

DECISION TREE FOR COMPUTER SIMULATION AND SPAM DATASET

Fan Zhang

STATISTICAL LEARNING

Content

1. INTRODUCTION.....	3
2. EXPERIMENT.....	4
2.1 The Implement of Decision Tree.....	4
2.2 Feature Selection Method.....	6
2.3 Cross Validation.....	8
2.4 Simulation of Training Data.....	9
3. RESULT AND DISCUSSIONS.....	11
3.1 Classification Error and The Number of Features.....	11
3.2 Classification Error and The Number of Training Data Samples.....	12
3.3 Classification Error and The Number of Relevant Feature.....	13
3.4 Classification Error and The Feature Selection Method.....	15
3.5 Classification Error and The Noise Level.....	16
3.6 Performance on Spam Data Set With and Without Feature Selection.....	18
4. CONCLUSION.....	20
REFERENCE.....	20

1. INTRODUCTION

Project description: Implement the decision tree classifier with and without a feature selection method, like COV, GINI, ENTROPY, etc. Use the decision tree classifier on the manual simulation data set, which satisfies the Gaussian distribution. Investigate the performance of this classifier and how the classification error is affected by noise level, feature number p , related feature number p_1 , training data sample number n and feature selection method. And run the decision tree classifier on a real data set in order to compare the results and the classifier's performance on two different data sets with and without a feature selection method.

2. EXPERIMENT

2.1 The Implement of Decision Tree

First of all, let me just briefly clarify what is a decision tree. The classifier may do better choosing the attributes one at a time, according to the demands of the situation. The most popular tool targeting this scenario is a decision tree. Take an simple example:

Example	Crust-size	Shape	Filling-size	Class
E1	big	circle	small	pos
E2	small	circle	small	pos
E3	big	square	small	neg
E4	big	triangle	small	neg
E5	big	square	big	pos
E6	small	square	small	neg
E7	small	square	big	pos
E8	big	circle	big	pos

Table 2.1 Eight training examples described by three symbolic attributes

The training set shown in Table 2.1 consists of eight examples described by three attributes and labeled as positive or negative instances of a given class. To simplify the explanation of the basic concepts, we can assume that all attributes are discrete. In fact, the generation of decision tree is to select these attributes in turn to form its own nodes. When the final conclusion can be clearly drawn (i.e., leaf node), the whole tree is generated^[1]. So, here we generate two of the decision trees in table 2.1.

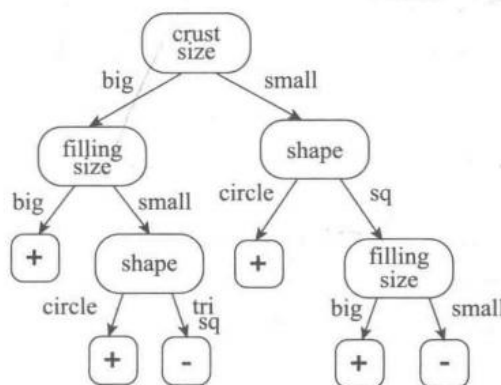


Fig 2.1 One of example decision trees for the table 2.1

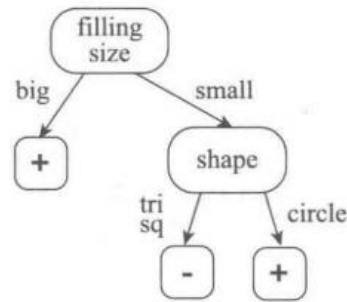


Fig 2.2 One of example decision trees for the table 2.1

Figure 2.1 and 2.2 show two example decision trees that are capable of dealing with the data from Table 2.1. The internal nodes represent attribute-value the edges indicate how to proceed in the case of diverse test results, and the leaves contain class labels. An example to be classified is first subjected to the test prescribed at the topmost node, the root. The result of this test then decides along which edge the example is to be sent down, and the process continues until a leaf node is reached. Once this happens, the example is labeled with the class associated with this leaf^[1].

```

list_accuracy=[]
kf = KFold(n_splits=10)
flag=0
for train, test in kf.split(dataset):
    flag += 1
    print(flag)
    print("K fold division: %s %s" % (train.shape, test.shape))
    X=pd.DataFrame()
    Y=pd.DataFrame()
    for i in range(len(train)):
        X=X.append(dataset.iloc[train[i],:],ignore_index=True)
    for i in range(len(test)):
        Y=Y.append(dataset.iloc[test[i],:],ignore_index=True)
    X_train=X.iloc[:,0:(df.shape[1]-1)]
    X_test=Y.iloc[:,0:(df.shape[1]-1)]
    y_train=X.iloc[:,(df.shape[1]-1)]
    y_test=Y.iloc[:,(df.shape[1]-1)]
    print("Training set size: ", X_train.shape, y_train.shape)
    print("Test set size: ", X_test.shape, y_test.shape)
    mode = tree.DecisionTreeClassifier(criterion='entropy', splitter='random', max_features=1)
    #mode = tree.DecisionTreeClassifier(criterion='entropy', splitter='best') # use entropy/gini for feature selection
    # mode = tree.DecisionTreeClassifier(splitter='random', max_features=1)
    #mode = tree.DecisionTreeClassifier(criterion='gini', splitter='best') # use entropy/gini for feature selection
    mode.fit(X_train, y_train) # Use the training set to train the model
    score=mode.score(X_test, y_test)
    print('Accuracy: ', score) # Calculate the measurement of test set (accuracy)
    list_accuracy.append(score)
    print("-----")
    dot_data = tree.export_graphviz(mode, out_file=None)
    graph = pydotplus.graph_from_dot_data(dot_data)
    graph.write_pdf("test"+str(flag)+".pdf")
print('Average Error rate: ', 1-np.mean(list_accuracy))
  
