International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

© Global Society of Scientific Research and Researchers

http://ijcjournal.org/

Binary Image Segmentation Using Classification Methods: Support Vector Machines, Artificial Neural Networks and Kth Nearest Neighbours

Saman Sarraf*

Department of Electrical and Computer Engineering, McMaster University, Hamilton, Ontario, Canada, The Institute of Electrical and Electronics Engineers, IEEE

Email: samansarraf@ieee.org

Abstract

The principal objective of this work is to demonstrate efficient parameter selection for various networks used in binary image segmentation. The Support Vector Machines using four kernel functions (i.e., Radial Basis Function, Quadratic, Polynomial, and Linear), Neural Networks (i.e., Feed-forward Back-propagation) and Kth Nearest Neighbours algorithm were applied to five different datasets that had been generated from a given image. Pixel coordinates (x,y) were considered as inputs. Grid search and cross-validation were performed to identify the optimal network parameters. All experiments were repeated five times in order to develop confidence in the obtained results. High accuracy was achieved in most cases 95% for SVM-RBF, 90.4% for SVM-Quadratic, 90.8% for SVM-Polynomial, 60% for SVM-Linear, 88% for Neural Networks and 97% for K-NN. After grid search for SVM-RBF, the accuracy reached 98%. In this project, SVM-RBF showed a high level of accuracy and consistency. It was also found that the selected features (pixel coordinates) were discriminative.

Keywords: Image Segmentation; Binary Classification; SVM; ANN; K-NN; Decision Making.

1. Introduction

“One of the most interesting aspects of the world is that it can be considered to be made of patterns. A pattern is essentially an arrangement,” said Norbert Wiener, Professor of Mathematics at MIT, “It is characterized by the order of the elements of which it is made, rather than by the intrinsic nature of these elements” [1]. Pattern recognition and classification by machine involve techniques that are based mostly on mathematical algorithms which automatically assign patterns to their respective classes.

----

* Corresponding author.

This means that a designed pattern recognition system acts to reduce human intervention as much as possible [1]. Pattern recognition systems are comprised of several elements, including input, sensing, segmentation, feature extraction, classification, post-processing and decision [2]. It may be necessary to make adjustments for missing features and context in the classification and post-processing steps. Furthermore, system designers sometimes penalize the last part of the pipeline when the final decision is made [2]. In a pattern recognition system designed for image processing, the input is an image and the output is often an image too [2]. In image processing, one of the most important steps in the analysis of processed image data is image segmentation. The goal of image segmentation is to divide an image into parts that have a strong correlation with the objects being represented in the image (i.e., Binary Segmented Image). Binary Image Segmentation is the process of dividing an image into two classes (e.g., two colors, or black and white) [3]. As mentioned above, one of the most important steps in training an intelligent network is to select efficient parameters for it. This step, which is time-consuming, consists of two major procedures grid search and cross-validation. Intelligent networks such as Support Vector Machines and Neural Networks have complex structures, and their efficiency depends on the network parameters. Thus, the best values for the network parameters must be extracted by grid search and cross-validation. In this study, binary image segmentation was carried out by means of various classification methods Support Vector Machines, Neural Networks and Kth Nearest Neighbours algorithms. This study focused primarily on efficient parameter selection for these classification methods, then compared the results of these methods. For SVM, four kernel functions were selected Radial Basis Function, Quadratic, Polynomial (with order of 3), and Linear functions. Neural Networks and K-NN were also applied to datasets. To train and test the networks, five different datasets were used with 100, 500, 1500 and 2000 datapoints. The pixel coordinates (x,y) were considered as the inputs of networks. The average accuracy obtained for SVM, ANN and K-NN were approximately 95%, 88% and 97%, respectively. The SVM-RBF showed a high level of accuracy and consistency for this classification problem. The pixel coordinates that were used as the features in this image classification were recognized as highly discriminative and separable features. Finally, grid search and cross-validation highly increased the level of accuracy and confidence in the results.

2. Background

Firstly, binary image segmentation was performed using Support Vector Machines. The support vector machine (or ‘support-vector network’) is a supervised learning machine for two-group classification problems. This algorithm theoretically implements the following idea: input vectors are non-linearly mapped to a very high-dimensional feature space, in which a linear decision surface is constructed. A special property of the decision surface guarantees high generalizability of the learning machine. The idea behind the SVM was previously implemented for the restricted case where the training data can be separated without errors [4]. Vapnik and his colleagues [4] invented the original SVM algorithm as well as the current standard incarnation (soft margin). In this algorithm, there are L training points, where each input xi has D dimensions and belongs to one of two classes yi = - 1 or +1. It is assumed that the data is linearly separable so a line can be drawn on a graph of x1 vs. x2 that separates the two classes when D = 2 and forms a hyperplane on graphs x1, x2, …XD when D > 2. This hyperplane is described by $\mathbf{w.x+b=0}$, where w is normal to the hyperplane and $\frac{\mathbf{b}}{\|\mathbf{w}\|}$ is the perpendicular distance from the hyperplane to the origin. Support Vectors are the examples closest to the separating hyperplane and the aim of Support Vector Machines is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes [5]. This algorithm will be discussed in more detail in Chapter 3.

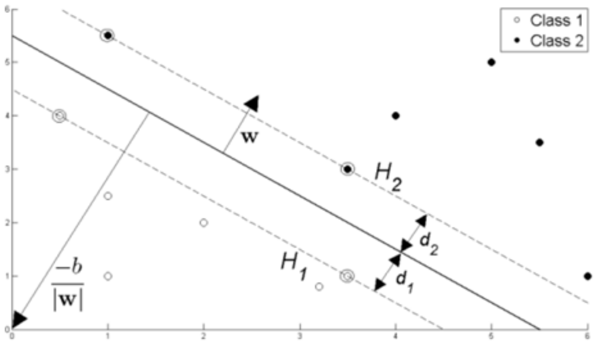

Figure 1: Hyperplane through two linearly separable classes [5]

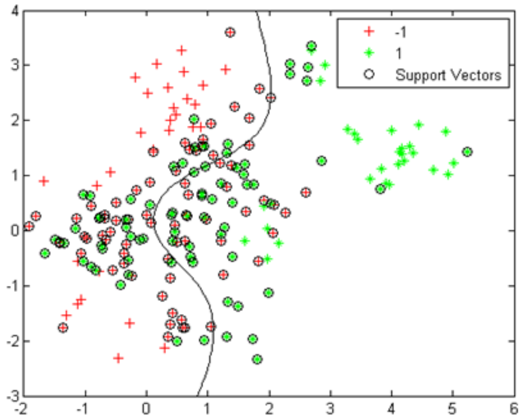

Figure 2: Binary classification of random data [6]

Binary image segmentation using Multilayer Neural Networks was implemented in the second section of this course project. Neural Networks, sometimes called ‘Artificial Neural Networks’, are a mathematical algorithm inspired by biological neural networks (i.e., in the human brain), which consist of simple artificial neurons connected by directed weighted connections. When the system is set running, the activation levels of the input units are clamped to certain desired values. Afterwards, the activation is propagated along the directed weighted connections to other units at each time step (iteration). The activations of non-input neurons are computed using each neuron's activation function [7,8]. A neural network has three main classes of units: input, hidden, and output units. An activation pattern is presented on its input units and spreads forward from the input units through one or more layers of the hidden units to the output units. The activation coming into a unit from other units is multiplied by the weights on the links over which it spreads. All incoming activation is then added together, and the unit becomes activated only if the incoming result exceeds the unit’s threshold. In summary, the basic elements of a neural network are the units, the connections between units, the weights, and the thresholds [7,8,9]. This classification method will be discussed in more detail in Chapter 3 [10,11].

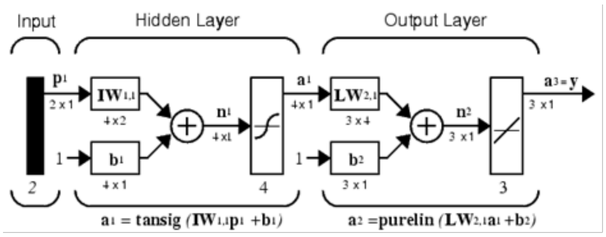

Figure 3: Schematic Model of Multilayer Neural Network Architecture

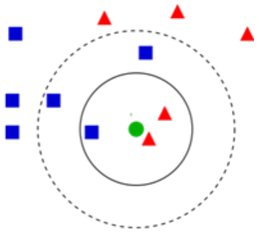

Figure 4: Example of K-NN classification

The Kth Nearest Neighbours algorithm was applied to image data as the third approach. K-NN is a method for classifying objects based on the closest training examples in the feature space. K-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. K-NN is the simplest of all machine learning algorithms [12]. In Figure 4, the test sample (green circle) should be classified either to the blue squares or the red triangles. If k = 3 (solid circle), it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed circle), it is assigned to the first class (3 squares vs. 2 triangles in the outer circle) [13,14].

3. Theory and Approach

The design of the following algorithm was based on a literature review [1,2,7,15] and was successfully performed in MATLAB R2010b, Version 7.11.0.584, 64-bit. The CPU used in this project was Intel(R) Core(TM) i7-2640M CPU @ 2.80GHZ, and the installed memory (RAM) was 16.0 GB.

3.1. Pre-processing

The first step of the algorithm was to apply some pre-processing tasks to the given image. First, the image was converted from an RGB to a gray-scaled image, and then it was converted into a binary image based on the following equation [14,16].

$$if \begin{cases} i_{m,n} > 128 \rightarrow i_{m,n} = 255 \\ i_{m,n} \leq 128 \rightarrow i_{m,n} = \quad 0 \end{cases} (1)$$

In Equation 1the binary image segmentation, equation of decision, where i is image intensity of pixel (m,n) is described. Due to certain software and hardware restrictions, and in order to prevent “out of memory” errors, the default image (Fig. 6) was slightly cropped and rescaled from 2100×2738 to 658×800 (Fig. 7). It is worth mentioning that these final image dimensions were achieved by dimension reduction through a process of trial and error.

