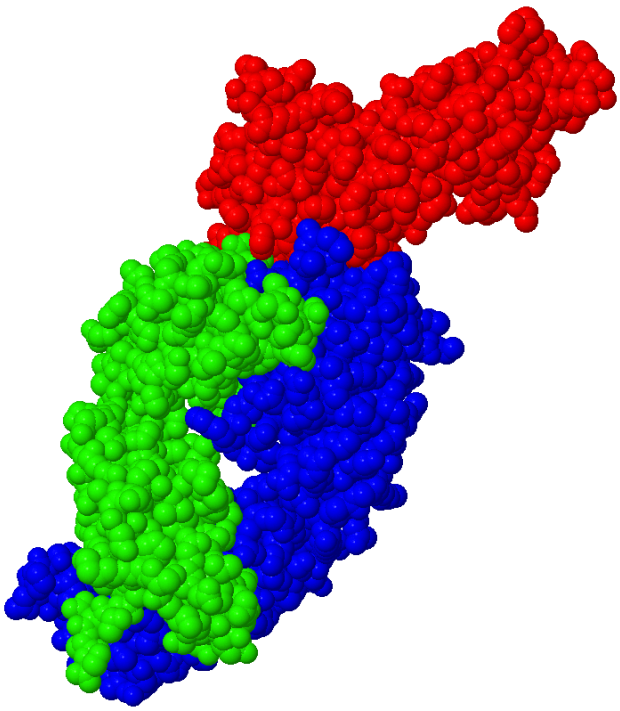

Conformational B-Cell epitope prediction

Master's thesis in Computer Science – algorithms, languages and logic

FREDRIK STRÖM

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2016

Master's thesis 2016

Conformational B-Cell epitope prediction

FREDRIK STRÖM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2016

Conformational B-Cell epitope prediction
FREDRIK STRÖM

Supervisor: Graham Kemp, Department of Computer Science and Engineering
Examiner: Wolfgang Ahrendt, Department of Computer Science and Engineering

Master's Thesis 2016
Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Antigen-Antibody complex visualized in Jmol, colored the different protein chains with different colors.

Typeset in LaTeX
Gothenburg, Sweden 2016

Conformational B-Cell epitope prediction
FREDRIK STRÖM
Department of Computer Science and Engineering
Chalmers University of Technology

Abstract

There is demand for higher quality therapeutic proteins in our society. Medical drug developing companies strive for lower development cost and shorter development time. This require faster and more reliable ways of testing the therapeutic proteins before releasing them to the public. This project aimed to investigate one part of this problem, the conformational B-cell epitopes which is the interface between an foreign molecule (antigen) and an antibody. It was done by development of two different epitope models, which then was used as a base for creation of two training data sets. These training sets were then used to train machine learning algorithms in order to classify areas on molecule surfaces which are prone to be an epitope. Different problems in this research area is discussed and possible solutions is proposed.

Keywords: epitope, conformational epitope, artificial neural network

Acknowledgements

Graham Kemp, Helena Lyberg, Gothenburg, 2016

Contents

List of Figures

List of Tables

1
Introduction

This chapter gives an introduction to the thesis. It contains a brief background, followed by the purpose and method of the thesis. At the end of this chapter there is an outline of this report.

1.1 Conformational B-cell epitopes

The development of therapeutic proteins (medical drugs) is going faster and are subject of higher quality requirements than ever before. There is demand for medical drugs which can cure diseases and still leave minimal side effects. Due to the high demand of quality there is need to rigorous testing an validation before any therapeutic protein is approved to be used by the general population. These test include animal and human testing, which is an ethical grey zone.

There is basically three main properties of therapeutic proteins which must be considered. They must not be toxic or harmful, effectively treat the disease and must be accepted by the immune system of the host. These properties makes the development process very complex and time consuming. In recent years there have been breakthroughs in computational tools which can aid in the development. This thesis focuses on tools which predict the compatibility of a therapeutic protein with the host immune system. This problem is very complex and there is very limited knowledge in the specific focus area of this thesis.

The prediction tools aim to predict whether a foreign protein (antigen) would bind to an antibody. This binding can occur in two ways, either in a linear binding where the antibody binds to a continuous part of the protein chain or a conformational binding which binds to different parts in the protein chain. Both types of binding sites are called epitopes and this project explores the prediction of conformational antigen-antibody epitopes.

1.2 Purpose and goals

The purpose of this thesis is to develop and evaluate an algorithm to predict conformational B-cell epitopes. By feeding the algorithm a protein it should output the likelihood that there is a epitope present on its surface and a possible area where it is present. To keep the problem feasible, the algorithm targets epitopes and antigens in a given size spectrum.

The aim for this algorithm is to aid in the development of therapeutic proteins. It would make the development faster and give the end product less side effects if it is successful.

This thesis compare the results found within related work, compare models and performance, this project aims to develop its own models and classification algorithm, based on related work. Additionally, this project uses existing data and gather no new data.

1.3 Method

There have been multiple attempts at solving this problem with various approaches and data. This related work are evaluated and used in design decisions and evaluation.

Since data sets of confirmed conformational B-cell epitopes already have been gathered by previous work, this thesis uses these existing datasets to train the classification algorithm. These are combined in order to access a larger data set. Gathering of any data outside existing data sets is not in the scope of this thesis.

This project develops its own models, with inspiration from the previous work. There are several different models which have achieved similar performance, however none of them delivers satisfying results.

The evaluation of the algorithm is revealing and comparable to the related work. This might not be an easy task, since there is no unified definition of an epitope, also the evaluation techniques is different from article to article. This means that the comparison might not be fair.

1.4 Thesis Outline

Chapter 2, gives an more comprehensive background to the research area and to the problem at hand. In addition it present some relevant related work in the topic. Chapter 3 provides details on the available data and its availability.

Chapter 4 begins the description of design and development of the models. This is followed by Chapter 5 which focuses on the design and training of the final classification algorithm.

At the end of the thesis Chapter 6 provides the results gathered during the thesis and Chapter 7 discusses the whole process and its results together with the related work. Finally the conclusions of this thesis is provided in Chapter 8.

2
Background

This chapter provides insight into the conformational B-cell epitope prediction problem. It contains a brief background in immunology and classification, it is not necessary for the reader to have a comprehensive understanding of the biology or the chemistry in order to grasp the problem at hand.

2.1 Immunology

This section gives an introduction to immunology processes and terminology which is relevant to this thesis.

The human immune system fights foreign pathogens[1] invading the body, such as bacteria and viruses. It does so in many different ways, these can be divided into two main subsystems, the nonspecific and specific immune system.

The nonspecific part, accounts for mechanisms which do not target specific pathogens. A few examples of such mechanisms is barriers such as the human skin and stomach acids. There is also internal defence which mainly consists of white blood cells which produces antibodies. This thesis will focus on a subset of the specific immune system.

The specific immune system targets specific intruding pathogens. It is the part of the immune system which gives the host a resistance to certain pathogens. This part consist of lymphocytes. There are two different types of lymphocyte[2], B-lymphocytes (B-cells) and T-lymphocytes (T-cells). Both originate from the bone marrow, but the B-cell is prepared in the bone marrow, while T-cells is prepared in the Thymus. This thesis will focus on the B-lymphocytes also called B-cells.

2.1.1 B-Cells

Each B-cell has receptors on its surface, which can bind to pathogens. These receptors are unique to that B-cell and are not inherited by its children, which are produced in the bone marrow. These receptors will later be a part of antibodies which binds and tags pathogens[3]. While the receptors sit on the B-cell surface their

[1] pathogen: is anything which can produce a disease, such as bacteria and viruses.
[2] lymphocytes: Is a subset of the white blood cells.
[3] Antigens is a subset of pathogens

purpose is to identify pathogens and use their findings to activate the B-cell with the help of a helper T-cell[4].

When a B-cell receptor binds to a pathogen, it will ingest it and ask for a helper T-cell confirmation (this is not always the case, but the most common). If it concludes that this is a threat the B-cell is activated. The B-cell with the successful binding receptors will start cloning itself into plasma cells and memory cells. The memory cells will be used for future invasions and the plasma cells are antibody factories, which start producing antibodies to respond to the immediate threat. These antibodies has the same binding site design as the receptors from the origin B-cell. The antibodies are then released into the blood stream to bind and tag the invading pathogens which are then destroyed by other parts of the immune system. The origin B-cell is then acting as a memory for the immune system, waiting for the next attack from the same antigen.

2.1.2 Antibody-Antigen bindings

The antibodies originating from a B-cell are designed to bind to a specific antigen at a specific site on the antigen surface. This binding site on the antigen is called epitope and is the focus of this thesis. Only the tip of the antibody is binding to the antigen and it is this tip which is the main subject to change or evolve from B-cell to B-cell. Even if only this small part is the core of the binding, this part has to possibility of changing into a huge diversity of shapes and are able to bind to an uncountable number of areas on antigens.

The surface of an antigen is the result of the folded amino-acid chain forming a three dimensional structure. The surface is then defined by the atoms accessible to other molecules.

There are two kinds of epitopes, linear epitopes and conformational epitopes. The linear epitope is a binding site which runs along a continuous part of the amino-acid[5] chain, while the conformational epitope contains different parts of the amino-acid chain. In figure 2.1 the chain is colored with a gradient, a linear epitope would contain one color while a conformational would contain multiple colors.

[4]A helper T-cell is basically confirming the finding of a B-cell

[5]Amino-acid is a reoccurring collection of atoms which is labeled with a name

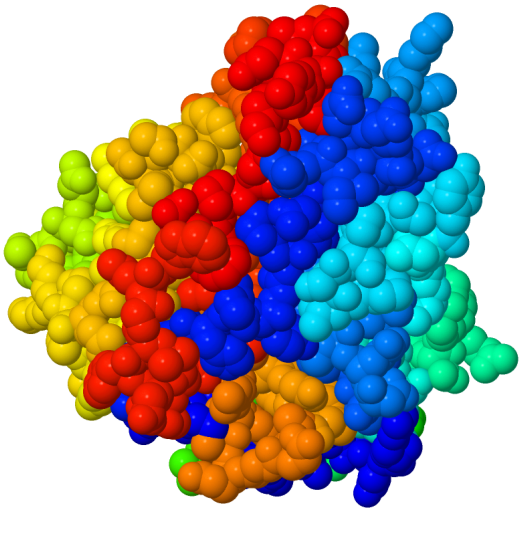

Figure 2.1: This figure shows the atom representation the from folded N chain from the PDBid 1a14. The chain is colored with a gradient, one can follow the color to follow the protein chain.

2.2 Classification

There are many ways to classify data. One major area is machine learning and this is the focus of this thesis, specifically artificial neural networks(ANN). The main purpose of machine learning is to construct a mathematical model which takes the data as input and gives back a classification output. One of the core constraints on the model is that it must be possible to optimize it, in other words 'train' it, in order to acquire a desired behavior.

In machine learning there are a many different approaches to take in order to create a suitable algorithm. Some require a lot of example data for the training and some require less. There are those which need the examples to be annotated with respective class, these methods are called supervised learning. Some which can discover classes within a data set of it's own, which is called unsupervised learning. This thesis will use a basic ANN with supervised learning, where the training requires a lot of annotated data examples.

2.2.1 Artificial neural networks

The basic structure of an ANN has an input node which is connected with a weight to a output neuron. This neuron has a activation function, with a threshold, which takes the sum of the input and decides whether or not it should fire a signal. The weight and threshold are variables and prone to the optimization. The optimization tunes the weight and threshold until the given input gives a desired output signal. This optimization process is called training an ANN.

2.2.2 Training data

As with all training, one must have something to train on. A neural network needs a representative data set to train on. For example, one wants to classify farm animals such as cows, pigs and sheep. If the training data set contains 60% cows, 30% pigs and 10% sheep, then the resulting network will be biased towards cows. It might even over fit the cow classification which will cause the network to loose the overall features of a cow and only learn each and every example in the data set, so when a new cow is presented to the network, it will not be able to classify it.

When it comes to farm animals, we humans have an intuition on how many examples are needed and which are representative. But when there is new abstract data, which humans can not relate to, it becomes much harder to determine the quality of the data set. One might not even know which features in the data is important or if the data set is too small, narrow or biased.

2.3 Related work

Research in conformational B-cell epitopes is an ongoing and open area of research. One of the first articles on 3D structural analysis and epitope prediction was published in 1984 [13]. They discovered some correlation between segmental mobility (temperature readings from x-ray crystallography) and epitopes, however this discovery only applied to linear epitopes and not to conformational epitopes. In recent years there have come a few more methods which attempt to predict conformational epitopes with various results and approaches. This section will present and describe a few influential methods.

