis of this research is that the two heuristic approaches will help to improve the model's predictive power. The effect of these heuristic approaches will be shown by a performance comparison between this program and HMMER.

Chapter 4

Data and methodology

4.1 Data

This program is tested on data for two human MHCs, namely HLA-A*0201 and HLA-B*3501. The data sources for these alleles are existing databases and literature. For each allele, there are two types of models, a positive type ("positive models") and a negative type ("negative models"), being set up. The positive models are built from peptides that bind to the allele ("positive training data") and the negative models are built from peptides that do not bind to the allele ("negative training data").

Positive training data are extracted from MHCPEP [21]. Note, that peptides in this database are labeled with four levels of binding affinity: strong, mediate, weak and unknown. For both alleles, only peptides with strong binding affinity are used. In this case, 116 peptides were collected for allele HLA-A*0201 and 70 were collected for HLA-B*3501. The reason that only strong binders were selected is that it is suspected that the patterns in these sequences is more consistent, and this makes the model training easier.

Negative training data are from two sources, namely MHCBNa and a search paper. Sixty sequences[1] were extracted from MHCBN [61] and an additional 237 peptides were selected from a research paper[64], which lead to a total of 297 sequences of non-HLA-A*0201 binding peptides. According to the literature, the 237 peptides do not bind to HLA-B*3501 either. Thus they were used as negative training data for HLA-B*3501.

Testing data are chosen from the training data in a cross validation approach (discussed below in testing environment for proposed solutions). The training and testing data for HLA-A*0201 and HLA-B*3501 are listed in Appendix 6.3 and 6.3, respectively.

[1] 32 of the sequences do not bind to HLA-A*0201 and the other 28 do not have T-cell activity, which implies they do not bind to HLA-A*0201.

4.2 Model's ability to achieve the global maximum

The model training and testing program is written in Java programming language. The implementation of the Baum-Welch algorithm was checked by a hand trace through a small parameter and data set. The correctness was further tested by the experiment shown in this section. Baum-Welch algorithm is not guaranteed to find the global maximum. However it should, under certain conditions (such as a large amount of trianing data, a small parameter space and a model training point that is close to the global maximum) have a good chance to find the global maximum. This experiment tends to create such a condition so that if the model achieves a probability that is higher than that given by the starting point by not too much, then the Baum-Welch algorithm is implemented as expected. This test is composed of a small artificial data set that consists of only a few states and symbols, a known global maximum, and a set of simulated training data.

4.2.1 Testing approach

The testing approach is as follows. First, the transition and emission matrices are set randomly. Using these matrices, the model generates input sequences. Using the same matrices, the model is trained by the input sequences. Finally, the distance that the parameters have strayed from where they started is examined. This process is diagramed in Figure 4.1. This process is elaborated on in the following paragraphs.

The topology of this small testing model is the fcHMM described in Section 2.5.3. However, this testing model has only four states: 1, the starting state, 2, the first emission state, 3, the second emission state, and 4, the ending state. A, C, D and E are the only symbols which can be emitted. The initial emission and transition matrices are shown below.

a) emission matrix

	A	E	D	C
1	0	0	0	0
2	0.20	0.15	0.50	0.15
3	0.20	0.40	0.20	0.20
4	0	0	0	0

b) transition matrix

	1	2	3	4
1	0	0	0	0
2	0.3333	0.78	0.30	0
3	0.6667	0.2	0.68	0
4	0	0.02	0.02	1.0

These two matrices are named "fixed matrices". With these fixed matrices, the model is used to generate 350 sequences. Thus, the fixed matrices give a probability that is close to the global maximum for this set of sequences. Then using the fixed matrices as initial matrices, the model is trained by the sequences just generated. When the model converges, the probability of generating

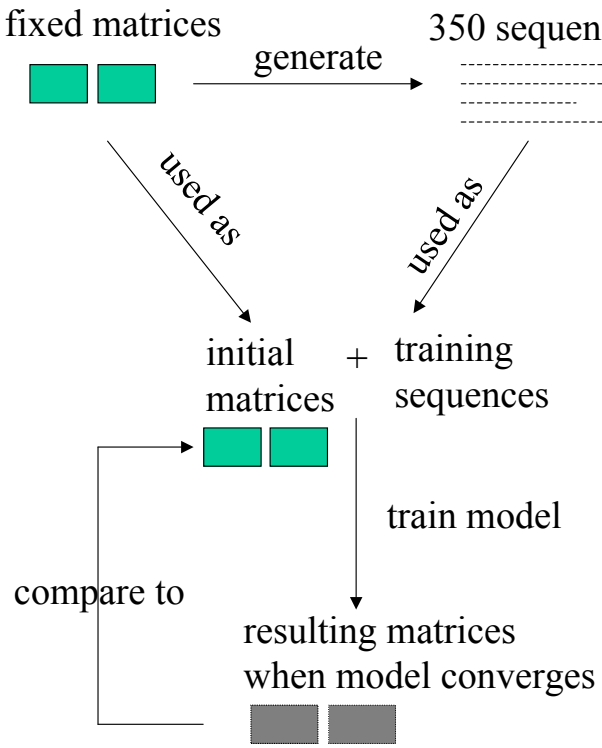

Figure 4.1: The process of model testing

all 350 sequences by the resultant two matrices is compared to that by the fixed matrices. The former value is expected to be higher than the latter one but the distance between these two values should not be far. The method to evaluate if the distance is too far is discussed below. In order to make sure that the model does converge, the model is trained for 15,000 cycles (the threshold to stop the training is 8E-18). The result of this experiment given in Sections 5.1 and 5.2 shows that this number of training cycles is enough to make the model converge. Details of explanation are given in Section 4.2.1.

Each of the 350 sequences are generated as follows: to determine the symbol for the next position of the sequence, the model first selects a state randomly, which is based on the state of current position. Then with the state chosen, a symbol can be randomly picked up for this state. The state selection is based on the transition probabilities in the transition matrix and the symbol selection is based on the emission probabilities in the emission matrix. As this process of state and symbol selection goes on, a sequence of symbols is generated. When the ending state is picked, the symbol generation ends.

The method to analysis the result of this experiment is called likelihood ratio test: Let P_1 be the probability of generating all the 350 sequences from the fixed matrices. Let P_2 be the probability of generating all 350 sequences from the resultant matrices. Then $-2 * \ln \frac{P_1}{P_2}$ gives the deviance. If the deviance is smaller or equal to the number of parameters (or degree of freedom), then it means

the model has the potential to achieve the global maximum [65].

4.3 Proposed solutions

An HMM is not a new method to solve the MHC-binding peptide prediction problem. Previous attempts [41, 10, 40] have achieved a certain amount of success. However, there are a number of issues that have not been addressed by their work. Four of the issues are discussed below.

4.3.1 Model's topology

The first issue is related to the model's topology, or structure of the HMM, which has a big impact on the model's performance. An HMM can have various topologies. Researchers choose a topology according to the specification of the problem at hand. Section 2.5.3 discussed two types of topologies, pHMM and fcHMM, and problems associated with each of them. In response to these problems, a topology whose connectivity lies between that of a pHMM and a fcHMM, called a partially connected or pcHMM, is explored in this research.

A pcHMM is constructed by first forming a fcHMM. Then the model training process eliminates some of the transitions or states in the fcHMM to generate the pcHMM. This allows the training data to "decide" the correct topology for the model starting from a fcHMM.

The pcHMM is constructed from the fcHMM in an iterative process. We give two heuristic topology reduction approaches: the first is a transition removal approach and the second is a state removal approach. For the transition removal approach, after the fcHMM is formed, the transition with the lowest probability is removed. This means that this transition is no longer considered in the subsequent training. In the next stage, training starts with this transition being omitted and the transition rates of the remaining transitions in the model increased proportionally. When the model converges again, the transition with the lowest transition probability is once more removed. That is, each time a transition is removed a model has to be trained again. Consequently if N transitions are removed, then N models are trained. In this research, 120 transitions out of 144 transitions (all transitions in a 12x12 transition matrix) are removed. This left 24 transitions in the model and it means that every state gets about 2 transitions out from it on average. The complexity of the resulting model is considered too low for the current problem. For the state removal approach, instead of transitions, states are removed at each iteration. Deciding which state to remove is a simple process. First, the states are ordered by their highest incoming transition rates. The state with the lowest incoming transition rate is removed and the transition rates for all the rest transitions in the model are increased proportionally. This process continues until 8 states out of 12 states are removed. Again, the number 8 is chosen because the rest of the model is considered too small after this. The process of transition and state removals is illustrated in Figure 4.2. After each

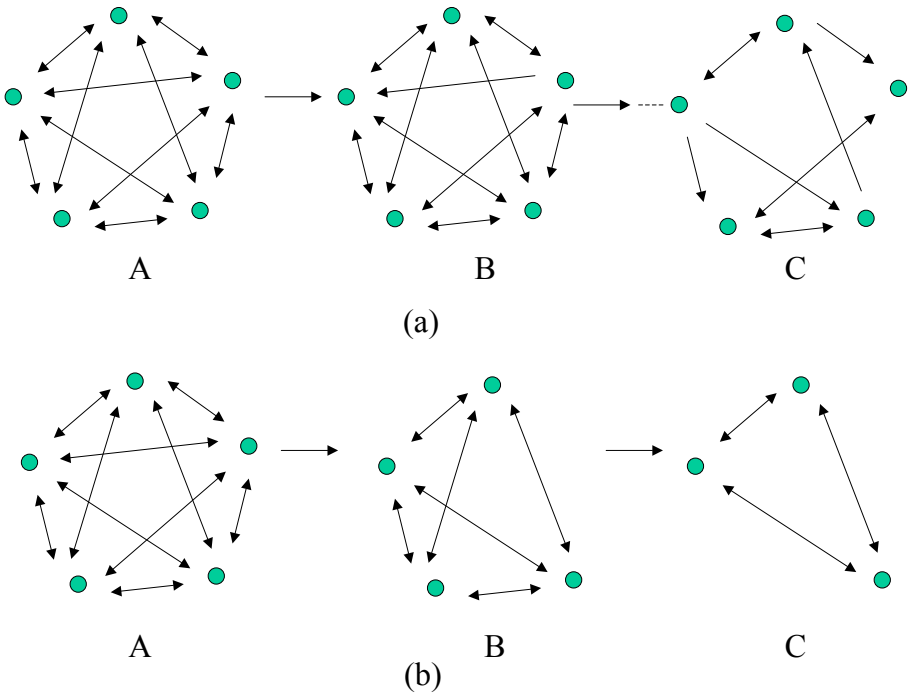

Figure 4.2: Transition and State Removals

transition/state removal, the model's predictive accuracy is tested. With the relationship between model's topology and the model's performance available, the topology with the best performance will be the proper topology for the problem at hand.

The reason to do topological reduction from a fcHMM is that in some cases, the model's overall performance will increase as the number of transitions/states decrease. In these cases, there are patterns occurred frequently in the training data and there are patterns occurred with low rates. When a model has a high level of connectivity, not only those frequently occurred patterns are recognized as true positives, but those rarely occurred ones have good chances to be recognized by the model as true positives too. This leads to a high rate of true positives (high sensitivity) and a high rate of false positives (low specificity). Now if some ignorable connectivities in the model are removed so that the complexity of the model is just enough for it to pick up patterns occurred frequently (true positives) but not those occurred rarely, then the model's sensitivity does not decrease much while its specificity increases dramatically.

4.3.2 Biological meanings of states

The first issue to consider with HMMs is the biological meaning of the states. In the fcHMM created by Mamitsuka [39], a state doesn't have a real-world meaning. Because of this lack of meaning, the number of states to have in a fcHMM has to be determined through a brute force approach, i.e. try out a range of numbers and pick out the one that gives the best result. Again

because of the lack of meaning for a state, the state transition and symbol emission matrices are forced to be initialized uniformly. The reason as to why uniformly initialized matrices are not preferable is discussed below 4.3.3. Alternatively, the brute force approach can also be used in finding good initial matrices. However, the problem with brute force is that in average it takes a lot of time of find a good result. In a pHMM, there are two kinds of states that emit symbols, the normal state and the insert state. Each of these emitting state corresponds to one position of the binding peptide. But this correspondance has to be decided by MSA, which only gives one out of many possible good alignments. To measure the transition and the emission rates base only on one alignment is certainly not preferable.