```

Fig 2.3 The implement of decision tree, feature selection method and 10-fold cross validation

Here, I choose python to implement the decision tree classifier. Thus, as you can see in the figure 2.3, I can use sklearn library to import tree and use the “tree.DecisionTreeClassifier()” to construct our decision tree, since the

implementation and operation of the sklearn's decision tree are based on the construction and operation principles of the decision tree described above. However, there's a problem. Different order of selection of attributes tends to generate different decision trees, as you can see Figure 2.1 and Figure 2.2. So how to choose the attributes? Which decision tree is better? These two problems are discussed in feature selection method.

2.2 Feature Selection Method

Firstly, let's answer the second question. Which decision tree is better? Big or small? The answer, of course, is small trees. There are the following four benefits:

- A)Easier to interpret
- B)Lower danger of training-data over-fitting
- C)Tendency to eliminate irrelevant and redundant attributes
- D)Cheaper classification when the attribute values are expensive or difficult to obtain

Therefore, we ought to form small decision tree. There are two ways to construct small trees, pruning and feature selection method. Here we only discuss feature selection method. Here I'd like to introduce two feature selection methods for decision tree.

1. Gini coefficient

Gini coefficient is a method of feature selection, which can be used to indicate the impurity of data.

For a given sample set D , $Gini(D) = 1 - \sum_k (|C_k|/|D|)^2$, here C_k is the subset of samples in D that belong to the K th class, and k is the number of classes. If sample set D is divided into $D1$ and $D2$ according to A feature A , then under the condition of feature A , gini index of set D is defined as follows: $Gini(D,A) = D1/D \cdot Gini(D1) + D2/D \cdot Gini(D2)$. Gini index $gini(D,A)$ represents the uncertainty of data set D of different groups of feature A . The larger the gini index, the greater the uncertainty of the sample set. Therefore, based on the above theory, gini index can be used to determine the optimal segmentation point of a feature.

2. Entropy

Information gain represents the degree to which the uncertainty of data set classification is reduced due to feature A. Different features often have different information gain, and features with large information gain have stronger classification ability.

Definition: Gain(D,A) of information Gain of feature A on training data set D is defined as the difference between empirical entropy $H(D)$ of set D and empirical entropy $H(D|A)$ of feature A given condition, namely: $\text{Gain}(D,A) = H(D) - H(D|A)$

The feature selection method according to the information gain criterion is: for the training data set (or subset) D, calculate the information gain of each feature, compare their size, and select the feature with the largest information gain.

The training data set is set as D, and $|D|$ represent its sample size (total number of samples). There are K classes C_k , $K = 1, 2, \dots, K$ $|C_k|$ are the number of samples belonging to class C_k , and the sum of $|C_k| = |D|$. Suppose feature A has n different values $\{a_1, a_2, \dots, a_n\}$, divide D into n subsets D_1, D_2, \dots, D_n , $|D_i|$ is the number of samples of D_i , and the sum of $|D_i| = |D|$. The geometry of samples belonging to class C_k in subset D_i is D_{ik} , that is, $D_{ik} = D_i \cap C_k$, and $|D_{ik}|$ is the number of samples of D_{ik} .

Information gain algorithm:

Input: training data set D and feature A

Output: Gain of confidence of feature A on training data set D (D,A)

(1) calculate empirical entropy $H(D)$ of data set D

$$H(D) = - \sum_{k=1}^K \frac{|C_k|}{|D|} \log_2 \frac{|C_k|}{|D|}$$

(2) calculate the empirical condition entropy $H(D|A)$ of feature A on data set D

$$H(D|A) = \sum_{i=1}^n \frac{|D_i|}{|D|} H(D_i) = - \sum_{i=1}^n \frac{|D_i|}{|D|} \sum_{k=1}^K \frac{|D_{ik}|}{|D_i|} \log_2 \frac{|D_{ik}|}{|D_i|}$$

(3) calculate information gain

$$g(D, A) = H(D) - H(D|A)$$

As the figure 2.3 shows that, for these two feature selection methods, we can directly use the parameter “criterion='gini/entropy'” to implement. In the case of no feature selection method, because the decision tree defaults to gini coefficient, we can use the parameter “splitter='random', max_features=1” to implement, which means that one feature is randomly selected at a time to divide the data set.