2.3.1 Discotope

In 2006 Discotope [8] was developed as a novel method for identifying epitopes on antigen surfaces. They use a combination of amino acid statistics, spatial information and surface exposure. The amino acid statistics are analysed by the log odds ratio and together with the other parameters they tuned/trained their algorithm. For this they used a refined data set containing 75 PDB[6] files containing the epitope examples. These were then used to train their method to identify which residues (a collection of atoms within a 10 Å radius sphere) are likely to be a part of a epitope. Their results and evaluation show a specificity(true negative rate) of 95 % and sensitivity(true positive rate) of 15%. Later in 2012, Discotope 2.0 was developed and published [11]. This improved method combines the same log odds ratio with different spatial information, which yields a minor improvement to the prediction performance. They also highlight the importance of the evaluation data set as well as the training set. They discuss the complications of having incomplete data and evaluating the methods in a biased manner.

[6]PDB file: a standardized format for molecule structure information.

2.3.2 ElliPro

ElliPro was developed in 2008 and is one of the methods which does not require training. It is based on Thornton's method [12] where it approximates the protein shape with a ellipsoid in order to calculate a protrusion index (PI)for each residue. Then it uses these PIs to cluster the residues as epitopes or not. The performance of ElliPro is quite similar to Discotope, they reported specificity at 89.1% and sensitivity at 16.5%.

2.3.3 Epitopia

Epitopia is one of the few methods which relies solely on one machine learning classification method, Naive Bayes, and was developed in 2009 [14]. It divides the surface of the given antigen into overlapping patches. These patches have the size of a typical epitope and is used to calculate 44 physicochemical and structural-geometrical properties which are then used as input to the Naive Bayes classification algorithm. Just as Discotope, this algorithm requires training on a data set of epitope examples and they used a updated version of Discotopes data set which, after refinement, contains of 66 examples.

Epitopia can in addition to taking a PDB file as input also take a sequence of amino-acids as input and do the B-cell conformational epitope prediction. This algorithm tries to cluster the amino-acids and calculate the probability that they are prone to be a part of an epitope.

The performance of this method is hard to evaluate from the published paper. They report a 70% successful prediction rate of epitopes, however they fail to report any rate of false predictions and specificity and sensitivity, which gives a better view of the performance of the algorithm.

3
Epitope data

This chapter shows the most common methods for discovering epitope data and where it is available.

There are multiple steps in discovering an epitope. The first step is to find a potential molecule, an antigen-antibody complex. Next there is determining the structure of that complex, see section 3.1. Once the structure is determined and the atoms are labeled to their corresponding protein chain, the final step is to identify the antigen antibody binding site. Depending on the epitope definition, the atoms or amino acid residues will be considered a part of the epitope in this complex. A common measure, which is used in this project, is to consider all atoms in the antigen which are within 4 Å from any atom in the antibody [8, 1].

3.1 Gathering methods and quality

X-ray crystallography is the most common way to determine the structure of a protein and is the only method used for discovering the proteins in this thesis's data set. The basic idea of X-ray crystallography is to fixate the atoms in the protein by crystallizing it, then use an intense X-ray to determine their positions. This is not a perfect method and it works better on some proteins than other. For example this method is well suited to analyze molecules which are stiff, because then the fixation will not have a big impact on the structure and it is easier to reproduce the result. But if the molecules are flexible and move around a lot, then it can be harder to interpret the result from the X-ray and the result will not be as reliable.

A protein is not a fixed molecule. The protein chain and the bonds between atoms is most of the time continuous, but the atoms may rotate around these bonds in various ways. This means that it is not essential to know the absolute position of each atom, but the resolution becomes a problem when it is too low to determine the atom bonds in a sufficiently accurate way. One of the most important factors is the quality of the crystallization and the resulting resolution. If the quality is low then the resolution of the atoms will also be low, the atoms position will have a larger error margin.

3.2 Raw data

One way of storing detailed structural data about a protein is to use the RCSB PDB. They have an ever growing data bank of protein structures stored in the PDB

file format [1], where one can find information regarding each atom present in the protein. It will contain information such as the atom position in three dimensional space, type of atom and which protein chain it belongs to.

The information which was interesting to this project was the coordinate position in space of each protein, the atom type (carbon, hydrogen and so on). The information regarding the chain is also important for determining to which molecule the atom belongs. One should not forget the atom identification number, so one can keep track of each individual atom. In most PDB files there are a lot more meta data and data regarding the discovery method, which was not used by this project.

3.3 Publications and databases

The Immunology Epitope Database (IEDB)[2] is a ongoing project which aims to collect and distribute epitope data, including both linear and conformational epitopes. The database has modeled the epitopes as a collection of residues, which is a part of the epitope. A residue is a small collection of atoms, which has been given a label. This means that if a residue is a part of an epitope, then there is at least one atom in that residue binding to the antibody and is a member of the epitope. The database also contains references to the origin research article, in which the discovery was made.

The RCSB protein data bank (RCSB PDB)[3] provide sequence and structural data regarding proteins. It is a part of the worldwide protein data bank, which is a network of collaborating Protein Data Banks (PDBs) and forms a single repository for biological molecules such as proteins.

Initially, this project investigated the IEDB to find a big qualitative data set of epitopes. But it fell short, due to the lack of structural information and no clear reference to any PDB. However, they did provide links to previously assembled data sets, see section 3.4.

3.4 Data sets

The epitope data sets contains the PDB id, to download from the rcsb.org, specifies the antibody chains and the antigen chain. In some cases it specifies the residues which are in the epitope. The data set contains antigen-antibody complexes, where the antibody is not exclusively human, but can also come from other species such as mice.

Due to limited time and resources only two of the data sets listed in the IEDB were considered and extracted. The first one was developed for Discotope 2.0 [11], [16], this data set is one of the most recent and refined datasets freely available.

[1] http://www.wwpdb.org/documentation/file-format
[2] www.iedb.org
[3] www.rcsb.org

They started with a set of 801 PDB files, which were identified from www.pdb.org. The analysed and filtered the data to remove data which did not contain antigen-antibody complexes, redundant and epitopes which were too small. This resulted in a data set containing 107 epitope examples [16].

The second data set was created from data gathered from the rcsb.org in 2007. They used a 4 step method for assembling their data set of 65 representative examples [10]. These two data sets were merged together to form a data set containing 126 examples.

4

Surface model

This chapter provides details on the models developed during this project and their development process.

4.1 Antigen surface

A conformational epitope is found on the surface of the molecules, meaning that there is only direct interaction between the surfaces of the antigen and antibody. This is not the whole truth since there is also indirect interaction from the atoms surrounding the epitope atoms, by changing the binding properties of the epitope atoms allowing for the antibody to bind. This project takes this dynamic in consideration when modeling the surface and epitopes, in the following section provides details on how this was done.

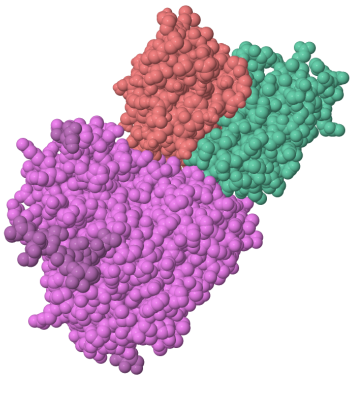

Figure 4.1: The Jmol illustration output from pdb id '1a14', showing all the atoms in the protein complex. Where the purple part is the antigen and the red/green part is the binding antibody.

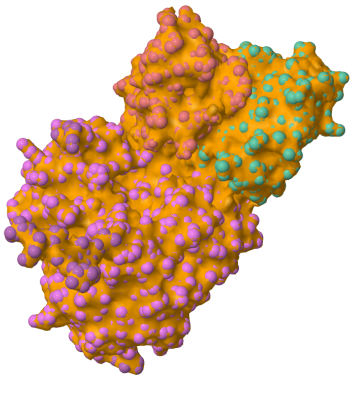

Figure 4.2: The solvent accessible surface of pdb id '1a14', all the atoms which peek through the orange surface are atoms which are solvent accessible to the outside. From Jmol's built in functions.

4.1.1 Solvent accessible surface

To extract the continuous accessible surface from a protein, a program developed by Graham Kemp called 'Triominoes' was used. Triominoes runs a probe, with set

radius, over the protein and finds all triplets accessible by this probe. A triplet is defined by three atoms which the probe can touch at the same time without being within a set distance from any atom in the protein.

In addition to identifying the surface atoms, it classifies the atoms into groups. These atom groups are based of the surrounding atoms and gives more information on the properties of the atoms [17]. The groups are provided to Triominoes by a meta file, the one used by this project is found in appendix A.1. This program was also used in a thesis by W. Mehio [18].

4.1.2 Surface representation

The surface was modeled as a graph, where each node is a atom and each of its neighbouring atoms is connected to it by an edge. Triominoes made it easy to find all neighbouring atoms by checking the triplets for an atom. All the atoms in a triplet are neighbouring each other.

This graph representation makes it easy to create a continuous patch of the surface, which will be explained in more detail in the next section 4.2.

4.2 Surface patches

The surface is divided into overlapping patches. A patch is a continuous area on the surface and they are labeled as an epitope patch or a non-epitope patch. A patch is annotated as an epitope patch, only if it contain at least 80% of the atoms in the confirmed epitope in that antigen. This parameter is debatable, but it seems to be a fairly good way to increase the number of epitope patches without loosing too much of the raw information in the epitope. All other patches in the antigen is labeled as non-epitope patches.

A patch is constructed by selecting a starting atom and adding it to the patch, then incrementally adding atoms neighbouring the patch until a desired patch size is reached, which is easily done due to the graph representation of the surface where it is easy to traverse the graph to retain the neighbouring atoms. This thesis has considered two different approaches to patch creation which will be described in detail in section 4.2.2.

4.2.1 Comparability

One of the biggest problems of this patch model, is to compare two patches from different areas on the surface. They are in two different parts of the three dimensional space and the surface which they represent is probably facing different directions. In order to address these issues a 'normalization' algorithm was developed. First it calculates the orientation vector which points from the center of the protein center of mass to the starting atom. Then it moves the patch to the origin O, centering the starting atom at O, followed by aligning the orientation vector with the Z-axis. This way it becomes easier to compare two different patches.

After the 'normalization' is performed one must also consider different rotations around the z-axis. In addition one can consider tilting the z-axis, but this is not considered in this project. The rotations is performed clockwise with a set degree interval. After some testing it was found that a 10 degree interval is a good balance in order to capture the structure of a patch and being able to recognize it with an arbitrary rotation.

4.2.2 Patch creation algorithms

The patch creating algorithm have two implementations, described below in section 4.2.2.1 and 4.2.2.2. Both of them picks a starting atom and then creates a set of atoms (patch). In order to cover the whole surface one has to pick several starting nodes with a satisfying distance between each other. One could pick every atom as a starting node, but then there might be a lot of redundant patches and a huge amount of data.

This project developed an algorithm for creating a set of starting atoms, which first adds all atoms on the surface to a set of nodes A. Then it picks one starting node, adds it to a set of starting nodes B, then removes all of its neighbouring atoms in the surface graph and itself from A. Then it picks the closest atom still in A, to be the next starting atom. Then repeat the process until A is empty. This way there will be a gap of one atom between each patch, it also reduces the number of patches extracted from the surface and reduces the redundant data. With the data set used, hardware available and time constraint this became a good compromise, as there is not much information lost due to the extensive overlapping of the original patches.

4.2.2.1 By radius

The idea is to pick all atoms, on the surface, within a given radius to be a part of the resulting patch.

Pick a starting atom(node) and add it to the patch. Then traverse the surface graph and add the next node to the patch if the node has a path to the starting node which is less than the radius.

4.2.2.2 N closest

This algorithm will always pick a set number of atoms to be a part of the resulting patch. This implementation picks the N closest atoms.

Pick a starting atom(node) and add it to the patch. Then add the node which has the shortest path to the starting node. Keep adding until there are N nodes in the patch.

4.2.3 Data sets

The resulting patches have as much diversity as the surface of the proteins. There are varying atom densities, surface shapes, deep valleys, high mountains, thin branches

and cylinder shapes to name a few. It could be helpful to categorise them by resemblance in order to divide them into more manageable groups, but this is no easy task and in the end finding resemblance and correlations is the hard problem which this project is to solve.

This project has been looking at two varying patch attributes, the protein size and the epitope size. These two attributes have a dynamic which can be used as a base for dividing the data set. The patch size should not cover the whole surface of the protein, this would defeat the purpose of the patch method. Second, the patch should cover the epitope on the given protein, if doesn't then it would be harder to identify the epitope patches just because there is a lack of knowledge in which parts of the epitope is important for the antibody binding.

Analysis of the data set revealed the two attributes for each protein, the number of epitope patches and the width 'WG' over the graph of the corresponding epitope, which is listed in Appendix A.2. WG is measuring the width of an epitope by traversing the graph instead of the Euclidean distance. WG is the number of nodes (atoms) traversed by the shortest path between the two atoms, in the epitope, which are furthest apart. The attributes vary too much to be represented by one single patch size, which is why the data set was split into three divisions, small, medium and big. Each set is satisfying the dynamic described previously, that they must cover the epitope while not covering too much of the actual molecule. In addition the patch should cover the epitope by some margin, which provides additional epitope labeled patches to the training set. It was not possible to draw a hard line between the three parts and therefore there is overlap between them.

