Comparison of two neural network models (MLP and SVM) for classification using U.C.I. Car Evaluation data set

Shengqiang Fan & Harry Li

1. Introduction

When people tend to purchase a new car, lots of aspects play an important role to influence people's decision, such as car price, appearance, safety, fuel consumption etc. A hierarchical decision model was developed by Bohanec and Rajkovic [1], which was aimed for dividing all aspects of car hierarchically based on its properties. Car evaluation dataset [2] is collected and created based on Bohanec and Rajkovic's model, which is a good dataset for us to learn the relationship between different aspects and different level of acceptable from buyers. Neural network models applied to this data can provide clear commercial direction and competition-benefit to manufacturer's since desirable features can be embedded in the future design process. Two models, multilayer perceptron (MLP) and support vector machine (SVM) were chosen for this particular classification problem. The performance of MLP and SVM will be analyzed and compared in this report.

2. Dataset

The dataset has 1,728 observations, including 6 categorical attributes with class labels (unacc, acc, good, vgood). No missing value was obtained in this dataset. The distribution of all 4 classes of cars is summarized in table below, which shows that majority of the cars are classified as unacceptable and very few cars are good or very good. This skewed dataset may affect our machine learning models.

Table 1: Summarized distribution of 4 classes of cars

Class	Count	Count (%)
unacceptable	1210	70.023 %
acceptable	384	22.222 %
good	69	3.993 %
very good	65	3.762 %

There are 6 variables in the original dataset, which are buying, doors, lug boot, maint, persons and safety related to different car features. During data exploration, 'predictorImportance' function was used to estimate the importance of features. It can be seen in figure 1 that safety and number of persons are highly important in car evaluation. Without further feature engineering the importance of the various car sub-attributes unfortunately remain hidden. Therefore, 21 new feature variables were created from the sub-attributes of the initial 6 features by using one-hot-encode. Each of the new attributes is a binary variable for the 1728 observations. Similarly, the importance of different features to the classification outcome for the car was calculated as shown in figure 2. It complements the first chart in that it highlights the underlying sub-attributes which make a high-level feature important. For example, we observe that the car being 'highly safe', having capacity for 2 persons, and maintenance are first consideration when people purchasing a car. In addition, we also tend to see the support vector machine's performance on dealing with different dimension of features.

