Islam Ibrahim & Saeed Ahmed

Detecting Viral News Articles with a Multilayer Perceptron and a Support Vector Machine

I Motivation

The ubiquity of the internet has driven many news organisations to pivot their business model towards their online presence. With online articles, the revenue is predominantly driven by advertisements and in turn, advertisement pay is largely determined by the number of views that an article receives. Thus, the ability to predict with some degree of accuracy how popular an article will be holds undeniable value to media organisations. Which as a result, has led many researchers to explore this domain[1].

This project aims to address this research area by attempting to predict the virality of a news article, defined by the number of shares it receives, based on properties known prior to its publication. The two methods used to predict the virality are a multi-layer perceptron (MLP) neural network and a support-vector machine (SVM) model. This project will naturally offer a comparative study of the two different supervised learning techniques and consequently their ability to classify viral articles.

Section II will present the dataset used in this project and its preparation prior analysis. Section III introduces our learning techniques and Section IV will be our hypothesis statement. The training choices made for designing our models in detailed in Section V, before Section VI then presents our results and Section VII gives an in-depth analysis of these results and gives a holistic critique of our project.

II Data

The data used to conduct this project was available on the UCI Machine Learning Depository[1] and consists information of 39797 Mashable articles. For each article, the dataset presents a plethora of features, including a mix of both discrete variables, such as the day of the week the article was published or its category, and continuous variables, such as the sentiment polarity of the language used or the number of words in the article. Overall, there were over 20 available input features and thus the first stage of data preparation involved feature selection in order to remove any that were deemed irrelevant; leaving 9 input features used in the analysis. The target feature, which is the variable that our models aimed to predict, was a continuous variable stating the number of shares that each article received. As the two methods chosen in this project are more conducive to classification tasks, the target variable was binned into two categories – ‘Ordinary’ and ‘Viral’- and thus the models would aim to predict the class of an article based on 9 input features.

To assign the threshold that defined which class each article fell into, a distribution plot was produced showing the spread of articles by number of shares, figure 1 (left). From this, articles with over 3,000 shares were compartmentalised into the ‘viral’ class and any articles with less than 3,000 in the ‘ordinary’ class. These input features were then pre-processed through normalisation to allow the models to work effectively on them. The effect of this normalisation is exampled in figure 1 (right), showing the normalised distributed of number of shares for each day.

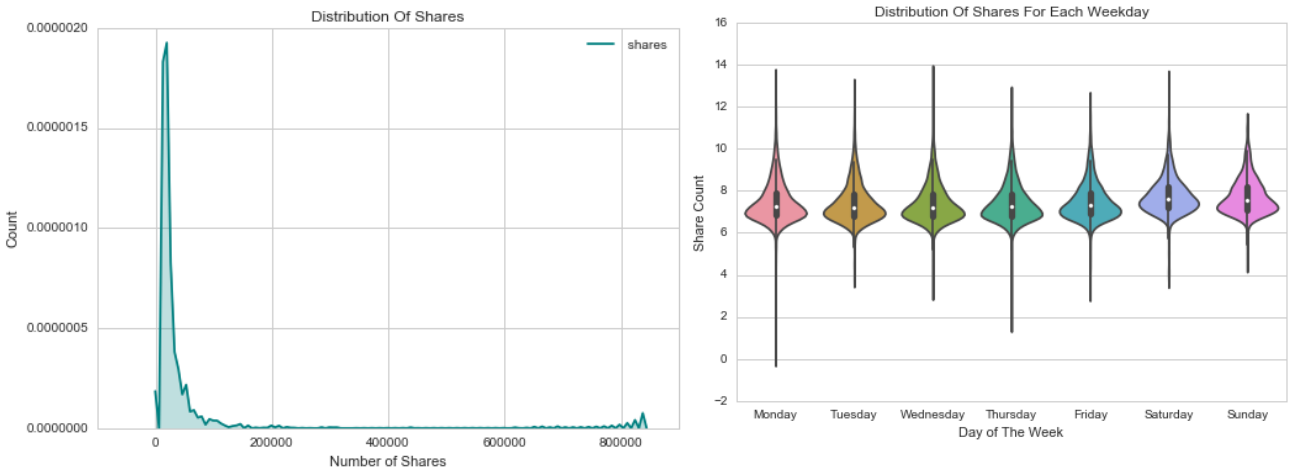

Figure 1: left distribution of articles by number of shares *and right* distribution of number of shares articles receive by day of the week of publication, normalised.