4.2.3.1 Analysis

Correlation analysis was performed on the total set of patches where the correlation between the number of atoms belonging to each atom group and the epitope patches was calculated. Table 4.1 shows the correlation values $[-1, 1]$ from the by radius algorithm generated patches. Table 4.2 shows the correlation values $[-1, 1]$ from the N closest algorithm generated patches. They do not reveal any significant correlation.

	small	mid	big
S2H0	0.024	-0.003	-0.058
C4H3	-0.073	-0.022	0.003
C4H2	-0.048	0.000	-0.008
C4H1	-0.059	-0.007	-0.007
C3H1	0.011	0.015	-0.021
C3H0	-0.031	0.010	-0.008
O2H1	-0.062	0.010	0.005
N3H0	0.032	-0.008	-0.059
N3H1	-0.039	-0.005	-0.014
N3H2	-0.015	-0.003	-0.008
O1H0	-0.051	-0.014	-0.005
S2H1	-0.024	-0.011	-0.039
N4H3	-0.048	0.004	0.058
UNKN	-0.031	-0.023	0.011

Table 4.1: This table shows the correlation values $[-1, 1]$ for a patch being an epitope patch and the #number of atoms belonging to the corresponding atom group. Each column shows the correlation for the corresponding data set generated by the 'by radius' algorithm.

	small	mid	big
S2H0	0.004	-0.023	-0.118
C4H3	-0.090	-0.063	-0.080
C4H2	-0.106	-0.052	-0.081
C4H1	-0.101	-0.065	-0.087
C3H1	-0.030	-0.017	-0.057
C3H0	-0.083	-0.048	-0.081
O2H1	-0.081	-0.017	-0.040
N3H0	0.016	-0.035	-0.124
N3H1	-0.085	-0.056	-0.087
N3H2	-0.040	-0.042	-0.072
O1H0	-0.108	-0.068	-0.077
S2H1	-0.032	-0.016	-0.053
N4H3	-0.075	-0.021	0.044
UNKN	-0.034	-0.015	-0.011

Table 4.2: This table shows the correlation values $[-1, 1]$ for a patch being a epitope patch and the #number of atoms belonging to the corresponding atom group. Each column shows the correlation for the corresponding data set generated by the 'N closest' algorithm.

5
Classification

This chapter describes the classification and training algorithm used during this thesis.

The aim of this project was to find areas on proteins which are likely to be an epitope. This was done by classifying patches, extracted from the surface (see chapter 4), as either a epitope patch or a non-epitope patch. The classification can be done in many different ways, the two main approaches is scoring and machine learning.

A scoring algorithm takes some attributes from the subject and gives it to a function which calculates a score representing a class. This requires knowledge of the subject and its classes and how the attributes weigh in on determining different classes. Usually these scoring functions are quite efficient and fast at classifying data, in addition they do not require any training.

5.1 Machine learning

Machine learning techniques can be successful when the classification subject is complex or when there is too little knowledge of the subject and the target class. The requirement is example data. The success of any machine learning technique mainly lies in the available data to train on.

In the case of this thesis, there is very limited knowledge about the conformational B-cell epitopes. In addition the available data sets is also limited, it is not known how comprehensive or biased the data is. It is however a representative data set of the known examples. Therefore a machine learning algorithm would be a good start and this is the direction of this thesis.

5.1.1 Neural networks

Artificial neural network (ANN) is a machine learning method which was discovered and developed in the 70s-80s, but due to very limited computational power this method was put on the shelf for many years, it was simply not possible to perform all the calculations in a reasonable time. In recent years ANNs have become more popular due to the increase in computational performance and development in parallel computing. Most of the computations in an ANN is matrix multiplication and can be calculated in parallel. There are many different ANN structures, all with different advantages. One of the most basic and efficient ANN structures is the

"feed forward neural network" (FFN) and this is the structure used in this project. The flow of connections is going only one way, from the input neurons to the output neurons.

5.1.2 Training

Training an ANN is basically optimizing the weights of the connections and thresholds of the neurons to the given pattern and target class. There are various ways to optimize these variables. One of the most popular is called "back-propagation". The basic idea is to compute the error of the ANN output and then calculate the gradient of the ANN towards the target output. Then update the weights and thresholds proportional to the steepest gradient descent and error.

Matlab has a training function called "trainscg"(Scaled conjugate gradient back-propagation), which takes advantage of the conjugate gradient method in order to find the steepest descent of the gradient. This is a faster method than the ordinary back-propagation, but it can also lead to faster convergence which can both be good and bad, in addition it must be possible to calculate the conjugate gradient which limits the method but works just fine with FFN. If the training method converges too fast, prematurely, then the resulting ANN will not reach its full potential and will give a bigger error in its output. On the other hand, if it is too slow then the training can go on for a very long time. It can also be the case of jumping between two or more local minimas, in such a case the weight and threshold update step is too large and the training algorithm will have a hard time converging to any one point. The scaled part in trainscg controls this step size in a different way than other methods, which allows a faster convergence and training [15].

The learning ratio can also be controlled by a learning rate parameter, which is multiplied with the weight and threshold update. For normal training the ratio is usually in the range $0.1 - 0.001$.

When training an ANN to recognize a pattern to then classify it by the ANN output, one feeds the ANN with the given example, calculates the ANN error and updates the weights and thresholds according to the training algorithm. Usually, there is more than one example to train on and since there is only one ANN, the examples has to feed to the ANN one at a time and calculate the update after each one or use the average error and gradient to update once all examples have been be fed to the ANN. The training will depend on the order sequence of the training examples fed, there are sequences which gives a better results than others. These are not known beforehand and therefore the common practice is to pick a random sequence. Such a data set is also called a 'batch' of training data. This is useful when there is too much training data that it does not fit in memory at one time, it can also speed up the training depending how the batches are processed. When using batches, there are basically two ways to assemble them. Either to divide the data into the batches prior to the training or to assemble the batches during the training. In both cases it is useful to assemble them randomly for the same reasons as before. One can argue which one is better, but it differs from case to case. In order to avoid

premature convergence when using batch training, one usually lowers the learning rate, otherwise the initial batches will be too influential on the ANN performance.

This project used Matlab's trainscg function together with batch training and accelerated with GPU computing. The batches are assembled by randomly picking patches from the data set, balancing the number of positive and negative patches to 50% each. This forces the ANN to prioritise the positive examples as much as the negative. More than 90% of the examples are annotated as negative (non-epitope), if there were no balancing in the batches it would be hard for the ANN to learn the desired patterns. It would probably just classify all examples as negative and receive great overall performance. The down side to the balancing is that it might cause an over fitting to the positive examples, hard mapping each positive examples to the positive class rather than learning the patterns.

5.1.3 GPU computing

In the training process the project used GPU acceleration instead of the standard CPU. The GPU allows for much more efficient parallel computations than the CPU and therefore performs the computations much faster. In ANNs, each computation within the same layer can be computed in parallel. But since the layers are dependent of each other, each layer has to be calculated in sequence. In this case the GPU was more than 50 times faster that the CPU, which was available, at calculating the largest ANN designed by this project.

5.2 Training data

This section shows how the two models are encoded into feature vectors which are accepted by the classification algorithm. The classification algorithm, which in this case is an ANN, will not accept feature vectors which change in length. It would also prefer the features in the vector to come in a consistent order, if they change around in a chaotic way it will become harder for the classification algorithm to learn the desired pattern.

5.2.1 By radius - Encoder

The biggest issue with the "by radius" patches is the inconsistent number of atoms within the patch. An ANN requires a fixed size feature vector containing the inputs, it cannot handle a feature vector which is varying in size. Therefore these patches must be transformed into a format which satisfies consistent size.

To solve this issue with varying number of inputs, a 3d pixel map was designed to hold the feature vector. The project started with the mid size data set and conducted analysis on the patches created and rotated (see table 5.1). From this analysis the project concluded a radius of 20 Å, which will cover all possible positions. This radius is used to create a pixel sphere, to which the patch will be mapped. Each pixel in the sphere covers the size of a 1 cubic Å, where all the sides has the

length of 1 Å. For example: The pixel which corresponds to this position $(x, y, z) = (3.456, 12.212, -14.556)$ will have the map position of $(3, 12, -15)$, all atoms in the range $([3, 4), [12, 13), [-15, -14))$ will be correspond to this pixel. In other words, the coordinates of each atom is floored. If two or more atoms is covered by the same pixel, information will be lost. Therefore the dimension of 1 Å was set, there were very few examples where two atoms were closer than 1 Å from each other and within the mid size data set, no such collisions ever occurred.

	Min:	Max:
Z-axis	-19.16049161	18.71747987
X-axis	-18.85064611	18.77314049
Y-axis	-18.75570473	19.1335069

Table 5.1: Patch dimensions, this table shows the max and min positions of atoms in the mid size data set.

This sphere of pixels is then concatenated down to one long feature vector. Each index has either the value 0 if there were no atoms present, or a value $1 - 14$ representing the different atom groups given by Triominos.

5.2.1.1 Algorithm

Given a normalized patch and a radius, create a 3d cube array A, with the side length of 2 times the radius and the initial value 0. For each atom, take the integer value of its coordinates (x,y,z) (floor the floating point value) and add the radius to the (xi,yi,zi) to make them positive. Then use the integer coordinates to place its atom group number in A[xi][yi][zi]. Create the feature vector V by iterating over A. Start at index [0][0][0] in A, then check if its coordinates are within the sphere with the given radius and originating at the starting atom (center) of the patch. If it is, then add it to V, if not then discard it and continue until all indexes have been visited.

5.2.2 N closest - Encoder

The N closest feature vector is not as complicated as the by radius feature vector. This vector is simply a vector of the coordinates and atom groups of each atom in the patch. It starts with the origin atom, at the center of the patch, adds the atom group followed by x,y and z coordinates of that atom. Then it takes the atom closest to the origin and repeats that same pattern until all atoms in the patch are represented in the feature vector.

Compared to the by radius feature vector, this vector is much shorter, only 4 times the number of atoms in the patch. For example the mid data set has the fixed patch size of 200 atoms, see Appendix A.2, which gives a feature vector of length 800 nodes. While the by radius feature vector for the mid data set has a feature vector of length 33401, see figure 5.1.

5.2.3 Decoder

There is only one output neuron from the ANN and is simply a 1 for a epitope and 0 for a non epitope. The raw output is a floating point number which is rounded to 0 or 1, there is no constraint on range of this output. When all the patches of an antigen is classified it counts the number of times an atom has been classified as an epitope and non-epitope, which will give the final prediction of the possible epitopes on the surface.

5.3 Classifier

To determine the structure of the ANN one can calculate the pattern capacity(how many patterns it can efficiently store) of an ANN or follow a trial and error procedure. Calculating the capacity of an ANN becomes very complex and inaccurate when using more than one layer of weights, therefore the project chose to test its way forward with different settings in order to find a suitable ANN structure.

A protein epitope example was chosen from the mid data set to be the initial testing example, it is represented as the PDB id '1a14' in the chain 'N'. This was done in order to find an ANN structure which can learn and recognize the patches of one single example, this will be the base for continuing with more examples and finally the whole data set. There is no point to train an ANN on the whole data set if it cannot fit a single example. It might be the case that the ANN which can learn a single example is too small to fit more example data, but it is a starting point.

The following two sections will present the process of finding suitable ANN structures, with some of the training results, for the two different models.

5.3.1 Evaluation

Evaluation of an ANN can be done and interpreted in many different ways. This project has mainly looked at three types of evaluation. First, the performance of the ANN during the training which indicates how well the ANN performs on training data. Second, after the training is done the ANN is to classify the training examples but with different rotations, from this one gets booth the performance(error of the ANN) and a confusion matrix showing the ANN classification outputs against the target values. Last, there is a test on classifying example proteins which are not in the training data, as it is the ultimate goal to be able to classify new proteins.

The first evaluation is done during the training and is the mean of the squared errors of the ANN output, which is called the performance of the ANN. This indicates how well the ANN is performing at classifying the training data, in other words how big the classification error of the ANN is.

The second evaluation tests how good the ANN is at recalling the training data but with different rotations. The best way to evaluate this is to look at an confusion plot, see figure 5.3, where one can see more specifically if the ANN is just very good

at classifying the negative or positive data. This evaluation of different rotations also gives information whether or not the ANN is able to capture the structure, regardless of the orientation of the raw data from the protein.

The last evaluation is to test whether the ANN is able to classify epitope areas in proteins not in the training set. This can be done in a few different ways, one of the most common ways is to exclude a percentage of the examples from the training data set, to use as a validation set. This is usually in the range of $5 - 15\%$ of the total number of examples. This project has 61 examples in the mid data set and therefore used 6 of them as a validation set for the classifiers. These were chosen randomly, but remained consistent over the different ANN structures and training sessions. See table 5.2 for the validation set and Appendix A.2 for the complete data set.

