ECS797P COURSEWORK ASSIGNMENT 3: FACE AGE ESTIMATION

Animesh Devendra Chourey – 210765551

Queen Mary University of London – April 29, 2022

Getting Started

In this coursework, we are needed to estimate the age of the given facial images. To achieve this task we need to model a regression function that takes the *Active Appearance Model* (AAM) parameters as representation of human faces so that it can predict the age of the unseen faces. AAM is a statistical model of the shape and appearance of a deformable class. They have been widely used in face recognition and face modelling.

To train the regression model, *FG-NET Aging* dataset is used which contains 1002 high-resolution as well as gray-scale images. The dataset contains images from 82 disparate races with huge variations of lighting, pose and expression. On average, every person has 12 images each. *data_age.mat* file contains both the training and testing data separated evenly. The data in the .mat file is in the form of AAM feature vector along with the true age label for each corresponding image. The age ranges from 0 to 69 years with each image present in chronological order.

Complete the lab3.m file

1. Read in the training and test data by loading the ".mat" file

Training and test database is loaded from the *data_age.mat* file. On loading the file we can see that the number of training samples($nTrain$) are 500 and the number of test samples($nTest$) are 502. Training features and labels are stored in the variables *xtrain* and *ytrain* respectively. Similarly, the features and labels from test data is stored in *xtest* and *ytest* respectively. Number of AAM features extracted from each image are 201. This gives the *xtrain* size of (500x201) and *xtest* the size of (502x201).

2. Call the Matlab built-in function "regress()", which takes the training features and labels as input and learn a linear regression model. Read Matlab document to understand which linear regression model is implemented.

The *regress()* function is used here to build a *"multiple linear regression model"*. Multiple Linear Regression model attempts to model the relationship between two or more explanatory variable and a target variable by fitting a linear equation on the data. The function takes the labels and training features as its parameters and returns coefficient estimates for the multiple linear regression model created. The labels are in the form of vector and the training features are are in the form of matrix to be passed to the *regress()* function. *w_lr* variable stores the coefficients for the multiple linear regression model.

3. Read in the test data, and apply the learned linear regression model to estimate the age for each test data point

In this section, the multiple linear regression model is applied to the test data. The coefficients(weights) of the model are multiplied with the test data and the resulting values are stored in the variable *yhat_test* (502x1). These values are the estimation(prediction) of age for the test images provided. One thing to note here is that if the AAM features corresponds to an age closer to 0 the trained model may predict a negative age beacuse of the linear mapping done by the model between the features and the labels.

4. Compute the MAE and CS value (with a cumulative error level of 5) by comparing the estimated ages with the ground truth ages. You need to write your own code here.

Firstly, Absolute Error (*abs_error*) here calculates the error while predicting the age of the faces. The predicted value is subtracted from the ground truth value. Their absolute values of the difference is stored. Then the Mean Absolute Error (*mean_abs_error*) is calculated which is the average of absolute error between the estimated ages and the ground truth. The MAE calculated by the model is **7.704**. This indicates that the mean error in predictions by the model is 7.704 years. The code for the following is :

```
% Compute the absolute error of each test image
abs_error = abs(yhat_test - ytest);
% Mean of the absolute error
mean_abs_error = mean(abs_error);
```

Output :

Mean Absolute Error obtained by the Linear Regression model is: **7.70435**

The CS values is calculated by comparing the absolute error with the threshold (*err_level*). This comparison indicates the percentage of predictions done by the model whose values are less than or equal to the threshold value. Here what that means is we are trying to calculate the percentage of predicted ages that have error values less than or equal to 5 years. The code for the following is :

```
% Check if the absolute error is greater than the threshold value
% Calculate the percentage of error values greater than the threshold value
cs_value = sum(abs_error <= err_level) / nTest *100;
```

Output :

CS value obtained by the Linear Regression model is: **41.23**

41.23% of predictions have an error less than or equal to 5 years.

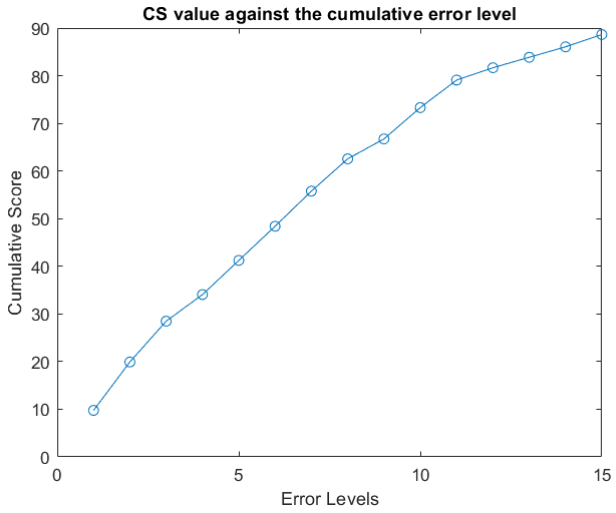
5. Vary the cumulative error level from 1 to 15 and generate a plot of the CS value against the cumulative error level. You need to write your own code here. See the lecture slides for Week 6 for an example of the plot

In this section, we are needed to plot the cumulative score for different cumulative error levels. To generate the plot the following snippet was used:

```
% Create an array to store the cumulative score for different error levels
cumulative_score = zeros(15,1);
% Calculate the score for 1:15 error levels
for i = 1:15
    cumulative_score(i) = sum(abs_error <= i) / nTest *100;
end

figure
plot(cumulative_score, '-o');
xlabel('Error Levels');
ylabel('Cumulative Score');
title('CS value against the cumulative error level');
```