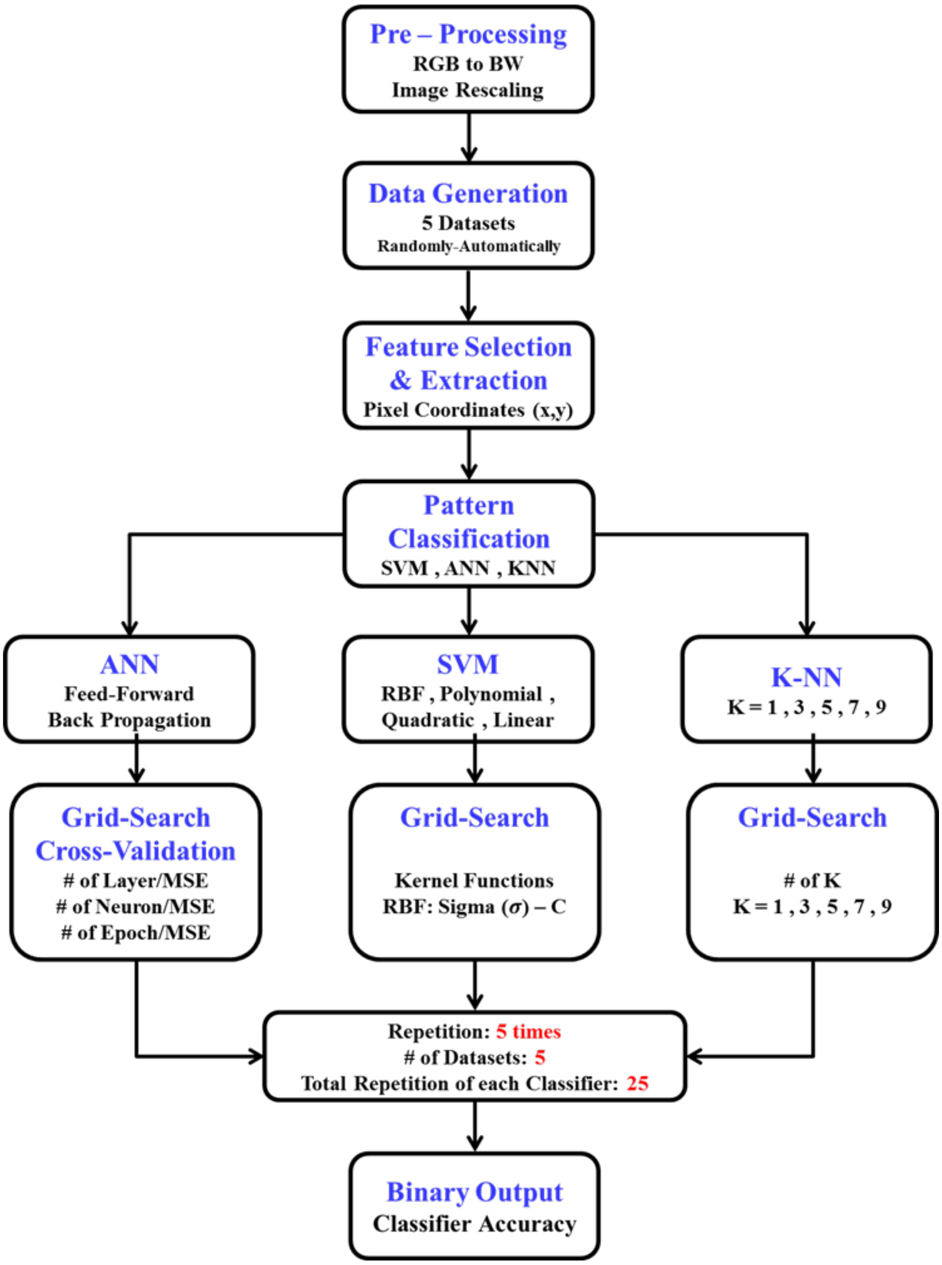

Figure 5: Performed algorithm applied to 5 datasets, repeated 5 times

3.2. Data generation

Five different datasets were randomly and automatically generated based on binary image pixel coordinates. These datasets had 100, 500, 1000, 1500 and 2000 datapoints. In each dataset, the coordinates of pixels were generated, and the corresponding label (image intensity of 0 or 255) for each pixel was extracted. In other words, each dataset consists of two vectors of coordinates (x,y) and one vector of corresponding labels. The five datasets helped to fully study the classifiers' performance and achieve reliable accuracy for each method (after taking an average).

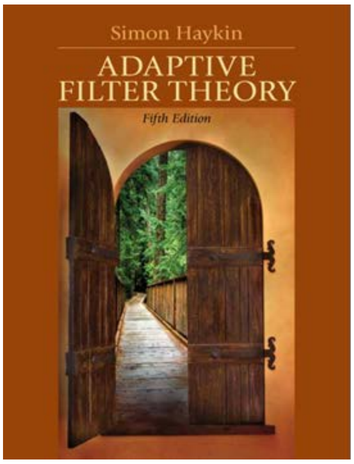

Figure 6: Given image (2100x2738)

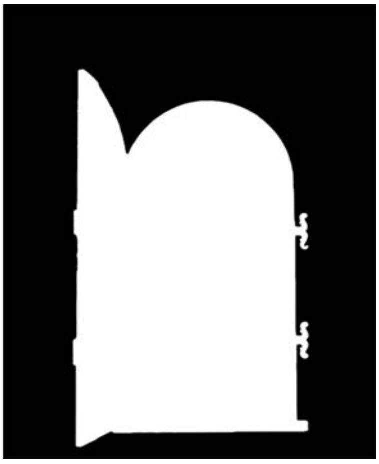

Figure 7: Binary image (658x800)

Exactly two-thirds of each dataset were considered as the training dataset, and the remaining third was used to test the networks.

3.3. Feature Selection and Extraction

In this course project, the coordinates of each pixel in the binary image were selected as the features. Therefore, each dataset contained two columns and many rows. As mentioned above, the image was rescaled. One of the reasons for rescaling was that the MATLAB function for training SVM calculates the ratio of features and datapoints and then tries to generate the hyperplanes based on that ratio. If this ratio exceeds the amount of memory, an “out of memory” error occurs. Since we hoped to use two features in this project, we needed to decrease the feature space by decreasing the datapoints to a reasonable number.

3.4. Pattern Classification

Three different networks were used for the classification. The parameter selection for these networks was performed using grid search, cross-validation, and experiment repetition when needed. In order to fully understand the difference and uniqueness of each classifier, the results were compared with each other. Support Vector Machines, Neural Networks and Kth Nearest Neighbours were applied to the five datasets.

3.4.1. Support Vector Machines

As discussed in Chapter 2, in SVM, input vectors are non-linearly mapped to a very high-dimensional feature space. In this feature space, a linear decision surface is made. Two important parts of the SVM algorithm are the Kernel Function and the Optimization Method. Equation 2 describes the optimization problem in SVM.

$$w(\alpha) = \sum_{i=1}^{l} \alpha_i - (\frac{1}{2}) \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

Where $\mathrm{w}(\alpha)$ is explained by $\|\mathrm{w}\|^2 = \mathrm{w.w}$, where w is the normal vector to the hyperplane. The α_i are the Lagrange multipliers, and $\mathrm{K}(\mathrm{x_i}, \mathrm{x_j})$ is the kernel function [17]. The above equation determines the Lagrange multipliers and the classifier that implements the optimal separating hyperplane in the feature space. In this study, the following four kernel functions were utilized:

$$Linear: K(x_i, x_j) = (x_i . x_j)$$

$$Quadratic: K(x_i, x_j) = (x_i . x_j + 1)^2$$

$$Polynomial: K(x_i, x_j) = (x_i . x_j + 1)^d$$

$$Radial\ Basis\ Function(rbf): K(x_i, x_j) = e^{(-\gamma \|x_i - x_j\|^2)}\ , \gamma > 0\ , \gamma = {}^{1}\!/_{\sigma^2}$$

Other methods such as Sequential Minimal Optimization or Least Squares can be used as the optimization method, but the Quadratic Programming (QP) algorithm was selected. According to literature review [7], QP returns better results than other methods. The goal of the QP algorithm is to minimize Equation 4:

$$f(x) = \frac{1}{2}\, x^T Q x + c^T x, x \in R^n$$

3.4.2. Finding Optimal Parameter Values

In the case of RBF kernel function, the optimal values of σ and C were selected by grid search. A grid search tries values of each parameter across the specified search range using geometric steps. Grid searches are computationally expensive because the model must be evaluated at many points within the grid for each parameter [18].

3.4.3. Neural Networks

As the second classification approach, a feed-forward back-propagation network (MLP) was implemented, where the transfer function was Tangent Sigmoid and the output layer was a linear function. A grid search and cross-validation were performed in order to find the optimal number of layers and neurons for the designed network. The performance of the network was measured by Mean Squared Error (MSE) over epochs. In each training step, the best validation performance of the network was calculated and monitored by plotting the network performance (MSE/epochs).

3.4.4. K^{th} Nearest Neighbours (K-NN)

The objective of K-NN is to classify an object by a majority vote of its neighbours. K, representing the class of Nearest Neighbours, is a positive integer and is typically small. In binary classification (like this project), K will be an odd number selected by experience or through cross-validation [20]. In this study, the optimal value to maximize network accuracy was achieved through cross-validation. K = 1, 3, 5, 7, 9 were examined to obtain the best K value.

