Classifying Vehicle Dataset in Neural Networks:

Heuristic Pattern Reduction and Other Techniques

Bokun Kong,

Research School of Computer Science, Australian National University

2601, ACT, Australia

{Bokun Kong, u6342099}@anu.edu.au

Abstract. Much research has recently been carried out using feed-forward networks of a few layers trained by back-propagation to solve a variety of problems. A number of problems are existed in using such neural networks, including the speed of training, the unexplainable black box and the avoidance of local minima. Here I report on a series of experiments to test the hypothesis that a reduction in the complexity of a training set can improve learning with multiple layers of deep learning neural network. I investigate the vehicle silhouettes [4] dataset collected from UCI Machine Learning Repository conducting a simple heuristic method of reduction of the size of a training set as well as over-sampling in data preprocessing processes. The performance of neural net on the validation test set is improved by these approaches.

Keywords: Heuristic Pattern Reduction, Over-sampling, Neural Network

● Introduction

I investigate the vehicle silhouettes dataset [4] collected from UCI Machine Learning Repository in this paper. The motivation for the choice of that dataset is: it is relatively complex dataset with 18 attributes and 946 instances. Besides, this dataset is general and simple to implement for theoretical research of techniques of neural network, thus there are a number of relevant papers and papers that cite this dataset, which can compare and evaluate the theoretical approaches clearly. Most importantly, this dataset is an imbalanced dataset and accuracy of learning patterns can be improved by over-sampling heuristic pattern reduction.

Some constraints are introduced which limit the resources available to the network and force it to generalise rather than learn the specific patterns presented. This sets the scenario for the hypothesis that a reduction in the number of training patterns as well as hidden units may be usable as a resource limiting constraint to improve generalisation and accuracy. My investigation aim is carrying out a number of approaches in the model such as over-sampling heuristic pattern reduction [1] and other techniques to see whether they improve the neural net performance.

A number of contributions in the aspect of pruning of neural networks have shown that the generalisation capabilities of the neural network can be improved by reducing in network size to some minimal size. The seminal work on pruning by inspection was by Sietsma and Dow [6]. Besides, outlier removal has been introduced to reduce the initial variance in the training set and thus improve the trade-off of variance/bias [3].

In this paper, I will generally assume a multi-layer feed-forward network of three layers of processing units trained using back-propagation. All connections are from units in one level to the subsequent one, with no lateral, backward or multilayer connections. Each unit has a simple weighted connection from each unit in the layer above. Validation set of patterns were provided to the network for testing. These patterns are never seen by the network during training and thus can offer a good measure of the generalisation capabilities of the network.

● Method

The experiments are performed on a sample of 946 patterns taken from vehicle silhouettes csv files. The raw dataset "original" consists of 18 attributes and 4 classes. I initially convert categorical target attribute "classes" to numeric "0,1,2,3". Then I write the new data frame into a new csv file "processed". I load the "processed" dataset and convert pandas' data frame to array. For splitting training and test set, I import train_test_slit package from sklearn and separate 80% of the whole training set as training dataset, while the rest is test dataset. There are 19 columns divided into features and target, the first 18 columns are features and the last column is target.

After splitting features and target, normalization is attempted by computing the maximum and minimum values in one particular column, then using formula:

```
xs = (1 - 0) * (x_array - xs_min) / (xs_max - xs_min) + 0
```

to compress the values in the range of 0 to 1. Because normalization before data input can improve convergence speed as well as effectiveness and reduce the chances of getting stuck in local optima. Next, I create Tensors to hold inputs and outputs, and wrap them in Variables as Torch only trains neural network on Variables.

Sequential layers structure of neural network is used for the convenience to add and modify layers. I attempt both Sigmoid and ReLU activation function and add dropout layer: `torch.nn.Dropout(0.1)`. When the model is too complex, for example, there are too many input parameters or too much noise, the model will have an overfitting problem as it overreacts to minor fluctuations in the training data. There are several solutions such as cross-validation, regularization, early stopping, pruning and dropout layer.