2.3 Cross Validation

For this experiment, I used k-fold cross validation. K-fold cross validation first divides all the data into K sub-samples. One of the sub-samples is selected as the test set without repetition, and the other k-1 samples are used for training. The results were repeated K times, averaged K times or used other indicators, and a single estimate was obtained. The advantage of this method is to ensure that every sub-sample is trained and tested to reduce the generalization error. And using k-fold cross validation can avoid over-fitting and reduce the variance of the model. Here we take k is equal to 10.

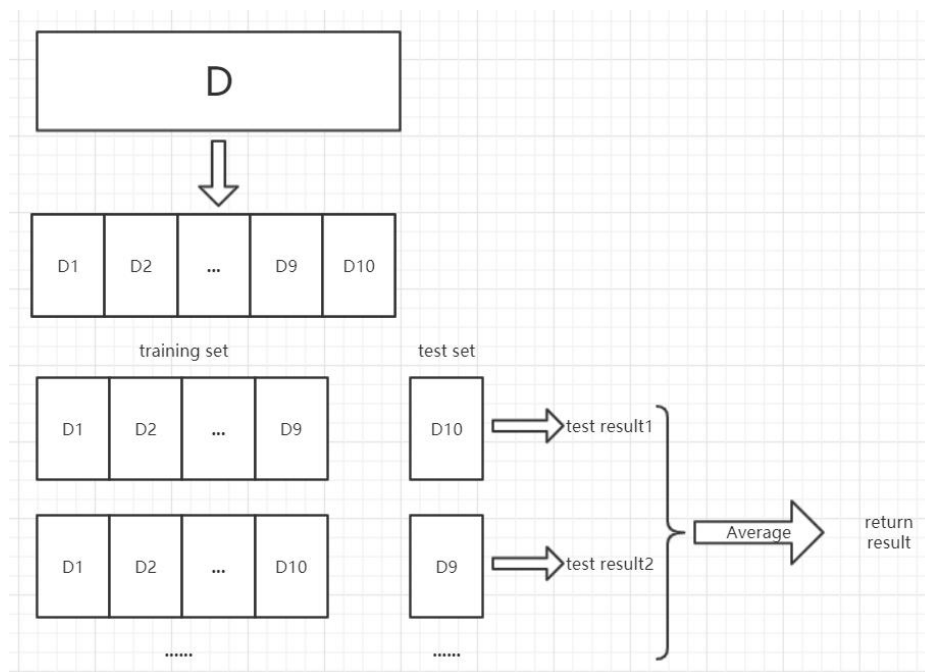


Fig 2.4 The processing of 10 folds cross validation

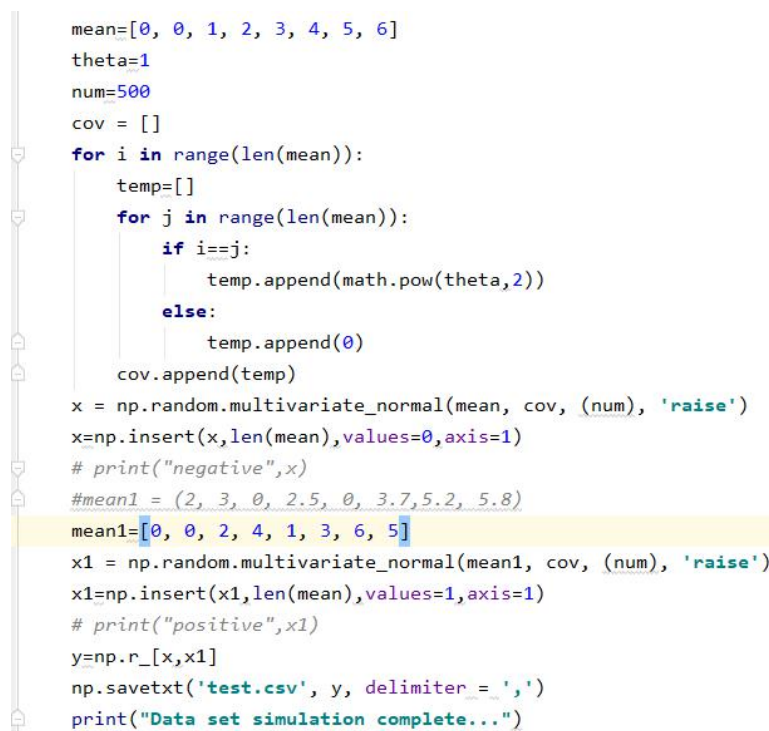
From figure 2.4, we can see that the data set is divided, a different subset is selected as the test set each time, and the average result is output. Here I implement the 10 folds cross validation by using sklearn.modelselection. I import the K-fold function and use the parameter “n_splits=10” in Kfold function, so it can return ten different group of indexes to make the whole data set into ten subsamples. And every time we

choose one of sub-samples as our test set to get the test result. You can see the details in the figure 2.3. I ran into some problems when I was doing this part. The return value on the partition of the data set was indexes, so I have to initial a data frame to store data read from our data set using an index.

2.4 Simulation of Training Data

In this experiment, I implement a simple python program for simulation of training data. The core idea is to generate a data set that satisfies the Gaussian distribution. From figure 2.4, this is a simple case of K that equals to 2 and 8 features.

