AMATH 563 Homework 3: Machine Learning on Extended Yale Faces B Database

Gloria(Tiange) Tang

5/26/2020

Abstract

This study is based on Extended Yale Faces B Database for a variety of tasks such as eigenface detection, individual face classification, gender recognition, and unsupervised clustering for faces.

1 Introduction and Overview

This study consists of four major parts, *Eigenface Detection* using Singular Value Decomposition, *Face Classification* using Naive Bayes, Suport Vector Machines, Extreme Gradient Boosting, and Random Forest, *Gender Recognition* using Logistic Regression, Random Forest, Support Vector Machines, Linear Discriminant Analysis, Naive Bayes, and *Unsupervised Clustering for Faces* using K-means Clustering. The dimensionality of Extended Yale Faces B Database is reduced by Principal Component Analysis before *Face Classification* and *Gender Recognition* to improve model performance and computational efficiency. With the thereotical foundation and experimental results, used algorithms are compared for each part respectively.

2 Theoretical Background

2.1 Singular Value Decomposition(SVD)

In linear algebra, SVD states that any matrix X can be factorized as

$$X_{m*n} = U_{m*m} S_{m*n} V^T_{n*n} \tag{2.1.1}$$

where U and V are orthogonal matrices with orthonormal eigenvectors chosen from XX^T and X^TX respectively.S is a diagonal matrix with r elements equal to the root of the positive eigenvalues of XX^T or X^TX. An important property of the SVD is that it provides an optimal low-rank approximation to a matrix X. The eigenvalue matrix S indicates the importance of each principal components(Columns of U).V shows how the data X is projected onto the principal components.

2.2 Principal Component Analysis(PCA)

The goal of PCA is to reduce te dimensionality of the input data while retaining the variation present as much as possible. PCA can remove correlated features, speed up the performance of the algorithm, and reduce noises.
The algorithm works as follow:

1. Perform SVD of the normalized data matrix $X = USV^T$
2. Sort eigenvalues in descending order and choose the k eigenvectos W from U that correponds to the k largest eigenvalues where k is the number of dimensions of the new feature subspace
3. Compute the principal components $Y = WX$

2.3 Naive Bayes(NB)

Naive Bayes is based on the Baye's Theorem to determine the class of a query point. A critical assumption of this algorithm is that the predictor features are independent of each other.

$$P(S|x_1, x_2, ..., x_n) \propto P(x_1|S) * P(x_2|S)...P(x_n|S) * P(S)$$

where S is a class of prior belief. The prediction is the class with the biggest posterier probability. In this study Gaussian Naive Bayes is used where we assume the prior distribution of each input feature follows a Gaussian distribution.

2.4 Suport Vector Machines(SVM)

The objective of applying SVM is to find the decision boundary with the largest margin(the maximum distance between data points of both classes) for classification. For linear supprt vector clasifier, define the decision boundary $w^T X + b = 0$ where w is the weight vector and b is the bias. We need to minimize the cost function that helps maximize the margin and prevent overfitting:

$$L(w) = \sum_{i=1}^{n} C[max(0, 1 - y_i[w^T x_i + b])] + \|w\|_2^2$$

where n is the number of the trainining samples, y_i is either 1 or -1. The function of the first term, hinge loss is to penalize misclassifications. The second term is for the regularization to avoid overfitting. If the decision boundary is non-linear, we use kernel trick which takes lower dimensional input and transform it into a higher dimensional space. In this study, Radical Basis Function(RBF) kernel is used.

2.5 Selected Tree Based Methods

2.5.1 Decision Trees

A decision tree can be used to visually represent decision for classes. All the features and different split points are considered and tested with certain metrics, for example, entrophy and information gain. Entropy $H(S)$ is a measure of the amount of uncertainty in the data set S.

$$H(S) = \sum_{c \in C} -p(c) log_2 p(c)$$

where C is a set of classes in S, $p(c)$ is the proportion of the number of elements in class c to the number of elements in set S. And Information Gain is the measure of the difference in entropy from before to after the set S is split on an Attribute A.

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

where $H(S)$ is the entropy of S, T is the subsets created from splitting set S by attribute(node) A such that $S = \cup_{t \in T} t$, $p(t)$ is the proportion of the number of elements in t to the number of elements in set S, and $H(t)$ is the entropy of subset t.
The algorithm works as follow:

1. Calculate entropy of the target class
2. The input dataset is split on different attributes. Calculate the information gain on different attributes.
3. Choose attribute with the largest information gain as the decision node, divide the dataset by its branches and repeat the same process on every branch.
4. A branch with entropy of 0 is a leaf node. A branch with entropy more than 0 needs further splitting.
5. Run recursively on the non-leaf branches, until all data is classified.

