Neural Network Application in Income Classification

Tianyu Li
Research School Computer Science, ACT 2601,
Canberra, Australia
u5953423@anu.edu.au

Abstract. We used neural network to solve income classification based in the UCI Census Income dataset. The RPROP optimization algorithm, the data normalization strategy and genetic algorithm feature selection increased the overall network prediction accuracy to approximately 85%.

Keywords: neural network, classification, feature engineering, optimization, data normalization, genetic algorithm

1 Introduction

This paper is about the process of neural network modelling and optimizing. The neural network is based in the tailored data from Census Income Data Set (http:// http://archive.ics.uci.edu/ml/datasets/Census+Income). The network design gets inspiration from the encoding and scaling methodologies by Bastos and Gedeon (1995), uses a Stochastic Gradient Descent (SGD) optimizer, and adapts Cross Entropy function as Loss function for the original network. Then the RPROP optimization algorithm, the data normalization strategy and genetic algorithm feature selection will contribute to the enhanced prediction accuracy. The report makes comparison between the outputs of neural network method and its counterpart, C2.5 and Naïve-Bayes algorithm and analyses, and its improved accuracy (~85%) is between C2.5 and Naïve-Bayes algorithm.

2 Data Pre-processing

The original data includes 14 attributes and 48842 instances in total. There are 32561 instances in the training data and 16281 instances in the testing data. The data set provides basis to predict the income level of individuals and classify the personal income level by 50k US dollars per year. Before we start to design the neural network, data pre-processing is required.

The Census dataset has enough instances for network training purpose, and it is a combination of categorical and continuous numeric attributes. The prediction problem is a typical classification problem which is suitable for the adaption of neural network method.

2.1 Data Cleaning

It is noticed that the raw data include some missing values. It is not appropriate to delete any rows with some data missing, as the completeness of the other non-missing values will be disrupted. Therefore, we need to handle the missing values using data cleaning techniques. The '?' values should be replaced with np.nan, which can be handled by numpy as a kind of recognized empty value.

2.2 Data Reduction

The data provided by the donor have already been partly encoded by introducing independent numeric attributes. For instance, the attribute education_num is respectful to the adjacent attribute education, and they represent the same sort of concepts.

Table 1. Eduation-num and Education attribute values.

Education-num	Education
1	Preschool
2	1st-4th
3	5th-6th
4	7th-8th
5	9th
6	10th
7	11th

8	12th
9	HS-grad
10	Some-college
11	Assoc-voc
12	Assoc-acdm
13	Bachelors
14	Masters
15	Prof-school
16	Doctorate

As Education-num is a numeric substitution of attribute Education and it is a continuous set of values, the network prediction could skip Education and adopt Education-num. We must pinpoint that the encoding in Education-num might not be the perfect way to describe the characteristic of different education levels. Some kinds of better rescaling might be an effective method to improve the accuracy of the classification.

Similarly, we choose Workclass and neglect Occupation because Workclass is a higher level of categorization of occupations; and keep Relationship and get rid of Marital_status, because the Relationship attribute covers most the information in the Marital_status through deduction, one example is that a 'Married' in Marital_status is a prerequisite of value 'Husband' or 'Wife' in the attribute Relationship.

Data reduction is a trade-off between the data integrity and the model training cost. This might have negative effect in the accuracy of the network's final prediction, but helpful when reducing the complexity of the dataset and respectful neural network training.

2.3 Data Transformation

For neural network building, the data should be of proper pattern. The categorical data should be all encoded in an appropriate manner. According to Bastos and Gedeon (1995), a group of columns with numeric values can be a substitution of single column with a wide range of values. So-called "One-Hot Vector" encoding pattern is suitable for neural network building, as a group of columns with value 0 or 1 work simultaneously to describe a single categorical value. Compared to value them a series of number in the same column, one-hot encoding pattern can be handled better by the decision functions. For instance, the categorical attribute Workclass in the raw data set have 8 different values. It could be transformed into 8 one-hot attributes subsequently: workclass_private, workclass_self_emp_not_inc, is_self_emp_inc, is_federal_gov, is_local-gov, is_state-gov, is_without-pay, is_never-worked.

2.4 Generalization

Most of the categorical attributes in the dataset could be directly transformed into one-hot encoding pattern, yet for the attribute native_country there are too many possible values, making the following network training hard and of high time cost because of the subsequent large number of added. There are 41 possible country values in total; in the training data most of instances are of value United-States, for some categories such as Laos and Outlying-US, their frequencies are less than 0.1% as for the entire dataset. Therefore, it is necessary to reorganize this attribute into a binary split: United-States (Value 1) and Non-United-States (Value 0).

Table 2. Modified training dataset pattern.