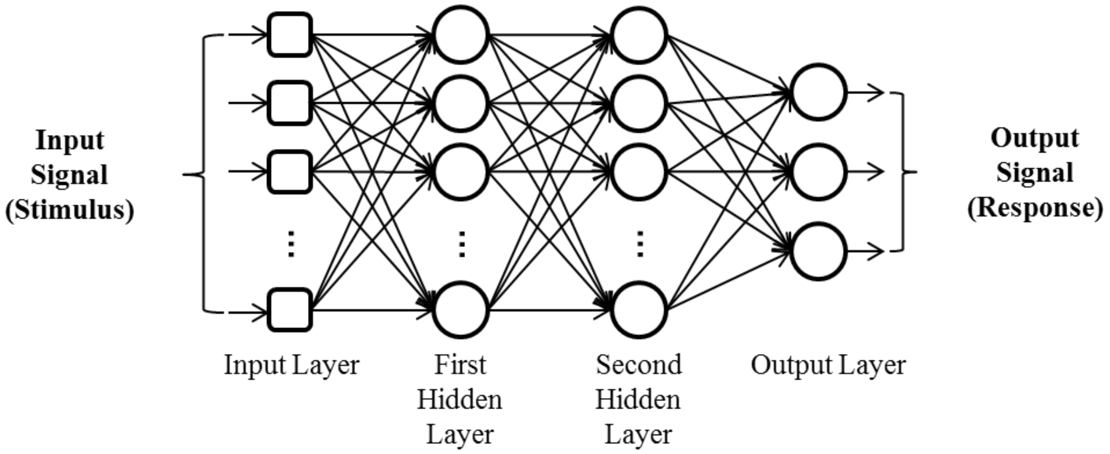

Figure 8: Architectural graph of multilayer perceptron with two hidden layers – Concept from Neural Networks, A Comprehensive Foundation, Simon Haykin [7,19]

3.4.5. Output of Networks

As indicated above (Fig. 5), each classification method was applied to each of the five datasets and repeated five times in order to obtain reliable results. Thus, each classifier was repeated 25 times in total. After training the networks on two-thirds of the datapoints, they were then tested on the remaining third. The output of all methods was a binary matrix that included labels corresponding to pixel coordinates in the test data. The accuracy of each network was calculated by Equation 4:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (4)$$

4. Discussion of Experiments and Results

The algorithm implementation in MATLAB that was designed for this project will be discussed step by step in this chapter. The results acquired through extensive experiments will be studied and compared in order to fully understand how parameter selection impacts the classification methods.

4.1. Dataset Generation

The pixel coordinates of the binary image (Fig. 7) were considered as the source for dataset generation. Next, five datasets were extracted with different numbers of datapoints 100, 500, 1000, 1500 and 2000. Studying several datasets made it possible to assess the impact of dataset size on the classification algorithm and parameter selection. As mentioned earlier, each dataset contained two features [pixel coordinates (x,y)] and the corresponding label (e.g., image intensity). The number of datapoints in each dataset was considered based on experience in order to prevent “overfitting”, “underfitting”, and “out of memory” errors in MATLAB. Generally, the number of datapoints should be 10 times that of network parameters and features, according to the rule of thumb.

4.2. Support Vector Machines Classification

SVM was applied to datasets as the first approach, and various kernel functions were examined. In order to find the optimal network parameters, the grid search was performed if needed. The Quadratic Programming algorithm was selected as the optimization method.

4.2.1. Radial Basis Function (RBF)

The table below shows the accuracy of SVM-RBF for (C=1 and σ=1). It should be mentioned that the MATLAB function for SVM training normalized input data before training.

Table 1: Accuracy of SVM-RBF, 5 datasets with 5 repetitions of the experiment

	Radial Basis Function – Accuracy (out of 1)					
	Datapoints	**100**	**500**	**1000**	**1500**	**2000**
# of experiments	**1**	0.91176471	0.89820359	0.94311377	0.942	0.95652174
	2	0.88235294	0.92814371	0.9491018	0.944	0.92503748
	3	0.94117647	0.92814371	0.93712575	0.948	***0.95952024***
	4	0.94117647	0.94610778	0.95808383	0.948	0.95352324
	5	0.94117647	0.94011976	0.95808383	0.954	0.95202399
	Average	**0.92352941**	**0.92814371**	**0.9491018**	**0.9472**	***0.94932534***

Table 1 indicates that SVM-RBF classified data very well and the highest accuracy was about 96% for 2000 datapoints. The highest average accuracy belonged to 2000 datapoints at 94.93%. This table also shows that, although SVM-RBF accuracy has low sensitivity to the number of datapoints, it increases slightly over the number of datapoints. As is known, the radial basis function is a Gaussian-based function. This high accuracy achieved by SVM-RBF confirms that dataset distributions follow a normal (Gaussian) distribution. Figure 9 illustrates the SVM-RBF boundary decision and support vectors for 1500 datapoints on the left and 2000 datapoints on the right, where the red plus **"+"**, green asterix **"*"** and black circle "o" represent class 1, class 2 and support vector, respectively. The SVM-RBF showed the highest rate of accuracy and high consistency in this classification. Figure 10 shows the average accuracy of five experiments for each dataset. The error bar for each dataset was calculated by the standard deviation of the five experiments' accuracy in each dataset. This figure also demonstrates that the highest accuracy occurs with 2000 datapoints.

4.2.2. Grid Search of SVM-RBF parameters: σ and C

There are two important parameters in the optimization problem of SVM-RBF **σ** and **C**, respectively. In order to find the optimal values for these parameters, a grid search with 12 values of **σ** (from 10^{-5} to 10^{+6} by a factor of 10) and 12 values of **C** (from 10^{-5} to 10^{+6} by a factor of 10) was performed (total iteration: 12*12=144). As mentioned in the previous step, the most reliable results belonged to 1,500 and 2,000 datapoints, so the grid search was performed only for these two datasets.

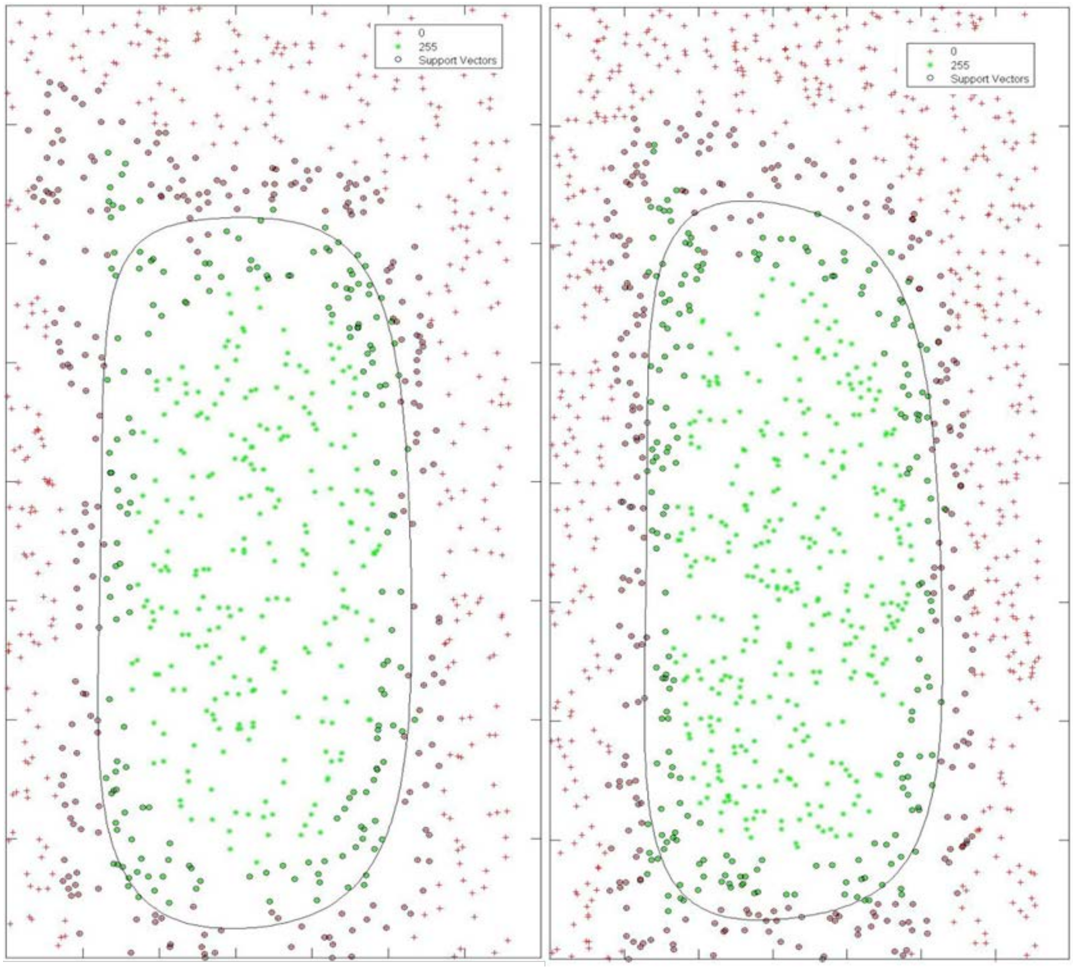

Figure 9: SVM-RBF boundary decision and support vectors – Left: 1,500 datapoints, Right: 2,000 datapoints. Red plus = class 1, green asterix = class 2, black circle = support vector

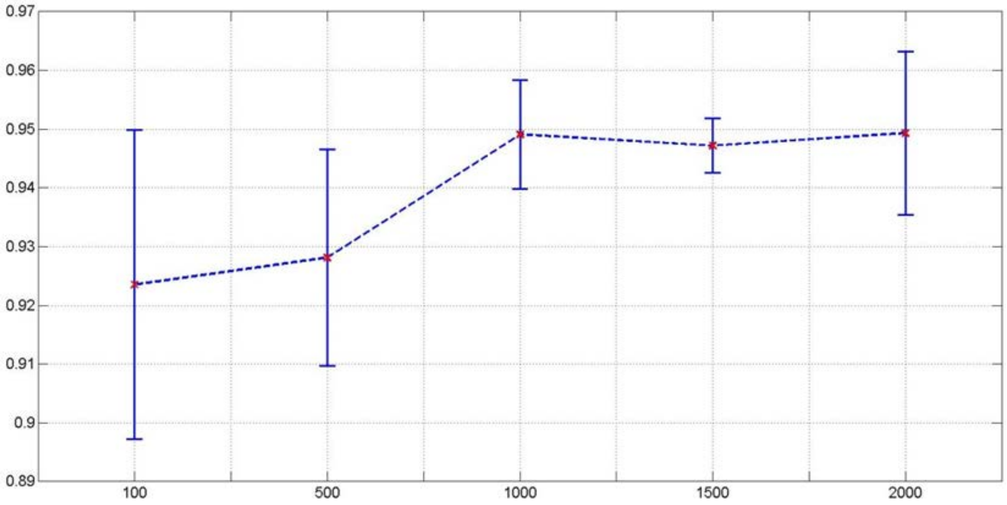

Figure 10: SVM-RBF average accuracy over datasets. Error bars: Standard deviation of 5 experiments for each dataset

