Assignment 3: Implementation of PCA and k-means algorithms

TAK YIN PANG
University of Adelaide
a1796036@adelaide.edu.au

Abstract

PCA stands for Principal Component Analysis which is commonly used to reduce the dimensions of the dataset. K-means is an unsupervised clustering method that groups the datapoints belong to the nearest mean cluster.

1. Introduction

In thi assignment, I will explain the algorithm of PCA and K-mean in the below. Then, I will show some analysis on both of the algorithm.

2. Description of the algorithm

2.1. PCA algorithm

PCA algorithm is an orthogonal linear transformation that transforms the data into a new coordinate system. The transformation is defined as below.

$$T = XA$$

where
T is a transformed matrix (N x l)
A is a matrix of orthonormal eigenvectors of matrix S (d x l)
X is a matrix of the dataset (N x d)
Matrix S is either the scatter matrix or the covariance matrix of X. (d x d)
N is the number of datapoint
d is the number of dimension(feature)
l is the reduced number of dimension

For a scatter matrix, it is calculated as below.
$S = \sum_{k=1}^{n}(x_k - m)(x_k - m)^T$
where m is the mean vector
$m = \frac{1}{n}\sum_{k=1}^{n} x_k$

For convariance matrix, it is very similar to scatter matrix but with a scaling factor $\frac{1}{n-1}$, n is the number of data points.

2.1.1 Implementation and intuition

Fristly, we calculate the mean vector of the dataset for each dimension. Then we subtract the data by the mean vector to zero center the data. After this, we compute the coviriance matrix and sort the eigen vectors according the eigen value. Here, we select the top L eigen vectors with highest eigen value and stack them into a matrix A. Finally, the transformation is done by taking a dot product of X and A to become T.

2.2. K-mean algorithm

2.3. Classifier weight - Alpha

Figure 1. Alpha values against Error Rate [**?**]

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$$

The alpha value is defined as the above formula.From Figure 1, there is three intuitions to note. First, the alpha value is zero when the error rate is 0.5. Since $\epsilon = 0.5$ means that the classifier is the same as random guess, it is reasonable to assign zero weight to the classifier. Second, the alpha values grow exponentially when the error approaches zero. More weight is assigned to the classifier if the error rate is low. Third, the alpha values become negative when the error is larger than 0.5. Negative weight means that do the opposite prediction for these poorly performed classifiers.

2.4. Weak learner and decision stump

Weak learner is defined as the classifier with a prediction error $\epsilon <= 0.5$. Decision stump is the classic default choose of the weak classifier for the adaboost algorithm but the weak learner choice in adaboost algorithm can be any classifier perform better than randon guess.

Figure 2. Decision Stump [**?**]
Decision stump is a decision tree with 1 depth which means that it splits the data points into two class nodes based on one attribute(feature) value threshold. In Figure 2, it illustrates how a decision tree looks like.

To determine the goodness of the spilt, there is several criterions including gini impurity and information gain.

For gini impurity, the split attribute value is selected if the impurity is the least.

$$gini = 1 - \sum_c p(c)^2$$
$$impurity = \frac{leftSample}{totalSample} * leftGini + \frac{rightSample}{totalSample} * rightGini$$

For information gain, the split attribute value is selected if the gain is the highest.

$$entropy = \sum_c p(c)log(p(c))$$
$$informationGain = \frac{leftSample}{totalSample} * leftEntropy + \frac{rightSample}{totalSample} * rightEntropy$$

3. My understanding of Adaboost

In this section, I will talk about several properties about the adaboost algorithm.

3.1. Hyperparameter

Adaboost, unlike the other machine learning algorithms, has less hyperparameter to choose during the training process. For adaboost, there are two hyperparameters to tune. The frist one is the choice of weak learner. Although decision stump is a default choice for the adaboost algorithm, every classifier that is better than a random guess can be selected as the weak classifier. The second one is the number of iteration. The number of iteration highly depend on the dataset. I will talk about them one by one.

3.1.1 Choice of classifier

For a weak classifier like decision stump, it does not fit the training data well and has a high bias. Adaboost algorithm works well on reducing the bias by boosting the weak learners in sequencial order. The weight put on the misclassified data points helps the next classifier corrects the mistakes made by the last classifier and hence help to reduce the bias.

In theory, every classifier that is better than random guess (prediction error ¡ 0.5) can be applied to adaboost algorithm. However, the result of ensembling strong classifiers in adaboost may or may not perform well. This may be because the strong classifier has already low bias and it does not help to push the bias lower by boosting.