Column	Attribute name	Comments
1	Age	
2	Fnlwgt	
3	Education_num	
4	Sex	Valued into 0,1
5	Capital_gain	
6	Capital_loss	
7	Hours_per_week	
8	Native_country	Transformed, generalized into 2 values: US and non-US
9	workclass_private	Deducted from workclass
17	workclass_self_emp_not_inc	Deducted from workclass
18	workclass_self_emp_inc	Deducted from workclass
19	workclass_federal_gov	Deducted from workclass
20	workclass_local_gov	Deducted from workclass
21	workclass *state*_gov	Deducted from workclass
22	workclass _without_pay	Deducted from workclass
23	workclass_never_worked	Deducted from workclass
24	Relationship_Wife	Deducted from Relationship

25	Relationship_Own-child	Deducted from Relationship
26	Relationship_Husband	Deducted from Relationship
27	Relationship_Not-in-family	Deducted from Relationship
28	Relationship_Otherrelative	Deducted from Relationship
29	Relationship_Unmarried	Deducted from Relationship
30	Race_White	Deducted from Race
31	Race_Asian_Pac_Islander	Deducted from Race
32	Race_Amer_Indian_Eskimo	Deducted from Race
33	Race_Other	Deducted from Race
34	Race_Black	Deducted from Race

3 Neural Network

A neural network is built for handling classification issue in the processed Census dataset.

3.1 Network design overview

To begin with, I build a basic feed forward neural network with 3 layers, one of them is a hidden layer. The input layer contains 27 neurons, representing the features of the Census dataset; the hidden layer has 100 neurons, using Sigmoid as activation function; the output layer has 2 neurons, representing the binary classes of prediction result. We set the original epoch number as 500, and its value could be higher in case of need. The the hidden layer should also be capable with all variable computing, so the minimum number of neurons in the hidden layer should be 27, which is the number of the input neurons, otherwise the fitting issue will be likely to appear. The learning_rate is set to be 0.01. The performance of the neural network will be evaluated using cross-entropy loss function.

3.2 Optimisation Algorithms

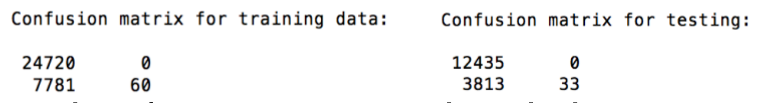
3.2.1 Stochastic Gradient Descent (SGD)

The neural network is originally chosen to be trained with Stochastic Gradient Descent (SGD) as an optimiser, that will hold the current state and will update the parameters based on the computed gradients. Bottou (2010) pointed out that SGD algorithm is especially effective in large-scale computation. As our Census dataset has 27 feature attributes and over 30000 instances in the training set, SGD will benefit in the massive calculation and accelerate the classification. Nevertheless, the SGD optimizer has its drawback that unlike mini-batch training and Semi-Stochastic Gradient Descent (Semi-SGD) which update the gradients with each new training data sample, the weight calculations is based on a uniform noisy estimation of the entire dataset. This leads to moderately lower precision and fluctuation in the loss rate and the accuracy against the number of epochs. The result, 76%, is not comparable with other listed algorithms, this indicates that the raw dataset might be translated to the training data in a defective manner, it is also worthy to consider whether SGD is good enough as an optimization algorithm for this circumstance.

3.2.2 Mini-batch Gradient Descent Algorithm

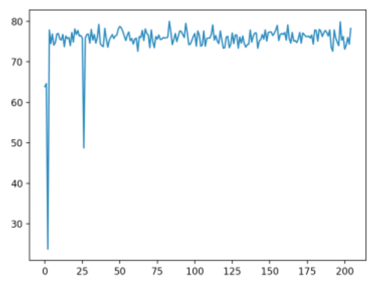
The neural network is originally chosen to be trained with Stochastic Gradient Descent (SGD) as an optimiser, that will hold the current state and will update the parameters based on the computed gradients. Bottou (2010) claimed that SGD algorithm is not perfect because the algorithm use a static gradient all around. Compared to SGD, Mini-batch training is less cost-effective as it updates the gradient each batch. Theoretically, mini-batch training is a trade-off between GD and SGD, to get moderately high training efficiency while reducing part of model inaccuracy caused by SGD's one and only gradient parameter. Papamakarios (2014) shows that Mini-Batch versions of SGD have predetermined sub-set of functions, which form the mini-batches. The algorithm then use the mini-batch instead of the single gradient. To see the result, please run the code_mini_batch.py.

Fig. 1. Confusion matrix of mini-batch training model given hidden neuron=100, learning rate=0.01.


```
Confusion matrix for training data:     Confusion matrix for testing:

 24720      0                            12435      0
  7781     60                             3813     33
```

Testing accuracy according to the confusion matrix: 76.58%, epoch=100, batch_size=800.

Fig. 2. Accuracy rate(y) of mini-batch training model versus epoch(x), given hidden neuron=100, learning rate=0.01



3.2.3 SGD using backpropagation as a gradient computing technique (Rprop)

Resilient backpropagation (Rprop) is a fast first-order optimizer (Riedmiller and Braun, 1992). The Rprop algorithm only consider the sign of partial derivative over all patterns(not the magnitude) on each weight. Neural networktrained with SGD using Rprop as a gradient computing technique generally provides higher accuracy.

