

Master's Thesis Project

Prediction of MHC Class I Binding Peptides, Using a Machine Learning Approach

Pierre Dönnès

LiTH-IFM-Ex 1039

Abstract

T-cells are the most important cells in regulating a specific immune response. Activation of cytotoxic T-cells requires recognition of specific peptides bound to Major Histocompatibility Complex (MHC) class I molecules. MHC-peptide complexes are potential tools for diagnosis and treatment of pathogens and cancer, as well as for the development of peptide vaccines.

This Master's Thesis investigates the usage of Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) for MHC-binding peptide predictions. The binding data used for SVMs and ANNs training are extracted from the public database MHCPEP.

SVMs showed in preliminary studies to have better performance than ANNs. SVMs are therefore applied to a large number of MHC types. SVM predictions are benchmarked with the public MHC-peptide prediction tool SYFPEITHI. On the test set used SVMs predicts 95% correct compared to 91% for SYFPEITHI.

The SVM approach for predicting MHC-binding peptides shows a higher predictive performance than SYFPEITHI. Limited studies also show that SVMs might be superior to ANNs for this type of prediction. SVMs are also easy to apply to a large number of MHC types.

Contents

1	Introduction	5
2	Background and theory	7
2.1	The immune system	7
2.1.1	Innate and acquired immunity	8
2.1.2	Cells involved in acquired immunity and specific recognition	8
2.1.3	Clonal selection theory and tolerance	10
2.1.4	The Major Histocompatibility Complex (MHC)	10
2.2	MHC peptide databases	13
2.2.1	Peptide purification and sequencing	13
2.2.2	The MHCPEP database	14
2.3	Methods for predicting MHC binding peptides	14
2.3.1	Sequence based predictions	14
2.3.2	Structure based predictions	15
2.4	Pattern recognition (classification) and machine learning	16
2.4.1	Separable patterns	17
2.4.2	Machine learning	17
2.5	Support Vector Machines (SVMs)	19
2.5.1	Training of SVMs	20
2.6	Artificial Neural Networks (ANNs)	20
2.6.1	Training of ANNs	24
2.7	Peptide data representation	24
2.8	Cross-validation	25
2.9	Performance measurements	27
3	Materials and Methods	28
3.1	Peptide data extraction	28
3.2	Peptide data representation	29
3.3	SVM ^{light}	29
3.3.1	SVM_learn and SVM_classify	29
3.3.2	SVM ^{light} implementation	30
3.4	Netlab	30
3.4.1	Netlab implementation	30
3.5	SVM vs ANN	30
3.6	Benchmarking against SYFPEITHI	31
3.7	A public prediction tool	31
4	Results and Discussion	32
4.1	SVM vs. ANN	32
4.2	Amount of data needed for SVM training	32
4.3	Results from all the MHC types used to train on	33
4.4	Benchmarking against SYFPEITHI	34
4.4.1	Overall prediction	36
4.4.2	SVMs vs. SYFPEITHI comparison for each MHC type	37

4.5	The public prediction tool	37
4.5.1	Test with a bacterial protein	37
5	Conclusions and future improvements	40
6	References	41
7	Acknowledgements	43

1 Introduction

The immune system is a defense system that is present in vertebrates to protect them from invading pathogens and cancer [1]. The protection against foreign molecules (antigens) can be both specific and non-specific. Non-specific immunity includes, for example, skin (as a physical barrier), mucous membranes and inflammation. Specific immunity allows the organism to specifically recognize and selectively eliminate antigens. There are four types of molecules present in the immune system responsible for the specific recognition: antibodies produced by B-cells, T-cell receptors (TCRs), MHC class I molecules on all nucleated cells and MHC class II molecules on antigen presenting cells. MHC is an abbreviation for the Major Histocompatibility Complex and this Master's Thesis deals with the development of a prediction method to discern which peptides associate with MHC class I molecules.

There is a natural turnover of proteins in living cells, which means that they are broken into smaller peptide fragments. Some of these peptides bind to MHC molecules and travel to the cell surface, where the MHC-peptide complex can be recognized by TCRs on T-cells [2]. T-cells are the most important cell-type in regulating an immune response and can distinguish between self and non-self peptides (i.e. fragments of proteins from pathogens or cancer cells). Section 2.1.3 explains how T-cells learn to distinguish between self/non-self. The presentation of MHC-peptide complexes is a way to monitor what is going on in the body. If there is no foreign antigen in the body, MHC presents only self-peptides and hence there is no activation of T-cells. On the other hand, if the peptide presented by MHC is, for example, a viral protein, T-cells can be activated. Figure 1 shows a summary of presentation of MHC-peptide complexes and the activation of T-cells.

MHC-peptide complexes recognized by T-cells are potential tools for the diagnosis and control of pathogens and cancer [3]. One example is the development of a vaccine against malaria. The peptides that will be presented by MHC during a malaria infection can be made in vitro and introduced to the host. This would lead to a high level immunity against malaria. One problem is to find out which peptide that will bind to the MHC. Suggestions have been made that only one in 100-200 candidate peptides actually binds to MHC [4]. In real cases the protein sequence is usually all that is known. A protein of length 350 amino acids gives 342 possible binding peptides, if the peptide length is 9 amino acids. Synthesizing that many peptides is both expensive and time-consuming. A reliable prediction method that reduces the number of candidate binders is therefore useful.

The genetic loci for MHC are highly polymorphic, leading to many alternate forms of genes (alleles). Peptide binding to MHC molecules is allele specific, meaning that each MHC allele binds to a certain type of peptides. Studies of peptides with the ability to bind a certain MHC type, have shown that there are certain requirements for binding. For example a lysine might be required in position N+1 in the peptide (one amino acid away from the amino terminus), and a valine is needed in position N+8. Amino acids in certain positions like

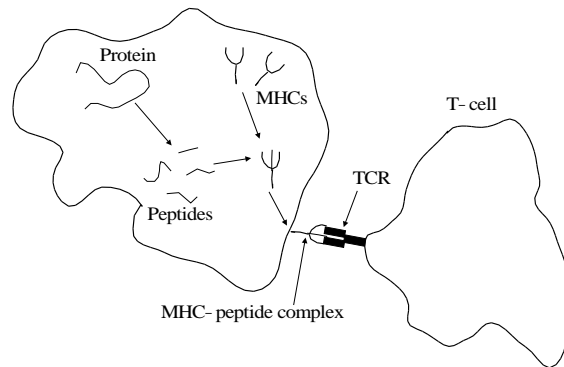


Figure 1: Principles for MHC-peptide presentation to a T-cell. The interaction between MHC-peptide complexes and the T-cell receptor (TCR) is necessary for the activation of T-cells.

this are called anchor residues. Simple sequence motifs of anchor residues like this can sometimes be seen in the peptide data and this can be used to predict binding peptides [2]. However, other position specific residues are usually also important for binding and these are called secondary anchor residues. This information has also been used for prediction of MHC binding peptides, as in the scoring matrix approach explained below.

The two main approaches to predict MHC binding peptides are either sequence based or structure based.

An example of a sequence based prediction method is to create a scoring matrix for a certain MHC type. This can be done by looking at frequencies of different amino acids in different positions of the peptide (i.e. looking for anchor residues). For example, one might have peptide sequences with a length of 9 amino acids giving a 9 x 20 matrix. Each column in the matrix represents a position in the sequence (1-9) and each row represents a certain amino acid (there are 20 different amino acids). Each element of the matrix will then have a score representing the probability to find certain amino acids in a certain position (anchor residues have high scores). The scoring matrix can then be used for prediction by summing up the scores for a test-peptide. Further a cutoff has to be decided for the definition binder/non-binder. Peptides with total scores above the cutoff are predicted to be binders and peptides with scores below the cutoff to be non-binders.

Structure based predictions make use of crystal structures of MHC-peptide complexes. In short, a new peptide is threaded in the template of the one bound in the crystal structure, to see how well it fits into the MHC molecule. Section 2.3 gives a more detailed explanation of scoring matrix methods used to predict MHC binding peptides and also explains more about structure based

predictions.

The approach in this Master's Thesis is to predict MHC binding peptides, using machine learning methods. These are sequence based and the two methods tested are Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs). Sequence data of peptides that bind different MHC types can be extracted from public databases. The sequences can then be used as training data for SVMs and ANNs. A machine learning algorithm tries to map the functionality between input/output pairs. The input will be a peptide sequence and the output will be a yes or no (1 or 0) decision based on whether it binds or not. The machine learning algorithm will then, hopefully, be able to predict if a peptide binds or not given its sequence. The purpose with this report is not to give a complete mathematical description of how ANN and SVM work. Instead, section 2.4.2 will give a general introduction to machine learning. The special features of SVM and ANN are also introduced later in sections 2.5 and 2.6.

The aim of this project is to create a good prediction method for MHC binding peptides. It should be fast and easy to apply to as many different MHC types as possible and it should have a good predictive power. It will be benchmarked against a publicly available prediction method, called SYFPEITHI, and the aim is to obtain better accuracy on predictions with the new method.

The first step is to extract data of binding peptides from a publicly available database and represent it in a way that can be used by SVMs and ANNs. The next step is then to optimize the prediction performances of SVMs and ANNs. The method with the best performance will then be used for the development of the prediction method. A study of the number of binding examples needed for training is also carried out, a step that will decide how many MHC types that prediction can be used for. The final step is then to implement the prediction method as a public web service.

2 Background and theory

2.1 The immune system

This section will give a brief introduction to immunology. The aim is to give the reader a very general overview of the immune system and to highlight the importance of MHC complexes.

The immune system has evolved in vertebrates to protect them from invading pathogenic microorganisms and cancer. It is extremely adaptive and can generate a vast number of cells and molecules used to recognize and kill an almost limitless variety of foreign invaders [1]. The immune system can be divided into two interrelated parts- recognition and response. A foreign molecule can be distinguished from other foreign- and self molecules by small chemical differences. Once the foreign molecule is recognized, an appropriate immune response is raised.

2.1.1 Innate and acquired immunity

The word immunity refers to all the mechanisms that the body uses as protection against foreign agents. Immunity can be divided into two main types: innate (non-specific) immunity and acquired (specific) immunity.

Innate immunity includes all those elements with which an individual is born. These are always present and can be used with short notice. This type of immunity can be divided into four subgroups:

Anatomic barriers: In short these are skin and mucous membranes.

Physiological barriers: Temperature, pH and chemical mediators. An example is lysozyme that cleaves bacterial cell walls.

Phagocytic/endocytic: Some cells internalize foreign molecules and break them down (endocytosis). Other cells internalize, kill and digest whole microorganisms (phagocytosis).