Table 2: SVM-RBF, Grid Search, and accuracy over parameters σ and C, 2000 datapoints

	SVM-RBF, Accuracy, 2000 Datapoints											
	Parameter "C" from 10^{-5} to 10^{+6}											
Parameter sigma "σ" from 10^{-5} to 10^{+6}	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652
	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652
	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652	0.5652
	0.6132	0.6132	0.6132	0.6132	0.6132	0.6132	0.6117	0.6117	0.6117	0.6117	0.6117	0.6117
	0.973	0.973	0.973	0.973	0.9715	0.964	0.967	0.9685	0.967	0.967	0.967	0.967
	0.8756	0.8771	0.8891	0.919	0.928	0.949	0.9715	0.9685	0.976	0.9805	***0.9805***	0.9775
	0.6417	0.6417	0.6417	0.6402	0.6417	0.6837	0.8666	0.91	0.91	0.91	0.916	0.9205
	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6372	0.6372	0.6402	0.6822	0.8666	0.91
	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6372	0.6372	0.6372	0.6372
	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6372	0.6372
	0.6387	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402
	0.5652	0.5982	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402	0.6402

Table 3: SVM-RBF, Grid Search, and accuracy over parameters σ and C, 1500 datapoints

	SVM-RBF, Accuracy, 1500 Datapoints											
	Parameter "C" from 10^{-5} to 10^{+6}											
Parameter sigma "σ" from 10^{-5} to 10^{+6}	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562
	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562
	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562	0.562
	0.604	0.604	0.604	0.604	0.604	0.604	0.604	0.604	0.604	0.604	0.604	0.604
	0.972	0.972	0.972	0.972	0.978	0.972	0.968	0.966	0.966	0.966	0.966	0.966
	0.882	0.882	0.882	0.904	0.914	0.932	0.944	0.956	0.962	0.97	0.97	0.97
	0.592	0.592	0.592	0.594	0.606	0.646	0.846	0.916	0.902	0.906	0.904	0.914
	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.594	0.644	0.848	0.916
	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588
	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588
	0.584	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588
	0.562	0.548	0.584	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588	0.588

Table 2 indicates that the best sigma value is $\sigma = 1$ and the best for C is 10^{+5}. So the SVM-RBF was repeated for 2000 datapoints with new parameters: $\sigma = 1 , C = 10^{+5}$. In this case, the accuracy increased to ***0.9805***. This shows that performing the grid search and finding the optimal values for network parameters increased accuracy by **3.32%**. Table 3 shows that the optimal sigma and C value for the 1500 datapoints are 0.1. In case of $\sigma = 0.1$, $C = 0.1$, the SVM-RBF was applied to 1500 datapoints, and the new accuracy reached 0.978. In this case, the accuracy improved by almost 5%.

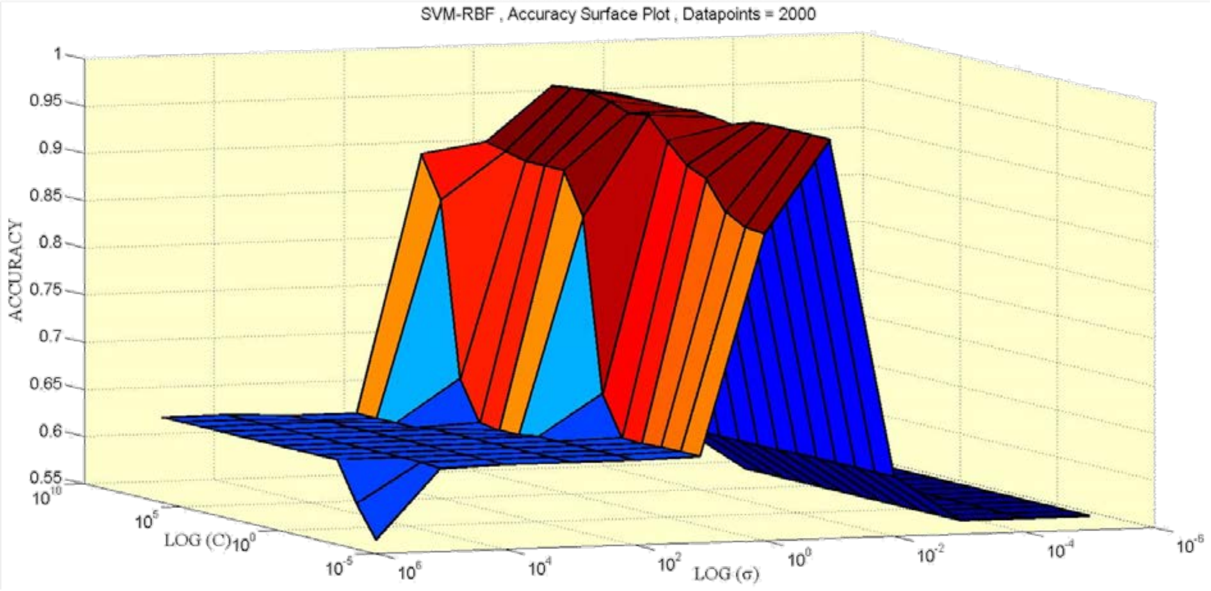

Figure 11: SVM-RBF, Grid Search, Accuracy Surface Plot, and 2000 datapoints

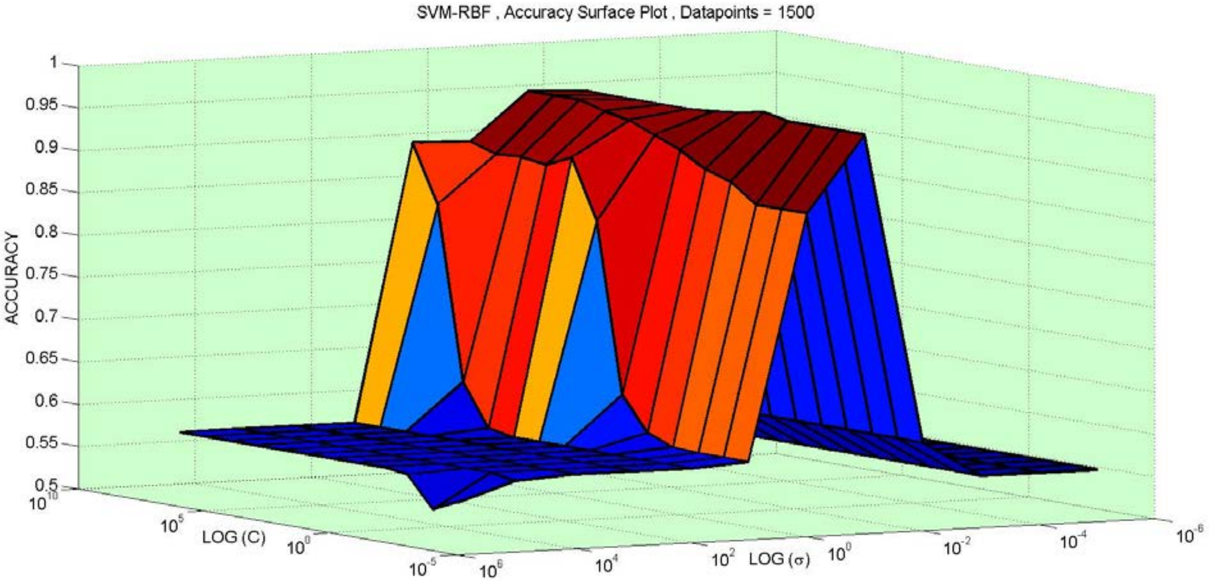

Figure 12: SVM-RBF, Grid Search, Accuracy Surface Plot, and 1500 datapoints

Figures 12 and 13 illustrate the accuracy surface plot of SVM-RBF over Log (σ) and Log (C) in the performed grid search. In these figures, dark blue indicates lower accuracy and dark brown indicates higher accuracy of SVM-RBF. These graphs give a clearer sense of accuracy changes over parameters σ and C.

4.2.3. Quadratic Kernel

Table 4 explains the accuracy obtained by SVM-Quadratic for five different datasets. This step was also repeated five times, and the last row of the table indicates each dataset’s average accuracy.

Table 4: Accuracy of SVM-Quadratic, 5 datasets, 5 repetitions of experiment

Quadratic Kernel						
	Datapoints	100	500	1000	1500	2000
# of experiments	1	0.88235294	0.86826347	0.91317365	0.904	0.91304348
	2	0.94117647	0.90419162	0.90718563	0.916	0.88005997
	3	0.97058824	0.92215569	0.90718563	0.908	0.90854573
	4	0.94117647	0.90419162	0.90419162	0.912	***0.91154423***
	5	0.94117647	0.91616766	0.94011976	0.922	0.90854573
	Average	**0.93529412**	**0.90299401**	**0.91437126**	**0.9124**	***0.90434783***

As shown in Table 4, Quadratic kernel showed low sensitivity to the number of datapoints. No regular pattern of accuracy emerged regarding the number of data. Although the highest accuracy belongs to 100 datapoints, the result of 2000 datapoints is still more reliable as the SVM-Quadratic had been trained and tested by sufficient data. The average accuracy of SVM-Quadratic was reported at ***90.4%***, which confirms that the classification was well performed. Figure 13 shows the average accuracy of five experiments for each dataset. The error bar for each dataset was calculated by the standard deviation of five experiment accuracies in each dataset. Quadratic Programming was used as the optimization method.