The function of the optimization algorithm is to minimize (or maximize) the loss function E(x) by improving the training

method. Some of the parameters in the model are used to calculate the deviation between the true value and the predicted value of the target Y in the test set. The form of loss function E(x) is based on these parameters. The internal parameters of the model play a very significant role in effectively training the model and producing accurate results. Therefore, a variety of optimization algorithms are supposed to be considered to update and calculate the network parameters that affect the model training and model output to make it approach or reach the optimal value. Therefore, I attempt SGD, Momentum, RMSprop and Adam to train neural network and use loss function to calculate error. Moreover, I should mention that in avoidance of overfitting, I also apply L2 regularization. The L2 regularization on the parameters of the model is already included in most optimizers, including `optim.Adam` and can be controlled with the `weight_decay` parameter, so I use syntax:

```
optimiser = torch.optim.Adam(net.parameters(), lr=learning_rate, weight_decay=1e-5)
```

Then I define the loss functions with:

```
loss_func = torch.nn.CrossEntropyLoss()
```

Because this task is classification and loss function should be evaluated by cross-entropy.

In the process of training the network, gradient descent is a classical common method to minimize the risk function/loss function. Except for statistic gradient descent, batch gradient descent and mini batch gradient descent are also prevalently used. Stochastic training tends to be faster than batch training on big datasets, however, the theoretical analysis of the weight dynamics and convergence rate of batch training is simpler. In batch learing, Batch_size is a significant parameter in neural network. Firstly, the choice of batch determines the direction of the descent. If the dataset is small, Full Batch Learning can definitely be applied as the direction determined by the full data set can better represent the sample population and thus more accurately toward the direction of the extreme value. However, Full Batch Learning does not suit the investigation of massive datasets. The other extreme Online Learning means train only one sample at one time, that is: `Batch_Size = 1.` Using online learning, each correction direction is corrected in the direction of the gradient of the respective sample, and it is hard to achieve convergence by rushing into each other. Consequently, I attempt mini batch learning in the experiment to investigate whether the performances will be better on this dataset. Initially I transform the array to tensor dataset that can be recognized by torch, then I train the loader:

```
train_loader = Data.DataLoader(dataset=torch_dataset, batch_size=batch_size, shuffle=True)
```

After that, batch data is allocated, features and target are normalized when iterating the loader.

Similarly, Heuristic Pattern Reduction (HPR) is a method which attempt to map the individual training patterns onto a one-dimensional adjacency measure using their contribution to the total sum of squares for the reduction of the overall size of the training set. This adjacency is used as the basis for arbitrary assumptions about the sizes of clusters of training patterns [1]. If the dataset is sufficient, then the calculated gradient with merely half (or much less) of the data is almost the same as the gradient that was trained with all the data. Thus, assuming a homogenous distribution of patterns, the training set can be reduced to half its size by 15% each time then evaluate the results. Moreover, because of the imbalance

of dataset, the model is more likely to learn the higher distributed values, thus the outcomes of predicted classes are more possibly to be imbalanced distributed. In this dataset, the number of entries of target classes is 212, 217, 218 and 398 respectively. In order to get better results, I implement over-sampling by reducing the size of patterns from 398 to 220.

However, patterns should not be removed too quickly, which is attributed to those patterns with midrange errors that could eventually be learnt by the network. Moreover, patterns should not be removed indefinitely, because the network is devoting a large number epochs of training to a reduced set of examples when the training set becomes smaller, hence dramatically increasing the overfitting effect potentially [2].

After defining a neural network, I train the neural network. Computing loss is important and should be printed in progress, as well as accuracy. In order to evaluating the results and see how well the network performs on different categories, I will create a confusion matrix, indicating for every sample (rows) which class the network guesses (columns). Next, I load and setup testing dataset which has the same data pre-processing and training techniques. The neural network is tested by passing testing data to the built neural network and get its performance. It is actually performing a forward pass computation of predicted y by passing x to the model. Accuracy is calculated and confusion matrix is generated to evaluate the results.