Firstly, we suppose that the first, second, fourth, sixth, seventh and eighth determine the class label and the remaining features are irrelevant. Set those irrelevant attributes equal to 0 and remaining features to different values.



```

mean=[0, 0, 1, 2, 3, 4, 5, 6]
theta=1
num=500
cov = []
for i in range(len(mean)):
    temp=[]
    for j in range(len(mean)):
        if i==j:
            temp.append(math.pow(theta,2))
        else:
            temp.append(0)
    cov.append(temp)
x = np.random.multivariate_normal(mean, cov, (num), 'raise')
x=np.insert(x,len(mean),values=0,axis=1)
# print("negative",x)
#mean1 = (2, 3, 0, 2.5, 0, 3.7,5.2, 5.8)
mean1=[0, 0, 2, 4, 1, 3, 6, 5]
x1 = np.random.multivariate_normal(mean1, cov, (num), 'raise')
x1=np.insert(x1,len(mean),values=1,axis=1)
# print("positive",x1)
y=np.r_[x,x1]
np.savetxt('test.csv', y, delimiter=',')
print("Data set simulation complete...")

```

Fig 2.5 The implement of simulation of training data

Secondly, generate 50 data points from the Gaussian distribution with μ equal to mean showed in the figure 2.5 and covariance $\sigma^2 I$, which I is the 8×8 identify matrix.

To implement the Gaussian distribution, we can use “np.random.multivariate_normal(mean, cov, (50), 'raise')” this function from the library numpy. And we use “np.insert(x,8,values=0,axis=1)” to label our data points. Here 0 represents negative class and 1 represents positive class. For another set of data labeled 1, we take the same approach to simulate the data points. And we can change the value and number of mean and covariance to simulate different training data.

3. RESULT AND DISCUSSIONS

3.1 Classification Error and The Number of Features

For this part, I conducted 12 times of experiments. So I generate 12 groups of training data. And the feature number of each group equals to $2n$, $n=1, 2, 3 \dots 12$. And I assume that all the features are relevant, so there is no zero attributes. The number of the positive examples and negative examples of data is 5000 respectively. For all the values of features, the positive class is one more than the negative class. For example, if the mean of the negative class is $[1,2]$, the mean of the positive class is $[1+1,2+1]$. And noise level θ equals to 1. At the same time, I tested the error rate of the decision tree classifier with the feature selection method, entropy. The result is showed in the table 3.1.

The serial number	The number of features	Error rate
1	2	0.3959999999999999
2	4	0.26300000000000001
3	6	0.189000000000000006
4	8	0.183000000000000005
5	10	0.18799999999999994
6	12	0.17599999999999993
7	14	0.16999999999999993
8	16	0.16999999999999993
9	18	0.175000000000000004
10	20	0.18499999999999994
11	22	0.17799999999999994
12	24	0.207000000000000007

Table 3.1 Classification error and the number of features

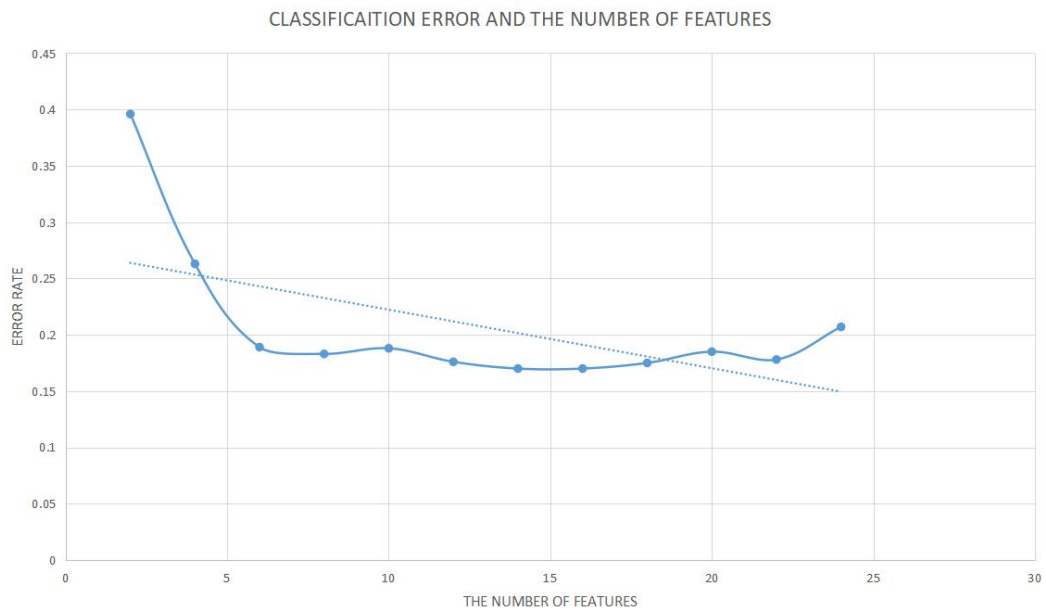


Fig 3.1 Classification error and the number of features

From the figure 3.1, we can see that the error rate as the increase of the number of features have obviously dropped by observing the trend line, but when the features are too many, error rates tend to be gentle. This shows that a bit more features number really can reduce the error rate, but when the number of features are too many, the impact on the error rate is less, and by learning materials we know that too many features tend to negatively affect the learnability.

