Contents

1 Classifying Spam Emails 2
1.1 Support Vector Machines (SVMs) 2
1.2 Data 3
1.3 Model Techniques 3
1.3.1 Linear Kernel 3
1.3.2 Radial Kernel 4
1.3.3 Polynomial Kernel 4
1.4 Results & Discussion 4

2 Classifying Odd & Even Numbers 6
2.1 Artificial Neural Networks (ANNs) 6
2.2 Data 7
2.3 Hyperparameter Tuning & Regularisation 7
2.4 Results & discussion 8

A R Code 9
A.1 Part 1 9
A.2 Part 2 13

1 Classifying Spam Emails

The goal of this section of the assignment is to use Support Vector Machines to correctly predict whether an email is spam or not. Different kernels will be explored to find the optimal model, and within each kernel different hyperparameters will also be explored. The classification error (i.e., accuracy) will be used to measure performance across the training and testing error, however other classification metrics will be used to further compare the models based on the test set.

1.1 Support Vector Machines (SVMs)

The objective of Support Vector Machines (SVMs) is to find a hyperplane in a p-dimensional space (p is the number of features) that can classify two classes. Although SVMs can be used for regression problems, it is a popular supervised learning algorithm for classification problems. Before implementing support vector machines (SVMs), its important to mention the many terminologies used when discussing SVMs.

A hyperplane can be defined by the following equation:

$$\beta_0 + \sum_{i}^{n} \beta_i X_i = 0 \tag{1}$$

For certain values of X = $(X1, X2, ..., Xp)^T$, if equation **??** is satisfied, the point lies on the hyperplane[3]. In a 2-dimensional setting, equation 1 is simply the equation of a line, where β_0 & β_1 are parameter estimates. If the left hand side (LHS) of the equation is greater than zero, the point simply lis on one side of the point, and the opposite holds if the LHS is less than the equation[3]. Therefore, the hyperplane is essentially a division in a p-dimensional space. Overall, one can test various hyperplanes to find the optimal hyperplane that *linearly* splits the data perfectly. This method is known as Maxmial-Margin classifier[3].

The maxmial-margin classifier works well when the assumption is that the classes in a category can be split perfectly (i.e., achieve an accuracy score of 100%)[3]. This assumption is quite unrealistic as it is rare to find real-world applications that adhere to this assumption. To relax this assumption, a soft-margin classifier is used in replacement of the maximal-margin classifier. This technique simply lets the algorithm make a certain number of mistakes, but, simultaneously making the margins wide enough for observations to be correctly classified.

Most times, it is inappropriate that classes of the predictor variable can be linearly separable. To 'correct' for this assumption, one may choose to enlarge the feature space using various functions of the predictors (e.g. quadratic, logarithms), but this is not sufficient, hence the use of Support Vector *Machines*. SVMs are simply an extension of SVC, where you can enlarge the feature set using kernels[3]. This report will use three popular kernels: Linear, D^{th}-Degree polynomial, and Radial (Gaussian) kernel. Each kernel has its own hyperparameters (although all of them have a cost hyperparameter), and this will be discussed in later sections of the report.

1.2 Data

The data consists of 4,601 observations, with 57 features (A.1,A.2,...,A.57) and one predictor variable (email or spam). All the features are continuous variables, specifically features A.1 to A.54 are measured in terms of proportions, whilst A.55 feature is average length of uninterrupted sequences of capital letters, and A.56 and A.57 features are integers. This indicates that the features are measured in different scales, hence standardising the feature set is Paramount.