2.5.2 Extreme Gradient Boosting(XGBoost)

The XGBoost algorithm is based on gradient boosting decision tree with empirically faster execution speed and better model performance. The gradient boosting classification algorithm implements an ensemble technique called Boosting with Gradient Descent to minimize the cross entrophy loss. The intuition is to repetitively leverage the patterns in the prediction residuals to strengthen a model with more weights on weak predictions in a sequential way. The algorithm works as follow:

1. Build the first tree with training set (X, Y)

2. Compute the predicted log-odds ratio values of training input X
3. Compute the residule $\vec{r}$ with learning rate α and fit the next tree with $(X, \vec{r})$
4. Repeat the process until the residuals do not have any patterns that could be modeled
5. Combine all the predictor trees for future prediction

For multi-class problem, we build K one-vs-rest trees per iteration where K is the number of classes.

2.5.3 Random Forest(RF)

RF is a set of decison trees from randomly selected subset of training set. Each subset is selected with replacement. It aggregates the votes from different decision trees to decide the final class of the query data.

2.6 Logistic Regression(LR)

LR is a slight modification of Linear Regression to produce binary outputs. The hypothesis is

$$h\Theta(X) = sigmoid(w^T X + b)$$

To predict which class a data belongs, a threshold on $h\Theta(X)$ can be set. The loss function is defined as

$$L(h\Theta(X), Y) = \frac{1}{n}[\sum_{i=1}^{n} -y^i log(h\Theta(x^i)) + (1 - y^i) log(1 - h\Theta(x^i))]$$

where n is the number of data in the training sample (X, Y). LR uses gradient descent to minimize the loss function and get the maximum likelihood estimation of w.

2.7 Linear Discriminant Analysis(LDA)

The goal of LDA is to project a dataset onto a lower-dimensional space with good class-separability. LDA assumes that each attribute of the training data has a Gaussian distribution with the same variance.

2.8 K Means Clustering(K Means)

K Means is an unsupervised algorithm to a given data set into a number of clusters K. The algorithm works as follow:

1. K points are placed into the object data space representing the initial group of centroids

2. Each object or data point is assigned into the closest cluster

3. After all objects are assigned, the positions of the K centroids are recalculated.

4. Steps 2 and 3 are repeated until the positions of the centroids no longer move.

One popular method to determine the number of clusters K is the elbow method where we fit the data to a number of different Ks. As K increases, the Within-Cluster-Sum-of-Squares(WCSS) which measures sum of distances of observations from their cluster centroids are gettings smaller. Plotting WCSS against increasing K can show an 'elbow' which demarks significant drop in rate of increase. Selecting number of clusters corresponding to elbow point achieves reasonable performance without having too many clusters.

3 Algorithm Implementation and Results

The data comes from Extended Yale Faces B Database with two versions, cropped and uncropped. There are 38 headshots in the cropped dataset with a total of 2433 pictures. There are 15 headshots in the uncropped dataset with a total of 165 pictures. The pictures in both datasets are flattened to single arrays of pixels.

3.1 Eigenface Detection

For this task, we conduct Singular Value Decomposition(2.1) on both cropped data and uncropped data. For cropped images, the energy contained in the first 161 eigenvectors (threshold: eigenvalue>0.001) is 58.3%. For uncropped images, the energy contained in the first 150 eigenvectors (threshold: eigenvalue>0.001) is 99.6%.

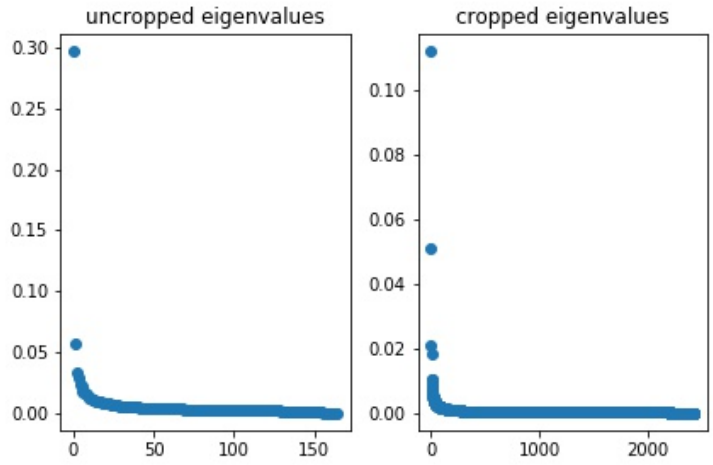

Figure 1: Eigenvalues of Uncropped Data and Uncropped Data

3.2 Unsupervised Clustering for Faces

K Means Clustering(2.8) is performed on the cropped data to find the clusters of faces. We select the optimal number of clusters from 1 to 10 with Elbow Method.

