Data Mining and Bioinformatics

Project 3

Clustering Algorithms

Saeb Ragani Lamooki | 50205221

K-Nearest Neighbors

We extracted the input data with each sample as a row and each feature as a column and the last column was the actual class value. Next, we searched through the first row of the data set to find string type values and we stored the corresponding column indices. The string values were next replaced using one hot transformation. Each feature in the training and testing processed data sets was next normalized by subtracting the feature mean and division by the feature's standard deviation. We subsequently calculated the Euclidean distance of each test data point and the entire train data set using the normalized features. We predicted the test class by taking the majority class values of the K (9) data points with smallest distances with the test point. Then we calculated the accuracy measures using the actual and the predicted class of the test data in each iteration of the 10-fold cross validation. The results present these measures in each iteration of the cross validation as well as the average measures.

Pros

It is easy to implement.

Cons

It is sensitive to the selected K value based on the configuration of the input data. Too small K can lead to adverse effects by the outliers. Also, large K's can cause selecting data from the wrong classes. It does not construct explicit model and unlike decision tree it is a lazy learner. Also, it is an expensive algorithm.

Naïve Bayes

Bayesian classifier which is a statistical classifier predicts the probability of the class memberships. In this algorithm we are to estimate $P(H_i|X)$ which is the class posterior probability and assign the class with the highest probability among all classes $C_{1,}\dots C_m$. However, as we will see the equation for $P(H_i|X)$ below, it is easier to only calculate the numerator in the equation and compare it across different classes to select the class membership.

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

$P(H_i)$ is the class prior probability which is the number of class i divided by the total number of data samples. Also P(X) is the prior probability of the data sample X= (x_1, x_2, …, x_d) which is equal to $P(X) = P(x_1)\, P(x_2) \ldots P(x_d)$ where $P(x_d)$ is the probability of value x_d in normal distribution of feature d.

In our implementation of the algorithm we looped through the rows of the test data set. For each data point we calculated the numerator of the equation for $P(H_i|X)$. Since we dealt with two class labels in our data sets, we assigned the class with the higher value of this numerator to each test data point we evaluated.

The $P(H_i)$ value which is the class prior probability was calculated by counting the number of each class label in the training data set divided by the total number of the train data sample. The second term in the numerator which is the posterior probability of X given H_i, was calculated based on the assumption that the features are independent. After mapping the features in the data set into float values, for the data samples corresponding to label H_i, we multiplied the conditional probability of $P(x_j|H_i)$ over the entire features in the given sample. This conditional probability was calculated using the probability density function (pdf) of the normal distribution (another simplifying assumption in the algorithm). Eventually each sample in the test data set was assigned a class label by comparing $P(H_1) * P(X|H_1)$ against $P(H_2) * P(X|H_2)$ and choosing the class with the higher value.

Pros

This is an efficient and fast algorithm which makes it comparable in performance to other efficient algorithms such as decision trees. Also, the algorithm is efficient when applied to large databases.

Cons

It is a simple classifier that assumes attribute are independent. However, in reality the features are correlated most of the times. Also, the distribution of each feature belonging to individual class labels are assumed to be normally distributed which can be an inaccurate assumption.

Decision Tree

We have stored the data in a data array where the rows represent the samples and the columns are feature values. Since the feature values could be comprised of continuous and string values, the column indices of the string values were extracted and each unique string value was replaced by an integer starting from 0. Then we implemented the decision tree algorithm as follows.

In our implementation of the decision tree algorithm the split in each root was carried out by selecting the best split value in the best feature from the training data. This process was done by trying every single sample value of every feature in the dataset. The best split value and best feature was selected by calculating the gain in each iteration. The split with the highest gain was selected as the split value at each root. The splitting process was continued until one of the stopping criteria was met. These stopping criteria included the following:

1- If a branch had only one class value
2- If the branch had a size less than minimum node size
3- If the depth of the split reached the maximum tree depth

Next, to select the class value of the test dataset, each test data point was evaluated against the first split root. If the feature value of the test data point was less than the split value of the split feature in the corresponding root of the decision tree the data point was compared against the left branch and it was compared against the right branch. If the branch it was compared against was a class value, the class value was assigned to the test data point. If the branch value was of dictionary type, this process was continued recursively until it reached the branch with class value. This was done for all the data in the test data points until every data point was assigned a prediction class value.

