A Comparison of Multilayer Perceptron and Support Vector Machine for Oncological Data

Saman Sadeghi Afgeh and Shakeel Raja

1 INTRODUCTION AND MOTIVATION

Breast cancer is the most common form of cancer in women, and the second most common form of cancer worldwide. The American Cancer Society projects that 1.688.780 cancer cases will occur in the United States in 2017, 35.6% of which will lead to death [1]. The early diagnosis and prognosis of breast cancer involves detection and classification of cancerous cells. This has led biomedical and bio-informatics specialists to become interested in the application of Machine Learning and other AI methods, including Artificial Neural Networks, Support Vector Machines, Decision Trees and Bayesian networks [2]. These predictive methods proved to be very effective in identifying pathological conditions in cells and organs.
This paper presents the application of two such algorithms, Multilayer Perceptrons (MLPs) and Support Vector Machines (SVMs), to the task of identifying the malignant or benign nature of cancerous cells removed from a breast mass. The aim of the experiment is to compare and discuss the accuracy, efficiency and general performance of the two algorithms while performing a predictive analysis on diagnostic data.

2 DATASET DESCRIPTION AND PRE-PROCESSING

The Breast Cancer dataset [3] [4], first obtained from Dr. William H. Wolberg at the University of Wisconsin Hospitals, Madison, is composed of 30 continuous variables and 569 observations. The dataset is based on ten original features describing cancerous cell nuclei derived from a digitized image of a fine needle aspirate of a breast mass. For each of these ten features, the mean, standard error and the ’worst’ value (defined as the mean of the three largest values) have been calculated, resulting in a total of 30 continuous features: the original variable *area*, for example, has been split into three separate features, *area_mean*, *area_SE* and *area_worst*. The dataset reported only these derived features, not the original variables. The response variable is a categorical variable indicating whether the tumour is malignant or benign. The dataset contains 357 benign and 212 malignant examples. The distribution of all variables with respect to response variable is shown as violin plot in Figure 1.

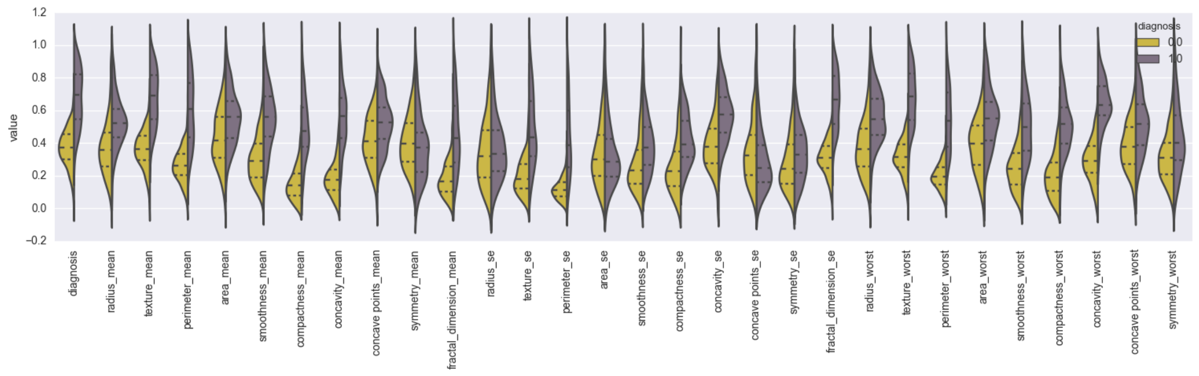


Figure 1: A violin Plot showing class distributions for each variable

2.1 Data Cleaning and Normalisation

Initial analysis of the data revealed a huge variation in attributes values and the presence of a few outliers. It was hence decided to apply some initial cleaning to the data for better generalisability. For each feature, all observations having values greater than 3 standard deviations away from the mean were removed from the dataset. This resulted in a slightly smaller dataset, with 495 observations having 332 benign and 163 malignant examples. No missing values were found in the dataset.
Standardizing the inputs makes training faster and reduces the chances of getting stuck in local optima. Furthermore, learning rate decay for MLPs can be performed more efficiently when inputs are standardized. The data pre-processing for this experiment included standardizing all variables to a 0/1 interval and coding the categorical response variable to a binary vector (equal to 1 if the tumour is malignant, and 0 otherwise), which was then adjusted to -1/1 for training and testing with SVM.

