CSE 601 : Data Mining and Bioinformatics

Classification Algorithm Project Report

Weiyang Chen
University at Buffalo, NY
wchen38@buffalo.edu

ZhiWen Huang
University at Buffalo, NY
zhiwenhu@buffalo.edu

ABSTRACT

The project aim to implement four classification algorithms which are K-Nearest Neighbors ,Decision Tree, Random Forests and Naive Bayes classification to compare their performance with same given data set by applying 10 - fold Cross Validation to calculate their average accuracy, Precision, Recall, and F-1 measure, and discuss their pro and cons.

1 DATA

1.1 Description

Each row represents one data sample.

(1) The last column of each row is class label, either 0 or 1.
(2) The rest columns are feature values, each of them can be a real-value (continuous type) or a string (nominal type).

1.2 Data Pre-processing

1.2.1 Continuous and Nominal. Divide the data into three subsets by the types of features, first subset contains all the continuous type features (check if the column contains real value), and second subset contains all the nominal type features (check if the column contains the string). third subset contains all the labels which is last column of data. After the division, the same row in the two subsets corresponding to the same sample.

1.2.2 10-fold cross validation. Calculate the number of samples / 10 to get a number N, if it contains decimal, then round it. And for each time, it will return the index from i to $i + N$ which i start from 0. And the range between i and $i + N$ is current testing set index, index excluding testing set index are training set index. So we use those index to get training and testing set for using later.

2 K - NEAREST NEIGHBORS

2.1 Introduction

The idea of the K-NN algorithm is that a sample is most similar to the k samples in the data set, and if most of the k samples belong to a certain category, the sample also belongs to this category. In this process, we will find the k most similar data in the training set by calculating the similarity of all the data in the training set to the point we want to predict and classify by voting.

Supervised by Jing Gao.

Project in Computer Science (CSE601),
2020.

2.2 Pre-Processing

2.2.1 Min-Max Normalization. Applying Min-Max Normalization to Normalize the continuous type feature value to a scale of (0,1), Using the formula below :

$$x_{(normalization)} = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{1}$$

2.3 Algorithm

Algorithm 1: K-Nearest Neighbor

```
Function KNN($Training\ set, Testing\ set, K$):
    foreach $test\ data\ i \in Testing\ set$ do
        foreach $train\ data\ x \in Training\ set$ do
            Calculate the distance between two sample $x$ and $i$ by using Euclidean distance.
            Add result distance and training data $x$ into array $result$.
        end
    end
    Find K smallest distance number in $result$, to voting by their training labels and return the final predict label.
End Function
```

2.4 Implementation

First we accept the parameter of training index set, testing index set and value of K.

Then for each index in testing index set, we will find the this test sample by calling its index in continuous and nominal subsets. And run all the training index set to calculate their distance, for the continuous type features, we have already normalized them, and we just simply subtract them, which in euclidean distance is $p_i - q_i$ process. for the nominal type features, we will check if their string is the same or not, if they are the same then we used 0 represents $p_i - q_i$ otherwise, we set it as 1. Then we sum of square each features' subtract value and square root to get final distance.

After we get final distance, we need add them with corresponding training labels into an array.

After we finished, a testing index loop, we will make predict its label by find k smallest distance, and voting by their training labels.

$$Euclidean\ Distance : d(p, q) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2}$$

project3_dataset1.txt				
	K = 3	K = 5	K = 7	K=10
Accuracy	0.96484	0.96835	0.97183	0.96835
Precision	0.97467	0.98143	0.98107	0.97996
Recall	0.92697	0.93411	.94327	0.93634
F_measure	0.94883	0.95578	0.96127	0.95719

Table 1: K-NN with dataset 1

project3_dataset2.txt				
	K = 3	K = 5	K = 7	K=10
Accuracy	0.69021	0.66023	0.68179	0.68831
Precision	0.56616	0.52683	0.54731	0.57277
Recall	0.47797	0.40719	0.39942	0.38535
F_measure	0.50624	0.44381	0.44707	0.44338

Table 2: K-NN with dataset 2

2.5 Discussion

2.6 Result Analysis

- KNN performs better in data-set 1 than data-set 2 which first data set only contains the continuous type features and second data set mix the continuous and nominal type features.
- Different number of K value will make result different.
- Different Distance metric will make result different too.

