Project4: Learning

Xuejian Ye

December 12, 2018

Table of Contents

1. Project Summary

This project aims to construct two learning algorithms to do classification tasks.

a. Language: Python 3

b. Individual work

c. How to run project:
   1) Run the code in terminal and it will show the result. (More details in README)
   2) Decision tree: test the algorithm with two datasets
      (Iris flower/ Restaurant data)
   3) Neural network: test the algorithm with two datasets
      (Iris flower/Car Evaluation data)

d. Notes for my project
   1) For the dataset, I have combined the describe file and dataset file as one which I could input the dataset more convenience. (Just add attribute name or column name into dataset file, please use the data file in my RAR)
   2) For the Neural Network model. Because it needs take more time to train model, so I use the parameters after tuning to get the best weight and save as a weight file.
      python3 nn.py -d iris.discrete.txt -w iris_weights_7.txt
      python3 nn.py -d car_evaluation.txt -w car_weights_8-6.txt -hl 8-6
      And it could be used to test data. (more details show in README)

2. Decision tree

2.1 Introduce:

It is a decision support tool that uses a tree-like model of decisions and their possible consequences, trying to find rules consisting of the sentences (if and then) to make the decision that to which class a target with specific attributes belongs. We can find an attribute which can decrease entropy biggest to be the first attribute which used for splitting data, then go deeper to find out the next best attribute to split data. Finally, process will be completed when it ends with no more attribute could be used for splitting. In order to select the best attribute correctly, we use the information gain to make the decision, so it is intuitional to structure the code with this method.

2.2 Result analysis:

a. For Decision tree algorithm, we use two datasets: discrete Iris flower data and Restaurant data. And the accuracy depends on the degree of splitting data.
b. For Iris flower discrete dataset, the accuracy of result:

```
[xuejianye|Wed Dec 12|05:49:38]~/Desktop/project4_final $ python3 dt.py -d iris.txt
{'petal-width': {'MS': 'Iris-versicolor', 'L': 'Iris-virginica', 'S': 'Iris-setosa', 'ML': {'petal-length': {'MS': 'Iris
-versicolor', 'L': 'Iris-virginica', 'ML': {'sepal-length': {'MS': {'sepal-width': {'MS': 'Iris-versicolor', 'S': 'Iris-
versicolor', 'ML': 'Iris-versicolor'}}, 'L': 'Iris-versicolor', 'S': 'Iris-virginica', 'ML': {'sepal-width': {'MS': 'Iri
s-versicolor', 'S': 'Iris-versicolor', 'ML': 'Iris-versicolor'}}}}}}}}
accuracy: 95.33333333333334%
```

The accuracy is 95.34%

c. For Restaurant dataset. First, we use information gain to select the attribute mentioned in AIMA and the tree show below.

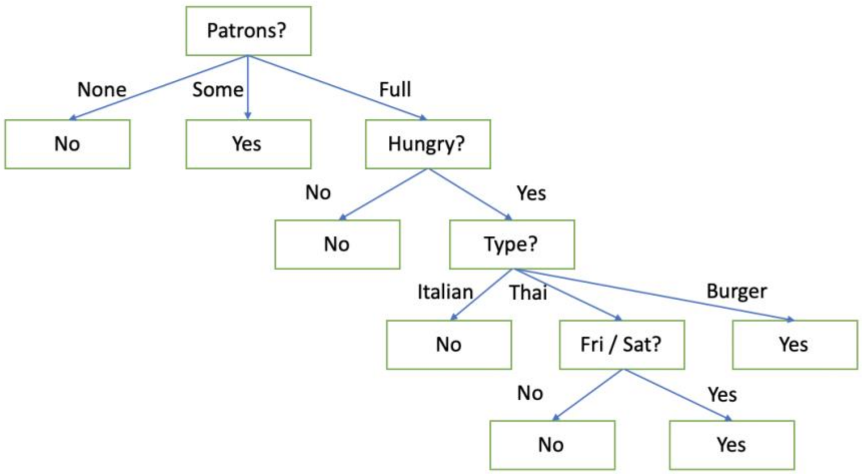

Figure.1 tree for Restaurant dataset

```
[xuejianye|Wed Dec 12|05:46:49]~/Desktop/project4_final $ python3 dt.py -d restaurant.txt
{'Pat': {'Full': {'Hun': {'No': 'No', 'Yes': {'Type': {'Thai': {'Fri': {'No': 'No', 'Yes': 'Yes'}}, 'Burger': 'Yes', 'It
alian': 'No'}}}}, 'None': 'No', 'Some': 'Yes'}}
accuracy: 100.0%
```

The accuracy is 100%

3. Neural Network

3.1 Introduce:

Neural network is kind of magic in learning and computing systems vaguely inspired by the biological neural networks that constitute animal brains. It constructs a network like neurons in brain and change its connection weights from training data to learn implicit knowledge under data. Then constructing model and training the model, but we need to tune the different parameters to find out the best parameters which optimize the model with best performance. Finally, using the model with best performance to predict the test data.