Inflammation: Tissue damage and infection can cause leakage of vascular fluid containing serum proteins with anti-bacterial activity. In an affected area there is also an influx of phagocytic cells.

Acquired immunity is a supplement to innate immunity and is more specialized. It is thought that this type of immunity came to play relatively late in evolution and is only present in vertebrates. Acquired immunity means that the immune system can specifically recognize and eliminate foreign molecules. The characteristics of an acquired immune response is summarized below:

Specificity: The ability to respond to an antigen in a highly specific way, rather than making a random undefined response. An example is antibodies that can differentiate between two molecules that differ by only a single amino acid.

Adaptiveness: The ability to respond to molecules that have never been seen by the immune system before. The immune system can recognize billions of uniquely different structures on foreign antigens.

Discrimination between "self" and "non-self": This is one of the most important aspects of an immune response. It is a vital function to be able to raise a response to foreign molecules and to avoid making responses to molecules that are self. This task is carried out by specialized cells, the lymphocytes, which have surface receptors specific for antigen.

Memory: The immune system is able to recall previous contact with a foreign molecule. This makes it possible for a faster and heightened response to a second response and is the general idea of vaccination.

2.1.2 Cells involved in acquired immunity and specific recognition

There are three major cell types involved in acquired immunity. Two of these cell types develop from a common ancestor and they are the *B- and T lymphocytes*. B cells mature in the bone marrow and T cells in the thymus. *Antigen presenting cells (APCs)*, such as macrophages and dendritic cells, are the third major cell type.

The main feature of B and T cells is their specificity to antigen via antigen binding surface receptors. They are also responsible for other important features of immunology such as diversity, memory and self/non-self recognition.

The antigen binding surface receptors of B-lymphocytes are membrane bound antibodies. When a naive B-cell encounters an antigen for which it is specific, it differentiates into memory B-cells and plasma B-cells (effector B-cells). Plasma cells do not express membrane bound antibodies, instead they produce antibodies in a form that can be secreted. Memory B-cells are long lived and express the specificity of the naive B-cell.

T cells have a different type of surface receptor, the **T-cell receptor (TCR)**. This receptor is expressed during their maturation in the thymus. This receptor can not recognize antigen alone, in contrast to the membrane bound antibodies of B-cells. The TCR can only recognize antigen in conjugation with certain cell-membrane proteins known as the **Major Histocompatibility Complex (MHC)** molecules. When a naive T-cell becomes activated by a antigen associated with a MHC molecule it differentiates into memory T-cells and various effector T-cells.

The T cells can be divided into two major groups, T-helper (Th) and T-cytotoxic (Tc) cells. The distinction of the two subtypes is done by glycoproteins on their surface known as CD4 and CD8. T cells that have CD4 on their surface generally function as Th cells and those that have CD8 as Tc cells.

Tc-cells monitor the body for tumor cells and virus infected cells. They do not secrete much cytokines (immunological "communication" molecules), instead they kill cells with their cytotoxic mediators. The killing of cells requires recognition of MHC I-peptide complexes on the target cell's surface.

Th cells become activated when they interact with cells displaying MHC II molecules complexed with antigen. They secrete various cytokines as well as growth factors important in the activation of B cells.

APCs on the other hand have no such antigen-specific receptors. Their function is to process and present antigens to specific TCRs. The most important function of APC is to activate Th cells. Th cells are very important in directing immune responses and therefore their activation must be carefully regulated. This activation of Th cells is as mentioned above carried out by MHC II-peptide molecules, but another important function of APC's are their ability to produce co-stimulatory molecules that also are important.

A summary of this section would state that there are four different types of receptor molecules responsible for this in the immune system. They are the ones responsible for the specific recognition in the immune system and they are:

- Membrane-bound antibodies on B-cells.
- T-cell receptors
- MHC class I molecules present on all nucleated cells.
- MHC class II molecules on antigen presenting cells.

2.1.3 Clonal selection theory and tolerance

The fact that there are specific recognition of antigen by B- and T cells is the foundation of modern immunology. This gave rise to the clonal-selection theory in the 1950s [5]. This theory states that individual lymphocytes express membrane receptors that are specific for a certain antigen. The antigen specificity of the receptor is determined before exposure to antigen. Binding of an antigen to the receptor causes the lymphocyte to proliferate into a clone of cells, each with the same antigenic specificity. Tolerance of T-cells is induced in the thymus by two processes. First there is a positive selection of T-cells that can bind self-MHC molecules. The second step is a negative selection by eliminating all those T-cells that have too high affinity for self-MHC. A similar process of clonal deletion occurs for B-cells in the bone marrow, where B-cells with too high affinity for self antigens are deleted. These processes of clonal deletion ensure that self-reactive cells are removed from the immune system.

2.1.4 The Major Histocompatibility Complex (MHC)

The role of MHC was briefly introduced in section 2.1.2. Every mammal studied so far have a cluster of tightly linked genes, MHC, which is associated with intercellular recognition and with self/non-self discrimination. The importance of MHC molecules in graft rejection was discovered by Gorer and Snell in the 1930s, a milestone in immunology. They discovered that surface molecules (now called MHC) were responsible for the immune responses causing graft rejection in mouse, something that Snell was awarded the Nobel Prize for in 1980. The human MHC system is also known as the human leukocyte antigen (HLA) system and is located on chromosome 6 [6]. In humans, three loci called A,B and C encode MHC class I molecules. Three regions called DP, DQ and DR encode MHC II genes. There is also a region coding MHC class III genes which are proteins associated with the immune response, but not in the specific presentation to T cells. The function of both MHC I and MHC II molecules is to bind peptides and present them to T cells. This section will deal with MHC I and MHC II molecules by explaining the general structure and how binding peptides are generated.

MHC I molecules consist of two chains, a large α -chain associated with a much smaller β_2 -microglobulin molecule. As can be seen in figure 2, the α -chain have three extracellular domains (α_1 - α_3) and β_2 -microglobulin is associated to this structure. The α_1 and α_2 domains associate to form a groove, where a processed peptide can bind. The peptides bound to the peptide-binding cleft of MHC I molecules come mainly from intracellular proteins. The proteins are generally digested in the cytosol and the peptide fragments are then transported to the endoplasmatic reticulum (ER). In the ER the association of the α -chain, β -chain and peptide take place. The size of the peptide that is presented by MHC I on the cell surface is usually 8-10 amino acids long.

MHC II molecules have also an α -chain and a β -chain noncovalently associated. Each of these chains has two extracellular domains, see figure 2. The

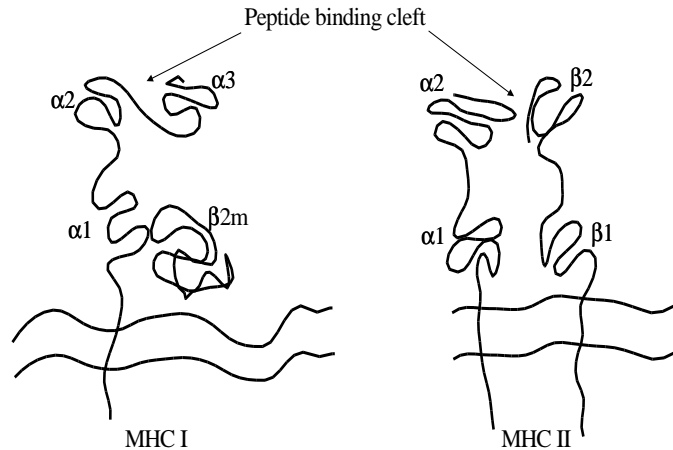


Figure 2: MHC I and MHC II

peptide binding cleft is formed between the α_1 and β_1 domains. The peptides bound to MHC II molecules are often derived from exogenous proteins. These peptides are generally degraded within the endocytic processing pathway, and associate with MHC in endocytic compartments. This means that binding of the peptides does not occur in ER as for MHC I peptides, but the association of the two chains of MHC II occurs in ER just as for MHC I. However, the size of the peptides that bind to MHC II are often 13-18 amino acids.

The processing pathways of both MHC I and MHC II peptides are summarized in figure 3.

One of the main differences between MHC I and MHC II is the peptide binding cleft. MHC I has a cleft that is closed at both ends, whereas the ends of the MCH II cleft are open. This causes the size of the peptide in the MHC I case to be quite fixed and the whole peptide is in the binding groove. MHC II peptides vary more in length and are generally longer than MHC I binding peptides. Since the binding cleft of MHC II molecules is open, the peptide bound can stick out on both ends. However, it is only about 9 amino acids that are located in the binding groove of MHC II molecules. Figure 4 shows how peptides are bound to MHC molecules. MHC I peptides have anchor residues at both ends, whereas MHC II molecules have anchors distributed all over the peptide. This is one of the main reasons why it is easier to predict binding to MHC I molecules. Imagine that you are given sequences of peptides that bind MHC I or MHC II molecules. The peptides that bind MHC I molecules are almost of the same size and it is easy to identify anchor residues. In the MHC

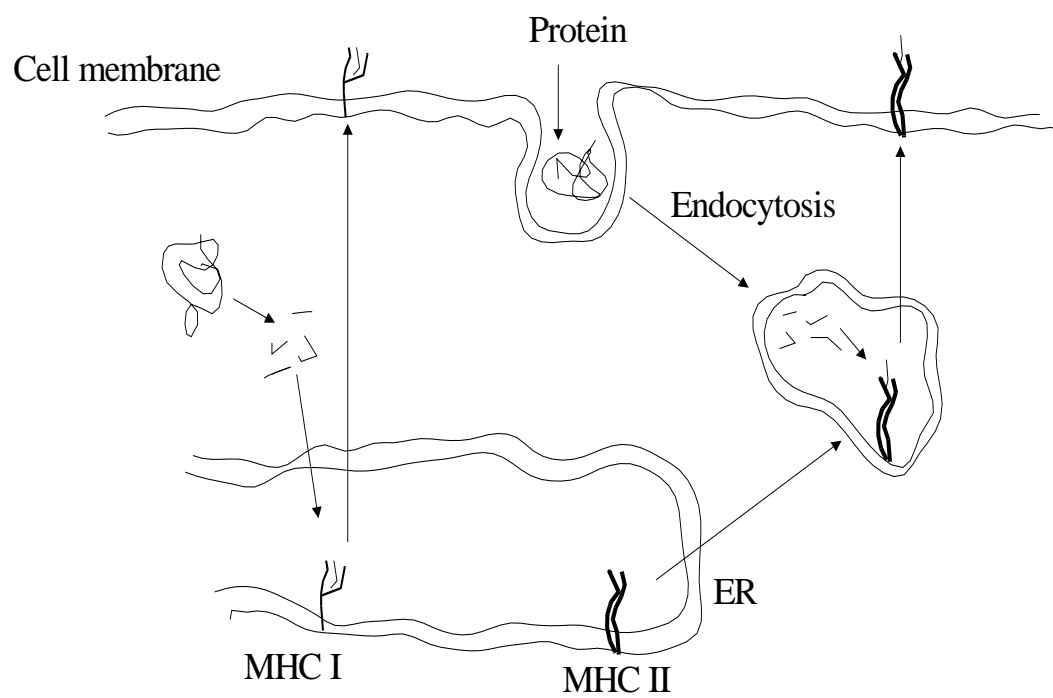


Figure 3: A summary of the MHC I and MHC II peptide processing pathways. MHC I peptides are derived from endogenous proteins and MHC II peptides are derived from exogenous proteins.