3.2 Classification Error and The Number of Training Data Samples

In this section, I choose 8 features to investigate the classification error. Two of them are irrelevant attributes and remaining are relevant attributes. I conducted 10 times of experiments. So I generate 10 groups of training data. And the number of data samples of each group equals to $200n$, $n=1, 2, 3 \dots 10$. The number of class is equal to 2 here. In each group, the number of positive examples equal to the number of negative examples. And the values of features of negative class are $[0, 0, 1, 2, 3, 4, 5, 6]$. The values of features of positive class are $[0, 0, 2, 4, 1, 3, 6, 5]$. And noise level θ equals to 1. Besides, I used feature selection method, entropy. The result is showed in the table 3.2.

The serial number	The number of samples	Error rate
1	200	0.15999999999999992
2	400	0.13749999999999996
3	600	0.14
4	800	0.13249999999999995
5	1000	0.121000000000000011
6	1200	0.10666666666666669
7	1400	0.0971428571428572
8	1600	0.10875000000000001
9	1800	0.09333333333333338
10	2000	0.09600000000000009

Table 3.2 Classification error and the number of samples

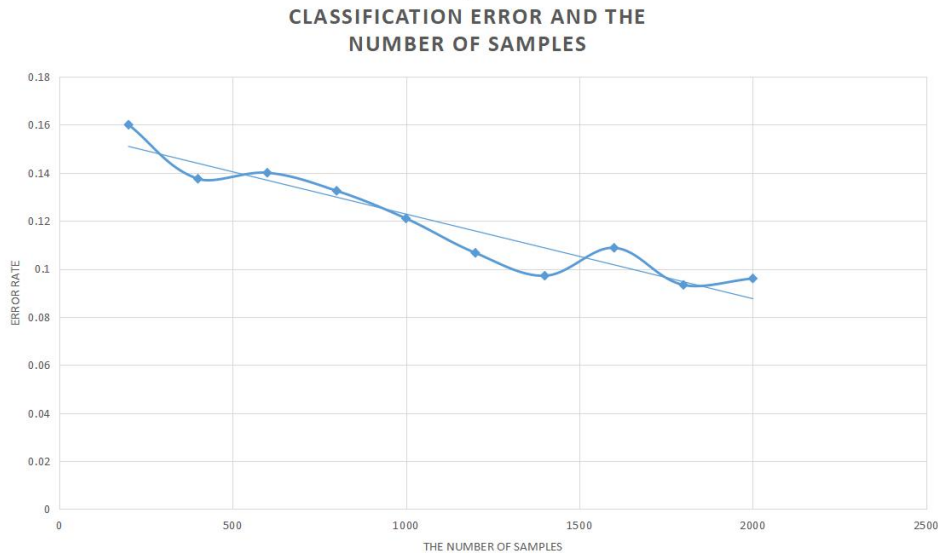


Fig 3.2 Classification error and the number of samples

From figure 3.2, we can see that, with the increase of sample size, the error rate decreases gradually. This is very obvious on the trend line. And through PAC, we can know that, on a very large sample, the error rate will be flat, because here we don't know the size of the hypothesis space, so we can't calculate that sample size.

3.3 Classification Error and The Number of Relevant Feature

Here I conduct 10 times of experiments. Firstly, I set the mean of negative class is [1] and the mean of positive class is [4]. And noise level theta always equals to 1. The number of the positive examples and negative examples of data is 5000 respectively. And then in each of the following experiments, I add two irrelevant attributes 0 to

both negative class and positive class. For feature selection method, I us information entropy method. The result is showed in the table 3.3.

The serial number	Number of irrelevant attributes	Error rate
1	0	0.09199999999999997
2	2	0.10600000000000001
3	4	0.10600000000000001
4	6	0.09299999999999997
5	8	0.10999999999999976
6	10	0.13600000000000012
7	12	0.118
8	14	0.10899999999999999
9	16	0.11300000000000001
10	18	0.126