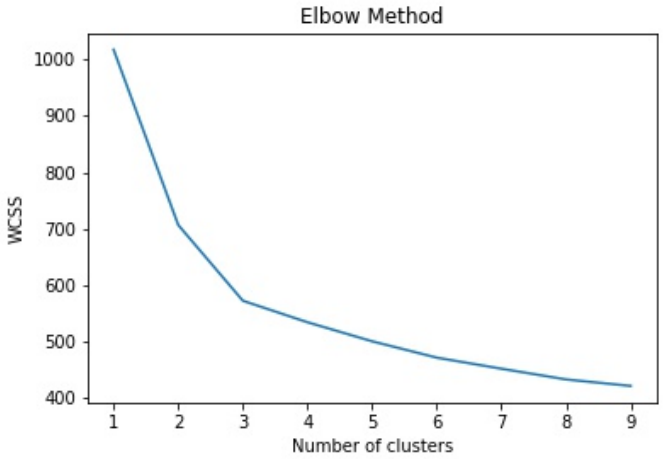

Figure 2: Elbow Method

The elbow point is 3 as shown in the above picture. Then the training data is fit into the algorithm with 3 clusters. We can visualize the clusters with centroids in eah cluster as follow:

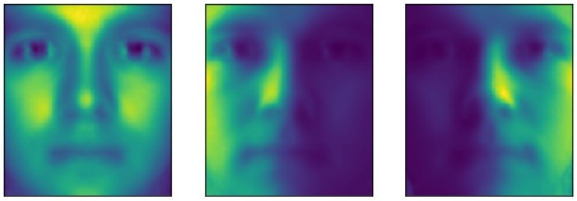

Figure 3: Centroids with 3 Clusters

For the following sections *Face Classification* and *Gender Recognition*, the training set is the 160 (because it is close to the number of dominant features from Singular Value Decomposition of Cropped Images in 3.1) principal components from PCA(2.2) on normalized cropped dataset to improve the following algorithms' computational efficiency and performance.

3.3 Face Classification

Gaussian Naive Bayes(2.3), Support Vector Machines with RBF kernel(2.4), Extreme Gradient Boosting(2.5.2) and Random Forest(2.5.3) are selected to classify the 38 faces with 5-fold cross validation. Due to the computational limitation, hyperparameters in each model are set to default values.

Method	Accuracy Score	F-1
NB	0.813	0.813
SVM	0.920	0.920
XGBoost	0.855	0.864
RF	0.910	0.904

Table 1: Face Classification Results

3.4 Gender Recognition

Logistic Regression(2.6), Gaussian Naive Bayes(2.3), Support Vector Machines with RBF kernel(2.4), Linear Discriminant Analysis(2.7) and Random Forest(2.5.3) are selected to classify the gender of 38 faces with 5-fold cross validation. Due to the computational limitation, hyperparameters in each model are set to default values.

Method	Accuracy Score	F-1
LR	0.967	0.978
SVM	0.964	0.965
LDA	0.961	0.961
RF	0.941	0.945
NB	0.723	0.721

Table 2: Gender Recognition Results

4 Summary and Conclusions

As shown in Figure 1, the first few eigenvectors of uncropped dataset generally take up a higher percentage in reconstructing the data than the same amount of eigenvectors of the cropped dataset. This is due to the fact that uncropped dataset has more noises than cropped dataset resulting in larger variation in the uncropped eigenvalues .
For Unsupervised Clustering for Faces, as shown in Figure 3, it is very likely that K means is clustering portraits based on the shades of the photos. If the shades and lighting of the portraits for each person are consistent, we might get a different cluster result based on facial features.
As shown in Table 1, Support Vector Classifer outperforms all other selected algorithms for face classification. It is probably because SVM works very well on sparse features and high dimensionality data such as images, and it also tends to overfit less than the other models. In both face classification and gender recognition, all models except for Naive Bayes have pretty decent performances. Since Naive Bayes assumes the features are conditional independent, even though PCA is applied, it cannot guarantee that the conditional independence will be improved and the assumption of Naive Bayes can be satisfied.

A Main Python Functions Used in this Study

1.sklearn.preprocessing.normalize()
Normalize the data

2.numpy.linalg.svd()
Find singular value decomposition of a matrix.

3.sklearn.decomposition.PCA()
Perform principal component analysis of the input

4.sklearn.naive_bayes.GaussianNB()
Perform Gaussian Naive Bayes

5.sklearn.svm.SVC()

Support Vector Classifier

6.xgboost.XGClassifier()
Extreme Gradient Boosting

7.sklearn.ensemble.RandomForestClassifier()
Random Forest

8.sklearn.linear_model.LogisticRegression()
Logistic Regression

9.sklearn.discriminant_analysis.LinearDiscriminantAnalysis()
Linear Discriminant Analysis using Bayes Rules

10.sklearn.cluster.KMeans()
K Means Clustering

B Original Jupyter Notebook

Please see `https://github.com/gloriatang0325/AMATH-563`