● Results and Discussion

I set the deep learning neural net with three hidden layers, and the network is initially trained for 500 epochs on the full training set. The individual training patterns are mapped onto a one-dimensional adjacency measure using their contribution to the total sum of squares. In cleaning data processes, I attempt different data pre-processing method including normalization and regularization. Moreover, as define the neural network I also try to modify sequential layers, for example, adding dropout layer and attempting Sigmoid and ReLU activation functions. In training neural networks, different optimizers are attempted to get better results. Different batch sizes are attempted when applying mini batch learning. Considering HPR, I train the neural net with a reduced size of training set and over-sampling approach in data pre-processing. Finally, I compare these improvement to the original neural network.

I performed 10 runs of each configuration, with different initial weights. The results of the comparison study are displayed in Table 1 with the predicted testing accuracy being represented by the total sum of squares (tss) value, the lower tss the better the prediction. Note that the training accuracy and the testing accuracy shown are the maximum values of the number of runs done and can be from different runs.

TECHNIQUES	TRAINING ACCURACY	TESTING ACCURACY	TSS
ORIGINAL(RELU)	56.79%	54.26%	64.43
SIGMOID	56.69%	57.98%	50.80

NORMALIZATON(SIGMOID)	51.82%	49.47	35.90

Table 1.

In Table 1, the original network is defined with one hidden layer and ReLU avtivation function. I compare the original network to the neural network using normalization and sigmoid activation function. The results are summarized by Table 1, the neural networks applying ReLU as activation function tend to have slightly higher training accuracy than those using Sigmoid. However, the neural networks with Sigmoid are inclined to perform better on testing accuracy and the tss of sigmoid is less than that of ReLU which means the network is more stable and prediction is better. In order to get better results on prediction of testing dataset, I will apply Sigmoid instead of ReLU.

Unexpectedly, normalization do not perform well and get higher accuracy on both training and testing dataset as we expected. Normalization before data input can be very effective in improving convergence speed and effectiveness. The undesirable outcome of accuracy by normalization might be attributed to the reason that any rescaling of an input vector can be effectively undone by changing the corresponding weights and biases, leaving the exact same outputs as investigated before. Although, standardizing the inputs can make training faster and reduce the chances of getting stuck in local optima, perhaps the corresponding weights and biases in the neural network with normalization are trained worse than the original network.

When adjusting the way that the neural network model updates weights and bias parameters, finding the best optimization algorithm is considered to produce better results more efficiently. The same number (10) of runs of each configuration are performed in the experiment of different optimizers: SGD, Momentum, RMSprop, Adagrad and Adam. The results are compared and shown in Table 2 with the training accuracy, testing accuracy and tss.

OPTIMIZER	TRAINING ACCURACY	TESTING ACCURACY	TSS
ORIGINAL(SGD)	56.69%	57.98%	50.80
MOMENTUM	71.88%	68.62%	27.83
RMSPROP	59.73%	53.72%	84.18
ADAGRAD	76.90	75.00%	18.76
ADAM	96.20%	79.26 %	15.15

Table 2.

According to Table 2, the networks using Stochastic Gradient Descent tend to have lower training and testing accuracy and greater tss. SGD updates the parameters of each training sample and the execution is surprisingly fast. The problem of SGD is frequent updates result in the high variance between the parameters and the loss function fluctuates at different intensities. Due to frequent updates and fluctuations, it will eventually converge to a minimum, and there will be frequent overshoots due to fluctuations. Thus, testing accuracies are highly variant and tss of SGD is greater than average. Because of the high variance and fluctuations, the momentum technique is introduced by optimizing training of network in the relevant directions and attenuate oscillations in irrelevant directions to speed up SGD training. Adagrad is applied by

adjusting the appropriate learning rate η through parameters, substantially updating sparse parameters and slightly updating frequent parameters. Therefore, Adagrad is very suitable for dealing with sparse data. However, the vehicle silhouettes dataset is not sparse data, so Adagrad is not appropriate in this experiment. Adaptive Moment Estimation can calculate the adaptive learning rate for each parameter. According to the results, Adam has better results compared to other optimizers as it will fix the problems of disappearance of learning rate, slow convergence, or high variance of parameter updates leading to large fluctuations in loss function.