Table 3.3 Classification error and the number of irrelevant attributes

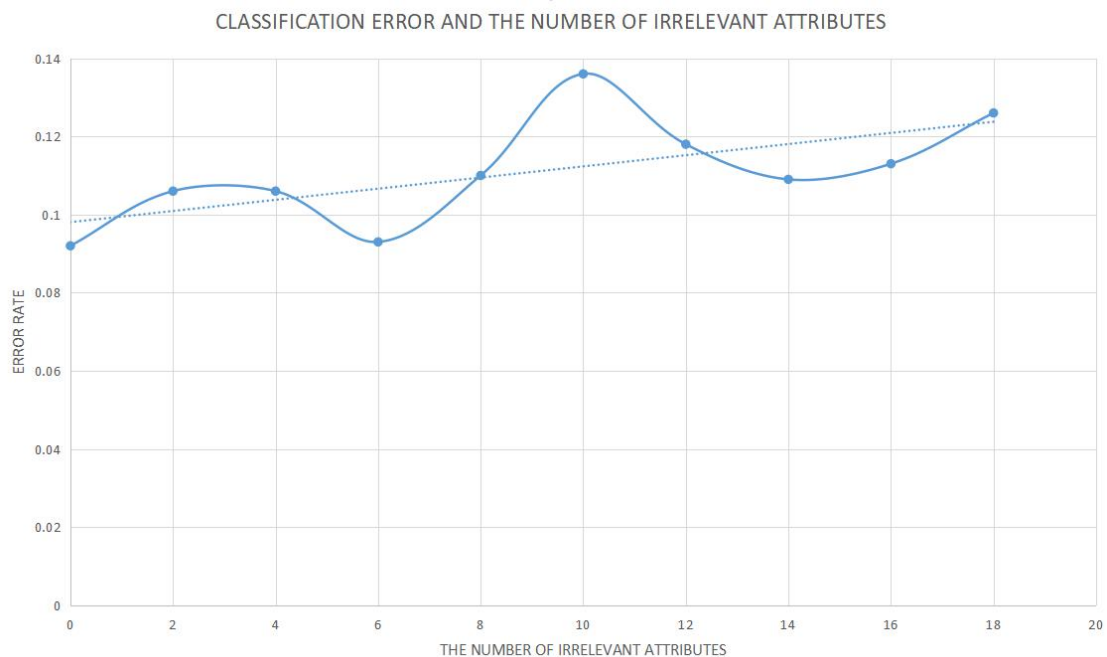


Fig 3.3 Classification error and the number of irrelevant attributes

From the figure 3.3, compared with the first group without irrelevant attributes, we can see that the error rate is a little higher than before. With the increase of the number of irrelevant attributes, we observe that the trend line, and we can see that it shows an upward trend. Therefore, we can come to the conclusion that the error rate will increase when the number of irrelevant attributes increase.

3.4 Classification Error and The Feature Selection Method

In this part, I conduct 9 times of experiments. Three of them are without the feature selection method. Three of them are with Gini. And the remaining are used entropy. For our simulation data set, I choose 8 features. Two of them are irrelevant attributes and remaining are relevant attributes. The number of data set is 2500 positive examples and 2500 negative examples. And the values of features of negative class are [0, 0, 1, 2, 3, 4, 5, 6]. The values of features of positive class are [0, 0, 2, 4, 1, 3, 6, 5]. And noise level theta equals to 1. The result is showed in the table 3.4.

The serial number	Feature selection method	Error rate	Average error rate
1	None	0.15100000000000002	0.16
2	None	0.16000000000000014	
3	None	0.1514000000000001	
4	None	0.16280000000000006	
5	None	0.17479999999999996	
6	Gini	0.08879999999999999	0.08848
7	Gini	0.08979999999999999	
8	Gini	0.08740000000000003	
9	Gini	0.08779999999999988	
10	Gini	0.08859999999999999	
11	Entropy	0.08060000000000012	0.08212
12	Entropy	0.08539999999999992	
13	Entropy	0.08239999999999998	
14	Entropy	0.08040000000000003	
15	Entropy	0.08179999999999998	

Table 3.4 Classification error and the feature selection method

From table 3.4, we can see that the decision tree without the feature selection method has a higher error rate than those with the feature selection method. And for gini and entropy, the classifier with entropy perform a little better than it with Gini coefficient, but they all effectively reduce the error rate by nearly 0.08. And for the decision tree, I use function graphviz to draw the tree and export as the pdf format. You can see the figure 3.5, 3.6 and 3.7.(To see more details, you can see the pdf file in the fold called pdfs) The tree without the feature selection method is obviously bigger than the tree with entropy or gini, This also shows that feature selection obviously eliminates the influence of irrelevant attributes, so that the error rate decreases.

