Machine Learning for Visual Data Analysis – Age Estimation by Regression

Description

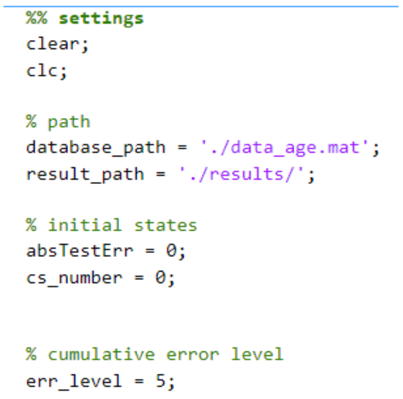
Background:

This assignment makes use of the FG-NET Aging Dataset. The dataset contains Active Appearance Model (AAM) features from 1002 face images, 500 of which are reserved for training and 502 for testing. The dimensions of AAM features are 201. The dataset includes images of 82 people of various ethnicities. Each person has about 12 photos with varying lighting, poses, and expressions. The dataset is labelled according to the age of the people, ranging from 0 to 69, and is ordered chronologically.

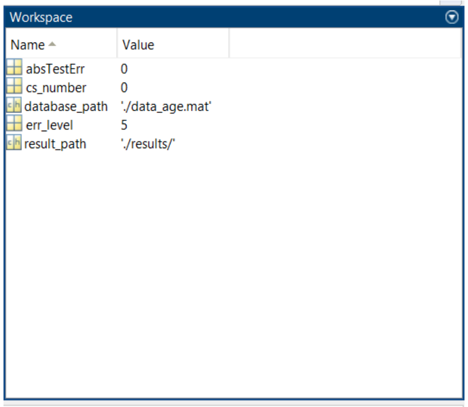
Age Estimation by Regression:

Section 1

When the given code section `%% settings` is executed successfully, following output was obtained.



```
%% settings
clear;
clc;

% path
database_path = './data_age.mat';
result_path = './results/';

% initial states
absTestErr = 0;
cs_number = 0;


% cumulative error level
err_level = 5;
```



```
Workspace
Name ▲          Value
absTestErr      0
cs_number       0
database_path   './data_age.mat'
err_level       5
result_path     './results/'
```

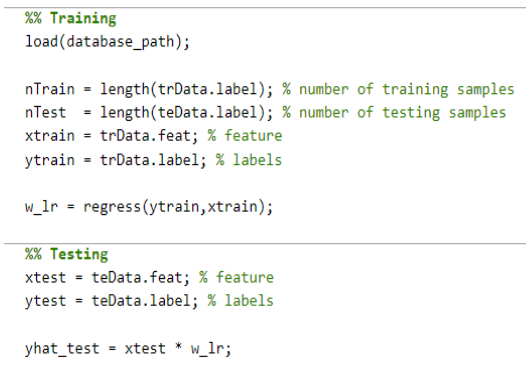
Section 2

1. The file data_age.mat contains the AAM feature vector and label pair. The MATLAB function *load(….)* which has been implemented as follows – '`load(database_path);`' could read this file. This command returns two struct variables, *teData* and *trData*, which represent testing and training data, respectively. These struct variables contain two variables: label (a vector of length N) and feat (a matrix of size N by 201), where N is the number of samples in the training and testing data.
2. In this section we will be using MATLAB function *regress(…)* which has been implemented as follows – '`w_lr = regress(ytrain,xtrain);`' to construct a linear regression model. The (multivariate) linear regression takes the following form, where beta are the weighting parameters and epsilon is the coefficient of determination.

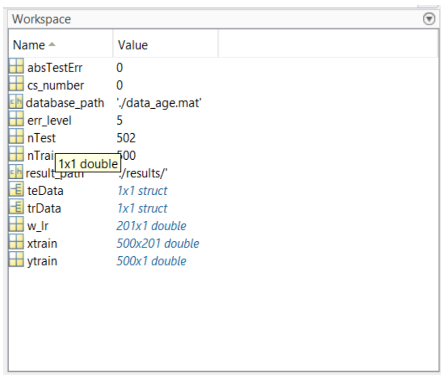
$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

3. The test dataset can be read using the same way when reading training data. The MLR regression is defined by $\boldsymbol{Y} = \boldsymbol{X}\,\boldsymbol{\beta}$. This can be realized easily by multiplying test features (of size 502 by 201) with the optimized weighting coefficients (of size 201 by 1).