3 THE MODELS

3.1 Multilayer Perceptron

The Multilayer Perceptron is an example of Artificial Neural Network (ANN), using supervised learning methods to approximate a linear or non-linear function for classification or regression tasks [5]. Based on biological models of the brain, ANNs are composed of nodes (called *Neurons*) connected by synaptic weights and separated into input and output layers. Each neuron has an associated activation function, which determines the neuron's activation state and transforms the input data before passing it on to the next layer. The most basic ANN model is the Perceptron, which is limited in its architecture to a single input and output layer, with no hidden layers in between [6].
A Multilayer Perceptron is an evolution of the original Perceptron model, with the addition of at least one hidden layer between the input and output layers and without intra-layer connections. Each neuron i in the network computes its activation s_i with respect to an incoming excitation signal (or *Input Potential*). The net input net_i is calculated as $\boldsymbol{net_i = \sum_{j \in pred(i)} s_j \, w_{ij} - \theta}$, where $pred(i)$ is the set of predecessor neurons of unit i, w_{ij} is the connection weight from predecessor unit j to i, and θ is a bias term, which can be treated during learning as a weight from a neuron with activation 1. The MLP is a feed-forward network, using sigmoid, hyperbolic or logistic activation functions for activation.
MLPs are usually trained with back-propagation learning [7], an implementation of the Stochastic Gradient Descent minimization algorithm [8] which consists of an iterative adjustment of the weights on the synapses in order to minimize a given error function. This is achieved by computing an error on each training example and by adjusting the weights in the direction of the gradient of the error function with respect to the weights. Thus the learning rule becomes $\Delta \boldsymbol{W_{ij}} = -\boldsymbol{k} \, (\boldsymbol{\partial E_p} / \boldsymbol{\partial W_{ij}})$ where E is the error and k is the constant of proportionality. MLPs have been very successful in tackling very diverse problems [9] [10]. Cybenko showed that an MLP with a single hidden layer and sigmoid activation function can approximate an arbitrary function g(x) [11], while Hornik proved the result for a network with any bounded non-constant activation function [12].

3.2 Support Vector Machine

Support Vector Machines (SVMs) are a supervised learning model aiming to find an optimal separating hyperplane between classes of observations, where optimality is defined as maximizing the margin of separation between positive and negative examples [13] [14]. The separating hyperplane of an SVM is found by maximising the distance between the hyperplane and the support vectors. The nature of the problem is then one of convex optimization, with a quadratic objective function and linear constraints. The SVM algorithm, given functional margin $\boldsymbol{\gamma^{(i)} = y^{(i)}(w^T x + b)}$, and training examples (x,y) can then be reduced to the following optimisation problem: $\boldsymbol{min_{\gamma,w,b} \frac{1}{2} \left||w|\right|^2 \; s.t. \;\; y^{(i)} \left(w^T x + b\right) \geq 1,..,m}$.
SVMs can achieve a good generalization performance on pattern classification problems [13], have an intuitive and easy geometric interpretation, and unlike MLP training SVM training always achieves a global minimum [15]. A parallel can be drawn between the SVM algorithm and other types of neural computing methods. The standard linear SVM attempts to draw a separating line between two sets of data, basically solving the same set of problems that the Perceptron model is trying to solve. However, Perceptrons and SVMs deal with non-linearly separable data in different ways. To deal with non-linearly separable data, the Perceptron model is augmented by adding at least one hidden layer, thus defining the Multilayer Perceptron model, while SVMs use kernel functions to map the features to a higher dimensional space where the classes are assumed to be linearly separable (the so-called 'kernel trick') [16]. Finally, a variation of the original SVM model, the Soft Margin SVM, allows for a certain number of data points to be misclassified in the training set in order to avoid over-fitting, and make the separating hyperplane less dependent on outliers [17]. Due to the nature of our problem and dataset, this is the version of SVM we use in our experiment.

