Classifying People's policies

by using simple neural network

and improvement

Zhibo Zhang

1 Abstract

Nowadays, there are many voting in the world, and by using the result of voting, analyse it and get what we want is important. In this paper I am going to train a simple neural network to classify a person is democrat or republican by input a result of vote, I pre-process the data by change yes,no and ? to 2, 1,and 0, with output democrat and republican replaced by 1 and 2. Then I improve the neural network by changing the number of hidden neurons and epoch, validated by a k-fold cross-validate method to make it better. Finally, I use evolutionary algorithms genetic algorithms to select features I will use to make the model better.

2 Introduction

To find a person is democrat or republican directly is hard, but there are many factors have relationships with a person's position, so it can be predicted by getting other factors and predict a person is democrat or republican.

I use the data set from UCI, which is a records of congressional voting in United Stated in 1984, and there are 17 attributes in the data set , and 435 instances. There are 16 attributes that can influence the classification of democrat or republican. And there are only three classes in these attributes, such as yes, no and unknown, like in table 1.

Table 1: sample of data,it has 435 raws

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	republican	n	y	n	y	y	y	n	n	n	y	?	y	y	y	n	y
2	republican	n	y	n	y	y	y	n	n	n	n	n	y	y	y	n	?
3	democrat	?	y	y	?	y	y	n	n	n	n	y	n	y	y	n	n
4	democrat	n	y	y	n	?	y	n	n	n	n	y	n	y	n	n	y
5	democrat	y	y	y	n	y	y	n	n	n	n	y	?	y	y	y	y
6	democrat	n	y	y	n	y	y	n	n	n	n	n	n	y	y	y	y
7	democrat	n	y	n	y	y	y	n	n	n	n	n	n	?	y	y	y
8	republican	n	y	n	y	y	y	n	n	n	n	n	n	y	y	?	y
9	republican	n	y	n	y	y	y	n	n	n	n	n	y	y	y	n	y
10	democrat	y	y	y	n	n	n	y	y	y	n	n	n	n	n	?	?
11	republican	n	y	n	y	y	n	n	n	n	n	?	?	y	y	n	n
12	republican	n	y	n	y	y	y	n	n	n	n	y	?	y	y	?	?

I want to train the neural network, so the data set must be numeric, I do pre-processing to the data set. I replace 'yes' with 2, replace 'no' with 1, and replace '?' with 0, then replace 'republican' with 1 and replace 'democrat' with 0 to make it easy to train and test my neural network.

Table 2: sample of data set after pre-processing

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2	0	1	1	0	1	0	0	0	1	1	1	0	2	0	0	0	1	0
3	1	1	1	0	1	0	0	0	1	1	1	1	1	0	0	0	1	2
4	2	0	2	0	0	2	0	0	1	1	1	1	0	1	0	0	1	1
5	3	0	1	0	0	1	2	0	1	1	1	1	0	1	0	1	1	0
6	4	0	0	0	0	1	0	0	1	1	1	1	0	2	0	0	0	0
7	5	0	1	0	0	1	0	0	1	1	1	1	1	1	0	0	0	0
8	6	0	1	0	1	0	0	0	1	1	1	1	1	1	2	0	0	0
9	7	1	1	0	1	0	0	0	1	1	1	1	1	1	0	0	2	0
10	8	1	1	0	1	0	0	0	1	1	1	1	1	0	0	0	1	0
11	9	0	0	0	0	1	1	1	0	0	0	1	1	1	1	1	2	2
12	10	1	1	0	1	0	0	1	1	1	1	1	2	2	0	0	1	1
13	11	1	1	0	1	0	0	0	1	1	1	1	0	2	0	0	2	2
14	12	0	1	0	0	1	1	1	0	0	0	1	1	1	0	1	2	2

Then I use column B to R in table 2 as my input, and column A is the target, and with two output 1 or 2. To prevent overfitting, I use k-fold cross-validation, I divide the data set to k fold , and take one of them to be a test set, and rest of them to be train set. And I will do k times and take the average of accuracy of k times to get an average accuracy.

For the neural network, there are three layers, input layer, hidden layer and output layer. There are 16 input neurons and 20 hidden neurons as a basic number, and two output neurons. I input 16 attributes from the data set, named handicapped-infants, water-project-cost-sharing, adoption-of-the-budget-resolution, physician-fee-freeze, el-salvador-aid, religious-groups-in-schools, anti-satellite-test-ban, aid-to-nicaraguan-contras, mx-missile, immigration, synfuels-corporation-cutback, education-spending, superfund-right-to-sue, crime, duty-free-exports, export-administration-act-south-africa, with the value 0,1,2 in it , and two output named class name with value 0 and 1.