III Methods

Neural networks are a family of algorithms inspired by models of the brain and are used in a range of domains and applications, realising novel performances in tasks such as pattern recognition, natural language processing and computer vision [2]. Known for their ability to learn from data, they offer interesting properties and capabilities that arise due to the nature and architecture of the networks. Neural

1. http://archive.ics.uci.edu/ml/datasets/Online+News+Popularity

networks are known for their nonlinearity, by virtue of an interconnection of nonlinear neurons, and their adaptivity, in that they have in-built capability to learn directly from the data through adjusting connection weights between the neurons. They are trained in absence of assumptions of the underlying probability distribution of the data and in essence, they function through input-output mapping. Whereby the model is presented with input data and their corresponding outputs, it then adjusts the connection weights of the network to minimise the discrepancy between the network output and the desired output [3].

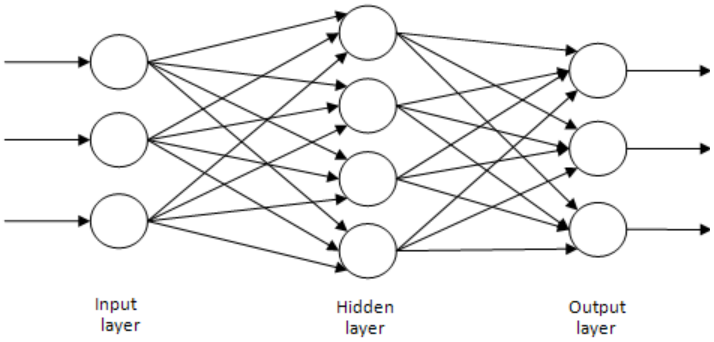

Figure 2: Illustration of the architecture of a MLP, showing 3 input nodes, 1 hidden layer of 4 nodes and 3 output nodes

Multi-Layer Perceptron (MLP) is a feed-forward neural network that comprises multiple layers and each node acting as a processing element. Each layer is made up of nodes, and every node is connected to all the nodes in the previous layer, figure 2 illustrates a basic architecture of these models. MLP apply a non-linear, sigmoid, activation function to the neurons in the hidden and output layers, which enables their adaptability in approximating a wide-ranging variety of functions. They are trained using a technique called backpropagation – which is an optimisation method of implementing gradient descent in attempts to minimise the error function of the network with respect to the set of synaptic weights [4]. In that regard, the model can at times encounter the problem of local minima. Where in attempts to find the global minimum of the error function the model can instead get 'stuck' in a local minimum due to the initial state of the model. This can be overcome by adjusting the learning rate parameter or adding a momentum function, or both [5].

The support-vector machine (SVM) algorithm is a supervised-learning algorithm that is effective in classifying linearly separable, binary, data. It is a particularly popular tool in handling large datasets and can often offer fast training speeds with little memory requirement [6]. The goal of SVM is to create a hyperplane that can best distinguish between the two classes – as shown in figure 3. The SVM algorithm can be extended to patterns of data that are non-linearly separable through the use of kernels, which map the data onto a high-dimensional space where it may experience linearity. For more intricate datasets, polynomial or Gaussian kernels are often used to achieve this. To supplement this, SVM can also use soft margins, which separates some but not all data points, when handling data that is difficult to separate. This use of soft margins can also be useful when handling datasets susceptible to outliers which can otherwise distort the SVM algorithm [7].

