STATE UNIVERSITY AT BUFFALO

Project 3 : Classification Algorithms

Yuze Liu 50207903
Luting Chen 50133507
Vicky Zheng 50037709

12/10/2016

1 K-NEAREST NEIGHBORS ALGORITHM

1.1 ALGORITHM DESCRIPTION

K-Nearest-Neighbors algorithm is a method used for classification and regression. In K-NN classification, the output will be the class membership. The objective will be classified by the majority votes of its neighbors, with the objective being assigned to the class most common among its k nearest neighbors.
In this project, we implement the 10-fold cross validation with the K-NN classify algorithm.
First,we split the whole dataset into 10 parts, we pick one part as the testing set and other 10 parts as the training set. For each sample in the testing set, we calculate the Euclidean distance between this sample to all the other samples in the training set. Then we pick the k nearest samples as its neighbors. We will classify this sample by the majority votes of that k neighbors.
In the project, there are only 2 classes. So we just need to compare the two weights.If the neighbor is classified as class 0, then the weight of this sample to be classified in class 0 will be : $weight0 = weight0 + \frac{1}{dis}$, dis is the Euclidean distance between the neighbors and the sample, other wise $weight1 = weight1 + \frac{1}{dis}$, then we compare weight0 and weight1 after go through all the neighbors of this sample, the sample will be classified with the higher weight.
After get the classification result from the test set, then we calculate all the four measurement values and then we pick another dataset from the remaining 9 datasets as the test set and other 9 as training set, repeat this procedure until all the ten datasets have been used as test set once.
In the end, we calculate the average of the four measurement values.

1.2 Pros and Cons

Advantage:
The algorithm is effective when the dataset is very large.
The algorithm is robust to noisy training data.

Disadvantage:
Need to determine the value of parameter K.
There are a lot of choices to choose what kind of distance we will use to determine the neighbors.
Computation cost is quite high because we need to compute the distance of each query instance to all training samples.
If we have nominal data, we need to label the nominal value first and then calculate the distance.

1.3 Result Evaluation

Results for project3 dataset1.txt are:
Accuracy: 0.92615914787
Precision: 0.922329566995
Recall: 0.87079481314
F: 0.89442003084

The above value is the optimal solution I get, k = 3

Results for project3 dataset2.txt are:
Accuracy: 0.60625
Precision: 0.411791272453
Recall: 0.318770209838
F: 0.357137604338

The above value is the optimal solution I get, k = 4

2 Decision Tree

2.1 Algorithm Description

2.2 Pros and Cons

2.3 Result Evaluation

3 Decision Tree with Random Forest

3.1 Algorithm Description

3.2 Pros and Cons

3.3 Result Evaluation

4 Decision Tree with Boosting

4.1 Algorithm Description

4.2 Pros and Cons

4.3 Result Evaluation

5 Naive Bayes

5.1 Algorithm Description

Naive bayes is a classification algorithm that is based on the Bayes Theorem. Naive bayes aims to classify the probability of a sample X being in a class H_i using Bayes Theorem. The Bayes theorem is:

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

The things that we are classifying often have multiple attributes which we will refer to as A where $A = (A_1, A_2, ..., A_d)$. Let X be something we are trying to classify and $X = (x_1, x_2, ..., x_d)$. $P(X)$ is the prior probability of X where $P(x_j) = \frac{n_j}{n}$ where n_j is the number of of training samples in our training set that have the value x_j for attribute A_j.

$P(H_i)$ is simply the class prior probability.

$P(X|H_i)$,the descriptor posterior probability, can be calculated by:

$$P(X|H_i) = \prod_{i=1}^{d} P(x_j, H_i)$$

Calculating the descriptor posterior probability reveals on of the weaknesses of Naive Bayes. If a single value $P(x_j, H_i)$ is 0 then the entire product of $P(X|H_i) = \prod_{i=1}^{d} P(x_j, H_i)$ will evaulate to 0 which will cause $P(H_i|X) = \frac{P(H_i)P(X|H_i)}{P(X)}$ to also evaluate to 0. This can be corrected by using a Laplacian correction where if there is an $n_j = 0$, then

we will simply add 1 to every n_j and increase the total number of samples to $n+k$ where k is the number of possible values the attribute A_j can take on.
You will also notice that $P(H_i|X) = \frac{P(H_i)P(X|H_i)}{P(X)}$ does not work for continuous values because we cannot count continuous values to get posterior probabilities. I chose to address this by assuming a Gaussian distribution to use:

$$P(H_i|x_j) = \frac{1}{\sqrt{2\pi\sigma^2_{H_i,x_j}}} e^{-\frac{1}{2}(\frac{H_i-\mu_{H_i,x_j}}{\sigma_{H_i,x_j}}^2)}$$

I chose to use this method because it seemed to be one of the most popular methods out there for dealing with continuous values when implementing Naive Bayes.
After we calculate $P(H_i|X) = \frac{P(H_i)P(X|H_i)}{P(X)}$ for all H_i, we assign X to the class H_i where $P(H_i|X)$ is the maximum probability.

5.2 Pros and Cons

Some of the pros of using Naive bayes is that its simple to implement and it is efficent. One of the cons, however, is that it assumes attribute independence, which of course is not true for all datasets. Another con is that the descriptor posterior probability may evaluate to 0 - we have to prevent this by using a Laplacian correction. This can happen often in small datasets so Naive bayes performs best on large datasets.
Something else that may be considered a con is that there are multiple ways to deal with continuous attributes. I chose to handle this by using Gaussian Naive Bayes. This may not perform well for other datasets that have different distributions.

5.3 Result Evaluation

I implemented k-cross fold by randomly shuffling my dataset, and partitioning it into k test sets and k training sets. I took the average of the k accuracy, precision, recall and F-values.

Results for project3 dataset1.txt are:
Accuracy: 0.712582417582
Precision: 0.589052375343
Recall: 0.748140191881
F: 0.657284240816

Results for project3 dataset2.txt are:
Accuracy: 0.671195652174
Precision: 0.51574025974
Recall: 0.761451858741
F: 0.612021438585