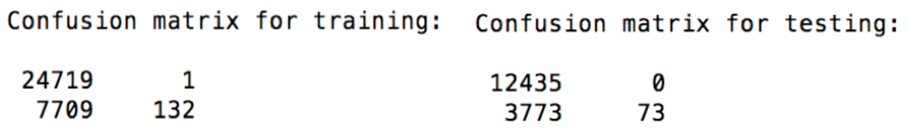
3.2 Original Network test results and discussion

According to Kohavi (1995), for the same adult dataset the Naïve-Bayes algorithm had accuracy between 83.5% and 84% given 32500 instances; and the C4.5 decision tree algorithm had accuracy between 85% and 85.5% given approximately 32500 instances.

In the meantime, the accuracy of the network prediction in training and test data is around 76%. The reason of lower accuracy is possibly the modification to the raw data. The information loss due to the dataset tailoring might result in the lower overall accuracy.

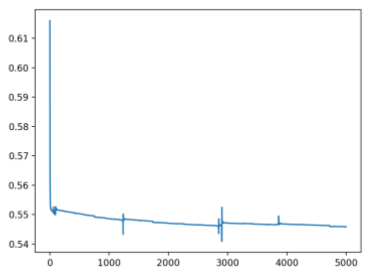
The accuracy values for test data (76.52%) and training data (76.32%) are at a similar level, and This depicts that there is not overfitting issue during the model's training, and the relatively low level of accuracy (compared 75%) means improvement steps are still required to increase the fitting quality. The possible solution might be reducing features, adopting better optimization algorithms, adding more hidden layers to make a deeper neural network or adding more neurons in the hidden layer. Enhancing the scale of epochs could also beneficial, given the chance of the accuracy curve keep going upward.

Fig. 3. Confusion matrix given hidden neuron=100, learning rate=0.01



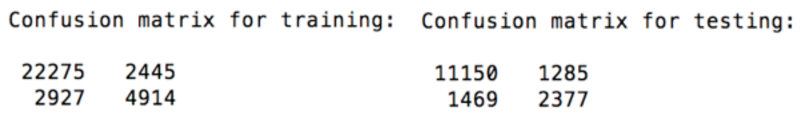
```
Confusion matrix for training:   Confusion matrix for testing:

 24719       1                    12435       0
  7709     132                     3773      73
```

Fig. 4. Loss rate given hidden neuron=100, epoch=5000, learning rate=0.01.

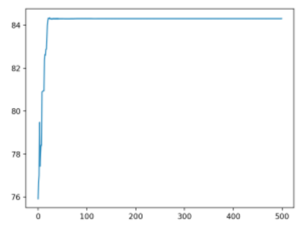


It can be found that SGD-Resilient Backpropagation algorithm (Rprop) has a better testing accuracy compared to SGD (76.52%), as it helped the model prediction to acquire 83.08% accuracy in the test set. To see the result, please run the code_rprop.py.

Fig. 5. confusion matrix for training and testing & accuracy given Rprop optimiser, hidden neuron=100, epoch=5000, learning rate=0.01



```
Confusion matrix for training:   Confusion matrix for testing:

 22275    2445                    11150    1285
  2927    4914                     1469    2377
```



3.3 Improvement step: Adding neurons in the hidden level

According to we discussed above, more neurons and more epochs leads to higher accuracy anticipation. Given epoch value 5000 and hidden layer neuron value 500, we found that the test accuracy increased by 1% to 77.24%. However, further movement to increase the value of epochs or hidden layer neurons would be inappropriate, as the training will cost too long time and the training model often become over-fitted.

To see the result, please run the code_hidden_neuron_500.py.

Fig. 6. Loss rate(y) versus epoch(x), given SGD optimiser, hidden neuron=500, epoch=5000, learning rate=0.01

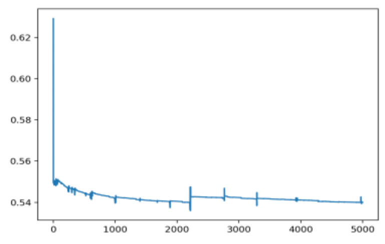
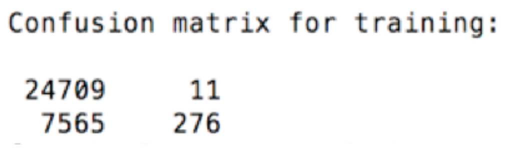
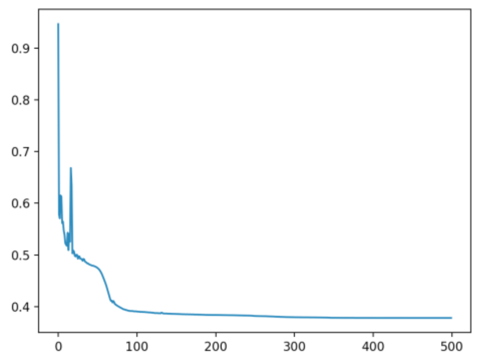


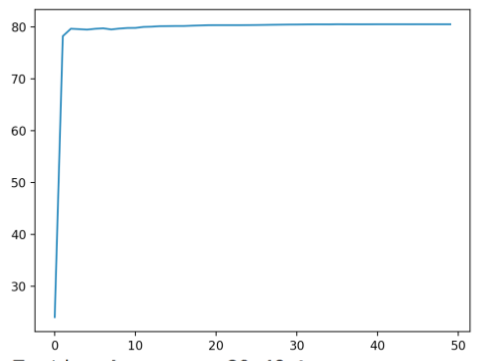
Fig. 7. Accuracy (77.24%) for testing set and confusion matrix for training given hidden neuron=500, epoch=5000.

