A Comparative Analysis of Multilayer Perceptrons and Support Vector Machines in Predicting Credit Card Defaults

Axil Sudra
Axil.Sudra@city.ac.uk

Abdirahman Khanog
Abdirahman.Khanog@city.ac.uk

Introduction

Description and Motivation of the Problem

Financial institutions around the world face the challenging task of predicting credit card defaults in active loan portfolios. In the past, the inadequate process and eligibility requirements for acquiring a credit card have led to an increase in default rates and in turn raised concerns with financial regulators. More recently, through intervention, financial institutions have deployed superior measures to deter unqualified applicants gaining access to credit cards and thus attempted to reduce overall default rates. However, even with reasonable checks in operation credit card defaults are still prevalent; according to a credit conditions survey conducted by the Bank of England [1], default rates on credit cards increased significantly in the second quarter of 2018. Whilst credit card defaults are always going to be present within financial institutions, identifying potential defaulting clients through data-driven prediction could substantially reduce default rates.

This paper aims to critically evaluate two data-driven models designed to identify whether credit card clients will default on their loans through pattern recognition. Specifically, the models considered are a feedforward Multilayer Perceptron (MLP) and a Support Vector Machine (SVM). We will inspect various configurations of the two models and determine the most accurate in prediction.

Hypothesis Statement

A research paper by Zanaty [2] comparing MLP and SVM classifiers reported that SVMs generally outperform MLPs on high dimensional datasets. Based on this analysis and given the complexity of our chosen dataset, our hypothesis is that the SVM will outperform the MLP.

Dataset

Description

The dataset used to analyze and evaluate the two models is based on consumer (client) credit card payments data which was originally sourced from an unnamed Taiwan bank [3] and is available from the UCI Machine Learning Repository [4]. This dataset consists of 30000 samples with 23 features of which 9 are categorical and 14 are numerical; these are shown in Table 1 with a statistical summary.

Feature Name	Scaling	Mean	SD
LimitBalance	Numerical	167480.00	129750.00
Gender	Categorical	N/A	N/A
Education	Categorical	N/A	N/A
MaritalStatus	Categorical	N/A	N/A
Age	Numerical	35.49	9.22
PaymentSep	Categorical	N/A	N/A
PaymentAug	Categorical	N/A	N/A
PaymentJul	Categorical	N/A	N/A
PaymentJun	Categorical	N/A	N/A
PaymentMay	Categorical	N/A	N/A
PaymentApr	Categorical	N/A	N/A
BillAmountSep	Numerical	51223.00	73636.00
...	...	...	...

Feature Name	Scaling	Mean	SD
...	...	...	...
BillAmountAug	Numerical	49179.00	71174.00
BillAmountJul	Numerical	47013.00	69349.00
BillAmountJun	Numerical	43263.00	64333.00
BillAmountMay	Numerical	40311.00	60797.00
BillAmountApr	Numerical	38872.00	59554.00
PaymentAmountSep	Numerical	5663.60	16563.00
PaymentAmountAug	Numerical	5921.20	23041.00
PaymentAmountJul	Numerical	5225.70	17607.00
PaymentAmountJun	Numerical	4826.10	15666.00
PaymentAmountMay	Numerical	4799.40	15278.00
PaymentAmountApr	Numerical	5215.50	17777.00

Table 1: Feature Scaling and Statistical Summary (2.d.p)

The target variable of the dataset indicates whether a client will default on their credit card payment next month or not; 0 corresponds to ‘no default’ and 1 corresponds to ‘default’. Through observation, we identified a class imbalance of 23364 (77.88%) ‘no default’ samples to 6636 (22.12%) ‘default’ samples.

Exploratory Data Analysis

To uncover any notable hidden details, we completed a brief exploratory data analysis of the dataset. Figure 1 shows a pearson correlation coefficient heatmap of the features (left) and a series of histograms highlighting the distribution of (credit) limit balance between the original data and split between the two classes (right). It can be observed that many of the features in the dataset have minimal correlation with each other; features of payment status from April to September have the highest positive correlation between each other. This would be expected as each month is dependent on the previous month (i.e. if a client fails to keep up with payments for one month, there is a high probability that they will fail in subsequent months). Note that this positive correlation could possibly influence the decisions made by the MPL or/and the SVM in classifying samples. Some key features of the series of histograms in Figure 1 includes the observation that clients that default on their credit card payments are granted a smaller limit balance than those who do not default. Furthermore, the distribution of the ’no default’ histogram compared to that of the original data appears to be identical; this shows the magnitude of the class imbalance in the dataset.