Observation of known binding motifs of HLA-binding peptides shows that amino acids commonly replace each others in the peptide. Thus, in this research, a state in the pcHMM will correspond to a set of related amino acids. The symbols that a state emits correspond to the set of amino acids that share some properties. Deciding which amino acids can be emitted by a state can be derived from various methods, e.g. chemical properties of the amino acids (charge, hydrophobicity etc.), a substitution matrix ([1]), etc. Based on the examination of some existing motifs, using physical/chemical properties of amino acids seems to be the best criterion for choosing how amino acids should be grouped together and thus it is adopted by this research. Biochemists classified 20 amino acids into eight groups according to one of the physical/chemical properties. This type of grouping will be referred to as "single property grouping". Figure 4.3 shows the eight groups, with one group per line.

aliphatic	L I V
aromatic	F Y W H
non-polar	L I V M C_S C_H F Y W H K A G
polar	Y W H K R D E C_H S T N Q
charged	D E K H R
positive	H K R
tiny	A G C_H S T
small	V C_S C_H P A G S T D N

Figure 4.3: Chemical classification

Although the states could be formed based just on the above figure, it was considered worthwhile to form states of amino acids that share more properties. Table 4.1 shows a 21x21 matrix Z, where Z_{ij} contains the number of groups in which both amino acid i, and j appear. There are 21 amino acids because C is differentiated as C_H and C_S. Information from this matrix is extracted and

	L	I	V	CS	P	A	G	C_H	S	T	D	N	E	Q	K	R	H	Y	W	F	M
L	0	2	2	1	0	1	1	1	0	0	0	0	0	0	1	0	1	1	1	1	1
I	2	0	2	1	0	1	1	1	0	0	0	0	0	0	1	0	1	1	1	1	1
V	2	2	0	2	1	2	2	2	1	1	1	1	0	0	1	0	1	1	1	1	1
CS	1	1	2	0	1	2	2	2	1	1	1	1	0	0	1	0	1	1	1	1	1
P	0	0	1	1	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
A	1	1	2	2	1	0	3	3	2	2	1	1	0	0	1	0	1	1	1	1	1
G	1	1	2	2	1	3	0	3	2	2	1	1	0	0	1	0	1	1	1	1	1
C_H	1	1	2	2	1	3	3	0	3	3	2	2	1	1	2	1	2	2	2	1	1
S	0	0	1	1	1	2	2	3	0	3	2	2	1	1	1	1	1	1	1	0	0
T	0	0	1	1	1	2	2	3	3	0	2	2	1	1	1	1	1	1	1	0	0
D	0	0	1	1	1	1	1	2	2	2	0	2	2	1	2	2	2	1	1	0	0
N	0	0	1	1	1	1	1	2	2	2	2	0	1	1	1	1	1	1	1	0	0
E	0	0	0	0	0	0	0	1	1	1	2	1	0	1	2	2	2	1	1	0	0
Q	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	1	1	0	0
K	1	1	1	1	0	1	1	2	1	1	2	1	2	1	0	3	4	2	2	1	1
R	0	0	0	0	0	0	0	1	1	1	2	1	2	1	3	0	3	1	1	0	0
H	1	1	1	1	0	1	1	2	1	1	2	1	2	1	4	3	0	3	3	2	1
Y	1	1	1	1	0	1	1	2	1	1	1	1	1	1	2	1	3	0	3	2	1
W	1	1	1	1	0	1	1	2	1	1	1	1	1	1	2	1	3	3	0	2	1
F	1	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0	2	2	2	0	1
M	1	1	1	1	0	1	1	1	0	0	0	0	0	0	1	0	1	1	1	1	0

Table 4.1: The number of common chemical properties shared between each pair of amino acids

turned into a directed graph 4.4 in the following way: for the numbers in each column, if $Z_{ij} <= 2$ for all i, and if $Z_{ij} = 2$, then amino acid pair ij is selected and drawn as two nodes with a directed connection from j to i in the graph; if $Z_{ij} <= 4$ for all i, and if $Z_{ij} >= 3$, then amino acid pair ij is selected and drawn as two nodes with a directed connection from j to i. From this graph all cliques (a group of fully connected nodes) are picked out to form groups. Any amino acid not in a group forms its own group of one. Note that at this point,C_H and C_S are treated as the one amino acid C. The result of this technique can be seen in figure 4.4. The biological meaning of this approach is that it groups amino acids by multiple chemical/physical properties that are shared by amino acids in a group. This approach gives 10 states and will be referred to as "multiple property grouping".

The result of multiple property grouping is show in figure 4.5, with one group on each line.

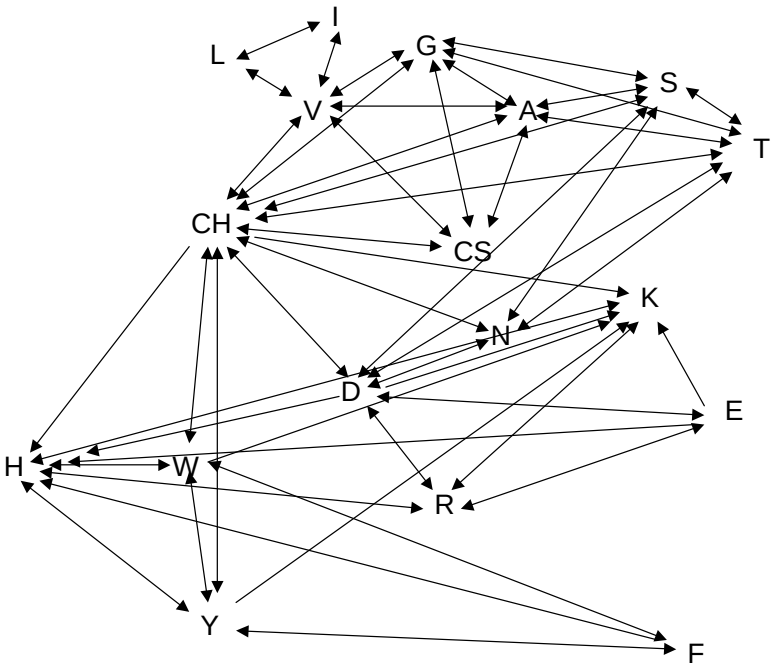

Figure 4.4: A directed graph that encodes the amino acid pairs with more than 3 (inclusive) common chemical properties

L I V

V C G A

A S T G C

N C S T D

E D R K H

K C Y W H

Y F W H

M

Q

P

Figure 4.5: New sets of amino acids developed by using multiple property grouping

4.3.3 Heuristic approaches to initialize matrices

Another issue to be solved when using HMMs is how to find good initial states for emission and transition matrices. Had an HMM training algorithm been able to achieve the global maximum, the initial state of the matrices should not matter; from any matrices, the model would converge to the matrices that give the global maximum. However, this is not the case for the Baum-Welch (BW) algorithm which is only guaranteeed to find a local maximum. One way to get around this problem is to use many different initial matrices, each of which will generate a local maximum. The one that gives the largest local maximum would then be used [36]. Unfortunately, this approach is not feasible here as the number of possible initial matrices is too large. A better way to tackle this problem is to use knowledge about the peptide sequences to help select initial states of the emission and transition matrices [35]. In this proposal, the physical/chemical properties of amino acids are used to define states and determine the emission probabilities for states, and the adjacent amino acid counts in the training sequences are used to determine the state transition probabilities. By doing this, initial emission and transition matrices can be generated. The details of how to do this follows.

The initial transition matrix is formed from training peptide sequences as follows. Adjacent amino acids in a peptide are considered transitions from the amino acid on the left to the amino acid on the right. With the different states in the HMM defined in Section 4.3.2, amino acid transitions can be mapped to state transitions. An example of this is given in Figure 4.6. The left column of the figure shows all adjacent amino acid pairs in the peptide. The entries in the right column show all possible transitions that can generate the amino acid pair in the left column. 1 is the starting state, 10 is the ending state. State numbers 2 to 9 in this figure correspond to the first line to the last line in Figure 4.3. Figure 4.7 records the counts of all the transitions in this sequence. The numbers in the first row are "from" states and the numbers in the first column are the "to" states.

With the data in Figure 4.6, we can calculate the rate of transition from one state to any other. For example, the number of transitions from state 9 to state 2 is 2 and the total number of transitions from state 9 to all other states is 8. Thus, the rate of transition from state 9 to 2 is $\frac{2}{8}$, or 0.25.

The emission matrix E is an $|N|$x$|M|$ matrix, where N is the set of states, M is the set of 20 amino acids, and E_{ij} is the probability of amino acid j ($j \in M$) being emitted by state i ($i \in N$). We define for each state i, the set of amino acids that belong to it as A_i (the cardinality of this set being $|A_i|$), and those that do not as $\bar{A}_i$ (whose cardinality is $|\bar{A}_i|$). We then define two probabilities, Y, the probability of emitting amino acids that belongs to state i when in state i, and $Z = 1 - Y$, the probability of emitting some other amino acids when in state i. Then $E_{ij} = \frac{Y}{|A_i|}$ if j is an element of A_i, otherwise $E_{ij} = \frac{Z}{|A_i|}$. Arbitrarily, Y and Z are set to be 0.9 and 0.1, respectively. Appendix

peptide: GILGFVFTL

all adjacent amino acids pairs: (G, GI, IL, LG, GF, FV, VF, FT, TL, L), where "(" represents the starting position (state 1) and ")" represents the ending position (state 10).

amino acid pair	state transitions (from–>to)
(G	1–>4, 1–>8, 1–>9
GI	9–>2, 9–>4, 8–>2, 8–>4, 4–>2, 4–>4
IL	2–>2, 2–>4, 4–>2, 4–>4
LG	2–>4, 2–>8, 2–>9, 4–>4, 4–>8, 4–>9
GF	4–>3, 4–>4, 8–>3, 8–>4, 9–>3, 9–>4
FV	3–>2, 3–>4, 3–>9, 4–>2, 4–>4, 4–>9
VF	2–>3, 2–>4, 4–>3, 4–>4, 9–>3, 9–>4
FT	3–>8, 3–>9, 4–>8, 4–>9
TL	8–>2, 8–>4, 9–>2, 9–>4
L)	2–>10, 4–>10

Figure 4.6: All the state transitions in peptide GILGFVFTL

6.3 show the training sequences that are used to derive the initial matrices for allele HLA-A*0201. Appendix 6.3 and 6.3 show the initial matrices derived for HLA-A*0201, based on the states given by "single property grouping" and "multiple property grouping", respectively.

4.3.4 Training data

The fourth issue concerns the training data. Many previous HMMs do not use negative training data to act as a control for the model. Without a corresponding model developed from negative training data, the criterion to categorize a testing peptide is subjective to a certain degree. In this proposed solution, two separate models are trained from positive and negative training data, respectively. Except that the training data are different for the positive and the negative model, the ways to derive states, initial matrices, and the way to do model training and testing are the same for the two types of models.

Above issues are carried (tested) out in the following environment:

- A fcHMM that is implemented with the Baum-Welch algorithm.
- Two types of training data, positive and negative training data; a separate model is set up from positive data and negative data, individually.