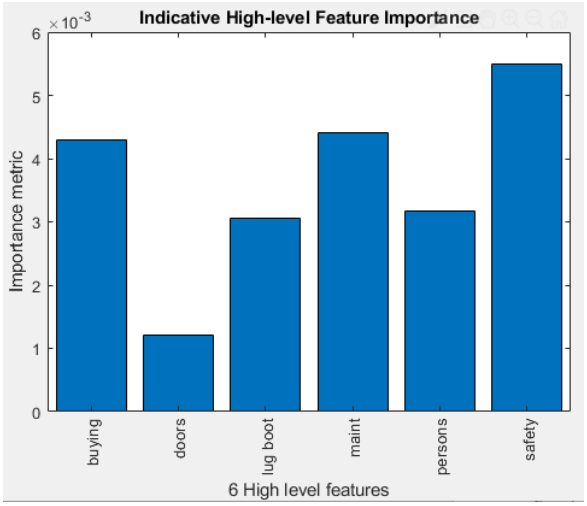

Fig 1: 6variables Feature Importance

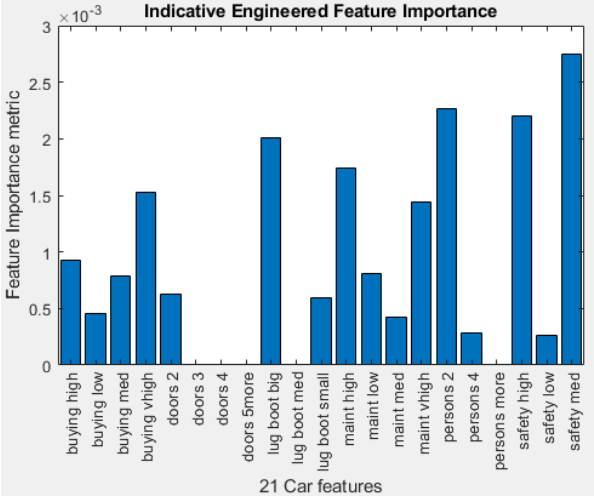

Fig 2: 21variables Feature Importance

3. Algorithms

3.1. Multilayer Perceptron (MLP)

A multilayer perceptron is a class of feedforward artificial neural network, which has three essentials elements named input layer, hidden layer and output layer as shown in figure 3. All nodes/neurons are fully connected associated with weights and bias. Each neuron uses a nonlinear activation function except for input nodes, common activation functions are sigmoid/tanh/ReLu/softmax etc. Input data pass though model by multiply with weights and add bias at every layer and find the calculated output of the model, which is called forward pass. Then, backpropagation is introduced for training, which is a supervised learning technique proposed by Rumelhart et al [3]. In backpropagation, error/loss calculated from forward pass and then propagates to update weights and bias by using gradient. In a conclusion, there are three basic steps for training an MLP model, which are forward pass, error/loss calculation and backpropagation.

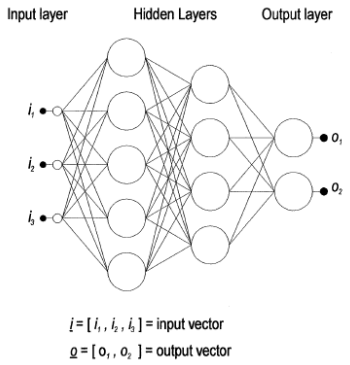

Fig 3: An illustration of multilayer perceptron (two hidden layers)

Advantages:

- it has the ability to learn and model non-linear and complex relationships
- it is very simple to implement with small number of layers
- it is a benchmark classifier in many pattern recognition task
- it has ability of parallel processing to deal with large dataset

Disadvantages:

- there are many hyperparameter to be tuned, and even small change of parameter has large impact on the performance of MLP model
- lack of explanation if there are many neurons or layers
- may run into local minimum (saddle point) when dealing with complex nonlinear problem
- no standard guidance for net size (number of neurons and number of layers)

3.2. Support Vector Machine (SVM)

Support vector machine (SVM) is a supervised learning algorithm that can be used for both classification and regression tasks (Pontil [4]). SVM classifies by defining an optimal hyperplane. When there are serval hyperplanes that classifies the data, it then finds the hyperplane that maximizes the margin between nearest data points from either class as shown in figure 4. SVM has a unique technique called the kernel trick. It is used for non-linear separable problems. We transform the dataset into a higher dimension, and then the problem become separable. Initially, SVM is used for binary classification. However, we can extend the algorithm to multi-class task by converting it to several binary problems, which is demonstrated in this project.

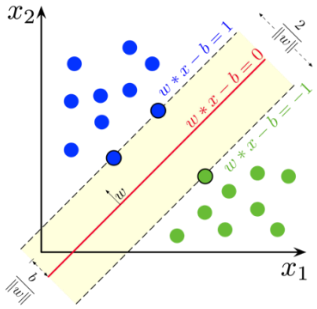

Fig 4: Maximum-margin hyperplane and margins for an SVM model

Advantages:

- it is robust to outliers.
- it is effective in high dimensional spaces.
- it can be more efficient than other algorithms because it uses a subset of the training set (Pontil [4]).

- Once a hyperplane is found, most of the data other than the support vectors become redundant. This means that small changes to data cannot greatly affect the hyperplane. Therefore, SVMs tend to generalize very well for small amount of data, compared with other models such as the MLP (Moro [5]).

Disadvantages:

- it can be computationally expensive for large datasets, because it would take time to find the support vectors. Whereas some of the simple models such as Naïve Bayes only require a very small amount of training time and the results can be surprisingly accurate.
- it doesn't perform well for data with lots of noises, which would cause to target classes to overlap
- SVM share a common disadvantage with other non-parametric techniques, that is it is lack of transparency of results, hence the model maybe difficult to interpret. Whereas algorithms such as decision trees are easy to interpret (Moro [5]).