PDB id	chain
3b2u	B
2r29	A
2dqf	C
1v7m	V
1ndg	C
1eo8	A

Table 5.2: The validation set for the mid data set. These PDB examples were randomly chosen from the mid data set. The chain column specifies the antigen chain in the file.

5.3.2 By radius classifier

Initially a small ANN containing only one hidden layer with 10 neurons was constructed. The ANN was trained on one example protein with one epitope and aimed to classify the epitope patches. See table 5.3 for the results. The ANN design of three hidden layers with 1000 neurons each got the best result and is the structure of choice.

Layers	Performance
10	bad
100	bad
1000	0.32
100/100	0.067
1000/100	0.077
1000/1000	0.1
1000/1000/1000	0.07
1000/1000/1000/1000	0.09

Table 5.3: This table presents the performance of artificial neural networks (ANN). The "Layers" indicates how many layers there are and how many neurons they are containing. The "Performance" indicates the ANNs performance after 50 iterations of training, lower is better. The performance is calculated by the mean squared error of the ANN. The 'bad' rows is results which showed so bad performance that it was not noted.

The next step is to determine how many rotations of each example are needed. The aim for the ANN is to identify any rotation of the epitope patches, not only those in the training data. As before the project started with few rotations and iteratively increased the number of rotations. The project found that 72 rotations got an ideal result for one protein example, with a slightly bigger ANN of 1200 neurons in each of the three hidden layers. The down side where the harddrive and memory size of the data generated, therefore 36 rotations gave a sufficient result with manageable data size.

Further on the ANN size was increased to 1400 neurons in each of the three hidden layers, see figure 5.1. This is the maximum ANN size to fit in the 4 GB GPU memory available. The main factor to the size of the ANN is the input vector, in this case it is 33401 number of nodes. Hence there are $33,401 * 1,400 = 46,761,400$ number of weights from the input to the first hidden layer. One can compare this to the rest of the weights in the ANN, $1400^2 + 1,400^2 + 1,400 = 3,921,400$. It is easy to see that in this case the input vector is limiting the ANN size. The final results are presented in chapter 6.

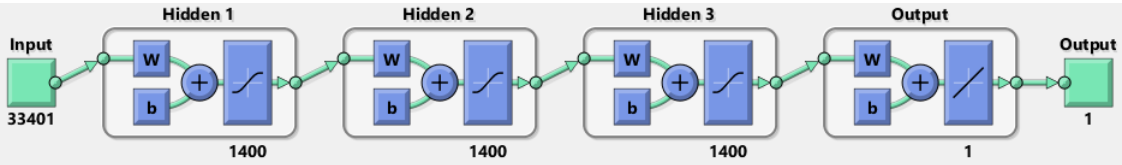

Figure 5.1: The visualization of the 'by radius' ANN from Matlab.

5.3.3 N closest classifier

Starting off where the 'by radius' finished, with an in-1400-1400-1400-out ANN, then training on the same protein in '1a14' with chain 'N', it got close to a perfect

recall at 36 rotations, just as the other model. It even got a perfect result down to 5 rotations.

One big difference between the two models, is the size of the input vector see figure 5.1 and figure 5.2. In the 'by radius' model the input vector is significantly bigger than in the 'N closest' model. However this does not necessarily mean that the hidden layers can be made smaller and less complex in the 'N closest' model, but the results so far suggest that there is no need to have such a big ANN structure in this case.

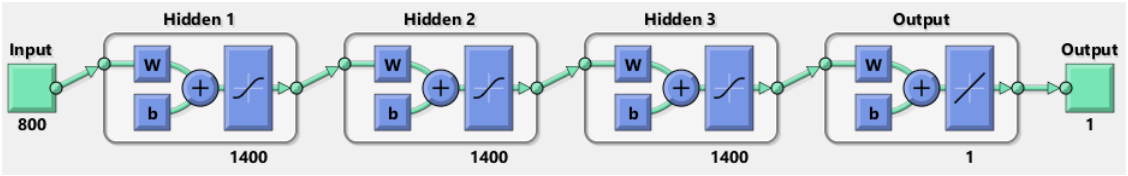

Figure 5.2: The visualization of the 'N closest' ANN from Matlab.

It is hard to evaluate the performance value. For example when there are few positive values and a lot of negative values, then classifying all to be negative will give a quite good performance value. A better measurement is to look at the confusion matrix, where all the true positives, true negatives, false positives and false negatives are listed. In figure 5.3 is the results from the evaluation of a ANN trained on '1a14-N' and the figure 5.4 shows the training progress. The training progress shows that the performance rapidly improved in the beginning and then almost stagnated before finishing, which can indicate that it converged to a optimum or overfitted the data.

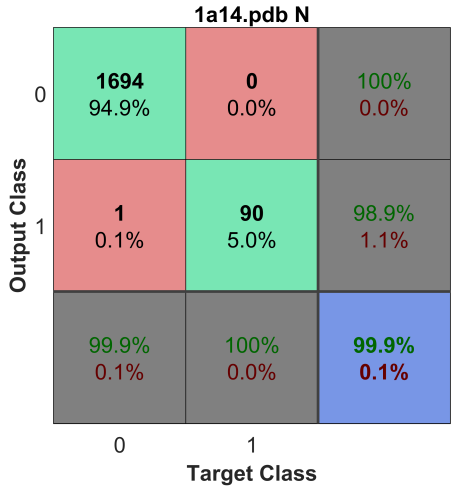

Figure 5.3: The confusion matrix from the final 517:th iteration figure 5.4, when evaluated with the data from '1a14-N' with 5 rotations. Class 0 indicates a non epitope and class 1 indicates a epitope patch.

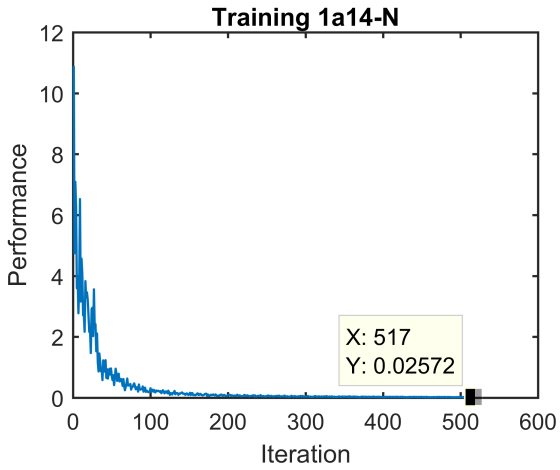

Figure 5.4: The performance over each iteration in the training of the ANN in figure 5.2. Training on '1a14-N' with 36 rotations.

6

Results

This thesis has shown in two different ways how to model epitopes on the surface of proteins. Then it showed to use these models to train artificial neural networks (ANN) in order to classify the models as epitopes or non-epitopes. The result is classifiers which can recall the proteins which it has trained on, however it was not successful with classification of proteins outside of the training examples.

This chapter shows the results from the classification algorithms developed during this thesis. Statistics from the validation and graphs showing the performance development during the training of different key ANNs are shown. In addition some examples of the recall classification will be displayed as the final epitope prediction plot, on the protein surface.

6.1 Classification

In this section the results from the different classification algorithms will be displayed. Only those ANNs which are interesting and most successful have their results on display.

6.1.1 By radius

The feature vector from 'by radius' model became large, when modeling the mid size data set it contained 33401 points. The ANN structures which were developed, see section 5.3.2, became limited to the available memory in the GPU. In addition, training this ANN was very time consuming, even on the GPU. Together these two factors may have limited the results, for instance the project was not able to train the ANN on more than 10 examples from the mid data set 6.1 which is suboptimal.

ANN-Iteration	TN	FP	FN	TP	Sens	Spes	TNr	TPr	Tot	AUC
1400-5ex-4k	4998	517	325	20	0.037	0.939	0.906	0.058	0.856	0.482
1400-5ex-6.5k	4806	709	310	35	0.047	0.939	0.871	0.101	0.826	0.486
1400-10ex-8.5k	4585	930	248	97	0.094	0.949	0.831	0.281	0.799	0.556
1400-10ex-10k	4659	856	251	94	0.099	0.949	0.845	0.272	0.811	0.558
1400-10ex-12k	4691	824	250	95	0.103	0.949	0.851	0.275	0.817	0.562
1400-10ex-8.5k	3124	696	232	53	0.071	0.931	0.818	0.186	0.774	0.501
1400-10ex-10k	3178	642	232	53	0.076	0.932	0.832	0.186	0.787	0.508
1400-10ex-12k	3206	614	231	54	0.081	0.933	0.839	0.189	0.794	0.514

Table 6.1: This table shows the performance of the 'by radius' classifier. The ANN-Iteration column shows the ANN structure (1400 means 1400/1400/1400 as in 5.1), in this table all the ANNs have the same structure. The structure is followed by the number of examples it used in the training. 5ex is the top 5 PDB ids in the Mid data set A.3 and respectively 10 ex means the top 10 in the same data set. At the end there is the iteration in the training. The training set of the top 10 is overlapping with one PDB id '1eo8', therefore there is a second evaluation with the same evaluation data set excluding '1eo8' which is noted in **bold**. True negatives (TN), false positive (FP), false negative (FN), true positive (TP), Sensitivity (Sens), specificity (Spes), true positive ratio (TPr), true negative ratio (TNr), total success rate (Tot).

The training graph in figure 6.1 shows how the performance (lower is better) developed during the training of the classifier. It reveals the large initial performance which then rapidly decreases to 0.2. The performance still improves over time but stays in the range 0.1-0.2 for a long period. It can be hard to read details from these graphs, however it is possible to compare the fluctuations in figure 6.1 with the two figures from the N closes model 6.7 and 6.8, where it is observable that the fluctuation is much lower in the N closes model. This indicates that the by radius model is more complex to train or requires a different ANN or additional data.

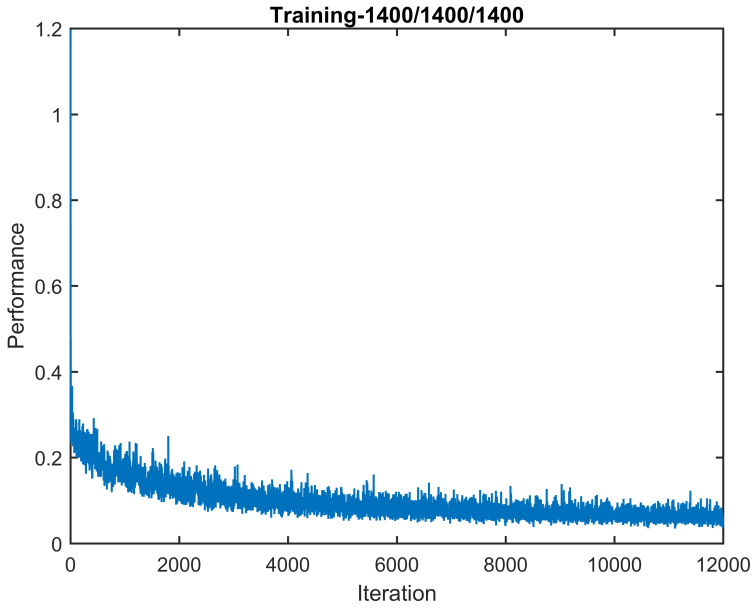

Figure 6.1: The performance over each iteration in the training of the ANN in figure 5.1. Training on the first(when sorted in alphabetic order) 10 PDBs in the mid data set A.3 with 36 rotations.

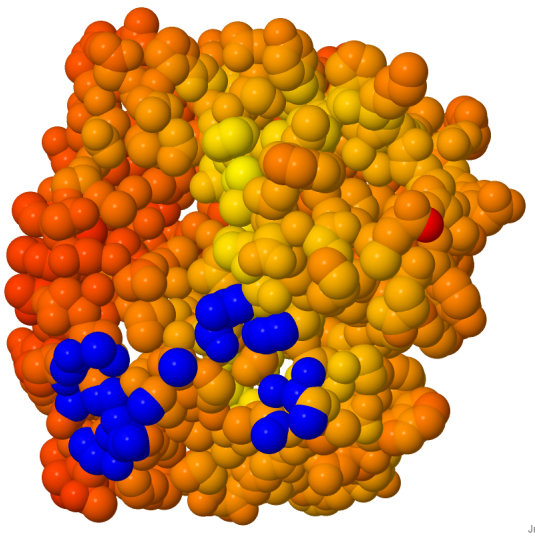

Figure 6.2: This figure shows the surface atoms from chain N in PDB 1a14. The colors indicate the predicted epitope sites, ranging from low (red) to high (yellow) prediction. The blue shows the actual confirmed epitope. This prediction is performed by the 1400-10ex-12k ANN in table 6.1. The 1a14-N is modeled with 5 rotations (training is done with 36 rotations) and is in the training set of the ANN. This is a demonstration of the recall of the ANN. The prediction color range is relative to the highest predictions, which means that the highest prediction will be yellow even tho it might be predicted with a low confidence.