Parts 1, 2 and 3 have been already implemented by the assignment, so the implementation details will be summarised. When the given code section `%% Training` and `%% Testing` is executed successfully, following output was obtained.



```
%% Training
load(database_path);

nTrain = length(trData.label); % number of training samples
nTest  = length(teData.label); % number of testing samples
xtrain = trData.feat; % feature
ytrain = trData.label; % labels

w_lr = regress(ytrain,xtrain);

%% Testing
xtest = teData.feat; % feature
ytest = teData.label; % labels

yhat_test = xtest * w_lr;
```



Workspace

Name	Value
absTestErr	0
cs_number	0
database_path	'./data_age.mat'
err_level	5
nTest	502
nTrain	500
result_path	'./results/'
teData	1x1 struct
trData	1x1 struct
w_lr	201x1 double
xtrain	500x201 double
ytrain	500x1 double

1x1 double

Section 3

MAE (Mean absolute error) represents the difference between the original and predicted values extracted by averaged the absolute difference over the data set. The Mean Absolute Error (MAE) is calculated as follows, where N is the total number of samples, y_i is the ground truth for the kth sample, and y_i is the predicted value for the ith sample:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}|$$

Where,
$\hat{y}$ − predicted value of y
$\bar{y}$ − mean value of y

Cumulative Score (CS) has the following formula, where N is the total number of samples and $\#e_j$ is the number of samples with an MAE of less than j years with respect to the ground truth:

$$CS(j) = \frac{1}{N} \#_{e\leq j} \times 100$$

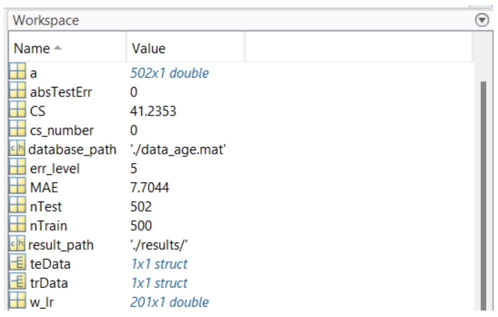
As seen in the ***Output*** section, for the MLR model cumulative error with 5 levels is found to be 41.2353 and the mean absolute error is recorded as 7.7044. MAE indicate the average error in predictions of MLR model which is found to be 7.70 years. Cumulative Error with a level of 5 indicate that 41.2% of the predictions have an error less than 5 years.

Code

```
%% Compute the MAE and CS value (with cumulative error level of 5) for
linear regression

MAE = sum(abs(yhat_test - ytest))/(size(ytest,1))

a = abs(yhat_test - ytest);
CS = size(find(a<= err_level))/size(a) * 100
```

Output

```
MAE =

    7.7044


CS =

   41.2353
```



Workspace

Name	Value
a	502x1 double
absTestErr	0
CS	41.2353
cs_number	0
database_path	'./data_age.mat'
err_level	5
MAE	7.7044
nTest	502
nTrain	500
result_path	'./results/'
teData	1x1 struct
trData	1x1 struct
w_lr	201x1 double

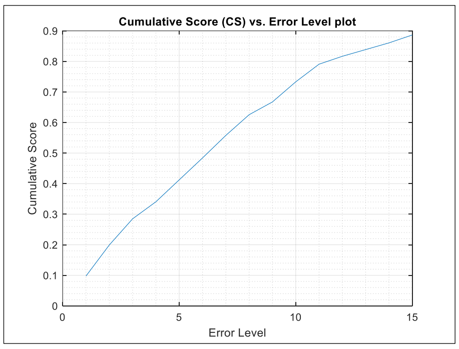
Section 4

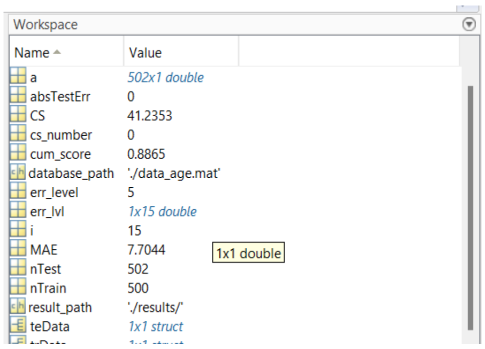
This question generates a plot of CS against the cumulative error level by looping the calculation of CS multiple times. The code for this section can be found in the **Code** section, which is provided below. It can be seen that the output for CS(5) is correct from Question 4. As the levels rise, the cumulative score approaches one, as the margin of error acceptance rises, increasing #e≤j. The metric will reach 1 at an error level of 69, because all samples are counted by #e≤j, and the error level is equivalent to the maximum age value that the samples can achieve. If the error level is set to 69, ground truth of 69 and prediction of 0 would be counted towards #e≤j.

Code

```
%% generate a cumulative score (CS) vs. error level plot by varying the
error level from 1 to 15. The plot should look at the one in the Week6
lecture slides

values = [];
err_lvl = [];
for i = 1:15
    cum_score = size(find(a<= i))/size(a);
    values = [values, cum_score];
    err_lvl = [err_lvl, i];
end

% plot CS vs Error Level
figure(1)
plot(err_lvl, values)
xlabel('Error Level')
ylabel('Cumulative Score')
title('Cumulative Score (CS) vs. Error Level plot')
grid on
grid minor
```

Output





Name	Value
a	502x1 double
absTestErr	0
CS	41.2353
cs_number	0
cum_score	0.8865
database_path	'./data_age.mat'
err_level	5
err_lvl	1x15 double
i	15
MAE	7.7044
nTest	502
nTrain	500
result_path	'./results/'
teData	1x1 struct

Section 5:

This question will use MATLAB's built-in functions to implement the Partial Least Squares regression model and the Regression Tree model, and will output MAE and CS with a level of 5. PLS models can be constructed using MATLAB's *plsregress(...)* function.

