Introduction to Data Mining: 21191441
Zhejiang University — Dec 14, 2020
Professor Deng Cai — Homework 4

Homework 4

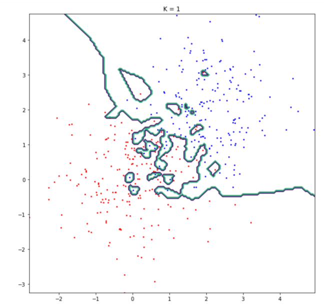
Collaborators:
Name: Chen Zhibo
Student ID: 3190100923

Problem 4-1. K-Nearest Neighbor

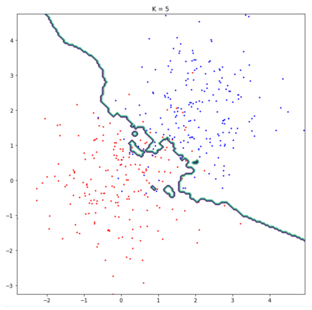
In this problem, we will play with K-Nearest Neighbor (KNN) algorithm and try it on real-world data. Implement KNN algorithm (in knn.py), then answer the following questions.

(a) Try KNN with different K and plot the decision boundary.

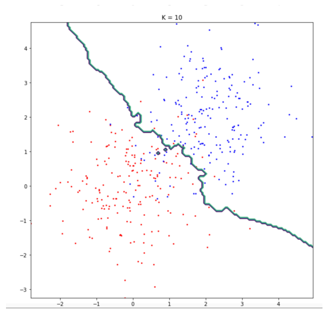
Answer:



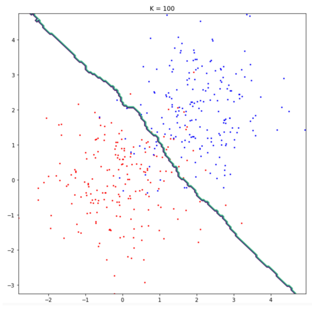
(a) k=1



(b) k=5



(c) k=10



(d) k=100

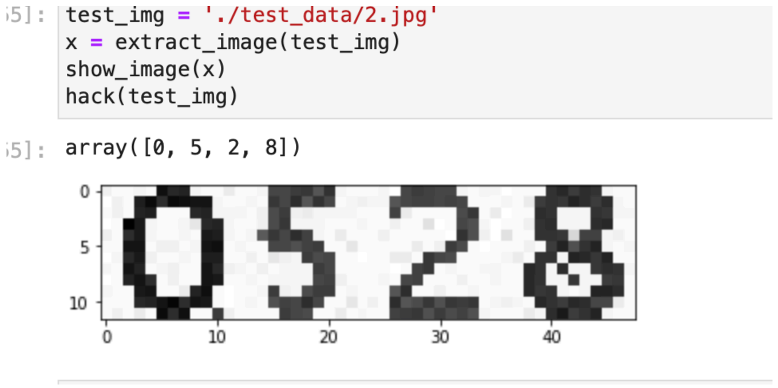
Figure 1: imag of KNN

(b) We have seen the effects of different choices of K. How can you choose a proper K when dealing with real-world data ?

Answer: We can firstly try some numbers in a large range, then pick the one with relatively closer boundary to the expected boundary. After that try numbers in a small range around that specific point and pick the k with best performance.

(c) Finish hack.py to recognize the CAPTCHA image using KNN algorithm.

Answer:



```
[5]: test_img = './test_data/2.jpg'
     x = extract_image(test_img)
     show_image(x)
     hack(test_img)
```

```
[5]: array([0, 5, 2, 8])
```

Figure 2: CAPTCHA

Problem 4-2. Decision Tree and ID3

Consider the scholarship evaluation problem: selecting scholarship recipients based on gender and GPA. Given the following training data:

Answer:

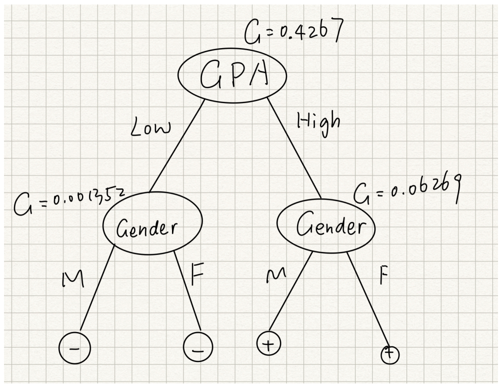


Figure 3: Figure of DT

Problem 4-3. A Walk Through Ensemble Models

In this problem, you will implement a whole bunch of ensemble models and compare their performance and properties.

