Melanoma Detection and Classification of Birthmarks Using Neural Networks and Genetic Programing

Henrik Mygind

Kongens Lyngby 2011
IMM-M.Sc.-2011-53

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

Summary

This is the master project of Henrik Mygind developed in collaboration with Rigshospitalet, Pallas Informatik and the DTU Informatics at The Technical University of Denmark. In this thesis a prototype software is developed which is able to classify digital color images of birthmarks determining if a birthmark is melanoma or harmless.

A feature extraction algorithm is developed in Matlab and two classification algorithms are developed in C#. One classification algorithm is build on genetic programing while the other use neural networks trained by backpropagation and a genetic algorithm.

The software is tested using a dataset received from Rigshospitalet containg 164 images of melanoma and 172 images of harmless birthmarks. The tests have shown that the feature extraction is able to find the birthmarks in 64% of the images while the classification algorithms works as intended when tested on a general dataset.

Using the feature extraction developed for this project and the best classification algorithm, a classification rate of 78% is received. The classification rate is found when the false positive rate and the false negative rate are equal. When using images on which the feature extraction works perfectly a slightly higher classification rate of 82% is reached.

Even at this early stage in the development the software looks promising and when a better feature extraction algorithm is developed the software can be a huge help in the early detection of melanoma around the world.

Resumé

Dette er Henrik Myginds kandidat speciale udviklet i samarbejde med Rigshospitalet, Pallas Informatik og DTU Informatik på Danmarks Tekniske Universitet. I dette projekt er der blevet udviklet en prototype på et stykke software der er i stand til at klassificere digitale farvebilleder af modermærke og inddele dem i kategorierne ufarlig og modermærke med kræft.

I forløbet er der blevet udviklet et billedanalyseværktøj i Matlab som kan finde forskellige features i modermærkerne. Desuden er udviklet to klassifikationsalgoritmer i C# baseret på genetisk programmering og neurale netværk trænet ved hjælp af backpropagation og en genetisk algoritme.

Softwaren er testet ved hjælp af et fra Rigshospitalet modtaget datasæt indeholdende 164 billeder af modermærke med modermærkekræft og 172 billeder af ufarlige modermærker. Testene har vist at billedanalyseværktøjet er i stand til at finde modermærkerne korrekt i 64% af billederne mens klassifikationsalgoritmerne fungerer fejlfrit når de bliver testet på et standarddatasæt.

Ved at bruge billedanalyseværktøjet udviklet i dette projekt og ved at bruge den bedste klassifikationsalgoritme opnås en klassifikationsrate på 78%. Klassifikationsraten er fundet hvor andelen af falsk positive og falsk negative er lige stor. Når man kun bruger de billeder hvor billedanalyseværktøjet virker fejlfrit opnås en lidt højere klassifikationsrate på 82%.

Selv på dette tidlige stadie i udviklingen har softwaren vist kæmpe potentiale. Når der bliver udviklet et bedre billedanalyseværktøj har softwaren mulighed for at blive en stor hjælp i arbejdet med at opdage modermærkekræft tidligt i hele verden.

Acknowledgements

I would like to thank my supervisor at DTU, Carsten Witt, for his inputs and guidance through a half year of intenssive and interesting work.

I would also like to thank Rune Saaby, my supervisor at Pallas Informatik, for his guidance and his ideas on solving the problem.

Linda Jakobsen, Doctor at the Department of Plastic Surgery and Burns Treatment at Rigshospitalet, should have a special thanks for presenting the idea to me and for giving me access to the image database.

I would like to thank the people at Visionday 2011 for the many useful inputs on the half-done work presented here and Svend Vitting Andersen at Pallas Informatik should have a thanks for introducing me to this event in the first place.

To all my colleagues at Pallas Informatik I owe a thanks. Without your support it would have been a much harder job to do. Especially my colleagues at the student isle I owe a thanks.

Lastly I would like to thank my girlfriend, Marie Top Hartmann, for being supporting and understanding while I was working full time on this project. Your support has meant a lot to me.

Contents

Chapter 1

Introduction

This is the master thesis of Henrik Mygind, DTU Informatics 2011. In this project a software is developed which is able to classify birthmarks in groups of harmfull and harmless birthmarks using only information found in an ordinary color image.

1.1 The process

The software developed in this thesis is meant to be a prototype and therefore not a full bug-free application with a nice user interface etc. However to find out what the software should be capable of the software is put into a process as it would look if it was an application which was to be used in the industry.

The software is thought to be part of the normal diagnosis system where it should supprt the house doctor in his decision. It is not meant to be used as a stand-alone software. The house doctor should receive feedback from the system describing what has been analysed, what features have been found in the image, what classification the software believes the birthmark should have and a percentage telling how certain it is that this answer is correct.

The process, which can be found in figure 1.1, is divided into two parts: a training process and an evaluation process.

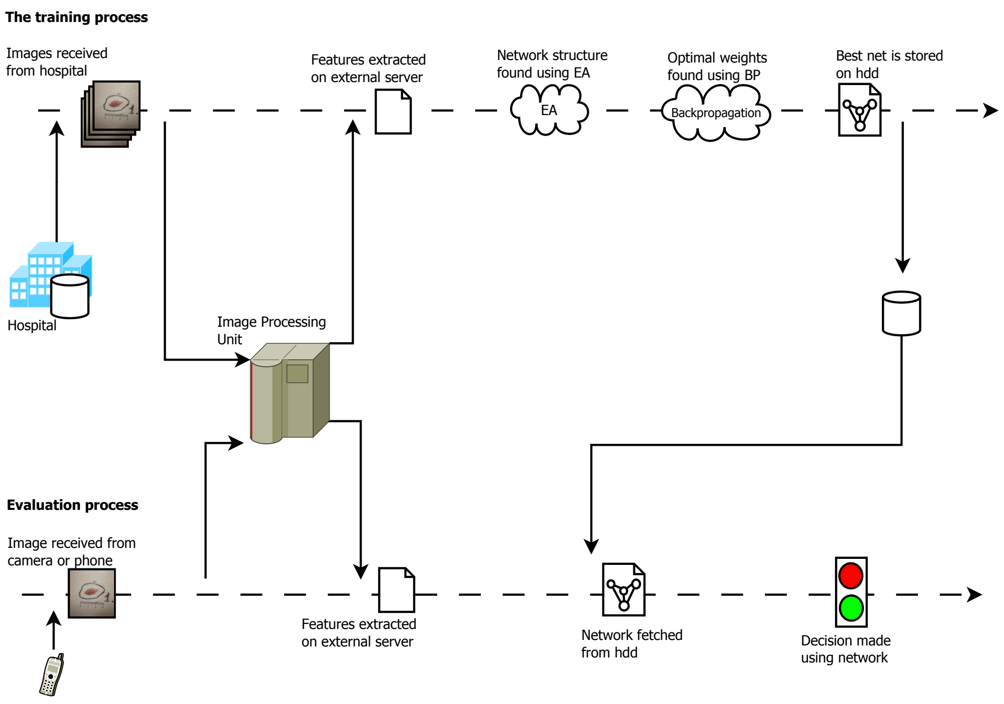

The training process
Images received from hospital
Features extracted on external server
Network structure found using EA
EA
Optimal weights found using BP
Backpropagation
Best net is stored on hdd
Hospital
Image Processing Unit
Evaluation process
Image received from camera or phone
Features extracted on external server
Network fetched from hdd
Decision made using network

Figure 1.1: The training process and the evaluation process

1.1.1 Training process

The purpose of the training process is to train the classification algorithm such that the neural network or genetic program used for classification, has the best classification rate and has a minimum of false negatives.

In this project a positive sample is when the birthmark is classified as melanoma.

A false negative is therefore when the classification algorithm classifies are birthmark as harmless even though the correct classification would be melanoma.

The training is done using images from a database containing images of birtmarks, for example from a dermatological clinic, which have already been classified. Each image is sent to a feature extraction server (the Image Processing Unit) which returns a list of inputs and their real output. These inputs are given to the classification software which trains a neural network or a genetic program. The best network/program is stored to the harddrive and is used to classify new images.

1.1.2 The evaluation process

In the evaluation process the actual use of the software is found. It is in the evaluation process that an unclassified birthmark is classified.

The process starts with a user capturing an image of a birthmark. This could be done using the camera found in the mobile phone or by a camera handled by the doctor. The image is sent to the Image Processing Unit which extracts the features of the given birthmark. The features are then given to classification software which uses these features as inputs for the best network/program stored on the harddrive. The network/program will then come up with a classification of the birthmark; it will give a likelyhood of the pressence of melanoma. Based on the output of the classification software appropriate action will be taken. If a melanoma is present the software could automatically warn the doctor or the user could be noticed.

The software developed in this thesis includes a basic Image Processing Unit and the classification software. The interfaces between the individual parts, the web services used to upload images etc. has not been implemented and is therefore not functional.

1.2 Background knowledge

Melanoma is one of the rarest types of skin cancer, but the deadliest. 4% of skin cancer diagnoses are melanoma, but melanoma accounts for 77% of skin cancer related deaths.[Cal] Early detection is important, when melanoma is detected in its early state the 5-year survival rate is $> 90\%$ while at the later stages it drops to less than 10%. [al.01]

Melanoma evolves in pigment producing cells such as birthmarks. It will quickly start to change the outside appearance of the birthmark and is therefore easier to spot than other types of cancer. To spot its pressence a checklist developed in 1985 and revised in 2004, is often be used.[al.04] This check list, called the *ABCDE* criteria, is used as inspiration for the feature extraction algorithm in this project.

1.2.1 The ABCDE criteria

The *ABCDE criteria* is a checklist which can be used to categorize birthmarks into a group of harmless and a group of harmfull marks. A birthmark having one or more of these features will by the doctor be considered possible harmful. One disadvantage of the *ABCDE criteria* is that some types of skin lesions (for example *seborrheic keratoses*) might show some or all of the same features as described in the list. But as a few false positives is better than a lot of false negatives it is still widely used in the dermatoscopic world. The list is as follows:

Assymetry. One side of the birthmark is of a different shape than the other side.

Border irregularity. The edge is notched, blurred or ragged.

Color variegation. The birhtmark contains multiple colors: red, brown, white, black and blue.

Diameter. The diameter is greater than 6mm. Most melanoma are greater than 6mm in size, but a great part is still smaller, so this measure cannot be used on its own.

Evolution or enlargement. The birthmark changes color or shape.

Another widely used approach is the *ugly duckling* sign. If a birthmark varies greatly from any other birthmark on the patient this could be a sign of melanoma.

In this project the *ABCDE criteria* are used for the feature extraction process. This is due to the fact that the method need only one image of the birthmark and because no information about the patient and his/her other birthmarks is needed. The feature extraction will focus on the A, B, C and to some extend the D part of the criteria as only one image is used and it therefore is not possible to see any evolution or enlargement in the birthmark. The diameter will only partly be used because the size of the birthmark is only measured in pixels and this size is relative to the cameras position and resolution.

1.3 Work done in the report

In this masters project a framework for genetic algorithms has been implemented. Along with the framework two implementations have been made, a neural network and a genetic program, which both use the framework to evolve. These parts have been implemented in C# on the Windows platform.

Further more a feature extraction has been created for Matlab with the Image Processing Toolbox. This software can run on both Windows and Unix and has in this project been running on the Unix platform.

All parts of the software has been tested.

1.4 Structure

The thesis starts out with this chapter in which an introduction to the project is found. The introduction contains describtions of the problem, the background knowledge and the method used in this thesis.

In chapter 2 it is described how the feature extraction works and the theory behind is explained. Chapter 3 contains the description of the neural networks and the theory on which it is based, and in chapter 4 the theory of the genetic programming is found.

The theory used for the genetic algorithm used by both the neural network and the genetic programing can be found in chapter 5. In chapter 6 a description of the design of the genetic algorithm and some special implementations used in this project can be found.

The description of the tests performed and the test results can be found in chapter 7. This chapter is followed by the conclusion, chapter 8, in which the results of the work is discussed and a discussion of further work can be found.

Chapter 2

Feature Extraction

To classify the birthmarks some input parameters has to be found, this is done using image analysis or feature extraction as it will be called in this project. The feature extraction is done using the *ABCDE criteria* described in section 1.2.1. Limited to the possibilities of a computerised extraction process it is the hypothesis that the following features (also used by [HM98]) extracted from the image will be sufficient for the classification process. The actual implementation is described more thoroughly in the following sections.

Assymetry (A) which is found by mirroring the birthmark along one of the principal axes, and measuring how much of one side of the birthmark is "outside" the area of the other side of the birthmark.

EdgeAbruptness (B) , the steepness of the edge gradient. The edge abruptness is measure on how sharp the edge between the birthmark and the skin is.

Size (B+D) of birthmark as well as the length of the edge compared to the size of the birthmark.

Number of colors (C) found by putting the pixels of the image into one of six color classes and counting how many of the classes are present in the image.

Size of black, blue and white (C) areas in the birthmark.

The parentheses indicate what part of the *ABCDE criteria* each of the features fit into. In figure 2.1 the image extractions being performed on an image of a birtmark can be found.

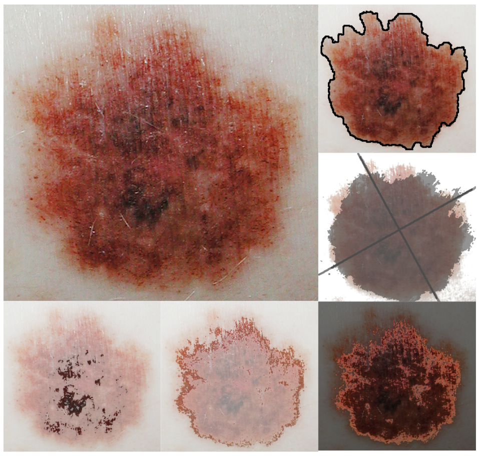


Figure 2.1: Image of a birthmark and the types of feature extractions that has to be done on it. The big image is the original image. On the top right the border of the birthmark is found, below the asymmetry is measured and in the bottom row 3 different color classes are highlighted.

As the focus of this thesis is not mainly on image analysis only the high level ideas of the transformations are described. For a more thorough understanding of the individual feature extraction steps please see the source reffered to in each section.

2.1 The feature extraction process

The feature extraction process starts with an image being received by the algorithm. A lot of computation and manipulation is done to the image and finally 9 features are extracted.

The input of the feature extraction is an ordinary color image. The quality of the image should be as high as possible, but the actual quality needed will be tested in section 7.3.2. The sharper the image and the less glare from a camera flash, the better.

The feature extraction is done in a number of steps, each described in the following sections. The process can be described as follows:

- First the image is read from the disk.
- A copy of the image is made and this copy is filtered using a median filter.
- The filtered image is transformed using the Karhunen-Loève transformation.
- A mask is generated by thresholding the first principal component.
- The asymmetry is measured on the mask
- The edgeabruptness is extracted using the mask and the original image
- The size of the edge and the size of the birthmark are measured using the mask.
- The number of color classes and the ratio of the critical colors are measured in the area of the original image lying inside the mask.
- The results are added to the output file.

2.1.1 MedianFilter

To minimize the bad effects of noise such as jpeg artifacts, highligted areas (due to a strong light source) and hairs a median filter is applied to the image before the location of the birthmark is determined.

A 7×7-matrix is used as filter and it is applied to each pixel in each image layer (R,G and B). The filtering works by taking all the pixel values inside the matrix and finding the median value. This value is then used as the new pixel value. In figure 2.2 an example of the effect of the median filter is shown.

2.1.2 Karhunen-Loève Transformation

Some birthmarks are very similar to the skin while others stand out more clearly. Using the grayscaled image the colors can therefore easily be mixed with each other and it can be hard to locate the edge of the bithmark. To improve the edge detection the Karhunen-Lève transformation (also known as the principle components analysis [Jol02]) can be used.[HM98]

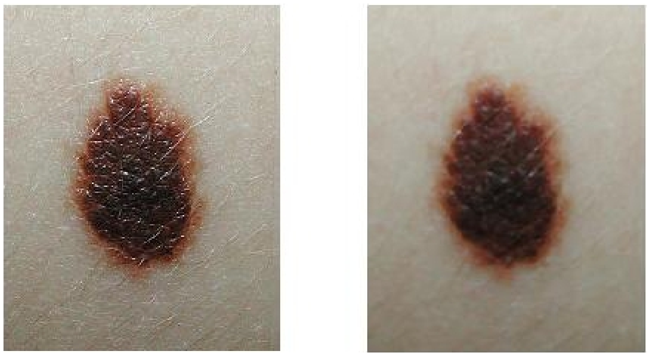


Figure 2.2: Median filter. The effect of using the filter on an image of a birthmark.