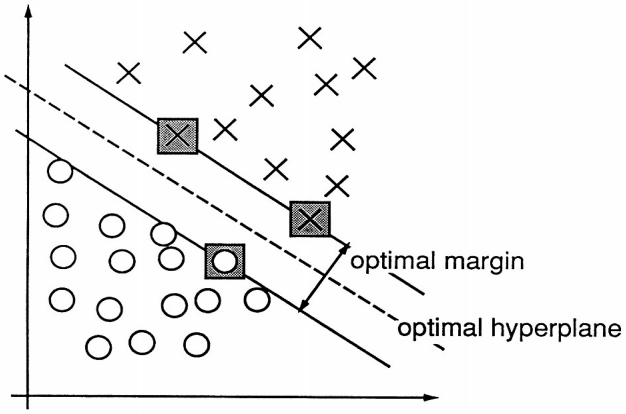

Figure 3: Visualisation of a SVM hyperplane used to separate two classes [8]

IV Hypothesis

We hypothesise that the MLP model will outperform the SVM model in terms of correctly classifying viral articles based on the input data. This is largely due to the fact that the SVM model will depend on only two input features and the two may not be adequately separable so as to permit effective classification, even with a kernel function. With the sheer training choices available for an MLP architecture, we are likely to come upon one that is effective for our dataset.

V Training

Defining the training choices of the models is an important step prior to implementing them on the dataset, as this not only governs the performance, but also the ability of the models. The core architecture of our MLP comprised 9 nodes in the input layer, each node corresponding to an input feature, and two output nodes, each corresponding to a target class. To assess how the neural network behaves with the data, it was trained over three different configurations by adjusting the parameters of the network, namely the number of nodes in the hidden layer, the learning rate and the backpropagation training function. This allowed us to deepen the exploration of how the network behaves at different configurations of its model.

To train the network, a varying number of hidden layer nodes, ranging from 5 to 40 in steps of 5, were used and subsequently employed as a parameter in deducing the optimal neural network design for this

dataset. For each number of hidden nodes, the network was also trained at different learning rates of 0.001, 0.01, 0.1 and 0.5. The training function used was a gradient descent (GD) backpropagation algorithm, and then also using a Levenburg-Marquadt (LM) algorithm as it reportedly offers quicker convergence [9]. They key distinction between the two functions is that the GD function enabled us to add a momentum function into the algorithm, which as mentioned in the section II, it is a method that can help mitigate the problem of local minima when training.

For each training setting, the dataset was randomly split into three subsets of training, testing and validation sets, according to a 70-15-15 proportional splits. Where the network learns the internode parameters for that data using the training set, then tests the network using the testing set and finally validates and tunes the hyper-parameters using the validation set. Consequently, the validation set is also the set that postulates the model's early stopping criteria. Whereby the model stops training once the validation set mean-squared error (MSE) begins to deviate and subsequently preventing the model from overfitting.

Due to the nature of the input data, containing 9 features, it appeared unwise to attempt the SVM algorithm on such high-dimensional data. The inherent nature of SVM pertains that even if classification is possible at this dimensionality, the time taken to train the model, and subsequently to visualise the resulting support vectors, would be impractical. As such, SVM algorithm was applied to just two chosen input features as opposed to the nine used for the neural network task. This dimensionality reduction permitted the visualisation of the resulting classification and the corresponding support vectors with ease.

In spite of this feature reduction, the classification between the two classes was not linearly separable. As such, it was necessary to assign a kernel function to the SVM, in which a radial basis function was chosen. To train the SVM algorithm, 10-fold cross validation was applied to the dataset. Whereby the entire dataset is partitioned into 10 equally sized subsets and then for each training instance the algorithm is trained using 9 subsets and then tested using the remaining subset. Then by repeating this 10 times and averaging the error means the results are more robust and consequently improves one's confidence in the generalizability of the algorithm.

As a binary classification task, the primary approach used to evaluate the result of both will be the classification error. Based on the trained MLP and SVM, the dataset is partitioned so that a random 20% of the overall dataset is used to test the classification ability of the models. By using that subset as the input into the models, the output classification for each instance is cross-checked with the actual target value to gain the classification error.