4. Hypothesis Statement

i. We expect both models produce a great accuracy rates, i.e. close to 100%. As deep learning algorithms are very powerful, and we only have a relatively small amount of dataset.
ii. We expect that the classification accuracy of the respective models can be improved via hyper-parameter tuning, which can be completed prior to the learning process for both implementations.
iii. We expect SVM can outperform MLP, as SVM is generally considered as a more powerful and easier model as stated by Collobert and Bengio [6]. In particular, we are working on a 21-dimensional dataset, which may give SVM an advantage.

5. Choice of training and evaluation methodology

Both the MLP and SVM models are trained with 70% of the data and tested by exposure to the remaining 30% of data, which remains unseen until the point of testing. Both models load the same mat file containing identical train/test data sets, generated by running the '~splitter' .m-file, in order to make a justice comparison between two models. As an extension to the general models, we attempt to tune model specific hyper-parameters and improve classification results. We will establish if the tuned models offer up any improvement in classification performance and demonstrate findings and results graphically.

In MLP, three-layer MLP is chosen for this task. Specifically, one input layer, one hidden layer and one output layer. Activation function of hidden layer and output layer are chosen as sigmoid function and root mean square error is defined as loss function. Mini-batch gradient descent with momentum method is used to minimize the loss function.

In SVM, we aim to minimize an objective function against with the number of objective functions. Our objective function is to the 5-fold cross-validation error via hyperparameter optimization. For final results, we compare the testing accuracy and use confusion matrix to evaluate the performance of two methods.

6. Choice of parameters and experimental results

6.1 Parameters and results of MLP

Learning rate, number of neurons in hidden layer, momentum and mini-batch size are considered as hyperparameters, where momentum value is normally recommended to set as 0.9. From different combination of hyperparameter as learning rate among [0.01, 0.1, 0.2], number of neurons in hidden layer among [40, 50, 60 ,70], mini-batch size among [32, 64, 128], the best parameters combination is learning rate=0.1, number of neurons in hidden layer=50, momentum=0.9, mini-batch size=32. Figure 5 shows an illustration of example training progress, which was created by changing different combination of number of neurons and learning rate. It can be concluded that learning rate = 0.01 is not efficient enough to achieve global minimum compared to learning rate value of 0.1 and 0.2. The training loss is reduced rapidly before 10 epochs and gradually decreased with increased of epoch.

Based on best tuned hyperparameters (learning rate=0.1, number of neurons in hidden layer=50, momentum=0.9, mini-batch size=32), the performance is evaluated using test set as shown in figure 6. It can be calculated that test accuracy is 99.6%, which means only 2 prediction is not correct. In fact, 2 incorrect prediction are both classified to 'good' by MLP classifier but should belong to 'acceptable' class.