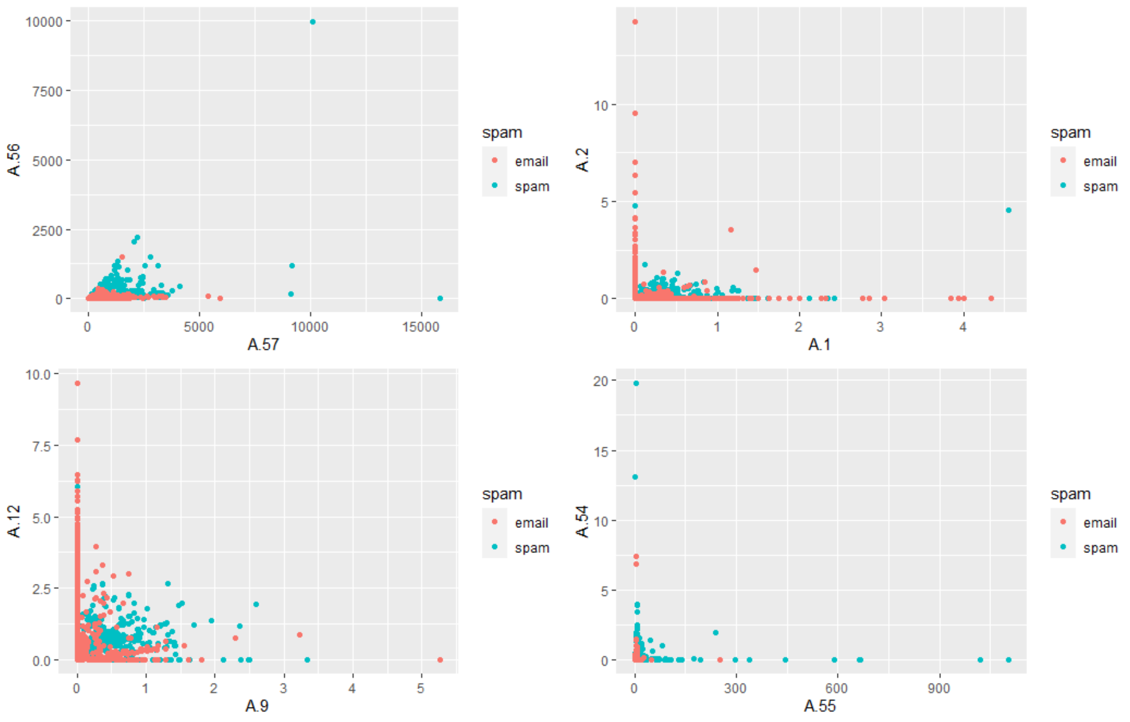

Figure 1: Various feature maps.

Figure 1 looks at 4 feature maps, each map representing two continuous variables in the dataset. From this sample, it is evident that without enlarging the feature set, it will be difficult to improve on existing models, hence the need to explore the various kernels in our disposal.

The data will be split 80% and 20% to create the training and test set respectively.

1.3 Model Techniques

In this section, 3 sets of models are explored with each set relating to a particular kernel: Linear, Polynomial, and Radial. For all the models, a 5-fold cross validation is implemented on the training set.

1.3.1 Linear Kernel

The linear kernel (4 only has one hyperparameter that needs tuning, which is the cost hyperparameter. This is simply the regularisation hyperparameter. According to Hsu et.al [2], they recommend increasing the cost parameter exponentially. Using this technique: the various cost values are used:

$$K(x_i, x_{i'}) =< x_i, x_{i'} >= \sum_{i=1}^{p} x_{ij}x_{i'j} \tag{2}$$

For the purposes of this report the following cost hyperparameter are used: 0.03125, 2.03125, 4.03125, 6.03125 8.03125, 10.03125,12.03125, and 14.03125. Therefore, there are 8 different models.

1.3.2 Radial Kernel

Equation 3 represents the radial kernel. This kernel has two hyperparameters, C (cost parameter) and γ. Similar to the cost hypeparameter, Hsu et.al [2] suggests trying exponentially increasing values when doing a grid search, there for the same cost values will be used from the Linear kernel model, including the following γ values: 0.01,0.1,1,10. γ controls the curvature of the hyperplane. Using the combinations of these two hyperparameters, 32 models are explored.

$$K(x_i, x_{i'}) =< x_i, x_{i'} >= \exp(-\gamma \sum_{i=1}^{p} (x_{ij} - {x_{i'j}}^2)) \tag{3}$$

1.3.3 Polynomial Kernel

The polynomial kernel only makes use of two hyperparameters, cost and degree hyperparameter. Similar to the radial and linear, the same cost values will be used. The following degree values are explored: 1,3,7, and 10. The total number of models these combinations yield is (coincidentally) 32.