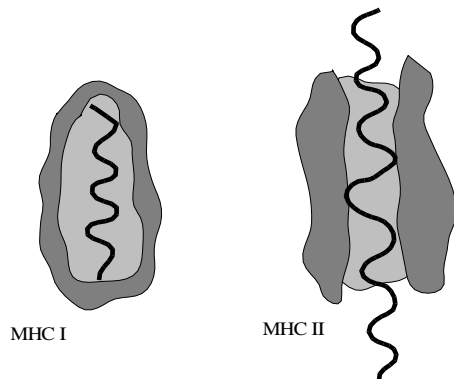


Figure 4: Principles of peptides binding to MHC molecules. MHC I molecules have closed binding clefts and the length of the binding peptide is limited. MHC II molecules are open at both ends, which allows the binding peptides to vary more in length.

In case it is more difficult to find out which amino acids of the peptide that really interact with the binding groove.

2.2 MHC peptide databases

There are a number of publically available databases containing information about peptides known to bind MHC molecules. This section will first explain how binding peptides are purified and sequenced. After that, information about the database used in this study, MHCPEP, is given.

2.2.1 Peptide purification and sequencing

The following procedure can be used for purification and sequencing of MHC-binding peptides [7]:

1. A source of MHC-expressing cells are treated with detergent. (Examples of cells are tumor cells, transformed cells, cells transfected to express a certain MHC molecule, or fresh frozen tissue)
2. MHC molecules are precipitated with solid-phase bound antibodies.
3. The peptides are dissociated from the MHC molecules using acid.
4. After ultracentrifugation the peptides are separated using reverse phase HPLC.
5. The peptides are sequenced using Edman degradation.

```

>HUM10002#
MHC MOLECULE: HLA- A2, CLASS- 1, (HUMAN)#
METHOD: HPLC, cytotoxicity as.#
ACTIVITY: yes, moderate#
BINDING: yes, ?#
SOURCE: HIV reverse transcriptase (460- 485)#
DB REFERENCE: SWISS:
(POL_HV1RH,POL_HV1PV,POL_HV1H2,POL_HV1EL,POL_HV1JR,#
&
POL_HV1B1,POL_HV1N5,POL_HV1OY,POL_HV1Y2,POL_HV1MN,#
&
POL_HV1Z2,POL_HV1BR,POL_HV1ND,POL_HV1MA,POL_HV1B5)#
&      PIR1: (GNVWLV,GNLJND,GNVWVL,GNVWH3,B44001)#
&      PIR2: (B47175,S54378)#
REFERENCES: tsomides91a,kageyama95a#
COMMENT:#
SUMMARY: HLA- A2,actyesm,bindyesu,EILKEPVHGV*##
SEQUENCE: EILKEPVHGV*##
...#

```

Figure 5: A typical entry in the MHCPEP database. The MHC molecule is HLA-A2 and the binding peptide sequence is EILKEPVHGV.

2.2.2 The MHCPEP database

MHCPEP is a curated database with over 13 000 peptide sequences known to bind MHC molecules. The entries come from both published reports and from submissions of experimental data. Each entry contains the peptide sequence, its MHC specificity and where available, experimental method, observed activity, binding affinity, source protein, anchor positions and publication references. An example of a typical entry in the MHCPEP database can be seen in figure 5. There are a number of potential errors in the MHCPEP database. Early reports of MHC binding peptides were usually less specific. The reported peptides were often longer than the optimum size and the fine specificity of the MHC molecules were not determined [8].

2.3 Methods for predicting MHC binding peptides

Different methods have been tried over the years to predict MHC binding peptides. The different methods can be divided into two groups: structure based predictions and sequence based predictions.

2.3.1 Sequence based predictions

Sequence based prediction requires large amounts of peptide binding data. By looking at the frequency of different amino acids in different positions, sequence motifs can often be seen. An example of a sequence motif might be the one seen for peptides that bind to a MHC I molecule called HLA-A*0201. It is very common that peptides that bind to this MHC molecule have a lysine in position 2 and a valine in position 9, the length is often 9 amino acids. The frequently occurring residues in certain positions are called anchor residues. This sequence

motif could of course be used as a simple prediction method. One could search a protein sequence looking for places where lysine and valine are positioned with 6 amino acids in between.

By more statistically looking at frequencies of different amino acids in different positions, a scoring matrix can be produced. This approach gives a matrix with scores for all amino acids in every positions along the binding peptide for a certain MHC allele. A public prediction method that uses scoring matrices is SYFPEITHI. The SYFPEITHI method is based on a two-dimensional matrix where the one letter codes of the amino acids represent the row index and the pocket number represents the column index. An example of a matrix used by SYFPEITHI for the prediction of peptides that bind to HLA-B*1510 is shown in figure 6. The peptides predicted with this matrix have a length of 9 amino acids. The score of a sequence can be addressed directly by a pair of indices. With this matrix as an example, one can slide along a protein sequence and calculate the score for each nonamer. The advantage of using a matrix based approach instead of just sequence motifs is that it gives a quantitative score to each peptide. This makes it possible to rank the peptides according to score [9]. In general the matrices for SYFPEITHI are created in such a way that amino acids that occur frequently at a position are given the value 10, the value 8 is given to amino acids present in a significant number of ligands, and 6 for rarely occurring residues. Amino acids of auxiliary anchor positions are given the value 6, less frequent amino acids the score 4. Preferred amino acids have coefficients 1-4 according to the strength of signals in pool sequencing or the occurrence in individual sequences. Amino acids that are considered to have negative effects on binding are scored -1 to -3 [10].

Another sequence based strategy for predicting MHC binding peptides is to use machine learning approaches. Artificial Neural Networks (ANNs) have previously been used for this type of prediction. The ability of ANNs to classify non-linearly separable data sets may be an advantage. Studies have shown that binding energy for individual amino acids within the peptide is not independent of neighboring residues. This might be one reason why motif based predictions fail in some cases [11]. This Master's Thesis deals with different aspects of using machine learning approaches to MHC-binding peptide prediction. The two machine learning approaches are SVMs and ANNs.

2.3.2 Structure based predictions

An alternative method to scoring matrices is structure based prediction. This procedure is based on structural information about MHC-peptide complexes and evaluates how well a new peptide fit in the binding groove of a MHC molecule. A new peptide is threaded through a structural template to obtain a rough estimate of the binding energy. The energy estimation is based on the interactions defined in a solved crystal structure [12]. A solved structure can be used to construct a matrix of MHC contact residues for each position in the peptide. For a new peptide threaded in this structure, the contributions of pairwise interactions can be calculated. This gives a calculated energy for each peptide

AA	1	2	3	4	5	6	7	8	9
A	0	0	1	0	0	0	0	1	0
C	0	0	0	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0
E	1	0	1	1	0	0	0	1	0
F	0	0	0	0	0	0	0	0	6
G	1	0	0	1	1	0	0	0	0
H	0	10	0	0	0	0	0	0	0
I	2	0	0	0	0	1	0	0	0
K	0	0	0	1	0	1	0	0	0
L	0	0	0	0	0	0	0	0	10
M	0	0	0	0	0	1	0	0	6
N	0	0	0	0	1	0	0	0	0
P	0	0	0	2	1	1	1	0	0
Q	0	0	0	1	0	0	0	0	0
R	0	0	0	0	0	1	2	2	0
S	0	0	1	0	0	0	0	0	0
T	1	0	0	0	0	0	0	1	0
V	0	0	0	1	0	1	2	2	0
W	0	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0	0
Y	1	0	0	0	0	0	0	0	0

Figure 6: The SYFPEITHI matrix for HLA-B*1510. The highest scoring binding peptide should have a histidine in position 2 and a leucine in position 9.

and candidate binders should have low binding energies.

2.4 Pattern recognition (classification) and machine learning

The term pattern recognition includes a wide range of information processing problems of great practical significance, from speech recognition and the classification of handwritten characters, to fault detection in machinery and medical diagnosis [13]. In a way the term covers any context in which a classification is made on currently available information.

Machine learning can be seen as a direct descendant from statistical model fitting. A machine learning algorithm can learn to recognize certain patterns in a data set. This pattern is usually too complex to put a mathematical formula on. Instead, the aim is to extract useful information from an amount of data by building a good probabilistic model. Machine learning strategies find much inspiration from the biological predecessor, the brain. This is the reason why one often talks about "learning" instead of "fitting".

This section will give a simple example of classification and what is meant by separable patterns. This is followed by a general introduction to machine learning.

Table 1: Weight, shoe size and gender for 10 different persons. F stands for female and M for man.

Weight	Shoe size	Gender
67	38	F
73	41	M
87	42	M
79	42	M
84	44	M
78	40	F
107	46	M
56	37	F
55	38	F
82	41	F

2.4.1 Separable patterns

If patterns are to be used for classification, they must in some way be separable. The best way to introduce separable patterns is to give a simple example. Let us start off with a hypothesis that men have bigger feet and weigh more than woman. If this is true it should be possible to predict whether a person is a man or a woman, given the weight and shoe size of that person (i.e. classify the person either as male or female). Table 1 shows weight, shoe size and gender for ten persons. The data in Table 1 is plotted in Figure 7, showing females marked '+' and men marked '*'. As can be seen in Figure 7 a line can be drawn to separate the data points into two groups, one for women and one for men. Since the plot is two dimensional there is a line separating the two groups. In a dimension of order N , the problem is to find a hyperplane in $N-1$ dimensions that separate the groups.

This is an example where the data points are linearly separable in the feature space. In a more complex real-world-application, more complex functions than linear ones usually are needed. Using multiple layers of threshold linear functions is a possible solution to this problem, something that have led to the use of multi-layer neural networks and learning algorithms such as back-propagation. Another approach is to use kernels that project the data into a high dimensional feature space, which is the foundation of a Support Vector Machine approach. These two approaches are discussed in more detail in sections 2.5 and 2.6.