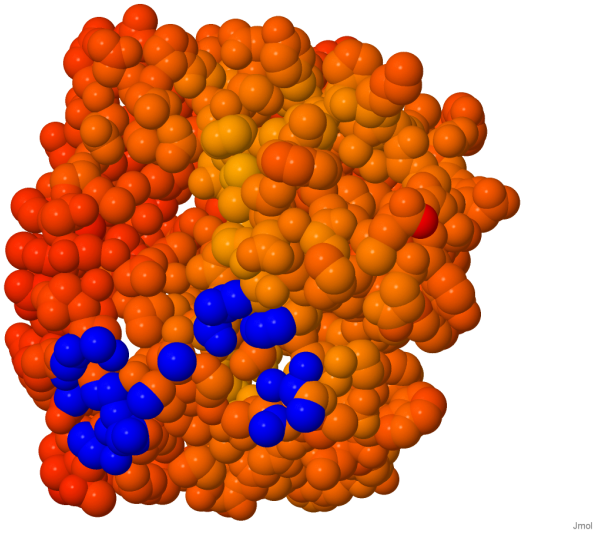

Figure 6.3: This figure shows the same protein as figure 6.2. This figure uses a set threshold for the color range, which will yield a yellow color only when there is a high confidence of an epitope.

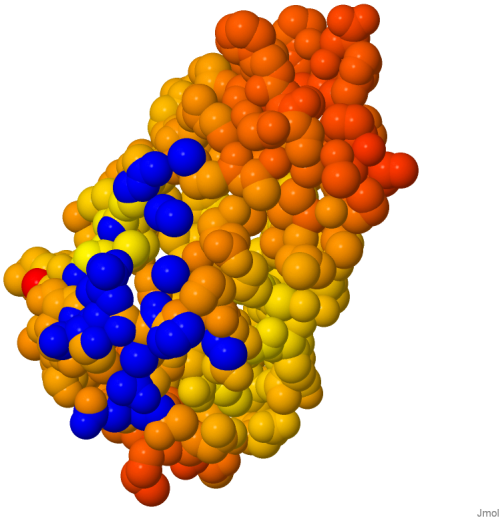

Figure 6.4: This figure shows the surface atoms from chain V in PDB 1v7m. The colors indicate the predicted epitope sites, ranging from low (red) to high (yellow) prediction. The blue shows the actual confirmed epitope. This prediction is performed by the 1400-10ex-12k ANN in table 6.1. The 1v7m-V is not in the training set and is modeled with 5 rotations. The prediction color range is relative to the highest predictions, which means that the highest prediction will be yellow even tho it might be predicted with a low confidence.

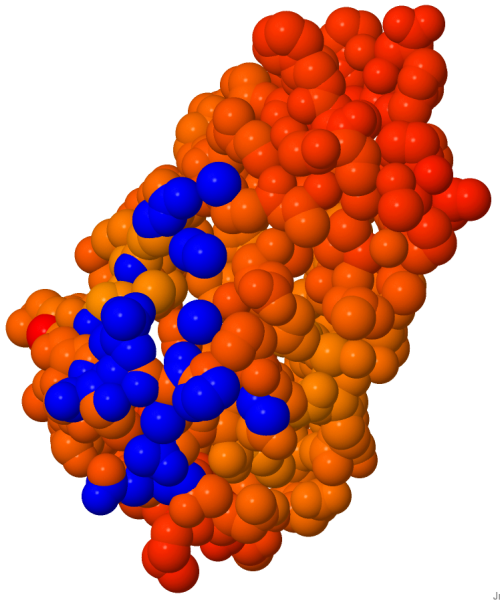

Figure 6.5: This figure shows the same protein as figure 6.4. This figure uses a set threshold for the color range, which will yield a yellow color only when there is a high confidence of an epitope.

6.1.2 N closest

Here is the results from various ANN structures classifying the 'N closest' model 5.2.1. This model created a much smaller feature vector, hence the training went much faster and it also performed better with smaller ANNs.

It is not easy to compare details in these graphs but it is possible to get a general view of the training process. The training graphs in figures 6.7, 6.8, 6.9, 6.10 and 6.11 is quite similar, they all have low fluctuation and they show slow improvements after their fast initial improvements. Figure 6.6 is displaying the graph in more detail and shows more fluctuation as the graph in figure 6.1, but in general is shows a similar curve as the other training graphs.

ANN-i	TN	FP	FN	TP	Sens	Spes	TNr	TPr	Tot	AUC
10-1k	13537	5165	1747	647	0.111	0.886	0.724	0.270	0.672	0.497
10-3k	15223	3479	2037	357	0.093	0.882	0.814	0.149	0.739	0.481
10-5k	15427	3275	2060	334	0.093	0.882	0.825	0.140	0.747	0.482
10-7k	15783	2919	2108	286	0.089	0.882	0.844	0.119	0.762	0.481
100-1k	16475	2227	2120	274	0.110	0.886	0.881	0.114	0.794	0.497
100-3k	17204	1498	2206	188	0.112	0.886	0.920	0.079	0.824	0.499
100-5k	17594	1108	2262	132	0.106	0.886	0.941	0.055	0.840	0.497
100-7k	17707	995	2270	124	0.111	0.886	0.947	0.052	0.845	0.499
300-1k	16686	2016	2190	204	0.092	0.884	0.892	0.085	0.801	0.488
300-3k	17593	1109	2272	122	0.099	0.886	0.941	0.051	0.840	0.495
300-5k	17906	796	2305	89	0.101	0.886	0.957	0.037	0.853	0.497
300-6k	17884	818	2308	86	0.095	0.886	0.956	0.036	0.852	0.496
700-1k	14544	4158	1817	577	0.122	0.889	0.778	0.241	0.717	0.509
700-2k	16254	2448	2107	287	0.105	0.885	0.869	0.120	0.784	0.494
700-3k	16595	2107	2177	217	0.093	0.884	0.887	0.091	0.797	0.488
700-3.5k	17021	1681	2214	180	0.097	0.885	0.910	0.075	0.815	0.492
1400-1k	12947	5755	1706	688	0.107	0.884	0.692	0.287	0.646	0.489
1400-2k	15217	3485	1940	454	0.115	0.887	0.814	0.190	0.743	0.501
1400-3k	15798	2904	2039	355	0.109	0.886	0.845	0.148	0.766	0.496
1400-5k	16290	2412	2132	262	0.098	0.884	0.871	0.109	0.785	0.490
4000-1k	11688	7014	1496	898	0.114	0.887	0.625	0.375	0.597	0.500
4000-3k	13373	5329	1715	679	0.113	0.886	0.715	0.284	0.666	0.499
4000-5k	13718	4984	1763	631	0.112	0.886	0.734	0.264	0.680	0.498
4000-10k	15115	3587	1920	474	0.117	0.887	0.808	0.198	0.739	0.503

Table 6.2: This table shows the results from the ANNs classifying the N closest model. There are 6 different ANN structures shown, each with at 4 different stages in training. The ANN used the A.3 as training set, excluding the evaluation set 5.2 which is used as the evaluation set in this table. ANN-Iterations column shows the ANN structure followed the iteration stage in the training. True negatives (TN), false positive (FP), false negative (FN), true positive (TP), Sensitivity (Sens), specificity (Spes), true positive ratio (TPr), true negative ratio (TNr), total success rate (Tot), area under the ROC (AUC).

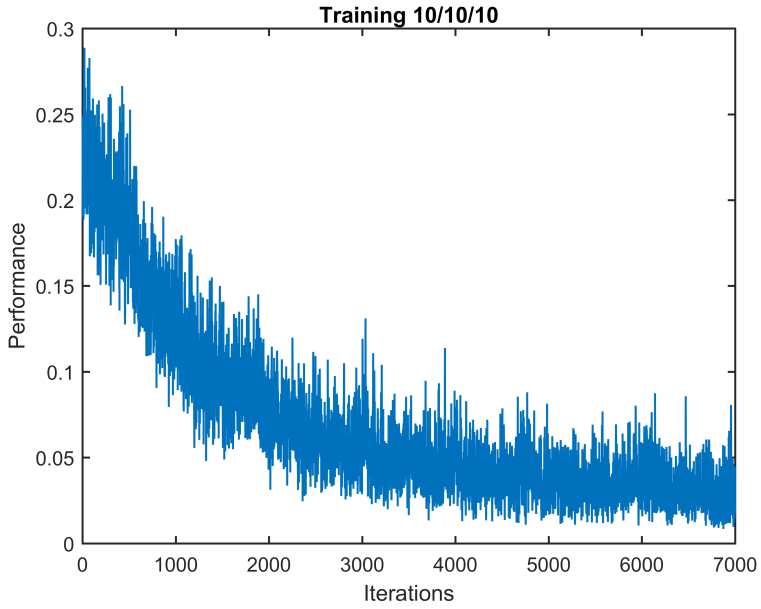

Figure 6.6: The performance over each iteration in the training of the ANN in figure 5.2 with 10 neurons in each hidden layer. Training on the mid data set A.3 with 18 rotations.

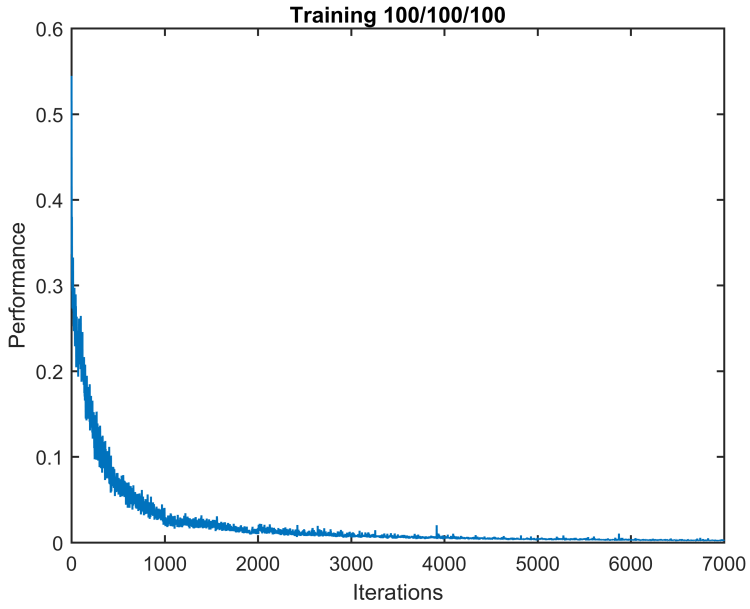

Figure 6.7: The performance over each iteration in the training of the ANN in figure 5.2 with 100 neurons in each hidden layer. Training on the mid data set A.3 with 18 rotations.

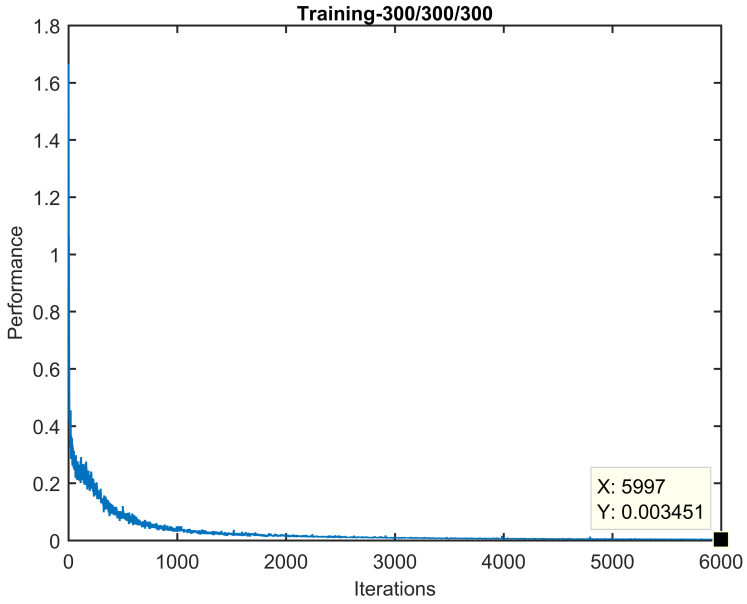

Figure 6.8: The performance over each iteration in the training of the ANN in figure 5.2 with 300 neurons in each hidden layer. Training on the mid data set A.3 with 18 rotations.