And to improve my neural network, I make the number of hidden neurons increase to find out whether it makes my model better, and I find it indeed improve my model.

3 Method

3.1 Sample Neural Network

I build a sample neural network to classify the people who participates the vote, I set input neurons 16 and 20 hidden neurons as a basic number, set epoch number with 500, and learn rate is 0.01.Then I build the neural network with linear hidden layer output and output layer output. Then I set the activation function with sigmoid function, cross-entropy loss as loss function, and set optimiser with Stochastic Gradient Descent(SGD) algorithm.

Then I start training my neural network with 80% data as a train set ,and 20% data as a test set. For accuracy I disrupt the order of original dataset to make sure that my data is randomly select from the original dataset. For epoch equals to 500, every time it will be trained 500 times. Also during training, I compare result of prediction with the value in training set to see progress.

Then I use test set to test my neural network and I get a test accuracy which is 58.62%, it can be seen that the neural network doesn't work very well with 20 hidden neurons.

3.2 Improve Neural Network

To improve neural network, I first increase the number of epoch, because I set 500 as a basic number, I increase it to 1000, and I find it does better when hidden number is 20 and epoch number is 1000, the accuracy of test is about 88.51%, which is much higher than epoch number is 500.

To improve neural network, I also try increase the number of hidden neurons from 20 to 40, and show the result in Table 4, and it shows a better performance on the train accuracy.

And I use this neural network to test on my test set , and I get accuracy 90.80%.Obviously, the number of hidden neurons does contribution to train neural network.

To find more relationship between number of hidden neurons and the accuracy, I increase the number step by step, I use 20 as a basic number every time add 5 to it and record the accuracy.

3.3 K-fold Cross-validation

In last part, I find sometimes there is relationship between the number of hidden neurons and accuracy, but sometimes there is no relationship, I guess when I train the neural network, it maybe influenced by the overfitting problem. So I use cross-validation to avoid the problem.

Cross-validation is a simple technique for model validation, it divides the set to two parts, train set and test set, then train and test to avoid overfitting, and k-fold cross-validation is to divides the set to k set, and one of them is used to test, k-1 sets are used to train, repeat k times, every time choose different set as a test set. With a different k, there are differences in result (Kohavi,1995).

First, I set k equals to 5, and I divide the data set to 5 data set, to make it randomly, I use sample function and set frac equals to 1, to change the order in data set randomly. Then I just take same number of data according to the index in data set. In the cross validation processing, first I take the first data set I divide and use it as test set, then take rest of data use as a train set, for example, I divide the set to 5 and number it 1, 2, 3, 4, 5, first time I will take number 1 set as test set and 2,3,4,5 as train set.

Then I use train set for training and test it by test set, and record the accuracy of test, next I set number 2 as a test set and 1, 3, 4, 5 as train set, train and test, record the accuracy. Finally I take average of accuracy as accuracy for this neural network.

3.4 Evolutionary algorithms

Now I have found that with the number of hidden neuron 50 and 5-fold cross-validation, there will be a better result. Then I will use evolutionary algorithms, genetic algorithms to make a feature selection, to choose which feature I will use to train the neural network.

I set a matrix with 0 and 1 to judge which feature I will use, 1 means using and 0 means not using. With setting the matrix same length with the number of features, my way to judge feature through matrix is to delete the column by finding the number on the index of column is 1 or 0, if it is 0 in matrix then delete the column. For example, if my matrix is [1,0,1,0,0], then I will delete the column 1, 3, 4 in the data, because the column in data start from 0.

To do the genetic algorithms, I set DNA size equal to the number of features, pop size which is matrix size 10, and cross rate 0.8, the max generations 10.

I initialize the matrix randomly, to make sure than I choose features randomly first, and I set the matrix size with 10, which means the matrix is 16 columns and 10 rows. Then I train the neural network with features after selection, then get the test accuracy and record it with an array. Then I get the max of this accuracy array, and get the index, with this index get the feature selection, named it with most fitted DNA. This is elitist selection in genetic algorithms, every time only choose the best result and DNA.