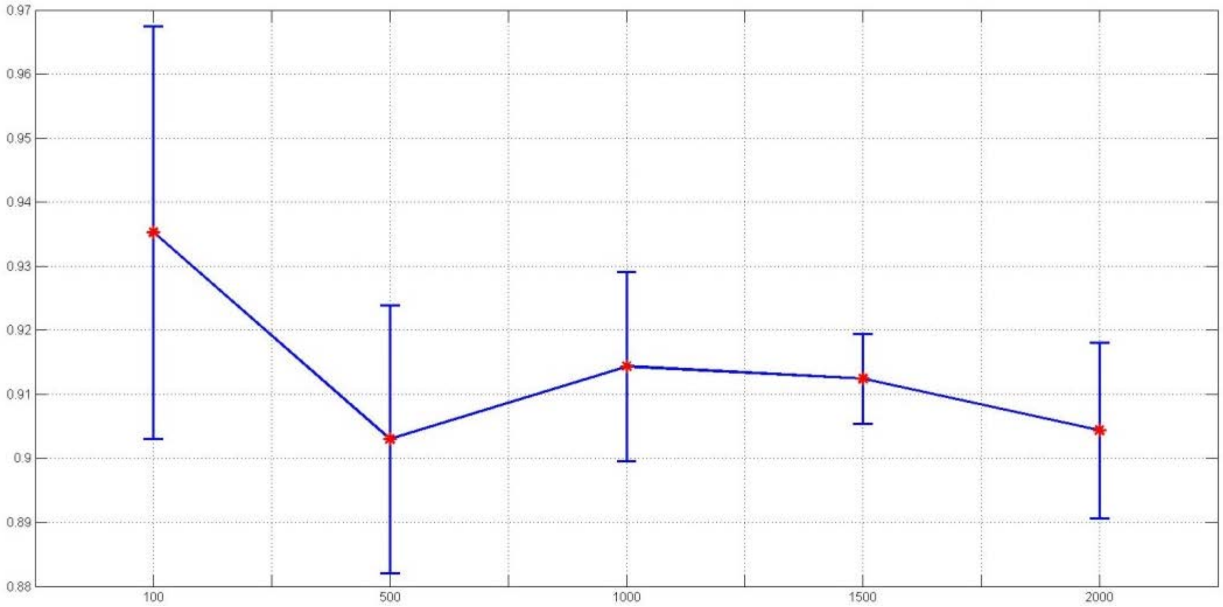

Figure 13: SVM-Quad average accuracy over datasets. Error bars: Standard Deviation of 5 experiments for each dataset

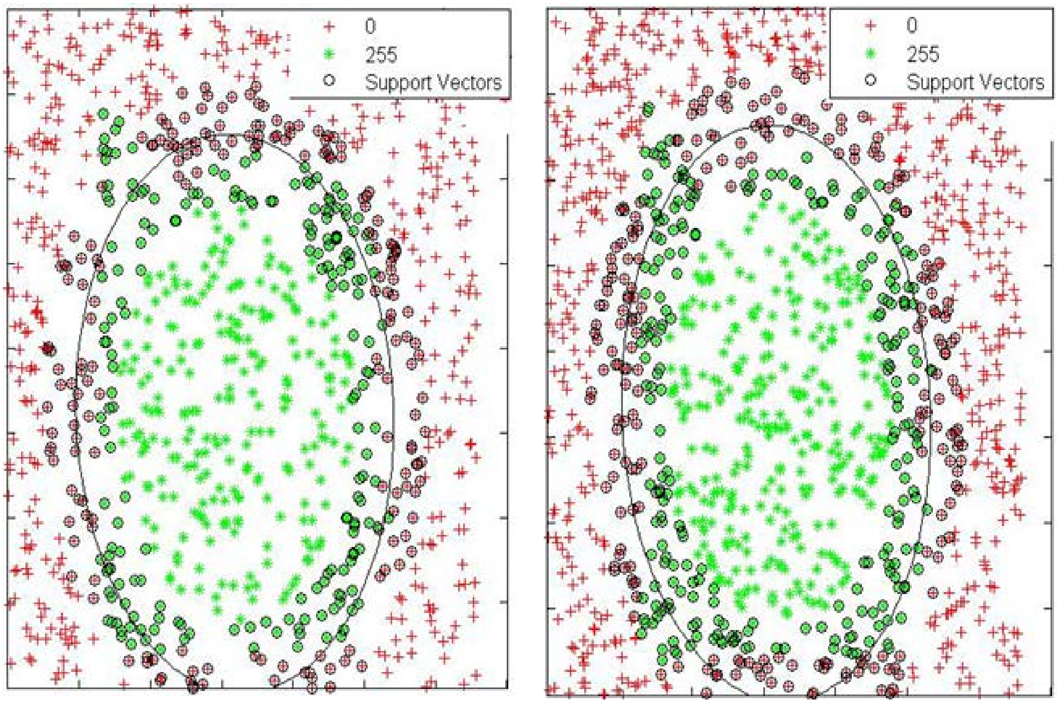

Figure 14: SVM-Quadratic, boundary decision and support vectors, Left: 1500 datapoints, Right: 2000 datapoints

4.2.4. Polynomial kernel with order of 3

In this step, SVM classification was implemented using Polynomial kernel function with an order of three. Table 5 shows the accuracy of this classification.

Table 5: Accuracy of SVM-Polynomial (order of 3), 5 datasets, 5 repetitions of experiment

Polynomial (order of 3) Kernel						
	Datapoints	100	500	1000	1500	2000
# of experiments	1	0.94117647	0.88622754	0.91916168	0.904	0.91754123
	2	0.91176471	0.91017964	0.90718563	0.916	0.88605697
	3	0.94117647	0.92215569	0.92814371	0.896	***0.91754123***
	4	0.94117647	0.91017964	0.90718563	0.922	0.91304348
	5	0.94117647	0.91017964	0.94311377	0.916	0.91004498
	Average	**0.93529412**	**0.90778443**	**0.92095808**	**0.9108**	***0.90884558***

SVM-Quadratic and SVM-Polynomial both show a decrease in accuracy as the number of datapoints increases. This can be explained by the fact that larger datasets are more complicated, and because a small decrease in accuracy is found in Quadratic and Polynomial SVM classification; however, approximately ***91%*** accuracy was obtained in this step, which was the average of five experiments for 2000 datapoints.

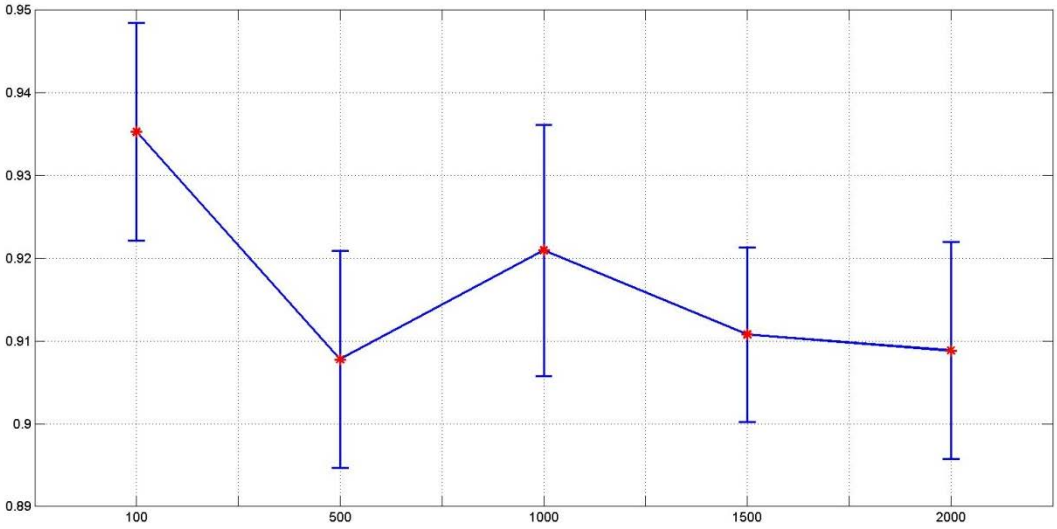

Figure 15: SVM-Poly, average accuracy over datasets. Error bars: Standard Deviation of 5 experiments for each dataset

Figure 15 shows the average accuracy of five experiments for each dataset. Quadratic Programming was used as the optimization method.

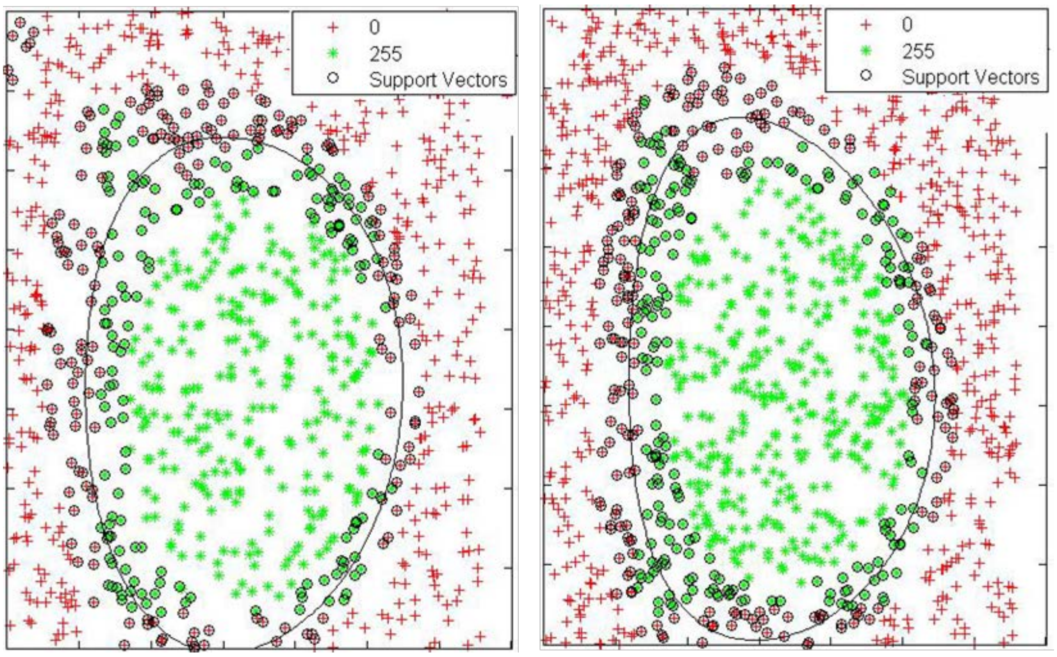

Figure 16: SVM-Polynomial, boundary decision and support vectors, Left: 1500 datapoints, Right: 2000 datapoints

4.2.5. Linear Kernel

The simplest kernel function used in SVM classification was the linear kernel function, which had the lowest accuracy of the kernel functions in this classification. Table 6 shows the accuracy of SVM-Linear classification for different datasets.