The KL transformation is a linear transformation widely used in statistics to convert possibly correlated values into sets of uncorrelated varibles where most variance is found in the first set, second most in the second set etc. The transformation is done by finding the eigenvectors of the sample covariance matrix which describes the characterisitcs of the covariance in the different dimensions. (In this thesis they describe the covariance in the three color bands.) Combined with the deviation each value has from the mean, these eigenvectors are used to transform the distribution into a new coordinate space with possibly lower dimensions.[HM98]

In this thesis the KL is applied to the 3D color space and the effect of the KL will be that regions which varies a lot from the rest of the image will be placed in the first principal component due to the high variance while other regions will be placed in the second and third region.

The effect of the principle components analysis can be seen in figure 2.3.

2.1.3 Masking

Once the median filtering and the Karhunen-Loève transformation has been performed it is time to do the actual thresholding in order to create a mask which describes the area of the birthmark.

The thresholding is done using the iterative selection thresholding method [HM98] in which the optimal thresholds are found by moving the thresholds until the luminance region means are stable.

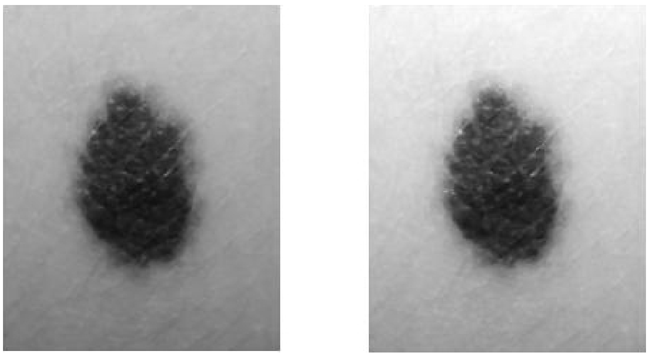


Figure 2.3: Karhunen-Loève transformation. On the left normal grayscale image is shown and found on the right is the image found using the principle components analysis.

In figure 2.4 an example of the thresholding is shown.

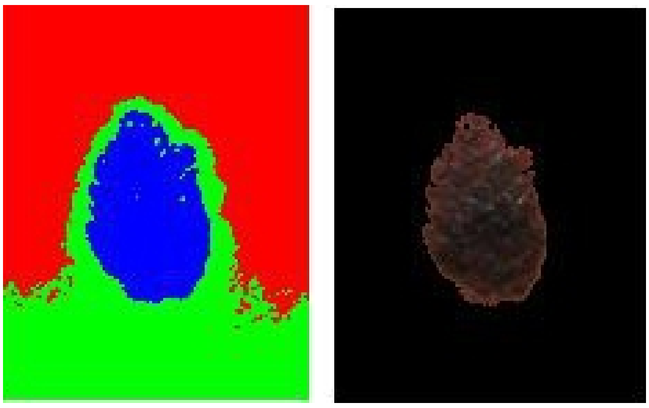


Figure 2.4: Thresholding. The left image shows the three areas found using the thresholding. The right shows the masked image. Due to the fact that the green and the blue areas take up more than 40% of the space in the image only the blue area is chosen as mask.

2.1.4 Assymetry

Asymmetry in a birthmark might indicate that the mark is growing. A healthy birthmark does not normally grow why a growing birthmark is a strong indicator of the pressence of melanoma.

The asymmemetry is measured by finding the two principal axes of the birthmark. This is done by calculating 2-D moments: the inertia and the center of mass. The major axis is the axis going through the center of mass giving the smallest amount of inertia and minor axis is the one giving the largest amount of inertia.[HM98]

The image is then mirrored on axis at a time, measuring how much of the image is not overlapping compared to the amount which is overlapping. This way two features are extracted, the asymmetry in the major axis and the asymmetry in the minor axis.

In figure 2.5 an example of the asymmetry measurement is shown.

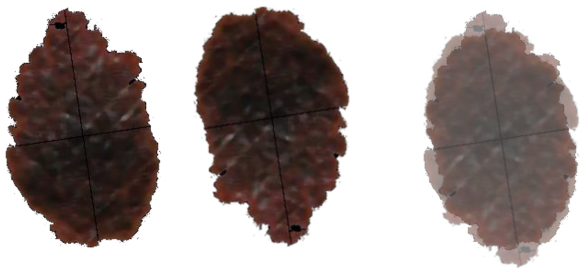


Figure 2.5: Asymmetry measurements. Mirroring is done in the minor axis.

2.1.5 EdgeAbruptness

If a birthmark has a very sharp edge from skin to birtmark it can be a sign of melanoma, where as a more gradual transision can be a sign of an unharmful birthmark.[HM98]

In order to find the edge abruptness the image is first grayscaled and then filtered using the sobel operators which are applied to every pixel in the image. The sobel operators are as follows:

$$H_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, H_2 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

These operators will when applied to the individual pixels of the image describe the magnitude (and the direction) of the transition between the neighbour pixels.

And as only the magnitude is used in this thesis they can be said to describe how smooth the transition is from the pixels lying above the current pixel to the pixels found below.

Filtering the image results in two images, g_1 and g_2 which are combined in to an image showing the abruptness of the entire image:

$$z = \sqrt{g_1^2 + g_2^2}$$

This image is shown in figure 2.6 on the left. To find only the *edge* abruptness the filtered image is masked to contain only the edge of the birthmark. The parameter used by the classification algorithm is then the mean abruptness of the pixels found in the edge.

In figure 2.6 an example of the result of running the edge abruptness algorithm is shown. Due to the fact that the image is in jpeg-format the edge abruptness algorithm might be malfunctioning and might therefore be useless. The influence of it will be tested later in section 7.7.

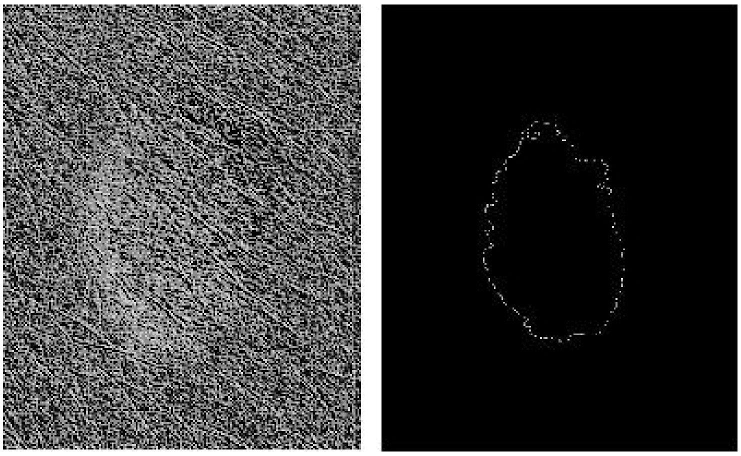


Figure 2.6: Edge abruptness. On the left the abruptness of the entire images is shown, on the right only the abruptness of the edge is shown. The total abruptness in the edge is used as a feature in the classification algorithm.

2.1.6 Edge2area and area size

The shape of the edge can be used to determine whether a birthmark is harmful. For example could a rough edge a sign of melanoma and therefore a measure on how big the edge is compared to the size of the birhtmark, is used as a feature. This measure combined with the actual pixel size of the birthmark can give a hint to what state the birthmark is in.

The edge-to-area ratio is measured as

$$ratio = \frac{|mask_{edge}|}{|mask_{area}|}$$

where $mask_{edge}$ is the pixels being inside the area of the bithmark but having at least one neighbour outside the area, and $mask_{area}$ is all the pixels inside the area of the birthmark. The masks are shown in figure 2.7.

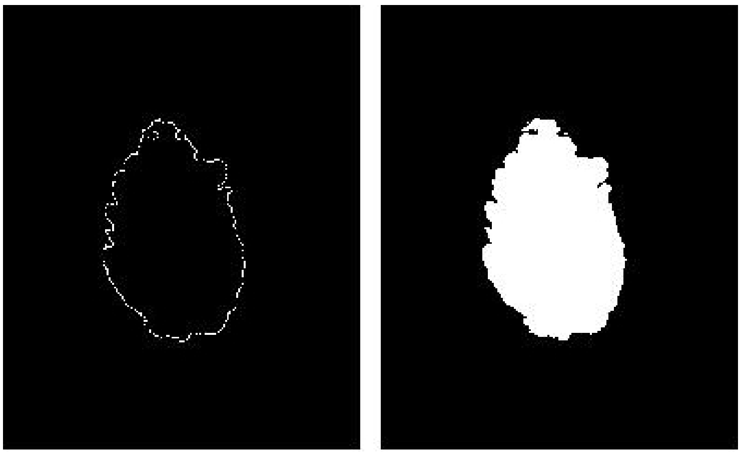


Figure 2.7: Edge to area ratio. On the right the edge is shown and on the left the total area of the birthmark is shown. The ratio between these areas is used to indicate how rough the edge is.

2.1.7 Colors

The final features which are extracted from the images are the color related features. The colors of the birthmark can be very informative when it comes to birthmarks. To limit the number of features the pixels in the image is assigned one out of six color classes. The color classes have been chosen by hand from an image in the test dataset and have the following RGB-values:

Color class	R	G	B
Dark brown	29	18	14
Light brown	111	39	24
Red	138	64	65
Black	54	31	39
White	68	67	67
Blue	69	66	73

Once every pixel is assigned a color class, which is done by calculating the distance to each color class and choosing the color class with smallest distance, four features are extracted: the number of colors and the share of pixels which are black, blue and white respectively.

The distance is measured as follows:

$$dist_{p,cc} = |p_r - cc_r| + |p_g - cc_g| + |p_b - cc_b|$$

where p is the pixel value (red,green,blue) and cc is the rgb-values of the color class.

Black, blue and white areas can be a sign of melanoma and the ratio of these are therefore extracted as well as the number of colors. If many colors are present in the image there is a bigger chance of melanoma.

In figure 2.8 a bithmark with its pixels assigned a color class is shown.

2.1.8 The final outcome

The output of the extraction process is a file containing a line for each image processed. Each line contains the name of the image and the extracted features shown as 9 decimal numbers in the following order:

- the asymmetry of the birthmark measured in the major axis
- the asymmetry of the birthmark measured in the minor axis
- the mean edge abruptness of the birthmark
- the area of the border of the birthmark (1px border) compared to the total birthmark area
- the size of the area
- the number of color classes present in the birthmark

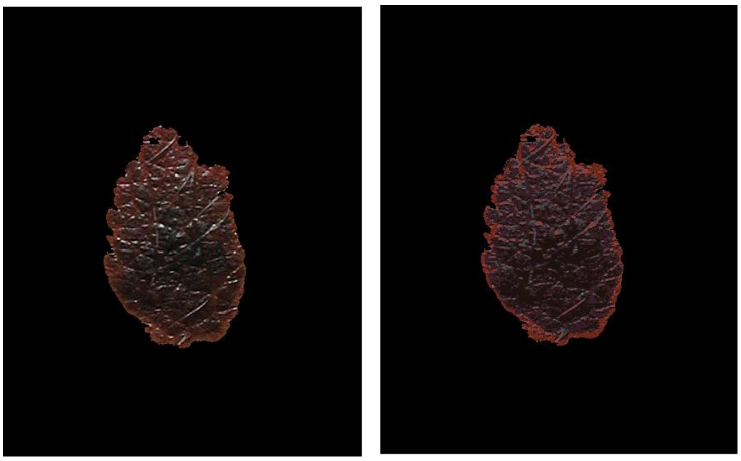


Figure 2.8: Color classification. The individual pixels of the birthmark are assigned to a certain color class. On the left the original image is shown, on the right every pixel is converted into the nearest color class.

- the ratio of the black color class
- the ratio of the blue color class
- the ratio of the white color class

The input for the classification algorithm should be decimals in the range $[0.0; 1.0]$ and some of the numbers therefore have to be converted into this range. The following numbers are transformed using the transformation $new = \frac{1}{old}$.

- mean edge abruptness
- size of the area
- the number of color classes

In section 7.7.1 on page 66 the distribution of the values of the individual parameters are found.

Chapter 3

Neural Network

The brain consists of nerve cells also known as *neurons*. Each neuron collects, procceses and distributes electrical signals in the brain. The neurons are connected in networks via *synnapses*. Depending on the processing of the collected inputs a neuron might distribute an electrical output through the synnapses. The brains capability of processing information is thought to primarily come from these networks.[RN03]

Due to the fact that the human brain consist of such networks the early AI work tried to create such artificial neural networks (ANN). It is however not possible to fully replicate the networks in the brain as the complexity of the brain is huge: the brain consists of about 100 billion ($100 \cdot 10^9$) neurons and each neuron communicates with 1000-10000 neurons).[Col01]

3.1 The simplified mathematical model

It is possible to make make a simplified mathematical model of such networks. A widely used mathematical model of a neuron is shown in 3.1.

A neural network consists of a number of input and output neurons (the number is decided by the data which the network is to evaluate) and of a number of hidden neurons put between the input and output nodes. Different types of

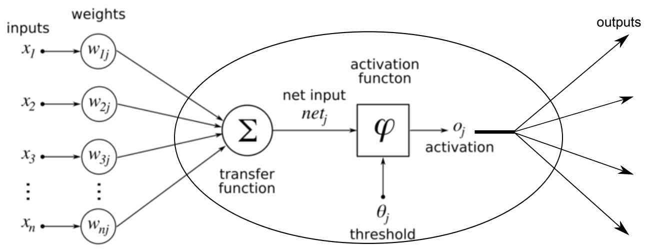


Figure 3.1: A mathematical model for a neuron.[Chr05]

networks can be made: There are feed forward nets where the net does not contain any cycles and recurrent networks containing cycles. Recurrent networks is cabable of having a sort of short term memory as the outcome is depending on the input state of the network; and this input state can be dependent of the outcome of the last interation. This also increases the complexity of model why the feed forward network is used in this project.

The neuron (shown in figure 3.1) computes its output using the inputs as well as an activation function. First all the input values are linearly combined, for example through sumation or averaging. Then the activation function is applied to the combined inputs forming the output. The algorithm is also shown in figure 3.2.

The activation function can be a simple threshold function:

$$threshold(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

or a more gradual transistion such as the sigmoid function:

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

where x is the weighted sum of inputs.

```
inputSum = 0
for(i = 0; i < inputs.count; i++)
  inputSum += weight[i] * inputs[i]
output = actFunc(inputSum)
```

Figure 3.2: Pseudocode showing what processing a neuron does.

To give the inputs different influence on the result a weight is added to each input. The weights can be any decimal number but have most power in the interval $[-5.0 : 5.0]$ as the sigmoid values of these boundaries are very close to 0 and 1 respectively. Due to this an initial network should not have weights outside this interval.

The sigmoid function and other threshold functions often trigger around $x = 0$ where x is the weighted sum of inputs. To move this trigger midpoint a constant (a threshold or a bias) can be added to the equation.[Bis97]

$$output = sigmoid(\sum_i w_i \cdot input_i + w_b) \tag{3.1}$$

In this way the threshold of the function will shift as shown in figure 3.3. In the implementation of the neural network this functionality is added by adding a bias node to all hidden and output nodes in the network. The bias nodes all have the output value -1 and their strength is then based on the weight of the synapsis going from the bias node to the neuron.

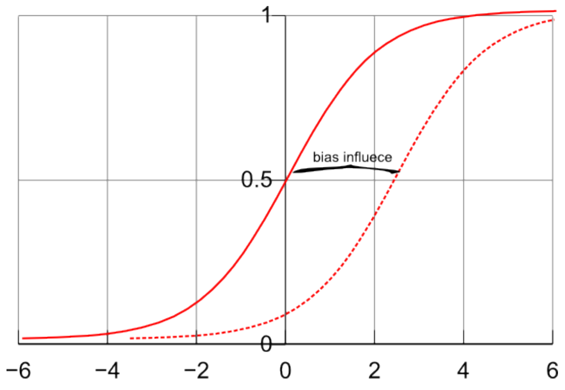


Figure 3.3: The logistic sigmoid function. When the bias is added to the equation it is possible to move the entire function along the x-axis.

3.1.1 Feed forward networks

The simplest neural network is a variant of the feed forward network called a perceptron. The perceptron is a neural network with no hidden nodes. All input nodes are connected to all output nodes and only weights can differ. This network is only able to replicate a linear function, why it usually is not good enough.

An easy improvement of the perceptron is the multilayer network where the neural network contains a number of layers containing hidden nodes. Each layer is fully connected to the previous and the following layer. This type of network is able to find any type of function as long as the number of hidden nodes - and layers - is high enough.

Finally an almost unstructured neural network, called unsorted network in the following, is used. In this network the connections between the nodes are randomly chosen. The unsorted network consists of a number of hidden nodes. These nodes along with the input and output nodes are enumerated from 1 to n where n is the total number of nodes. The input nodes have the lowest numbers and the outputnodes have the higest.