2.4.2 Machine learning

The term Machine learning is generally used for automatic computing procedures based on logical or binary operations, that learn a task from a series of examples. When computers are used for solving practical problems, the required output from a given input can usually be described explicitly. A programmer's task is here to set up a number of rules so that a given input gives the right output. The problem arises when very complex systems are to be analyzed. If

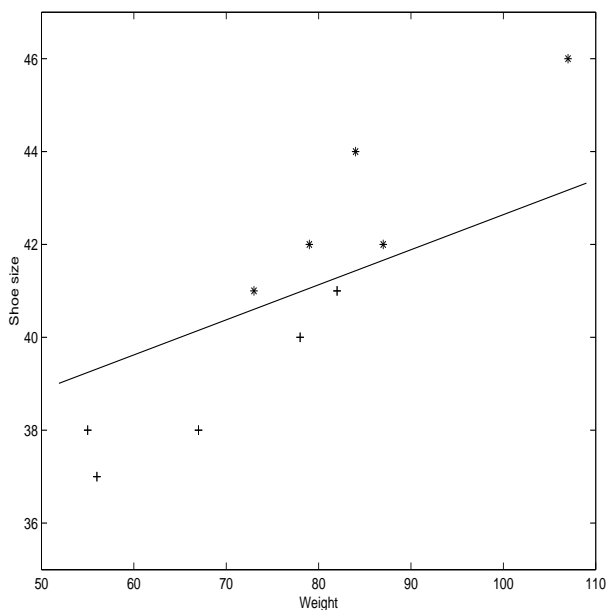


Figure 7: An example of weight and shoe size plotted for 10 persons. As can be seen in the figure, a line can be drawn to separate men from woman.

no general rules are given, it might be impossible to compute the desired output from a given input. An alternative solution is then to learn the input/output functionality from examples. An understandable example of this could be a child learning what cars are sports cars. This can be done simply by telling the child which of a large number of cars that are sporty rather than by giving a precise specification of sportiness. This type of learning is called supervised learning and the examples of input/output are referred to as the training data [14].

The function that maps input to output is called the target function. The solution to the learning problem is an estimate of the target function and is the output of the learning algorithm. The quality of a learning algorithm can be assessed by the number of miss-classifications done during the learning phase. In a classification case the output is often referred to as the decision function. If a classification is of the type sick/healthy or regular car/sports car, it is called a binary classification.

The ability of a function to map unseen data into the right class is known as the generalization, and it is this property that the machine learning algorithm tries to optimize.

By trying to optimize the generalization instead of the true function there is a more "loose" task to carry out and other constraints are put on the learning algorithm. If our estimate of the true function gives the right output, it satisfies the generalization criterion. (There is no constraints on the size or "meaning"

of the function now.) We do not want a function that correctly map all the training examples, but make essentially uncorrelated predictions on unseen data. Functions like this are said to overfit and is usually due to too complex decision functions. There are many different ways to keep away from overfitting, e.g. keeping the complexity of the decision function low.

The best generalization performance will be achieved when the right balance between the accuracy on a particular training set and the "capacity" of the machine is found. Capacity in this sense means the ability of the machine to learn a training set without errors [15]. An example of a machine with too high capacity could be a botanist with a photographic memory. When the botanist is presented with a new tree, he concludes that it is not a tree because it has a different number of leaves from everything he has seen before. A machine with too little capacity could be the botanist's lazy brother, who declares anything as a tree if it is green. Neither of them generalizes very well.

2.5 Support Vector Machines (SVMs)

Support Vector Machines (SVMs) are learning systems that use theory from both statistics and optimization. Their function space consists of linear functions in a high dimensional space. SVMs are trained with a learning algorithm from optimization theory and implements a learning bias from statistical learning theory. The aim of SVMs is to efficiently "learn" good separating hyperplanes in a high dimensional space and optimize the generalization.

The example in section 2.4.1 gave an introduction of how two classes might be separated with a hyperplane, but also explained that this approach does not apply to many real-world examples. The idea of SVMs is to use a different representation of input data. This is done by mapping the data into a high dimensional feature space, something that can increase the computational power of linear machines.

A thorough introduction of SVMs is given by Cristianini [14]. Therefore the aim of this section is rather to introduce some concepts of SVM, without too heavy mathematical verification.

A summary of SVMs could be:

1. Map the data into a high dimensional feature space.
2. Find the optimal hyperplane for separating the features discovered in Step 1.

Figure 8 shows some important steps in SVM learning. Lets start off by assuming that training data have been collected and represented in a satisfying way. The first step is a kernel induced mapping from input space to a high dimensional feature space. The example given here needs some imagination from the reader. The scattered points in input space represent that the points can not be linearly separated in the input space. It is drawn in 2D but could represent a dimension of N . The kernel mapping step transforms the points from the input space into the feature space and makes it possible to use linear machines for separation of classes. The kernel function computes the inner product of two feature vectors corresponding to two inputs, something that

do not increase the number of tunable parameters for the problem [14]. The tunable parameters refer to the optimization step for the separation.

The plot of the feature space show that there is a possibility to separate the points. It can be seen that the points in the feature space can be separated with a line. Remember now that the actual problem is in a higher dimension. Let us now think of the function that separates the two groups in the feature space. With some optimization theory it would not be hard to fit a line that separates the points, e.g. a least square regression model would do. The problem is that we do not know that it is a line that should be used. Other possible functions might be of the second, third or fifth degree, see three examples of possible functions in figure 8. In real SVM cases, the problem is to choose the right complexity of the linear functions in the feature space.

The Vapnik Chervonenskis (VC) theory takes care of the problem of choosing the right complexity for the separating linear functions. VC theory places reliable bounds on the generalization of linear classifiers. This means that they control the complexity of linear functions in feature space and in our 2D plot this could mean that we are restricted to choose a line to separate the points, see figure 8. VC theory gives a sophisticated definition of generalization and describes the factors that have to be controlled by the learning machine [14]. The problem now is the optimization of the linear functions in feature space to give the best generalization, in a way this can be viewed as something like using linear regression in our example. The approach of SVM is to optimize a cost function that is convex and quadratic with linear constraints, something that there are good solution strategies for. A main feature of this is that convex optimization problems have no local minima. The optimization of the cost function should hence give the best separating hyperplane.

2.5.1 Training of SVMs

There are a lot of parameters that can be modified during the training procedure of a SVM. The choice of kernel is the most important of these parameters. The problem of choosing the most suitable kernel for a SVM is analogous to the problem of choosing the architecture for a neural network [14]. Examples of other parameters that may be changed are: trade-off between training error and margin, cost-factor-by which training errors on positive examples outweigh errors on negative examples and different kernel specific parameters. The problem is that no exact theory of how to choose parameters exists.

2.6 Artificial Neural Networks (ANNs)

Many of the ideas concerning artificial neural networks (ANNs) come from biological networks. There is an analogy between neurons in the brain and ANNs in the ability of learning. Figure 9 shows a typical neuron of the human nervous system. The dendrites carry input from other neurons to the cell body (soma).

The number of dendrites might be up to several hundred thousand. All input signals are then summed up in the soma and if the sum is above a certain

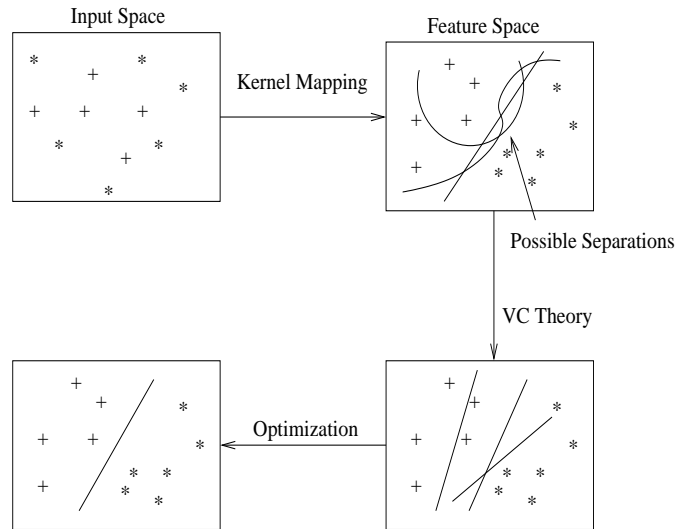


Figure 8: Basic principles of SVMs. The kernel mapping introduces a different representation of the data. In the new feature space it is possible to separate the points. VC theory puts constraints on the functions used to separate the data, ensuring a good generalization.

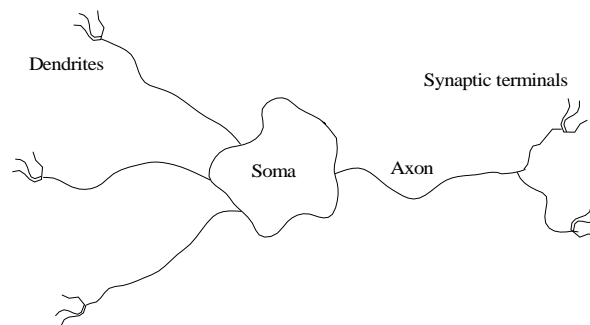


Figure 9: A typical neuron of the nervous system.

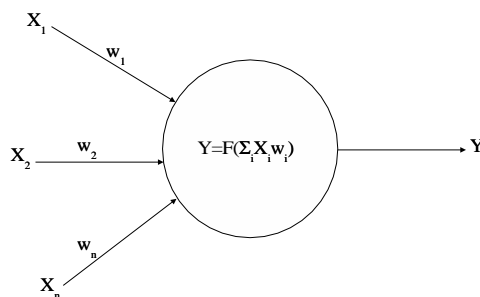


Figure 10: Model of a neuron. Every input X_i is associated with a weight ω_i . The summed inputs are then used by a transfer function to produce the output Y .

threshold, a signal of predefined amplitude is generated. This signal is also called an action potential or nerve impulse and travels along the axon. The axon then divides into smaller parts, synaptic terminals, which can effect the dendrites of other neurons via synaptic junctions. The most interesting feature of a neuron is the threshold for sending a signal. The signal sent does not depend on the input linearly. The process of firing is instead an all or none process. A simplified mathematical model of a neuron can be seen in figure 10. The effects of previous synapses are modulated by "weights" (w), which modulates the effects of the associated input signals (X). The non-linear characteristics of neurons is represented by a transfer function, $F(x)$. The output of the neuron is computed as the weighted sum of the input signals, transformed via the transfer function.