$$K(x_i, x_{i'}) =< (1 + x_i, x_{i'})^d >= (1 + \sum_{i=1}^{p} x_{ij}x_{i'j})^d \tag{4}$$

1.4 Results & Discussion

Table 1 resembles the accuracy scores for the best models in each type of kernel set. Looking at the training error, the polynomial kernel outperforms the rest of the models, but on the test set, radial outperforms polynimal kernel, and is marginally better than the linear kernel. From the results, high values of the cost hyperparameter are preferred, whilst for the γ value, 0.01 seems to produce good results.

Kernels	Cost	Gamma	Degree	Train error	Test error
Linear	14.03125			93.183	93.145
Radial	12.03125	0.01		95.953	93.254
Polynomial	6.03125	0.01	3	96.008	92.818

Table 1: Accuracy error of the training and test set for the best models performed in the 3 different sets according to the kernels.

Figure 2 plots the 5-fold cross validation accuracy (classification rate) score of the cost hyperparameter for the linear kernel case. This figure is of interest because from cost values greater than 5, there seems to be a general positive trend indicating that the model has not converged. As a result, the model could be improved by increasing the cost values further.

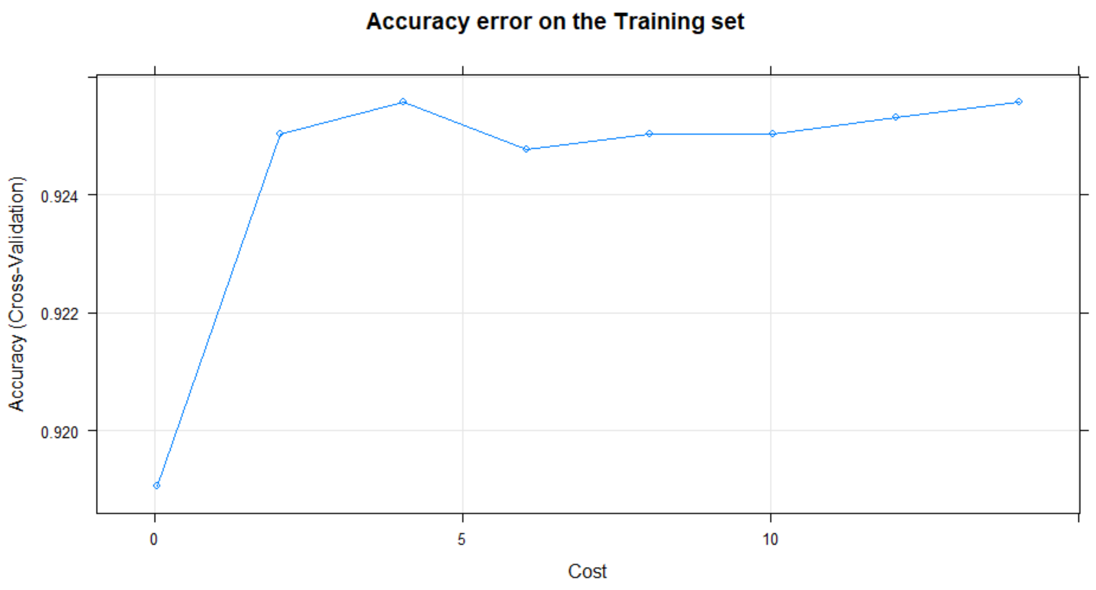

Figure 2: Training accuracy score, across different cost values for the Linear kernel.

Comparing models using only one metric does not give the full picture. Table 2 focuses on different classification metrics on the test set. Recall (true positivity rate) examines how many observations the model correctly classifies as spam. In this case the Linear kernel model marginally performs best. If the detection of spam email is of major importance, then an argument can be made for using a linear kernel. On the other hand if identifying the emails that are indeed not spam (specificity), then the radial model marginally performs better.