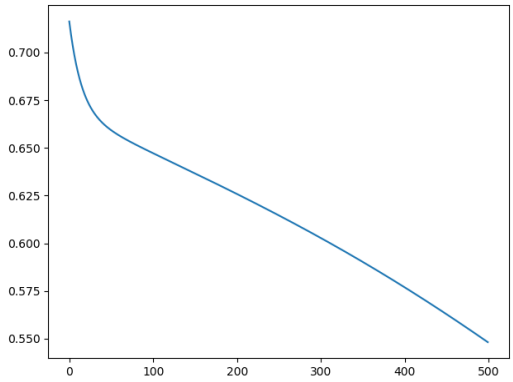
Then do the select function, which sets this features selection in matrix with a high possibility to choose by setting (accuracy / sum of accuracy). Then do crossover and mutation to update the pop matrix. This process will do n generation times which I set it 10 before.

Finally, we can get the most fitted features selection and the best test accuracy.

4 Results and Discussion

For the simple neural network, I just set 20 as the number of hidden neurons, 500 as the number of epoch, 80% of set as train set, and 20% of set as test set. Then during the processing of training , record the loss and accuracy per 50 times training, and show it in Diagram 1.

Diagram 1: The relationship between the number of epoch and loss and the progress of training

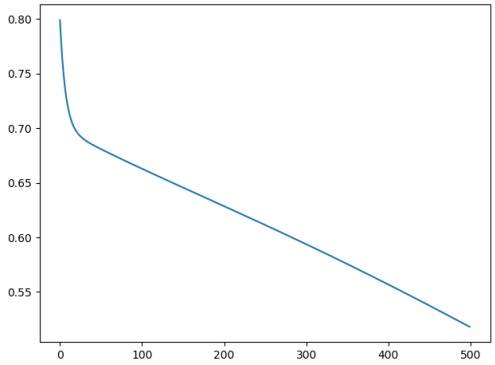



```
Epoch [1/500] Loss: 0.7689  Accuracy: 38.79 %
Epoch [51/500] Loss: 0.6874  Accuracy: 61.21 %
Epoch [101/500] Loss: 0.6719  Accuracy: 61.21 %
Epoch [151/500] Loss: 0.6581  Accuracy: 61.21 %
Epoch [201/500] Loss: 0.6446  Accuracy: 61.21 %
Epoch [251/500] Loss: 0.6311  Accuracy: 61.21 %
Epoch [301/500] Loss: 0.6174  Accuracy: 61.21 %
Epoch [351/500] Loss: 0.6033  Accuracy: 61.21 %
Epoch [401/500] Loss: 0.5887  Accuracy: 61.21 %
Epoch [451/500] Loss: 0.5735  Accuracy: 62.07 %
```

It can be seen that the accuracy is not very high is just 62.07 after training, and the loss decreases not fast.

So, to improve it, first I change the number of hidden neurons from 20 to 40 to see what happen and show it in Diagram 2.

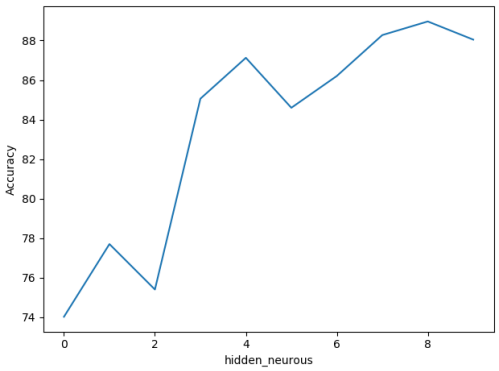
Diagram 2 : Process after increase number of hidden neurons




```
Epoch [1/500] Loss: 0.6931  Accuracy: 52.59 %
Epoch [51/500] Loss: 0.6475  Accuracy: 60.92 %
Epoch [101/500] Loss: 0.6264  Accuracy: 60.92 %
Epoch [151/500] Loss: 0.6057  Accuracy: 60.92 %
Epoch [201/500] Loss: 0.5848  Accuracy: 61.21 %
Epoch [251/500] Loss: 0.5638  Accuracy: 62.36 %
Epoch [301/500] Loss: 0.5424  Accuracy: 72.13 %
Epoch [351/500] Loss: 0.5207  Accuracy: 83.91 %
Epoch [401/500] Loss: 0.4990  Accuracy: 86.21 %
Epoch [451/500] Loss: 0.4773  Accuracy: 90.52 %
```

It can be found that the increase of the number of hidden neurons does contribution to improve the neural network. To find more, I increase hidden neurons' number step by step, I set 20 as a basic number, add 5 every time, and record the accuracy, and show it in diagram 3.