An ANN can be viewed as a network of simple computational units. A unit receives a number of inputs, each input is then multiplied with a certain weight and the output is a non-linear function of the weighted input sum (just as in the simplified model of a neuron). The computational units can be combined in different ways to give the structure of the ANN. An example of a three layered, fully connected, feed forward ANN is given in figure 11. This network has 3 input nodes, a hidden layer with 2 nodes and an output layer containing a single node. Let us now put some mathematical formulas on the network. A computational unit gets an input from the previous layer. This network has 3 input nodes, a hidden layer with 2 nodes and an output layer containing a single node. Let us now put some mathematical formulas on the network. A computational unit gets an input from the previous layer. The total input of a node 1 in the hidden layer is:

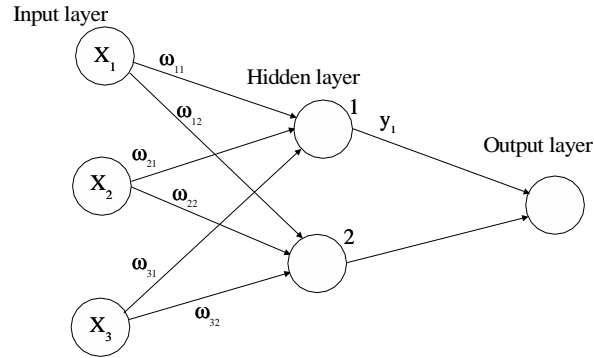


Figure 11: A model of a three layered, fully connected, feed forward ANN. The input layer has three nodes, there are two hidden nodes and a single output node.

$$X_{in} = X_1\omega_{11} + X_2\omega_{21} + X_3\omega_{31} + \omega_i$$

where ω_i is associated with the hidden node 1 and is called the bias, or threshold, of the unit [16].

The output of the node, Y_i will be:

$$Y_1 = F(X_{in})$$

where $F(X)$ is the transfer function of the unit. A frequently used activation function is the logistic one:

$$f(x) = \frac{1}{1+e^{-x}}$$

Examples of other possible functions are $f(x)=\tanh(x)$ and $f(x)=\arctan(x)$. If a nonlinear mapping should be done, the activation function must be nonlinear.

The learning or training procedure of an ANN consists of adjusting the weights and the biases. The goal is to adjust them in a way that when input variables are forwarded through the net, the error between the output and the known value should be minimized. There are different algorithms that deal with this problem.

2.6.1 Training of ANNs

The procedure of finding the weights and biases for an ANN is called training or learning. At the beginning of the training procedure the weights are random and the output produced is totally irrelevant. The weights are then adjusted in each training cycle to minimize the difference between the predicted output and the true output, i.e the error. One error function frequently used is the quadratic error measure:

$$E = \sum_i (P_i - T_i)^2$$

where P_i is the predicted value and T is the true value. There are also other error functions that can be applied in different types of problems [16].

Error back-propagation is an algorithm that carry out the task of minimizing the error function efficiently. First the algorithm evaluates the derivatives of the error function with respect to the weights:

$$dE/d\omega_{ij}$$

This put a restriction on the activation function used, which also have to be differentiable. The next step is to change the weights from the derivatives with a factor η . The gradient descent algorithm uses a fixed negative gradient, which changes the weights in the following way:

$$\Delta\omega = -\eta \frac{dE}{d\omega_{ij}}$$

A small η gives slow learning, but a too large η will often cause oscillations or divergence.

The error for the training data usually decreases to a certain extent at first, but then starts to increase again. This phenomenon is called over-training and means that the ability of the network to generalize is lost [13]. This usually means that the number of weights is too large compared to the number of training data. One way to cope with this problem is to reduce the number of hidden nodes, which decreases the number of weights introduced. Some suggestions have been made that the number of weight should not exceed the number of training examples [17].

2.7 Peptide data representation

A crucial factor when applying a machine learning approach to a problem, is that training data can be obtained and represented in a suitable way. The strategy is often to remove information that is thought to be useless and to try different representations of the data. One example of this might be when length and width of something are represented separately in the training data. Instead it might be possible to use the area (length times width) as representation. This will decrease the dimension of the input data. As explained by Bishop, [13], the

quantity of data needed to specify a certain mapping might grow exponentially with dimensionality. In real life examples the data is often quite limited, and this is the reason for being careful in data representation. One example of a problem is an ANN with many input nodes and few training examples. The risk for overfitting such a system is very high and if the learning data can be represented in a way that the number of input nodes decrease it might help.

In some cases it is also necessary to remove too similar examples of training data to ensure that the ability to generalize is not lost. Imagine a mapping that sorts out apples and pears from a large selection of fruits. If the training is carried out with 3 apples and 3000 pears, it might be possible that the trained machine only find apples. This is of course logical since the adjustable parameters of the algorithm are adjusted for pears 1000 times more than for apples.

The question at hand is how a peptide that binds to MHC should be represented. The peptide sequences in the MHCPEP database are represented by using single-letter codes for the amino acids. One way to represent the peptide is to use "sparse encoding". This means that a binary string is assigned to each amino acid in the peptide. Each binary string consists of 20 positions with zeros in all places except for one. Table 2 explains the sparse encoding representation of the 20 amino acids. If a peptide sequence is 9 amino acids long, the input vector will be 180 (9x20) elements long. This way to encode a peptide does not take any physical properties into account. Two other ways to encode peptide are presented by Brusica et al. [11]. One way is to assign a 6-place string to each amino acid of the peptide, where each place is a scalar value representing a feature (hydrophobicity, volume, charge, aromatic side chain, hydrogen bonds). Values for features are taken from the Database of Amino Acid Indices [18]. Another approach is to assign a 9-place string for each amino acid, by looking at other features (hydrophobicity, positive charge, negative charge, aromatic side chain, aliphatic side chain, small size, bulky size).

2.8 Cross-validation

It is important to put a measure on how good the learning and classification procedures of SVMs and ANNs are. The goal with a machine learning approach is to obtain good classification performance on new data. The simplest way is to use part of the data to train on (training set) and another part of the data for validation (test set).

In cases where the amount of labeled data is limited, cross-validation can be used [19]. The idea of cross-validation is to split the training set at random into N subsets. $N-1$ sets are then used for training and the remaining set is used for testing the performance. The procedure is repeated for all the N possible subsets which is omitted from the training procedure. One drawback of this approach is that the training procedure must be repeated N times. Figure 12 explains the procedure of splitting the training set for a three way cross-validation procedure.

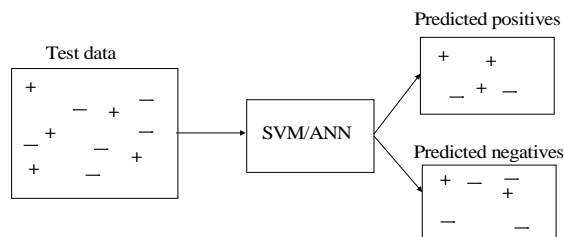


Figure 13: Picture explaining classification into TP, TN, FP and FN. For example, a positive in the test data predicted to be positive is called a TP. If it instead is predicted to be negative it is a FN.

2.9 Performance measurements

In our case with predicting MHC peptides, the training data is of two different types. One labeled as binding and one labeled as non-binding. For testing the performance of the algorithm cross-validation is used. For each input example in the test data, the algorithm will predict whether it is a binder or a non-binder. The predictions of both the SVMs and the ANNs are actually numbers, but given a certain cutoff, a classification either as binder or non-binder can be done. An example would be that the prediction from the algorithm is a number between 0 and 1. A cutoff is decided to be 0.5, meaning that predictions with values higher than 0.5 are predicted to be binders and predictions less than 0.5 are said to be non-binders. Remember now that every example also is labeled with the known class (binder/non-binder), something that can be used for assigning each prediction into one of four groups. A true binder that is predicted to be a binder is called a true positive (TP), a true binder that is predicted to be a non-binder is called a false negative (FN), a true non-binder predicted to be a non-binder is a true negative (TN) and a true non-binder predicted to be a binder is a false positive (FP). Figure 13 explains how predictions are sorted into the four groups. In figure 13 the test data consist of five positives and six negatives. These test examples are taken through the SVM or ANN and are predicted to be either a binder or a non-binder. The predicted binders contain three positives (TP=3) and two negatives (FP=2). The predicted non-binders contain four negatives (TN=4) and one positive (FN=2).

In general there is a trade-off between the amount of false positives and false negatives produced by the algorithm. One way to summarize this is ROC (receiver operating characteristics). A ROC plot displays, for different cutoffs, the

sensitivity (Se) ($TP/(TP+FN)$) versus false positive rate ($FP/(FP+TN)$). Another possibility is to plot the sensitivity against specificity (Sp) ($TP/(TP+FP)$) in a similar plot [16].

Sensitivity is a measure of how many of the positives that are actually classified as positives. If all positives are found, there will be no FN. This means a specificity equal to 1 (i.e. 100% of the positives are found). A specificity equal to 1 means that the classified positives only contain TP (i.e. 100% of the predicted positives are actually positives).

The four different groups of hits can also be used to calculate the Matthews correlation coefficient (Mc):

$$Mc = \frac{(TP \cdot TN) - (FP \cdot FN)}{\sqrt{(TN + FN)(TN + FP)(TP + FN)(TP + FP)}}$$

Matthews correlation coefficient can vary between -1 and 1. A value of 1 means a perfect prediction, 0 equals a prediction no better than random and -1 equals totally opposite predictions.

Another way to measure the performance is the percentage correct predictions on the test set. One thing must be kept in mind if this measure is used, the ratio between positive and negative examples. Assume that the ratio between positive:negative examples in the test set is 1:9. If the classification always classifies all test data as negative, the percentage correct predictions would be 90%. The prediction method is of course not good even if the percentage correct predictions in this case is high (no positives at all are found).

3 Materials and Methods

3.1 Peptide data extraction

The peptide sequences used for SVM and ANN training are extracted from the MHCPEP database. A peptide sequence in the database may contain all the one letter abbreviations for the twenty amino acids. Some sequences have an additional letter X, having different meanings in different cases. The sequences extracted for SVM training were therefore not allowed to contain the letter X. In some cases there are redundant sequences in the database, e.g. when the experimental methods for sequencing differ. Identical sequences were removed to obtain a non-redundant training set.

The training procedure of SVM and ANN needs both positive and negative examples. In our case the positive examples are the peptides from the MHCPEP database. The negative examples are extracted from the ENSEMBL database for human proteins. Protein sequences from the database are chopped up into the length of interest. The whole training set is put together from both positive and negative examples, but sequences in the negative pool of peptides also present in the positive pool are removed.

3.2 Peptide data representation

Training data for both SVMs and ANNs are represented by using sparse encoding for the sequences and a binary input for binder/non-binder.