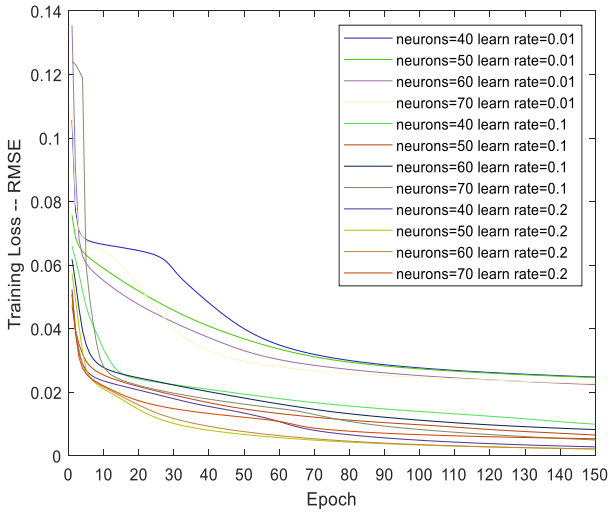

Fig 5: Illustration of MLP training process

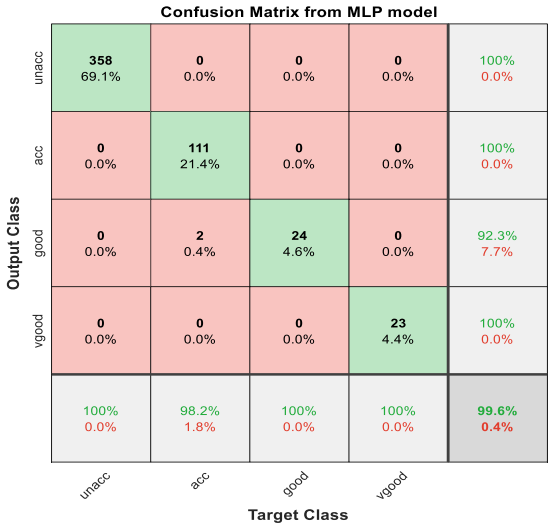

Fig 6: Confusion matrix of MLP

6.2 Parameters and results of SVM

We apply the MATLAB inbuilt function ‘fitcecoc’ to train our multiclass SVM model. Our model uses the name-value argument to optimize the parameters by setting ‘OptimizeHyperparameters’ to ‘all’. This means that Matlab will optimize all eligible parameters. So that we have control and understanding of the model. We search in both 'BoxConstraint' and 'KernelScale', which are positive values log-scaled in the range [10^-3,10^3], as well as ‘KernelFunction’ (‘Gaussian’, ‘linear’, or ‘polynomial’) and ‘PolynomialOrder’ (a positive integer). To study the effect of hyperparameters optimization, we compare our tuned model with an untuned default SVM model (fitcecoc). ‘fitcecoc’ uses K(K – 1)/2 binary support vector machine (SVM) models using the one-versus-one model (fitcsvm), where K is the number of unique class labels (K=4). ‘fitcsvm’ uses a linear kernel function and uses iterative single data algorithm.

In our Hyperparameter optimization options, the acquisition function name is set to ‘expected-improvement-plus’ to ensure that the algorithm is not overexploiting an area, thus we can escape from a local objective function minimum. This method is suggested by Bull [7]. If a point is overexploited, then the acquisition functions modified its kernel function by multiplying its kernel parameters θ by the number of iterations. It then generates a new point based on the new fitted kernel function. If the new point is again overexploiting, the acquisition function multiplies θ by an additional factor of 10 and tries again and it continues in this way up to 5 times. Once we found the new point that is not overexploiting, the algorithm accepts it as the next point. Lastly, the algorithm is iterated 30 times to search for the best estimated feasible point. Figure 7 shows minimum objectives against the number of evaluations to understand the performance the model. It is obvious that our observed objective converged with the estimated objective very quickly.

The turned model found the best observed feasible point as box constrain 87.592, kernel function polynomial, and polynomial order 4. Our tuned model has an excellent testing results with 99.4% accuracy rate. In fact, it only classified 3 data points incorrected (see figure 8 below). Meanwhile, confusion matrix from SVM classifier (only using 6 variable) is also plotted as shown in figure 9. The accuracy is 97.5% with 13 incorrected predictions.

