Wei Zheng 5002 6424
Ruijie Ge 5006 2092
Chao Zhang 5006 1505
December 11, 2013

Project 3 Report

1.Implementation

1.1 Ten Fold Cross Validation

For each iteration, we separate the original dataset to into two part, training data and testing data with proportion 9:1. For example, if we have n samples, we sample 9*floor(n/10) data samples without replacement as training data and the rest are testing data.

Training data will be used to train the model using different algorithms. For testing data, first we will hide the true class label and use the attribution part applied with the model get the prediction of class label. Then we compare the true class label with the predicted label and calculate accuracy, precision, recall and F1 score. Finally, we average the result for these ten iterations.

1.2 Preprocessing

Since there are nominal and continuous value in the data set and nominal attributes are strings, we record a array indicates which columns are nominal attributes. Nominal attributes in string will replaces with integer for easier further computation.

Then we normalize the data set column wise so that the attribute value scale from 0 to 1.

The preprocessed result is a n by d+1 matrix, where n is the number of samples and d is the number of attributes.

1.3 Nearest Neighbor

1.3.1 Algorithm Description

For each sample in the testing set, we select its k nearest neighbor in the training set based on the distance metric described below, and predict the class label using major voting.

1.3.2 Parameters

For this algorithm, we can specify the number of nearest neighbor taken into consideration.

1.3.2 Choose K

We tried K form 1:10 and report the best cross validation result.

1.3.3 Distance Metric

Here we use euclidian distance. For nominal attribute, we define the difference of two attribute value as 0 if they are equal, 1 otherwise. The distance is the square root of the sum of all the attributes difference.

1.4 Decision Tree

1.4.1 Algorithm Description

We implement binary decision tree in our code.

First, we initialized all the training samples as one node. In each iteration, we traverse one node and see if this node need split and how to split it.

We use an array record the best split for each attributes. We define the best spilt as the the split that can give the highest information gain. Then we choose the best attribute split to separate samples in current node into two node. Once the information gain is less than a threshold, the tree will stop growing in current branch. And current node is marked as pure and the we assign the class label of this pure node using major voting.

We repeat the process above until all the leaf node are pure.

For each sample in the testing data, we apply the generated tree and visit the tree until it falls into a pure node. Finally, we label the class label of this sample as the class label of this pure node.

1.4.2 Parameters

1. The information gain metric flag. Use Gini Index or Entropy.

2. The prune threshold. Once the information gain is less than the threshold, the tree will stop growing in current branch.

1.4.3 Binary Split

For continuous attributes, we first sort the attribute value along with the class label. Then we try difference threshold that can separate the attribute values into two part, less then the threshold and greater or equal than the threshold.

For nominal attributes, we should try all the possible ways to separate the set of unique attribute value into two group. For example, if the attribute values are [1 1 2 2 2 3 3], then the unique set is [1 2 3], the possible separation are {[1 2 3], []}, {[1 2], [3]},{[1], [2 3]},{[2], [1 3]}

1.4.4 Prune

As we described above, we can set the prune threshold which lead to early stopping and avoid over fitting.

1.5 Naive Bayesian

1.5.1 Algorithm Description

Calculate the Class Prior Probability. Then we separate the training data according to class label. In each of this subset, calculate the Descriptor Posterior Probability. Finally calculate the Class Posterior Probability and label the testing item to the class which gives higher Class Posterior Probability.

1.5.2 Continuous Attribute

For continuous attribute, we use a normal distribution to fit the data.

1.5.3 Laplacian Correction

To avoid the zero-probability problem, we use Laplacian correction, add 1 to each case.

1.6 Random Forest

1.6.1 Algorithm Description

First sample data from original training data with replacement to get the new training data of the equal size as original training data.

Use the new training data, apply the decision tree algorithm with some modification and get prediction result.

Repeat the above steps T times get the final result using major voting.

1.6.2 Parameters

1. the number of decision trees need to grow.

2. percentage of attribute used to find best split.

1.6.3 Bootstrap

Sample data from original training data with replacement to get the new training data of the equal size as original training data. Then use this new training set to build tree.

1.6.4 Modification on Decision Tree

1. Just use subset features to calculate the best split at each node.

2.Each tree is fully grown and not pruned.

2. Results

DATASET 1	Nearest Neighbor	Decision Tree	Naive Bayesian	Random Forest
ACCURACY	0.9369	0.9385	0.9262	0.9708
PRECISION	0.9191	0.9177	0.9198	0.9780
RECALL	0.9047	0.9138	0.8819	0.9404
F MEASURE	0.9103	0.9135	0.8988	0.9585

DATASET 2	Nearest Neighbor	Decision Tree	Naive Bayesian	Random Forest
ACCURACY	0.6687	0.6250	0.6833	0.6408
PRECISION	0.5727	0.5453	0.5679	0.5670
RECALL	0.4800	0.4348	0.5947	0.4794
F MEASURE	0.4737	0.4767	0.5717	0.4928

3. Pros and Cons

3.1 Nearest Neighbor

Pros:

1. Analytically tractable

2. Simple implementation

3. Nearly optimal in the large sample size

4. Uses local information, which can yield highly adaptive behavior.

5. Lends itself easy to parallel implementation.

Cons:

1.k-NN classifiers are lazy learners. It does not build models explicitly.

2.Classifying unknown records are relatively expensive

3.Suffer from curse of dimensionality

3.How to choose K

3.2 Decision Tree

Pros:

1. Decision tree provides expressive representation for learning discrete-valued function

2. Decision trees are considered to be a nonparametric method. This means that decision trees have no assumptions about the space distribution and the classifier structure.

Cons:

1. Not expressive enough for modeling continuous variables

2. Problem introduced by greedy approach

3. Over-sensitivity to the training set, to irrelevant attributes and to noise

3.3 Naive Bayesian

Pros:

1. Fast to train

2. Not sensitive to irrelevant features

3. Handles real and discrete data

4. Handles streaming data well

Cons:

Assumes independence of features

3.4 Random Forest

Pros:

1. Incorporate more diversity and reduce variances

2. Searching among subsets of features is much faster than searching among the complete set

Cons:

1. Difficult to interpret