3.1.2 Number of iteration

The number of iteration depends on the problem and the training data. The default number of iteration in sklearn adaboost classifier is 50. Some suggest early stopping [?] should be applied while Mease and Wyner (2008) [?] argue adaboost should run for a long time until it converges.

3.2. Overfitting

Although adaboost might overfit, it is quite robust to overfitting. It depends on several factors [?] whether it will overfit or not. The first one is the "strength" of the "weak" learners. It is less likely to overfit if the weak learner is very simple like decision stump. The second one is the noise level in the data. Adaboost is prone to overfitting on the noisy datasets. The last one is the dimensionality of the data. It is easier to overfit in high dimensional spaces.

3.3. Outliner

Adaboost is excellent in identifying the outliners in the training data. [?]This is becasue weight will repeatly put on the outliners during the iteration process. The final weight on the those outliners will become very high because most weak classifiers fail to classify them correctly.

4. Analysis

4.1. Introduction

In this assignment, I decide to experiment on the different setting of the adaboost algorithm on breast cancer dataset. Frist, I will compare the performace on the adaboost algorithm with the sklearn AdaboostClassifier as well as the parameter. Then, I will explore the impact of number of iteration and the depth of the decision tree classifier used in adaboost algorithm. Lastly, I will compare the performace of SVM classifier with different kernels with adaboost.

4.2. Experiment set up

In the experiment, Wisconsin Diagnostic Breast Cancer dataset is used for testing the adaboost algorithm. The first 300 samples are the training data and the remaining 269 samples are the testing data. The target label is changed from M,B to -1,1 for adaboost. Sklearn DecisionTreeClassifier is used as the base weak learner for both implemented adaboost and 3rd party adaboost. Gini coefficient is used for the split criterion.

4.3. Comparison with 3rd party adaboost

Table 1. Performance comparison with 3rd party sklearn adaboost

As shown in table 1, the train and test accuracy performance of both algorithm are exactly the same. However, the time performance of my implementation is faster than the sklearn model. I guess the reason is that my impletation did not include parameter like learning rate and random state which are in sklearn adaboost classifier. Then I further compare the alpha values and error rate in each weak classifier.

In figure 3, it shows the values of alphas of each iteration. The values in sklearn model is double to the values of my implementation but it will not affect the prediction of the final ensemble models.

In figure 4, the error rates of each weak classifier are exactly the same. Therefore, I can conclude that my implementation perform as well as the sklearn implementation. And yet the run time of my implementation is even slightly faster.

4.4. Performance with different setting

In this section, I experiment the impact on the performance of the adaboost algorithm of the depth of decision tree as well as number of iteration. I run adaboost with decision tree depth from 1 to 3 and number of iteration from 1 to 200 and plot the train error vs test error agaist number of iteration with respect to different depth.

The above figures show the the training error as well as the testing error with different depth against the number of iteration. All of them show that the train error reduce to zero very quickly while the test error drop significantly at frist then slowly converage at around 150 iterations. In this dataset, overfitting does not seem to occur and the test error still fluctuate for a long time after the train error dropped to zero.

The training error of the one with decision tree depth of 1 decreased to zero much slower then the other one with higher depth. However, the test error reached by the simpler model is the lowest at 0.015 at around 130 iteration. Meanwhile the test error with decision tree depth of 3 decrease faster but fail to converage in a low error rate than the one with lower depth.

4.5. Performance comparison with SVM classifier

Table 2. Performance comparison with SVM classifiers

In table 2, it shows that adaboost algorithm outperform all the svm classifier with different kernels and parameter C = 1. Among three kernels, SVM classifier with linear kernel performs the best and reach a 0.96 train accuracy and 0.95539 test accuracy. Also, it has the slowest train time but the fastest test time.

4.6. Boosting on SVC

I tried to applied the above svm classifier on the adaboost algorithm. Since they are not weak classifier, I would like to experiment on the behavior applying strong classifier on adaboost algorithm. I run them on adaboost for 10 iteration and plot the error curve for each of them.

5. Code

The code can be simply run by:

```
python3 Adaboost.py
```

6. Conclusion

The adaboost algorithm perfroms well on this dataset and achieve a very high train and test accuracy. Also, overfitting does not happen in this dataset and the algorithm converages slowly with more than 100 iterations. My implementation performs as well as the 3rd party adaboost classifier.

Adaboost has a better boosting performace on weak classifers. When using decision tree with depth of 3, it does not perform as well as the simpler model. Also it suffers a very strange bahavior when appling strong classifiers like svm classifier on the adaboost.

References

[1] Andrzej Makiewicz and Waldemar Ratajczak. Principal components analysis (pca). *Computers Geosciences*, 19(3):303 – 342, 1993.