4 HYPOTHESIS STATEMENT

While both MLP and SVM are well suited for regression and classification tasks, with studies finding that SVM and MLP tend to perform very similarly in different applications [18], numerical experiments indicate that SVM is generally a slightly better model for classification tasks [19]. We therefore expect our SVM model to perform generally better than the MLP in our classification experiment. Following the literature [19], we also expect SVM to be more computationally efficient and to train faster than the MLP.

5 TRAINING AND EVALUATION METHODOLOGY

We performed hyper-parameter optimisation in order to select the best model for final comparison and to evaluate the suitability of the two algorithms for our dataset. The data was split into 80% training and 20% test set for hold-out validation. The holdout test set was then set aside and reserved for evaluating the test performance of the two best performing models indicated by our grid search. In our grid search, both classifiers were evaluated with 10-fold cross-validation. The combination of hold out and K-fold cross-validation was chosen to maximize the classifiers' robustness and to increase generalisability.

5.1 MLP Architecture and Learning Optimization

Training a Multilayer Perceptron consists of an iterative adjustment of the network's weights in order to find an optimal set of weights that ensures maximum accuracy and computational efficiency. A number of hyper-parameters can be fine-tuned to achieve these goals by modifying the MLP network architecture and configuring the supervised learning behaviour. For this experiment, an MLP with a single hidden layer was chosen, as it was deemed sufficient for our purpose and is justified by Cybenko's result [11]. For training, we used Backpropagation with Adaptive Learning Rate and Momentum terms [20], and we chose Mean Squared Error (MSE) as the loss function. MSE measures the square of the sum of the difference between each output and target values. A maximum of 100 epochs or 10 validation fails was used as the stop criteria for training. Finally, we settled on an MLP with *logh* activation function for the neurons in the hidden layer and *logsig* activation function for the neurons in the output layer. Four other hyper-parameters were tuned starting with number of neurons and learning rate, and then by using optimal values for these parameters to further optimize learning rate decay and momentum. The following parameters were chosen for our grid search:

- <u>Number of hidden neurons:</u> [1, 5, 10, 15, 20, 25 , 30]
- <u>Learning Rate:</u> [0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.10]
- <u>Learning Rate Decay</u>: [0.1. 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8. 0.9. 1.0]
- <u>Momentum</u>: [0.1. 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8. 0.9. 1.0]

To identify the optimal values of these hyper-parameters, a grid search approach was implemented to continuously bring small changes in these hyper-parameters and observe model performance.

5.2 SVM Kernel Function and Box Constraint Optimization

We explored different kernel functions for our SVM model. We report the results for a run of the algorithm respectively with no kernel function ('linear' kernel) and with a Radial Basis Function (Gaussian) kernel function (defined as $\boldsymbol{K(x,y) = exp}(\frac{||w||^2}{2\sigma^2})$)). Asymptotic analysis has shown that an SVM with Gaussian kernel and properly tuned hyper parameters always outperforms linear SVM, as the latter is a degenerate case of the first [21]. As our initial analysis showed that the classes are likely not linearly separable, we expect linear kernel SVM to perform significantly worse than Gaussian SVM, even with soft margins allowing for misclassification.
The parameter search for our SVM is limited to two parameters, the Box Constraint (C) and the Kernel Sigma (s) for the RBF kernel, and to the Sigma parameter only for the linear SVM. The Box Constraint C controls for the degree of misclassification around the margin. In Soft Margin SVM, a data point can fall in the region of separation between the decision boundary and the support vectors. These non-separable points are correctly classified if they fall on the correct side of the decision surface. The regularization parameter C, controls the 'softness' of the margins, that is, it "controls the trade-off between complexity of the machine and the number of non-separable points" [13]. A large C raises the cost associated with each misclassification (i.e. it makes the margins "harder"), requiring more data points to be correctly classified by the decision boundary (the case with C = ∞ is equivalent to the Hard Margin SVM model). This will increase training set accuracy, but can cause overfitting and makes the classifier more sensitive to outliers. The second parameter, the Kernel Sigma, is the standard deviation of the Gaussian curve used in the RBF kernel, and thus controls the width of the Gaussian curve. A larger value of sigma increases the size of the decision boundary, thus generally lowering train set accuracy but reducing the risk of overfitting.