Each node is connected to every following node with probability p. So the probability that an outgoing edge is found in node 1 is $(n-1) \cdot p$ and the probability that there is an outgoing edge of node n is 0.

Using this network it is ensured that all nodes have the same expected number of synnapses connected to them. Further more it ensures that no cycles can exist in the network as all synnapsis go out of node with a number i and into a node with a number $j > i$.

There is no known solution for finding the best structure of a neural network.[RN03] Therefore experiments will have to lead to what kind of structure is better for a given problem. This experimenting can be automated by the use of genetic algorithms, see section 3.1.3.

The feedforward network calculates the outputs by feeding the inputs forward through the network. This is done by updating each of the neurons in the network by taking the weighted sum of the input neurons. A pseudo code for calculating the outputs is shown in figure 3.4.

3.1.2 Backpropagation

To train the weights of the neural network backpropagation is used. Backpropagation is an algorithm which minimizes the overall error gradient by correcting the weights in the network using the error found on the output nodes. [RN03]

When training, the network calculates an output on each output neuron and an error between the calculated output and the real output. This is error is propagated back through the network updating the weights along the way. By doing this a new smaller error will occur and this error wil be backpropagated until the error is as small as possible.

```
foreach $e$ in $trainingdata$ do
  foreach node $i \in$ input layer do
    $output[i] = e.input[i]$
    $queue.enqueue(i)$
  while $|queue| > 0$ do
    $n = queue.dequeue()$
    if $\exists_{j \in n.inputs} output[j] == null$ then
      $queue.enqueue(n)$
    else
      $output[n] = \sum_{j \in n.inputs} g(weight_{j,n} \cdot output[j])$
      foreach $j \in n.outputs$ then
        $queue.enqueue(j)$
```

Figure 3.4: The feedforward neural network algorithm used to calculate the ouputs of each node in the network.

In figure 3.5 pseudo code showing the backpropagation algorithm can be found.

3.1.3 Training the structure of network

To find the structure of the network a genetic algoirthm is used. An automated approach ensures that many structure combinations are tested which most likely would not be tested if started manually.

By using a genetic algorithm it is possible to maintain a sparse structure where the backpropagation algorithm will not have to crunch for a long time, which is the case for huge dense networks.

Furthermore it is possible to use the same framework for both the genetic algorithm and the genetic programming. The description of the general genetic algotithm is found in section 5 along with examples on how the different parts of it works with the neural networks and the genetic programs.

```
foreach $e$ in $trainingdata$ do
  foreach node $o \in$ output layer do
    $delta[o] = e.outputs[o] - output[o]$
    $queue.enqueue(o)$
  while $|queue| > 0$ do
    $n = queue.dequeue()$
    if $\exists_{j \in n.outputs} delta[j] == null$ then
      $queue.enqueue(n)$
    else
      $delta[n] = \sum_{j \in n.outgoing} weight_{n,j} \cdot delta[j]$
      foreach $j \in n.ingoing$ do
        $queue.enqueue(j)$
  foreach node $n$ in $network \backslash input\ layer$ do
    foreach node $j$ in $n.ingoing$ do
      $weight_{j,n} = weightj, n + \alpha \cdot g'(output(j)) \cdot delta[n]$
```

Figure 3.5: The backpropagation algorithm used for updating the weights. A queue implementation is used to support other types of networks than the standard multi layered network.

CHAPTER 4

Genetic Programing

Genetic programming is an extension of genetic algorithms. It works exactly as a genetic algorithm with the exception that the individuals is actual programs instead of a list of numbers or input parameters for a function. An individual in a genetic program can be visualized as a tree in a LISP-like structure.[Koz93] In figure 4.1 an individual is shown as a tree. The function it evaluates is

$$I(1) + (3 \cdot I(2))$$

where $I(1)$ and $I(2)$ are input values.

The individual programs consist of *functions* and *terminals*. A function can be anything from small code bits adding or subtracting two numbers, moving an agent etc. or bigger code chunks calculating the distance between two agents, finding the shortest path between some points or the like. The functions take a number of inputs being either other functions or terminals. A terminal is a leaf in the program tree. Typically it is a value, for example the boolean *true* or a random number between 13 and 42. But it can also be a command to an agent (`move forward`, `move to nearest box`) or something else, depending on the situation in which the programs is to be used.

In this classification software simple functions and terminals are used. The functions used are *addition*, *subtraction* and *multiplication* all working with 2

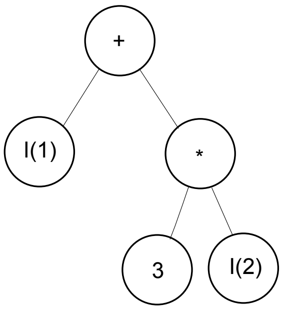


Figure 4.1: An example of a genetic program consisting of the the two functions "+" and "*" and the three terminals "input 1", "input 2" and the constant 3.

input functions/terminals. The terminals used in this software are either a random decimal number in the interval $[-10; 10]$ or an *input* which takes the input from the problem.

A gentic program individual is usually visualized as a tree, as shown in figure 4.1. As with trees a program has a depth which is equal to the number of layers or the number of nodes on the longest route from the root to a leaf. The program shown in figure 4.1 has depth of 3.

The genetic programming is just an extension of the genetic algorithm, therefore the genetic programming will be more thoroughly described in chapter 5 where the genetic algorith and how it evolves is described. This chapter also describes how the genetic programming extends the GA.

Chapter 5

The genetic algorithm

As both the neural networks and the genetic programs is based on the genetic algorithm this chapter will work with bith the theoretical genetic algorthm as well as two instantiations of it: a neural network instantiation and a genetic program instantiation.

5.1 The algorithm

The genetic algorithm (GA) usually optimizes a list of numbers (called an individual in the rest of the thesis), e.g. bits, decimal parameters for a function or the like. It works as a black box optimization where a fitness function evaluates the current individual. Black box optimization means that the GA does not know how the fitness functions evaluates the fitness. The fitness is blindly used as a measure of how good the individual is.

The GA works by initially having a population of individuals. These individuals are evaluated using the fitness function and based on their fitness they are picked as parents for the new generation. The parents are then combined with each other in what is called crossover or reproduction to create offspring. These offspring contain some parts of both of their parents and will together with

```
pop = InitializePopulation()
stopped = false;
while (!stopped) {
  EvaluatePopulation(pop);
  stopped = CheckForStopCriteria();
  if (!stopped) {
    parents = Selection(pop);
    nextGeneration = Reproduction(parents);
    pop = Mutation(nextGeneration);
  }
}
```

Figure 5.1: Algorithm used for the genetic algorithm

their parents form the population of the next generation. Before evaluating and selecting parents again, some of the individuals are copied and the copy is mutated. Both offspring and parents are in the pool of individuals which might be mutated. This mutation is done to broaden the search and prevent the evolution from getting stuck in a local optimum.[Siv08]

The process of evolving continues until some stopping criteria is reached. This algorithm is shown in figure 5.1.

In the following sections each part of the genetic algorithm is described along with a description of how it is implemented to support the neural networks as well as the genetic programing. Each section is followed by a subsection for describing the implementation in the neural network and a subsection describing the implementation in the genetic programing.

5.2 Encoding the problem

The simple GA uses individuals encoded as bit strings, but in general the encoding only needs to be strings containing characters from a finite alphabet. [RN03]

5.2.1 Neural networks

There exists many ways to encode a neural network. Some allows the network to have cycles while others do not and some encodings also include the edge weights. In this thesis only the encodings without weights are taken into consideration. This is due to the fact that backpropagation is used to optimize the weights.

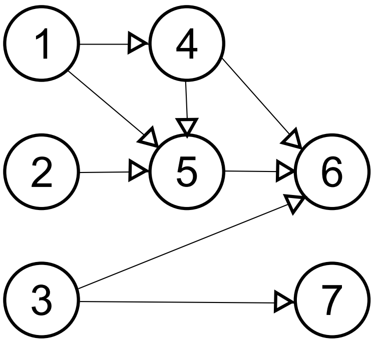


Figure 5.2: A simple neural network which must be encoded in order for the GA to work on it.

5.2.1.1 Direct matrix encoding

A simple way of encoding a network is to use a direct matrix encoding. In a DME the connections between the nodes are stored as 1's in a matrix otherwise containing 0's. As an example the neural network shown in figure 5.2 would be encoded as 0001100000010000000110000110000010000000000000000 which is the string representation of the matrix shown below.

	1	2	3	4	5	6	7
1	0	0	0	1	1	0	0
2	0	0	0	0	1	0	0
3	0	0	0	0	0	1	1
4	0	0	0	0	1	1	0
5	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0

The DME is easy to implement and is easy to manipulate through the GA. The encoding is a bitstring why the simple GA will work out of the box and any manipulation of the individual will result in a meaningful solution even though many of the solutions might be illegal.

One big problem with this encoding is that it allows illegal states to appear. The illegal states could be nodes without neighbors and worst of all loops occuring in the network. Loops are not allowed as they would cause the backpropagation algorithm to get stuck in an infinite loop.

Using this encoding the size of the string encoding would also get too big compared to the number of connections in sparse network structures.

5.2.1.2 Adjacency list encoding

Another direct encoding is the adjacency list encoding which is used in this project. It works by listing all the incoming connections for each node in the network. That is, if there are connections from node 2 and 3 into node 5 this would be encoded as $2 : 3 : 5 :$.[Sch00]

As neural networks distinguish between input, hidden and output nodes a header describing the number of input and output nodes is attached on the front of the encoding.

The neural network shown in figure 5.2 would be encoded as 030212314145345637. The first four digits is the header saying that the net has 3 input nodes and 2 output nodes. Then the following digits is the actual net structure. The decoding of the string is done such that everytime an old node (a node which has been seen earlier in the string) is spotted in the encoding string (when reading from left to right) it is added to a queue of incoming conections. Then whenever a new node is found, this node is added to the network and a connection is made from each node in the "incoming connections"-queue to the new node. Finally the "incomming connections" queue is cleared.

This encoding ensures that no loops can be found in the encoding as only connections going from a node which has been seen before to a node which has not been seen before is allowed. Further more the encoding length is not as big as the matrix encoding.

A major drawback of this encoding is that the ga can only perform crossover at a few crossover points and therefore a special mutation and reproduction has to be created.

5.2.2 Genetic programming

The genetic program is not encoded into a string. Instead the reproduction is created specifically for genetic programs as seen in section 5.6.2.

5.3 Initialization

Initially a random population is generated. This is usually done by either picking a number of individuals randomly from the uniform set of all possible individuals or by a human selecting what individuals to start with.

In this project there are an unlimited number of individuals as the neural networks can have an unlimited number of nodes and the genetic programs can be of infinite depth. Therefore tests are used to find good starting populations, see section 7.4 and section 7.5. In these tests the amount of hidden nodes and the depth of the gp's are found, but also the population size, the number of generations, the number of epochs along with various other parameters are found.

When the neural networks are created random weights are put on the synnapses. These weights are chosen randomly in the interval $[-10; 10]$.

5.4 Evaluation

The evolution of a good network structure/genetic program is done in iterations or generations. Each iteration starts with the population being evaluated. This is done by evaluating the classification rate and/or the size of the backpropagation error of each individual. For the neural networks this is a very computation heavy process as the network has to perform backpropagation on the problem in order to find the classification rate and backpropagation error.

The fitness of the individuals is then chosen as the classification rate, the size of the backpropagation error or a combination of these.

$$fitness = 0.7 \cdot classifiaction\ rate + 0.3 \cdot \frac{1}{1 + bp\ error}$$
$$fitness = classification\ rate$$

If the number of false negatives should be minimized this could also be included in the fitness evaluation.

$$fitness = 0.5 \cdot \frac{1}{10 \cdot FN} + 0.5 \cdot classification\ rate$$

In this project a high fitness means a good individual and vice versa.

5.5 Selection

When choosing what individuals are to included in the next generation selection has to be performed. If only the best individuals are chosen it is quite likely that even better individuals which could be found as a combination of a good and a bad scoring individual will never be found. At the same there it does not make much sense to spend too much time on combinations of bad individuals as their children are more likely to be bad as well. Therefore different approaches to the selection exists.

Uniform selection is the selection method which does not take the fitness into consideration. The individuals are chosen completely at random. The problem with this type of selection is of course that an evolution in the overall fitness will not occur. It works somewhat like a random search.

The roulette selection is the upgraded version of the uniform selection. Instead of picking the individuals at random they are chosen with a weighted probability where the weight is equal to their fitness. Do note that an individual can be chosen as parent multiple times.

As an example imagine that in a population with 5 individuals, 4 must be chosen as parents in the next generation. Their fitness scores, the propability with which they are chosen as parent and the roulette value are shown in the table below. The roulette value is the sum of probalities so far. Four times a number between 0 and 1 is chosen: $0.6429, 0.3240, 0.2630$ and 0.4098. The parent population is then build using the first individuald with a roulette value higher or equal to the random number. This results in the parent population being: { 4, 2, 2, 3}.

Individual	Fitness	Propability	Roulette Value
1	0.5	0.1613	0.1613
2	0.7	0.2258	0.3871
3	0.2	0.0645	0.4516
4	0.9	0.2903	0.7419
5	0.8	0.2581	1.0000
Total	3.1	1.0000	

The problem with both uniform and roulette selction is that the best individuals can be lost. Of course the overall best individual should be stored and used as the result, but often one would also want it to be used as parents for the next generations as well. To ensure this elitism can be used in combination with another selection. Elitism works by selecting the x individuals with highest fitness. These along with the parents chosen by the other selection is used as parents for the following generation.[Siv08]

In this project roulette selection and elitism are used.

5.6 Reproduction

The way the genetic algorithm differs from a randomized search is the way it uses reproduction to find new solutions. The idea behind reproduction is that a good indvidual might be good due to some *part*s of it being good. So in the same way the as in the nature offspring is created using different parts of the parents. Hopefully these parts are the parts that make the individual parents have a good fitness.

5.6.1 Neural networks

The reproduction in neural networks is a little difficult due to the special encoding.

First the possible crossover points are located. Crossover points are not allowed in input nodes and in the last output node. Furthermore crossover points are only allowed on synnapses between nodes. In the encoded string the synnapses are found where a number i is followed by a number $j < i$.

When the possible crossover points locations are found, a crossover point is selected in each of the parent networks. (This is done randomly) A crossover is then performed between the two. The crossover is done by taking the encoded string from start to the crossover point and combine it with the end of the other string and vice versa. Two new networks are created and is ready to be used in the evolution.

Before using them in the following generations the netwroks are pruned so that all nodes not used are removed. An example of a reproduction between two neural nets are shown in figure 5.3.

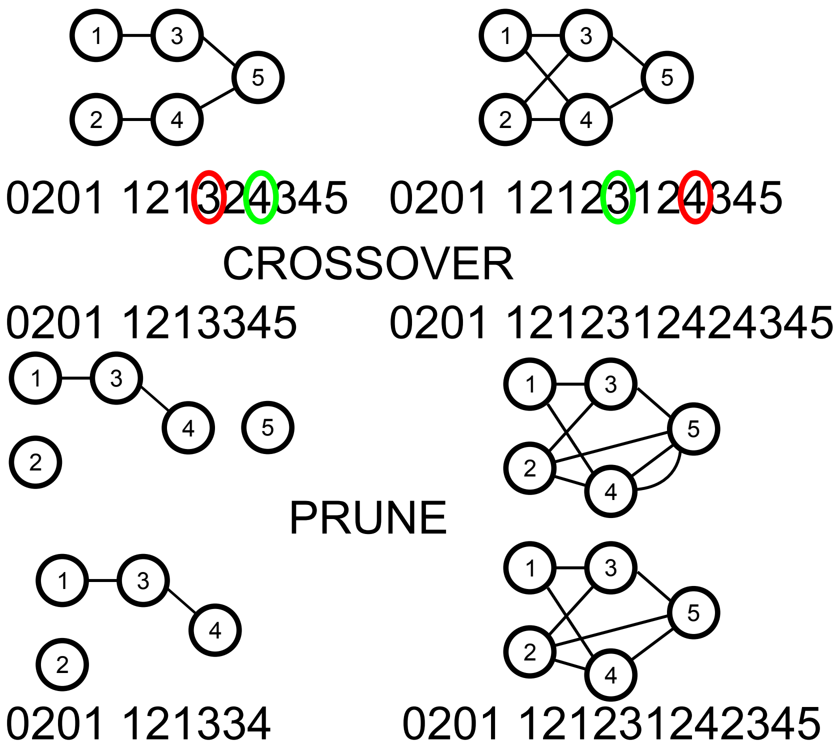


Figure 5.3: Crossover between two neural networks. The circles in the encoded strings indicate where the allowed crossover points are, the red circle show which crossover point is chosen. The pruning is done to remove all nodes which might interfere with the result: it is not allowed to have a hidden/output node without inputs, but any node is allowed to have no output.