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	1	1	3	0	0	0	2	2	0
3	0	1	0	2	0	0	0	1	2	0
4	1	3	1	6	0	0	0	2	4	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	0
8	1	1	1	2	0	0	0	0	0	0
9	1	1	2	3	0	0	0	0	0	0
10	0	1	0	1	0	0	0	0	0	0

Figure 4.7: Counts of transitions in the sample sequence in Figure 4.6

- A cross-validation approach for model training and testing: Each sequence in the data set (both positive and negative) is used, in turn, as testing data. When a particular sequence is selected to be the testing data, the remaining sequences are used to train the model. The model is then used to classify the testing sequence. The positive training data gives the testing data a score and the negative training data gives it another score. The higher score decides the binding identity of the testing sequence.

- The threshold to stop the training process is 0.008. Let the logarithm of the model's probability for generating all training sequences in the last cycle be P_{cur} and that of the second-to-last cycle be P_{prev}, then the training is stopped when $\frac{|P_{cur}-P_{prev}|}{P_{prev}}$ is smaller than 0.008. 0.008 is an emperical value, which is a tradeoff between the number of training cycles and the predictive accuracy.

- The training and testing programs are written in Java programming language. The testing program generates score (numerical values) for each testing sequence. The programs to analyze these scores and do subsequent classification are written in shell scripts and Perl.

- The machines used to run programs include a cluster of 32 PIV, 2G RAM, 2.38GHz CPU machines, a cluster of 32 PIII, 900M RAM, 866MHz CPU machines and 50 PIV, 256M RAM, 1.8GHz CPU machines.

The effectiveness of these approaches is examined through performance comparison to the most popular pHMM application, HMMER. The details of this comparison is given immediately below.

4.4 Comparison to HMMER

As a popular pHMM for various sequence alignment related tasks, HMMER can be used for vaccine discovery. HMMER is compared to the "local program" (program developed for the current project) in terms of predictive accuracy. The data used by HMMER is the same as the data used by the local program (see Section 4.1, and so is the training and testing approach (the cross-validation approach in Section 4.3.4). That is, each sequence in the data set (both positive and negative) is used, in turn, as testing data. When a particular sequence is selected to be the testing data, the remaining sequences are used to train the model. Thus, for each testing sequence, there is a positive model and a negative model, each of which gives the testing sequence a score. The higher one of the two scores determines the category of this testing sequence.

HMMER reports two scores for every testing sequence, a bit-score and a E-value. The bit-score indicates how likely this sequence could be a member of the training sequences. The E-value is how likely it is to get a sequence with this bit-score just by chance, when searching in the database formed by the training sequences. The binding identity of a testing sequence is determined by comparing its positive bit-score to its negative bit-score. This approach does not require the E-value to be considered. A bit-score is defined as:

$$\text{bit-score} = \log_2 \frac{P(seq|pHMM)}{P(seq|null)}$$

The numerator is the probability of generating the testing sequence by the profile HMM, which is built from the training sequence alignment. The denominator is the probability of generating the testing sequence by a null model. The null model is a model built from protein sequences in PROSITE, representing the general amino acid composition of proteins. When a negative bit-score is subtracted from a positive bit-score, the values given by the null model in the two scores cancel out. The subtraction results in a (log-value) ratio of the positive and negative pHMM scores.

HMMER's predictive accuracy is measured as follows: the positive (and negative) training sequences are aligned by ClustalW [37](with all settings as default). A profile HMM is built based on the alignment with "hmmbuild". Based on the this profile HMM, "hmmsearch" generates a score for a testing sequence. All the parameters in HMMER are set by default values.

Chapter 5

Results

This section will cover three aspects of the results: first, the model's ability to reach the global maximum is investigated using artificial training data; second, the effect of using two heuristic approaches (as out-lined in sections 4.3.1 and 4.3.3) on real biological data; last, a performance comparison of this program to a typical pHMM program, HMMER, when both are run on real biological data.

5.1 Model's ability to achieve the global maximum

Using the fixed matrices shown in section 4.2.1 as initial matrices and the 350 sequences as training data, the model converged to the "resultant" transition and emission matrices shown below. Figure 5.1 and Figure 5.2 show the value changes of the transition parameters and the emission parameters during the model training process. As the model training exceeds 3,000 cycles, the values of these parameters stay stable. When these parameters do not change values, the model's probabilities of generating all the sequences in two continuous cycles is so small that the model can be considered as converged.

a) emission matrix

	A	E	D	C
1	0	0	0	0
2	0.2027	0.1284	0.5080	0.1609
3	0.2069	0.3957	0.2204	0.1770
4	0	0	0	0

b) transition matrix

	1	2	3	4
1	0	0	0	0
2	0.2861	0.7353	0.2861	0
3	0.7139	0.2480	0.6903	0
4	0	0.0167	0.0236	1.0

The natural log-probability of generating all 350 sequences by the fixed matrices is -1369.8243. The natural log-probability of generating all 350 sequences by the resultant matrices is -1369.1762. According to the method described in section 4.2.1 for result analysis, $P_1 = e^{-1369.8243}$ and $P_2 = e^{-1369.1762}$, where $e = 2.71828....$ P_2 is larger than P_1 and the deviance is 1.29. Originally there are

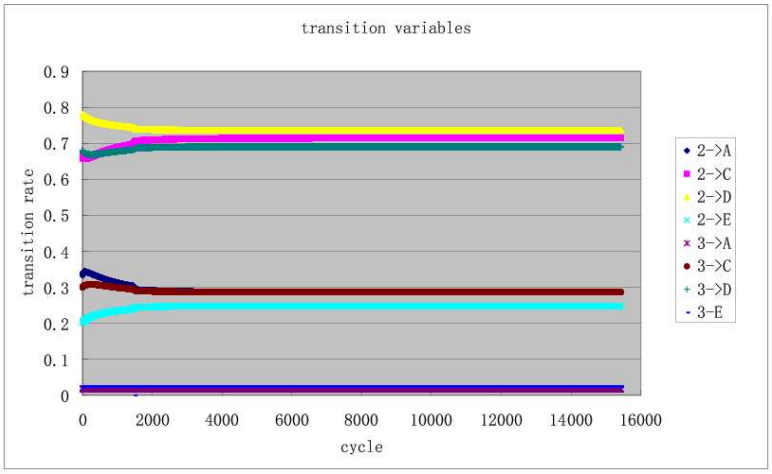

Figure 5.1: The convergence process of transition variables

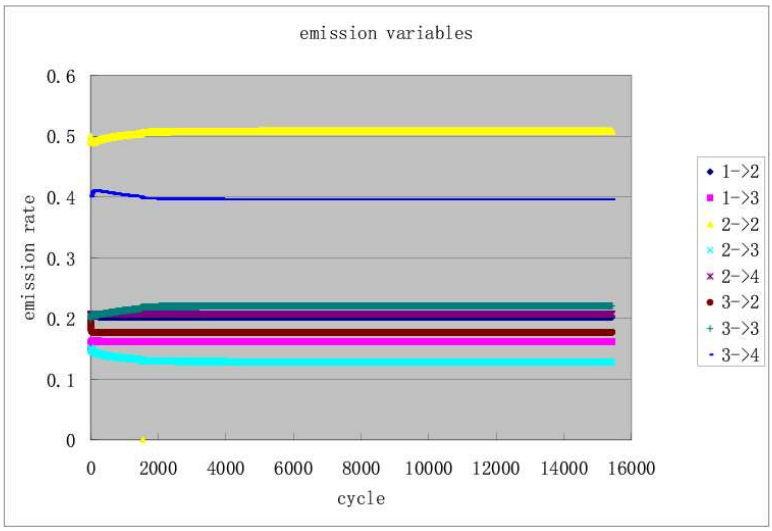

Figure 5.2: The convergence process of emission variables

8 variables in the emission matrix and 8 variables in the transition matrice (section 4.2.1). However, since all emission variables (4 of them) for a particular state add up to 1, one of the variables is automatically determined when the other three's values are fixed. This brings the total number of emission variables from 8 down to 6. The similar situation with transition variables reduces the total number of transition variables to 5. Thus the total number of parameters for the model is 11. Since 1.29 is far smaller than 11, this means the model has the potential to achieves the global maximum, which implies the Baum-Welch algorithm was implemented as expected.

5.2 Effects of proposed solutions

This section examines how the model's topology and initial matrix affect the model's predictive accuracy. Table 5.1 and Table 5.2 show the results for allele HLA-A*0201 and allele HLA-B*3501 respectively in the form of AROC values. The rows show the effect of using various grouping methods to initialize the transition and emission matrices (Section 4.3.3). The columns show the effect of using either transition or state removal topological reduction (Section 4.3.1). The values shown for transition and state removal topological reduction in the two tables are the best performance these approaches can reach. In other words, out of the 120 topologies derived from transition removal, the one that gives the best predictive accuracy is shown. Similarly for state removal, the one being shown is selected from 8 topologies derived. In both tables, the uniform matrix initialization and the fully connected topology are considered as controls. The accuracy of each technique is represented as an AROC value. These same results are presented as individual AROC curves in Figures 5.3, 5.4, 5.5, 5.6, 5.7, and 5.8.

A previous comparison of MHC binding peptide prediction programs performed by Yu et al. [10] showed AROC values in the range of 0.81-0.87. However, since their training data and the way they used the training data are different from those in this study, the results are not comparable. In order to have a performance reference for the local program, HMMER [58] will be used on the same data.

	uniform matrix initialization	single property grouping	multiple property grouping
fully connected	0.860	0.837	0.847
transition removal	0.860	0.858	0.862
state removal	0.861	0.948	0.991

Table 5.1: Model's predictive accuracy comparisons of different initial matrices and different model topologies (allele HLA-A*0201)

	uniform matrix initialization	single property grouping	multiple property grouping
fully connected	0.354	0.873	0.850
transition removal	0.976	0.902	0.917
state removal	0.854	0.854	0.986

Table 5.2: Model's predictive accuracy comparisons of different initial matrices and different model topologies (allele HLA-B*3501)

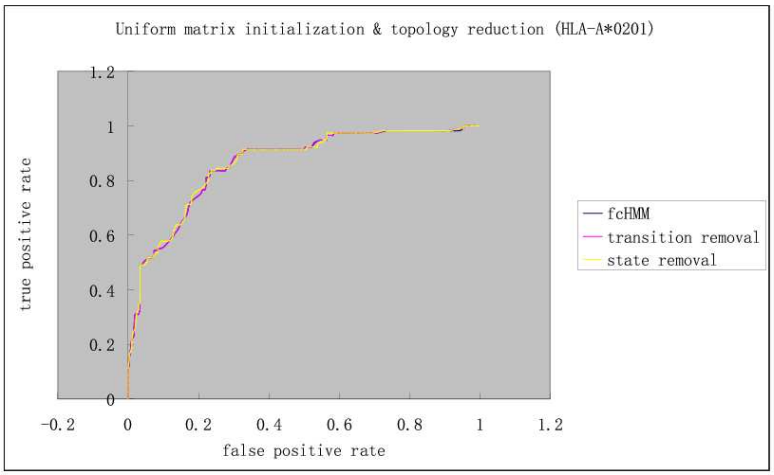

Figure 5.3: AROC curves of the programs using uniform matrix initialization and topological reduction for HLA-A*0201

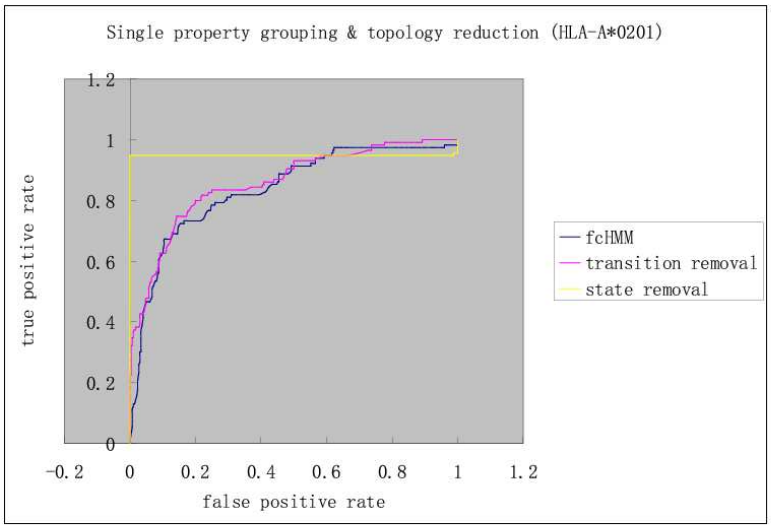

Figure 5.4: AROC curves of the programs using single property grouping matrix initialization and topological reduction for HLA-A*0201