Model	**Accuracy**	**Recall**	**Specificity**	**F1**
Linear	93.145	96.050	88.674	94.440
Radial	93.254	95.691	89.503	94.504
Polynomial	92.818	95.870	88.122	94.180

Table 2: Classification metrics on the test set for the different models that performed better than the rest in their respective sets. All values are expressed as *percentages*.

2 Classifying Odd & Even Numbers

For this segment of the assignment, the aim is to build a Feed Forward Artificial Neural Network to the digital dataset, where one has to classify digits as odd or even numbers (i.e., classification problem). By making use of different hyperparameters set in the NN, predictions will be made on the test set.

2.1 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are supervised learning algorithms used for predicting outcomes, depending on whether the output variable is numerical or categorical. For the purposes of this assignment, the focus is on Feed Forward Neural Networks. Since the aim of this section is to develop a NN that's sole purpose is to classify even and odd numbers, the fact that NNs results do not give meaningful interpretations is irrelevant in this setting. In other words, the interest is in the predictive power of this model.

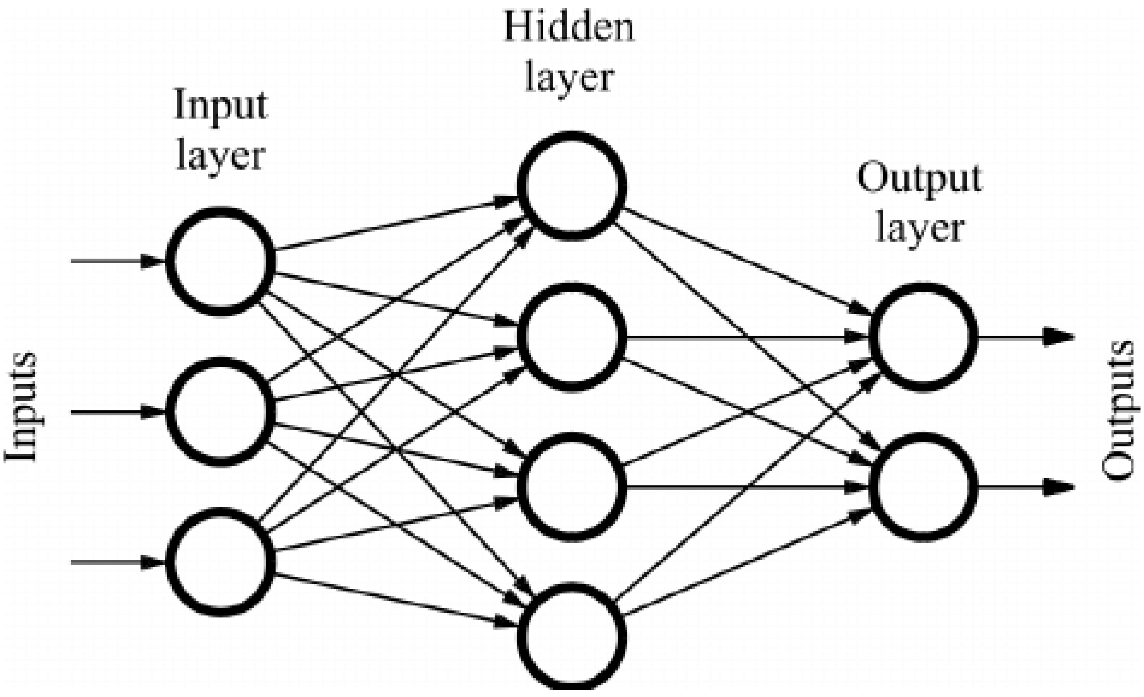

Figure 3: Example of feed-forward NN [4]

Figure 3 is a sample of a feed forward NN. Relating this to our case, the input layers is simply the 28 by 28 grid of a particular number, where the number of inputs equals 784. The output layer in this case is the number of classes in the category variables, hence we have two output layers. The number of hidden layers and neruons in each hidden layer is a hyperparameter that is set before training the model on the data.

2.2 Data

There are two datasets, the training and testing set, where they each contain 2500, and 2001 observations respectively, each containing 784 features, and one output feature, describing whether the digit is odd or even. For the purposes of this assignment, the training set is split where 80% of the data is training, and 20% of the data is validation. Since the feature set represents a 28 by 28 grid, there is no need to scale the features as the features are 'measured' in the same way.