5.6.2 Genetic programming

For the GP the reproduction works by finding 1 node in each parent which is to be the crossover point. The tree having this node as root in one parent is then switched with the tree found in the other parent. In this way two children are created having a part from each of the two parents. An example is shown in figure 5.4.

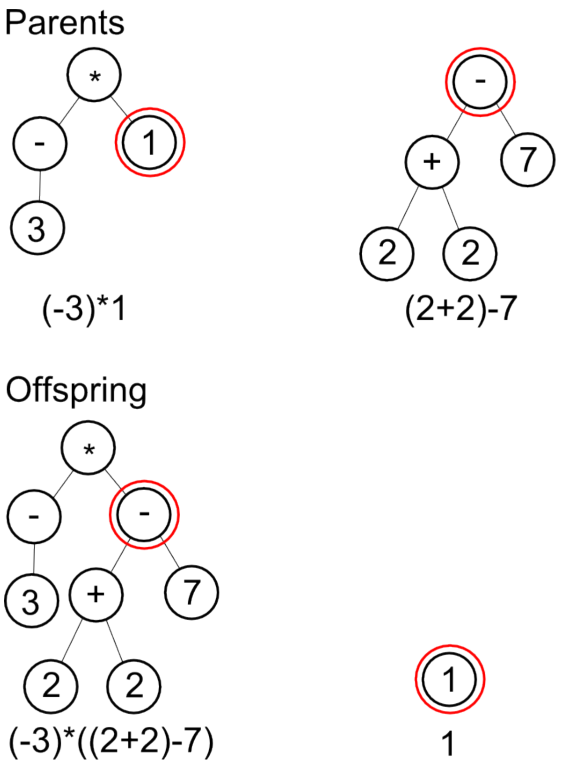


Figure 5.4: Crossover performed on two genetic programs resulting in one big and one small program. The red circles mark the location of the crossover point. As the root is chosen in the left tree and a leaf is chosen in the right tree, one of the children will only contain one node.

5.7 Mutation

If the genetic algorithm was only evolving using reproduction the population might eventually end up consisting of one type of individual. It might further more be liable to getting stuck in a local optimum.

To prevent this from happening mutation can be used. It makes a small change in some individuals and this mutation might be what was needed in order to kickstart/restart the evolution.

In the standard ga mutation is done by making a small change in the encoded string, for example by flipping a couple of bits or by setting one of the genes in an individuals to a new number.[Siv08]

5.7.1 Neural networks

In the neural network implementation of the mutation, the mutation is adding or removing a node from the network instead of working directly on the encoded string.

Only a subset of the population is chosen for mutation. Once chosen a coin is flipped to see if a node is to be added to the network or a node is to be removed from the network.

Choosing which node to remove is done by assigning a score value to each node. This score is calculated as:

$$score_n = \frac{1}{1 + |con\_in_n| + |con\_out_n|}$$

where $|con\_in_n|$ and $|con\_out_n|$ are the number of connections in and out of node n. Using the roulette principle also used in roulette selection (section 5.5) a node is randomly picked for removal. This weighting ensures that nodes with many connections are less likely to be removed.

If a node is to be added to the network it is given an id randomly chosen between 0 and the number of nodes in the network. Every node with higher id is then having its id increased by 1. Due to the encoding the order of the node id's are important, see section 5.2.1.2.

Ingoing and outgoing connections are then randomly added to the the node by making a connection from each node in the set of nodes with lower ids

($nodes\_in_n$) to the new node (n) with chance:

$$chance\_in_n = \frac{2}{|nodes\_in_n|}$$

Likewise a connection is made from the new node to each node in the set of nodes with higher ids with the chance:

$$chance\_out_n = \frac{2}{|nodes\_out_n|}$$

In this way the expected number of ingoing and outgoing synnapses are 2.

5.7.2 Genetic programming

Mutation in genetic programming could be switching the type of function in a randomly chosen node or switching two nodes in the tree, but currently no mutation is used in the genetic programming.

5.8 Stopping criteria

The genetic algorithm keeps running until some stopping criteria is reached. This criteria is usually when the algorithm has run for a fixed number of iterations or when the fitness of the best individual or of the entire population starts to converge.[Siv08]

In this project the genetic algorithm stops after a certain number of iterations.

Chapter 6

Design and implementation

To make it easier to test different neural network implementations, different genetic programs, different mutations etc. the software is build in a modular manner where every part is easily replaced.This is done by having the main functionality created using interfaces and abstract classes. A UML diagram giving an overview of the classification software can be found in figure 6.1.

Evolution is the backbone of the genetic algorithm. It knows in what order the evaluation, reproduction and mutation is to be called and it knows when to stop the algorithm.

What happens when reproduction and mutation is called is determined in the class *GABase* or in the classes implementing it. As both the genetic programming and the neural networks use genetic algorithms they both implement the interface. The class for the genetic programing (*NormalGPB*) knows exactly how to evaluate, reproduce and mutate itself while the neural networks mutates and reproduces differently depending on the encoding used (*AEncoding*). Further more the neural networks evaluate depending on the learning algorithm used (*ILearningAlgorithm*).

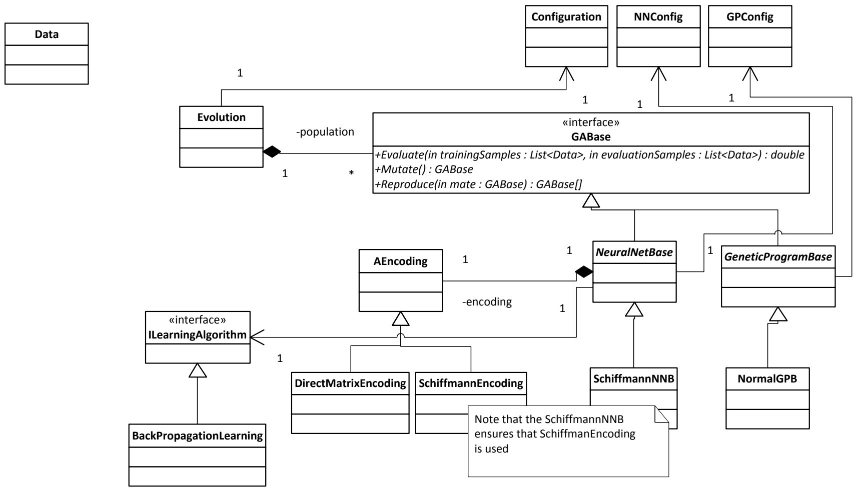


Figure 6.1: An overview of the design of the classification software.

6.1 Cloud solution

To get more computation power the software is furtermore developed to do its computations on multiple cores. To prevent having to struggle unnecessarily with the parallellization only the evaluation is done in parallel. Furthermore it is only the evaluation (the backpropagation for the neural network), which takes a lot of time, that is run in parallel.

To speed up the evaluation process even further only individuals that has not been evaluated will be evaluated. The evaluation is shown in figure 6.2. The process begins with finding the individuals in the population which has not yet been evaluated. This list is shuffled such that the distribution of small/large individuals is uniform. Then every individual is evaluated. This is done in parallel using the build in function Parallel.ForEach. All results are stored in the individuals themselves to prevent race conditions. Once all nets have ben evaluated they are marked evaluated. Once again the marking is not done in parallel as this could infer race conditions.

```
/* Evaluate in parallel */
/** Prepare for evaluation **/
List<GABase> toEvaluate = new List<GABase>();

foreach (GABase nnb in m_Pop)
  if (!evaluatedIndividuals.Contains(nnb))
    toEvaluate.Add(nnb);

/*  Shuffle list to encourage even distribution of individuals  */
/*  between cores                      */
toEvaluate = toEvaluate.OrderBy(a => Guid.NewGuid()).ToList();
int noEvaluated = toEvaluate.Count;

/** Do the actual evaluation **/
Parallel.ForEach(toEvaluate, nnb =>
{
  nnb.Evaluate(m_Problem.TrainingData, m_Problem.ValidationData);
});

foreach (var nnb in toEvaluate)
  evaluatedIndividuals.Add(nnb);
```

Figure 6.2: The evaluation of the individuals of the genetic algorithm. First the individuals which are to be evaluated is found, and the list is shuffled. Then the individuals are evaluated in parallel and finally they are marked as evaluated.

6.2 Debugging

In order to ensure that the software is working correctly a lot of debugging and analysis of the different parts of the software has to be done. This analysis and debugging takes a great amount of calcualtions why most of the debugging is done only when a debugger is attached. An example of this can be found in the class *BackPropagationLearning* where the classification rate is stored in the debugger if it is present.

```
if (debugger != null)
{
  var classificationRate = network.GetClassificationRate(
    debugger.EvaluationData, this.actFunc, this.finalActFunc
  );
  debugger.ClassificationRateHistogram.Add(classificationRate);
}
```

CHAPTER 7

Testing

In order to proove the correctness of the software and validate the effectiveness of it, a number of tests have been performed. In the following sections these tests are described and the test results are presented and discussed.

First the correctness of the individual parts of the software is tested to see if they are functioning. This is done in section 7.1 and 7.3. Then the parameters for the classification algorithms is tested in order to find a good set of parameter settings. This is done in section 7.4 and 7.5.

In section 7.6 and 7.7 the best classification algorithm and its performance and capabilities are measured.

7.1 Initial testing

In order to ensure that the classification algorithms are working correctly tests have been run on a number of benchmark problems [**Prechelt94**]. These benchmark problems have been used by others to test the classification rate of their algorithms and can therefore be used to evaluate the performance of the classification algorithms even without a fully functional feature extraction algorithm.

The classification algorithms are not fine tuned to get an optimal score on these benchmark problems. The benchmark problems are only used to see that the classification algorithms is able to perform decently.

The neural network used in the classification is an unsorted network with 6 hidden nodes and has the following parameter settings:

Learning rate	Falling from 0.7 to 0.03
Population	50
Mutation rate	0.6
Epochs	500
Generations	100
Elitecount	5

The genetic programing algorithm has an initial population consisting of 50 programs with depth 8. It has been running for 5000 generations, else the parameters are as above.

In the following section the datasets used for benchmarking is described and the test results are shown

7.1.1 Glass

The Glass training set has 9 decimal inputs and 6 boolean outputs. Only one of the outputs can be true and it is therefore similar to the melanoma problem (described in section 2.1.8) with the exception that there are 6 ouput classes instead of 2. As the genetic program is only able to result in 2 different classes this test set is only be used to validate the neural networks.

In table 7.1 the test results of other classification algorithms can be found and in figure 7.1 the results are visualized. The neural networks developed in this project is in the lower end when it comes to classifying these data, but as it is not developed to maximize the classification rate for these problems the classification rate is good enough.

7.1.2 Diabetes

The Diabetes training set consists of 768 samples of 8 inputs and 2 boolean outputs. This problem is therefore very similar to the melanoma problem which

Litterature	Method	CR
Fredriksson, 1997 [Fre97]	ANKKA	65.00
	Cascade Correlation	79.00
Dorsey, 2000 [Dor00]	BP	51.19
	GA	66.79
Grönroos, 1998 [Grö98]	Miller et. al	61.82
	Kitano	67.92
	Parisi	50.31
	Cangelosi	37.54
Prechelt, 1994 [Pre94]	PROBEN1	67.92
PedersenJensen, 2004 [PJ04]	GANN - RPROP / Schiffmann	79.25
Mygind, 2011	GANN / Schiffmann	62.26

Table 7.1: Proben glass test results. The column CR shows the classification rate.

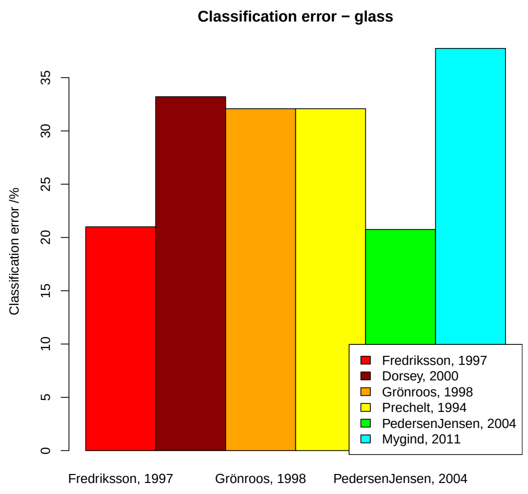


Figure 7.1: Results of other work using the proben/glass benchmark set.

has 9 inputs and 2 outputs. This data set can also be used to validate the genetic programming why this is an important data set for verifying the capabilities of both classification algorithms.

In table 7.2 the results of classification algoirthms used by others are shown and in figure 7.2 the results are visualized. It is seen that both the neural networks and the genetic programs have a classification rate comparable to those of other litterature. They do not score highest, but this was expected as the parameters are not fine tuned to these problems.

Litterature	Method	CR
Banzhaf, 1998 [BB98]	Genetic programing	76.92
Dorsey, 2000 [Dor00]	BP	72.39
	GA	73.76
Fredriksson, 1997 [Fre97]	ANKKA	77.10
	Cascade Correlation	76.60
Ma, 1997 [JM97]	Combined Weak Classifiers	77.30
	K Nearest Neighbour	74.20
	Neural Networks	76.48
	Combination of GA og ANN	77.21
Sikander, 2001 [SS01]	BP	72.39
	GA	76.56
Prechelt, 1994 [Pre94]	PROBEN1	75.00
PedersenJensen, 2004 [PJ04]	GANN - RPROP / Schiffmann	82.29
Mygind, 2011	GANN / Schiffmann	78.13
	Genetic Programing	73.44

Table 7.2: Proben diabetes test results. The column CR shows the classification rate.

7.1.3 Horse

The samples in the Horse data set consisist of 58 inputs, most of them are 0, and has 3 boolean outputs. This dataset is used to see how the neural network will perform with a lot of missing input data, which will most likely not be relevant as the feature extraction most likely will always be able to find a number for each of the 9 features. Due to the fact that there are 3 output classes the dataset can only be used to validate the neural networks.

In table 7.3 the test results of earlier work can be found and in figure 7.3 the results are visualized. It is seen that the classification rate of the neural network

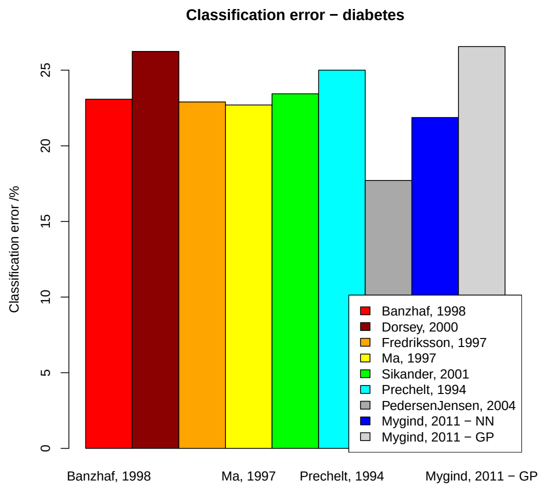


Figure 7.2: Results of other work using the proben/diabetes benchmark set.

is comparable to the classification rate of other litterature.

7.2 The dataset

The initial testing was done using the datasets described by Prechelt [Pre94], but the following tests will be performed using a dataset obtained from the dermatologic lab at Rigshospitalet.

The dataset contains images of 336 birthmarks of which 164 have been classified melanoma by the dermatologists and the rest has proven to be harmless. So 51% of the images are harmless and 49% are harmful. As the distribution is 50-50 ot will be harder to get a high classification rate compared to a dataset with a more uneven distribution.

The images have been taken using a standard camera operated by one photographer. The quality is varying and especially the luminance of the images vary. All images are sharp and the entire birthmark is always found inside the image. The test data set is therefore a little more controlled and the image quality is a little more stable compared to what could be expected in a real world implementation. Here different users would use different cameras and might have

Litterature	Method	CR
Banzhaf, 1998 [BB98]	Genetic programing	78.65
Bay, 1998 [Bay98]	Nearest neighbour	76.80
	K nearest neighbour	79.80
	Nearest neighbour forward selection	83.90
	Nearest neighbour backward selection	76.50
Dorsey, 2000 [Dor00]	BP	71.28
	GA	76.00
Prechelt, 1994 [Pre94]	PROBEN1	73.63
Sikander, 2001 [SS01]	BP	74.72
	GA	80.22
PedersenJensen, 2004 [PJ04]	GANN - RPROP / Schiffmann	85.71
Mygind, 2011	Neural Networks	79.12

Table 7.3: Proben horse test results

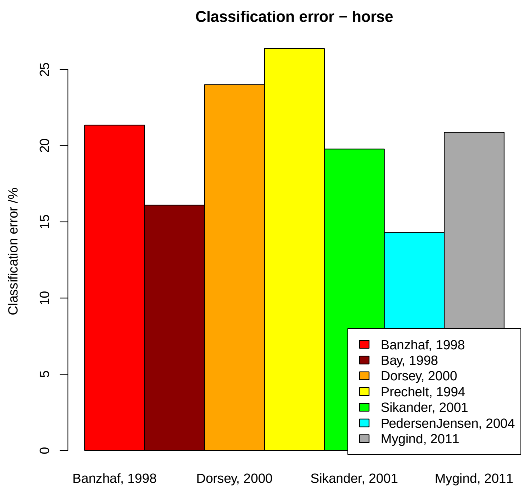


Figure 7.3: Results of other work using the proben/horse benchmark set.

even more differing luminance on the subjects.