6 EXPERIMENTAL RESULTS

6.1 MLP Grid Search Results

The results from grid-search based optimization are presented in Figure 2 (a-c). In 2a, a clear increase in validation performance (defined as 1-MSE) can be seen as the learning rate increases from 0 to 0.1. This indicates that the number of training epochs is not large enough for the smaller learning rate to reach the minimum of the loss function. As a small learning rate implies a higher number of iterations, the system is failing to reach the optimum solution before reaching the specified limit in number of epochs (100). A decrease in validation performance can be seen as the number of hidden neurons increases, indicating that the network might be overfitting the training data in each fold. An optimum point was located with 10 hidden neurons and learning rate of 0.1. These values were set to further tune learning rate decay and momentum parameters, the tuning of which is shown in Figure 2b.

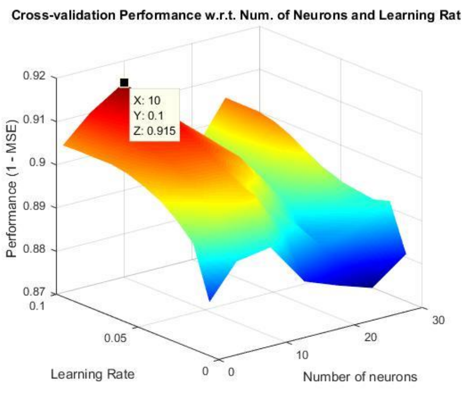


Figure 2(a) N. of neurons and Learning Rate

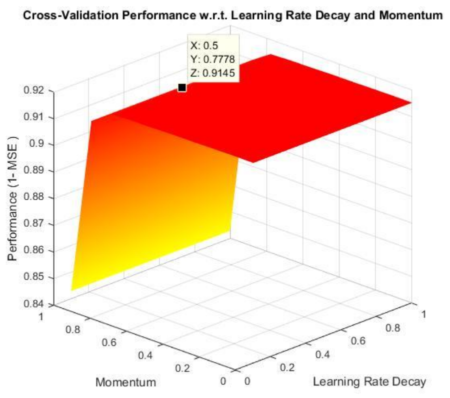


Figure 2(b) LR decay and Momentum

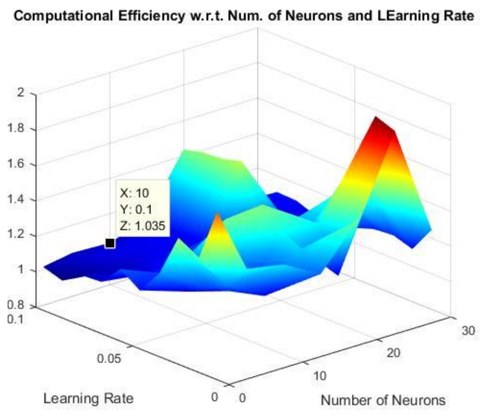


Figure 2(c) Training time(LR and NNeurons)

Maximum performance is achieved with momentum value less than 0.5, after which a sharp decline in performance is visible. This can be seen as an indicator of the fact that higher values of momentum do not let the model converge to the minimum of the loss function and force it to move away from minima with larger weights. Learning rate decay has no effect on the validation performance and the optimization is mainly driven by changing values of momentum. Based on this observation, a learning decay value of 0.5 and a momentum value of 0.77 were chosen for further analysis and comparisons. Figure 2c shows the effect of increasing the number of neurons and the value of the learning rate on computational efficiency of the classifier. The figure reports the time (in seconds) taken for training the MLP with given parameter values. As expected, an increase in number of neurons adds further level of complexity to the model and a sharp increase in training and validation time can be witnessed when the number of neurons becomes greater than 20.

An increase in learning rate at optimal points can be seen to reduce the computational time as it may help to find the minimum loss function, but with higher number of hidden neurons and more complex network, an increase in learning rate also proves to be computationally expensive as seen with 10 neurons with learning rate of 0.1.

These observations helped us select the optimal values of all chosen hyper-parameters. We then used those values to test the model on the hold-out test set and to compare the performance of MLP and SVM in terms of computational cost and performance.