There is no removal of similar sequences in the training data. The reason for this is that similarity in this case is hard to define. There is no doubt that the peptide sequences that bind to a certain MHC type should in some way be similar. This is the reason why just these peptides bind to this MHC molecule. A scoring matrix approach is a method based on similarity "rules" like this.

3.3 SVM^{light}

The SVM software used for prediction of MHC class I binding peptides is SVM^{light}. SVM^{light} is an implementation of Vapnik's Support Vector Machine for the problem of pattern recognition [20]. The optimization algorithm used in SVM^{light} has adjustable memory requirements and can handle problems with many thousands of support vectors efficiently [21]. SVM^{light} is free for scientific use and can be accessed from http://www.cs.cornell.edu/People/tj/svm_light/. The problem with general off-the-shelf optimization techniques is that they become repellant in their time and memory requirements if the learning task is hard. The implementation of SVM^{light} is a SVM learner which addresses the problem of large tasks, which makes large scale training more practical. The memory requirements are linear with the number of training examples and with the number of support vectors. Nevertheless, the algorithm gains from additional storage space, since a caching strategy allows an elegant trade-off between training time and memory consumption [21].

3.3.1 SVM_learn and SVM_classify

When the SVM^{light} source code is downloaded and compiled, it gives two "functional" modules for the user. SVM_learn is used to learn a model and the input are the input/output pairs of the training examples. In a case with 9 amino acids long peptide sequences the input vectors are 181 elements long (i.e. 180 elements for the 9 amino acids and an additional 1 or -1 for binder/non-binder), since sparse encoding is used. A model learned by the SVM_learn module can be used for classification by the SVM_classify module. The SVM_classify module takes an input vector, representing a peptide sequence, and produces a prediction. The prediction is a value that ranges from about -1.5 to 1.5. The sign of the prediction can then be used for classification, i.e. there is a cutoff of 0 for binders/non-binders. All predictions with a negative sign are predicted as non-binders and all with positive sign as binders.

There are a lot of parameters that can be modified during the training procedure. The choice of kernel is the most important of these parameters. The problem of choosing the most suitable kernel for a SVM is analogous to the problem of choosing an architecture for a neural network [14]. Examples of other parameters that may be changed are: trade-off between training error and margin, cost-factor by which training errors on positive examples out weight errors

on negative examples and different kernel specific parameters. There is no exact theory of how to choose parameters. The choice of parameters that gives the optimum classification has to be investigated for each functional class and is a central part to obtain a good model. The procedure of choosing parameters is carried out by using nested loops, trying a lot of parameter combinations. Training is performed by using three-fold cross-validation. This is done in a way so that all training data are used both for training and testing. More specific details of how this is carried out and is evaluated is given in section 2.9.

3.3.2 SVM^{light} implementation

The different kernels used in the optimization procedure of the SVMs are: linear, polynomial and radial basis functions. For all the three kernels the trade-off between training error and margin was changed as well as a kernel-specific parameter.

3.4 Netlab

The ANN software used in this study was Netlab. Netlab is a library of Matlab functions and scripts based on the approach and techniques described in Neural Networks for Pattern Recognition by Christopher M. Bishop [13].

3.4.1 Netlab implementation

The network used was a three-layer, fully connected feed-forward type that uses error-back propagation. There are, in the case of nonamers, an input layer of 180 nodes (9x20 amino acids), a hidden layer and an output layer with a single node. One part of the optimization procedure of the net is to try different numbers of hidden nodes. The other main part of the training procedure is usually to vary the number of training cycles. Previous studies by Brusica et al. have shown that there is a small effect of varying format of input data, number of training cycles or both [11]. The main part of the optimization will therefore be to optimize the number of hidden nodes and only the number of training cycles will be less investigated.

The training was performed using error-back propagation with a sum of quadratic error functions as described in section 2.6.1. Scaled conjugate gradients are then used to adjust the weights. Both linear and logistic activation functions were tried.

A PERL script is used to call Matlab and to evaluate the results of the predictions. The performance measurement used for ANNs is the same as for SVMs.

3.5 SVM vs ANN

The comparison between SVMs and ANNs is done for the MHC I type HLA-A*0201, which has over 350 entries in the MHCPEP database. The aim of this

comparison is to choose which of SVMs and ANNs to apply on a larger number of MHC types. The training and testing of both SVMs and ANNs are done by using three-fold cross-validation.

The representation of the training data is the same in the both cases, but the two approaches will be optimized only for a limited amount of time. The conclusion of which method to be used might change if the training data are represented in another way and the results can not be generalized to other problems. The performance of the two methods will be assessed by using Sp/Se plots, Mc and percentage total correct predictions.

3.6 Benchmarking against SYFPEITHI

The SYFPEITHI web service include a tool for prediction of MHC binding peptides [10]. To benchmark our method we compare our predictions to predictions done by SYFPEITHI. The SYFPEITHI prediction tool uses scoring matrices and is introduced in section 2.3.1.

The benchmarking procedure is simply to take a test set of known binding and non-binding peptides, run them in each of the prediction methods and evaluate the result. The measure of how good the methods are, is given by sensitivity/specificity plots, Matthews correlation coefficient and total percentage correct predictions.

The output for each sequence run with the SYFPEITHI method is a number, which maximum and minimum depend on the matrix used for the prediction. It is necessary to define a cut-off for binders and non-binders. Binders are usually given a high score and non-binders a low score, but there is of course an overlap. The cut-off chosen is the one that maximizes Mc, i.e. the cut-off is chosen in a way that the performance is as good as possible.

3.7 A public prediction tool

The results from training and testing can be used to create a public prediction tool for MHC binding peptides. The parameters that give the highest Mc during parameter optimization, for a certain MHC type, can be used for a new training procedure. The new training procedure uses the optimized parameters and all available training data for the specific MHC molecules at hand. The measures of the performance of the new models are the ones that were obtained with three-fold cross-validation during parameter optimization. The reason for using all possible training data is to increase the performance of the learning machine.

The training procedure creates a specific model for each MHC molecule. These models can be used for creating the public prediction tool, using CGI-scripting. The idea is that the user specifies a target protein and what MHC type and length of peptide that prediction should be done for. The protein is chopped up into peptides of the length specified and these peptides are then encoded in a way that the prediction algorithm can use. The predictions produced by the algorithm can then be used to create a ranked list of all the possible peptides.

Table 3: Results from the comparison between SVMs and ANNs. Matthews Correlation coefficient (Mc), Specificity (Sp), Sensitivity (Se) and percentage correct predictions are shown for both methods.

Learning Method	Mc	Sp	Se	% correct predictions
SVM	0.82	0.89	0.86	92
ANN	0.73	0.73	0.84	88

4 Results and Discussion

4.1 SVM vs. ANN

HLA-A*0201 was used as a test to compare the performance of SVMs and ANNs. Three-fold cross-validation was used and the training set had 356 binders and 712 non-binders. Specificity/sensitivity plots for the two approaches are shown in figure 14. ANN is the line marked with "+" and the other line is SVM. It can be seen in the figure that SVMs give better classification. The main difference is that ANN score several non-binders high, this can be seen in figure 14 as the inability to reach high specificity. This means that if one wish to choose a cutoff high enough to give high specificity of the prediction, almost no binders will be found (i.e. the sensitivity is low). A cutoff for SVM can be chosen in a way that the specificity is high and at the same time the sensitivity is reasonable. Table 3 shows Matthews Correlation coefficient (Mc), Specificity (Sp), Sensitivity (Se) and percentage total correct predictions for the choice of cutoff that gives the best Mc for each method. The results in table 3 once again show that SVMs perform better than ANNs. Therefore the choice of machine learning method for further studies is SMVs.

There are probably several reasons why a better result is obtained for SVMs than for ANNs. One reason might be that the optimization procedure in this case was easier for SVM. Brusica *et al.* used ANN for prediction of HLA-A2 binders, which showed that representation of the peptides as well as number of training cycles had little effect on the performance [11]. This indicates that it might be hard to find a better performance than the one with optimized number of hidden nodes for ANNs.

4.2 Amount of data needed for SVM training

The amount of available peptide data for different MHC molecules vary a lot. Some MHC molecules have only a few examples of binding peptides and some have several hundred. It is necessary to study the amount of binding data needed for training. The method here is to study how Mc varies with different number of positive training examples. This study is carried out for the machine learning method showing the best performance in initial tests. A MHC type with many binding peptides is chosen for this procedure. Hopefully a fast increase in Mc will be seen in a small range of increasing number of positive training examples. This will give an indication of the least number of binders needed for training,

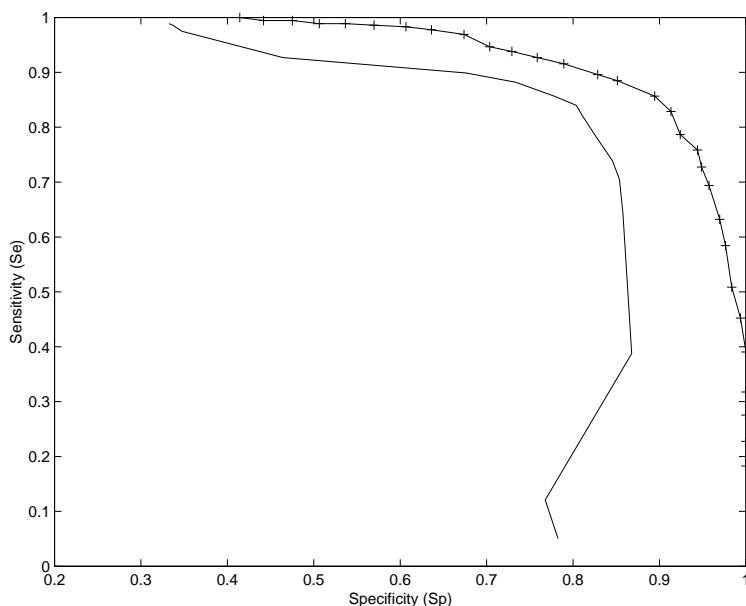


Figure 14: A specificity/sensitivity plot for SVMs and ANNs. ANNs are shown with a smooth line and SVMs are the line marked with "+".

i.e. a cutoff of training examples needed. Training will then be carried out on all the MHC types with more entries than the cutoff.

To investigate the amount of data needed for good SVM training, binding data from HLA-A*0201 are used. The number of entries for HLA-A*0201 non-amers in the MHCPEP database is 356. The procedure is to train SVMs for an increasing number of binding peptides. The number of binders used for training range from 1 to 125. The ratio for positive:negative examples in the training procedure is kept constant at 1:2. The test set for each training consist of 150 known binders and 600 non-binders. Mc for the test set is calculated for each number of binders in the training set. A plot for Mc versus the number of binders in the training set is then constructed, see figure 15. Figure 15 shows a fast increase in performance between 1 and 20 positive training examples. No significant improvement was observed for more than 50 peptides and the number of peptides chosen as a minimum for SVM-training s 20.