However, networks applying Adam tend to have high training accuracies but unexpected low testing accuracies, which might be attributed to overfitting. Although the model shows good prediction results on the training set, the prediction results are very poor. I will investigate regularization and dropout techniques by comparing the accuracy and tss in Table 3.

TECHNIQUES	TRAINING ACCURACY	TESTING ACCURACY	TSS
ORIGINAL(ADAM)	96.20%	79.26 %	15.15
DROPOUT	88.15%	80.32%	21.56
REGULARIZATION	95.29%	82.45%	19.72
DROPOUT+REGULARIZATION	87.99%	80.85%	42.32

Table 3.

All of the testing accuracies are higher after applying techniques avoiding overfitting in particular dropout. The training accuracy dropped dramatically and testing accuracy increased slightly, which means the network training does not rely too much on training dataset and avoids training noise. L2 regularization is applied when defining the optimizer. Although training accuracy of regularization is still high, the testing accuracy of regularization is improved and has better performance than dropout. Consequently, I add dropout layer in the neural network combined with L2 regularization. However, the tss of the network applying combined techniques is high, possibly because the dropout layer drops some hidden neuron and the unstable network leads to fluctuations of loss function, which results in the high variance of outputs.

Considering batch learning, investigations of mini batch learning is conducted and shown in Table 4.

BATCH_SIZE	TRAINING ACCURACY	TESTING ACCURACY	TSS
ORIGINAL	87.99%	80.85%	42.32
BATCH(SIZE=50)	96.50%	81.32%	18.70
BATCH(SIZE=60)	94.68%	82.98%	66.56

Table 4.

Table 4 illustrates that using batch learning can improve testing accuracy and find an appropriate batch_size is important. Yet, the tss of network applying batch_size = 60 is dramatically greater than that using batch_size = 50. That possibly because bias and variance are very sensitive to the complexity of the data and the batch_size trained in network. Eventually, the batch learning method leads to the most uncontrolled increase in variance of all the methods. Accordingly, selecting

a moderate Batch_Size value is necessary, because if the data set is sufficient, then the calculated gradient with almost half (or much less) of the data is almost the same as the gradient that was trained with all the data. That theory is similar to the Heuristic Pattern Removal [1]. I simply reduce the number of samples in training dataset and compare the results in Table 5. In the same time, over-sampling approach is also attempted. I modify the size of different target classes in the original training dataset by reducing the number of the fourth class "3" from 398 to 220 and present the results as cleaned dataset.

PERCENTAGE OF PATTERNS	TRAINING ACCURACY	TESTING ACCURACY	TSS
100%	93.68%	81.98%	66.56
100%(CLEANED DATASET)	94.32%	82.06%	60.32
85%	95.81%	82.51%	32.26
85%(CLEANED DATASET)	96.22%	83.59%	31.33
70%	93.68%	79.85%	36.69
70%(CLEANED DATASET)	94.91%	79.98%	35.80
55%	94.74%	75.60%	39.67
55%(CLEANED DATASET)	95.11%	76.28%	37.58

Table 5.