In the case of the neural network, the evaluation went beyond just reviewing the classification error. At each training setting, comprising varying numbers of hidden nodes and learning rate, the convergence properties of the neural network training were also examined. As such, the time taken and number of epochs required to train the network were evaluated to ascertain the most optimal conditions for the given dataset. Furthermore, the mean-squared error of the dataset at each training setting was also used to compare the model performance across the different configuration settings.

VI Results

For training the MLP, stopping criterion were set to mitigate the risk of overfitting and avoid excessive training time. Thus four different stopping criterion were assigned and training halted when one was reached. Those were, a maximum epoch threshold of 200, 600 second time limit, 5 validation failures and a minimum gradient for the validation MSE of 0.001. Otherwise, training would stop once convergence was achieved.

Table 1 shows the average training performance of the MLP network trained across the varying configurations listed in Section II, for both the LM and GD training functions. For the LM function, the model experienced a comparatively steep performance curve whereby the MSE dropped sharply after the first couple of epochs and then stabilised rather quickly. As table 1 shows, the model converged at an average of 6 epochs and the average training time was less than 3 seconds. Meanwhile, when using GD training function, the model learning was gradual and continued to improve even at higher epochs, failing to converge in some cases and thus training only halted once a stopping criteria was met. This is clear in that the average convergence was at 156 epochs. Nonetheless, the average training time was still close to that of the LM training function, at just over 3 seconds.

Table 1: average MLP training performance using LM and GD training function

Training Function	Average Epoch Convergence	Avg. time	Classification Error	MSE
LM	6	2.5s	24%	0.18

GD	156	3.4s	35%	0.80

Of the LM and GD training functions the optimal performance, according to the classification error and MSE, was found at a learning rate of 0.1 and hidden layer nodes 15 and 20, respectively. The optimal case for both did not differ much in their error or time taken to convergence, shown in table 2, however the epochs taken to reach this vary greatly.

Table 2: performance of MLP network at optimal configuration for both LM and GD training functions

Training Function	**Hidden Layer nodes**	**Learning Rate**	**Epoch**	**Time**	**Classification Error**	**MSE**
LM	15	0.1	3	0.8s	23%	0.17
GD	20	0.1	0.1	200	3.9s	0.19

The discrepancy between both training functions is illustrated more prominently in figure 4, showing the performance at the different training configurations for the LM (left) and GD (right) method. This gives a clearer view of the behaviour of the neural network training at different numbers of hidden nodes and training rates for both the LM and GD training function, by observing the classification error at each setting. For the MLP with a GD training function, the error varied greatly, reaching 80% at some points, depending on the learning rate and number of hidden nodes, with only the learning rates of 0.1 and 0.01 showing relatively stable results. Conversely, the MLP with a LM training function produced rather consistent MSE, ranging between 23% and 24%, at all settings. This indicates that the LM technique was less sensitive to the parameter choices for training the network.