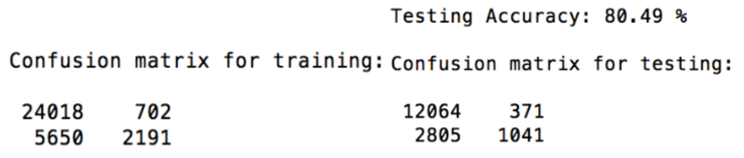


```
Confusion matrix for training:

 24709      11
  7565     276
```

3.4 Improvement step: 4-Layer DNN with Rprop and 120 neurons in hidden layers







```
                                  Testing Accuracy: 80.49 %

Confusion matrix for training:   Confusion matrix for testing:

 24018     702                    12064     371
  5650    2191                     2805    1041
```

Fig. 8. Loss & Accuracy for testing set and the confusion matrix given hidden neuron=120, epoch=5000.

The result(80.49% testing accuracy) is not as good as similar network with single hidden layer. Deeper neural networks(DNNs) tends to stick in the local extremum easier than 3-layer NNs. The sigmoid function is not the best choice for the activation function in deep feedforward networks, because with the increasing number of hidden layers, the sigmoid function leads to inevitable loss of information. Better choices for the activation function of DNN include Softmax and ReLU. To see the result, please run the dnn.py.

3.4 Improvement step: Data normalization, further data generalisation and feature engineering

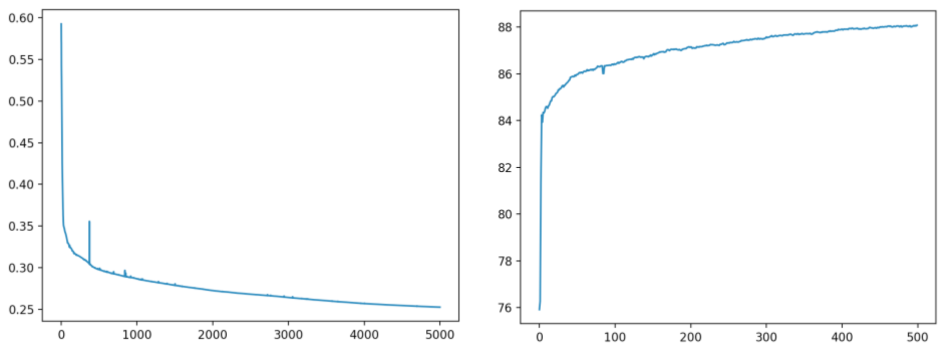
In the first stage of the paper, we technically skipped the normalization of some feature attributes in the raw data set. This might affect the effectiveness of the processed training data. In this section, we set an approach to normalize the relevant data and check the result.

Data Normalisation

Gedeon and Bustos (1995) pointed out that normalization for input data is necessary to fit the logistic function in the neural network. Thus, we need to normalize the data over range 0 to 1.
Amongst the training and testing dataset:
The values of attribute ‘age’ are ranged from 17 to 90: divide the values by 100.
The values of attribute ‘fnlwgt’ are ranged from 12285 to 1490400: divide them by 1600000.
The values of attribute ‘education_num’ are ranged from 1 to 16: divide them by 16.
The values of attribute ‘capital_gain’ are ranged from 0 to 99999: divide them by 100000.
The values of attribute ‘capital_loss’ are ranged from 0 to 4356: divide them by 5000.
The values of attribute ‘hours_per_week’ are ranged from 1 to 99: divide them by 100.

After the data normalization in the datasets, with the same number of epochs (5000) and hidden neuron numbers (100) the classification accuracy increased from 83.08%(as shown in figure 5, the accuracy of the rprop) to 84.96%, the loss was reduced from 0.38(as shown in figure 6, the loss rate of the rprop) to 0.2524. The prediction accuracy in the training set was 88.08%.

Fig. 9. Loss & Accuracy for testing set and the confusion matrix given 27 features, Rprop, hidden neuron=100, epoch=5000.



```
Confusion matrix for training:

 23354    1366
  2520    5321
```

```
Confusion matrix for testing:

 11488     947
  1501    2345
```

To see the result, please run the data_normalised.py.

Data re-generalisation
It must be pointed out that improper generalization of data also causes loss of information and inaccuracy. For instance, the Nationality attribute has been simply split into 2 sub-classes: the US and non-US in previous steps. Because there are huge variances between individuals from non-US countries, such as Laos and France, allocating them to the same sub-class of the training data set might result in increased bias to the trained model.