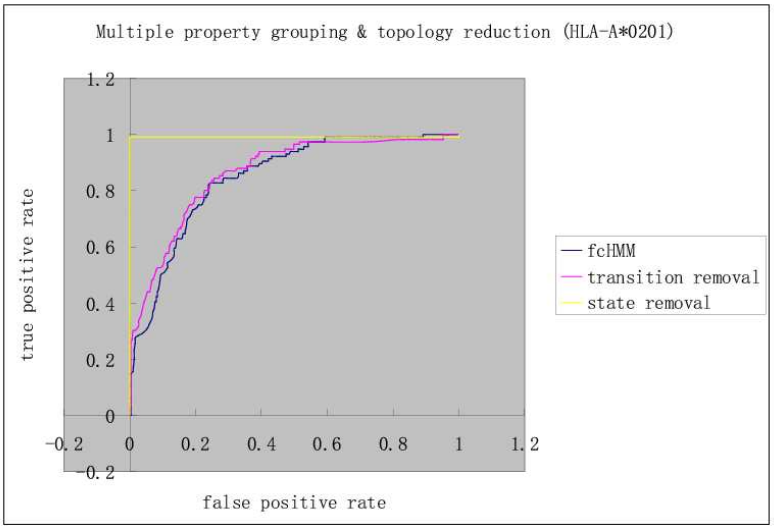

Figure 5.5: AROC curves of the programs using multiple property grouping matrix initialization and topological reduction for HLA-A*0201

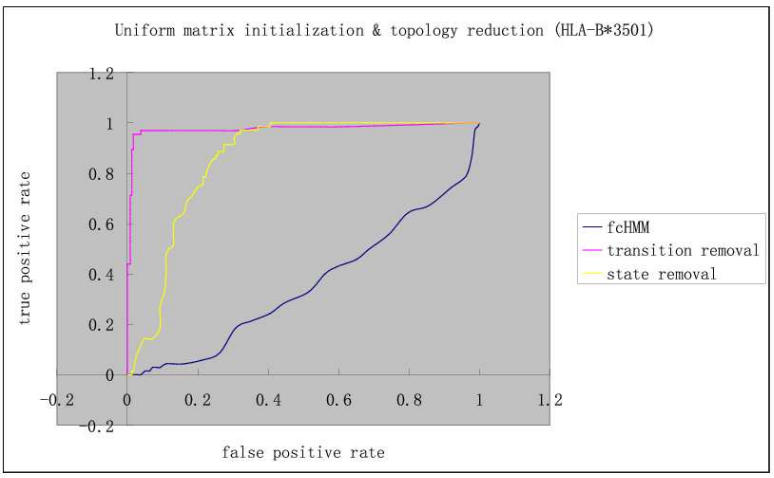

Figure 5.6: AROC curves of the programs using uniform matrix initialization and topological reduction for HLA-B*3501

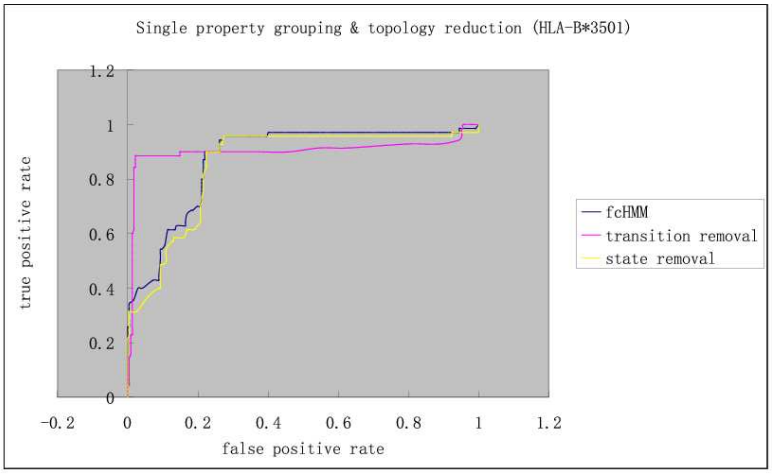

Figure 5.7: AROC curves of the programs using single property grouping matrix initialization and topological reduction for HLA-B*3501

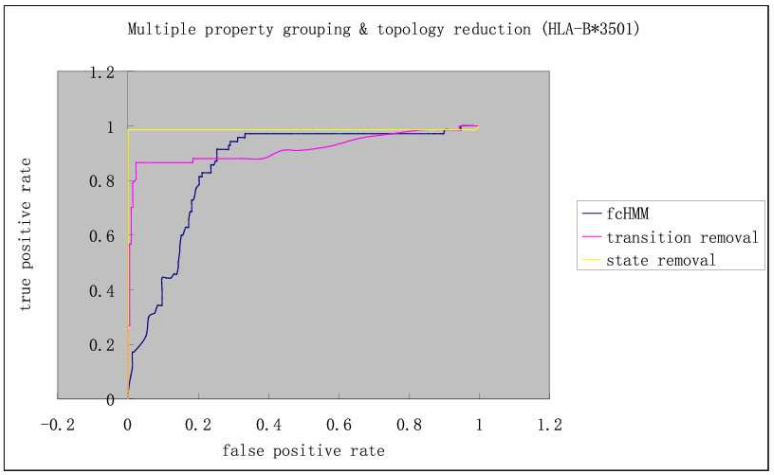

Figure 5.8: AROC curves of the programs using multiple property grouping matrix initialization and topological reduction for HLA-B*3501

5.3 A performance comparison to HMMER

5.3.1 Predictive accuracy

The predictive accuracies of HMMER and the local program are shown through AROC curves. Figures 5.9 and 5.10 show the results for allele HLA-A*0201 and HLA-B*3501, respectively. The predictive accuracies, which are the areas under the curves, are shown in Table 5.3.

Besides the AROC values, sensitivity and specificity are usually extracted from the AROC curves and used to compare performance. As discussed in Section 2.5.6, when the prevalence and the cost of false positives are known, there are methods to pick up an optimal threshold (a point on the AROC curve) to achieve the balance between the cost of false positives and the prevalence. This optimal threshold determines the sensitivity and the specificity. However, when the cost information is not available, the usual way to find values of these measuring parameters is to take the point which gives the minimum sum of false negatives and false positives. The point is the intersection of a line $y = x + c$ and the AROC curve, where c is the maximal value the linear function can get. The sensitivities and specificities of the local program and HMMER are calculated according to this method and displayed in Tables 5.4 and 5.5.

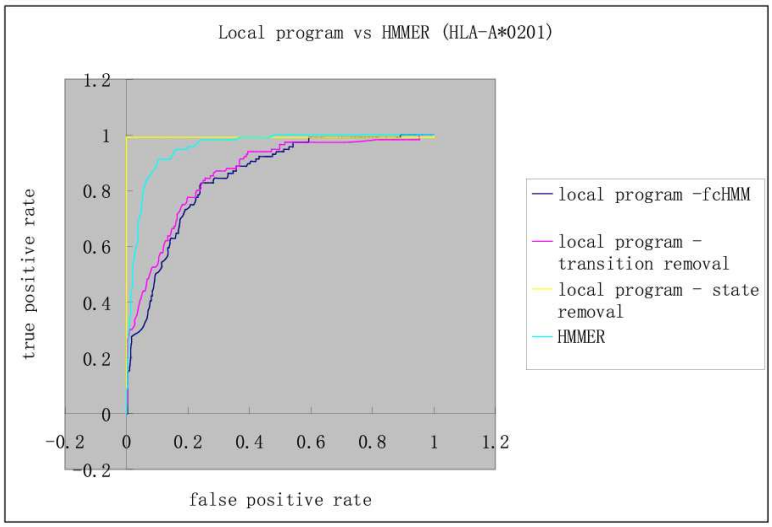

Figure 5.9: Comparison between HMMER and local program (multiple property grouping & topological reduction) on allele HLA-A*0201

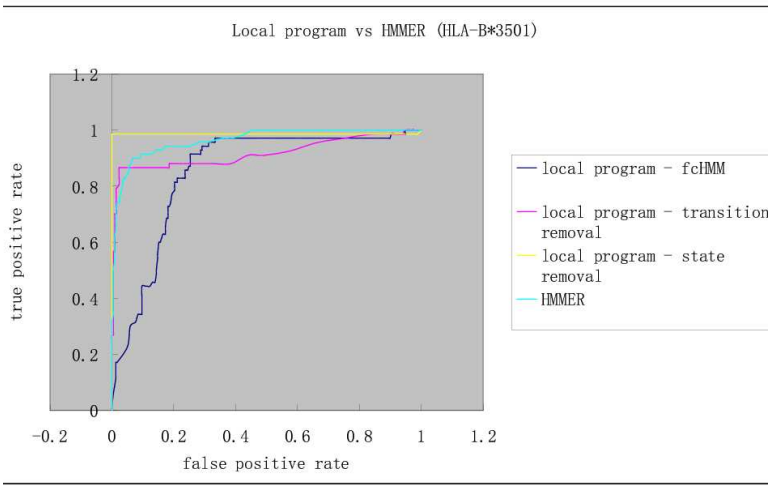

Figure 5.10: Comparison between HMMER and local program (multiple property grouping & topological reduction) on allele HLA-B*3501

	HLA-A*0201	HLA-B*3501
fully connected	0.847	0.850
transition removal	0.862	0.917
state removal	0.991	0.986
HMMER	0.956	0.964

Table 5.3: Predictive accuracy comparisons between the local program using multiple property grouping and HMMER

	sensitivity	specificity
fully connected	0.827	0.751
transition removal	0.845	0.745
state removal	0.991	1.000
HMMER	0.914	0.892

Table 5.4: Sensitivity and Specificity of models for allele HLA-A*0201

	sensitivity	specificity
fully connected	0.914	0.738
transition removal	0.866	0.968
state removal	0.986	0.996
HMMER	0.900	0.932

Table 5.5: Sensitivity and Specificity of models for allele HLA-B*3501

5.3.2 Model's computational time

Computational time used by local program and HMMER on allele HLA-A*0201 and HLA-B*3501 are shown in table 5.6 and table 5.7.

	training		testing	testing environment
	positive model (116 sequences)	negative model (297 sequences)	model testing (single sequence)	
fully connected	7hrs (50machines)	15hrs (50machines)	<1 sec (1 machine)	PIV, 256M DRAM, 100%usage of 1x1.8GHz cpu
transition removal	20hrs (32 machines)	182hrs (32 machines)	<1 sec (1 machine)	PIV, 2G DRAM, 100% usage of 2.38GHz cpu
state removal	8.3hrs (32 machines)	13hrs (32 machines)	<1 sec (1 machine)	PIV, 2G DRAM, 100% usage of 2.38GHz cpu
HMMER	10mins (1 machine)	4hrs7mins (1 machine)	<1 sec (1 machine)	PIV, 256M DRAM, 16.9% and 5.4% usage of 1x1.8GHz cpu for positive and negative training respectively

Table 5.6: Computational time comparisons between the local program and HMMER for allele HLA-A*0201. Each test is done on a single machine.

	training		testing	testing environment
	positive model (70 sequences)	negative model (237 sequences)	model testing (single sequence)	
fully connected	1.2hrs(50machines)	2hrs(50machines)	<1 sec(1 machine)	PIV, 256M RAM, 90%usage of 1x1.8GHz cpu
transition removal	20.4hrs (32 machines)	178hrs (32 machines)	<1 sec (1 machine)	PIII, 900M RAM, 90% and 44%usage of 1x866MHz cpu for positive and negative training respectively
state removal	15.7hrs (32 machines)	22hrs (32 machines)	<1 sec (1 machine)	PIII, 900M RAM, 90%usage of 1x866MHz cpu
HMMER	10mins (1 machine)	2hrs36mins (1 machine)	<1 sec (1 machine)	PIV, 256M RAM, 5.8% and 4.9% usage of 1x1.8GHz cpu for positive and negative training respectively

Table 5.7: Computational time comparisons between the local program and HMMER for allele HLA-B*3501. Each test is done on a single machine.