6.2 SVM Grid Search Results

To facilitate computation and comparison, the values of the parameters for the search were chosen as each of the 10 equally spaced values in the intervals (0.1,15) for s and (0.1,10) for C. We did perform similar searches for different intervals, but only report these two intervals for simplicity. For each choice of parameters, we trained a Soft Margin SVM model and performed 10-fold cross validation on it.

For the linear kernel SVM only C was adjusted and s kept constant at 1, while for the Gaussian kernel version both values were searched for. The reported accuracy is the average of the accuracy over the 10 different runs. Accuracy here is defined as ***1 - ClassificationError***, where Classification Error is the percentage of observations that are misclassified.

Figures 3a and 3b report the result of our grid search for the SVM models. Figure 3a shows the model accuracy for all searched values of C. In the figure, s is kept constant at 1 for the Linear SVM and for one of the two RBF models, while the second RBF has s kept constant at its best value (the one that achieves the highest accuracy) of 1.756.

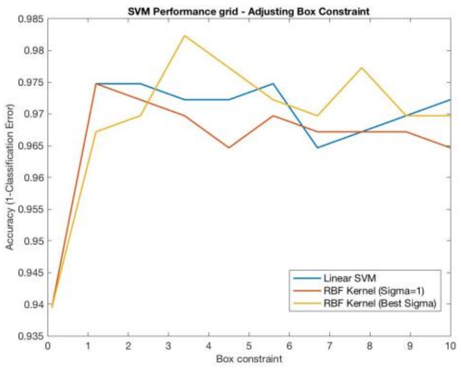


Figure 3 (a) Adjusting C (with constant Sigma)

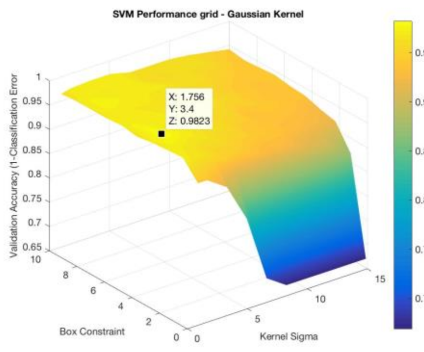


Figure 3 (b) RBF kernel s=(0:1;15) and C=(0:1;10)

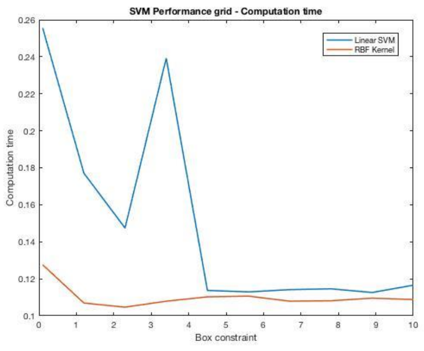


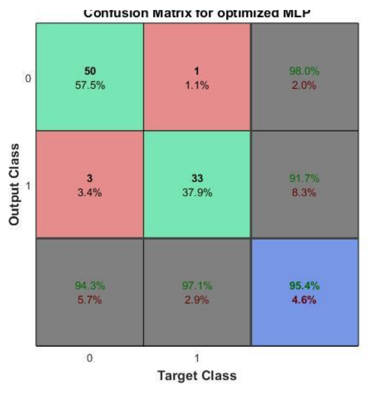
Figure 3(c) Training time for each run

All SVM models perform significantly worse for C between 0 and 1, and all achieve a better accuracy for larger values of C. This indicates that the data points are relatively well separated, and that a too low penalty for misclassification can results in a worse performance. Surprisingly, the soft margin linear SVM performed only slightly worse than the SVM with Gaussian kernel. This could be an indication that our data is close to being linearly separable, thus allowing for some misclassification achieves good results, with accuracy as high as 97.5%. As expected, the Gaussian kernel performs best, with a maximum accuracy in our grid search of 0.9823. Interestingly, the Gaussian SVM with constant sigma of 1 performed significantly worse than even the linear SVM over most of the range of values for C. Adjusting s however made the RBF SVM outperform both, reaching the maximum accuracy with s=1.756 and C= 3.4. Training times are comparable, with the small size of the dataset and of the grid used for the search not making any difference noticeable. Figure 3c shows training times for each of the runs of the linear and Gaussian SVM, for different values of C (s kept constant at 1). As we can see from the figure, linear SVM takes significantly longer to train, but only for lower values of C (C < 5), while training times are comparable for larger values of C. In practice, however, the grid search for the Gaussian kernel took much longer to compute, as it included one more parameter.