2.6.1 *Pros.*

- Easy to implement.
- Has high accuracy on data only contain the continuous type features.
- K-NN does not have training step.
- K-NN has no assumptions.
- Work well on classification and regression.

2.6.2 *Cons.*

- Work slowly on large data set.
- Different K value will make result different, so select a good K value is a big problem for the K-NN.
- K-NN algorithm is very sensitive to outliers as it simply chose the neighbors based on distance criteria.
- K-NN require a set of stored records, it will need large memory.

3 NAIVE BAYES CLASSIFICATION

3.1 Introduction

Naive Bayes Classifier is a "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features.[1] This assumption is called class-conditional independence.

Calculating the probability by applying Bayes theorem for a test to a training set, and predict the label that has maximum probability.

$$P(H_i|X) = \frac{P(X|H_i)P(H)}{P(X)} \tag{3}$$

- $P(H_i|X)$: Class Posterior Probability.
- $P(H_i)$: Class Prior Probability.
- $P(X|H_i)$: Descriptor Posterior Probability.
- $P(X)$: Descriptor Prior Probability.

3.2 Pre-Processing

3.2.1 Gaussian Distribution. Continues types data can't calculate the probability in a same feature directly, so we need to apply Gaussian distribution to calculate the probability for a continuous type attribute.

We will implement a function that accept parameter mean σ, standard deviation μ and current feature value x. which σ is the mean of particular attribute in this feature, μ is the standard deviation of particular attribute in this feature. Then we apply Gaussian distribution formula to return the probability, this function will be used in later of algorithm.

$$y = \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(x-\mu)^2}{2\sigma} \tag{4}$$

3.3 Algorithm

Algorithm 2: Naive Bayes Classification

```
Function NBC(Training set, Testing set):
    foreach test data ∈ Testing set do
        Calculate current test data 's Descriptor Prior Probability PX.
        Separate the training set by their labels into subsets.
        foreach Subset do
            Calculate the class prior probability PH and descriptor posterior probability PXH in current subset.
            Calculate the Class Posterior Probability by PH * PX/PX and add this probability with subset's label into list L.
        end
        Predict the label which has maximum class posterior probability in list L
    end
End Function
```

3.4 Implementation

We accept input parameter training index set and testing index set.

For each test data :

First, we calculate the descriptor prior probability

- calculate the each feature's probability in current test data corresponding to whole training set, and multiply them together to get descriptor prior probability.
- for each feature, if it is continuous types, then we will calculate the mean, and standard deviation with same feature in the whole training set. and apply mean, standard deviation,

current test feature value into Gaussian distribution function to calculate the probability.
- for each feature, if it is nominal types, calculate this feature's probability by counting current string appear times in whole training set within same feature, and divide by size of training set.

Second, we divide the training set into some subsets by their labels (Under same subset, the label should be equal).

Then For each subset we need to calculate the class Prior Probability.

$$Class\ prior\ Probability = \frac{Size\ of\ subset}{Size\ of\ whole\ training\ set} \tag{5}$$

And we also need to calculate the Descriptor Posterior Probability for each subset:

- Similar to the Descriptor Posterior Probability, calculate the each feature's probability in current test data corresponding to subset and multiply them together to get Descriptor Posterior Probability.
- If feature is continuous type, then calculate the mean and standard deviation with same feature in the subset, and apply them with current test feature value into Gaussian distribution function to calculate the probability.
- If feature is nominal type, then calculate feature's probability by counting current string appear times in subset within same feature and divide by size of subset.

After we get Class Prior Probability, Descriptor Posterior Probability and Descriptor Prior Probability. We can apply bayes theorem to calculate the Class Posterior Probability for each subset. And return the label which subset has maximum Class Posterior Probability value.

3.5 Result

project3_dataset1.txt	
Accuracy	0.9348997493734336
Precision	0.9178709362532891
Recall	0.9044426356826323
F_measure	0.9100318381263209

Table 3: Naive Bayes Classifier with dataset 1

project3_dataset2.txt	
Accuracy	0.7013586956521739
Precision	0.5707868713673667
Recall	0.6208264250369513
F_measure	0.5858132793890309

Table 4: Naive Bayes Classifier with dataset 2

3.6 Discussion

3.6.1 Result Analysis. Naive Bayes performs better in data-set 1 than data-set 2 which first data set only contains the continuous type features and second data set mix the continuous and nominal type features.