The birthmarks vary in both color and size and only a few does not fit inside the image. No image has been removed from the dataset even though it could be argued that birthmarks placed on rough areas, like the ear, should be removed. Due to this some birthmarks will inevitably have bad influence on the features extracted from these images.

All images are of images of birthmarks found in the dermatological laboratory at Rigshospitalet. The birthmarks are therefore all in the category where the house doctor has not been able to say that the birthmark with certainty is harmless. The dataset therefore does not contain any, or only very few, images of the standard harmless birthmark.

The training of the classification algorithms has been done using data found by the feature extraction. The performance of the classification algorithms is therefore only comparable to each other and not to classifications of other work. Also this has the effect that a classification rate of 100% is not necessarily reachable as the classification algorithm cannot perform better than the feature extraction allows.

The dataset is split into three: a training dataset, a validation dataset and a test dataset. These dataset are chosen at random and has the sizes 167, 84 and 85 respectively. The training dataset is larger as this is the main source of training and will ensure that the training is done a broader set of images making it more tolerant to different images.

For the neural network the training dataset is used to train the backpropagation and based on the score dound using the evaluation dataset the best network is chosen by the genetic algorithm. For the genetic programming the only training set is used to train the genetic program structure. When comparing results the test dataset is used for both the genetic program and the neural network.

7.3 Feature Extraction

In this section the feature extraction is tested. Initially random checks is used to find out if the algorithm is capable of extracting the areas of the birthmarks. Once a fitting algorithm has been found the robustness of it regarding the quality of the input images is tested.

7.3.1 Random sampling

The most basic way of testing the feature extraction is by seeing what information the feature extraction works with. To do this prints of the regions in which the algorithm is working has been analysed by hand. Examples of such prints is found in figure 7.4.

These initial tests showed that the feature extraction is not robust enough why the test dataset was cropped to be more consistent (the birthmark is centered in the image). A small group of birthmarks was found by both implementations, one of these is shown in figure 7.4 on the right. A big group was found only by the new implementation where one of these birthmark is shown in the middle.

Even after the cropping some birthmarks is not found correctly by any of the implementations, but the number of birthmarks which are not correctly found is lowered. On the top right an ear confuses the algorithm and in the bottom right the bright birthmark and the freckled skin confuses the feature extraction.

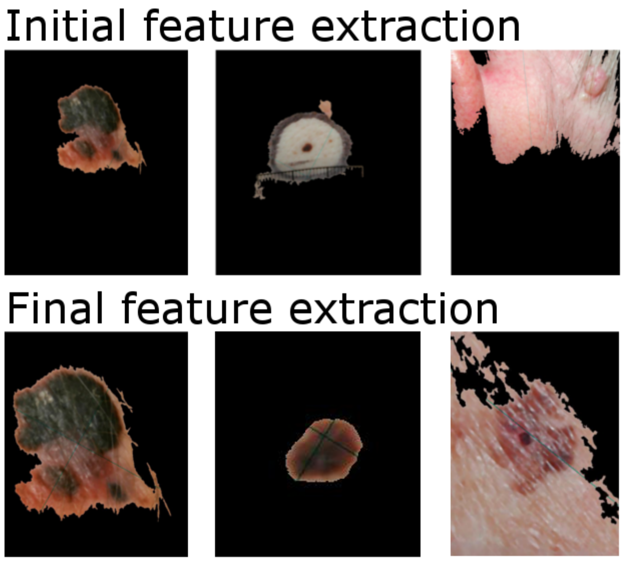


Figure 7.4: Testing the feature extraction using random samples. Upper row shows three examples of samples in the initial implementation. Second row shows for the final implementation. Both implementations have trouble finding some of the birthmarks, but the final is improved a lot.

During the random sampling all extracted images are classified by hand and given one of three categories. These categories, also visualized in figure 7.5, are as follows:

good where the imageextraction has found the exact birthmark in the image

mediocre where most or the birthmark is found or only a part of the surrounding skin has been included

bad where nothing is found in the image, half the image is marked as the birhtmark or the feature extraction has found the birthmark to have a shape completely different from the orignal shape of the birthmark.

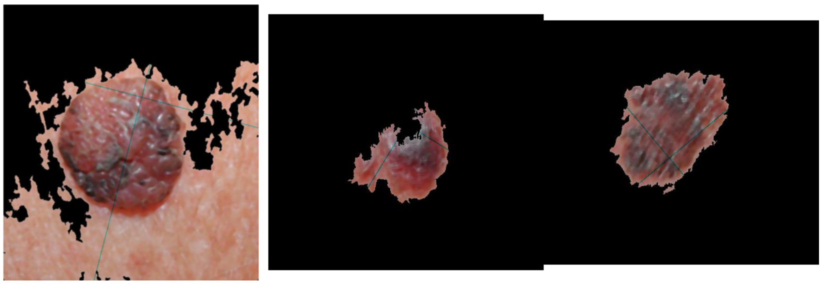


Figure 7.5: Images extracted by the image extraction algorithm. On the left a bad performance is shown, in the middle a mediocre performance where only a part of the birthmark is missing and on the right a good performance is shown where the entire image is correctly located by the feature extraction.

Using this categorization the feature extraction is able to find a birthmark in 214 of the 336 images. In 77 images the feature extraction performs really bad and in 45 images its performance is mediocre. Using only the good images, the performance of the feature extraction is therefore only 63.69%.

In the following all of the images with the bad feature extraction are used for testing. This is done because a lot of testing was done prior to this test and in order to compare these results to the following, all tests have been performed using the faulty dataset. In section 7.6.1 a test using the best network structure is performed in order to see the effect of these faulty images being removed.

7.3.2 Influence of image resolution

At least one of the business ideas for this software is that the hardware (the camera and the computer) should be more or less irrelevant for the effectiveness of the software. It is therefore important that the quality of the image is not very crucial for the outcome. In this section the influence of the image quality is measured.

The resolution of the image is very dependent on the camera. First of all the resolution of a standard cameras range from $1MP$ and up to more than $15MP$ ($1000 \times 1000px$ - $3900 \times 3900px$). Further more some cameras cannot put focus on objects close to the lense why the birthmark might not fill the entire image.

In order to find a lower bound of the birthmark size measured in pixels an image of a birthmark has been captured using the camera of a decent phone using its auto focus feature. The image is then cropped to fit the size of the birthmark and this size is used as the lower bound. The minimum size is found to be $200px \times 200px$.

All images are then resized to size $200px \times 200px$ and to size $400px \times 400px$ and classified using a good neural network from a training performed on the original sized images. An example of an image being resized is shown in figure 7.7. It can be seen that lowering the resolution results in an image of a birthmark with much lower border sharpness.

The classification rate and the rate of false/true positives/negatives are shown in figure 7.6.

From the figures it is seen that the resolution does have an impact on the classification rate and it can therefore be concluded that the resolution has an impact on the feature values extracted from the image. The tests show that the classification rate drops from 80% down to 70% when the image resolution is set to $200px \times 200px$ why a high resolution should be preffered in a final product. The classification is however still decent when the images are in a low resolution.

What is also interesting to see is that when the image resolution drops, the classification algorithm is more likely to classify the images as harmful. The false negative rate drops to 0% and the classification rate stays at 70%. This classification rate is not reachable with the high resolution images when the false negative rate has to be 0%.

7.4 Neural Networks

After having done the initial testing where it is seen that the classification software is working correctly, tests have been performed to find a set of parameters which results in a good classification network for melanoma data.

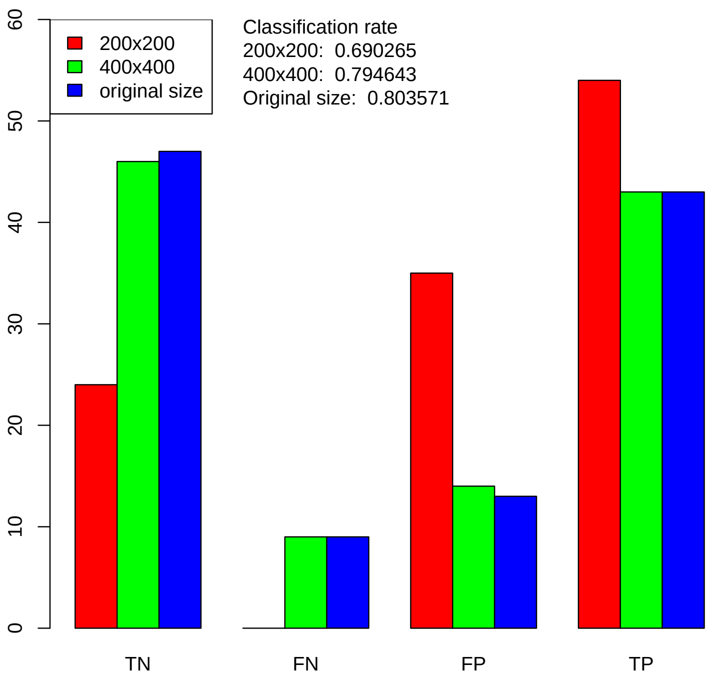


Figure 7.6: Testing the influence of the resolution of the image. The best neural network from a training using the original images are used to validate.

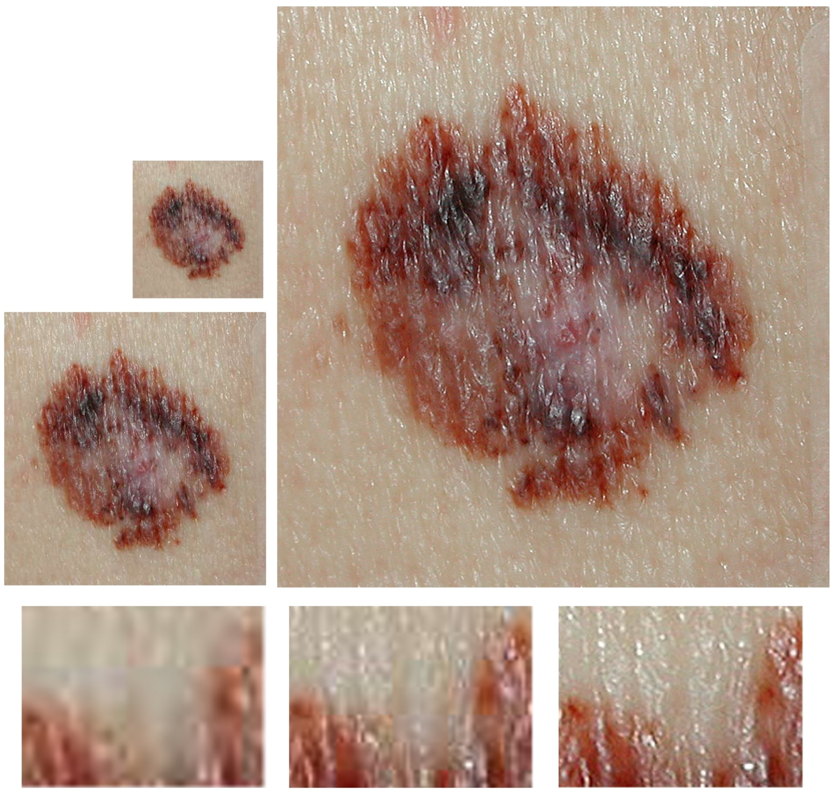


Figure 7.7: The same birthmark in different resolution. The bottom row shows a part of the image in low, medium and original resolution.

First the optimal number of epochs used in the backpropagation is found. Next the best network structure is found and finally the threshold for when to classify a birthmark as harmfull is determined.

7.4.1 Optimal number of epochs

In order to train the weights of the network, backpropagation is used. The network should be trained long enough for it to classify the training set correctly. At the same time it should not be trained for so long that it is overfitted to only classify the training set correctly. To find out how many epochs the backpropagation algorithm should run through, tests has been made where the classification rate of the network is evaluated every epoch.

In figure 7.8 graphs are found which shows the test results (the classification rate of a neural network every epoch). The test has been run with 4 diferrent learning rates, 3 times each.

It is seen that a learning rate between 0.03 and 0.4 gives the best result, but a variable learning rate is almost as good as the best as well. Further more there is no clear indication of overfitting as the graphs does not have a negative slope. Furthermore it is seen that the classification rate is not increased noteworthily when the algorithm has run for more than 500 epochs.

7.4.2 Finding a good structure

To find a good network structure different types of multi layer networks have been tested against the perceptron network and the unstructured network. Initially the unevolved network is testet to sort out the types which is not worth spending too much testing time on. The best of the tested networks will be used as start population for an evolution and hopefully a combination of the networks will result in even better networks.

7.4.2.1 The unevolved network

In this section the unevolved network, a network only trained using backpropagation, is used to classify the birthmarks. The edges of the networks are initialized using random weights in the interval $I = [-10; 10]$ and the networks are trained using backpropagation. A variable learning rate is used as this has

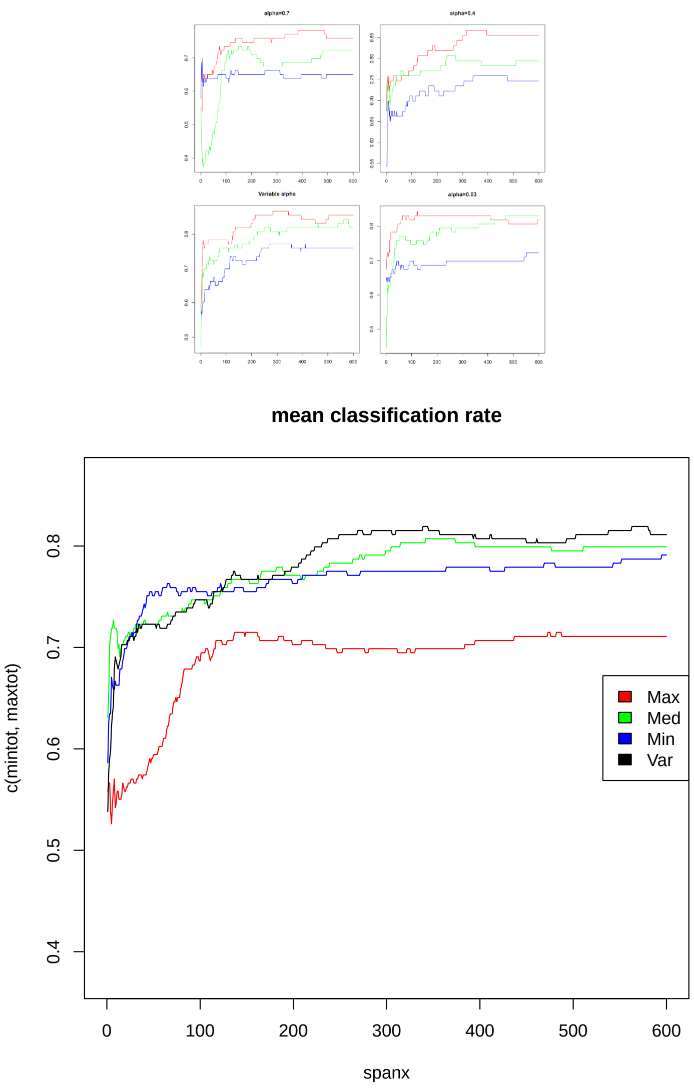


Figure 7.8: Graphs showing the classification rate as a function of the number of epochs. Three runs with each parameter setting is made. The first three graphs show runs with a fixed learning rate of 0.7, 0.4 and 0.03 respectively; And the fourth graph shows a run with a learning rate decreasing linearly from 0.7 to 0.03 over time. The last graph shows the mean classification rate of the three runs for each parameter setting. A larger version of the four first graphs can be found in appendix B.

proven to be effective in other tests (see section 7.4.3.). The learning rate is $\alpha(it) = 0.7 - \frac{it}{it_{total} \cdot (0.7 - 0.03)}$. That is the learning rate falls linearly from 0.7 to 0.03 which are the extremes of the learning rates mostly used in the litterature. [**FreemanSkapura1991**][KB94]

Finding the optimal number of hidden layers Three types of networks are tested: the perceptron, the unstructured network and the multi layer network. To find out what types of multi layered networks are to be compared to the other networks an initial test is made using 10 hidden nodes, but with varying structure: a varying number of hidden layers and a varying number of hidden nodes in each layer.