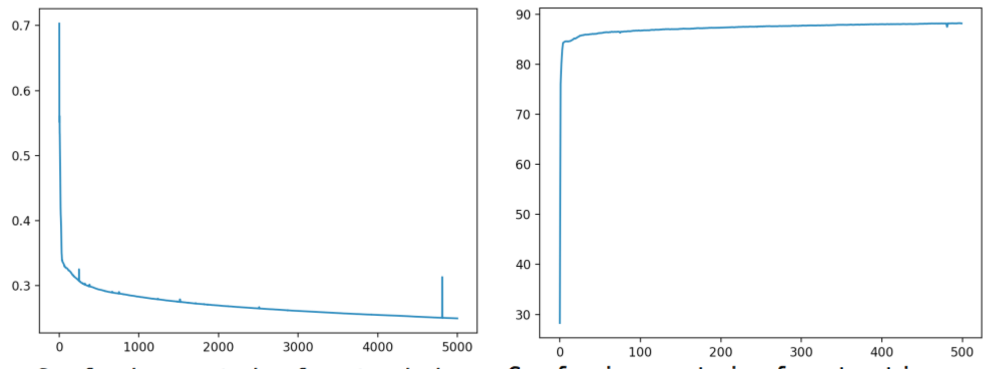
The original data set has 41 different values appeared in the column native_country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands. As the prediction target is financial-relevant, we could use geographic split together with categorization of national economic development level to make the final information extraction from the original native_country column.

Among them, United-States, Puerto-Rico, Outlying-US(Guam-USVI-etc), England, Canada, Germany, Japan, Greece, South (Intended South Korea in the dataset), Italy, Poland, Mexico, Portugal, Ireland, France, Taiwan, Scotland, Hong (intended Hong Kong in the dataset), Holand-Netherlands are defined as "advanced economies" by the International Monetary Fund (2011).

Language is an essential skill in the United States' domestic job market. Those whose first language are English tend to be more competent for high-income opportunities compared to their counterparts from non-english-speaking countries or regions. Hence, English-speaking country origin is very likely to be relevant of being high-income individual. United-States, England, Canada, Outlying-US(Guam-USVI-etc), Philippines, Jamaica, Ireland, Scotland, Trinadad&Tobago, Hong (Hong Kong) are English-speaking areas.

The re-generalised native_country forms 3 classes in total: the native_country_US, the native_country_advanced, and the native_country_eng_speaking. The optimized training dataset now has 29 feature attributes and 1 target attribute. After the re-generalisation of raw data, with the same number of epochs (5000), the classification accuracy increased from 84.96%(as shown in figure 5, the accuracy of the rprop) to 84.99%, the loss was reduced from 0.2524(the loss rate of the rprop) to 0.2499. The prediction accuracy in the training set was (23384+5337) / (23384+5337+1336+2504) =28721/32561=88.21%.

Fig. 10. Loss & Accuracy for testing set and the confusion matrix given 29 features, hidden neuron=120, epoch=5000.



```
Confusion matrix for training:    Confusion matrix for testing:

 23384    1336                     11506     929
  2504    5337                      1514    2332
```

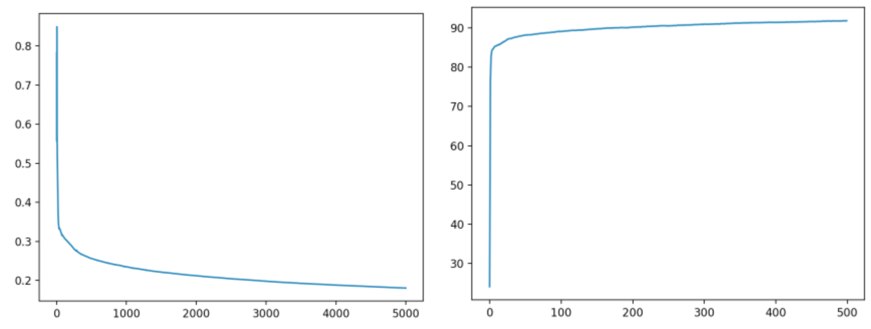
The contribution of adding new generalisation did not meet our expection. This might because the new features we added are not as fit the classification as other existing features in the dataset. We probably used less appropriate separations for the countries. Via the new classes, some countries such as India and China still have no proper expressions.

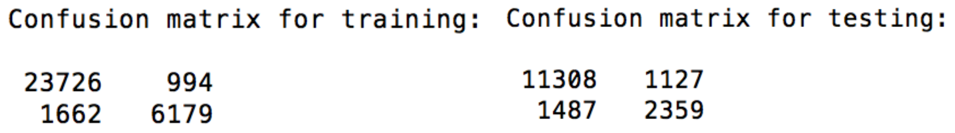
To see the result, please run the data_normalised_with_two_new_classes.py.