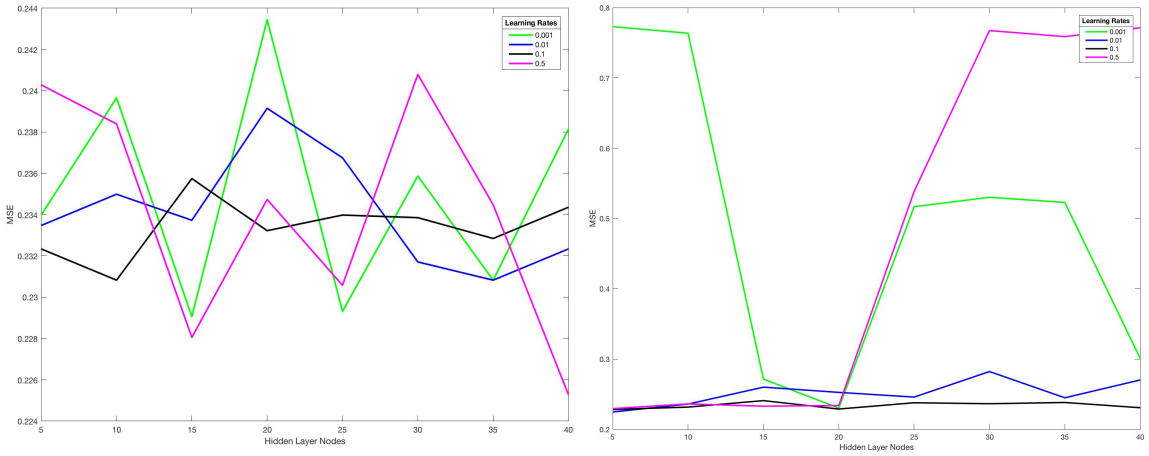

Figure 4: Training performance for LM (left) and GD (right) MLP networks, showing classification error at different number of hidden layer nodes and learning rates

The classification error is visualised using confusion plots (figure 5) which shows the classification performance of the MLP networks in more detail. Where the green boxes show those correctly classified and the red showing the incorrectly classified, and hence the classification error. The observation of interest here is the number of instances the network correctly classified the news articles as 'viral', in which the LM training function correctly classified more of such instances. Nonetheless, it is clear that the rate of misclassification is prevalent, with the number of articles the network is classifying as viral are predominantly indeed, not viral.

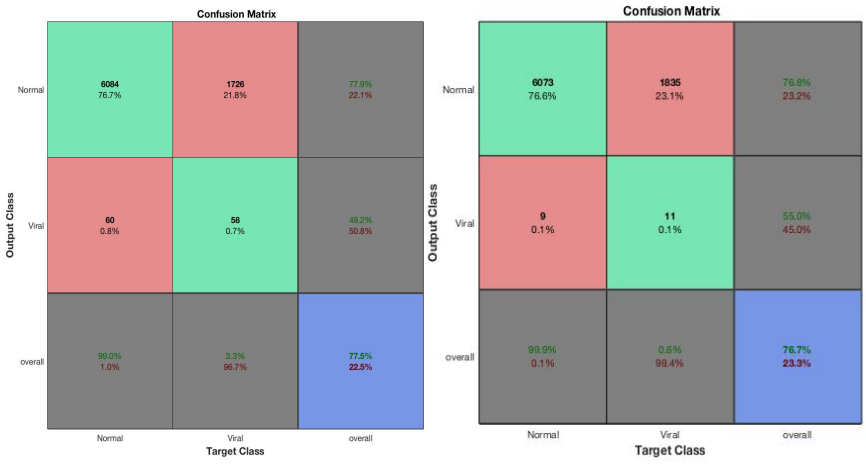

Figure 5: Classification error at optimal MLP network configurations for LM (left) and GD (right) training functions.

For the SVM model meanwhile, the classification error following the 10-fold cross-validation was 22.9%. Thus giving a better, albeit marginal, improvement than the most optimal MLP configuration. Besides, the cross-validation applied to the SVM model means the results are more indicative of its generalisation beyond the dataset studied in this project.

Figure 6 shows the result of the SVM model trained on the test data, by plotting the input features (number of unique tokens and sentiment polarity of the article) and their corresponding support vectors, shown by the black circles. Those support vectors indicate the data points that lay closest to the decision surface and thus were most difficult for the algorithm to classify.

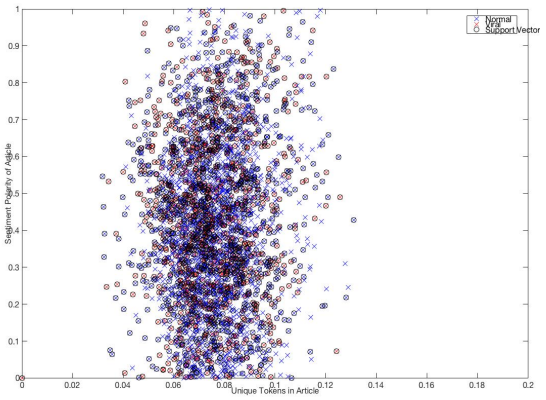

Figure 6: Illustration of support vectors on normalised input data of sentiment polarity and number of words in article