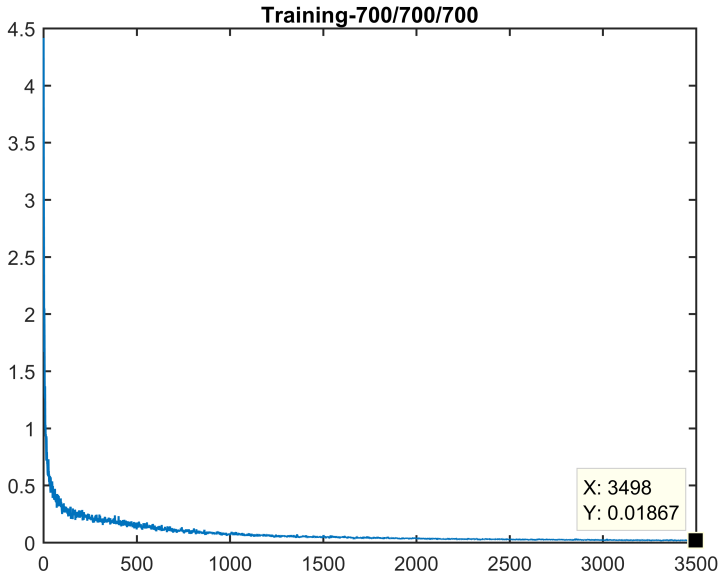

Figure 6.9: The performance over each iteration in the training of the ANN in figure 5.2 with 700 neurons in each hidden layer. Training on the mid data set A.3 with 18 rotations.

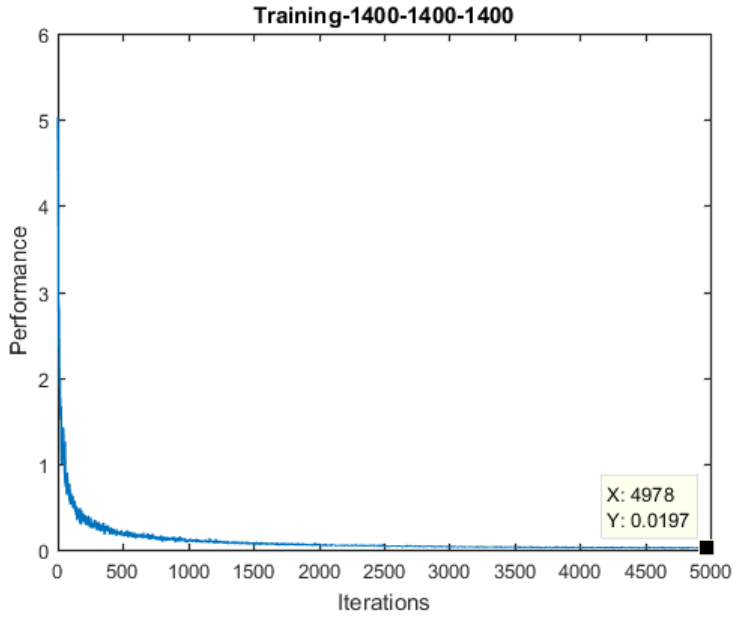

Figure 6.10: The performance over each iteration in the training of the ANN in figure 5.2. Training on the mid data set A.3 with 18 rotations.

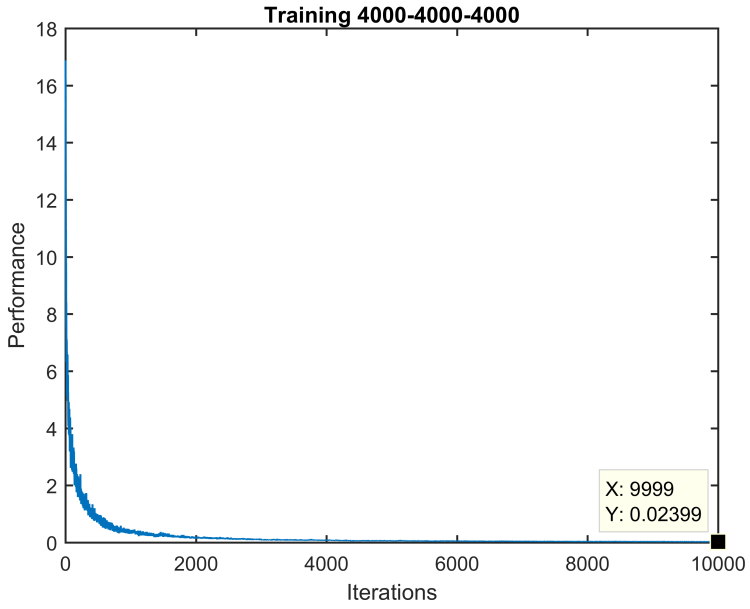

Figure 6.11: The performance over each iteration in the training of the ANN in figure 5.2 with 4000 neurons in each hidden layer. Training on the mid data set A.3 with 18 rotations.

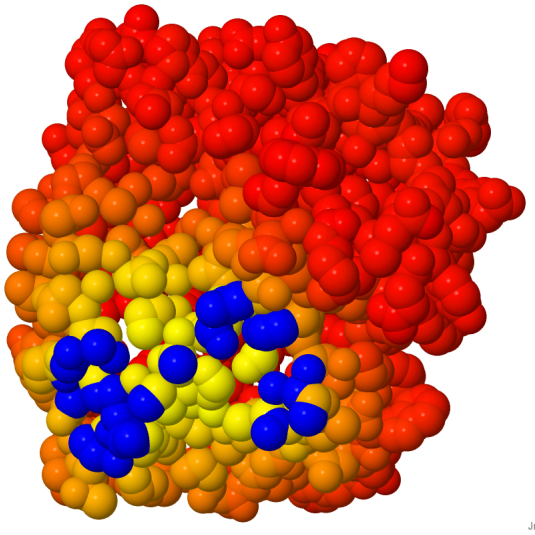

Figure 6.12: This figure shows the surface atoms from chain N in PDB 1a14. The colors indicate the predicted epitope sites, ranging from low (red) to high (yellow) prediction. The blue shows the actual confirmed epitope. This prediction is performed by the 1400/1400/1400-5k ANN in table 6.2. The 1a14-N is modeled with 5 rotations (training is done with 18 rotations) and is in the training set of the ANN. This is a demonstration of the recall of the ANN. The prediction color range is relative to the highest predictions, which means that the highest prediction will be yellow even tho it might be predicted with a low confidence.

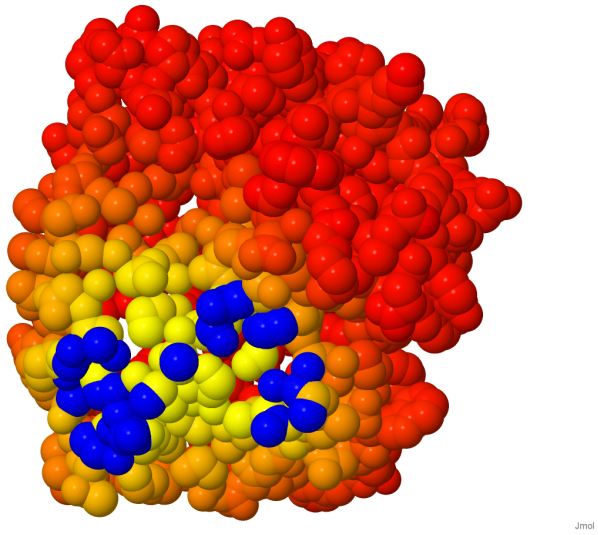

Figure 6.13: This figure shows the same protein as figure 6.12. This figure uses a set threshold for the color range, which will yield a yellow color only when there is a high confidence of an epitope.

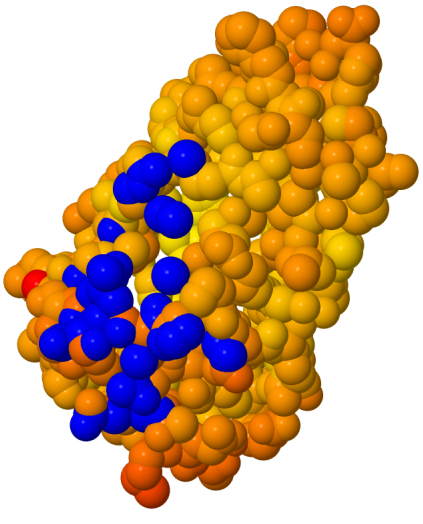

Figure 6.14: This figure shows the surface atoms from chain V in PDB 1v7m. The colors indicate the predicted epitope sites, ranging from low (red) to high (yellow) prediction. The blue shows the actual confirmed epitope. This prediction is performed by the 1400/1400/1400-5k ANN in table 6.2. The 1v7m-V is not in the training set and is modeled with 5 rotations. The prediction color range is relative to the highest predictions, which means that the highest prediction will be yellow even tho it might be predicted with a low confidence.

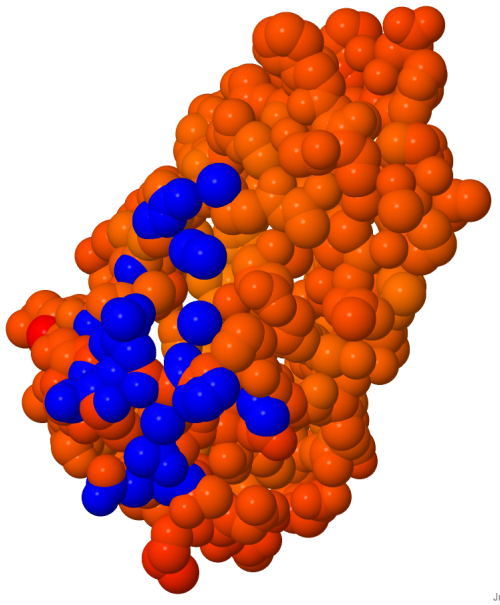

Figure 6.15: This figure shows the same protein as figure 6.14. This figure uses a set threshold for the color range, which will yield a yellow color only when there is a high confidence of an epitope.

7
Discussion

This chapter discusses the thesis, process and related work, based of the thesis results.

7.1 Results

This section will discuss the results from the classifiers developed during this project. Further discussion of topics surrounding the algorithms, data and training will be brought to light later in this chapter.

The tables in chapter 6 shows performance up to 85% correctly classified patches. However, neither of the two models were able to successfully classify the epitopes from the validation set, table 5.2. One can read on the last row in the table 6.1 that the ANN was only able to classify 18.9% of the epitope patches as an epitope patch (TPr) and 83.9% of the non epitope patches as non epitope patches (TNr). This leaves a sensitivity of 8.1%, which means that only 8.1% of the patches classified as an true epitope patch is an actual confirmed epitope patch. There is marginally better results for the 'N closest' model, shown in table 6.2, it achieved sensitivity of around 11% which is still bad. This is illustrated in figures 6.4, 6.5, 6.14, 6.15, where one can see the V chain from the PDB '1v7m'. The goal is to find distinct yellow areas as in figure 6.13, but this is clearly not the case.

The results show that the classifiers are not able to find epitopes in new subjects, but it does show that it is good at recalling the training examples, even when they come in different rotations than in the training set. Figure 6.12 and 6.13 illustrates the result from the 'N closest' classifier with the input from 1a14, which is in the training data set. The result is almost perfect in contrast to the 'by radius', which is illustrated in figure 6.2 and 6.3. The later is not able to recall the epitope which it was trained on, however when training with fewer PDB examples 1-4, this recall becomes similar to the 'N closest' classifier. This can indicate either that the training is too short or that the ANN is too small to fit the patterns from the data.

One of the goals of the classifiers was to be able to identify any rotation of a patch. As mentioned in the last paragraph the 'by radius' classifier are successful at identifying any rotation of the patches when trained on few examples. And it seems possible that it actually learns the patterns at any rotation due to the tests where the training data had fewer and more than the 36 rotations which was used in the final classifier, see section 5.3.2. When fewer rotations were used, the performance at recalling

the training examples also went down and vice verse. This was not the case of the 'N closest' classifier, at least not to the same extent. When training with the 'N closest' model, it performed very well at recalling the training examples down to 3-5 rotations when training on few (1-4) examples. There were no testes with this few rotations when training on the whole mid data set. It can be the case that the ANN is learning the order of the atom groups in the feature vector, see section 5.2.2, of the 'N closest' model, instead of learning the relative positioning of the atoms. This needs more investigation before any conclusions can be drawn, it is hard to analyse which features in the vector the ANN is relying on. The analysis , see section4.2.3.1, shows that there is very small and non-existent correlation between the atom group count in a patch and the patch being apart of the epitope, however this analysis does not regard the order of the atom groups. It is also possible that the order of the atom groups in the feature vector acts as a unique identifier for the patches. For the mid data set there are 200 atoms in each patch, each atom is labeled with one of 14 atom groups, which yields 14^{200} possible atom group patterns.

7.2 Model

This thesis has shown two epitope models, the 'by radius' and 'N closest'. They take a similar approach of modeling the epitope as a continuous patch of atoms on the surface of a protein, as described in chapter 4. They also have the same basic requirements, that a patch should cover the target epitopes and not to be too big for the target protein. Fulfilling these two requirements was done by dividing the data into smaller overlapping subsets, where the data examples in one set have similar epitope and protein size. It was just a matter of finding a good radius or atom count to fulfill the requirements. In these first steps, the two models are quite similar and they use the same data sets. However when it came to assembling the feature vector, there are some big differences.