Chapter 6

Discussion and future work

6.1 Effects of proposed solutions

6.1.1 HLA-A*0201

The AROC values in a certain column of Table 5.1 show the effect of topological reduction for a particular matrix initialization approach. Values across the first column show how topological reduction helps to improve the predictive accuracy for the uniform matrix initialization approach. As the values in this column almost do not change, it implies that the topology reduction does not improve the predictive accuracy for this type of matrix initialization. The second and the third columns show the results for the two heuristic matrix initialization approaches, i.e. the single property grouping and the multiple property grouping. The values in these two columns do show that the predictive accuracy is improved as topologies are reduced. Since state removal give notably higher AROC values than transition removal, state removal is more effective than transition removal for allele HLA-A*0201.

The AROC values in a certain row of Table 5.1 shows the effect of matrix initialization for a particular topology. Values across the first row shows the heuristic matrix initializations do not improve accuracies for a fcHMM. Instead programs incorporating heuristic matrix initializations have lower accuracies. Values across the second row do not show a better performance for heuristic approaches, although one of them, the multiple property grouping has a slightly higher AROC value than the uniform matrix initialization does. Nevertheless, values in the third row do show improvements made by the two heuristic approaches and multiple property grouping works more effectively than single property grouping does.

The possible explanation to these observations is as follows. The uniformly initialized matrices start the fcHMM training at a place where the local maximum is larger than those of the heuristic matrices. However, the place where uniformly initialized matrices started is too far away from the global maximum, but the places where the heuristic matrices started are close to the global maximum. Because of this, topological reduction does not help the model training (with uniform matrix initialization) much in finding higher local maximums or the global maximum, while it does

to the model trainings with heuristic matrix initializations. For heuristic matrix initializations, the state removal approach seems more effective than the transition removal approach. This may be because the state removal reduces the model connectivity more dramatically than the transition removal does and thus the former leads the model trainings to explore larger areas that cover higher local maximums or even the global maximum.

In Table 5.1, the AROC values of programs that incorporate the two heuristic approaches (heuristic matrix initialization and topological reduction) are in the range of 0.858 to 0.991, inclusive. These AROC values indicate that the local program has good (>0.800) to excellent (>0.900) predictive power for HLA-A*0201. One more feature to notice is that among all values in the table, the highest is given by the combination of multiple property grouping and the state removal approach.

6.1.2 HLA-B*3501

Results for allele HLA-B*3501 in Table 5.2 are arranged in the same way as those for allele HLA-A*0201 in Table 5.1. The AROC values in the first column show that a fcHMM has a very poor predictive power and topological reductions improve the situation dramatically. The AROC values in the second column show that transition removal helps to improve predictive power, while state removal does not. The AROC values in the third column show that both transition and state removal improves the performance notably and state removal works better than transition removal. There is one obvious difference between the results for HLA-A*0201 and HLA-B*3501. For HLA-A*0201, transition removal does not improve performance much on a fcHMM, but for HLA-B*3501, it does.

The AROC values in the first row of Table 5.2 show that heuristic matrix initializations dramatically outperform uniform matrix initialization and single property grouping works slightly better than multiple property grouping. However, the second row shows different features than the first row. All matrix initialization approaches got excellent predictive power, but uniform matrix initialization performs the best. The third row shows that the heuristic matrix initializations perform equally or better than uniform matrix initialization and multiple property grouping works far better than single property grouping.

In Table 5.2, the AROC values of programs that incorporate the two heuristic approaches (heuristic matrix initialization and topological reduction) are in the range of 0.854 to 0.986, inclusive. These AROC values indicate that the local program has good to excellent predictive power for HLA-B*3501. Like HLA-A*0201, among all values in the table, the highest is given by the combination of multiple property grouping and the state removal approach.

A possible explanation for the very low AROC value (0.354) is that the uniform matrices put the starting point of the model training at the place where the local maximum is very small, compared

to other local maximums. This leads to a very low AROC value for a fcHMM. The problem of being trapped by local maximums may be fixed by using simulated annealing approach. For a pcHMM derived from state removal or state removal, the model training may get out of the area where it starts, which leads to a different and higher local maximum.

A possible explanation for the observation that transiton removal works better than state removal in some cases in table 5.2 is as follows. Determined by the characteristics of the training data for HLA-B*3501, there are many high local maximums that are close to each other. This feature, in some cases, lets transition removal which does more subtle adjustments to a model's topology to achieve a higher local maximum than state removal does. By comparing the results of HLA-A*0201 and HLA-B*3501, it is found that the state removal is somehow less effective than transiton removal for HLA-B*3501 but more effective for HLA-A*0201. However, for both alleles, the best performance are given by the combination of the multiple property grouping and the state removal. This implies that the multiple property grouping (or the initial matrices) plays an important role in determining the performance.

Taking results from both alleles into considering, the following features are observed:

- Transition removal (a pcHMM) performs better than an fcHMM in all cases. State removal outperform fcHMM in most cases (5 out of 6 cases).
- A fcHMM never has an accuracy higher than 0.900. Transition removal can, in some cases (3 out of 6 cases), achieve excellent predictive accuracy. So does state removal.
- Heuristic matrix initializations (single property and multiple property) improves performance in some cases (4 out of 6 cases).
- For both alleles, the best predictive performance comes from the combination of multiple property grouping and state removal.

In summary, the topological reduction can be used to increase predictive accuracy, while heuristic matrix initializations can be used to increase predictive power in some cases. The most effective approach is to combine multiple property grouping with state removal. This result implies that the heuristic matrix initialization starts model training at a place that is closer to the global maximum than the uniform matrix initialization does. However, there are still chances in which the model are trapped into the local maximums. To get around this problem, the model training needs topological reduction to help it to explore more area and find a high local maximum or the global maximum. Since the state removal reduces the model's topology more dramatically than the transition removal does, it takes the model training out of the starting area more effectively. This is probability the reason why the state removal has higher AROC values than the transition removal does.

One thing to keep in mind is that the testing results of the proposed solutions are obtained from testings on two human MHCs only and the results have already shown some differences between

these two alleles. Hence, the above discussion only give an idea of how well the proposed solutions may work. The effectiveness of these solutions need to be confirmed by testings on more alleles from more species. Another item to consider in this work is the well-known problem of "data overfitting". This means that a program's performance has been overly tuned to fit the current set of training data so that when it is used on another set of data, the performance changes. The main reason that data overfitting may have occurred is that all the topologies derived from topological reduction are tested on the current set of training data and the topology that gives the best performance is selected. Consequently, the topology selected works the best for this set of training data but it may not work as well for another set. This is one major limitation of this program. The ideal solution to this problem is probably to use a diverse training data set for each allele, one that represents the entire population of data for this allele. It is not possible to have the entire population of data, but as more data (for training) become availabe in the future, the training data will be more representative to the whole population. Thus, the situation for data overfitting will be improved.

6.2 A performance comparison with HMMER

6.2.1 Predictive accuracy

The predictive accuracy and computational time of the local program are compared to those of HMMER on allele HLA-A*0201 and allele HLA-B*3501. In terms of predictive accuracy, the AROC curves in Figure 5.9 show that predictive accuracy of the local program is worse than HMMER when the model is fully connected and when the fcHMM is modified with transition removal. However, the local program has better predictive accuracy (AROC value of 0.991) than HMMER (AROC value of 0.956) when the fcHMM is modified with state removal. Unfortunately the local program only achieves an AROC value of 0.948 when a singe property grouping method and state removal is used (see Table 5.1). Therefore, only the combination of multiple property grouping and the topological reduction with state removal allows the local program to exceed the performance of HMMER. This result holds true for allele HLA-B*3501 according to Figure 5.10 and Table 5.2. These data again indicate that the combination of multiple property grouping and the proposed topological reduction is an effective approach to increase predictive accuracy. Sensitivity and specificity calculated from the AROC curves (Tables 5.4 and 5.5) again show that only state removal combined with multiple property grouping performs better than HMMER, for both allele HLA-A*0201 and HLA-B*3501.

6.2.2 Computational time

In terms of computational time, the local program needs far more time than HMMER. The local program takes time in units of hours while HMMER only takes time in units of seconds. The long computational time is another major limitation of the local program. This problem stems from two main issues. First, the time complexity of one model training in the local program is $\#\ of\ training\ cycles * \#\ of\ training\ sequences * L^2 * N^2$, where L is the average length of the training sequences and N is the number of states. In contrast, HMMER is a pHMM which is a linear topology. The time complexity of HMMER will be linear of the number of states. The second reason is that the transition removal involves training 120 such models and state removal involves training 8 such models.

6.3 Future work

Overall the local program gives good predictive accuracy to real biological data and the two proposed heuristic approaches improve performance. Thus the goal of this thesis has been achieved. However, there are features of the local programs that should be tested and some limitations that need to be tackled. Some instances of these are as follows:

- Simulated annealing can be tried to get the model training out of local maximums.
- The predictive accuracy of the local program should be tested with thresholds (to stop model training) lower than 0.008.
- To have a better picture of the performance of the local program, it should be compared to available programs that use different prediction methods.
- This work has only shown the effectiveness of proposed approaches on limited data resources. To get a better grasp on the effectiveness of the local program, various experiments could be performed, including:
  - The performance of the local program could be tested on more alleles to see how consistent the effects of the heuristic approaches observed is across alleles.
  - The performance of the local program could be tested with data that includes all levels of binding affinity instead of just strong binders. This data would be more representative to the population of real-world peptides for a particular allele.
  - The local program could also be tested on a limited amount of data to determine the minimum amount of data required to train the model successfully.

References

[1] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. In *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1999.

[2] M. Hagmann. Computers aid vaccine design. *Science*, **290**:80, 2000.

[3] D. R. Flower and I. A. Doytchinova. Immunoinformatics and the prediction of immunogenicity. *Applied Bioinformatics*, **1**:167, 2002.

[4] A. Townsend and H. Bodmer. Antigen recognition by class I-restricted T lymphocytes. *Annual Review of Immunology*, **7**:601, 1989.

[5] M. T. Heemels, T. N. M. Schuhmacher, K. Wonigeit, and H. L. Ploegh. Peptide translation by variants of the transporter associated with antigen processing. *Science*, **262**:2059, 1993.

[6] F. Momburg, J. C. Roelse, G. W. Howard, G. W. Butcher, G. J. Hammerling, and J. J. Neefjes. Selectivity of MHC encoded peptide transporters from human, mouse and rat. *Nature*, **367**:648, 1994.

[7] A. Sette, A. Vitiello, and B. Reherman. The relationship between class I binding affinity and immunogenicity of potential cytotoxic T cell epitopes. *Journal of Immunology*, **153**:5586, 1994.

[8] V. Brusic, N. Petrovsky, G. Zhang, and V. B. Bajic. Prediction of promiscuous peptides that bind HLA class I molecules. *Immunology and Cell Biology*, **80**:280, 2002.

[9] I. A. Doytchinova and D. R. Flower. Quantitative approaches to computational vaccinology. *Immunology and Cell Biology*, **80**:270, 2002.

[10] K. Yu, N. Petrovsky, C. Schonbach, J. L. Y. Koh, and V. Brusic. Methods for prediction of peptide binding to MHC molecules: a comparative study. *Journal of Molecular Medicine*, **8**:137, 2002.