3.2 Evaluation of Neural Network model

(based on Iris discrete dataset)

a. Results of implementing algorithm on two datasets:

1) Iris.discrete.data.txt:

```
[xuejianye|Wed Dec 12|08:51:38]~/Desktop/project4_final $ python3 nn.py -d iris.discrete.txt -w iris_weights_7.txt
model layers: [4, 7, 3]
trained model accuracy: 90.0%
```

Clear Introduction for tuning parameters shows in evaluation part (based on Iris discrete dataset)

2) Car_Evaluation.txt:

```
[xuejianye|Wed Dec 12|09:14:31]~/Desktop/project4_yxj $ Python3 nn.py -d car_evaluation.txt -w car_weights_8-6.txt -hl 8
-6
model layers: [6, 8, 6, 4]
trained model accuracy: 91.55092592592592%
```

With learning rate=0.1, epoch=5000 and two hidden layers (8 and 6 nodes respectively), the training data has 91.55% accuracy while the testing data has 95.4% accuracy.

b. Comparison between different learning rate
(Epoch = 1000, cross validation = 3)

Learning rate	0.01	0.05	0.1	0.15	0.2	0.25	0.3
Train data accuracy	86.33%	89.17%	90.83%	89.00%	90.33%	90.33%	91.33%
Test data accuracy	86.00%	84.67%	82.00%	88.00%	86.67%	86.00%	84.00%

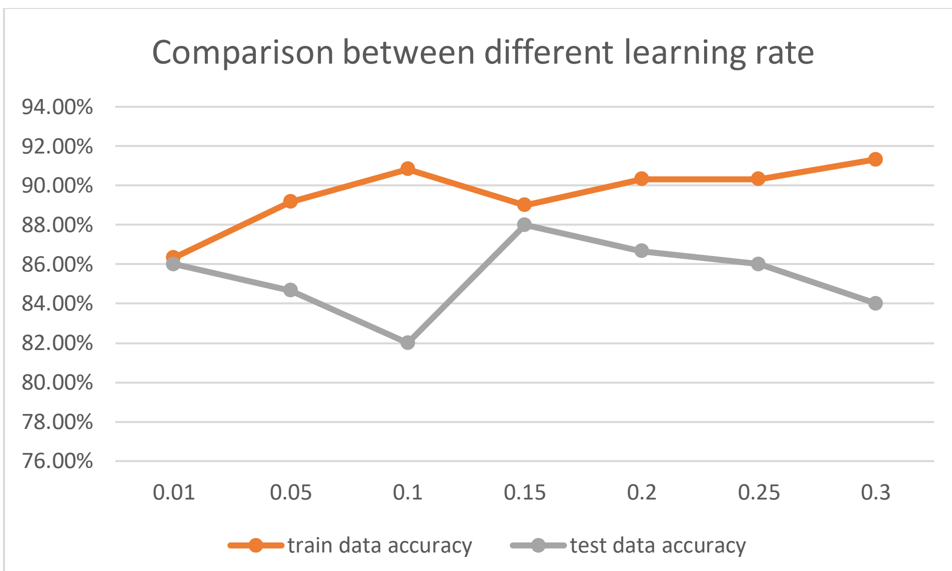

Figure2. Comparison between different learning rate

By comparing different learning rate with same epoch times and cross validation. We find out that learning efficiency is a tradeoff between learning rate and time. Learning time might be short if the epoch is not big enough and we should find the best learning rate which will bring the reasonable model performance. In the table and figure above, with the learning rate going up, the performance on training data becomes better. However, on test data, we notice that when learning rate is between 0.15 and 0.25, model has best performance, so if the learning rate is too large means that the moving pace is too large to get close to the befitting point for the model.

c. Comparison between different epoch number
(learning rate=0.01, 0.05 0.1 and 0.2, cross-validation=3):

Training data							
learning rate	0.01	0.05	0.1	0.15	0.2	0.25	0.3
epoch 1000	86.33%	89.17%	90.83%	89.00%	90.33%	90.33%	91.33%
epoch 5000	89.16%	92.00%	91.50%	90.83%	92.33%	90.33%	89.16%
Testing data							
learning rate	0.01	0.05	0.1	0.15	0.2	0.25	0.3
epoch 1000	86.00%	84.67%	82.00%	88.00%	86.67%	86.00%	84.00%
epoch 5000	88.00%	89.33%	86.66%	84.00%	84.66%	86.00%	87.33%