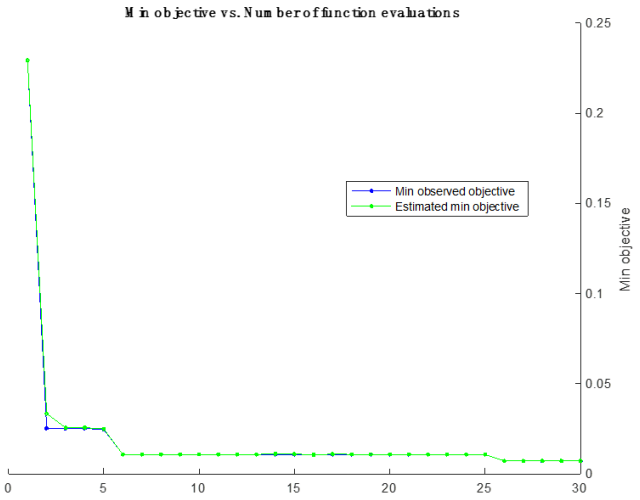

Fig 7: Objective function vs number of function evaltions

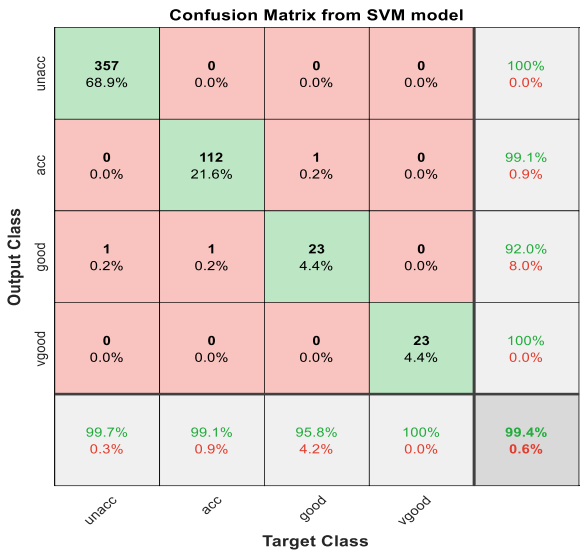

Fig 8: confusion matrix with 21variables

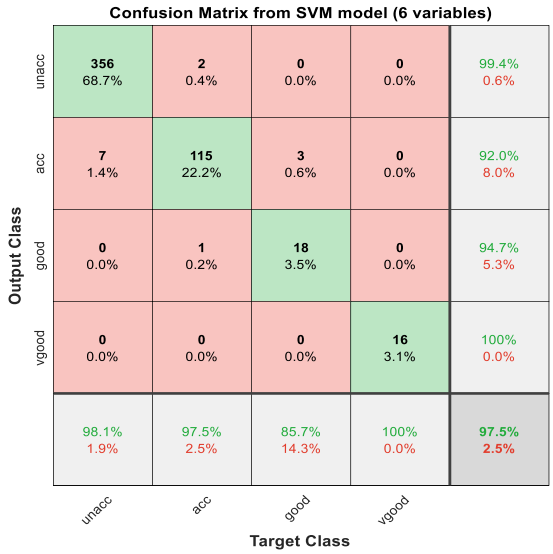

Fig 9: confusion matrix with 6 variables

7. Analysis and critical evaluation

Initially, we split the dataset as 80% training set and 20% test set. We achieved 100% actuary in the test set for both tuned models. Therefore, we increased the test set to 30% to reduce the over-fitting. The test accuracy for tuned SVM then becomes 99.4% and MLP becomes 99.6%. MLP performed slightly better than SVM, which proves our hypothesis to be incorrect. This could due to the continued over-fit problem in the SVM model. Another explanation could be this overall size of our dataset is relatively small to see obvious difference between two models. We believe both models may have high variance for their high accuracy rate in the testing set. However, we tried to a 50-50 split between training and testing set, and we still managed to produce fairly accuracy results. Thus, the variance can be ignored. In theory, the SVM should outperform the MLP based on fundamentally different learning strategies. In MLP, weight and bias are adjusted such that the loss function between the network output and the target is minimized. In SVM, by contrast, it is an explicit determination of the decision boundaries directly from the training data by minimizing the aggregate distance between the maximum-margin hyperplane and the support vectors.

In terms tuning process of MLP, it is difficult to decide the net size, which are number of hidden layers and number of neurons in each layer because there is no standard specification and it is also depends on different situations. Thus, one hidden layer with neurons from 40-70 were selected. It is proved that it is one single hidden layer is enough with accuracy of 99.6% on this car evaluation dataset so that no more hidden layers were added. But if dataset is large and complex, more layers are essential to achieve relative high accuracy. From illustration of MLP training process (figure 5), it can be concluded that the training loss is reduced rapidly before 10 epochs and gradually decreased with increased of epoch. It indicates that learning rate decay method can be considered to find global minimum with increased epochs.

In accordance with our hypothesis, hyperparameter tuning in SVM improved our result from an untuned model from 92.9% test accuracy to 99.4%. Our training accuracy also improved from 94.7% to 100%. However, the tuned model requires a lot of more computational power. The elapsed time is 389 seconds, compared with 1.1 seconds for the untuned model. From figure 7, We also learnt that the minimum estimated objective and the minimum observed objective converged only after 5 iterations. This suggest that we can reduce the number of iterations to save computation power.