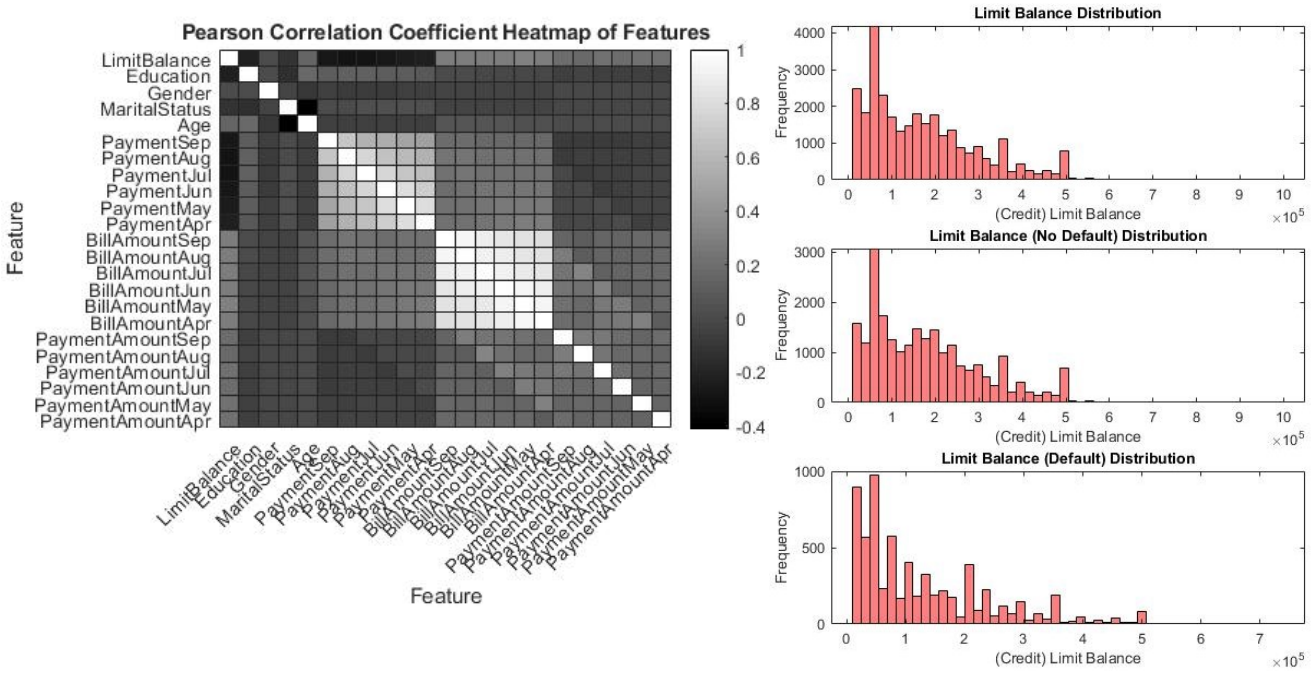

Figure 1: Left: Correlation Heatmap of Feature | Right: Histograms of Limit Balance

Data Preprocessing

Before training the models on the data, we completed a few preprocessing steps. It was observed that two categorical features (Education and Marital Status) contained unidentified levels; for simplicity, these were grouped into the level corresponding to ‘other’. Next, we observed that the scale of some features were much higher than other features (which can also be observed from Table 1). Therefore, the features were normalized using a standard score (z-score) to give them a mean of 0 and standard deviation of 1. As a final step, we decided to address the critical class imbalance present in the dataset. As the original dataset is currently quite large, we thought a sampling method such as SMOTE (Synthetic Minority Over-Sampling Technique) [5] would be unsuitable; instead, we randomly under-sampled the majority class (‘no default’). Thus, our sampled dataset contained 13272 samples of which 50% corresponds to ‘no default’ and 50% corresponds to ‘default’.