2.3 Hyperparameter Tuning & Regularisation

Feed forward NNs have several hyperparameters which can be tuned. For the purposes of this assignment, various hyperparameters are explored using a grid search.

Probably the most important hyperparameters are the number of hidden layers, and how many neurons/nodes to be used in each hyperparameter. In terms of the number of hidden layers, 1 or 2 is sufficient. It is highly unlikely that the data can be linearly separated therefore this report makes use of 1 and 2 hidden layers when doing grid search. According to Heaton, J[1], there are several rules of thumb that can be followed. Number of neurons in a hidden layer can be between the number of input and output layers (i.e., between 2 and 784), or it is two-thirds of the size of the input layer (i.e., approximately 522 neurons). Lastly, Heaton, J[1] suggests that the number of neurons should be less than twice the size of the input layer (i.e.,1568 neurons). Taking these considerations accounts, the following represent the different architectures are constructed: (522), (50), (150), (350), (50,150), and (522,270).

Activation functions in NNs speify how the weighted sum of the inputs are transformed to outputs. Since this assignment makes use of the *h20* R package, two activation functions are used: Rectifer Linear and Tanh. The learning rate (α) is also of great interest as this specifies the speed at which the algorithm learns. Low values of α means the algorithm learns slowly, whilst larger values of α indicates the algorithm learns faster, but is prone to overlook the optimal point. To cater for this 3 α values are used: 0.005, 0.5, and 0.1.

Controlling the complexity of the model is important, because if the model is too complex, you run the risk of overfitting, but if it is to simple, then under-fitting is likely to occur. To control for this, L2 (Ridge regularisation) is used. Whilst training the model, a sequence of L2 values are used to control the complexity. Input dropout rate is usually used when regularization is used. The input dropout rate is the probability that a node will be trained in a given layer. Values of 0.5-0.8 as dropout rates are usually suitable, but due to the complexity of the model, input dropout rate is set at 0.5

Other settings this report has taken into account is setting the stopping criteria to misclasiification, stopping tolerance of 1%, stopping rounds amounting to 2; that is if there is no improvement in the scoring metric for 2 rounds, the algorithm comes to an end. A 5-fold cross validation is set and the number of epochs is set to 10.

2.4 Results & discussion

Table 3 ranks the perfromances of the 10 models produced by the algorithm in terms of a 5-fold cross validation log loss metric, where the best model had a score of 0.217. The Rectifer seems to be the appropriate activation function, but Tanh activation function could also be used as it performs well as it is the 2nd to 5th best model. In terms of architecture, a 2 hidden layer model out performs the other variation of the model, implying that a complex model was indeed needed for this problem.

Activation Function	**Architecture**	**L2 (λ)**	α	**CV Log Loss**
RectifierWithDropout	[522, 270]	1.4E-5	0.1	0.217
TanhWithDropout	[50]	6.7E-5	0.1	0.336
TanhWithDropout	[150]	4.7E-5	0.1	0.340
TanhWithDropout	[522]	1.1E-5	0.5	0.340
TanhWithDropout	[50, 150]	7.0E-5	0.005	0.341
RectifierWithDropout	[150]	5.2E-5	0.5	0.341
RectifierWithDropout	[350]	8.9E-5	0.1	0.343
RectifierWithDropout	[350]	6.2E-5	0.1	0.388
TanhWithDropout	[522, 270]	8.9E-5	0.005	0.477
TanhWithDropout	[522]	0.5 8.4E-5	0.1	0.596

Table 3: NN algorithms ranked according to 5 fold cross validated Log Loss.

References

[1] J Heaton. Heaton research: The number of hidden layers. *Availablefrom: https://www. heatonresearch. com/2017/06/01/hidden-layers. html [13/3/2019]*, 2017.

[2] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. A practical guide to support vector classification, 2003.

[3] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning.* Springer, 2013.

[4] Ramon Quiza and J. Davim. *Computational Methods and Optimization*, pages 177–208. 01 2011.