Table 6: Accuracy of SVM-Linear, 5 datasets, 5 repetitions experiment

Linear Kernel						
	Datapoints	100	500	1000	1500	2000
# of experiments	1	0.47058824	0.53293413	0.56287425	0.578	0.59070465
	2	0.52941176	0.54491018	0.61077844	0.586	0.57721139
	3	0.44117647	0.56886228	0.59580838	0.586	0.59370315
	4	0.44117647	0.62275449	0.58083832	0.566	0.56671664
	5	0.44117647	0.56287425	0.64371257	0.566	***0.62968516***
	Average	**0.46470588**	**0.56646707**	**0.5988024**	**0.5764**	***0.5916042***

In this experiment, the accuracy increases as the size of the dataset grows, with the exception of the fourth dataset. The overall accuracy is significantly lower than with other kernel functions, demonstrating that the linear kernel function is not adequate for the datasets in this classification problem. Presumably, much larger training datasets are required for the linear kernel function.

As mentioned above, the data generation was based on normal (Gaussian) distribution. Consequently, it could predict that a linear function would not return a highly accurate classification.

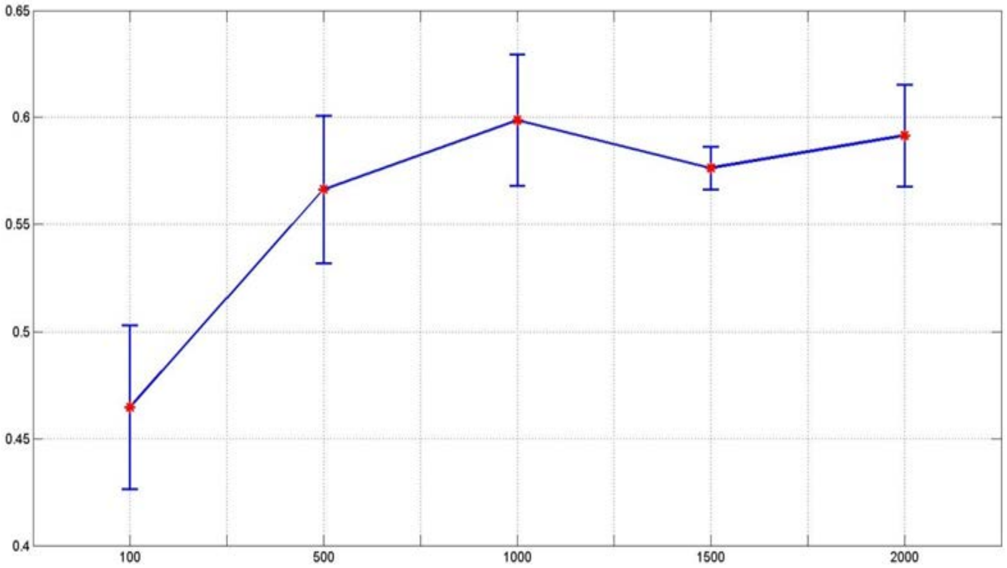

Figure 17: SVM-Linear average accuracy over datasets. Error bars: Standard Deviation of 5 experiments

Figure 17 shows the average accuracy of five experiments for each dataset. The error bar for each dataset was calculated by the standard deviation of five experiment accuracies in each dataset. An overall average accuracy of ***59.1%*** for 2000 datapoints was achieved.

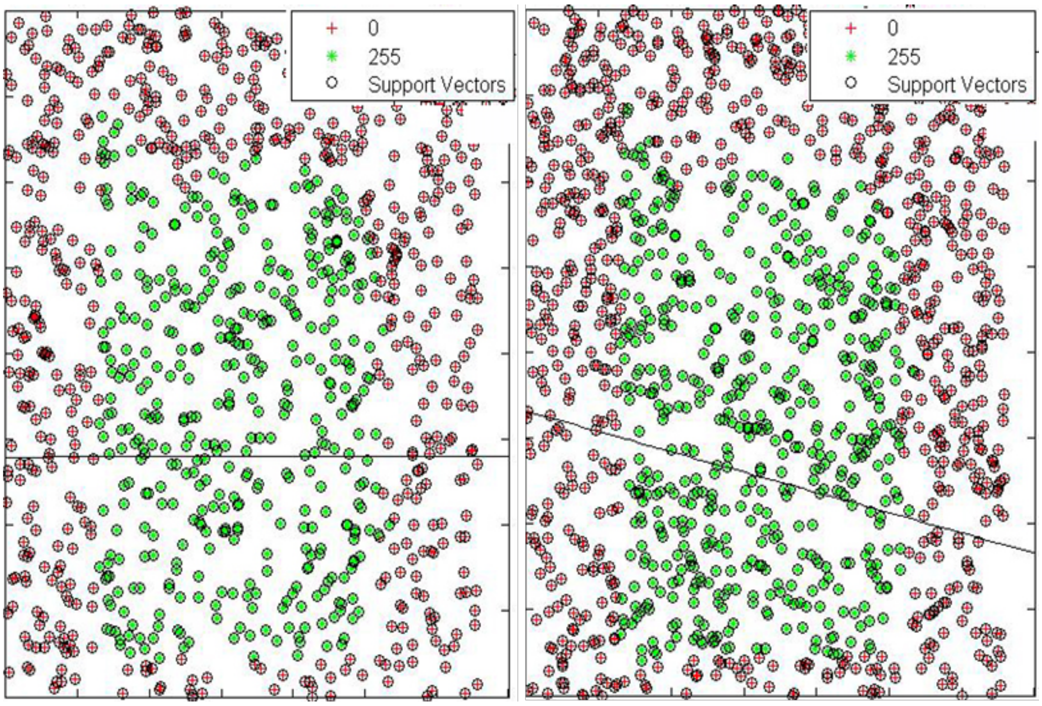

Figure 18: SVM-Linear, boundary decision and support vectors, Left: 1500 datapoints, Right: 2000 datapoints

The linear decision of SVM-Linear (Fig. 18) achieved the lowest accuracy. The data and the binary segmented image were not linearly separable. A linear boundary was unable to segment the data into two classes as the image structure (e.g., edges and corners) formed a non-linear and complicated structure.

4.3. Neural Networks Classification

The Neural Networks algorithm was implemented in MATLAB as the second classification method. Data was normalized into [0, 1] in the first step before network training. In the next step, a grid search was performed in order to find the best number of layers and neurons to minimize the Mean Squared Error (i.e., the network performance function).

Cross-validation was also performed to avoid overfitting the network. This experiment was repeated five times to yield a reliable final result. When performing classification via Neural Networks, there are small fluctuations (or noise) in the output; therefore, these results should be interpreted with caution.

A threshold is often defined to remove the noise from the output and to classify the output into two or more classes. Equation 5 describes the threshold applied to the Neural Networks binary output:

$$if \begin{cases} output_i > 0.5 \ \rightarrow i_i = 1 \\ output_i \leq 0.5 \ \rightarrow i_i = 0 \end{cases}$$

Where $output_i$ is the binary output of Neural Networks, which must be 0 or 1. For instance, if the output is greater than 0.5, it will be classified as class 1.

Table 7: Neural Networks accuracy, 5 datasets, 5 repetitions of experiment

Neural Networks – Accuracy						
	Datapoints	100	500	1000	1500	2000
# of experiments	**1**	0.647059	0.754491	0.973054	0.562	0.923538
	2	0.823529	0.670659	0.622754	0.936	0.802099
	3	0.705882	0.580838	0.643713	0.946	0.91904
	4	0.441176	0.736527	0.919162	0.922	***0.932534***
	5	0.588235	0.916168	0.823353	0.734	0.823088
	Average	**0.641176**	**0.731737**	**0.796407**	**0.82**	***0.88006***

Table 8: Neural Networks MSE, 5 datasets, 5 repetitions of experiment

Neural Networks – MSE						
	Datapoints	100	500	1000	1500	2000
# of experiments	**1**	0.105363	0.052332	0.013017	0.032222	0.025058
	2	0.108706	0.025356	0.038938	0.024164	0.04297
	3	0.088873	0.064776	0.043564	0.017823	***0.007891***
	4	0.094633	0.026603	0.023216	0.027415	0.015953
	5	0.185173	0.036897	0.026876	0.013726	0.030071
	Average	**0.11655**	**0.041193**	**0.029122**	**0.02307**	***0.024389***

As demonstrated above, Tables 7 and 8 show the accuracy and MSE of test data for Neural Networks classification, respectively. Mostly, high accuracy and low MSE were achieved for the Neural Networks method, as expected. In the experiment with 2000 datapoints, the highest accuracy **93.2%** was obtained, with an average accuracy of **88%**.