The amount of data needed for training depends on the divergence of the peptides that bind to a particular MHC. Here, we have to assume that the divergence is approximately the same for all MHCs.

4.3 Results from all the MHC types used to train on

This section presents the results from all the different MHC types that SVMs were applied to. The result is presented in table 4 showing, MHC-type, number

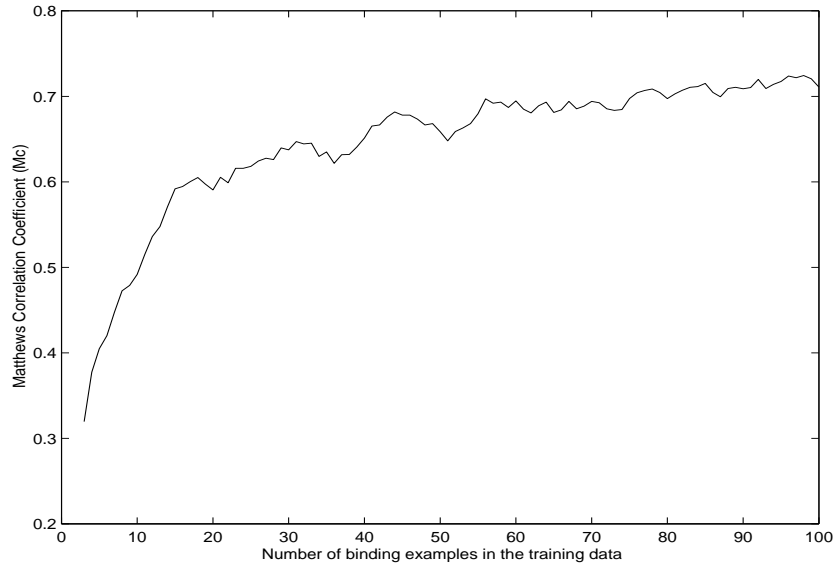


Figure 15: Number of peptides in the training set versus Mc

of training examples and Mc, Sp and Se.

The question that arises when one looks at the results is of course how high Mc should be for a good prediction. However, there is no easy answer to this. Figure 15 shows how Mc increases with the number of binders used for training. This is true in the case of HLA-A*0201 nonamers where there is a lot of binding data available. What happens for a MHC type with a low number of binders is not that easy to say. For some of the small training sets, the peptide sequences are quite similar. This means that the task that the machine has to carry out, in a classification sense, is trivial. The test sequences are almost identical to the training sequences, and hence almost everything is predicted correctly. The conclusion is that the performance of the SVM might be overestimated in cases where there are few training examples. One advantage with a SVM approach is that a new training procedure can be done easily if new binding data are added to the database.

4.4 Benchmarking against SYFPEITHI

Table 5 shows all the 10 MHC types and lengths of peptides that both SVM prediction and SYFPEITHI prediction can be done for.

The results from the benchmark experiment are divided into two sections. The first deals with a measure of how well we do against SYFPEITHI for all the MHC types. The reason for doing this is to get some kind of "overall" comparison between SVMs and SYFPEITHI. It is possible that one method is

Table 4: Results from all the MHC types SVMs were applied to.

MHC	Length	No. of training examples	Mc
HLA-A1	9	78	0.97
HLA-A*1011	9	69	0.80
HLA-A*1011	10	25	0.82
HLA-A11	9	80	0.83
HLA-A11	10	27	0.69
HLA-A2	9	205	0.78
HLA-A2	10	44	0.69
HLA-A*2402	9	84	0.82
HLA-A*0204	9	23	0.37
HLA-A3	9	104	0.73
HLA-A3	10	20	0.74
HLA-A*0201	9	356	0.82
HLA-A*0201	10	130	0.75
HLA-A24	9	22	0.93
HLA-A*3301	9	33	0.86
HLA-A*0301	9	59	0.70
HLA-A*0301	10	35	0.72
HLA-A31	9	40	0.81
HLA-A*6801	9	58	0.80
HLA-A*0205	9	23	0.64
HLA-B7	9	56	0.92
HLA-B8	8	50	0.93
HLA-B8	9	55	0.90
HLA-B14	9	41	0.91
HLA-B44	9	34	0.93
HLA-B*2705	9	87	0.90
HLA-B*3501	8	20	0.85
HLA-B*3501	9	106	0.89
HLA-B*3501	10	39	0.94
HLA-B35	9	27	0.61
HLA-B*2703	9	30	0.83
HLA-B*5301	9	56	0.92
HLA-B27	9	182	0.93
HLA-B*2706	9	20	0.85
HLA-B51	9	82	0.81
HLA-B*5102	9	31	0.83
HLA-B*0702	9	67	0.90
HLA-B*5103	9	30	0.76
HLA-A*0202	9	29	0.74
HLA-B*5401	9	57	0.90
HLA-B*5101	9	42	0.82

Table 5: MHC molecules that both SVMs and SYFPEITHI do prediction for.

MHC Molecule	Length of peptide
HLA-A*0201	9,10
HLA-A1	9
HLA-A3	9,10
HLA-B*0702	9
HLA-B08	8,9
HLA-B*2705	9
HLA-B*5101	9

Table 6: Results from the comparison between SVMs and SYFPEITHI. As can be seen SVMs have higher percentage correct predictions than SYFPEITHI.

Predictionn Method	Mc	Sp	Se	% correct predictions
SVMs	0.85	0.83	0.95	95
SYFPEITHI	0.75	0.72	0.91	91

better than the other for a certain MHC type and the "overall" measure gives an indication of how good the predictions are in a general case. The second section deals with the MHC complexes "one against one".

4.4.1 Overall prediction

There are 10 different MHC types for which both SVM and SYFPEITHI predictions are possible. Each of these types was tested with 15 peptides known to bind MHC and 60 peptides supposed to be non-binders. It is necessary to use the same number of binders/non-binders for the different test sets. If all possible test data were used, good predictions for one MHC type could "hide" bad predictions from another. This is the reason for choosing 15 binders and 60 non-binders. The test set used for each MHC type was identical for both SVMs and SYFPEITHI. The measurement of the performance are Mc, Sp, Se and percentage total correct predictions. Table 6 shows the results of the predictions from both SVMs and SYFPEITHI. A specificity/sensitivity plot for these predictions can also be seen in figure 16. As can be seen from the plot, SVMs give a better general result than SYFPEITHI. The main difference seems to be the ability of SVMs to do more specific predictions. The maximum score differs between different matrices used by SYFPEITHI. This is one reason why the specificity/sensitivity plot for SYFPEITHI looks like it does. This also means that it is hard to define a general cutoff for binder/non-binder in SYFPEITHI prediction. Since the result here is an average of ten different predictions, this does not mean that SVMs necessarily are better than SYFPEITHI in every single case. In section 4.4.2 a comparison of each MHC type is done for the two prediction methods.

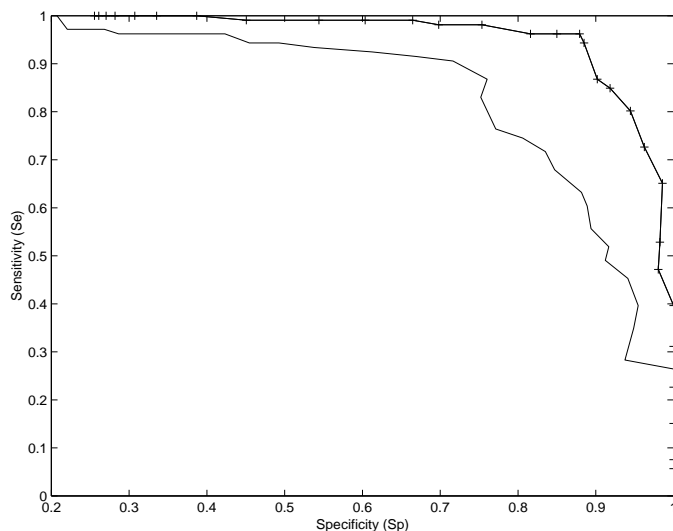


Figure 16: Specificity/Sensitivity plots for SVMs and SYFPEITHI. SVMs is the line marked with "+". The conclusion from this figure is that SVMs all the time have better specificity and sensitivity than SYFPEITHI.

4.4.2 SVMs vs. SYFPEITHI comparison for each MHC type

The result here comes from the predictions during SVM training and a benchmark run against SYFPEITHI. The test set for each MHC type consists of a certain number of binders and twice as many non-binders. The result for each prediction is presented in table 7. These results show that SVM perform better than SYFPEITHI in eight out of ten cases. Four examples of specificity/sensitivity plots are given in figure 17.

4.5 The public prediction tool

All the MHC molecules with a M_c higher than 0.5 in table 4 are implemented in the prediction tool. The prediction tool can be found at <http://www.sbc.su.se/>. The output from the prediction tool is a ranked list of scores for all possible nonamers.

4.5.1 Test with a bacterial protein

The numbers of MHC-binding peptide sequences in the databases are continuously growing. A way to test the prediction method is to run a whole protein, with known binding peptides, in the algorithm. The protein chosen for this is a membrane protein from the bacterium *Chlamydia trachomatis*, SWISS-PROT id P17451. This protein is 371 amino acids long and has three verified nonamers that bind to HLA-A*0201 [22]. The procedure is to chop up the protein sequence