The results of the test can be seen table 7.4 and is visualized in figure 7.9. It is clearly seen that when the backpropagation algorithm runs for at most 500 epochs the multi layer network should not have too many hidden layers.

The tables are read using the following legend:

ML x means a single layer network containg x nodes.

ML $x - y$ is a multi layer network with 2 layers containing x and y nodes respectively.

US x is an unsorted network with x hidden nodes.

Type	Run 1	Run 2	Run 3
US10	0.80723	0.722892	0.746988
ML10	0.710843	0.626506	0.626506
ML5-5	0.638554	0.638554	0.722892
ML4-3-3	0.626506	0.614458	0.493976
ML3-3-4	0.626506	0.493976	0.493976
ML2-3-3-2	0.506024	0.493976	0.493976
ML2-2-2-2-2	0.674699	0.493976	0.493976
ML1-ˆ10	0.493976	0.493976	0.493976

Table 7.4: Test results from running the classification algorithm on different network structures.

From the test results it is seen that the best classification rate is bad in networks containing more than 2 layers. Therefore only the unsorted network and the multilayer network with 1 or 2 hidden layers are tested.

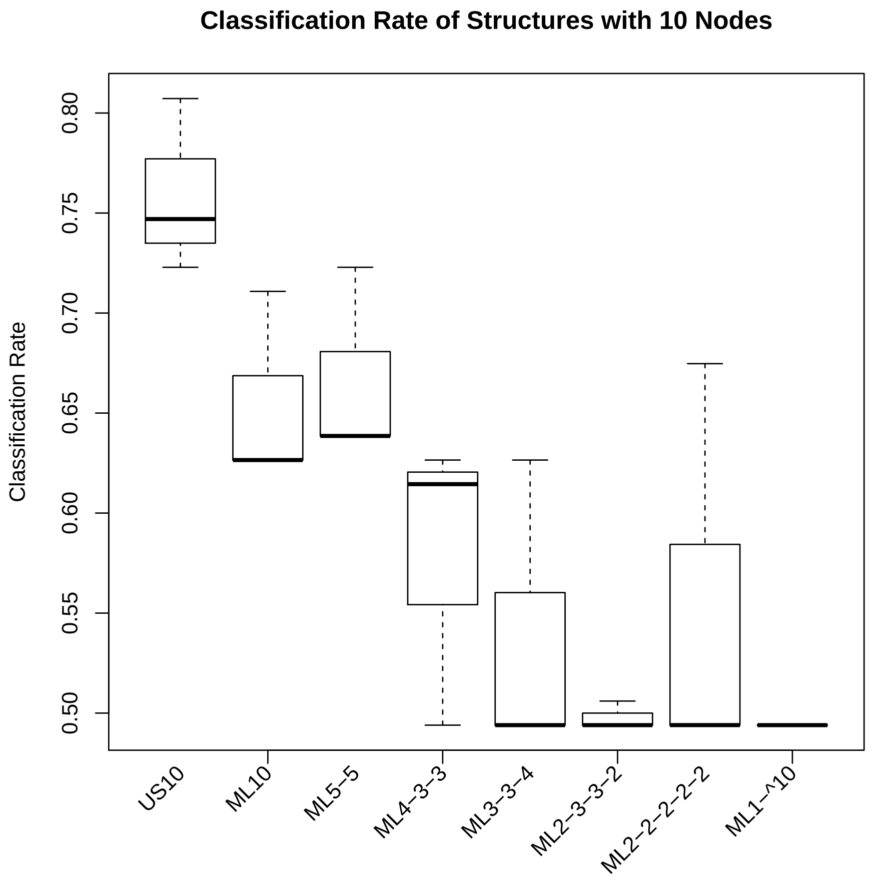


Figure 7.9: Classification rate of different network structures with 10 nodes. Three runs has be made for each structure.

Finding the best number of hidden nodes As we found out that the multi layer network with 1 or 2 hidden layers performs best a new test using these two types of networks and the unsorted network is performed. The aim of the test is to find the best number of hidden nodes in a network.

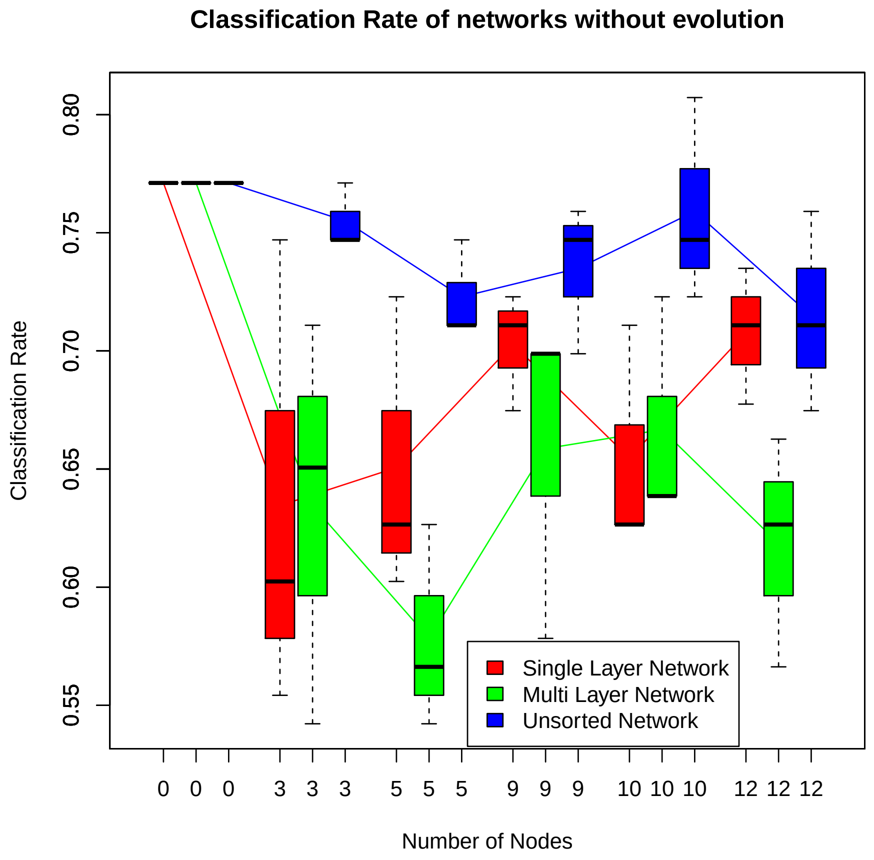


Figure 7.10: Classification rate of neural networks initialized as unsorted (US), single layered (SL) and dual layered (ML) networks having varying number of hidden nodes. The multilayer networks have one hidden layer.

Once again the network structure is not evolved, only the weights. The parameters mentioned above is still used.

From the results, shown in figure 7.10, it is seen that when the structure is not evolved the perceptron results in a very high classification rate. Further more it is seen that extending the perceptron, as it is done in the unsorted network, can result in an even better classification rate. This might indicate that evolving the perceptron or the unsorted network will result in the best classification rate.

7.4.3 Finding a good threshold

If and when this software is to be included in a real world application a low amount of false negative (FN) results would be prefered. If there is a a slight chance a person is having melanoma the application should tell the user so. Therefore a threshold has to be found which ensures that most melanomas are found without ending up saying all birthmarks are harmfull.

A test has been run to see what the values of the output nodes in the neural network is for harmless and harmful nodes respectively. The results, shown in figure 7.11, 7.12 and 7.13, can then be used to find a threshold for when to say a birthmark is harmfull or harmless. If the classification is based on the difference (the result is shown in figure 7.11 on the right) between the two output nodes the equation would look like this:

$$\text{Classification} = \begin{cases} \text{Melanoma} & \text{if } output_{negative\_node} + threshold \\ & > output_{negative\_node} \\ \text{Harmless} & \text{if } output_{negative\_node} + threshold \\ & \leq output_{negative\_node} \end{cases}$$

On the left of figre 7.11 the distribution of the output values of each node is shown. On the right the percentage of wrong classifications ($PoWC$) of harmful/harmless birthmarks is shown. Ideally 0%of the melanoma is wrongly classified while the $PoWC$ of the harmless birthmarks is kept very low.

From the graph it is seen that the $PoWC$ for harmless birthmarks increases much faster than the $PoWC$ for melanoma decrease. Therefore the classification rate will decrease too much if the false negative rate should be lowered noticeably. A threshold value of $t = 0.365$ looks like it would give a nice result where only few melanomas are not spotted and most harmless birthmarks ($\approx 53\%$) are classified as harmless. This point is marked with blue on the right in figure 7.11.

The test results are also used to see if a better classification rate and false-negative rate can be found if only one of the output nodes are used. These results are shown as “Positive node” and “Negative node” in figure 7.12. If only

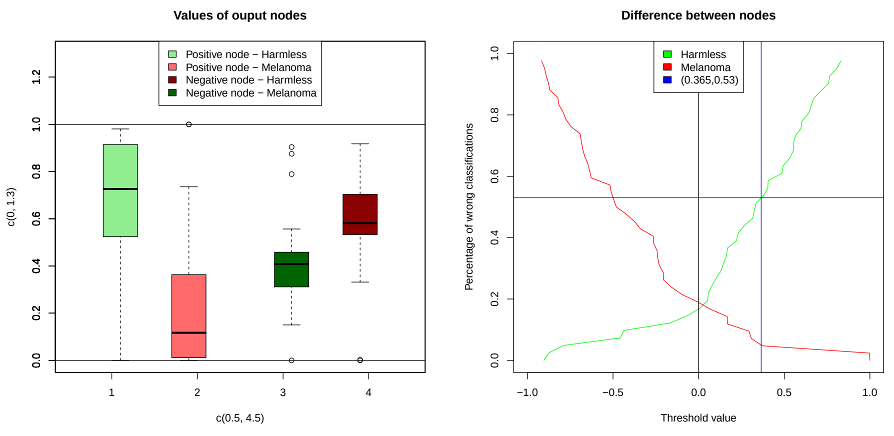


Figure 7.11: Graphs showing the test results used to determine a good threshold for the best neural network

one output node is used the equation with the threshold would be one of these:

$$\text{Classification} = \begin{cases} \text{Melanoma} & \text{if } output_{positive\_node} < threshold \\ \text{Harmless} & \text{if } output_{positive\_node} \geq threshold \end{cases}$$

$$\text{Classification} = \begin{cases} \text{Melanoma} & \text{if } output_{negative\_node} > threshold \\ \text{Harmless} & \text{if } output_{negative\_node} \leq threshold \end{cases}$$

From figure 7.13 it is seen that there is no big difference between using both output nodes (which is done during training) and using only one of the outputnodes. The slope of the individual curves are almost identical and the point where the lines cross is almost at the same threshold and has almost the same classification rate.

There could be a small advantage in using only the positive node because the number of false negatives could be lowered without sacrificing too much of the classification rate. The advantage is however only very small and might not be present in a different dataset.

7.4.3.1 Punishing the pressence of false negatives

The test results show that the classification rate drops very quickly when the number of false negatives are lowered by moving the threshold. Therefore a test

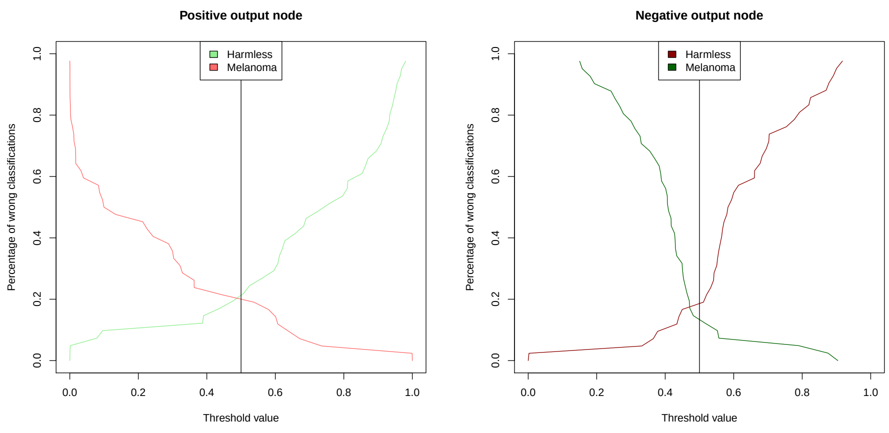


Figure 7.12: Graphs showing the test results used to determine a good threshold for the best neural network

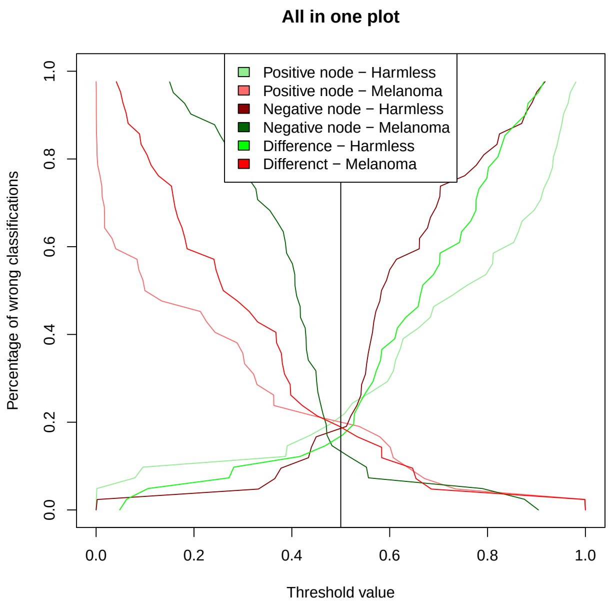


Figure 7.13: Graphs showing the test results used to determine a good threshold for the best neural network

has been run where the fitness are calculated using the number of false negatives as well.

$$fitness = 10^2/(10^2 + FN^2) \cdot ClassificationRate;$$

where FN is the number of false negatives found in the test.

The test however did not show any great improvements. The classification rate was much lower and the number of false positives where lowered a little.

7.5 Genetic Programming

As a simple genetic programming implementation has been chosen there is not many parameters to test, but in the following sections tests to find the depth of the program tree and an optimal threshold will be performed.

7.5.1 Finding a good depth

One parameter to test is the depth of the individuals in the initial population. As the initialization is done using complete binary trees, the depth has a direct influence on the number of leaves on the program tree.

$$leaves = 2^{depth-1}$$

where a tree containing only one node has $depth = 1$.

If the number of leaves gets too low it will be hard for the program to find a solution using all the inputs from the feature extraction as well as getting enough constants to add to these inputs. On the other hand a tree which is too deep will also result in the algorithm being unable to find a good result as the amount of possible solution will explode and finding a good will be like finding a needle in a haystack.

To find out what depth is best, a test has been made in which different combinations of depths and number of generations has been tried out. The test results are shown in figure 7.14. The tests with $250 - 1000$ generations have not been run with depth 14 due to the time span these tests would require.
It is seen that a depth 8 resulting in 128 leaves/terminals and 64 functions in the individuals of the initial population gives the best result. Especially if the number of generations are not too high. That a high number of iterations has a negative effect on the outcome is a sign of overfitting where the genetic program

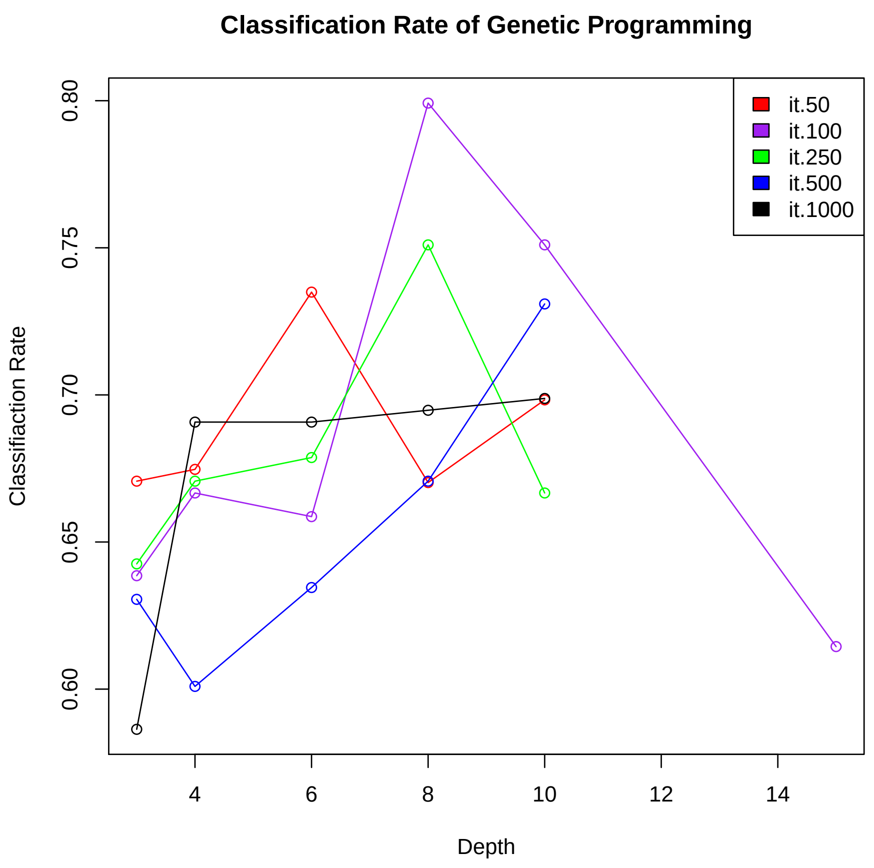


Figure 7.14: Classification rate of genetic programs with varying depths and running different number of iterations. Each line corresponds to a different number of iterations.

is too specialized to the training dataset and therefore has bad performance on the evaluation dataset.