Pattern reduction techniques aim to simplify the error surface of the pattern space, however simple elimination of original training patterns did not show significant improvement. Training accuracy, testing accuracy and tss are improved slightly by simply eliminating 15% of original training patterns. The outcome is most likely because of the original outliers in the remaining training data set, and the process of error surface simplification might be affected by these outliers. The obtained classification accuracy is the same as the whole dataset, when the outliers were defined and removed from the training set. Moreover, the network performs worse by simple elimination of 30% or 45% of original training patterns. Perhaps the dataset that I used is not massive, and thus the network as well as bias and variance are extremely sensitive to the changes in the dataset. Insufficient training dataset results in undesired performance including low accuracy and high tss. Moreover, the over-sampling approach also improves the performance on the model in any circumstances of heuristic pattern reduction because the model learns from a well distributed dataset rather than an imbalanced dataset, which is beneficial to the extraction of interesting patterns and generalization. The best result is conducted with heuristic reduction of 15% of original training patterns and over-sampling approach, which is 96.22% training accuracy and 83.59% test accuracy.

Implementation of Heuristic Pattern Reduction and other techniques contribute to satisfied results on neural network, I compare the results to other papers that cite the vehicle silhouettes dataset, which is displayed in Table 6.

	TRAINING ACCURACY	TESTING ACCURACY	TSS
BEST RESULT	96.22%	83.59%	31.33
NEURAL NETWORK	91.90%	83.70%	NA

SVM	70.21%	70.21%	NA

Table 6.

In Constructive Neural-Network Learning Algorithms for Pattern Classification, the training accuracy 91.90% is lower than that in this experiment using Heuristic Pattern Reduction with over-sampling approach and other techniques, however, the testing accuracy from the paper performs slightly better [5]. The reason might be the insufficiency of training dataset leading to the bad performance on HPR. Moreover, I also compare the best result to another paper using Support Vector Machines and conclude that this machine learning algorithm performs much worse than neural network in this experiment [7]. Because SVM produces good outcomes on small dataset, while this vehicle dataset is a large dataset which is suitable for neural network.

● Conclusion and Future Work

I have shown that a simple heuristic pattern reduction method can be applied on reduction of the size of the training pattern set properly as well as improvement of performance on the testing set. A reduction in the number of training patterns may also accelerate the training of feed-forward networks. Besides, Sigmoid activation function and normalization are also applied. I compare the optimizers and use Adam to train the network combined dropout layer and regularization in mini batch. Finally, I have the best result 96.22% training accuracy, 83.59% testing accuracy and 31.33 for tss.

According to the experiment, I notice that heuristic pattern reduction has undesired performance on this comparatively small dataset, and thus for future research, investigations on heuristic pattern reduction are supposed to be conducted on massive dataset and evaluate the outcomes. Meanwhile, an advanced approach evolutional algorithm can be implemented in this neural network.

● References

[1] Gedeon, T. D., & Bowden, T. G. (1992). Heuristic pattern reduction. In *International Joint Conference on Neural Networks* (Vol. 2, pp. 449-453).

[2] Gedeon, T. D., Wong, P. M., & Harris, D. (1995, June). Balancing bias and variance: Network topology and pattern set reduction techniques. In *International Workshop on Artificial Neural Networks* (pp. 551-558). Springer, Berlin, Heidelberg.

[3] Geman, S., Bienenstock, E., & Doursat, R. (1992). Neural networks and the Bias/Variance dilemma. *Neural*

Computation, 4(1), 1-58. doi:10.1162/neco.1992.4.1.1

[4] Mowforth, P., Shepherd, B. Statlog (Vehicle Silhouettes) Data Set. *Turing Institute*, Glasgow, Scotland

[5] Parekh, R., Yang, J., & Honavar, V. (2000). Constructive neural-network learning algorithms for pattern classification. *IEEE Transactions on neural networks*, *11*(2), 436-451.

[6] Sietsma, J., & Dow, R. J. F. (1991). Creating artificial neural networks that generalize. *Neural Networks, 4*(1), 67-79. doi:10.1016/0893-6080(91)90033-2

[7] Sindhwani, V., Bhattacharya, P., & Rakshit, S. (2001, April). Information theoretic feature crediting in multiclass support vector machines. In *Proceedings of the 2001 SIAM International Conference on Data Mining* (pp. 1-18). Society for Industrial and Applied Mathematics.