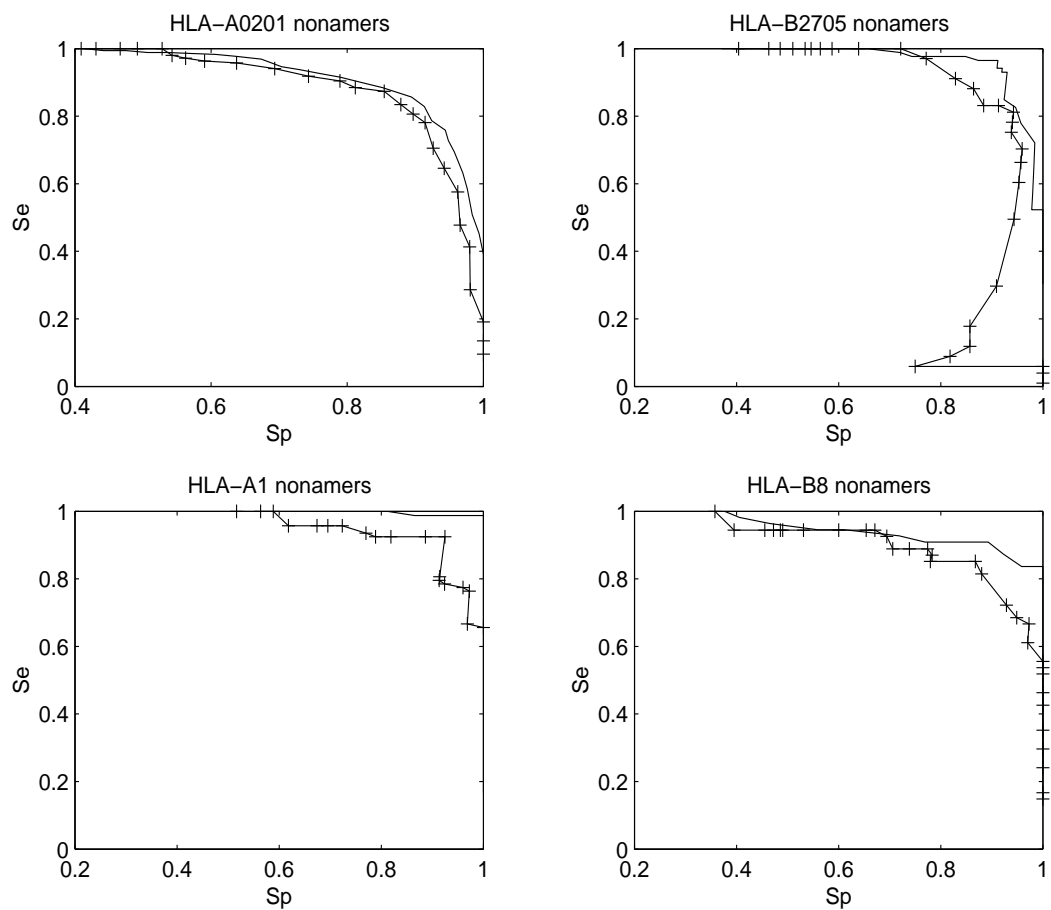


Figure 17: Four specificity/sensitivity plots for different MHC types. SVMs are all the time represented with smooth lines. In all these four examples SVMs are better than SYFPEITHI. The biggest difference can be seen for HLA-B2705 nonamers as score quite a bit of the nonbinders high. This can be seen as the inability to reach high specificity.

Table 7: Results from the comparison of each MHC type between SVMs and SYFPEITHI. As can be seen in the table SVMs are better in eight out of ten cases.

MHC type	Length	SVM Mc	SYFPEITHI Mc
HLA-A0201	9	0.82	0.79
HLA-A0201	10	0.75	0.73
HLA-A1	9	0.99	0.88
HLA-A3	9	0.73	0.63
HLA-A3	10	0.74	0.77
HLA-B*0702	9	0.90	0.88
HLA-B08	8	0.93	0.89
HLA-B08	9	0.88	0.76
HLA-B*2705	9	0.90	0.83
HLA-B*5101	9	0.82	0.89

into nonamers, translate them with sparse encoding and run them in the prediction algorithm. A protein that is 371 amino acids long is chopped up into 364 candidate binding peptides. All these candidate binders are run through the algorithm and a ranked list can be produced from the outputs. Hopefully the three known binders are among the highest scoring candidate binders.

From the 364 possible nonamers of the protein only 19 are predicted to be candidate binders. The question if the three known binding nonamers are among these 19 candidates and what rank they might have. The sequences for the three binding peptides are: NMFTPYIGV, RLNMFTPYI and SLDQSVVEL. The sequences and the scores for the 19 candidate binders are presented in table 8. The three known binding peptides are marked "*" in the table and they are ranked 1, 5 and 6.

The protein was also tested with the SYFPEITHI prediction tool. The sequences are in this case ranked in 1st, 6th and 10th position. The scores were 30, 23 and 21. The problem is that there are 50 peptides scored between 15-25, meaning that SYFPEITHI scores quite a lot of non-binders high as well. SVMs are hence better in this case for reducing the number of total candidate binders.

Table 8: Sequences and scores for the candidate binding peptides of the *Chlamydia trachomatis* membrane protein. The prediction is done for HLA-A*0201 nonamers.

Rank	Sequence	Score
1	*NMFTPYIGV	1.071
2	QLGDTMQIV	1.041
3	GIAVGTTIV	0.844
4	EMFTNAACM	0.842
5	*RLNMFTPYI	0.712
6	*SLDQSVVEL	0.565
7	TLNPTIAGA	0.507
8	LIDERA AHV	0.505
9	QMGDKPTST	0.412
10	ALIAGTDAA	0.393
11	TMQIVSLQL	0.378
12	ALWECGCAT	0.171
13	ILWEGFGGD	0.150
14	KLLKSVLVF	0.120
15	YKGN SASF	0.104
16	TINKPKGYV	0.100
17	ATAPTTLTA	0.095
18	LLKSVLVFA	0.086
19	MIDGILWEG	0.022

5 Conclusions and future improvements

There are several conclusion that can be drawn from the studies carried out in this Master's Thesis project:

- Preliminary studies show that SVMs have better performance, on predictions of MHC-binding peptides, than ANNs.
- Around 20 sequences are required for SVM training.
- In a comparison between 10 different MHC types, SVMs perform better than SYFPEITHI.
- SVMs prediction can easily be applied to a large number of MHC molecules.

Suggestions for future improvements are:

- The effects of peptide data representation should be investigated. Better ways to represent the peptide data than sparse encoding might be possible.
- More training and testing of ANNs could lead to improved performance. Another way to represent input data could decrease the number of total

parameters that need to be optimized, something that might increase the performance.

- Better method for purification and sequencing of MHC-binding peptides are developed all the time. This could lead to more accurate databases. It is likely that the usage of more "high quality" data would increase the performance of machine learning methods.
- More accurate prediction methods could be developed by using structural information in combination with sequence information. Another idea is to use information about the cleavage sites of proteins, i.e. can a specific peptide actually be produced from a protein. A third possibility is to look for transport signals in the protein. Peptides are chopped up and associate with MHC, but there must also be a mechanism that guides the peptides to the MHC molecules.

6 References

References

- [1] Kuby J., *Immunology*, W.H. Freeman and Company, New York, 1997
- [2] Rammensee, H.-G., Falk K., and Rötzschke O., MHC Molecules as Peptide Receptors. *Current Opinion in Immunology*, 5:35-44, 1993
- [3] Honeyman M.C., Brusica V., Stone N.L., Harrison L.C., Neural Network-Based Prediction of Candidate T-cell Epitopes, *Nature Biotechnology*, 16:966-969, 1998
- [4] Mamitsuka H., Predicting Peptides That Bind to MHC Molecules Using Supervised Learning of Hidden Markov Models, *PROTEINS: Structure, Function and Genetics*, 33:460-474, 1998
- [5] Jerne N. K., The Natural Selection Theory of Antibody Formation, *Proceedings of the National Academy of Science USA*, 41:849-857, 1955
- [6] van Someren H., Westerveld A., Hagemeyer A., Mees J.R., Meera Khan P., Zaalberg O.B., Human Antigen and Enzyme Markers in Man-Chinese Hamster Somatic Cell Hybrids: Evidence Synteny Between the HLA-A, PGM3,ME1 and IPO-B Loci, *Proceedings of the National Academy of Sciences of the USA*, 71:962-965, 1974
- [7] Rammensee H.-G., Friede T., Stevanovic S., MHC Ligands and Peptide Motifs: First Listing, *Immunogenetics*, 41:178-228, 1995
- [8] Brusica V., Rudy G. and Harrison L. C., MHCPEP, a Database of MHC-binding Peptides: Update 1997, *Nucleic Acids Research*, 26:368-171, 1998

- [9] Parker K. C., Bednarek M. A. , Coligan.J. E., Scheme for Ranking Potential HLA-A2 Binding Peptides Based on Independent Binding of Individual Peptide Side-Chains. *Journal of Immunology*, 152:163-175, 1994
- [10] Rammensee H.-G., Bachman J., Philipp N., Emmerich N., Bachor O. A., Stefan Stevanovic, SYFPEITHI: a Database for MHC Ligands and Peptide Motifs, *Immunogenetics*, 50:213-219, 1999
- [11] Brusic V., Rudy G. and Harrisson L. C., *Prediction of MHC Binding Peptides Using Artificial Neural Networks*, Complexity International 2, 1995
- [12] Schueler-Furman O., Altuvia Y., Sette A., Margalit H., Structure-Based Prediction of Binding Peptides to MHC Class I Molecules: Application to a Broad Range of MHC Alleles, *Protein Science*, 9:1838-1846, 2000
- [13] Bishop C. M., *Neural Networks for Pattern Recognition*, Clarendon Press, Oxford, 1995
- [14] Cristianini N., Shawe-Taylor J., *Support Vector Machines and Other Kernel-Based Learning Methods*, University Press, Cambridge, 2000
- [15] Burges C., A Tutorial on Support Vector Machines for Pattern Recognition, *Knowledge Discovery and Data Mining*, 2:1-43, 1998
- [16] Baldi P., Brunak S., Chauvin Y., Andersen C.A., Nielsen H., Assessing the Accuracy of Prediction Algorithms for Classification: an Overview, *Bioinformatics*, 16:412-24, 2000
- [17] Lawrence S., Giles B., Tsoi A.C., *Lessons in Neural Network Training: Overfitting May Be Harder Than Expected*, Proceedings of the Fourteenth National Conference on Artificial Intelligence, 540-545, 1997
- [18] Nakai K, Kidera A., Kenehisa M., Cluster Analysis of Amino Acid Indices for Prediction of Protein Structure and Function, *Protein Engineering*, 2:93-100, 1988
- [19] Stone M., Cross-Validatory Vhoice and Assessmant of Statistical Predictions, *Journal of the Royal Statistical Society*, B36, 111-147, 1974
- [20] Vapnik V. N., *The Nature of Statistical Learning Theory*. Springer, Berlin, 1995
- [21] Joachims T., *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1999
- [22] Kim S. K., Angevine M., Demick K., Ortiz L., Rudersdorf R., Watkins D., DeMars R., Induction of HLA class I-restricted CD8+ CTLs specific for the major outer membrane protein of Chlamydia trachomatis in human genital tract infections, *Journal of Immunology*, 162:6855-6866, 1999

7 Acknowledgements

I would like to thank my supervisor Arne Elofsson at the Stockholm Bioinformatics Center (SBC), for ideas and support during the project. I also want to thank my examiner at Linköpings Institute of Technology, Uno Carlsson, for his helping hand with many of the practical details concerning the project.

I would also want to thank all the people working at SBC for technical assistance, ideas and proofreading.

Finally, I would like to thank Annette, my family and friends for everything else.