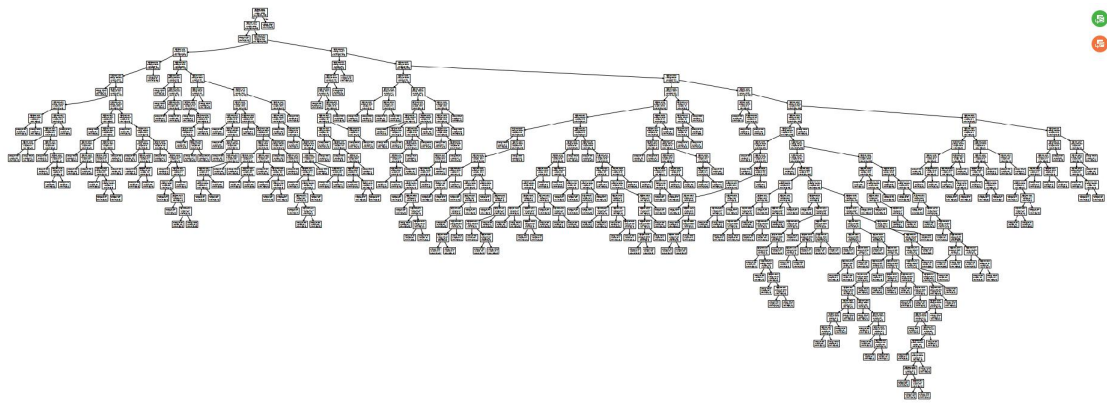


Fig 3.5 Decision tree without feature selection method

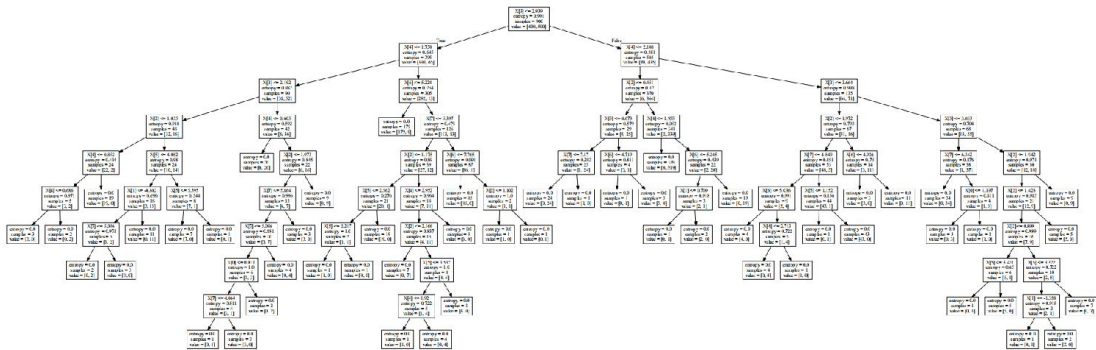


Fig 3.6 Decision tree with entropy

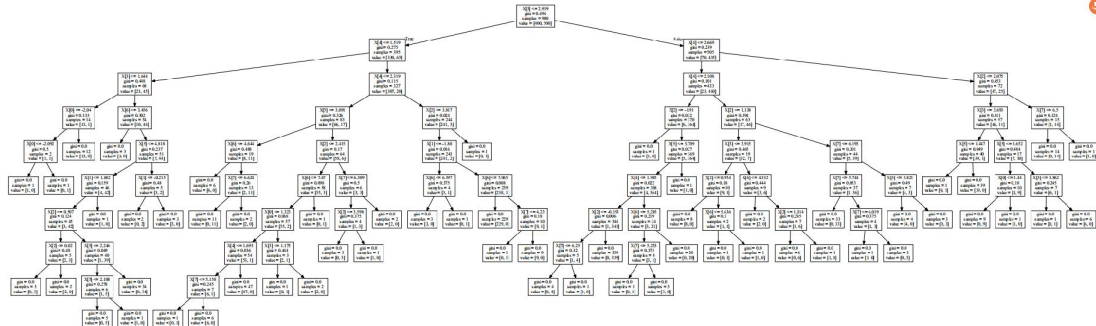


Fig 3.7 Decision tree with GINI

3.5 Classification Error and The Noise Level

In this section, I conduct 10 times of experiments. For our simulation data set, I chose eight features. Two of these are irrelevant attributes, and the rest are relevant attributes. The number of data sets is 2500 positive examples and 2500 negative examples. The feature value of the negative class are $[0,0,1,2,3,4,5,6]$. The feature value of positive class is $[0,0,2,4,1,3,6,5]$. And I set noise level, theta, to n ,

$n=1,2,\dots,10$. So θ^2 equals to n^2 , $n=1,2,3,\dots,10$. For feature selection method, I use information entropy method. The result is showed in the table 3.5.

The serial number	σ	σ^2	Error rate
1	1	1	0.08799999999999986
2	2	4	0.32299999999999995
3	3	9	0.45199999999999996
4	4	16	0.46900000000000001
5	5	25	0.44699999999999984
6	6	36	0.508
7	7	49	0.5269999999999999
8	8	64	0.514
9	9	81	0.549
10	10	100	0.55