3.6.2 Pros.

- Easy to implement.
- When the independent assumption holds then this classifier gives outstanding accuracy.
- Make prediction by compare probability.

3.7 Cons

- The assumption of the independence is often invalid in real life.
- Zero probability problem might happen.
- Product of many small probability might made vanishing value happen

4 DECISION TREE CLASSIFICATION

4.1 Introduction

Decision tree builds a regression models in forms of tree structure. This will help us breakdown the dataset into smaller chunks of data and branches are features in the dataset. Some keypoints of the decision tree model is that we will have nodes to represent our each branch and leaf node which is the labels of the given selection. This can help us easily work with discrete data and continous data.

4.2 Pre-Processing

In order to seperate the numerical features and categorical features, on the specific column that contains the categorical features, we give a mapping to the unique name with unique id. For example in dataset2.txt, we have Present and Absent, so we map them into [0, 1] respectively. The other numerical data are kept as it is.

4.3 Implementation

There are few keypoints to the implementations:

- Gini index to determined the tree split.
- Node to keep track value and left right nodes.
- Terminal Node also the leaf node which tell us the label it leads to.

With these keypoints, my first step is to find the gini index of each samples from the dataset. Then we compare the gini index from other samples and get the lowest gini index. The lowest gini index will determined our best split. Each split then will have the left node and right node which then contains the next feature value. Unless the split tells us that the left node or right node is a terminal node, then the value of left or right will either be [0,1] because this means we reach the end of the tree and the classification has 2 labels [0,1].

4.4 Results

The following are the results from 10 folds cross validation. The accuracy of each folds and the average measurements of all 10.

project3_dataset1.txt
Accuracy
0.9298
0.9298
0.9649
0.9824
0.8596
0.9473
0.8771
0.9824
0.8947
0.875

Table 5: 10 Folds of Cross-Validation

project3_dataset1.txt	
Accuracy	0.9243
Precision	0.9147
Recall	0.8853
F_measure	0.8979

Table 6: Average Measurements

project3_dataset2.txt
Accuracy
0.6304
0.6956
0.6521
0.5434
0.5652
0.5434
0.5652
0.7391
0.5869
0.625

Table 7: 10 Folds of Cross-Validation

project3_dataset2.txt	
Accuracy	0.6146
Precision	0.4461
Recall	0.4937
F_measure	0.4609

Table 8: Average Measurements

4.5 Result Analysis

4.5.1 *Pros.*

- Simple to read the tree structure.
- Able to determine categorical features and numerical features.
- Very accurate on small dataset with many features

4.5.2 *Cons.*

- Can cause over-fitting.
- Limitation of tree depth.

5 RANDOM FOREST

5.1 Introduction

Similarly to implement the decision tree, random forest tree algorithm is slightly different. We still find gini index and create nodes to build trees. The difference part of random forest tree is we take in the amount of trees we want to build. For each tree, we want to choose the feature base on the number of features. Then we train based on those features to build our tree.

5.2 Pre-Processing

All the pre-processing remain the same as the decision tree. The only different is the training dataset and the testing dataset. Instead of taking just the number of fold cross validation to train our data and test, we also include the sample size base on the features. Nomrally the number of feature we take is 20 percent of the dataset.

5.3 Implementation

Similar to Decision Tree, there are few keypoints to the implementations:

- Gini index to determined the tree split.
- Node to keep track value and left right nodes.
- Terminal Node also the leaf node which tell us the label it leads to.
- Number of features are used to get sample size.
- Number of trees to create our random forest.

With these keypoints, my first step is to find the gini index of each samples from the dataset. Then we compare the gini index from other samples and get the lowest gini index. The lowest gini index will determined our best split. Each split then will have the left node and right node which then contains the next feature value. Unless the split tells us that the left node or right node is a terminal node, then the value of left or right will either be [0,1] because this means we reach the end of the tree and the classification has 2 labels [0,1]. Additionally to random forest, we set the size of the number of features and number of trees. This is to determine how much trees we are running to test our data. Then we take the highest probability of the tree to get the correct label.

5.4 Results

The following are the results from 10 folds cross validation. The accuracy of each folds and the average measurements of all 10.