We used gain as the cost function to find the best split in the construction of the tree. We calculated the gain by calculating the gini index. The equations for the gini index and gain are as follows:

$$Gini\ Index = 1 - \sum_{i} [P(i|t)]^2$$

Where p(i|t) is the relative frequency of class i at point t.

The gain was next calculated by subtracting the gini index of the children nodes from the gini index of the parent node.

$$Gain_{split} = gini\ index_p - \frac{n_1}{n} \times gini\ index_{child1} - \frac{n_2}{n} \times gini\ index_{child2}$$

Where p is the parent node, n_1 is the number of data samples in the left branch, n_2 is the number of data samples in the right branch and n is the number of data samples in the parent node.

Pros

Decision tree is an in expensive and fast algorithm. The accuracy of the algorithm is comparable to other classification techniques for simple data sets. Also, the algorithm is easy to interpret for small size trees. Moreover, the algorithm is robust to outliers.

Cons

Decision tree algorithm is prone to overfitting. Also, this algorithms may produce poor result if the test data distribution is different from the train data.

Random forest

In our random forest implementation, we performed bagging to improve the performance. To implement the bootstrap aggregation (bagging) we selected a sample of the training data with replacement to build each tree. Each time we selected a sample of equal length of the original training data set. Next, we randomly selected 20% and 45% of the features for data set 1 and data set 2, respectively to find the best split using the "gain" cost function and built a decision tree. Ten trees were constructed using the previously developed decision tree algorithm. The test data set's class was predicted using each of the ten decision tree and the final class of each data point in the test data set was predicted by taking the majority vote of the classes from all the trees.

We implemented this algorithm on each of the training and test data sets from the 10-fold cross validation data and computed the accuracy measures. The measures in each of the cross validation data as well as the average measures are reported under the results for random forest.

Pros

Random forest algorithm is a suitable fit for large data with large number of features.

Cons

Unlike decision tree algorithm, random forest is difficult to interpret. Also, choosing the number of trees as a hyper parameter is a disadvantage in this algorithm.

Kaggle competition

For Kaggle competition we implemented 6 different classification algorithms from sklearn package as follows: KNN, SVM, Naïve Bayes, logistic regression, random forest, and decision tree. We used ensemble learning by taking the majority vote from the prediction outputs of these algorithms.

Using SVM on the provided data generated an error which was due to the large size of the data. Therefore, we decided to skip the SVM algorithm. Also, since we are using the majority of the class prediction values, odd number of prediction/ algorithms makes more sense.

Moreover, we implemented bootstrap aggregation (bagging) and sampled new training data from the original training data. This sampling with replacement was implemented to generate the new sampled training data 1000 times as large as the original training data. This increased our accuracy measure drastically.

Accuracy (general)

In each algorithm, to evaluate the accuracy of the predicted class labels, four accuracy measure were calculated in each iteration of the cross validation and the average measure was reported (along with the measures in each iteration of the cross validation). These parameters are accuracy, precision, recall, and f1-measure. These parameters are computed as follows:

$$Accuracy = \frac{a + d}{a + b + c + d}$$

$$Precision = \frac{a}{a + c}$$

$$Recall = \frac{a + d}{a + b + c + d}$$

$$F1\ measure = \frac{2a}{2a + b + c}$$

Where a is the number of true positives, b is the number of false negatives, c is the number of false positives, and d is the number of true negatives.

Cross validation (general)

We implemented 10-fold cross validation in each of the classifying algorithms. To that end, we first split the data into 10 roughly equal length parts by rows. Each part was assigned as the test data and the remaining parts (9 parts) were concatenated to form the training data. The test data adopted each of the parts in a recursive process (10 iterations) and the training data were formed accordingly. In each iteration the algorithm was trained on the training data and was evaluated on the corresponding test data by calculation of the accuracy measures. These measures are reported in each iteration in the results section of the report and the average of the measures over the 10 iterations are calculated and reported as well.

Results

K-Nearest Neighbors

The accuracy in each iteration of the cross validation from the K-nearest neighbor algorithm is presented below for data set 1 and data set 2.