7.5.2 Finding a good threshold

For the genetic program it is also possible to move the threshold for when the birthmark should be classified as melanoma or not. In figure 7.15 two graphs showing the test results are found. On the left it seen that if the threshold

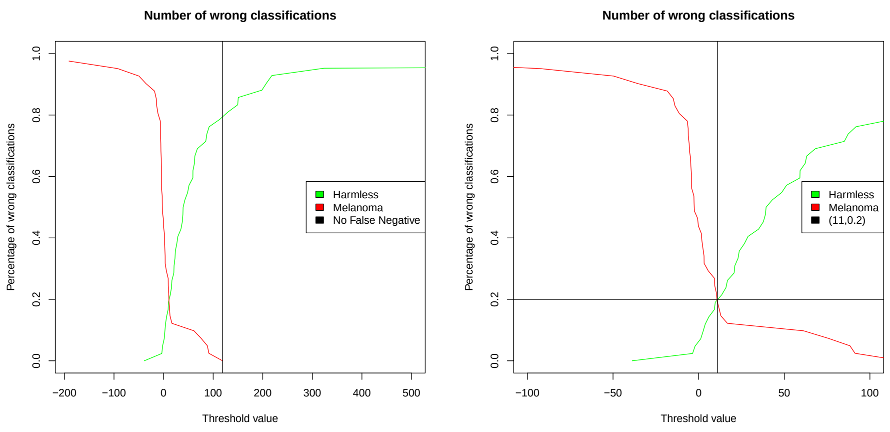


Figure 7.15: Threshold test for genetic programing. Number of false positives/negatives at different threshold levels. The black lines in the left graph shows the location where no false negatives are found. In the right graph it shows the location where both the false negatives and the false positives are minimized.

is around 115 no false positives will be found. At the same time 80% of the harmless birthmark will be classified as melanoma. On the right graph, view is zoomed in around threshold 0 it is seen that the lines cross at $(11, 20)$ which means that the best classification rate will be with a threshold of 11. The classification rate will then be around 80%.

Increasing the threshold value a little to approximately 20 will result it what looks as a good compromise between a good classification rate and a low false negative rate.

50 Generations	
Type	Classification rate on training data set
Perceptron	76.51
ML 9	83.73
ML 5-4	50.12
ML 10	78.31
ML 5-5	54.22
US 10	85.52
US 9	67.47
US 3	84.94

Table 7.5: The best neural networks - 50 generations

7.6 The best classification algorithms

Having found a good set of parameters for the neural networks a longer test is made with fewer variations in the parameter settings. Only the perceptron, the single layer, the dual layer and the unsorted neural network is tested.

In this test the networks are evolved such that small mutations made in the network might open up for new structures being even better at classifying the birthmarks.

A test run with the following parameter settings is made for the different networks.

Learning rate A linearly falling learning rate from 0.7 to 0.03.

Population A fixed population size of 50.

Mutation rate A fixed mutation rate of 0.6.

Epochs A fixed number of epochs. 500.

Elite count A fixed elite count. 5.

The test are evaluated using the score on the test data set (on which they are trained). Based on the score of the best network/genetic program a final classification rate is chosen based on both the test dataset and on the classification rate on the entire set. The results are shown in table 7.5 and 7.6.

These results are compared to the results of the test of the genetic programing done in section 7.5 where the best programs are shown in table 7.7.

400 Generations

Type	Classification rate on training data set
Perceptron	86.75
ML 9	82.53
ML 5-4	58.43
ML 10	N/A
ML 5-5	63.25
US 10	N/A
US 9	69.88
US 3	59.64

Table 7.6: The best neural networks - 400 generations. Due to the long running time some tests were not performed - therefore N/A is shown.

Genetic programing

Depth	Iterations	Classification Rate
8	100	79.92
10	100	75.10
8	250	75.10
6	50	73.49

Table 7.7: The best genetic programs found in section 7.5.

Using these results it is seen that the best classification rate is received using the perceptron as basis for the initial populations and by evolving the structure of the neural network for 400 generations. By doing so a classification rate of 86.75 on the test dataset is found.

Using the best network to evaluate the entire set and on the test data only, the classification rates shown in table 7.8 are found. It is seen that the neural networks are better at classifying the birthmarks when all images are used, and it is seen that is able to classify 78% of the images correctly.

All images	Type	Classification rate on test data set	Classification rate on entire set
	NN	78.82%	77.98%
	GP	70.69%	75%

Table 7.8: The results of the best classification algorithm.

Good images only	Type	Classification rate on test data set	Classification rate on entire set
	NN	91.84%	80.57%
	GP	89.80%	82.46%

Table 7.9: The results of the best classification algorithm.

7.6.1 Performance on good images only

As shown in section 7.3.1 the feature extraction is only able to locate the birthmark exactly in some of the images. Therefore a single test has been run to find the classification rate of the network and the genetic program on only these images.

The results, shown in table 7.9, show that both the neural network and the genetic program is able to reach a classification rate of more than 80% when only the good images are used for classification.

It is seen that the genetic program is better than the neural network, but as the tesing on this dataset has not been very thorough not conclusions on which is better will be made.

7.7 Influence of features

One thing is to have a functioning classification algorithm another is to have good inputs for the algorithm. In order to improve the feature extraction in the future one of the best neural network has been used to find the most important features in the current classification.

7.7.1 The values of the features

Initially the output values of the feature extraction has been analyzed. In figure 7.16 a boxplot of the values of the features found by the feature extraction is shown.

From the figure it is seen that some values does not vary much while others do. The neural networks are easilier trained when the values use the entire range

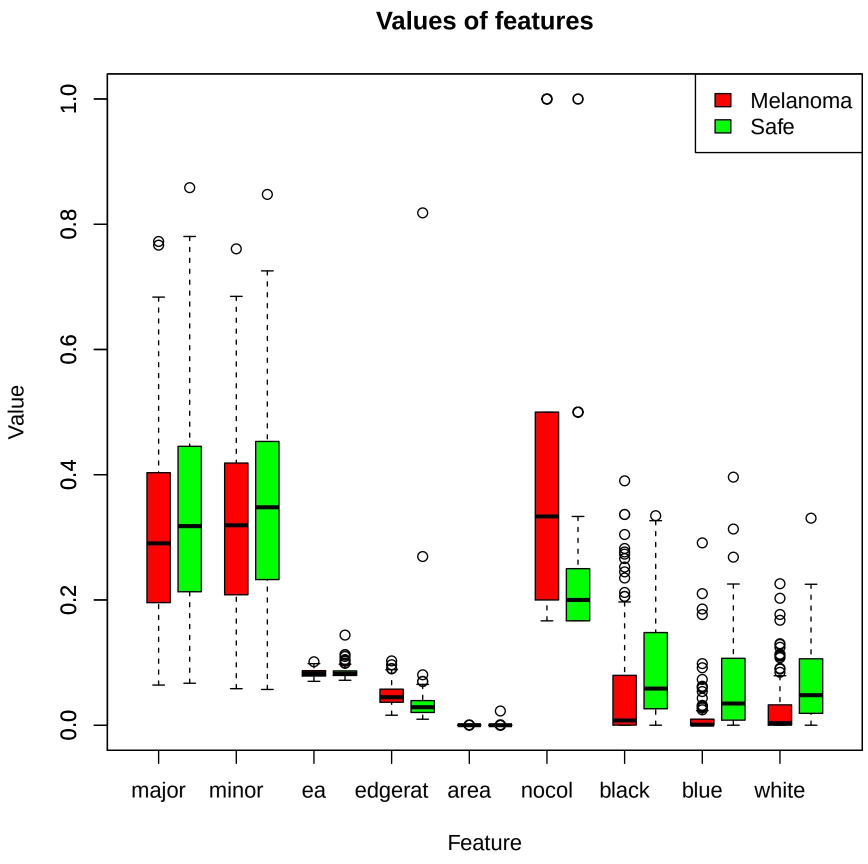


Figure 7.16: Feature values in harmfull/harmless birthmarks. The boxplots show the ranges in which the values are located depending on their diagnosis. The whiskers can at most have the length of 1.5 times the length of the box.

from 0 to 1 so the feature extraction should eventually be improved such that edge abruptness, edge-area ratio and area size take better use of the interval.

Most imprtantly however the graph shows that none of the values can be used to unambiguously determine if the birthmark is harmful or not. All features do overlap and can at most give a hint to the final result. This result was expected as a non-overlapping feature outcome would result in the intelligent algorithm being irrelevant.

7.7.2 Influence of individual features

To see the individual effect of each of features a test has been run where one feature has not been included. In figure 7.17 the results of the test is shown.

The barplot shows that the area does not contribute to the classification at all and the asymmetry in the minor axis, the edge abruptness and the blue color ratio has only a limited effect. The asymmetry in the major axis and the number of colors as well as the black color ratio has quite an impact on the classification rate, but the most important feature is the edge-area ratio. When this feature is removed the classification rate drops to 50% (the same classification rate as random classification).

It is also worth noting that when the white color ratio is removed, the classification rate is improved a little. Why the classification rate is improved is hard to tell as the backpropagation should have neglected the white input note if it found it to be confusing. The color information in general is probably not that useful as the colors have been picked manually and the images are not color corrected in any way.

From the graph it is seen that all features, except a few which is by-product of other features, do contribute to the classification rate. If the color extraction is improved the classification rate might improve as only litle of the current color information is used to classify the birthmarks.

7.7.2.1 Limited features test

A test where the classification algorithm is trained using only the good features or at least without the white color information could be run. This however has not been done and will therefore be a thing to do in a future development of the software.

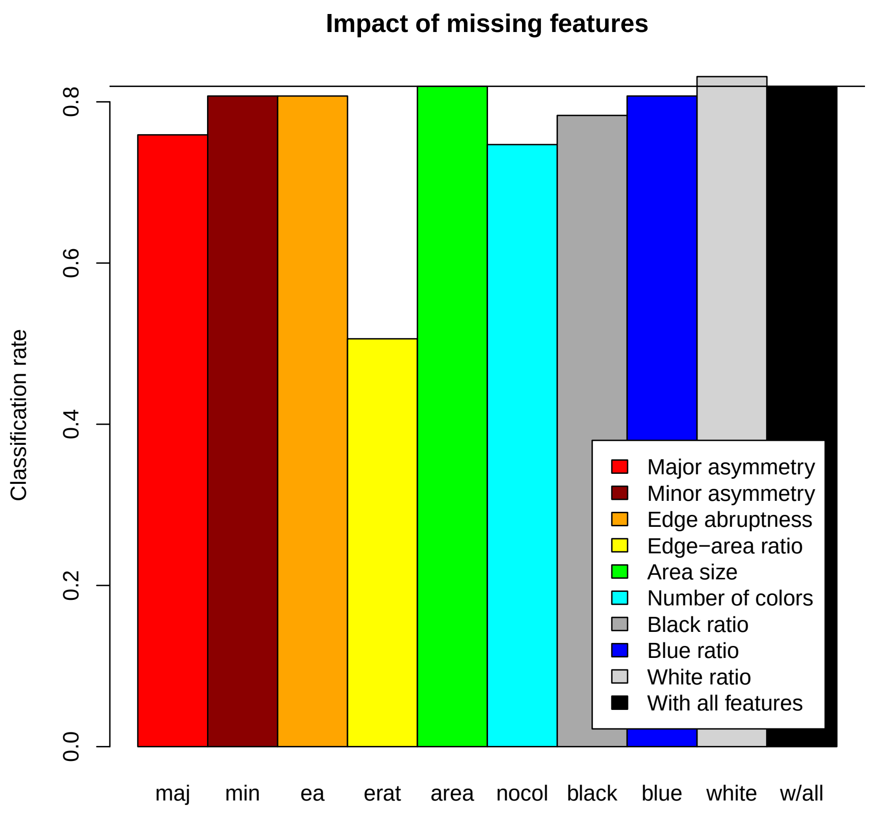


Figure 7.17: Impact of missing features. The plot shows the classification rate of the neural network when a single feature is missing. The black line shows the classification rate of the network with all features present. And each bar shows the classification rate of the network when the feature shown in the legend is missing.

Chapter 8

Conclusion

In this work I have succesfully created a piece of software which given an image of a birthmark is capable of extracting 9 features from the birthmark and classifying the birthmark based on the extracted features. The features, inspired by the *ABCDE* criteria, describe the asymmetry of the birthmark, the size and abruptness of the edge and the colors in the birthmark.

The software has been designed with modularity in mind such that both classification algorithms use the same genetic algorithm to evolve and such that a new classification algorithm can easily be included. The software has furtermore been designed to work on a multicore computer and has succesfully been used in Microsoft Azure, the cloud service developed by Microsoft.

The individual parts of the software has been tested and the results show that the feature extraction algorithm is working on most images. The current feature extraction is able to find the birthmark correctly in 64% of the images in the dataset and in only 23% of the images the algorithm is completely wrong.

Both the implementations of the neural networks as well as the genetic programing are functioning and their capabilities in form of classification rate is at the same level as classification algorithms developed by others when tested on a benchmark dataset.

The best classification algorithm is the neural network evolved from a starting population of perceptrons and has a classification rate of 77.98% when all images, including those where the feature extraction was not able to find the birthmark, are used.

When only good images are used tests show that a higher classification rate can be reached and an initial test has resulted in a classification rate of 82.46%. This classification rate is reached with a genetic program trained on the good images.

The high classification rates are based on classification algorithms that produces as many false negatives as false positives, and tests show that moving the threshold for when a birthmark is harmfull in order to lower the false negative rate, will result in a much lower classification rate.

The resolution of the images does have a significant influence on the results of the feature extraction and it can be concluded that a resolution if $400px \times 400px$ on an image cropped to fit the birthmark is to be preffered. The effect of lowering the resolution was however that more images was classified as positive why the false negative rate actually dropped. Therefore a low resolution image would more like be classified melanoma and would more likely be checked more thoroughly by the doctor.

Finally tests show that the most important feature is the edge-to-area ratio which, when removed, makes the classification rate drop to approximately 50%. A classification rate of 50% is as bad as pure guessing. The same test also shows that the color information only has very little contribution to the classification as they only move the classification rate a few percentage points.

8.1 Future work

Despite the initial very good results the system can be improved and especially the feature extraction has some potential for improvement. In a future work the thresholding algorithm should be improved such that the birthmark would be found in a bigger part of the images.

The color information in the current feature extraction algorithm has a minor influence on the classification rate. This could be a sign that either the color information is not relevant or that the extraction of color information could be improved. In the dermatoscopic world the colors are very important and can often be used solely to determine the classification of a birthmark. It is therefore most likely the case that the feature extraction has room for improvement why this should be in focus in a future work as well.

In a future work the classification algorithm should be trained to punish the presence of false negatives harder than it does at present. Currently only minor tests with an evaluation function punishing the presence of false negatives has been tried out with little success.

For the project to be able to be used in the industry a framework automating the process of uploading of the image to the feature extraction, sending the features to the classification algorithm and returning a classification to the user should be developed.

Finally when all these improvements have been implemented the feature extraction should be trained and tested against a larger population with more images and a greater share of harmles birthmarks. The current training data contains only birthmarks which the house doctor has classified as harmful or maybe harmful. It therefore does not give us much information on how the classification algorithm will behave on clearly harmless birhtmarks.

The prototype has shown promising results and with just a little more effort a great software helping in the melanoma diagnosis process, can come out of this. The potential of the software is huge as only minor modification will make it work on other types of visual diagnosis such as wound healing.

Appendix A

Glossary

Backpropagation Backpropagation is the algorithm used to update the weights. It uses the error gradient on the output nodes to update the weights in the network. It is more thoroughly described in section 3.1.2.

Bias (node) The bias node is a node which added to all hidden and output nodes in the neural network. Its output values is -1 and is used to moved the threshold function.