VII Discussion and Analysis

The trained MLP neural networks produced adequate results on the dataset. At the best configuration deploying a LM training function, 0.1 learning rate and 15 hidden layer nodes, the MSE was 0.17 and the classification error was 23%. On average, the neural network at each configuration took little time to train, circa 3 seconds, and often converged after the first few epochs, 6 in the LM training function case. Now although the results at the optimal configuration did not vary much between the LM and GD training function method, the performance was more polarising when using GD. Therefore, the network is sensitive to not only the configuration parameters, but also the training function used. Ultimately, the LM training function was deemed the optimal set up for this dataset as, by shown in figure 4 and figure 5, the classification error was stable at circa 23% at all learning rates and numbers of hidden layer nodes. Whereas using GD, the classification error varied between 80% at its highest and 24% at the lowest. The results using LM training function also returned a lower classification error on average, converged at lower epochs, and was less dependent on the training parameters.

Therein also lies the key drawback with using neural networks – the number of training parameters needed to be set and subsequently the varying combination each. This is a time consuming process and it can be difficult to judge if the optimal configuration for the particular dataset has been found. However, once the best configuration has been discovered, training the neural network to the data takes minimal time, as shown in our results. Nonetheless, when factoring in the total time taken to train the neural network with all the different configuration of parameters, it is an onerous procedure in comparison to the SVM model.

Using the SVM algorithm, the classification error was marginally better than that of the best MLP configuration. In addition, the number of input features used to classify with the SVM model, 2, was lower than that used in the neural network model, 9. This is due in part to the pre-processing of the data that is necessary when using an SVM algorithm. But primarily it was due to the fact that the SVM algorithm attempts to linearly separate all the input features fed into the model according to their classes, and thus feature reduction was necessary to ensure the model was able to work effectively on the dataset. In our case, this presented a clear upshot for the SVM model in that the classification error greatly increased when attempting to use the neural network on only 2 input features. In addition, the 10-fold cross validation applied to the SVM algorithm implies that the results achieved by this method are more robust to generalising to examples beyond our dataset. For the neural network, there was no clear impetus to manually reduce the input features. With enough data, training of the neural network allows it to discover the key input features that influence the output by virtue of its architecture comprising nodes, connections and the weights.

The key hindrance of training the SVM model on our dataset is the sheer time taken to train the model relative to the training time of the neural network. And by observing figure 6, this is understandable. The data is confined within a small space and even with the radial basis function kernel applied, the model has difficulty separating the data effectively. This is evident in the large number of support vectors shown in the graph, which indicate the data points that lie in the decision surface and thus the SVM model has difficulty classifying.

All in all, to keep in line with Occam's razor, SVM becomes the clear candidate for the classification task presented in this project. SVM not only returns a better classification error, but does so with only 2 input features, as opposed to 9, and with only one main training parameter needed to be applied. The MLP meanwhile has over 5 different parameters that need assigning, and adjusting, with most of those varying the results of the model. Nonetheless, the training time for the neural network, once the parameters have been assigned is greatly lower than that of the SVM. In addition, the plethora of training configurations

available for MLP neural networks gives the ability for the user to tune the neural network. Which is not quite so available when using SVM.

Reviewing the classification error alone can be misleading. Although our models achieve a classification error of circa 23%, the model fails to detect over 80% of the viral articles – as shown by the confusion plots in figure 5. In defence of this, detecting viral news articles is akin to an abnormal event detection task insofar as there is no heuristic that subject's certain articles to achieve virality. There are myriad factors that can be the cause of a news article being widely shared, with many of them being unquantifiable, or unforeseen. Plus, ultimately, the biggest factor would be the actual content of the article. Thus this task by its very nature is a difficult undertaking. With that in mind, the results achieved by the models in this project are somewhat comprehendible in that at the optimal configurations they are able to detect some of the articles that are likely to end up being widely popular. This could be of great value for a media company as once the model is trained, it could supplement their judgment of whether an article would be popular. In that regard, the neural network would be more useful than the SVM as the training time is much shorter once the optimal parameters are chosen.