Cross validation iteration	project3_dataset1		project3_dataset2	
1	accuracy	0.982456	accuracy	0.510638
2	accuracy	0.982456	accuracy	0.744681
3	accuracy	0.947368	accuracy	0.673913
4	accuracy	0.982456	accuracy	0.717391
5	accuracy	0.912281	accuracy	0.630435
6	accuracy	0.947368	accuracy	0.543478
7	accuracy	0.964912	accuracy	0.717391
8	accuracy	0.929825	accuracy	0.804348

9	accuracy	0.894737	accuracy	0.673913
10	accuracy	0.928571	accuracy	0.695652

The average accuracy measures in our 10-fold cross validation from K-nearest neighbor is presented below for data set 1 and data set 2.

project3_dataset1		project3_dataset2	
average accuracy	0.947243	average accuracy	0.671184
average precision	0.943624	average precision	0.569962
average recall	0.91603	average recall	0.382622
average f1 measure	0.926601	average f1 measure	0.436436

Naïve Bayes

The accuracy in each iteration of the cross validation from the Naïve Bayes algorithm is presented below for data set 1 and data set 2.

Cross validation iteration	project3_dataset1		project3_dataset2	
1	accuracy	0.947368	accuracy	0.659574
2	accuracy	0.929825	accuracy	0.702128
3	accuracy	0.964912	accuracy	0.782609
4	accuracy	0.912281	accuracy	0.695652
5	accuracy	0.877193	accuracy	0.695652
6	accuracy	0.964912	accuracy	0.608696
7	accuracy	0.964912	accuracy	0.782609
8	accuracy	0.964912	accuracy	0.73913

9	accuracy	0.912281	accuracy	0.586957
10	accuracy	0.892857	accuracy	0.630435

The average accuracy measures in our 10-fold cross validation from Naïve Bayes algorithm is presented below for data set 1 and data set 2.

project3_dataset1		project3_dataset2	
average accuracy	0.933145	average accuracy	0.688344
average precision	0.913193	average precision	0.542202
average recall	0.904443	average recall	0.695079
average f1 measure	0.907873	average f1 measure	0.603425

Decision tree

The accuracy in each iteration of the cross validation from the decision tree algorithm is presented below for data set 1 and data set 2.

Cross validation iteration	project3_dataset1		project3_dataset2	
1	accuracy	0.929825	accuracy	0.595745
2	accuracy	0.929825	accuracy	0.702128
3	accuracy	0.964912	accuracy	0.630435
4	accuracy	0.929825	accuracy	0.543478
5	accuracy	0.859649	accuracy	0.673913
6	accuracy	0.982456	accuracy	0.608696

7	accuracy	0.912281	accuracy	0.652174
8	accuracy	0.964912	accuracy	0.5
9	accuracy	0.929825	accuracy	0.5
10	accuracy	0.875	accuracy	0.543478

The average accuracy measures in our 10-fold cross validation from decision tree algorithm is presented below for data set 1 and data set 2.

project3_dataset1		project3_dataset2	
average accuracy	0.927851	average accuracy	0.595005
average precision	0.902588	average precision	0.438952
average recall	0.904858	average recall	0.500554
average f1 measure	0.901808	average f1 measure	0.454606

Random forest

The accuracy in each iteration of the cross validation from random forest algorithm is presented below for data set 1 and data set 2.

Cross validation iteration	project3_dataset1		project3_dataset2	
1	accuracy	0.929825	accuracy	0.595745
2	accuracy	0.929825	accuracy	0.659574
3	accuracy	0.982456	accuracy	0.608696
4	accuracy	0.964912	accuracy	0.717391

5	accuracy	0.894737	accuracy	0.695652
6	accuracy	0.947368	accuracy	0.586957
7	accuracy	0.982456	accuracy	0.630435
8	accuracy	0.982456	accuracy	0.73913
9	accuracy	0.912281	accuracy	0.630435
10	accuracy	0.928571	accuracy	0.630435

The average accuracy measures in our 10-fold cross validation from random forest algorithm is presented below for data set 1 and data set 2.

project3_dataset1		project3_dataset2	
average accuracy	0.945489	average accuracy	0.649445
average precision	0.928497	average precision	0.506536
average recall	0.930023	average recall	0.524249
average f1 measure	0.928471	average f1 measure	0.501127