Diagram 3: relationship between hidden neurons and accuracy



(*0 means hidden neurons is 20 and every time add 5, for example 2 means (20+2*5) = 30*)

From the diagram 3 it can been found that with add number of hidden neurons, the accuracy is increasing, but sometimes it decreases, maybe it is influenced by the overfitting of data, or it is influenced by the float of accuracy.

And to improve it, I use k-fold cross-validation to decrease the influence of float, and overfitting. With k = 5 in k-fold, every time record the accuracy after training, then take the average accuracy, make plot shows avg_accuracy and the number of hidden_neurons in diagram 3.

Diagram 3: After using K-fold Cross-validation K =5

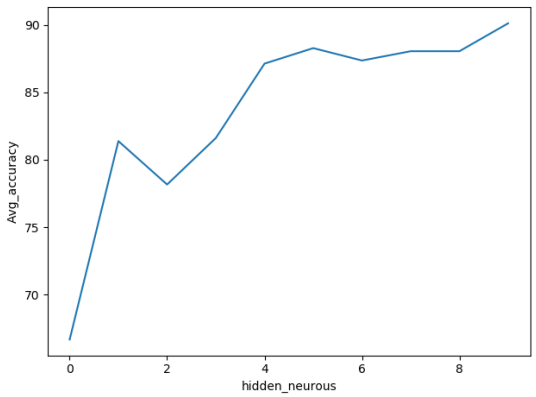
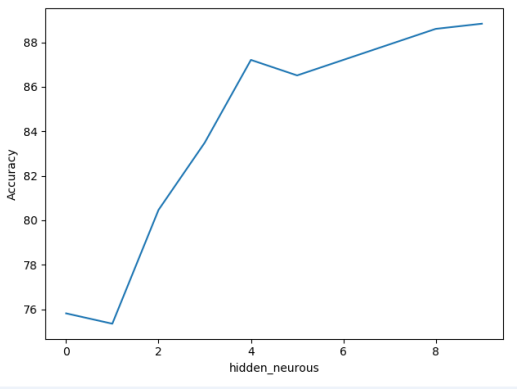


Diagram 4: After changing K = 10



There is less floating in the diagram, and to improve it more I set k =10 and shows in diagram 4.

Obviously, increasing k make accuracy less float during increasing the number of hidden neurons, and increasing hidden neurons makes contribution to accuracy in generally.

I choose k-fold k =5, the number of hidden neurons is 50 and do genetic algorithms to select feature, and I get the result in Diagram 4.

Diagram 4 : Result after selecting features through genetic algorithms.

```
Final Most fitted DNA:  [1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0]
Final accuracy:88.74 %
```

5 Conclusion and Future Work

I have used the result of Congressional Voting Records from UCI Machine Learning Repository (Jeffm ,1978), which is a record of Congressional Voting in United Stated in 1978.The data has been used to classified people's policies, using a simple neural network, and improve it with increasing the number of epoch and hidden neurons, and also use k-fold cross-validation to avoid overfitting.

Obviously, an increase on the number of epoch make contributions to improve neural network, and I can also find that increase the number of hidden neurons sometimes rise the accuracy, but with avoiding overfitting, the number of hidden neurons does contribution to improve network. And, with an increasing of k in k-fold validation, neural network is also improved.

I showed a sample neural network which classifies people's policies from Congressional Voting Records, and to improve it, use cross-validation technique, keep the correct classification results as high as possible. I used evolutionary algorithms, genetic algorithms to select the features which is most fitted with this model.

The next stage of my work will change the number of k-fold to make model much better, and because it will take a long time to train neural network, which leads to that parameter in my genetic algorithms

can not be too large, because it will take a very long time to finish selection. So the next work maybe optimize my algorithms in program to make it go faster. And with larger parameter, the selection model will be better, maybe using RNN from deep learning algorithms will work.

6 References

1. Kohavi, R. (1995, August). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*(Vol. 14, No. 2, pp. 1137-1145).
2. Setiono, R., & Liu, H. (1997). Neural-network feature selector. *IEEE transactions on neural networks*, *8*(3), 654-662.
3. Milne, L.K., Gedeon, T.D. and Skidmore, A.K. (1995) "*Classifying Dry Sclerophyll Forest from Augmented Satellite Data: Comparing Neural Network, Decision Tree & Maximum Likelihood*," Proceedings Australian Conference on Neural Networks, Sydney, 160-163.
4. C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.