(a) Implement decision tree algorithm (in decision tree.py), then answer the following questions.

Answer:

1. The visualization of the 2 depth DT is shown below:

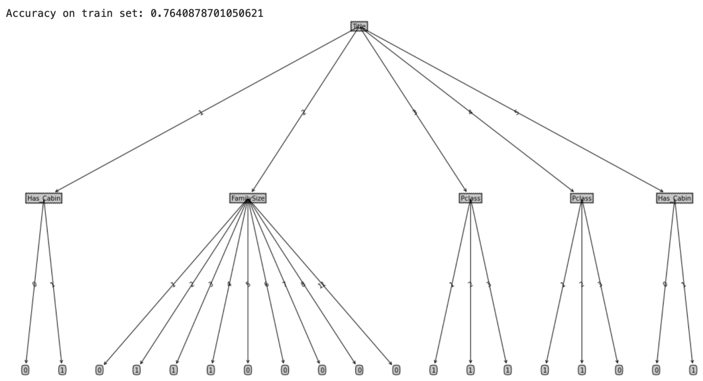


Figure 4: DT of depth 2

2. (a1) Title=1
   (b1) Has_Cabin=1
   (c1) Survived

   (a0) Title=2
   (b0) FamilySize=5
   (c0) Unsurvived

3. The larger the max_depth is , the more possible for the tree to be overfitting; and the mini_leaf makes it less possible to overfit as it grows larger. So get the max_depth and min_leaf both at a relatively large size will get a good performance.

4. Training Error: 14.14%
   Validation Error: 19.47%
   There is a 5% difference between error rate of training and validation, and by trying I found that it's not possible to further reduce the training error. So accordingly it's still overfitting.

5. max_depth = 8
   min_sample_leaf = 4
   test accuracy = 80.53%

6. It is very easy to overfit when max depth is big. But the min sample leaf number will eliminate the impact of max depth.
   The decision tree algorithm does not perform very well on this dataset. No

matter how I change the hyperparameters, it still get little advance on the accuracy.

(b) Implement Random Forest (in random forest.py), then answer the following questions.

Answer:

1. The max depth is greatly smaller compared with a single decision tree. It is a wierd fact because in theory random tree is designed to reduce the variance of a module. So it's ok for a base learner to have a high varience and overfitting. But the optimal parameters I trained turns out to be opposite, it's underfitting. Maybe its because of the dataset may be not that fit with this algorithm.

2. 77.5% accuracy on the 10 forest
   81.3% accuracy on the 100 forest

3. Random forest is designed to reduce the variance of several base models by taking the average of them. But it can do little to the bias. So theorically by adding the max depth and reduce the min leaf sample number will improve the performance though it may cause overfitting on a single tree.

4. with parameter of max_ depth=3, min_ samples_ leaf=5
   final test accuracy: 81.3%

(c) Implement Adaboost (in adaboost.py), then answer the following questions.

Answer:

1. Because adaboosting is a linear process , each base learner have to be trained one after one. So it has a large consumption on time . With decision stump it can greatly reduce the time of training. And by updating the weight can eliminate the bias small depth caused as long as we have a enough number of estimator.

2. n_ estimator=10
   training error= 25.6%
   testing error= 23.3%

   n_ estimator=100
   training error=21.2%
   testing error=19.1%

3. It update weight of each decision tree and weight of data by estimating its prediction. It will become more focus on the data that it mislabeld and will adjust the weight of each DT accordingly.
   It reduce the bias.
   We can find from the training that less max depth of each tree will get a relatively high accuracy.

4. max_depth=2 , min_sample_leaf= 50 n_estimators=50
   testing accuracy = 80.9%