Data Augmentation and feature engineering

We mentioned that we have chosen to neglect most of the minor countries information in the column native_country. This might affect the effectiveness of the processed training data and harm the accuracy of the trained network. It is worthy to re-introduce all of them in new sub-classes to see what effect the relatively complete data set can make to the final loss rate and the accuracy. Later, a feature engineering process could be done to determine which attributes in the new training data set are the most redundant and whether it is reasonable to remove them and to achieve better training velocity for the neural network.
The similar steps have been taken for re-introducing column occupation in one-hot encoding pattern.

Fig.11. Loss & Accuracy for testing set and the confusion matrix given 83 features, Rprop, hidden neuron=300, epoch=5000.





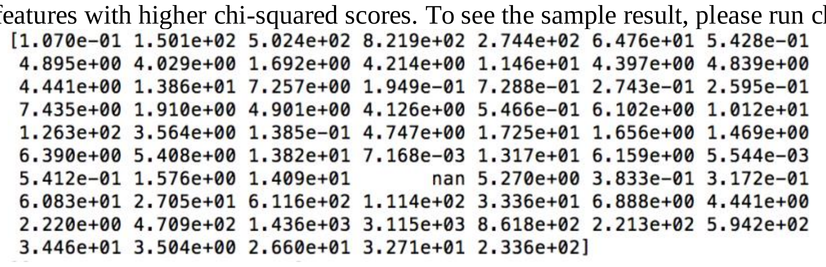
```
Confusion matrix for training:  Confusion matrix for testing:

23726    994                    11308   1127
 1662   6179                     1487   2359
```

The expanded training dataset now has 83 feature attributes and 1 target attribute. For a straight-forward comparison, the original processed dataset has 27 features. The increased features mean much longer time is required for same epochs of network training, with more input neurons and hidden neurons to be handled.
After the data's re-generalisation process in the datasets, with the same number of epochs (5000) and 300 hidden layer neurons, the classification accuracy for test set decreased from 84.96%(as shown in figure 5, the accuracy of the rprop) to 83.94%, however, the loss was reduced from 0.2524(as shown in figure 6, the loss rate of the rprop) to under 0.2. The prediction accuracy in the training set was (23726+6179) / (23726+6179+1662+994)=29905/32561=91.84%. Some sort of over-fit problem seems happened during the training process as the network got higher prediction accuracy in the training set compared to the results of the test set. The largely expanded dataset causes more missing values and subsequently more inaccuracy, as more missing values in the raw data become to take negative effect in the training procedure.

To see the result, please run the data_normalised_with_83_features.py.

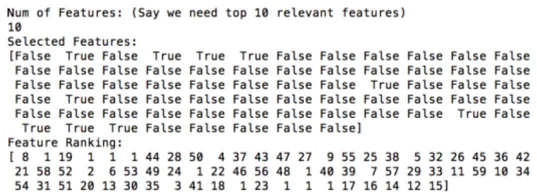
Guyon and Elisseeff (2003) indicated that feature engineering benefits the overall accuracy of machine learning models. By trimming down the feature vectors using their relevance to the target, the neural networks can often earn more than some other models such as random forest model. **In this paper, we used chi squared statistical test, RFE and genetic algorithm respectively for feature selection.**
The pool of candidate features (83 in total) as of previous model:
age,fnlwgt,education_num,sex,capital_gain,capital_loss,hours_per_week,occupation_Tech_support,occupation_Craft_repair,occupation_Other_service,occupation_Sales,occupation_Exec_managerial,occupation_Prof_specialty,occupation_Handlers_cleaners,occupation_Machine_op_inspct,occupation_Adm_clerical,occupation_Farming_fishing,occupation_Transport_moving,occupation_Priv_house_serv,occupation_Protective_serv,occupation_Armed_Forces,native_country_advanced,native_country_eng_speaking,native_country_United_States,native_country_Cambodia,native_country_England,native_country_Puerto_Rico,native_country_Canada,native_country_Germany,native_country_Outlying_US,native_country_India,native_country_Japan,native_country_Greece,native_country_South,native_country_China,native_country_Cuba,native_country_Iran,native_country_Honduras,native_country_Philippines,native_country_Italy,native_country_Poland,native_country_Jamaica,native_country_Vietnam,native_country_Mexico,native_country_Portugal,native_country_Ireland,native_country_France,native_country_Dominican_Republic,native_country_Laos,native_country_Ecuador,native_country_Taiwan,native_country_Haiti,native_country_Columbia,native_country_Hungary,native_country_Guatemala,native_country_Nicaragua,native_country_Scotland,native_country_Thailand,native_country_Yugoslavia,native_country_El_Salvador,native_country_Trinidad&Tobago,native_country_Peru,native_country_Hong,native_country_Holand_Netherlands,workclass_Private,workclass_Self_emp_not_inc,workclass_Self_emp_inc,workclass_Federal_gov,workclass_Local_gov,workclass_State_gov,workclass_Without_pay,workclass_Never_worked,relationship_Wife,relationship_Own_child,relationship_Husband,relationship_Not_in_family,relationship_Other_relative,relationship_Unmarried,race_White,race_Asian_Pac_Islander,race_Amer_Indian_Eskimo,race_Other,race_Black