Figure 3b shows the results of our grid search for the SVM with Gaussian kernel. The 3D plot shows the 10-fold cross-validation accuracy for each combination of the two parameters s and C. While accuracy is very low for very small values sigma, and for the region identified by small C and high sigma, most of the parameter space corresponds to a high accuracy (higher than 90%). Other than for the values of C between 0 and 1, accuracy is more or less constant across the range of values of the box constraint, and is higher for lower values of Sigma. More specifically, we can identify from the figure an area of high accuracy in the region delineated by the interval (0.1, 5) for s. This interval comprises the combination of parameters s=1.756 and C=3.4 (highlighted in the picture), which gives the highest 10-fold accuracy in our grid search (Accuracy of 98.23%). This parameter combination was chosen as our best performing SVM model.

7 HOLDOUT TEST SET RESULTS

Figure 4 MLP confusion matrix for hold-out test



Finally, we tested the MLP and SVM on the 20% holdout test set aside at the beginning. For each model, the parameters identified with the grid search have been chosen, and the models have been trained on the whole training set. Both models performed well in the test set, as measured by accuracy and MSE.

Figures 4 and 5 show the confusion matrices for the MLP and SVM models. As the weights in an MLP are initialised randomly, and because of the nature of the Stochastic Gradient Descent implementation of backpropagation, every training of the model is going to generate different results. We therefore trained and tested the model several times, inspecting various metrics (including accuracy, classification error, MSE, Sensitivity and Specificity) to determine the performance of our model. The MLP performs well, achieving high accuracy on the holdout test set. The percentage of observations correctly classified oscillated between 92% and 99%, with a few instances where the accuracy reached 100%. We performed training and testing 1000 times, and the MLP achieved an average test accuracy of 95.4%, as on average 4 out of 99 observations in the test set were misclassified. The quality of the predictions was generally good, with low errors (defined as the difference between the predicted value and the target).

Unlike MLPs, SVMs predictions do not vary depending on initial weights or due to stochastic optimisation, and always achieve a global minimum in the optimisation process. The Gaussian Kernel SVM performed much better than the MLP, achieving an accuracy of 98.99% on the test set. Out of the 99 observations in our test set, only one was misclassified. The SVM was also faster than the MLP, training in 0.10 seconds instead of 0.15 (on average).

This is in accord with the literature that often finds SVM to train faster than MLPs [19]. Given the small size of the dataset the difference was negligible, but it might become more important for larger datasets.

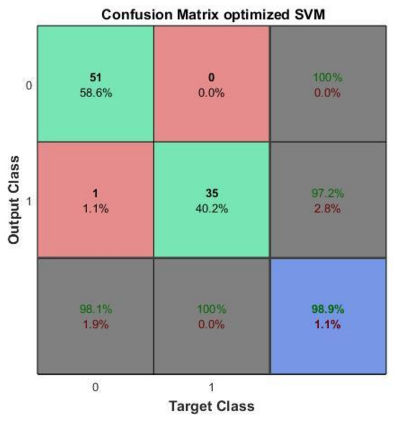


Figure 5 SVM confusion matrix for hold-out test

However, the high accuracy in our results could be due in part to the small size of our dataset, and of our test set in particular. In the future, a larger dataset would be required to gain a better estimate of whether our algorithms are as efficient and accurate as they seem. Furthermore, due to the very small size of the test set, the difference in accuracy is influenced heavily by a small number of misclassified observations. Another factor that is likely to have influenced our results is the removal of outliers from the original dataset. While this has reduced the size of the dataset used for training the models, it has also removed those data points that were likely to be the hardest ones to classify. We believe that our choice was justified by the anomalous nature of such observations, likely to be due to statistical anomalies, however, the performance of our classifiers would decrease significantly when faced with new data.