[11] M. F. Triola. *Elementary Statistics*. Addison-Wesley, 2004.

[12] P. Baldi, S. Brunak, K. Chauvin, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, **16**:412, 2000.

[13] C. K. Matthews, K. E. V. Hold, and K. G. Ahern. *Biochemistry (3rd Edition)*. Benjamin Cummings, 1999.

[14] European Bioinformatics Institute. IMGT/HLA sequence database. http://www.ebi.ac.uk/imgt/hla/, last accessed October 2004.

[15] International Society of Animal Genetics. BoLA Nomenclature. http://www.projects.roslin.ac.uk/bola/bolahome.html, last accessed October 2004.

[16] V. Brusic, G. Rudy, and L. Harrison. MHCPEP-a database of MHC-binding peptides: update 1997. *Nucleic Acids Rsearch*, **26**:368, 1998.

[17] R. M. Chicz, R. G. Urban, J. C. Gorga, D. A. Dignali, W. S. Lane, and J. L. Strominger. Specificity and promiscuity among naturally processed peptides bound to HLA-DR alleles. *Journal of Experimental Medicine*, **178**:27, 1993.

[18] Research Collaboratory for Structural Bioinformatics (RCSB). PDB: protein data bank. http://www.rcsb.org/pdb/, last accessed October 2004.

[19] T. Sturniolo, E. Bono, J. Ding, L. Raddrizzani, O. Tuereci, U. Sahin, M. Braxenthaler, F. Gallazzi, M. P. Protti, F. Sinigaglia, and J. Hammer. Generation of tissue-specific and promiscuous HLA ligand database using DNA microarrays and virtual HLA class II matrices. *National Biotechnology*, **17**:555, 1999.

[20] J. Ruppert, J. Sidney, E. Celis, R. T. Kubo, H. M. Grey, and A. Sette. Prominent role of secondary anchor residues in peptide binding to HLA-A2.1 molecules. *Cell*, **74**:929, 1993.

[21] V. Brusic, G. Rudy, and L. Harrison. MHCPEP: A database of MHC binding peptides. http://wehih.wehi.edu.au/mhcpep, last accessed October 2004.

[22] K. Takahashi, A.P. Rooney, and M. Nei. Origins and divergence times of mammalian class II MHC gene clusters. *Journal of Heredity*, **91**:198, 2000.

[23] X. Gu and M. Nei. Locus specificity of polymorphic alleles and evolution by a birth-and-death process in mammalian MHC genes. *Molecular Biology and Evolution*, **16**:147, 1999.

[24] M. Nei and A. L. Hughes. Balanced polymorphism and evolution by the birth-and-death process in the MHC loci. In *11th Histocompatibility Workshop and Conference*. Oxford University Press, 1992.

[25] M. Nei, X. Gu, and T. Sitnikoya. Evolution by the birth-and death process in mutigene families of the vertebrate immune system. *Proceedings of the National Academy of Sciences of the United States of America*, **94**:7799, 1997.

[26] A. Kiran. Pattern and motif (a. s. kiran). http://bioinformatics.org/pipermail/bio_bulletin_board/2001-December/000411.html, last accessed October 2001.

[27] A. K. Jain, R. P. W. Duin, and J. Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**:4, 2000.

[28] L. Duret and S. Abdeddaim. Tools for Multiple Alignments. http://pbil.univ-lyon1.fr/alignment.html, last accessed October 2004.

[29] DFG-Sonderforschungsbereich 510 and the European Union. SYFPEITHI. http://syfpeithi.bmi-heidelberg.com/, last accessed October 2004.

[30] The Molecular Immunology Foundation. RANKPEP. http://mif.dfci.harvard.edu/Tools/rankpep.html, last accessed October 2004.

[31] Computational Bioscience and Engineering Lab at Center for Information Technology of National Institutes of Health. Bioinformatics & Molecular Analysis Section. http://www-bimas.cit.nih.gov, last accessed October 2004.

[32] EpiVax Inc and TB/HIV laboratory at Brown University. EpiGenomix. http://epigenomix.com/pls/hiv/!www_user.front_end, last accessed October 2004.

[33] J. Hammer and F. Sinigaglia. Vaccinome. http://www.vaccinome.com/pages/600807/index.html, last accessed October 2004.

[34] Bioinformatics Center of Institute of Microbial Technology in India. ProPred: MHC class II binding peptide prediction server. http;//www.imtech.res.in/raghava/propred/, last accessed October 2004.

[35] L. R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of IEEE*, **77**:257, 1989.

[36] A. Krogh, J. Sidney, and S. Southwodd. Hidden Markov models in computational biology. Applications to protein modeling. *Journal of Molecular Biology*, **235**:1501, 1994.

[37] European Bioinformatics Institute. ClustalW. http://www.ebi.ac.uk/clustalw, last accessed October 2004.

[38] UCSC Computational Biology Group. Sequence Alignment and Modeling System. http://www.cse.ucsc.edu/research/compbio/sam.html/, last accessed October 2004.

[39] H. Mamitsuka. A learning method of Hidden Markov Models for sequence discrimination. *Journal of Computational Biology*, **3**:361, 1996.

[40] H. Mamitsuka. Predicting peptides that bind to MHC molecules using supervised learning of Hidden Markvo models. *Proteins*, **33**:460, 1998.

[41] K. Udaka, H. Mamitsuka, Y. Nakaseko, and N. Abe. Prediction of MHC class I binding peptides by a query learning algorithm based on Hidden Markov Models. *Journal of Biological Physics*, **28**:183, 2002.

[42] Center for Biological Sequence Analysis in Denmark. NetMHC 2.0 Server. http://www.cbs.dtu.dk/services/NetMHC, last accessed October 2004.

[43] V. Brusic, J. L. Y. Koh, G. Zhang, and T. Kandasamy of Computational Immunology Group in Sigapore. PREDICT: Prediction of MHC binidng sites. http://sdmc.lit.org.sg:8080/predict-demo, last accessed October 2004.

[44] R. Beale and T. Jackson. *Neural Computing: An Introduction*. Adam Hilger, Bristol, 1990.

[45] V. Brusic, G. Rudy, M. Honeyman, J. Hammer, and L. Harrison. Prediction of MHC class II-binding peptides using an evolutionary algorithm and artificial neural network. *Bioinformatics*, **14**:121, 1998.

[46] G. P. S. Raghava and M. Bhasin of nHLAPred team in India. ANNPred: a neural network based MHC class I binding peptide prediction server. http://www.imtech.res.in/raghava/nhlapred/neural.html, last accessed October 2004.

[47] G. P. S. Raghava and M. Bhasin of nHLAPred team in India. ComPred: a neural network based MHC class I binding peptide prediction server. http://www.imtech.res.in/raghava/nhlapred/comp.html, last accessed October 2004.

[48] P. Donnes and A. Elofsson. Prediction of mhc class I binding peptides,using SVMHC. *BMC Bioinformatics*, **3**:25, 2002.

[49] D. H. Freemont, M. Matsumura, E. A. Stura, P. A. Peterson, and L. A. Wilson. Crystal structure of two viral peptides in complex with murine MHC class I H-2Kb. *Science*, **257**:919, 1992.

[50] M. L. Silver, H. C. Guo, J. L. Strominger, and D. C. Wiley. Atomic structure of a human MHC molecule presenting an influenza virus peptide. *Nature*, **360**:367, 1992.

[51] W. Zhang, A. C. M. Young, M. Imarai, S. G. Nathenson, and J. C. Sacchettini. Crystal structure of the major histocompatibility complex class I H-2Kb molecule containing and T-cell receptor recognition. *PNAS*, **189**:8403, 1992.

[52] S. F. Ora, Y. Altuvia, A. Sette, and H. Margalit. Structure-based prediction of binding peptides to mhc class I molecules: Application to a broad range of MHC alleles. *Protein Science*, **9**:1838, 2000.

[53] The Hebrew University of Jerusalem. PREDEP: MHC class I epitope prediction. http://bioinfo.md.huji.ac.il/marg/Teppred/mhc-bind/, last accessed October 2004.

[54] G. Vanagas. Receiver operating characteristic curves and comparison of cardiac surgery rish stratification systems. *Interactive Cardiovascular and Thoracic Surgery*, **3**:319, 2004.

[55] J. A. Hanley and B. J. McNeil. The meaning and use of the area under the Receiver Operating Characteristic (ROC) curve. *Radiology*, **143**:29, 1982.

[56] M. Zweig and G. Campbell. Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical Chemistry*, **39**:561, 1993.

[57] Improved Outcomes software Inc. Predictive Patterns Software: Glossary of Terms/Acronym List. http://www.predictivepatterns.com/docs/WebSiteDocs/Glossary/Glossary_of_Terms_Acronym_List.htm#M_Index, last accessed October 2004.

[58] Washington University in St. Louis in U. S. A. HMMER: profile HMMs for protein sequence analysis. http://hmmer.wustl.edu, last accessed October 2004.

[59] Y. Miyata. A user's guide to PlaNet version 5.6. Computer Science Dept., University of Colorado, Boulders, 1991, last accessed October 2004.

[60] G. P. S. Raghava and M. Bhasin of nHLApred team in india. nHLAPred: Help Document. http://www.imtech.res.in/raghava/nhlapred/help.html, last accessed October 2004.

[61] G. P. S. Raghava. MHCBN version 3.1: comprehensive database of MHC binding and non-binding peptides. http://www.imtech.res.in/raghava/mhcbn, last accessed October 2004.

[62] K. C. Parker, M. A. Bednarek, and J. E. Coligan. Scheme for ranking potential HLA-A2 binding peptides based on independent binding of individual peptide side-chains. *Journal of Immunology*, **152**:163, 1994.

[63] T. Hubbard et al. The ENSEMBL genome database project. *Nucleic Acids Research*, **30**:38, 2002.

[64] W. M. Kast, R. M. P. Brandt, J. Sidney, J. Drijfhout, R. T. Kubo, H. M. Grey, C. J. M. Melief, and A. Sette. Role of HLA-A motifs in identification of potential CTL epitopes in human Papillomavirus type 16 E6 and E7 proteins. *Journal of Immunology*, **152**(8):3904, April 2000.

[65] D. D. Wackerly, W. Mendenhall, and R. L. Scheaffer. *Mathematical Statistics With Applications*. PWS Publishing, 1997.

Appendix A.1: The single-letter amino acid code

G	Glycine	P	Proline	A	Alanine
V	Valine	L	Leucine	I	Isoleucine
M	Methionine	C	Cysteine	F	Phenylalanine
Y	Tyrosine	W	Tryptophan	H	Tistidine
K	Lysine	R	Arginine	Q	Glutamine
N	Asparagine	E	Glutamic Acid	D	Aspartic Acid
S	Serine	T	Threonine		

For each emission matrix in the appendices, the upper-case letters in the first column are amino acids and integers in the first row are states. For each transition matrix in the appendices, the integers in the first row are "from" state and the first column are "to" states.