Univariate selection: chi squared statistical test using sklearn (selection on 68-feature set):
Generally, we select features with higher chi-squared scores. To see the sample result, please run chi-squared.py.



```
[1.070e-01 1.501e+02 5.024e+02 8.219e+02 2.744e+02 6.476e+01 5.428e-01
 4.895e+00 4.029e+00 1.692e+00 4.214e+00 1.146e+01 4.397e+00 4.839e+00
 4.441e+00 1.386e+01 7.257e+00 1.949e-01 7.288e-01 2.743e-01 2.595e-01
 7.435e+00 1.910e+00 4.901e+00 4.126e+00 5.466e-01 6.102e+00 1.012e+01
 1.263e+02 3.564e+00 1.385e-01 4.747e+00 1.725e+01 1.656e+00 1.469e+00
 6.390e+00 5.408e+00 1.382e+01 7.168e-03 1.317e+01 6.159e+00 5.544e-03
 5.412e-01 1.576e+00 1.409e+01       nan 5.270e+00 3.833e-01 3.172e-01
 6.083e+01 2.705e+01 6.116e+02 1.114e+02 3.336e+01 6.888e+00 4.441e+00
 2.220e+00 4.709e+02 1.436e+03 3.115e+03 8.618e+02 2.213e+02 5.942e+02
 3.446e+01 3.504e+00 2.660e+01 3.271e+01 2.336e+02]
```

RFE (selection on the 68-feature set)
The Recursive Feature Elimination (RFE) recursively removes attributes and builds a model on those remaining attributes. It uses the model accuracy to identify which attributes (and combination of attributes) contribute the most to predicting the target attribute. The sample feature ranking below uses RFE with the logistic regression algorithm to select the top k ranked features. To see the sample result, please run RFE.py.



```
Num of Features: (Say we need top 10 relevant features)
10
Selected Features:
[False  True False  True  True  True False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False  True False False False
 False  True False False False False False False False False False False
 False False False False False False False False False False  True False
  True  True  True False False False False False]
Feature Ranking:
[ 8  1 19  1  1  1 44 28 50  4 37 43 47 27  9 55 25 38  5 32 26 45 36 42
 21 58 52  2  6 53 49 24  1 22 46 56 48  1 40 39  7 57 29 33 11 59 10 34
 54 31 51 20 13 30 35  3 41 18  1 23  1  1  1 17 16 14 12 15]
```

Genetic Algorithm(GA) for feature selection
The genetic algorithm (GA) is inspired from genetic crossover and mutation process. GA starts with a candidate feature set called population, and filter the least fit offsprings after each round of 'reproduction'. The filtering process is based on a fitness function. Compared to other feature selection algorithms, GA deal with large search spaces with high efficiency and is less likely to get local optimal result (Huang and Wang, 2006).
The following implementation is based on the python framework of DEAP GLOBAL VARIABLES toolbox with SCOOP (Hold-Geoffroy, Yannick, Gagnon, Olivier, Parizeau, and Marc, 2014) imported.

gen	nevals	avg	std	min	max
0	100	0.7896160.0268298		0.76737	0.847217
1	52	0.8194590.0219886		0.7685220.847217	
2	69	0.8317910.0166612		0.7685220.847601	

3	57	0.8397160.00879886	0.7696740.847601
4	60	0.8430770.00495449	0.8082530.848369
5	56	0.84461 0.00810718	0.7681380.84952
6	64	0.84529 0.0111864	0.7656430.850288
7	60	0.8466370.00812442	0.7685220.850288
8	59	0.8478980.00220977	0.8360840.850288
9	69	0.8480880.00801524	0.7696740.85048
10	49	0.8490650.00274018	0.8335890.850864

---Optimal Feature Subset(s)---

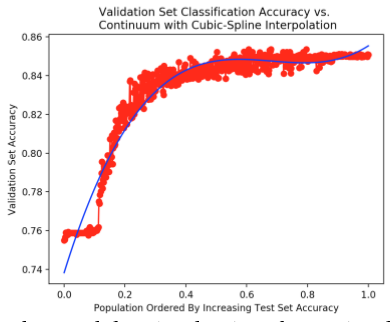
Percentile: 0.6069651741293532

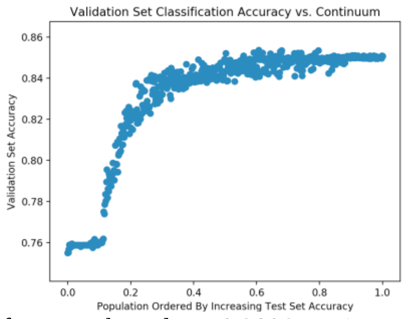
Validation Accuracy: 0.8536772608628896

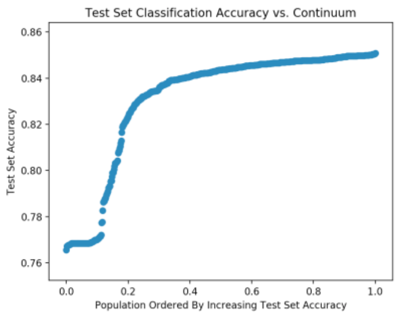
Individual: [0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0]