The graph below show that the cumulative score is increasing with increasing error levels. This is because we are expanding the miss-classified images range. The more the range is expanded, more miss-classified images will fall under the category, thus simultaneously elevating the cumulative score. The following graph was obtained:



6. Compute the MAE and CS values (with cumulative error level of 5) for both partial least squares regression model and the regression tree model by using the Matlab built-in functions

- **Partial Least Squares Regression Model :**

  It is a technique in which predictors are reduced to smaller sets of unrelated components. Least squares regression is then performed on these reduced components. This technique combines the features of principal component analysis and multiple regression. The emphasis is on developing predictive models. To enact this model, *plsregress()* function is used. The function returns a matrix *BETA* which are the coefficient estimates of our PLS regression model. A column of ones is added to the matrix X (here *xtrain*) to compute coefficient estimates for the model. After getting the

  The code to compute MAE and CS values for partial least regression model :

```
%Training the PLS Regression model
[XL,YL,XS,YS,BETA,PCTVAR,MSE,stats] = plsregress(xtrain, ytrain, 10);
% Estimate the model on the test data
yhat_plsr = [ones(nTest,1) xtest] * BETA;

% Compute the absolute error of each test image for PLS model
abs_error_plsr = abs(yhat_plsr - ytest);
% Mean absolute error of PLS Regression model
mean_abs_error_plsr = mean(abs_error_plsr);

% Check if the absolute error is greater than the threshold value
% Calculate the percentage of error values greater than the threshold value
cs_value_plsr = sum(abs_error_plsr <= err_level) / nTest *100;
```

  Output :

  Mean Absolute Error obtained by the Partial Least Square Regression model is: **6.0702**
  CS value obtained by the Partial Least Square Regression model is: **52.58**

  52.58% of predictions have an error less than or equal to 5 years.

- **Regression Tree Model :**

  Regression Tree is the model that is used for the task of regression which can be used to predict continuous valued output instead of discrete valued outputs. This regression tree is built by binary recursive partitioning. It is an iterative process that splits the data into branches. The model is fit to the target variable using each of the independent variable. After this the data is splitted at several points for each independent variable. The function used to perform the model operations is *fitrtree()* which returns a regression tree based on the input variables X

(here *xtrain*) and the output Y (here *ytrain*). The tree returned is binary tree whose each branching node is split according to the values of a column of X. Then the *predict()* function is used which returns a vector of predicted responses for the predictor data in the matrix X(here *xtest*) based on the regression tree. We get the predicted values as *yhat_regTree*. These predicted values are compared with ground truth values *ytest* to compute the mean absolute error(*mean_abs_error_regTree*). Then, the CS values are calculated similarly as done before for the previous models.

```
%Training the Regression Tree model
tree = fitrtree(xtrain,ytrain);
% Estimate the model on the test data
yhat_regTree = predict(tree, xtest);

% Compute the absolute error of each test image for Regression Tree model
abs_error_regTree = abs(yhat_regTree - ytest);
% Mean absolute error of Regression Tree model
mean_abs_error_regTree = mean(abs_error_regTree);

% Check if the absolute error is greater than the threshold value
% Calculate the percentage of error values greater than the threshold value
cs_value_regTree = sum(abs_error_regTree <= err_level) / nTest * 100;
```

Output :

Mean Absolute Error obtained by the Regression Tree model is: **8.235**
CS value obtained by the Regression Tree model is: **50.39**

50.39% of predictions have an error less than or equal to 5 years.

7. Compute MAE and CS values (with cumulative error level of 5) for Support Vector Regression by using LIBSVM toolbox (n by using the LIBSVM toolbox

Support Vector Regression follows the popular mechanism model Support Vector Machine (SVM). SVM are well known in classification problems. Support Vector Regression (SVR)is considered a non-parametric technique because it relies on kernel functions. SVR is a regression function that is generalized by SVM. SVR gives us the flexibility to define how much error is acceptable in our model and will find the hyperplane to fit the data. The objective function of SVR is to minimize the coefficients specifically the l2-norm of the coefficient vector.

We use *libsvm()* function to model the SVR algorithm. The function returns the full trained SVM regression model. This model is trained using the predictors value in X (here *xtrain*) and the response values in the vector Y (here *ytrain*). This trained model is then used to predict the values using the *predict()* function, which takes *Tb1* and *ytest* as its parameters. The predicted values are stored in the *yhat_svr* variable. After this, the mean absolute error (*mean_abs_error_svr*) and CS values (cs_value_svr) is calculated similarly as done before in other models.

The code to model the Support Vector Regression and to calculate the mean absolute error and CS values is as follow:

```
% Training the Support Vector Regression model
Tb1 = fitrsvm(xtrain, ytrain);
% Estimate the model on the test data
yhat_svr = predict(Tb1, xtest);

% Compute the absolute error of each test image for Support Vector Regression model
abs_error_svr = abs(yhat_svr - ytest);
% Mean absolute error of Support Vector Regression model
mean_abs_error_svr = mean(abs_error_svr);

% Check if the absolute error is greater than the threshold value
% Calculate the percentage of error values greater than the threshold value
cs_value_svr = sum(abs_error_svr <= err_level) / nTest * 100;
```

Output **-**
Mean Absolute Error obtained by the Support Vector Regression model is: **5.7313**
CS value obtained by the Support Vector Regression model is: **53.78**

53.78% of predictions have an error less than or equal to 5 years.

Table to compare the regression models:

Model	**Mean Absolute Error (MAE)**	**Cumulative Score (CS)**
Linear Regression	7.704	41.23
Partial Least Square Regression	6.07	52.58
Regression Tree	8.235	50.39
Support Vector Regression	5.73	53.78

As we can see from the table that the Support Vector Regression produces the best results with the smallest MAE and highest CS value.