The algorithms for creating the feature vectors can be found in section 5.2. The 'by radius' model has a much more complex way of assembling the feature vector than the 'N closest' model. It basically creates a pixel sphere with the same dimensions as the patch, then places each atom in the patch to the corresponding pixel in the sphere. This could solve one of the issues which the 'N closest' feature vector has, which is mentioned in section 7.1, because this vector is not as sensitive to the order of the atoms. On the other hand, it consumes much more memory space which can limit the performance of the classifier.

7.3 Neural network

ANN was the classifier of choice, which is excellent at classifying patterns in data when there is much example data to train on. When it is successful the algorithm becomes robust and precise. Robust in the sense that it is quite resilient to noise in the data and precise such that once it is trained with a suitable structure the classifications can be very fine grained. In addition it is very adaptable, it can be

trained with new data as it becomes available and its structure can also be changed, there is also the possibility to merge and stack different ANNs in order to get some control and understanding of the algorithm. The downside of ANN is its parameter complexity (hyper parameter), there are many parameters to tune in an ANN, more than in most other machine learning algorithms. The structure and architecture is possibly the most complex and most important part. There is a wide range of architectures that has been developed over the years, they have different dynamics and are suited for different problems. This project choose one of the most general and simplest architectures, feedforward neural network (FFN), which is usually a good place to start for any classification problem where one has limited knowledge. The next part to consider is the structure, how many layers and how many neurons there are in each one of them. In itself this is an optimization problem and with limited knowledge one has to turn to the trial and error method. As mentioned in section 5.1, it is possible to calculate the capacity for an ANN and that this can be very complex and it can be hard to find any reliable result. In addition one might not know how many patterns one has to fit into the ANN or how complex the patterns are. When it comes to deciding on the number of layers in the ANN, it can be worth to note that one can split the problem into smaller pieces and to do the classification in steps. As mentioned, it is possible to stack or connect one ANN with another. This is when the output from the first goes into the next one, meaning that the second one does another classification of the first one's output. This way one can solve complex classification problems without the need of solving it in one whole sweep [7]. The downside to more layers is more neurons and weights, in a big ANN such as the one in figure 5.1 this becomes a memory and time limitation. Bigger ANN does not only require more space in memory but also takes more time to process, which will in turn increase the training time of the ANN. This project is on a strict time limit which makes it hard to evaluate complex ANNs and to motivate additional hardware in time to be able to test bigger structures or train additional variations.

There are more parameters and features which look less important but can have a big impact on the performance and training time of the ANN. The neurons come in different variants, as with the architecture there is a wide range of neurons which gives different perks and performance to the ANN, some are more robust, flexible or faster to train. They are usually picked together with the training method. The training method is basically an optimization algorithm, which is specialized on optimizing different ANN architectures and neurons. Usually the training methods will use the gradient of the ANN is some way, this project choose the Scaled conjugate gradient back-propagation, section 5.1.2, which achieves state of the art training performance for training FFN [15].

Closely connected to the training method is the learning rate, the rate of training. This controls how big change the training method is allowed to make to the weights and thresholds. There is a balance where one does not want to make too big steps where the convergence of the ANN may come too soon and become suboptimal and stuck in a local optima. On the other hand, too small step can as well easily get stuck in local optima and the training can take much longer time. Again there is

need for some trial and error, one has to test what works best.

The learning rate is also dependent on how the training is done on the data. When there is a small and manageable amount of data which can fit in memory, then the training can be done with all the data in one batch. When there is only one batch the learning rate can be kept quite high without being biased towards any of the data (assuming that the data is not biased). However, when the data does not fit in memory one has to divide the data into smaller batches which will fit in memory, which is the case in this project see section 5.1.2. When there are several batches the training can become biased towards the first batches in the training if the learning rate is too high, therefore one has to lower the learning rate to avoid this issue. This was also done in the project and one can see in the results, see chapter 6, that the performance in the training plots fluctuate up and down. Bigger fluctuations can mean that the learning rate is high and less fluctuation that the learning rate is low, it can also be the case that the data is very different in the different batches which will also cause the fluctuations. The later can also be managed with a lower learning rate and possibly a bigger ANN to fit the data diversity.

In this project the ANNs were used for supervised learning, it means that one has training data which is annotated with the target classes to classify. In other words, the training is supervised and controlled. Another way to do it is unsupervised learning, this means that the machine learning algorithm must identify patterns in the training data on its own, without any help from the outside. One implementation area of this is clustering, where one uses a machine learning algorithm, possibly an ANN, to find either a set number or arbitrary number of clusters in the given data. This was not used in this project, but it would be interesting to see if there are any clusters within the patches in the data set. The project did some correlation analysis on the patches, section 4.2.3.1 which will be discussed further in section 7.5.

7.4 Data set

Data is the foundation of this project and in this area of research. It is obvious that without any qualitative data it would be hard to do any B-cell epitope prediction. Gathering data was not in the scope of this project, therefore data sets assembled by prior projects where a good way to access the needed data. This project has a combined data set from [16] and [10], which contains 126 examples, see section 3.4. According to the authors these data sets are representative and of good quality for conformational B-cell epitopes. This is true for the conformational epitopes which have been discovered and confirmed, which is a very small part of the total number of possible conformational epitopes. The Immunology Epitope Database (IEDB)[1] has records of just above 3000 conformational B-cell epitopes in its data base and just over 260 000 linear B-cell epitopes. It is quite reasonable that 126 could be representative when all duplicates, low quality and redundant examples are removed. However, "As crystallographic studies of antibody-protein complexes

[1] www.iedb.org

have shown, most B-cell epitopes are discontinuous." [9], which tells another story and indicates that no one has any idea if it is possible to create a true representative data set of the data available. This is a huge problem for research in this area, but due to this limited knowledge this shows even more the value of a good prediction tool.

The problem of the limited amount of data shows throughout the whole field. [11] discusses the issue where it is hard to evaluate the different prediction methods, due to the lack of verification data. In most tools developed there is usually only one confirmed epitope area on the epitope, but it is more than likely that any antigen has a lot more than one antigen. Each human or mouse has its own immune system, they might be similar but it is fair to say that it would be an achievement to find two identical immune systems with the same antibodies. This means that if a prediction tool flags more than the confirmed epitope area on the antigen, it would be difficult to confirm it. It might be better to just evaluate the tools by checking if they found the confirmed epitope and just disregard any other indications on the antigens. In addition one could collect known proteins which are accepted by the immune system, such as red blood cells, antibodies or any other molecule which the immune system should not attack and therefore is epitope free. One could then test the prediction tools on this data set to evaluate its performance to identify non epitope areas, which is just as important as flagging epitopes. This project did not do this non-epitope evaluation due to lack of assembled data sets and that gathering new data was outside of the project scope.

The issue of non-epitope areas is also bound to the training of prediction tools, both for this and prior projects. It is not easy to find what other researchers use as negative examples (non-epitope), some are simply taking the same approach as this project, to just treat all patches on the antigen which is not confirmed epitopes as a negative example and the confirmed epitope patches as an positive patch [1]. As mentioned above, this is a questionable approach due to the inability to confirm the non-epitope areas. This approach will probably lead to much noise in the training data, data which is annotated with the wrong class. In the case of this project, it is more than probable that the data set with negative annotation contains very much noise. A preferred way is to use confirmed data in both cases, however this would require data gathering which again was not in the scope of this project.

Not only is the data of questionable quality, it is also very biased. More than 90% of the training data is annotated as non-epitope, see section 5.1.2, which can make it harder for the ANN to train on. The batch balancing done in the project is one way to avoid biased training, but as mentioned in section 5.1.2 this can cause over fitting the epitope patches. A minor attempt to decrease the over fitting was done by defining the epitope patches as a patch containing 80% of the atoms in the confirmed epitope, see section 4.2, which increases the number of epitope patches and decreases the number of non-epitope patches. This might not be enough and it would be preferred to find more confirmed epitope examples to even out the positive-negative ratio.

7.5 Analysis

The correlation analysis in section 4.2.3.1 counted the number of atoms belonging to each atom group in each patch of the data sets. This is a different analysis method than what other projects have used. Usually there is a log odds ratio of the amino acid residues in the patch model, for example [8]. The problem with amino acids is that they are man labeled collections of atoms and it is very improbable that the whole amino acid is exposed at the surface. Usually there is a part of a single amino acid, a few atoms which binds to the antibody, it is rare that the whole amino acid is binding. The binding is rather to atoms exposed on the surface and that seems like a more logical substance to count. This project did count the atom groups, which brings more information about the atoms (see section 4.1.1). The analysis show that there is no clear correlation between any of the atom groups and the epitope patches, the project did however not analyse any joined atom group correlations.

7.6 Related work

The performance of an classifier depends on which statistics is considered. If we where for example just looking at the total success rate (see table 6.2), this classifier does perform better than the state of the art prediction tools. But this statistic is not telling the truth, one should look at the sensitivity and specificity, those are the statistics which can reveal the true performance of the classifier. Most other articles measures the performance by a sensitivity-specificity ratio (ROC) and calculates the area under the curve (AUC), this also gives a good measure of how well the classification is performing. This AUC value is calculated using the results from the evaluation, which means that the AUC value depends on the evaluation set. Different evaluations sets may give different AUC values and it can also be the case that for an individual example in the evaluation set the AUC and the performance is much better than the average. Ellipro [1] reports a best AUC of 0.732, which is a good performance for this problem, however the average AUC of Ellipro is 0.528, where 0.5 is a coin flip. Discotope 2.0 [11] reports an impressive AUC of 0.731 using their benchmark data set.

Due to the different data sets and evaluation techniques, there is a need for more consistent benchmarking and evaluation. In 2013 a review of the current conformational B-cell prediction tools was made [5]. This review shows with their evaluation data set that their AUC performance of the prediction tools did not exceed 0.6, except for an meta method combining the different results. The best average AUC of this project is 0.514, which tells us that it cannot predict anything, compared to the results from the review this is worse but only marginally worse. 0.6 is still closer to a coin flip than it is to a correct prediction.

7.7 Further work

The main issue in this area is the insufficient data available. As shown by this project the methods for recalling and identifying epitopes is available, but there is a need for better suited data sets to get satisfying results. Therefore I suggest that future work in this area should focus on assembling a more comprehensive data set, for example it would be better to have 127 epitope examples on the same antigen than having 127 different antigens with one epitope example each (as the one used by this project). It would also be interesting to assemble a confirmed non-epitope data set, using proteins which are accepted in the host. Another thing to note is to only use antigen-antibody complexes which are found in humans, because it is not the main goal to find out which antigens mice will react to.

7.8 Ethics

Being able to predict the conformational B-cell epitopes would help the development of therapeutic proteins. This would lower the costs and the development speed for the corresponding companies. The benefits of an accurate prediction tool for these epitopes lies beyond economical and time gains. Shortening the development time for new qualitative therapeutic drugs, it would benefit the end user by granting them access to new treatments, which could minimize their suffering. In addition the prediction tools could aid with the quality of the therapeutic proteins such that they would induce less of an immune response and decrease the side effects for the subject. The lower development costs would not only benefit the companies developing the drugs but also the customer by a lower price on the product.

Calculating the prediction of possible epitopes and immune response from the human immune system would decrease the need for accrual testing, but not remove it. Today the therapeutic proteins undergo clinical trials on animals, such as mice, before they are tested on humans and finally released to the open market. This animal and human testing would be minimized if the proteins would be screened for most of the initial possible epitopes.

One down side of predicting immune response from proteins is that it can be used for developing biological weapons. Developing a virus, bacteria or molecule which would be accepted by the immune system would be a disaster. This is always a balance to weigh the gains to the danger and the weaponizing argument will come back to haunt new research wherever it takes us.

8

Conclusion

This thesis shows the development of two models for conformational B-cell epitope prediction. It describes the classification method and the training process of different settings in that method. The final results did not show any improvement compared to related work in the area, however it showed that both the epitope models can be used to identify epitopes on the surface of an antigen.

As discussed in chapter 7 the data used in this project is not suited for machine-learning together with the desired goal of this project. The epitope models and classification method does show some potential and should not be excluded from future research.

Bibliography

[1] J. Ponomarenko, HH. Bui, W. Li, N. Fusseder, PE. Bourne, A. Sette, B. Peters. *ElliPro: a new structure-based tool for the prediction of antibody epitopes* BMC Bioinformatics, 9:514 2008.

[2] R. Vita, JA. Overton, JA. Greenbaum, J. Ponomarenko, JD. Clark, JR. Cantrell, DK. Wheeler, JL. Gabbard, D. Hix, A. Sette, B. Peters. *The immune epitope database (IEDB) 3.0*. Nucleic Acids Res. 2014 Available: www.iedb.org

[3] HM. Berman, J. Westbrook, Z. Feng, G. Gilliland, TN. Bhat, H. Weissig, IN. Shindyalov, PE. Bourne *The Protein Data Bank* Nucleic Acids Research, 28: 235-242. 2000 Available: www.rcsb.org

[4] S. Saha, GPS. Raghava *Prediction of Continuous B-Cell Epitopes in an Antigen Using Recurrent Neural Network*, Proteins, volume 65, 40-48. 2006.