5.5 Result Analysis

5.5.1 *Pros.*

- More accurate than decision tree.
- Able to determine categorical features and numerical features.
- Able to handle overfitting.

5.5.2 *Cons.*

- Can be really slow.
- Limitation of tree depth.

project3_dataset1.txt
Accuracy
0.98245
0.94736
0.98245
0.92982
0.92982
0.96491
0.92982
0.98245
0.92982
0.96428

Table 9: 10 Folds of Cross-Validation with 3 tree

project3_dataset1.txt	
Accuracy	0.95432
Precision	0.95036
Recall	0.92785
F_measure	0.93787

Table 10: Average Measurements

project3_dataset1.txt
Accuracy
0.94736
0.94736
0.96491
0.92982
1.0
0.91228
0.94736
1.0
0.98245
0.875

Table 11: 10 Folds of Cross-Validation with 5 tree

project3_dataset1.txt	
Accuracy	0.94890
Precision	0.95318
Recall	0.90944
F_measure	0.93015

Table 12: Average Measurements

6 KAGGLE - COMPETITION

6.1 Problem

Apply various tricks on top of any classification algorithm discussed in class (including nearest neighbor, decision tree, Naïve Bayes, SVM, bagging, AdaBoost, random forests) and tune parameters using training data. You can call packages for these algorithms but need to implement any improvement on top of these algorithms.

project3_dataset1.txt
Accuracy
1.0
0.92982
0.94736
0.85964
0.94736
0.98245
0.92982
0.92982
0.98245
0.92982

Table 13: 10 Folds of Cross-Validation with 10 tree

project3_dataset1.txt	
Accuracy	0.945488
Precision	0.95560
Recall	0.88745
F_measure	0.91943

Table 14: Average Measurements

project3_dataset2.txt	
Accuracy	0.655706
Precision	0.50833
Recall	0.46065
F_measure	0.47896

Table 15: Average Measurements

6.2 Main Idea

As bagging features, creating a number of weak classifiers each classifier has their own weight, and make vote by those weak classifier in order to combine into a strong classifier. I am using 5 different classification algorithms to create five weak classifiers (KNN , SVM, Decision Tree, Naive Bayes, Logical Regression) and using n_folds_cross_validation to calculate their average F_measure, and give them weight through accuracy. For example, if KNN's F_measure is 95%, then its vote will be 95% * KNN predict result.

6.3 Parameter Selection

6.3.1 KNN. K value select is a hard problem for the KNN, I apply 10_fold_cross_validation on training set, and change the K value and from 0 to 20 in order to find optimal k value which has highest F_measure. The result i got is 3.

6.3.2 SVM. Margin value will also affect the result of SVM, Similary to KNN, applying 10_fold_cross_validation on training set, and change Margin Value C from 1 to 20 and gamma value from 1e-1 to 1e-10 to find optimal C value and gamma value which has highest F_measure, The result i got is 3.5 and gamma is 1e-8.

6.3.3 Decision Tree. There two important parameters we need to look at, which are criterion and max_depth. I tried the 'gini' and 'entropy' formula for the criterion and change the max_depth value from 2 to 20 in order to find highest F_measure by 10_fold_cross_validation on training set. Then 'entropy' will be better performance for criterion parameter, and max_depth is 12.

6.3.4 Naive Bayes. Naive Bayes not have much parameters we can choose, but the most important one is using GaussianNB model to implement the Naive Bayes, because the competition data are all continuous type data.

6.3.5 Logical Regression. There are two parameters i tried to select which are penaty and max_iter, After I test, when i take 'l2' norm formula for the penality calculation and set iterations times as 120, it will reach the highest F_measure.

6.4 F_measure

This is only the average of F_measure for 10_fold_cross_validation on training set. And we use each algorithm's F_measure to give them the weight in the bagging.

Average F_Measure on competition training set	
KNN	0.8931751550271458
SVM	0.8990667620666641
Decision Tree	0.8332889402922363
Logical Regression	0.8685169407880642
Naive Bayes	0.6298112685956598

Table 16: Average F_measure

6.5 Improvement

- We use experiment results to select the current optimal parameters for the algorithms.
- We use their F_measure value to determine their weight in the bagging. This will reduce the impact on currently unsuitable algorithms.

REFERENCES

[1] [n. d.]. Naive Bayes classifier. https://en.wikipedia.org/wiki/Naive_Bayes_classifier