Number Features In Subset: 37

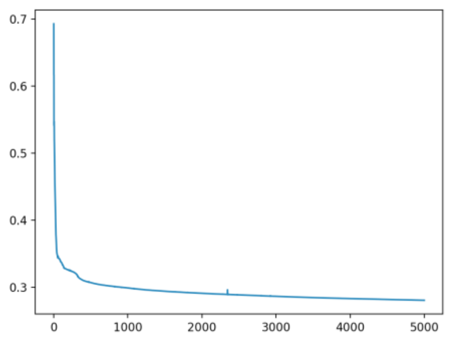
Feature Subset: ['age', 'education_num', 'capital_gain', 'occupation_Machine_op_inspct', 'occupation_Adm_clerical', 'occupation_Protective_serv', 'native_country_eng_speaking', 'native_country_United_States', 'native_country_England', 'native_country_Outlying_US', 'native_country_India', 'native_country_Iran', 'native_country_Philippines', 'native_country_Italy', 'native_country_Jamaica', 'native_country_Mexico', 'native_country_France', 'native_country_Laos', 'native_country_Ecuador', 'native_country_Taiwan', 'native_country_Haiti', 'native_country_Hungary', 'native_country_Thailand', 'native_country_Trinidad&Tobago', 'native_country_Hong', 'workclass_Self_emp_not_inc', 'workclass_Self_emp_inc', 'workclass_Federal_gov', 'workclass_Local_gov', 'workclass_Never_worked', 'relationship_Wife', 'relationship_Husband', 'relationship_Not_in_family', 'relationship_Other_relative', 'relationship_Unmarried', 'race_White', 'race_Other']

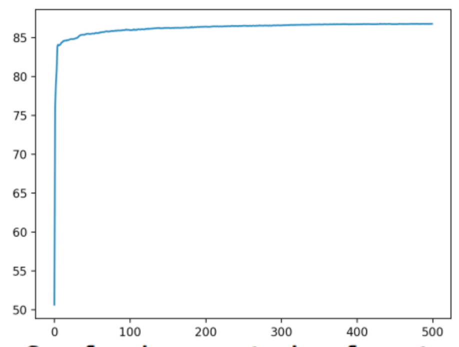






The model trained using the optimal feature subset: loss=0.2806, testing accuracy=84.71%. To see the result, please run the GA.py and the train_with_GA_opt.py.





```
Confusion matrix for training:

23313    1407
 2895    4946
```

```
Confusion matrix for testing:

11559     876
 1613    2233
```

Conclusion and Future Work

In this paper, we discussed different design and optimization methodologies of Neural network to settle the high-dimensional income classification task based in the Census Income data set. After the data transformation, data generalisation, data normalization process and the feature selection step using genetic algorithm, the testing accuracy finally increased from the original 76.38% to approximately 85% (84.99% with 0.2499 loss rate for 29-feature training set and 84.71% with 0.2806 loss rate). The performance of the optimal neural network is better than the Naïve-Bayes algorithm (accuracy between 83.5% and 84%) and basically matches the C4.5 decision tree algorithm (Accuracy between 85%-85.5%). This proved that our neural network implementation is acceptable to handle classification tasks according to the 1990s standards. Further improvement can be made in the data cleaning such as tackling missing values. In the future research, we will also focus on advanced neural network technology such as deep learning(DNNs) and reinforcement learning to obtain a broader view for solving similar issues.

References

1. Bustos, R. A., & Gedeon, T. D. (1995). Decrypting Neural Network Data: A GIS Case Study. In Artificial Neural Nets and Genetic Algorithms (pp. 231-234). Springer, Vienna.
2. Kohavi, R. (1996). Scaling up the accuracy of Naive-Bayes classifiers: a decision-tree hybrid. In KDD (Vol. 96, pp. 202-207).
3. Kohavi, R., & Becker, B. (1996). Census Income Data Set. UCI Machine Learning Repository. Retrieved 24 April 2018, from http://http://archive.ics.uci.edu/ml/datasets/Census+Income
4. Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010 (pp. 177-186). Physica-Verlag HD.
5. International Monetary Fund. (2011). World economic outlook. Retrieved 27 April 2018, from http://www.imf.org/external/pubs/ft/weo/2011/01
6. Guyon, I., & Elisseeff, A. (2003). An introduction to variable and feature selection. Journal of machine learning research, 3(Mar), 1157-1182.
7. Hold-Geoffroy, Yannick, Gagnon, Olivier, Parizeau, & Marc. (2014). Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment (p. 60). ACM.
8. Papamakarios, G. (2014). Comparison of Modern Stochastic Optimization Algorithms.
9. Riedmiller, M., & Braun, H. (1992). RPROP-A fast adaptive learning algorithm. In Proc. of ISCIS VII), Universitat.
10. Huang, C., & Wang, C. (2006). A GA-based feature selection and parameters optimizationfor support vector machines. Expert Systems With Applications, 31(2), 231-240. doi: 10.1016/j.eswa.2005.09.024.