Summary of Neural Network Models

Multilayer Perceptron (MLP)

The MLP is a supervised feedforward artificial neural network that consists of a set of sensory units (source nodes) that constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes [6]. Each node, excluding nodes contained in the input layer, represents a neuron and each layer (of neurons) in the MLP is connected through adjustable 'synaptic weights'. Each neuron is formed of a linear adder function and a nonlinear activation function. The linear adder function sums the input signals which are weighted by respective synaptic connections and the activation function limits the amplitude of the output of the neuron [7].

A popular method for training MLPs is the back-propagation algorithm [7]. Training usually consists of two phases. In the first phase, known as the forward phase, the synaptic weights of the network are fixed (initialized to random values during the first forward pass) and the input signal is propagated through each layer of the network until it reaches the output neurons. In the second phase, known as the backward phase, an error signal is computed through comparison of the network's output and the desired output. This error signal is then propagated through each layer in the backward direction, in which the partial derivatives of some loss function with respect to the synaptic weights are calculated; these are then used to adjust the values of the synaptic weights through a optimization method known as gradient descent [8]. These two phases are repeated until some loss function has been minimized to a specified criteria or a certain amount of passes through the network have been completed. Note that each pass through the network is known as an epoch.

Support Vector Machines (SVM)

The SVM is a powerful supervised algorithm used mostly for binary classification tasks (although multi-class extensions have been proposed [9]) and to a lesser extent regression tasks. A SVM constructs a hyperplane with n-dimensions as a decision boundary in such as a way that the margin distance of separation between data points from one class and data points from another class is maximized [7]. Note that support vectors are defined to be the boundary on data points closest to the hyperplane; the distance between this boundary and the hyperplane is the margin.

An important aspect of the SVM is that it is based on the 'structural minimization principle' whereas the MLP is based on the 'empirical risk minimization principle' [10]. The structural risk minimization principle attempts to minimize an upper bound on the expected risk whilst empirical risk minimization attempts to minimize the error on some training data [11]. For this reason, the SVM can generalize more efficiently (and successfully) than the MLP.

Pros and Cons

	MLP	SVM
Pros	Once trained, prediction speed of MLPs is very quick.	Memory efficiency whilst training is very good as SVMs use a few data points to compute the decision boundary.
	The nonlinear nature of MLPs (i.e. activation functions within neurons) makes them ideal to model on many real-life problems.	Useful for linearly separable data and non-lineraly separable data (with the use of the 'kernel trick').
Cons	It is computationally expensive and time consuming to train MLPs due to the large amount of hyper-parameters that need to be tuned.	The performance of SVMs can be drastically reduced if datasets contain many 'noisy' data points.
	A gradient descent optimization method can lead to MLPs being stuck in local minima and failing to locate the global maximum; therefore, the synaptic weight values might not be optimal.	Training SVMs can be quite slow for certain kernel functions and large datasets.

Table 2: Pros and Cons of the Models

Training and Evaluation Methodology

Note that implementation of the MLP and SVM was completed using MATLAB R2018b. The 'Deep Learning Toolbox' [12] was used for the MLP; SVM models were coded from scratch.

Training Methodology

The training methodology consisted of dividing our sampled dataset (features and target variable) into training and test set, locating the best hyper-parameters for model selection through the use of grid searches and lastly retraining each model on those hyper-parameter values. As our sampled dataset was fairly large (containing 13272 samples), we decided to hold out 25% for testing and the remaining 75% for training.

The model selection process entailed a grid search to locate the best hyper-parameters for both MLP and SVM models. The training set passed through the MLP and SVM grid searches was further divided into training and validation data through 10-fold stratified cross validation; this process involved training each combination of model hyper-parameters on 10 training and validation sets of equal sample size. According to Ramezan *et al*. [13] stratified-statistical-based sampling methods are known to generate the highest classification accuracy most of the time.