An increase in accuracy was noticed as the number of datapoints increased in the Neural Networks method, showing a moderate sensitivity to the number of datapoints.

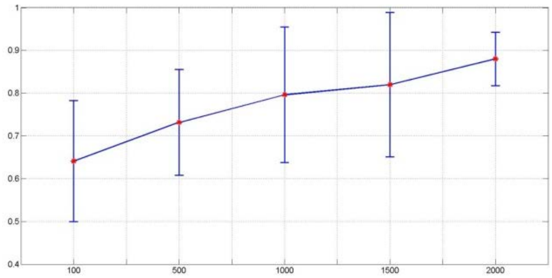

Figure 19: Neural Networks average accuracy over datasets. Error bars: Standard Deviation of 5 experiments

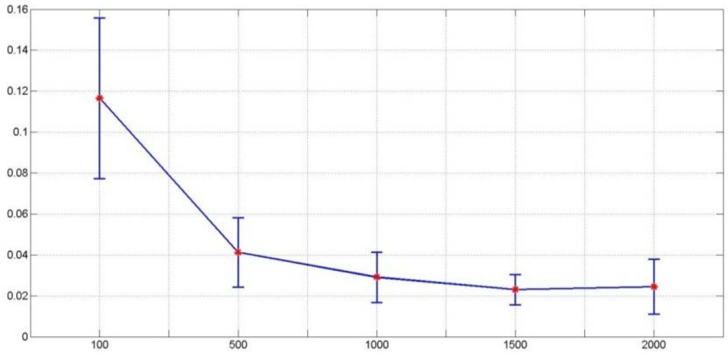

Figure 20: Neural Networks average MSE over datasets. Error bars: Standard Deviation of 5 experiments

Figures 19 and 20 illustrate the accuracy and MSE over datapoints for the Neural Networks approach for 5 datasets and five repetitions of the experiment. As mentioned before, increasing the number of datapoints results in greater accuracy.

4.3.1. Neural Networks Architecture and Grid Search

The optimal number of layers and neurons was obtained through a grid search process. In this step, the number of neurons changed from 3 to 6 and the number of layers changed from 1 to 3 according to the rule of thumb. In each iteration, the MSE was calculated as the network performance function for test data. After finding the optimal numbers, Neural Networks training was repeated using the optimal number of layers and neurons.

Based on the grid search performed, different network structures were achieved. For instance, in the case of 2000 datapoints as well as the first and second repetitions a feed-forward back-propagation network with three layers and six hidden neurons in each layer was designed and implemented as the optimal network. In this network, the activation function was Tangent Sigmoid, and there was a linear function in the output layer.

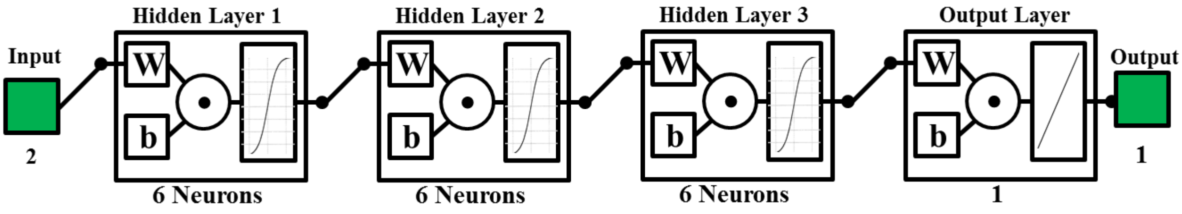

Figure 21: The implemented network structure for 2000 datapoints (1st and 2nd experiments), 3 layers and 6 neurons

4.3.2. Network Validation

In order to avoid overfitting, the best validation performance was obtained in each training step by drawing the MSE over the number of epochs for training, test and validation. Figures 22 and 23 illustrate the network validation for different datasets.

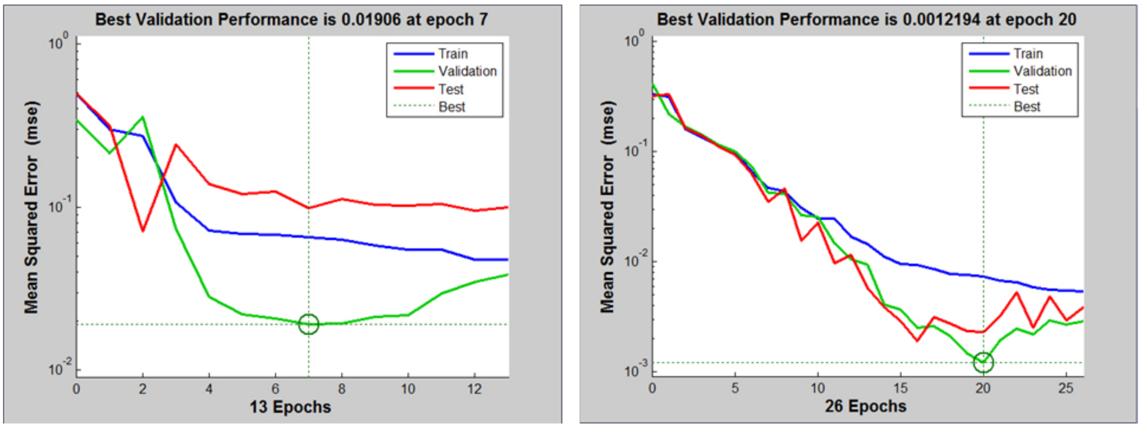

Figure 22: Network Validation Performance (MSE over Epochs), Left: 100 datapoints, Right: 500 datapoints

In these figures, the blue line represents “Training”, the green “Validation”, the red “Test” data and the dashed line represents the best number of epochs.

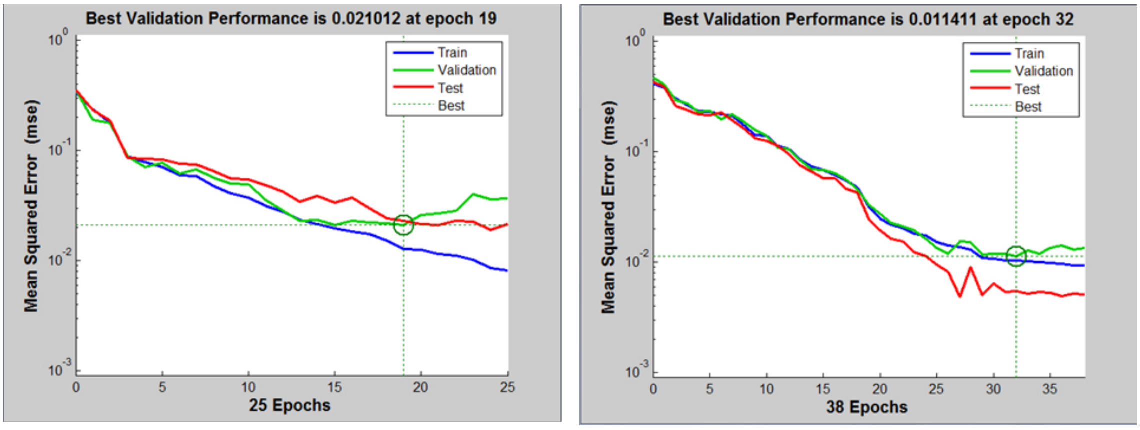

Figure 23: Network Validation Performance (MSE over Epochs), Left: 1500 datapoints, Right: 2000 datapoints

4.4. Kth Nearest Neighbours Classification (K-NN)

The K-NN classification method was implemented in MATLAB as the third and last approach in this study. As mentioned before, this algorithm has the simplest structure. Although the K-NN is the easiest and fastest classifier, it showed high accuracy in this study.

It can justify that data had a normal (Gaussian) distribution. Thus, K-NN could classify data into two classes suitably. Table 9 displays the K-NN accuracy for different datasets. The average accuracy for 2000 datapoints was around ***97%***. Almost the same accuracy values were achieved for 1000 and 1500 datapoints.

As can be seen, K-NN classified with high accuracy, and the accuracy increases as the number of datapoints increases. It should be mentioned that Table 9 shows the accuracy for the optimal value of K obtained after grid search.

Table 9: Neural Networks accuracy, 5 datasets, 5 repetitions of experiment

K-NN Accuracy						
	Datapoints	100	500	1000	1500	2000
# of experiments	**1**	0.882353	0.91018	0.964072	0.962	0.967016
	2	0.911765	0.97006	0.964072	0.976	0.955022
	3	0.852941	0.97006	0.967066	0.97	***0.983508***
	4	0.911765	0.964072	0.97006	0.96	0.971514
	5	0.911765	0.958084	0.97006	0.962	0.968516
	Average	**0.894118**	**0.954491**	**0.967066**	**0.966**	***0.969115***

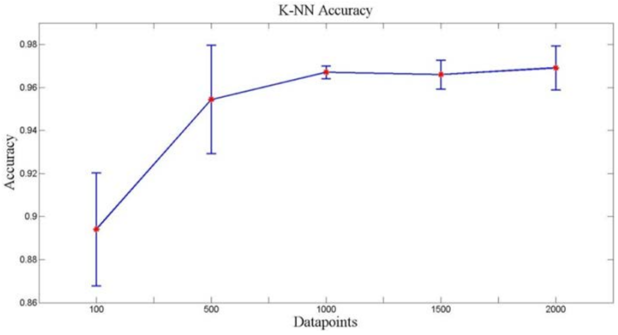

Figure 24: K-NN average accuracy over datasets. Error bars: Standard Deviation of 5 experiments

4.4.1. Grid Search to Find the Optimal K Distance Value

In order to find the best K distance value for minimizing the MSE in K-NN classification, the grid search was performed and the K value took only odd numbers from 1 to 9. In binary classifications such as this project, K should take only an odd value. Table 11 shows the best K value for different datasets and experiments. No specific pattern was found for K values, only for 2000 datapoints, which generally had larger K values than other datasets.