[5] B. Yao, D. Zheng, S. Liang, C. Zhang *Conformational B-Cell Epitope Prediction on Antigen Protein Structures: A Review of Current Algorithms and Comparison with Common Binding Site Prediction* PLoS ONE 8(4): e62249, 2013.

[6] M. Baker, HM. Reynolds, B. Lumicisi, CJ. Bryson. *Immunogenicity of protein therapeutics: the key causes, consequences and challenges.* Self Nonself ;1(4):314-322. 2010.

[7] A. Krizhevsky, I. Sutskever, GE. Hinton. *Imagenet classification with deep convolutional neural networks.* Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems, 1106–1114 2012.

[8] PH. Andersen, M. Nielsen, O. Lund. *Prediction of residues in discontinuous B-cell epitopes using protein 3D structures*. Protein Science, 15(11):2558–2567, 2006.

[9] P. Sun, H. Ju, Z. Liu, Q. Ning, J. Zhang, X. Zhao, Y. Huang, Z. Ma, Y. Li. *Bioinformatics Resources and Tools for Conformational B-Cell Epitope Prediction. Computational and mathematical methods in medicine* Computational and Mathematical Methods in Medicine Volume 2013, Article ID 943636, 11 pages, 2013.

[10] JV. Ponomarenko, PE. Bourne *Antibody-protein interactions: benchmark datasets and prediction tools evaluation*. BMC Structural Biology, 7:64. 2007.

[11] JV. Kringelum, C. Lundegaard, O. Lund, M. Nielsen *Reliable B Cell Epitope Predictions: Impacts of Method Development and Improved Benchmarking* PLoS Comput Biol 8(12): e1002829. 2012.

[12] JM. Thornton, MS. Edwards, WR. Taylor, DJ. Barlow *Location of 'continuous' antigenic determinants in the protruding regions of proteins.* EMBO J ;5(2):409-13. 1986.

[13] E. Westhof, D. Altschuh, D. Moras, AC. Bloomer, A. Mondragon, A. Klug , MH. Van Regenmortel *Correlation between segmental mobility and the location of antigenic determinants in proteins.* Nature ;311(5982):123-6. 1984.

[14] ND. Rubinstein,I. Mayrose, T. Pupko *A machine-learning approach for predicting B-cell epitopes.* Mol Immunol. Feb;46(5):840-7. 2009.

[15] MF. Møller *A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning* University of Aarhus, Neural Networks Volume 6, Issue 4, 1993, Pages 525–533. 1991.

[16] JV. Kringelum, M. Nielsen, SB. Padkjær, O. Lund *Structural analysis of B-cell epitopes in antibody:protein complexes* Mol Immunol ;53(1-2):24-34 2012.

[17] J. Tsai, R. Taylor, C. Chothia, M. Gerstein *The Packing Density in Proteins: Standard Radii and Volumes* Journal of Molecular Biology Volume 290, Issue 1, Pages 253–266. 1999.

[18] W. Mehio *A Novel Method to Predict Protein-Protein Binding Surfaces* Master Thesis at Chalmers University of Technology, 2007.

A

Appendix 1

A.1 Metadata, based on [17]

```
atomic_group C3H0 1.61
atomic_group C3H1 1.76
atomic_group C4H1 1.88
atomic_group C4H2 1.88
atomic_group C4H3 1.88
atomic_group N3H0 1.64
atomic_group N3H1 1.64
atomic_group N3H2 1.64
atomic_group N4H3 1.64
atomic_group O1H0 1.42
atomic_group O2H1 1.46
atomic_group S2H0 1.77
atomic_group S2H1 1.77
```

ALA C C3H0	GLN CG C4H2	LEU N N3H1	SER OG O2H1
ALA CA C4H1	GLN N N3H1	LEU O O1H0	THR C C3H0
ALA CB C4H3	GLN NE2 N3H2	LYS C C3H0	THR CA C4H1
ALA N N3H1	GLN O O1H0	LYS CA C4H1	THR CB C4H1
ALA O O1H0	GLN OE1 O1H0	LYS CB C4H2	THR CG2 C4H3
ARG C C3H0	GLU C C3H0	LYS CD C4H2	THR N N3H1
ARG CA C4H1	GLU CA C4H1	LYS CE C4H2	THR O O1H0
ARG CB C4H2	GLU CB C4H2	LYS CG C4H2	THR OG1 O2H1
ARG CD C4H2	GLU CD C3H0	LYS N N3H1	TRP C C3H0
ARG CG C4H2	GLU CG C4H2	LYS NZ N4H3	TRP CA C4H1
ARG CZ C3H0	GLU N N3H1	LYS O O1H0	TRP CB C4H2
ARG N N3H1	GLU O O1H0	MET C C3H0	TRP CD1 C3H1
ARG NE N3H1	GLU OE1 O1H0	MET CA C4H1	TRP CD2 C3H0
ARG NH1 N3H2	GLU OE2 O1H0	MET CB C4H2	TRP CE2 C3H0
ARG NH2 N3H2	GLY C C3H0	MET CE C4H3	TRP CE3 C3H1
ARG O O1H0	GLY CA C4H2	MET CG C4H2	TRP CG C3H0
ASN C C3H0	GLY N N3H1	MET N N3H1	TRP CH2 C3H1
ASN CA C4H1	GLY O O1H0	MET O O1H0	TRP CZ2 C3H1
ASN CB C4H2	HIS C C3H0	MET SD S2H0	TRP CZ3 C3H1
ASN CG C3H0	HIS CA C4H1	PHE C C3H0	TRP N N3H1
ASN N N3H1	HIS CB C4H2	PHE CA C4H1	TRP NE1 N3H1
ASN ND2 N3H2	HIS CD2 C3H1	PHE CB C4H2	TRP O O1H0
ASN O O1H0	HIS CE1 C3H1	PHE CD1 C3H1	TYR C C3H0
ASN OD1 O1H0	HIS CG C3H0	PHE CD2 C3H1	TYR CA C4H1
ASP C C3H0	HIS N N3H1	PHE CE1 C3H1	TYR CB C4H2
ASP CA C4H1	HIS ND1 N3H1	PHE CE2 C3H1	TYR CD1 C3H1
ASP CB C4H2	HIS NE2 N3H1	PHE CG C3H0	TYR CD2 C3H1
ASP CG C3H0	HIS O O1H0	PHE CZ C3H1	TYR CE1 C3H1
ASP N N3H1	ILE C C3H0	PHE N N3H1	TYR CE2 C3H1
ASP O O1H0	ILE CA C4H1	PHE O O1H0	TYR CG C3H0
ASP OD1 O1H0	ILE CB C4H1	PRO C C3H0	TYR CZ C3H0
ASP OD2 O1H0	ILE CD1 C4H3	PRO CA C4H1	TYR N N3H1
CYS C C3H0	ILE CG1 C4H2	PRO CB C4H2	TYR O O1H0
CYS CA C4H1	ILE CG2 C4H3	PRO CD C4H2	TYR OH O2H1
CYS CB C4H2	ILE N N3H1	PRO CG C4H2	VAL C C3H0
CYS N N3H1	ILE O O1H0	PRO N N3H0	VAL CA C4H1
CYS O O1H0	LEU C C3H0	PRO O O1H0	VAL CB C4H1
CYS SG S2H1	LEU CA C4H1	SER C C3H0	VAL CG1 C4H3
GLN C C3H0	LEU CB C4H2	SER CA C4H1	VAL CG2 C4H3
GLN CA C4H1	LEU CD1 C4H3	SER CB C4H2	VAL N N3H1
GLN CB C4H2	LEU CD2 C4H3	SER N N3H1	VAL O O1H0
GLN CD C3H0	LEU CG C4H1	SER O O1H0	

A.2 N closest - Data set

The following data sets where generated and analyzed with the following number of atoms in each patch.

Small size	110
Mid size	200
Big size	350

Table A.1: The number of atoms in each patch for respectively data set

A.2.1 Small data set

PDB id_chain	# epitope patches	epitope width over graph
1adq.pdb_A	2	12
1afv.pdb_A	15	13
1bql.pdb_Y	9	14
1cz8.pdb_W	12	17
1dzb.pdb_X	3	13
1e6j.pdb_P	13	13
1egj.pdb_A	14	14
1fdl.pdb_Y	15	11
1fns.pdb_A	13	11
1fsk.pdb_A	9	22
1g9m.pdb_G	6	12
1h0d.pdb_C	13	10
1hys.pdb_B	13	9
1iqd.pdb_C	1	15
1jhl.pdb_A	7	11
1jrh.pdb_I	11	14
1k4c.pdb_C	10	11
1kb5.pdb_A	12	12
1ken.pdb_A	5	10
1lk3.pdb_A	6	12
1n0x.pdb_P	17	13
1n5y.pdb_B	14	12
1nfd.pdb_B	15	13
1nfd.pdb_D	11	12
1nl0.pdb_G	14	12
1oak.pdb_A	13	10
1ors.pdb_C	19	13
1ots.pdb_A	16	12
1p2c.pdb_C	10	16
1pkq.pdb_E	5	19

1qfw.pdb_A	20	11
1qgc.pdb_5	28	7
1r0a.pdb_B	12	15
1r3j.pdb_C	11	11
1rjl.pdb_C	14	12
1tpx.pdb_A	13	17
1tqb.pdb_A	14	15
1tzi.pdb_V	3	11
1wej.pdb_F	9	12
1xiw.pdb_F	10	13
1ynt.pdb_F	3	11
1z3g.pdb_A	18	12
2a0l.pdb_A	22	8
2adf.pdb_A	6	10
2fd6.pdb_U	9	11
2fjg.pdb_W	9	11
2fjh.pdb_W	18	14
2hfg.pdb_R	21	14
2i9l.pdb_I	13	13
2j4w.pdb_D	20	13
2j6e.pdb_B	16	9
2jel.pdb_P	6	14
2jix.pdb_C	10	11
2osl.pdb_Q	18	8
2qqk.pdb_A	10	13
2qqn.pdb_A	13	9
2qr0.pdb_U	17	10
2r4r.pdb_A	19	14
2vir.pdb_C	3	11
2vit.pdb_C	5	11
3c09.pdb_A	12	12
3csy.pdb_J	10	14
3d85.pdb_C	9	18

Table A.2: The small data set.

A.2.2 Mid data set

PDB id_chain	# epitope patches	epitope width over graph
1a14_N	18	13
1ahw_C	23	14
1ar1_B	21	12
1bql_Y	26	14
1bvk_C	32	11

1dzb_X	27	13
1eo8_A	20	12
1ezv_E	25	14
1fbi_X	16	21
1fe8_A	24	15
1fe8_B	22	14
1fj1_F	25	19
1i9r_A	15	14
1iqd_C	24	15
1jps_T	19	13
1ken_A	28	10
1mhp_A	29	14
1mhp_B	24	15
1n8z_C	38	13
1nca_N	14	13
1ndg_C	24	15
1nmb_N	17	17
1nsn_S	13	23
1oaz_A	17	16
1ob1_C	20	15
1pkq_E	28	19
1qfu_A	16	22
1sy6_A	23	18
1txv_A	32	15
1v7m_V	22	15
1xiw_A	17	19
1yjd_C	24	14
1ynt_F	28	11
1za3_R	34	13
1ztx_E	30	18
2aeq_A	18	15
2arj_R	15	13
2bdn_A	18	15
2cmr_A	17	14
2dd8_S	20	16
2dqd_Y	24	15
2dqf_C	28	11
2dtg_E	18	14
2fd6_U	22	11
2fjh_W	36	14
2nr6_A	25	12
2nyy_A	17	17
2oz4_A	24	19
2qad_A	11	15

2r0k_A	19	14
2r0l_A	14	14
2r29_A	26	13
2r4r_A	27	14
2r56_A	24	14
2vc2_A	26	15
2vh5_R	31	13
2vit_C	30	11
2zch_P	20	15
3b2u_B	13	14
3bn9_B	8	17
3d85_C	30	18

Table A.3: The mid data set

A.2.3 Big data set

PDB id_chain	# epitope patches	epitope width over graph
1i9r.pdb_A	55	14
1ndg.pdb_C	78	15
1nsn.pdb_S	51	23
1osp.pdb_O	48	18
1rvf.pdb_1	61	23
1s78.pdb_A	23	21
1s78.pdb_B	2	21
1txv.pdb_A	62	15
1w72.pdb_A	43	16
1yy9.pdb_A	58	13
1za3.pdb_S	51	21
2aep.pdb_A	45	20
2b4c.pdb_G	33	17
2h9g.pdb_R	62	19
2r0l.pdb_A	46	14
2vc2.pdb_A	55	15
3b2u.pdb_B	51	14

Table A.4: The big data set