8 CONCLUSION

We presented a comparison of two Neural Computing algorithms, the Multilayer Perceptron and the Support Vector Machine. By applying these two methods to an oncological dataset, we found that while both algorithms produce excellent results, the SVM performed better, giving a more accurate test prediction and a marginally faster performance. However, the extremely high accuracy could be due to the small size of the dataset, and to the large number of features used for training the MLP and SVM. While we didn't see any indication of overfitting, in the future a larger dataset should be employed, and this would arguably lead to a less accurate classification and a larger errors.

REFERENCES

[1] American Cancer Society. Cancer Facts and Figures 2017. Atlanta: American Cancer Society. 2017. Accessed online at: *https://www.cancer.org/content/dam/cancer-org/research/cancer-facts-and-statistics/annual-cancer-facts-and-figures/2017/cancer-facts-andfigures-2017.pdf*

[2] Kourou, K., Exarchos, T.P., Exarchos, K.P., Karamouzis, M.V. and Fotiadis, D.I., 2015. Machine learning applications in cancer prognosis and prediction. Computational and structural biotechnology journal, 13, pp.8-17.

[3] W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on Electronic Imaging: Science and Technology, volume 1905, pages 861-870, San Jose, CA, 1993.

[4] O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. Operations Research, 43(4), pages 570-577, July-August 1995.

[5] Haykin, S.S., 2009. Neural networks and learning machines (Vol. 3). Upper Saddle River, NJ, USA:: Pearson.

[6] Rosenblatt, F., 1958. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological review, 65(6), p.386. Vancouver

[7] Williams, D.R.G.H.R. and Hinton, G.E., 1986. Learning representations by back-propagating errors. Nature, 323(6088), pp.533-538.

[8] Amari, S.I., 1993. Backpropagation and stochastic gradient descent method. Neurocomputing, 5(4), pp.185-196.Vancouver

[9] Gardner, M.W. and Dorling, S.R., 1998. Artificial neural networks (the multilayer perceptron) a review of applications in the atmospheric sciences. Atmospheric environment, 32(14), pp.2627-2636. Vancouver

[10] Hontoria, L., Aguilera, J. and Zufiria, P., 2005. An application of the multilayer perceptron: solar radiation maps in Spain. Solar Energy, 79(5), pp.523-530.

[11] Cybenko, G., 1989. Approximation by superposition of sigmoidal functions. Mathematics of Control, Signals and Systems, 2(4), pp.303-314.

[12] Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. Neural networks, 4(2), pp.251-257.

[13] Haykin, S., 2004. Neural Networks - A comprehensive foundation. Neural Networks, 2(2004), p.41.

[14] Vapnik, V.N., 1982. Estimation of dependences based on empirical data (Vol. 40). New York: Springer-Verlag.

[15] Burges, C.J., 1998. A tutorial on support vector machines for pattern recognition. Data mining and knowledge discovery, 2(2), pp.121-167.

[16] Boser, B.E., Guyon, I.M. and Vapnik, V.N., 1992, July. A training algorithm for optimal margin classifiers. In Proceedings of the fifth annual workshop on Computational learning theory (pp. 144-152). ACM.

[17] Cortes, C. and Vapnik, V., 1995. Support-vector networks. Machine learning, 20(3), pp.273-297.

[18] Barabino, N., Pallavicini, M., Petrolini, A., Pontil, M. and Verri, A., 1999. Support vector machines vs multi-layer perceptrons in particle identification. In ESANN'1999 proceedings - European Symposium on Artificial Neural Networks Bruges (Belgium), 21-23 April 1999, D-Facto public., ISBN 2-600049-9-X, pp. 257-262

[19] Osowski, S., Siwek, K. and Markiewicz, T., 2004, June. MLP and SVM networks - a comparative study. In Proceedings of the 6th Nordic Signal Processing SymposiumNORSIG (pp. 9-11).

[20] Moreira, M. and Fiesler, E., 1995. Neural networks with adaptive learning rate and momentum terms (No. EPFL-REPORT-82307). Idiap.

[21] Keerthi, S.S. and Lin, C.J., 2003. Asymptotic behaviors of support vector machines with Gaussian kernel. Neural computation, 15(7), pp.1667-1689.