MLP Architecture and Evaluation Methodology

During the training phase, a 'gradient descent with momentum and adaptive learning rate backpropagation' function was used to update the synaptic weights and bias values of each neuron in the MLP. This function adjusts the learning rate accordingly once the derivatives of some loss function is calculated with respect to the synaptic weight values after each epoch [14]; if the loss decreases towards the network error goal, the learning rate is increased to speed up the learning process. Furthermore, with this function a custom momentum coefficient can be specified for training.

We set the maximum number of epochs to 500 and the error goal to 0.001, as a early stopping criteria. For the grid search, we decided to vary the hyper-parameters for values displayed in Table 3. The choice of hyper-parameter values intended to model greater complexities in the training set, improve generalization, increase the learning speed of the MLP and avoid the 'con' of being stuck in a local minima for gradient descent computations respectively [15]. The evaluation metric used to assess the hyper-parameters in the grid search was the cross entropy (average of 10-fold cross validation), as this is suitable for binary classification tasks. Confusion matrices were computed during retraining and testing of the optimal MLP model to assess the accuracy, recall, precision and F1-score measures.

SVM Architecture and Evaluation Methodology

SVMs do not have many architectural/training parameters as MLPs. The hyper-parameters that were chosen to be tuned are displayed in Table 3. To being, a grid search was used to source the optimal kernel function for the SVM, and in the case of a 'polynomial' kernel, various orders were included in the search. Due to restraints on computational power and time, the box constraint and kernel scale were tuned using a random search [16] with 15 iterations. As 10-fold cross validation was used in the training phase, the evaluation metric used to assess the performance of each model in the grid search was a 'kfoldLoss' error function. Just as with the MLP model, confusion matrices were computed during retraining and testing of the optimal SVM model to assess the accuracy, recall, precision and F1-score measures.

MLP	SVM
• Size of hidden layers = [10, 20, 30, 40, 50] • Number of hidden layers = [1, 2, 3, 4, 5] • Learning rate = [0.05, 0.1, 0.3, 0.6, 0.9] • Momentum = [0.1, 0.3, 0.5, 0.7, 0.9]	• Kernel functions = ['rbf', 'linear', 'polynomial'] • Order (only polynomial kernel) = [2, 3, 4, 5, 6] • Box constraint (random search) = range[0.001, 1] • Kernel scale (random search) = range[0.001, 1]

Table 3: Hyper-parameter Tuning

Choice of Parameters and Experimental Results

Choice of Parameters

As a result of the MLP grid search, we identified that the best model has the hyper-parameter values displayed in Table 4. The average 10-fold cross validation cross entropy of these hyper-parameter values was 0.19883. Through re-running the grid search multiple times, We noticed that the results were highly influenced by the network's random synaptic weight initialization process. Therefore, it was impossible to reproduce exact results. The best SVM model's hyper-parameter values are also displayed in Table 4. Through the grid search, we noticed that a 'polynomial' kernel function with order 2 achieved the smallest 'kfoldLoss' error at 0.30510. Surprisingly, the box constraint and kernel scale values obtained during the random search produced a worst 'kfoldLoss' classification error than the default values. Therefore, we used the SVM's default box constraint and kernel scale values.

Best MLP Model	Best SVM Model
• Size of hidden layers = 10 • Number of hidden layers = 3 • Learning rate = 0.3 • Momentum = 0.9	• Kernel function = 'polynomial' • Polynomial Order = 2 • Box constraint (random search) = 1 • Kernel scale (random search) = 1

Table 4: Optimal Hyper-parameters

Experimental Results

On successfully identifying the best hyper-parameters, we retrained each model to obtain results for the evaluation metrics mentioned above on the training set. The MLP model achieved a cross entropy value of 0.19907 once retrained; as previously mentioned the difference in value between the grid search and retraining is most likely due to the network's random synaptic weight initialization process. Furthermore, the MLP attained an overall training accuracy of 69.51%. A possible reason for the low accuracy could be explained by the high complexity of the training set. As the SVM does not have any random initialization processes, it achieved a 'KfoldLoss' error of 0.3051, which was identical to the result obtained during the grid search. In terms of the accuracy rate, it managed to slightly outperform the MLP model gaining 70.99%. Whilst the SVM model performed better that the MLP model on most of the evaluation metrics when retraining, including recall (SVM = 0.59247 against MLP = 0.53376) and F1 score (SVM = 0.67189 against MLP = 0.63709), the MLP achieved a higher precision the SVM (MLP = 0.79004 against SVM = 0.77591).