Appendix B.1: Uniformly initialized emission and transition matrices for HLA-A*0201

>emission matrix

	1	2	3	4	5	6	7	8	9	10	11	12
A	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
C	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
D	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
E	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
F	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
G	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
H	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
I	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
K	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
L	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
M	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
N	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
P	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
Q	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
R	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0

S	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
T	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
V	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
W	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0
Y	0.0	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.0

>transition matrix

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.12	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
3	0.08	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
4	0.12	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
5	0.08	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
6	0.12	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
7	0.08	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
8	0.12	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
9	0.08	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
10	0.12	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
11	0.08	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0
12	0.0	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	0.0909	1.0

Appendix B.2: Initial emission and transition matrices derived from single property grouping for HLA-A*0201

>emission matrix

	1	2	3	4	5	6	7	8	9	10
Y	0.0	0.0059	0.225	0.075	0.0067	0.0059	0.075	0.0067	0.0091	0.0
W	0.0	0.0059	0.225	0.075	0.0067	0.0059	0.075	0.0067	0.0091	0.0
V	0.0	0.3	0.0062	0.075	0.0067	0.0059	0.0125	0.0067	0.1	0.0
T	0.0	0.0059	0.0062	0.0125	0.0067	0.0059	0.075	0.18	0.1	0.0
S	0.0	0.0059	0.0062	0.0125	0.0067	0.0059	0.075	0.18	0.1	0.0
R	0.0	0.0059	0.0062	0.0125	0.18	0.3	0.075	0.0067	0.0091	0.0
Q	0.0	0.0059	0.0062	0.0125	0.0067	0.0059	0.075	0.0067	0.0091	0.0

P	0.0	0.0059	0.0062	0.0125	0.0067	0.0059	0.0125	0.0067	0.1	0.0
N	0.0	0.0059	0.0062	0.0125	0.0067	0.0059	0.075	0.0067	0.1	0.0
M	0.0	0.0059	0.0062	0.075	0.0067	0.0059	0.0125	0.0067	0.0091	0.0
L	0.0	0.3	0.0062	0.075	0.0067	0.0059	0.0125	0.0067	0.0091	0.0
K	0.0	0.0059	0.0062	0.075	0.18	0.3	0.075	0.0067	0.0091	0.0
I	0.0	0.3	0.0062	0.075	0.0067	0.0059	0.0125	0.0067	0.0091	0.0
H	0.0	0.0059	0.225	0.075	0.18	0.3	0.075	0.0067	0.0091	0.0
G	0.0	0.0059	0.0062	0.075	0.0067	0.0059	0.0125	0.18	0.1	0.0
F	0.0	0.0059	0.225	0.075	0.0067	0.0059	0.0125	0.0067	0.0091	0.0
E	0.0	0.0059	0.0062	0.0125	0.18	0.0059	0.075	0.0067	0.0091	0.0
D	0.0	0.0059	0.0062	0.0125	0.18	0.0059	0.075	0.0067	0.1	0.0
C	0.0	0.0059	0.0062	0.075	0.0067	0.0059	0.075	0.18	0.1	0.0
A	0.0	0.0059	0.0062	0.075	0.0067	0.0059	0.075	0.0067	0.0091	0.0

>transition matrix – derived from 116 positive training data shown in appendix 6.3

	1	2	3	4	5	6	7	8	9	10
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0877	0.1257	0.1557	0.1378	0.1185	0.1375	0.1357	0.1636	0.1394	0.0
3	0.1039	0.0443	0.0560	0.0495	0.0450	0.0333	0.0434	0.0574	0.0539	0.0
4	0.3019	0.2527	0.2847	0.2649	0.2464	0.2667	0.2561	0.2755	0.2490	0.0
5	0.0519	0.0395	0.0511	0.0512	0.0711	0.075	0.0619	0.0646	0.0598	0.0
6	0.0487	0.0263	0.0219	0.0298	0.0355	0.0333	0.0322	0.0373	0.0324	0.0
7	0.1656	0.1269	0.1411	0.1389	0.1919	0.175	0.1747	0.1492	0.1544	0.0
8	0.1104	0.1066	0.0925	0.0962	0.1019	0.1	0.0891	0.0646	0.0830	0.0
9	0.1299	0.1581	0.1898	0.1687	0.1896	0.1792	0.1900	0.1736	0.1734	0.0
10	0.0	0.1198	0.0073	0.0630	0.0	0.0	0.0068	0.0143	0.0548	1.0

>transition matrix – derived from 297 negative training data shown in appendix 6.3

	1	2	3	4	5	6	7	8	9	10
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0705	0.0702	0.0436	0.0693	0.0807	0.0555	0.0774	0.0863	0.0912	0.0
3	0.0457	0.0543	0.0436	0.0464	0.0470	0.0593	0.0339	0.0357	0.0391	0.0
4	0.2010	0.2026	0.1783	0.2019	0.2073	0.1985	0.1943	0.1957	0.2042	0.0
5	0.1188	0.1056	0.0973	0.0970	0.1047	0.1025	0.1047	0.0982	0.0966	0.0

6	0.0601	0.0586	0.0474	0.0557	0.0566	0.0611	0.0572	0.0553	0.0513	0.0
7	0.2389	0.2344	0.2544	0.2287	0.2206	0.2305	0.2243	0.2118	0.2161	0.0
8	0.0953	0.0803	0.1147	0.0897	0.0892	0.0931	0.1015	0.0993	0.0987	0.0
9	0.1697	0.1418	0.1833	0.1654	0.1517	0.1590	0.1636	0.1731	0.1594	0.0
10	0.0	0.0521	0.0374	0.0461	0.0422	0.0405	0.0430	0.0446	0.0434	1.0

Appendix B.3: Initial emission and transition matrices derived from multiple property grouping for HLA-A*0201

>emission matrix

	1	2	3	4	5	6	7	8	9	10	11	12
Y	0.0	0.0059	0.0062	0.0067	0.0067	0.0067	0.18	0.225	0.0053	0.0053	0.0053	0.0
W	0.0	0.0059	0.0062	0.0067	0.0067	0.0067	0.18	0.225	0.0053	0.0053	0.0053	0.0
V	0.0	0.3	0.225	0.0067	0.0067	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
T	0.0	0.0059	0.0062	0.18	0.18	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
S	0.0	0.0059	0.0062	0.18	0.18	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
R	0.0	0.0059	0.0062	0.0067	0.0067	0.18	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
Q	0.0	0.0059	0.0062	0.0067	0.0067	0.0067	0.0067	0.0062	0.0053	0.9	0.0053	0.0
P	0.0	0.0059	0.0062	0.0067	0.0067	0.0067	0.0067	0.0062	0.0053	0.0053	0.9	0.0
N	0.0	0.0059	0.0062	0.0067	0.18	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
M	0.0	0.0059	0.0062	0.0067	0.0067	0.0067	0.0067	0.0062	0.9	0.0053	0.0053	0.0
L	0.0	0.3	0.0062	0.0067	0.0067	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
K	0.0	0.0059	0.0062	0.0067	0.0067	0.18	0.18	0.0062	0.0053	0.0053	0.0053	0.0
I	0.0	0.3	0.0062	0.0067	0.0067	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
H	0.0	0.0059	0.0062	0.0067	0.0067	0.18	0.18	0.0067	0.0053	0.0053	0.0053	0.0
G	0.0	0.0059	0.225	0.18	0.0067	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
F	0.0	0.0059	0.0062	0.0067	0.0067	0.0067	0.0067	0.0067	0.0053	0.0053	0.0053	0.0
E	0.0	0.0059	0.0062	0.0067	0.0067	0.18	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
D	0.0	0.0059	0.0062	0.0067	0.18	0.18	0.0067	0.0062	0.0053	0.0053	0.0053	0.0
C	0.0	0.0059	0.225	0.18	0.18	0.0067	0.18	0.0062	0.0053	0.0053	0.0053	0.0
A	0.0	0.0059	0.225	0.18	0.0067	0.0067	0.0067	0.0062	0.0053	0.0053	0.0053	0.0

>transition matrix – developed from 116 positive training data in appendix 6.3

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.1452	0.1895	0.1994	0.2670	0.2448	0.1953	0.2361	0.2490	0.1667	0.1864	0.2353	0.0
3	0.1398	0.1264	0.1108	0.1522	0.1463	0.1445	0.1545	0.1634	0.1190	0.1525	0.1412	0.0
4	0.1828	0.1607	0.1191	0.1054	0.1194	0.1680	0.1545	0.1479	0.1190	0.1356	0.2000	0.0
5	0.0806	0.0939	0.0720	0.0937	0.1224	0.1563	0.1116	0.1128	0.1905	0.1186	0.1882	0.0
6	0.0860	0.0596	0.0859	0.1054	0.1045	0.1172	0.0901	0.0817	0.1190	0.1186	0.0824	0.0
7	0.1559	0.0650	0.0748	0.0773	0.0836	0.0898	0.0901	0.0739	0.0714	0.1017	0.0	0.0
8	0.1720	0.0668	0.0886	0.0937	0.0925	0.0742	0.0773	0.0895	0.0714	0.1186	0.0588	0.0
9	0.0161	0.0144	0.0028	0.0094	0.0119	0.0195	0.0172	0.0156	0.0476	0.0169	0.0	0.0
10	0.0161	0.0144	0.0194	0.0304	0.0328	0.0195	0.0258	0.0117	0.0	0.0	0.0471	0.0
11	0.0054	0.0289	0.0499	0.0422	0.0328	0.0156	0.0258	0.0428	0.0714	0.0169	0.0471	0.0
12	0.0	0.1805	0.1773	0.0234	0.0090	0.0	0.0172	0.0117	0.0238	0.0339	0.0	1.000

>transition matrix – developed from 297 negative training data in appendix 6.3

	1	2	3	4	5	6	7	8	9	10	11	12
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.1107	0.1046	0.1010	0.1344	0.1626	0.1240	0.0961	0.0647	0.1690	0.1315	0.15	0.0
3	0.1086	0.1143	0.1059	0.1029	0.0934	0.0993	0.1212	0.1312	0.1972	0.0516	0.0563	0.0
4	0.1496	0.1197	0.1256	0.1548	0.1609	0.1371	0.1390	0.1701	0.2535	0.1315	0.1313	0.0
5	0.1660	0.1489	0.1626	0.1640	0.1592	0.13629	0.1787	0.2015	0.1268	0.1127	0.1313	0.0
6	0.1865	0.1575	0.1290	0.1529	0.1540	0.1609	0.1369	0.1442	0.0423	0.2207	0.2375	0.0
7	0.1291	0.1499	0.1355	0.0945	0.1056	0.1289	0.1317	0.1349	0.0423	0.1080	0.0625	0.0
8	0.0717	0.0809	0.0837	0.0556	0.0407	0.0723	0.0439	0.0647	0.1268	0.0423	0.0625	0.0
9	0.0123	0.0097	0.0209	0.0158	0.0078	0.0033	0.0188	0.0129	0.0	0.0	0.0	0.0
10	0.0348	0.0270	0.0382	0.0250	0.0156	0.0369	0.0376	0.0185	0.0	0.0845	0.0875	0.0
11	0.0307	0.0097	0.0320	0.0306	0.0329	0.0361	0.0324	0.0019	0.0	0.0329	0.0063	0.0
12	0.0	0.0777	0.0665	0.0695	0.0675	0.0649	0.0637	0.0555	0.0423	0.0845	0.075	1.0

Appendix C: Training and testing data

Appendix C.1: Data for HLA-A*0201

This section shows the positive (named "binders") and negative (named "non-binders") training and testing data used in this work. The data for HLA-A*0201 is listed first followed by data for

HLA-B*3501.

binders

The following 116 binders are extracted from MHCPEP [21]. They are labeled as peptides that have a high level of binding affinity to HLA-A*0201 and have T-cell activity.