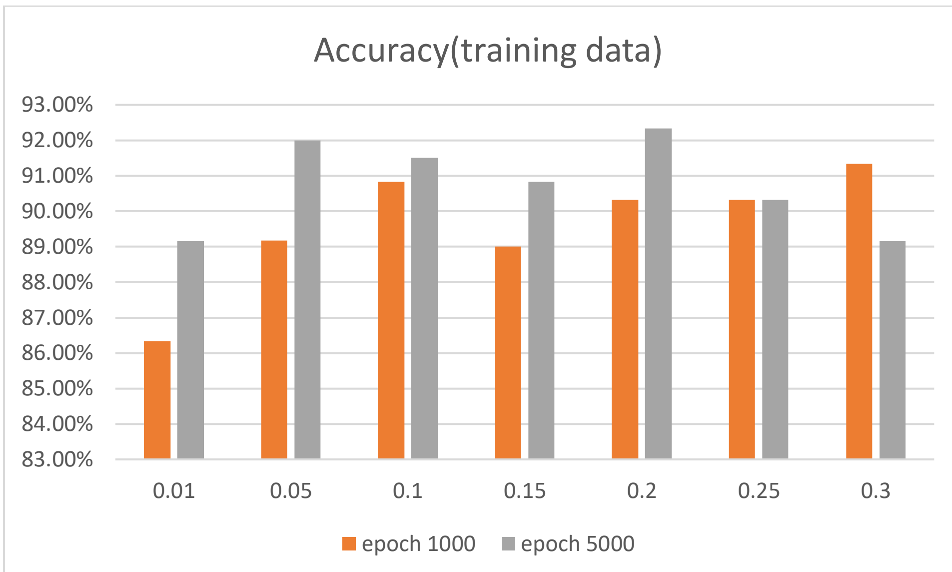

Figure2. Accuracy on training data

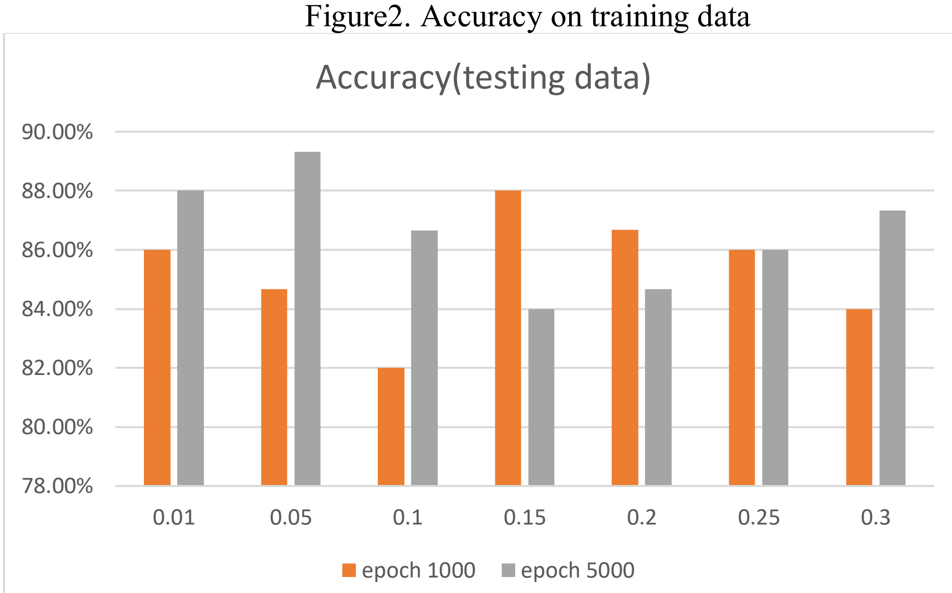

Figure3. Accuracy on testing data

In the Figure 2 and 3, basing on the condition (small learning rate), the model will have much better performance with high epoch. For example, when learning rate between 0.01 and 0.1, the more epoch we took, the better accuracy we got, however, with the increasing of learning rate, the benefit brought by epoch times will decrease gradually.

d. Comparison between different k-fold Cross validation
(learning rate = 0.2, epoch = 1000)

data size ratio	0.9(k=10)	0.8(k=5)	0.75(k=4)	0.5(k=2)
train data accuracy	89.34%	91.00%	89.67%	88.34%
test data accuracy	84.00%	84.67%	86.17%	87.34%

In this part, default values of learning rate and epoch is 0.2 and 1000 and we notice that the accuracy of model on training data is similar because the trainings on model are same with different k-fold cross validation. However, the accuracy on testing date is not similar because the size of dataset is small, we should choose a right ratio to split data, if the training data is very large, the model has high probability to overfit and bad generalization on those data. This problem could be smoothed if the data size is large enough, but for the Iris dataset, the ratio is very important. Finally, I choose 3 for cross validation

e. Comparison between different network structure

I try to add more hidden layers to check the change of model performance and it is reasonable that the more complex model could get better performance. In order to reach the reasonable time-consuming target, I only choose one hidden layer with seven nodes which also could get a good accuracy for Iris dataset. As for Car Evaluation dataset, we construct two hidden layers with eight and six nodes which also get a good accuracy on both training data and testing data. But I am sure that the model could be better with different number of hidden layers and nodes.