From our initial analysis, we suggested that we should use the 21 features from one-hot-encoder instead of label encoder's 6 features. We tested our hypothesis and the results do not completely agree with our suggestions. With 6 features, the tuned SVM model achieved an accuracy of 99.4%, hence the SVM model is unaffected by the label encoder. However, the untuned SVM model only achieved an accuracy of 97.5%, which is around 2% lower than the accuracy from one-hot-label (13 compared to 3 incorrect classification when considering individual prediction output). Hence, our results prove the necessity of using one-hot-encoder.

Behzad [8] produced similar result on the same car dataset using an SVM model. The model was trained in a similar set-up (21 variables) on a Python environment. However, our optimized model (99.4% testing accuracy) outperformed Behzad's (93% testing accuracy).

8. Conclusion

Both deep learning models (MLP and SVM) are capable of handling complex data with 99.6% and 99.4% accuracy separately. We feel our current dataset is too small to identify the actual performance of two classifiers. In addition, the performance of both models is not affected a lot by skewness of raw data as described in previous part. The imbalance was not such to necessitate the generation of synthetic data for the other 3 classes, however the use of synthetic minority oversampling for balancing of all 4 classes would potentially make an interesting extension for further work.

Both models are a great success when applied to this Car Evaluation dataset. However, for our model to be used commercially, we need to be able to understand and interpret our models. Unlikely decision trees, MLP and SVM are considered more as 'black-box' models, which are hard to interpret. We could apply the LIME algorithm to improve out interpretability [9]. LIME tests what happens to the predictions when you give variations of your data into the machine learning model. It generates a new dataset consisting of permuted samples and the corresponding predictions of the black box model.

In terms of MLP classifier, it would be interesting to apply the model on a similar larger data set inclusive of further car attributes such as extras, engine capacity and primary energy source which are quite important in developed markets. A viable extension of the work would be to examine the effect of different activation function as TanH, ReLu or Leaky ReLu. In addition, more layers of hidden layer, different optimization algorithm (e.g. Adam [10]), learning rate decay and regularization can be considered.

In terms of SVM, it achieved 99.4% test accuracy for tuned SVM model, which is likely over-fitted. To avoid the problem, we need to increase size of our dataset. SVM is a very good tool for classifying high dimensional data and we should introduce further car attributes to our model. Moreover, SVM uses support vectors to define the hyperplane. It would be interesting to see how the support vector changes if we increase the data size and the dimensionality.

In future, the application of data transformations and dimension reduction techniques (PCA) could also be investigated, perhaps for an expanded version of similar car evaluation data. This may improve the models and also provide useful high-level knowledge for car-industry decision makers.

Reference

[1] M. Bohanec and V. Rajkovič. "DEX: An expert system shell for decision support", Sistemica, 1(1), pp.145-157. 1990.
[2] UCI Machine Learning Group [online] http://archive.ics.uci.edu/ml/datasets/Car+Evaluation. [Accessed 27 March 2019].
[3] D.E. Rumelhart, G.E. Hinton and R.J. Williams. Learning internal representations by error propagation, in: Parallel Distributed Processing: Explorations in the Microstructure of Cognition, eds. D. E. Rumelhart and J. L. McClelland, Vol. 1, MIT Press, Cambridge, MA. 1986.
[4] T Evgeniou and M Pontil. (2001). Support Vector Machines: Theory and Applications.
[5] L. Auria and R. A. Moro. Support Vector Machines (SVM) as a Technique for Solvency Analysis, DIW Berlin, Aug 2008.
[6] R. Collobert and S. Bengio. Links between perceptrons, MLPs and SVMs. In Proceedings of the twenty-first international conference on Machine learning (p. 23). 2004, July.
[7] A. D. Bull, Convergence rates of efficient global optimization algorithms, Journal of Machine Learning Research, 2011.
[8] Behzad, classification, available at: https://www.kaggle.com/behzadmehmood82/classification. [Accessed 27 March 2019]
[9] M. T. Ribeiro, S. Singh, and C. Guestrin: "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv preprint arXiv:1602.04938.
[10] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980. 2014.