Both models performed equally well on the test set. Figure 2 shows testing confusion matrices (left) and ROC plots (right) for MLP and SVM. From the confusion matrices, we noticed that the SVM only mislabeled 29.02% of the test set samples whilst the MLP mislabeled 29.29%. Therefore, the accuracy rate of the SVM and MLP was 70.98% and 70.71% respectively. The recall served as a key measure during testing as it highlighted the severity of misclassification of defaulting clients; the SVM model achieved a total recall of 0.59088 compared to the MLP model's 0.55198. Thus, the SVM could be said to be more reliable than the MLP. Through inspection of the ROC plots for each model, we calculated the AUC for both classes. In respect to 'no defaulting' clients the SVM's AUC was 0.75132 and the MLP's was 0.75225; for 'defaulting' clients, the SVM astonishingly achieved an identical AUC to its 'no defaulting' clients AUC and the MLP achieved 0.75088.

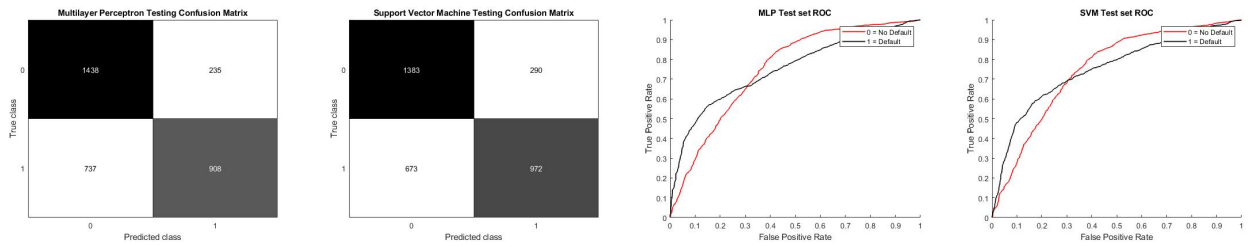

Figure 2: Left: MLP and SVM Test Set Confusion Matrices | Right: MLP and SVM Test Set ROC Plots

Analysis and Critical Evaluation of Results

Despite expecting marginally higher results from each model, both MLP and SVM performed to a satisfactory level in determining which credit card clients are likely to default on their loan payments or not. Through the use of grid searches (and a random search in the case of some hyper-parameters for the SVM) we were able to locate the best models to deploy on our test set. However, the process of hyper-parameter tuning for both models was immensely time consuming taking approximately 28 hours and 42 minutes from start to finish. As the MLP had the most architectural/training parameters, it's grid search had taken the most time. The best obtained hyper-parameter values from the grid searches (displayed in Table 4) proved to be effective, although the random search for the SVM's box constraint and kernel scale did not; the default values for the box constraint and kernel scale managed to minimize the 'KfoldLoss' error somewhat more than the values proposed by the random search, although only 15 iterations were computed during the random search so increasing this value could have resulted in a improved loss rate. After sourcing the best hyper-parameters for each model, we retrained both MLP and SVM again. The training time for the SVM proved to much slower than the MLP, although performance-wise it surpassed the MLP in terms of model accuracy, recall and F1 score. This was not surprising given the complexity and high dimensional nature of the overall dataset. By deploying the retrained models on the test set, the SVM demonstrated to be superior in terms of accuracy and efficiency once again. We suppose that the SVM's capabilities are more suited towards higher dimensional data than the MLP and therefore, gives it the edge in this case. As mentioned previously, we paid significant interest to the recall measure of each model during testing on the test set; a low recall rate on this problem and similar problems would result in dire financial consequences for financial institutions/lenders. Through the testing ROC plots and the recall measures, we identified that the SVM was more consistent with predictions than the MLP. A limitation of our study was that we were not able to test different MLP backpropagation algorithms [17] in the network's grid search which might have provided a higher performance compared to the SVM.