GILGFVFTL	LLFGYPVYV	FIAGNSAYEYV	GLLGFVFTL	FLPSDYFPSV
LLMGTLGIV	TLGIVCPI	YMLDLQPETT	WLSLLVPFV	YLVAYQATV
LLGRNSFEV	SLYADSPSV	GLSRYVARL	FLPSDFFPSV	ILCWGELMTL
KLWVTVYYGV	FLYCYFALV	LLPENNVLSPL	RMPEAAPPV	ALNKMFCQL
GLAPPQHLIRV	LLTEVETYVL	SLLTEVETYVL	QMVTTTNPL	
QTTKHKWEAAHVAEQWRAYL		GPEYWDGETRNMKA		
WDRETQICKAKAQTDRENLRIALRY		WDRETQISKTNTQTYRESLRNLRGY		
WDRETQISKTNTQTYREDLRTLLRY		KRWILGLNKIVRMYC		
SAAFEDLRVLSFIRG		KGILGFVFKLTV	FLPNDFFPSV	
FLPADFFPSV	FLPVDFFPSV	MLLSVPLLL	YMNGTMSQV	
KLVALGINAV	KTWGQYWQV	VLYRYGSFSV	YLNKIQNSL	
YIGEVLVSV	GILTVILGV	TVILGVLLL	ALMDKSLHV	RVIEVLQRA
LLNATDIAV	WLWYIKIFI	CLGGLLTMV	LLFNILGGWV	ALMPLYACI
FLLLADARV	LLGRDSFEV	SLYNTVATL	FLPSDCFPSV	SLMAFTAAV
FLPSDDFPSV	GILGFVKTLTV		NLYNELEMNYYGKQENWYSL	
MMRKLAILSV	YGKQENWYSL	YLKKIKNSL	YLKKIQNSL	SLKKNSRSL
SLNFLGGTTV	FLWGPRALV	KIFGSLAFL	CLTSTVQLV	RILGAVAKV
RGPGRAFVAI	WTDQVPFSV	ITSQVPFSV	YMDGTMSQV	ALWGFFPVL
ILTVILGV	LMDKSLHV	AAGIGILT	ILGVLLLI	FLIFFDLFLV
VLAGLLGNV	GLIMVLSFL	KILSVFFLA	GLLGNVSTV	VLPEKDSWTV
EILKEPVHGV	KLAEYVAKV	RTQDENPVV	SLSRFSWGA	SAHKGFKGV
GTLSKIFKL	LMLSPDDIEQ	QLAKTCPVQ	GLAPPQHLI	GLAPPQHEI
LLGRNSFEM	PLDGEYFTL	FLLSLGIHL	GLYSSTVPV	HLYSHPIIL
TLIDVCPI	LLAQFTSAI	YLVSFGVWI	LLVPFVQWFV	GLLGWSPQA
FLLTRILTI	MMWYWGPSL	ILLLCLIFLL	KLHLYSHPI	GLSPTVWLSV
LLVLQAGFFL	LLLCLIFLL	ILSPFMPLL	VLLDYQGML	LLPIFFCLWV
YLEPGPVTA	IVGAETFYV			

non-binders

The following 28 peptides are extracted from MHCBN [61]. These peptides do not have T-cell activity which usually implies that they do not bind to HLA-A*0201.

AGNSAYEYV	ALEAQQEAL	FIAGNSAYE	IAGNSAYEY	IFIAGNSAY
ILESLFRAV	KIFIAGNSA	LKIFIAGNS	FRYNGLIHR	LVMAPRTVL
SRYWAIRTR	YLKDQQLL	AASKERGSVSL	APRASRPSL	APRTVLVYLL
ARLYGIRAK	KRYEGLTQR	PSLKIFIAGNS	DCKTILKAL	GEIYKRWII
APRTVALTA	PSLKIFIAG	PSLKIFIAGNSAYE	SLKIFIAGN	
VITKKVADL	KRWIILGLNK	RRYQKSTEL	ELRSRYWAI	

The following 32 peptides are extracted from MHCBN [61]. They do not bind to HLA-A*0201 but their T-cell activities are unknown

KAMVQAWTFT	FDCLPLGVL	MTKKRKVDGL	SIEDLEVTC	SIEDLEVTCQ
DLEVTCTWKL	VTCTWKLPT	ELGRPSMVWL	DIKMILKMV	DLFLKEGAC
DLVFDECGI	DLVFDECGIT	ELFPPLFMA	ETFKAVLDGL	ETLSITNCRL
GITDDQLLA	GVLMKGQHL	KMILKMVQL	KVKRKKNVL	LVELAGQSL
LVFDECGIT	NLRRLLLSHI	PLQALLERA	QTLKAMVQA	RTFYDPEPIL
SLSHCSQLTT	STEAEQPFI	SVWTSPRRLV	VLRLCCKKL	WLSANPCPHC
WTSPRRLVEL				
WTVWSGNRA				

The following 237 nonamers are derived from the HPV 16 E6 and E7 proteins [64] by sliding a 9mer window along the protein sequence. According to the literature, these peptides do not to bind to HLA-A*0201 or HLA-B*3501. Thus these peptides are also used as negative training sequences for HLA-B*3501.

DGNPYAVCD	YRDGNPYAV	EQLNDSSEE	LCDLLIRCI	TLEQQYNKP
TTDLYCYEQ	TFCCKCDST	ILECVYCKQ	CDSTLRLCV	HYNIVTFCC
DLQPETTDL	PLCDLLIRC	LDLQPETTD	CMSCCRSSR	TLEDLLMGT
WTGRCMSCC	TPTLHEYML	LRLCVQSTH	DFAFRDLCI	DEIDGPAGQ
LDKKQRFHN	CDLLIRCIN	DGPAGQAEP	PAGQAEPDR	YCYEQLNDS
LLRREVYDF	YEQLNDSSE	NDSSEEEDE	PEKQRHLDK	QRHLDKKQR
EYRHYCYSL	DSSEEEDEI	QLCTELQTT	QRFHNIRGR	RCINCQKPL
LCTELQTTI	EQQYNKPLC	KCDSTLRLC	RDGNPYAVC	NCQKPLCPE

YCYSLYGTT	HVDIRTLED	CCRSSRTRR	LGIVCPICS	SLYGTTLEQ
YSLYGTTLE	EDEIDGPAG	RTLEDLLMG	GNPYAVCDK	KCLKFYSKI
MHGDTPTLH	SRTRRETQL	RHLDKKQRF	VYRDGNPYA	PRKLPQLCT
VYCKQQLLR	FRDLCIVYR	CCKCDSTLR	GRCMSCCRS	STLRLCVQS
QLLRREVYD	KQQLLRREV	SSRTRRETQ	KKQRFHNIR	HDIILECVY
ERPRKLPQL	SEYRHYCYS	FCCKCDSTL	DPQERPRKL	SSEEEDEID
DRAHYNIVT	KQRHLDKKQ	SKISEYRHY	AHYNIVTFC	EDLLMGTLG
LPQLCTELQ	IRTLEDLLM	RHYCYSLYG	QAEPDRAHY	LKFYSKISE
DIRTLEDLL	YCKQQLLRR	KRTAMFQDP	VCDKCLKFY	RFHNIRGRW
HYCYSLYGT	RCMSCCRSS	GPAGQAEPD	PTLHEYMLD	EYMLDLQPE
KPLCPEKQR	KFYSKISEY	YAVCDKCLK	NIVTFCCKC	INCQKPLCP
LYGTTLEQQ	YSKISEYRH	RSSRTRRET	YNKPLCDLL	KPLCDLLIR
AVCDKCLKF	REVYDFAFR	DLYCYEQLN	GTTLEQQYN	QPETTDLYC
QSTHVDIRT	TDLYCYEQL	LCPEKQRHL	RREVYDFAF	VTFCCKCDS
LEQQYNKPL	MGTLGIVCP	CRSSRTRRE	MSCCRSSRT	AEPDRAHYN
LHEYMLDLQ	NKPLCDLLI	IVYRDGNPY	LCIVYRDGN	RLCVQSTHV
LIRCINCQK	HQKRTAMFQ	EPDRAHYNI	TLRLCVQST	RWTGRCMSC
CPEKQRHLD	DKCLKFYSK	PDRAHYNIV	LMGTLGIVC	GDTPTLHEY
IILECVYCK	CLKFYSKIS	KQRFHNIRG	IDGPAGQAE	IVTFCCKCD
YGTTLEQQY	PYAVCDKCL	CINCQKPLC	CTELQTTIH	EKQRHLDKK
SCCRSSRTR	TTLEQQYNK	RPRKLPQLC	PQERPRKLP	QDPQERPRK
CKCDSTLRL	LLIRCINCQ	CDKCLKFYS	LQPETTDLY	ISEYRHYCY
RDLCIVYRD	LEDLLMGTL	IVCPICSQK	RAHYNIVTF	CVQSTHVDI
TELQTTIHD	NPYAVCDKC	LECVYCKQQ	THVDIRTLE	AFRDLCIVY
YDFAFRDLC	GIVCPICSQ	TAMFQDPQE	QERPRKLPQ	TGRCMSCCR
CQKPLCPEK	IRGRWTGRC	GRWTGRCMS	RKLPQLCTE	TTIHDIILE
VCPICSQKP	QLNDSSEEE	STHVDIRTL	VYDFAFRDL	PLCPEKQRH
YRHYCYSLY	CIVYRDGNP	DLLMGTLGI	QQYNKPLCD	LNDSSEEED
CVYCKQQLL	YNIVTFCCK	QKRTAMFQD	QTTIHDIIL	EVYDFAFRD
ETTDLYCYE	VQSTHVDIR	IRCINCQKP	MHQKRTAMF	QKPLCPEKQ
FHNIRGRWT	IHDIILECV	DKKQRFHNI	PQLCTELQT	QYNKPLCDL
CYSLYGTTL	LCVQSTHVD	NIRGRWTGR	GQAEPDRAH	CYEQLNDSS
RTAMFQDPQ	LYCYEQLND	EIDGPAGQA	HGDTPTLHE	MFQDPQERP
CKQQLLRRE	DTPTLHEYM	HEYMLDLQP	VDIRTLEDL	DLCIVYRDG
EEDEIDGPA	FQDPQERPR	DLLIRCINC	HLDKKQRFH	AGQAEPDRA
PETTDLYCY	FYSKISEYR	DSTLRLCVQ	QQLLRREVY	HNIRGRWTG

ELQTTIHDI	DIILECVYC	KISEYRHYC	LRREVYDFA	ECVYCKQQL
SEEEDEIDG	RGRWTGRCM	EEEDEIDGP	TIHDIILEC	CSQKPMHQK
KPMHQKRTA	PICSQKPMH	QKPMHQKRT	SQKPMHQKR	CPICSQKPM
PMHQKRTAM	ICSQKPMHQ			

Appendix C.2: Data for HLA-B*3501

binders

The following 70 binders are extracted from MHCPEP [21]. They are labeled as peptides that have a high level of binding affinity to HLA-B*3501 and have T-cell activity.

VPVMPRVPL	FPVTPRVPL	LPLDLSVIL	LPITCEYLL	HPLTNNLPL
LPLQIPIPL	VPVSETVEL	TPVQVQMCL	LPGFDFTPGY	SPIETVPVKL
EPVPLQLPPL	IPLGDAKLVI	EPIDKEIYPL	YPIVQNLQGQM	FPFDENGNVY
VPVWKEATTTL	RPIVSTQLLL	TPPLVKLWYQL	LPINALSNSL	IPVESMETTM
YPCTVNFTIF	YPGHVSGHRM	LPYIEQGMQL	NPAIASLMAF	LPVCQDHLEF
RPLLREDVTF	SPLTTQNTLLF	SPDADLIEANL	YPWPLYGNEGL	TPIPAASQLDL
TPTFVHLQATL	SPLTPELNEIL	QPLTSPTTSQL	QPISHEEQPRY	QPLGTQDQSLY
VPIKDIRNLY	CPNLVSYSSY	IPKKRKRVPY	FPVRPQVPL	FPVRPQVPM
FPVRPQVPF	FPVRPQVPA	FPVRPQVPW	FPVRPQVPY	FPASFFIKL
IPIPSSWAF	LPVFTWLAL	FPFVLAAII	FPHCLAFSY	LPWHRLFLL
FPHCLAFSI	FPIPSSWAI	FPGCSFSIF	FPISHLYIL	FPFCLAFSY
MPFAGLLI	EPIVGAETF	IPAETGQETAY	TPGPGIRY	LPCRIKQII
YPLTSLRSL	DSCMGLIY	NSCMGLIY	APCMGLIY	ASCIGLIY
YPNQNLQGQM	FPFDENGNPVY	MPLETCILAI	TPYAGEPAPF	FPHCLAFSYM

non-binders

The 237 non-binders for HLA-B*3501 are the same 237 non-binders for HLA-A*0201 listed above.