Boxplot A boxplot is a plot used to show the values of a number of samples. It consists of a box inside which 50% of the values are found. Inside the box a line showing the median is found and outside the box two whiskers showing the highest/lowest data within the 1.5 IQR (inter quartile range) of the lower/higher quartile ($1.5 \cdot$ the size of the box). Any data outside the whiskers are called outliers and is visualized with a small circle.

Crossover Crossover or reproduction is the way the genetic algorithm evolves. Crossover is performed by splitting two individuals (called the aprents) into two pieces each and combining these pieces such that two new indivduals (called the offspring) are found.

Epoch When training the neural network backpropagation is used to update the weights. The backpropagation runs for a number of iterations and each of these iterations is called an epoch. It is said that the backpropagation runs for a number of epochs.

False positives/negatives The rate of true positives(TP), true negatives(TN), false positives(FP) and false negatives(FN) are numbers indicating how many samples have been correctly classified and how many have not. In this project a positive sample is a melanoma. A sample said to be TP is therefore a sample which using the classification software is correctly classified as melanoma. A FN is a sample with melanoma which is classified as harmless.

Function A function is a part of the genetic program which take a number of inputs, in this project 2, and does a calculation based on these inputs. In this project the addition, subtraction and multiplication functions are used.

Generation A genetic algorithm is running for a number of iterations. Each of these iterations are called a generation. Sometimes the population in a given generation is reffered to as the generation.

Individual An individual of a genetic algorithm is one solution to the problem. The population contains a number of individuals and the reproduction is done between individuals.

Multi layer network The multi layer network is a neural network consisting of layers which are fully connected with one another. Each layer can have a different amount of nodes and many layers can exist.

Mutation Mutation is used by the genetic algorithm to make i minor change in an individual. It could be a single bit being flipped or the like. In this project a mutation is performed by either removing or adding a node in the neural network.

Neuron The neuron is the node in the neural network. It does the calculation of summing all the ouputs found in the input synnapses, passing this sum through the activiation function and enabling this output value for the outgoing synnapses.

Perceptron The perceptron is a neural network without any hidden nodes. All input nodes are connected to all output nodes.

Population A population is the individuals found in a given generation. The population size differs from implementation, but in this project a population size of 50 is used.

Synnapsis The synnapses are the edges between the nodes in network. Each synnapsis has a weight which is set by the backpropagation algorithm.

Terminal A terminal is part of the genetic program which does not take any inputs. In this project two terminals are used: a constant value and an input from the problem.

True positives/negatives see *false positives/negatives.*

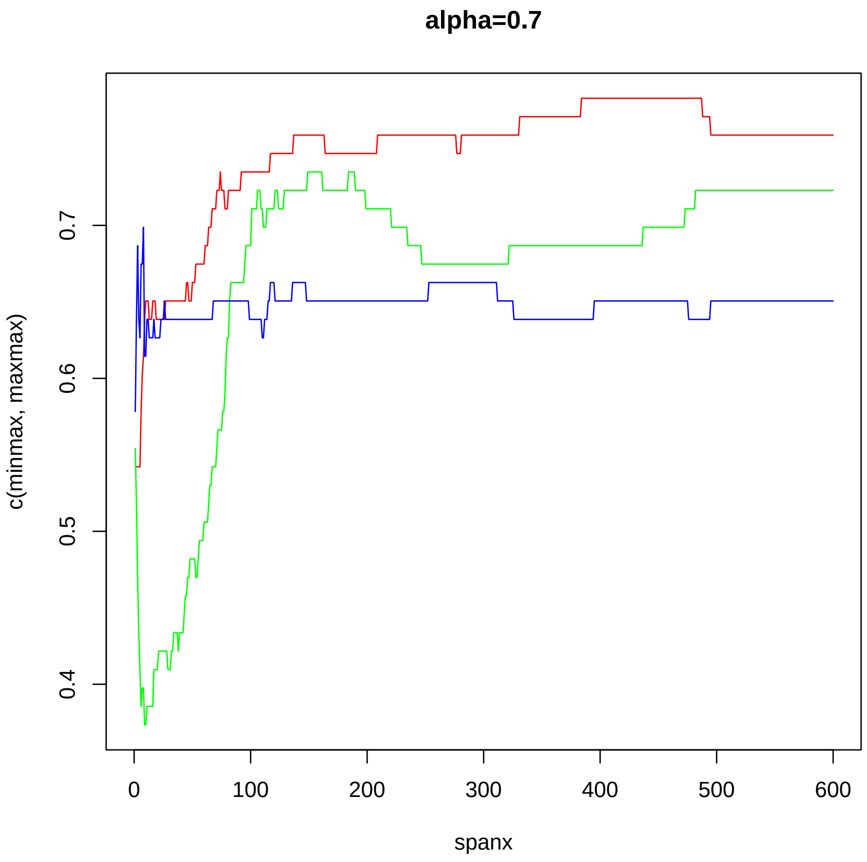
Unstructured network The unstructured network is a network containing a number of hidden nodes which are randomly connected to any other node in the network. The network is ordered such that any edge can only have a direction from a node with a low number to a node with a higher number to prevent cycles from existing in the network.

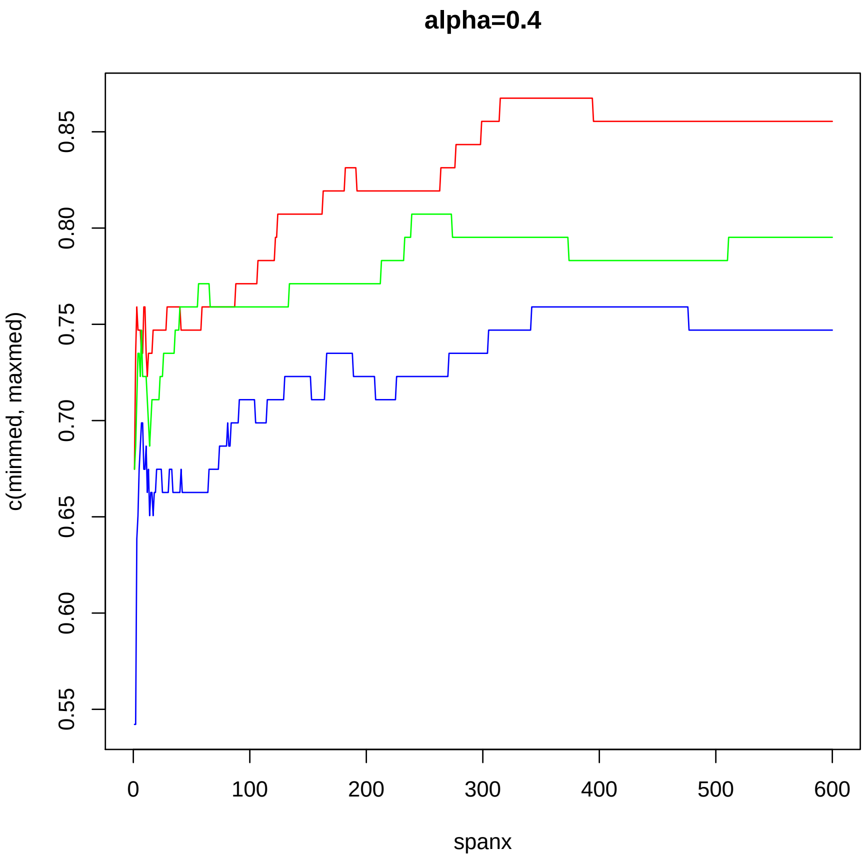
APPENDIX B

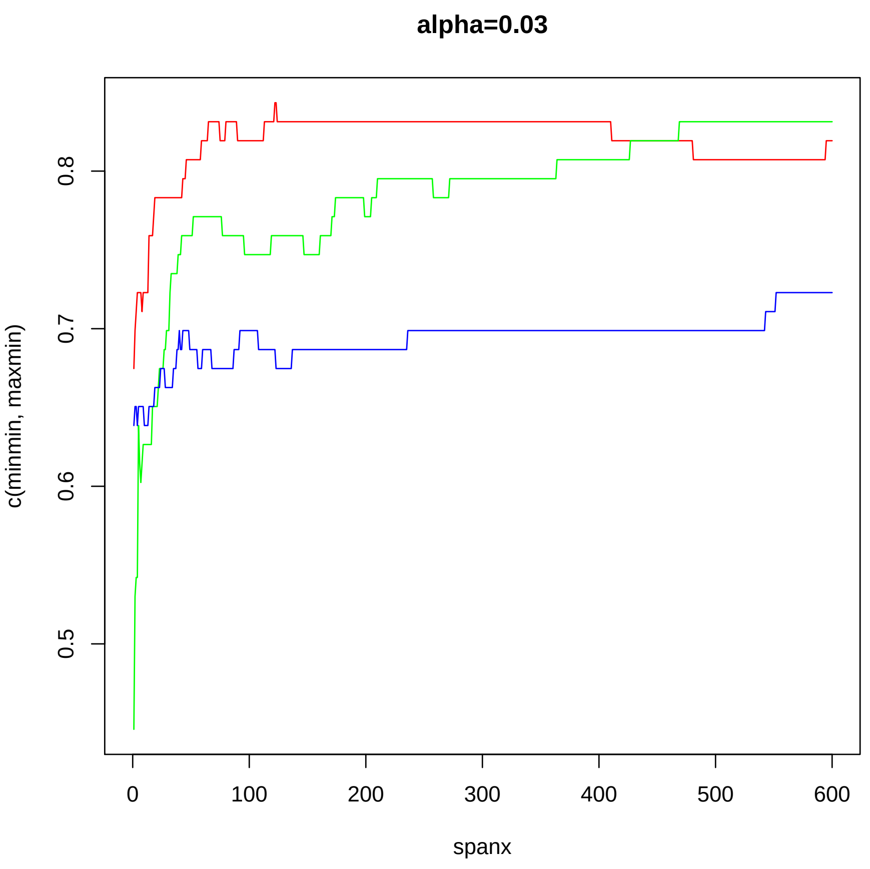
Classification rates

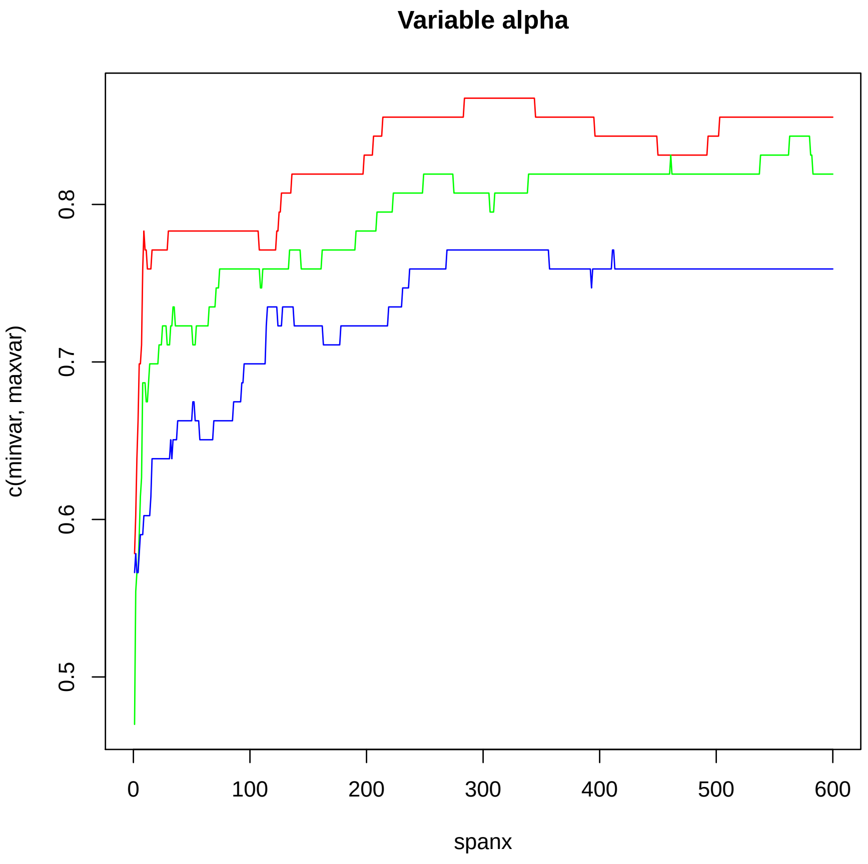
In the following four graphs are shown. They show the classification rate as a function of the number of epochs it has run. The learning rate, alpha, is new in each graph. In the first the learning rate i 0.7, in the second it is 0.4, in the third it is 0.03 and in the fourth graph the learning rate drops linearly from 0.7 to 0.03.

The graphs are used in section 7.4.1.


alpha=0.7
c(minmax, maxmax)
0.4
0.5
0.6
0.7
0
100
200
300
400
500
600
spanx


alpha=0.4
c(minmed, maxmed)
0.55
0.60
0.65
0.70
0.75
0.80
0.85
spanx
0
100
200
300
400
500
600


alpha=0.03
c(minmin, maxmin)
0.5
0.6
0.7
0.8
0
100
200
300
400
500
600
spanx



Variable alpha
c(minvar, maxvar)
0.5
0.6
0.7
0.8
0
100
200
300
400
500
600
spanx

Bibliography

[al.01] Charles M. Balch et. al. *Final Version of the American Joint Commitee on Cancer Stafing System for Cutaneous Melanoma*. Tech. rep. 2001.

[al.04] Naheed T. Abbasi et. al. *Early Diagnosis of Cutaneous Melanoma*. Tech. rep. 2004.

[Bay98] Stephen D. Bay. *Combining Nearest Neighbor Classifiers Through Multiple Feature Subsets*. Tech. rep. University of California, California, USA, 1998.

[BB98] Markus Brameier and Wolfgang Banzhaf. *A Comparison of Genetic Programming and Neurla Networks in Medical Data Analysis*. Tech. rep. Universität Dortmund, Germany, 1998.

[Bis97] Christopher M Bishop. *Neural Networks for Pattern Recognition*. 1st. Oxford University Press, 1997. ISBN: 0198538642.

[Cal] University of California. *Types of Skin Cancer*. Accessed: 2011-05-12. URL: `http://www.dermatology.ucsf.edu/skincancer/professionals/types.aspx`.

[Chr05] Chrislb. *Diagram of an artificial neuron*. Accessed: 2011-04-25. July 2005. URL: `http://en.labs.wikimedia.org/wiki/File:ArtificialNeuronModel_english.png`.

[Col01] Jeananda Col. *The Brain, Brain Cells*. Accessed: 2011-04-13. 2001. URL: `http://www.enchantedlearning.com/subjects/anatomy/brain/Neuron.shtml`.

[Dor00] Robert E. Dorsey. *Reliable classification using neural networks: a genetic algorithm and backpropagation comparison*. Tech. rep. May 2000.

[Fre97] Kimmo Fredriksson. *Genetic algorithms and generative encoding of neural networks for some benchmark classification problems*. Tech. rep. Helsinki, Finland, Aug. 1997.

[Grö98] Marko Grönroos. *Evolutionary Design of Neural Networks*. Tech. rep. University of Turku, Finland, 1998.

[HM98] Mads Hintz-Madsen. “A probabilistic framework for classification of dermatoscopic images”. PhD thesis. IMM - DTU Informatics, 1998.

[JM97] Chuanyi Ji and Sheng Ma. *Combinations of weak classifiers*. Tech. rep. Rensselaer Polytechnic Institute, New York, USA, Jan. 1997.

[Jol02] I. T. Jolliffe. *Principal Component Analysis*. 2nd. Springer, 2002. ISBN: 9780387954424.

[KB94] Iebeling Kaastra and Milton Boyd. *Designing a neural network for forecasting financial and economic time series*. Tech. rep. Aug. 1994.

[Koz93] John R. Koza. *Genetic Programming*. 3rd edt. A Bradford Book, The MIT Press, 1993. ISBN: 0262111705.

[PJ04] Rune Saaby Damgaard Pedersen and Thomas Brigsted Jensen. *Indlæringsstrategi og indkodning i et evolutionært neuralt netværksmiljø*. 2004.

[Pre94] Lutz Prechelt. *PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules*. Tech. rep. Fakulatät für Informatik - Universität Karlsruhe, Sept. 1994.

[RN03] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 2nd international edition. Prentice Hall, 2003. ISBN: 9788129700414. URL: http://aima.cs.berkeley.edu.

[Sch00] Wolfram Schiffmann. *Encoding feedforward networks for topology optimization by simulated evolution*. Tech. rep. University of Hagen (Germany), 2000.

[Siv08] S. N. Sivanandam. *Introduction to Genetic Algorithms*. Springer, 2008. ISBN: 9783540731894.

[SS01] Randall S. Sexton and Naheel A. Sikander. *Data mining using a genetic algorithm-trained neural network*. Tech. rep. Dec. 2001.