Conclusions and Future Work

The aim of this study was to compare and contrast the performance of a Multilayer Percpetron (MLP) and Support Vector Machine (SVM) in predicting whether credit card clients would default on their loan payments in the near future. Throughout our analysis, we identified the significant sophistication of both models' architecture, their pros and cons and evaluative performance. We learned that each model requires a considerable amount of computational power and time to optimize to a good degree. In regards to results, the SVM outperformed the MLP during training and testing which was expected as stated in our hypothesis; metrics, such as recall, calculated from confusion matrices and ROC plots proved to be powerful evaluation measures. Future work could include limiting the dataset to certain features through feature selection or using dimensionality reduction to analyze any improvements in performance of the MLP and SVM. Furthermore, investigating unsupervised network configurations, such as Self Organizing Maps [18], could potentially prove to be more useful than our current supervised learning approach at separating not defaulting and default credit card clients.

References

[1] *Credit Conditions Survey - 2018 Q2* (2018) Available at: https://www.bankofengland.co.uk/credit-conditions-survey/2018/2018-q2 (Accessed: March 2019).
[2] Zanaty, E.A. (2012) 'Support Vector Machines (SVMs) versus Multilayer Perceptron (MLP) in Data Classification', *Egyptian Informatics Journal*, 13(3), pp. 177-183.
[3] Yeh, I.C., Lien, C.H. (2009) 'The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients', *Expert Systems with Applications*, 36(2), pp. 2473-2480.
[4] Yeh, I.C. Chung Hua University (2016) 'Default of Credit Card Clients Data Set', Available at: https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients# (Accessed: March 2019).
[5] Chawla, V.N. *et al.* (2002) 'SMOTE: synthetic minority over-sampling technique', *Journal of Artificial Intelligence Research*, 16(1), pp. 321-357.
[6] Haykin, S.S. (1999) *Neural Networks: A Comprehensive Foundation (Second Edition)*. Upper Saddle River, New Jersey: Prentice Hall.
[7] Haykin, S.S. (2009) *Neural Networks and Learning Machines (Third Edition)*. Upper Saddle River, New Jersey: Pearson Education.
[8] Takase, T. *et al.* (2018) 'Effective neural network training with adaptive learning rate based on training loss', *Neural Networks*, 100(1), pp. 68-78.
[9] Rogers, S. and Girolami, M. (2016) *A First Course in Machine Learning (Second Edition)*. New York: Chapman and Hall/CRC.
[10] Burges, C.J.C. (1998) 'A Tutorial on Support Vector Machines for Pattern Recognition', *Data Mining and Knowledge Discovery*, 2(2), pp. 121-167.
[11] Vapnik, V. *et al.* (1996) 'Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing', *International Conference on Neural Information Processing Systems*, 9(1), pp. 281-287.
[12] *Deep Learning Toolbox Documentation* (2019) Available at: https://uk.mathworks.com/help/deeplearning/ (Accessed: March 2019).
[13] Ramezan, C.A., Warner, T.A. and Maxwell A. (2019) 'Evaluation of Sampling and Cross-Validation Tuning Strategies for Regional-Scale Machine Learning Classification', *Remote Sensing*, 11(2), pp. 185-206.
[14] *Gradient descent with momentum and adaptive learning rate backpropagation: traingdx Documentation* (2019) Available at: https://uk.mathworks.com/help/deeplearning/ref/traingdx.html#f8-671906 (Accessed: March 2019).
[15] Montavon, G., Orr, G.B. and Müller, K.R (2012) *Neural Networks: Tricks of the Trade (Second Edition)*. London: Springer.
[16] Bergstra, J., and Bengio, Y. (2012) 'Random Search for Hyper-parameter Optimization', *Journal of Machine Learning*, 13(1), pp. 281-305.
[17] Wani, S.M.A. (2013) 'Comparative Study of Back Propagation Learning Algorithms for Neural Networks', *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(12), PP. 1151-1156.
[18] Kohonen, T. (1990) 'The Self-organizing Map', *Proceedings of the IEEE*, 78(9), pp. 1464-1480.