Table 3.5 Classification error and the noise level

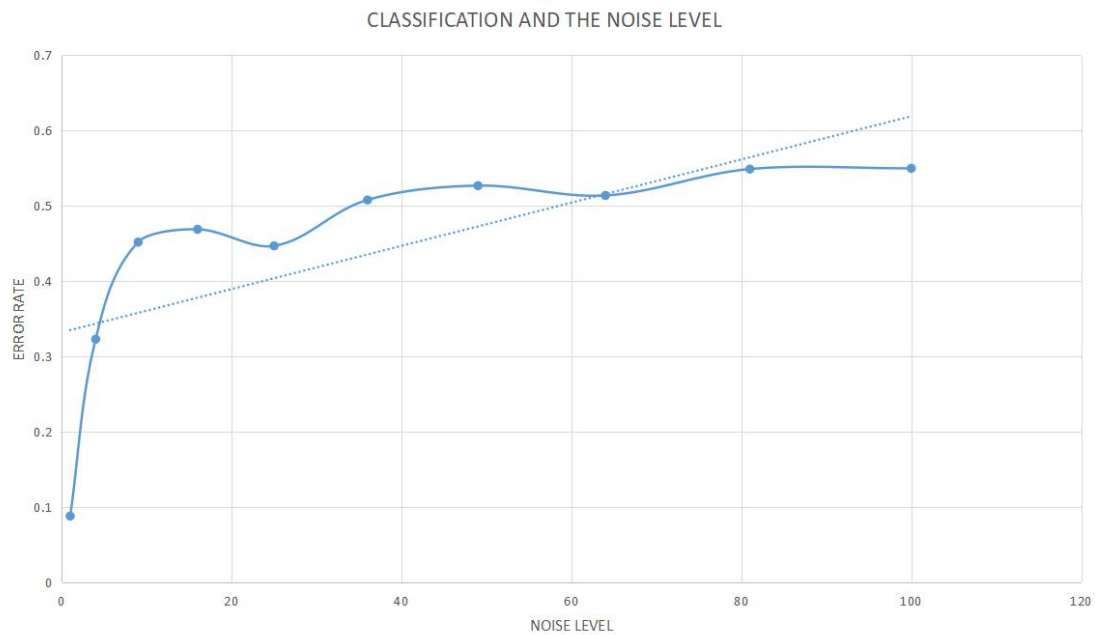


Fig 3.8 Classification error and the noise level

Figure 3.8 is the result of the table 3.5 drawing. From figure 3.8, we can see that, with the increase of noise level, firstly error rate have obvious rise, but with the increasing of noise level, error rate at around 0.5 are easing. According to the analysis, we can know that this is consistent with our expectations, because classifier error rate of around 0.5, due to too much noise, classifier is equivalent to classify the samples in random guesses.

3.6 Performance on Spam Data Set With and Without Feature Selection

Then, after analyzing our decision tree classifier on simulation data, we run our classifier on real data. Here I choose spam data set and run our decision tree classifier with and without feature selection. Spam data set has 57 attributes and 4601 instances. The data set characteristics are multivariate. And the attribute characteristics are integer and real^[2]. You can see the data set in the file named spam.cvs. I'm not going to go into data sets here. So if you want to see more details about this data set, you can visit this website "<https://archive.ics.uci.edu/ml/datasets/spambase>"^[2]. I use entropy, Gini and without feature selection method to run the spam data set separately and get three group of results. The results are showed below.

Each fold	Feature selection method	Error rate
1	None	0.281995662
2	None	0.206521739
3	None	0.215217391
4	None	0.27826087
5	None	0.1
6	None	0.141304348
7	None	0.095652174
8	None	0.104347826
9	None	0.156521739
10	None	0.308695652
Average		0.18885174

Table 3.6 The result without feature selection method

Each fold	Feature selection method	Error rate
1	Gini	0.213427332
2	Gini	0.144347826
3	Gini	0.107391304
4	Gini	0.146521739
5	Gini	0.046521739
6	Gini	0.059565217
7	Gini	0.079130435
8	Gini	0.076956522
9	Gini	0.087826087
10	Gini	0.266086957
Average		0.122777516

Table 3.7 The result with Gini

Each fold	Feature selection method	Error rate
1	Entropy	0.133167028
2	Entropy	0.133478261
3	Entropy	0.120434783
4	Entropy	0.163913043
5	Entropy	0.06826087
6	Entropy	0.057391304
7	Entropy	0.079130435
8	Entropy	0.059565217
9	Entropy	0.048695652
10	Entropy	0.233478261
Average		0.109751485

Table 3.8 The result with entropy

From the three sets of results, we can observe that the error rate with feature selection are significantly lower than the result without feature selection. The error rate with feature selection method Gini and entropy is nearly 0.11. That's a good error rate. It shows that the decision tree with feature selection performs well in real data sets. And the error rate without feature selection is nearly 0.19. This error rate is lower than others, but at least it is effective. And this is also consistent with the result from 3.4 classification error and feature selection method. Both result show that the decision tree classifier with feature selection method can efficiently reduce the error rate by nearly 0.08. The only difference here is that error rates on real data sets have increased. This might be caused by the different features and value.

And then, if we analyze the result of each fold, we can see that the error rate of some folds without feature selection, like 7 and 8, are not bad. This might be caused by 10-fold cross validation, since in each fold the test set is changed. Or maybe the classifier randomly select the “good” features to construct the decision tree. For the error rate of Gini and entropy, we also can see some error rate are even bad. This can also be explained by 10-fold cross validation.

4. CONCLUSION

Through this experiment, firstly, I learn how to implement a decision tree and its working principle. At the same time, I also understand the feature selection method of gini coefficient and entropy. Besides, I know that the classification error rate will increase with the increase of noise level, decrease to a certain extent with the increase of the number of features, increase to a certain extent with the increase of irrelevant attributes, and decrease significantly with the increase of training set. The decision tree with feature selection method performs better than it without feature selection method. On a real data set, spam data set, the decision tree with can classify each example with an error rate of nearly 0.12, which is nearly 90% accuracy. For the decision tree without feature selection method on a real data set, we can see that its error rate is nearly 0.19, which is nearly 80% accuracy. The difference between non-feature selection and feature selection is nearly 0.1, which is the consistent as the result obtained in the simulation data previously. This also verifies the correctness of the experimental results. Finally, I'd like to say that I have gained a lot from this experiment. Thank my classmates and professors for help and guidance.

REFERENCE

- [1] Kubat, M., 2017. An introduction to machine learning (Vol. 2). Cham, Switzerland: Springer International Publishing.
- [2] Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science.