Table 11: K-NN Grid-Search to find the optimal K distance value

K-NN, distance K						
	Datapoints	100	500	1000	1500	2000
# of experiments	**1**	1	9	3	1	5
	2	1	3	3	7	5
	3	1	7	1	1	5
	4	1	5	5	9	9
	5	5	1	1	1	3

4.5. Comparison of Classification Methods

Table 12 and Figure 24 compare the results obtained by the six classification methods used in this project. Generally, all classification methods showed good performance and reliability. The Support Vector Machines with Radial Basis Function demonstrated high levels of accuracy and consistency. Neural Networks also yielded reliable and accurate results. Quadratic and Polynomial (order of 3) kernel functions in SVM had roughly similar outcomes. The lowest accuracy was obtained from SVM using the linear kernel function. Kth Nearest Neighbours showed high accuracy and speed in implementation.

Table 12: Average Accuracy of 6 Classification Methods over Datapoints

			Average Accuracy of Classification Methods				
		Datapoints	**100**	**500**	**1000**	**1500**	**2000**
Classification Methods	**1**	**SVM-RBF**	**0.923529**	**0.928144**	**0.949102**	**0.9472**	**0.949325**
	2	**SVM-Quadratic**	**0.935294**	**0.902994**	**0.914371**	**0.9124**	**0.904348**
	3	**SVM-Polynomial**	**0.935294**	**0.907784**	**0.920958**	**0.9108**	**0.908846**
	4	**SVM-Linear**	**0.464706**	**0.566467**	**0.598802**	**0.5764**	**0.591604**
	5	**ANN**	**0.641176**	**0.731737**	**0.796407**	**0.82**	**0.88006**
	6	**K-NN**	**0.894118**	**0.954491**	**0.967066**	**0.966**	**0.969115**

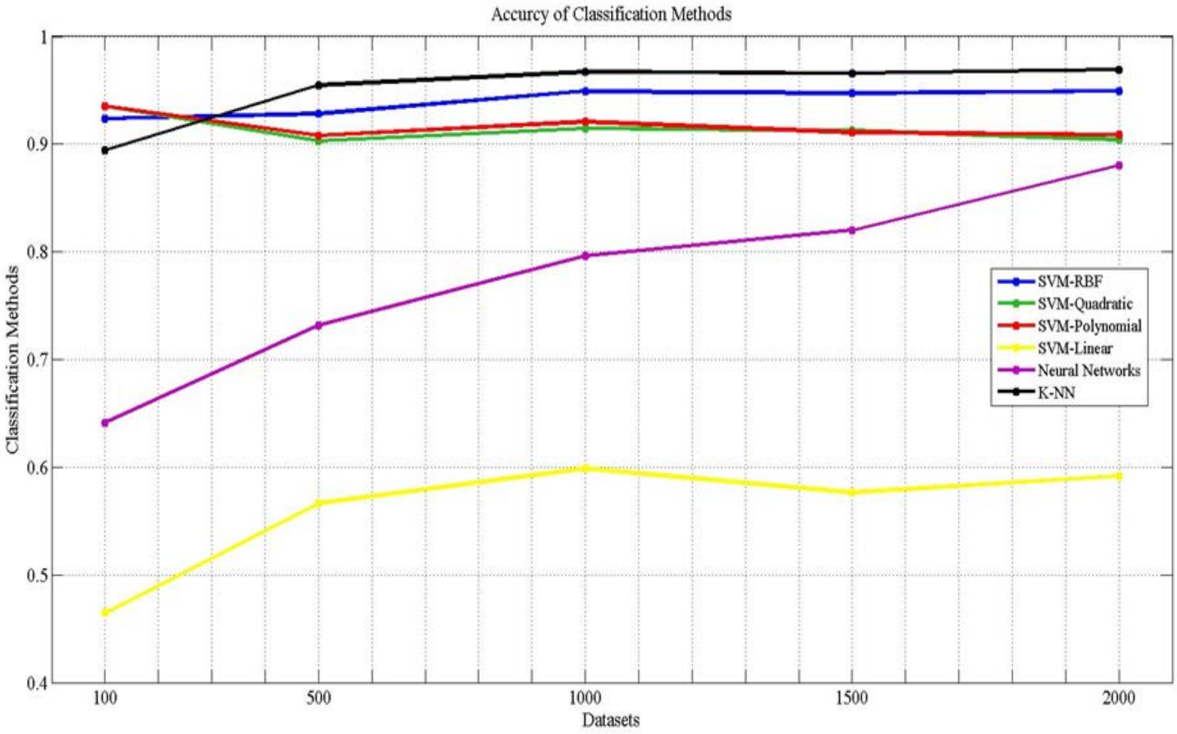

Figure 24: Average Accuracy of 6 Classification Methods over Datapoints

5. Conclusion

Binary image segmentation using six classification methods and efficient parameter selection for each network was implemented in this project. In most cases, high accuracy and consistency of the classification algorithms were achieved. This study found that:

- The proposed features in the feature selection and extraction steps, pixel coordinates (x,y), were discriminative and separable features.
- Support Vector Machines using Radial Basis Function demonstrated full potential to classify complicated data, especially for data having a normal (Gaussian) distribution. Therefore, SVM is highly recommended for classification problems.
- Although Neural Networks performed well in this classification problem, it is more highly recommended for prediction problems and predictive models.
- K^{th} Nearest Neighbours yielded high accuracy in this study; however, based on its structure, it is recommended for simple clustering and classification problems.
- In terms of execution time, K^{th} Nearest Neighbours had the shortest time and fastest execution speed in MATLAB. SVM and Neural Networks were in the second and third positions, respectively.
- The size of datasets impacted accuracy in some cases. The most reliable number of datapoints was 2000, which was yielded by experiment repetition.
- Grid search and Cross-validation improved the accuracy and confidence of the results obtained from all methods.
- In future work, a "pre-classification step" could be added to the algorithm to reduce the data dimensionality. For instance, the support vectors could be used to select a region of interest (ROI) and ignore the data located close to the centre.

References

[1] Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, New Jersey: Pearson Prentice Hall, 2008.

[2] Richard O. Duda, Peter E. Hart, David G. Stork, Pattern Claissification, New York: John Wiley and Sons, Inc., 2001.

[3] Milan Sonaka, Vaclav, Hlavac, Roger Boyle, Image Processing, Analysis, and Machine Vision, Toronto: Thomson, 2008.

[4] Corinna Cortes, Vladimir Vapnik, "Support-Vector Networks," Machine Learnin, vol. 20, no. 1995, pp. 273-297, 1995.

[5] Fletcher, Tristan, "Support Vector Machines Explained," University College London, London, 2009.

[6] MathWorks, "Support Vector Machines (SVM)," The MathWorks, Inc., 1994-2013 . [Online]. Available: http://www.mathworks.com/help/bioinfo/ug/support-vector-machines-svm.html. [Accessed 13 01 2013].

[7] S. S. Haykin, Neural networks and learning machines, New York: Prentice Hall, 2009.

[8] B. Wilson, "The Machine Learning Dictionary," 1998 - 2012. [Online]. Available: http://www.cse.unsw.edu.au/~billw/mldict.html#neuralnet. [Accessed 13 01 2013].

[9] Ferreira, Candida, "Designing Neual Networks Using Gene Expression Programming," Springer-Verlag, pp. 517-536, 2006.

[10] Cheryl Grady , Saman Sarraf, Cristina Saverino, and Karen Campbell, "Age Differences in the Functional Interactions among the Default, Frontoparietal Control and Dorsal Attention Networks," Neurobiology of Aging , 2016.

[11] Saman Sarraf, Cristina Saverino, Halleh Ghaderi, John Anderson, "Brain network extraction from probabilistic ICA using functional Magnetic Resonance Images and advanced template matching techniques," in Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference, Toronto, 2014.

[12] David Bremner, Erik Demaine, Jeff Erickson, Joh Iacono, Stefan Lagerman, Pat Morin, Godfried Toussain, "Output-Sensitive Algorithms for Computing Nearest-Neighbour Decision Boundaries," Springer-Verlag, pp. 451-461, 2003.

[13] Saman Sarraf , Cristina Saverino, Ali Mohammad Golestani, "A Robust and Adaptive Decision-Making Algorithm for Detecting Brain Networks Using Functional MRI within the Spatial and Frequency Domain," in The IEEE International Conference on Biomedical and Health Informatics (BHI) , Las Vegas, 2016.

[14] Saman Sarraf, Jian Sun, "ADVANCES IN FUNCTIONAL BRAIN IMAGING: A COMPREHENSIVE SURVEY FOR ENGINEERS AND PHYSICAL SCIENTISTS.," International Journal of Advanced Research, vol. 4, no. 8, pp. 640-660, 2016.

[15] C. Staelin, "Parameter Selection For Support Vector Machines," HP Laboratories Israel, Haifa, 2003.

[16] Cristina Saverino, Zainab Fatima, Saman Sarraf, Anita Oder, Stephen C. Strother, and Cheryl L. Grady, "The Associative Memory Deficit in Aging Is Related to Reduced Selectivity of Brain Activity during Encoding.," Journal of cognitive neuroscience, 2016.

[17] Yonas B. Dibike, Slavco Velickov, Dimitri Solomatine, "Support Vector Machines: Review and Applications in Civil Engineering," Proc. of the 2nd Joint Workshop on Application of AI in Civil Engineering, Cottbus, Germany, 2000.

[18] DTREG, "SVM - Support Vector Machines," DTREG, [Online]. Available: http://www.dtreg.com/svm.htm. [Accessed 13 01 2013].

[19] S. Haykin, Neural Networks, A Comprehensice Foundation, PEARSON, Prentice Hall, 1999.

[20] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin,, "A Practical Guide to Support Vector Classification," Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, 2010.

[21] Stephen C. Strother, Saman Sarraf, Cheryl Grady, "A Hierarchy of Cognitive Brain Networks Revealed by Multivariate Performance Metrics," in Signals, Systems and Computers, 2014 48th Asilomar Conference, Asilomar , 2014.