Following on from this, in terms of the practicality of this project, it would be undeniably more useful if we attempted to predict the shares of an article via a continuous target variable as opposed to the categorical target variable used in this project. In other words, a regression analysis that predicted the shares of an article would be more salient than a classification analysis. Classifying whether an article will gain over 3,000 shares, as done in this project, has limited practicality in comparison. Thus to truly extend the work done in this project, applying a random forest or logistic regression models to this data to predict the shares of an article could prove beneficial.

Besides this, other ways to extend this work could be to apply principal component analysis as a feature reduction method prior to applying the SVM algorithm. This may shorten the training time and allow the data to be separated more readily. Alternatively, a feature importance exercise could have been carried out following training of the algorithms. This analysis would be informative in that it would highlight which the features which are driving the classification of viral articles.

VIII Conclusion

Our project set out to train two different models to be able to predict the popularity of a news article based on descriptive features known prior to publication. To achieve this, data was categorised into two classes – with any articles amassing over 3,000 shares classed as 'viral' and thus supervised learning techniques were developed. The methods chosen for this classification task were a MLP, trained with both GD and LM functions, and a SVM, trained with a radial basis function kernel.

For each MLP training function, the model was trained with 32 different configurations, comprising different combinations of number of hidden layer nodes and learning rates. The MLP with GD achieved the best performance, in terms of the MSE and classification error, at 20 hidden layer nodes and 0.1 learning rate. While the MLP with LM achieved the best performance with 15 hidden layer nodes and 0.1 learning rate. Overall, the MLP trained with LM achieved better performance in that the MSE and classification error remained consistently near the optimal of 0.17 and 23%, respectively. It also achieved convergence at an average of 3 epochs and training time was minimal, with an average of around 3 seconds across all training configurations attempted. Meanwhile the SVM model achieved marginally better classification error of 22.9% after 10-fold cross-validation – although the time taken to train the model was significantly greater, reaching over 250 seconds.

All in all, considering the nature of the task whereby we are attempting to predict viral articles, which are in essence unique occurrences with no defined heuristic governing it, both models achieve adequate performance and each offer their own value to the user. SVM not only returns a better classification error, but does so with only 2 input features and with only one main training parameter needed to be applied. Conversely, the training time for the MLP is more optimal and it enables the user more control over its design. Nonetheless, for true applicability of the models, it would be more useful to predict the actual shares an article receives via a regression analysis, as opposed to the classification task conducted here.

References

[1] K. Fernandes, P. Vinagre, and P. Cortez, 'A proactive intelligent decision support system for predicting the popularity of online news', in *Portuguese Conference on Artificial Intelligence*, 2015, pp. 535–546.

[2] B. D. Ripley, *Pattern recognition and neural networks*. Cambridge university press, 2007.

[3] S. Haykin and N. Network, 'A comprehensive foundation', *Neural Netw.*, vol. 2, no. 2004, p. 41, 2004.

[4] M. Riedmiller, 'Advanced supervised learning in multi-layer perceptrons—from backpropagation to adaptive learning algorithms', *Comput. Stand. Interfaces*, vol. 16, no. 3, pp. 265–278, 1994.

[5] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, and others, 'Greedy layer-wise training of deep networks', *Adv. Neural Inf. Process. Syst.*, vol. 19, p. 153, 2007.

[6] S. Maji, A. C. Berg, and J. Malik, 'Classification using intersection kernel support vector machines is efficient', in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008, pp. 1–8.

[7] D.-R. Chen, Q. Wu, Y. Ying, and D.-X. Zhou, 'Support vector machine soft margin classifiers: error analysis', *J. Mach. Learn. Res.*, vol. 5, no. Sep, pp. 1143–1175, 2004.

[8] C. Cortes and V. Vapnik, 'Support-vector networks', *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[9] İ. Güler and E. D. Übeyli, 'A recurrent neural network classifier for Doppler ultrasound blood flow signals', *Pattern Recognit. Lett.*, vol. 27, no. 13, pp. 1560–1571, 2006.