According to the documentation, this method employs the SIMPLS algorithm, which first standardises the AAM features and age labels before decomposing the feature and label spaces using SVD. First, the *plsregress(...)* function as follows – '`[XL,YL,XS,YS,BETA] = plsregress(xtrain, ytrain, i);`' was used with 201 components (the maximum allowed given the data's original dimensionality) to determine how much variance is explained by each component. The bias parameter is also added in the code listing above by using the ones method.

When the predictions are continuous variables, regression trees are used. Many metrics, such as Gini impurity or information gain, can be used to classify the dataset. The *fitrtree(...)* is a built-in function in MATLAB that automatically constructs and optimises regression trees using training data and training labels (AAM features and corresponding age labels). The function has been implemented as follows – ‘`reg_tree = fitrtree(xtrain, ytrain);`’

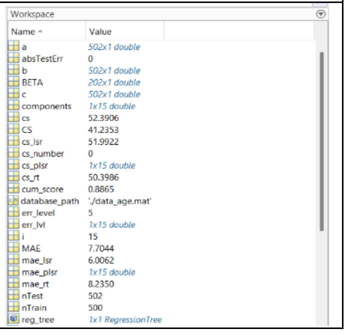
Code

```
%% Compute the MAE and CS value (with cumulative error level of 5) for both
partial least square regression and the regression tree model by using the
Matlab built in functions.

%partial least square regression
mae_plsr = [];
cs_plsr = [];

components = [];
for i = 1:15
    [XL,YL,XS,YS,BETA] = plsregress(xtrain, ytrain, i);
    yhat_test_plsr = [ones(size(xtest,1),1),xtest]*BETA;
    mae_plsr = [mae_plsr, sum(abs(yhat_test_plsr -
ytest))/(size(ytest,1))];

    b = abs(yhat_test_plsr - ytest);
    cs = size(find(b<= err_level))/size(b) * 100;
    cs_plsr = [cs_plsr, cs];

    components = [components, i];
end

%PLOT MAE plsr
figure(2)
plot(components, mae_plsr)
xlabel('Num of components')
ylabel('MAE PLSR')
grid on
grid minor

%PLOT CS plsr
figure(3)
plot(components, cs_plsr)
xlabel('Num of components')
ylabel('CS PLSR')
grid on
grid minor

% best MAE from ncomps
mae_lsr = min(mae_plsr)
% corresponding CS
cs_lsr = cs_plsr(find(mae_plsr == mae_lsr))

% Regression tree model

mae_rt = [];
cs_rt = [];

reg_tree = fitrtree(xtrain, ytrain);
yhat_test_reg = predict(reg_tree, xtest);
mae_rt = [sum(abs(yhat_test_reg - ytest))/(size(ytest,1))]
```

```
c = abs(yhat_test_reg - ytest);
cs_rt = size(find(c<= err_level))/size(c) * 100
```

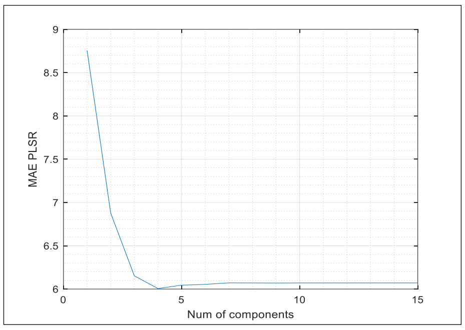
Outputs

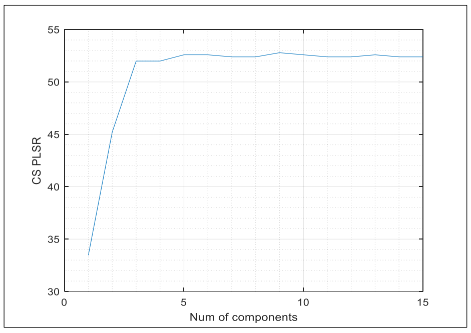
Partial least squares	*Regression tree*	*Workspace*

`mae_lsr =`
 	
`6.0062`
 	
`cs_lsr =`
 	
`51.9922`
	`mae_rt =`
	
	`8.2350`
	
	`cs_rt =`
	
	`50.3986`
		

Workspace

Name	Value
a	502x1 double
absTestErr	0
b	502x1 double
BETA	202x1 double
c	502x1 double
components	1x15 double
cs	52.3906
CS	41.2353
cs_lsr	51.9922
cs_number	0
cs_plsr	1x15 double
cs_rt	50.3986
cum_score	0.8865
database_path	'./data_age.mat'
err_level	5
err_lvl	1x15 double
i	15
MAE	7.7044
mae_lsr	6.0062
mae_plsr	1x15 double
mae_rt	8.2350
nTest	502
nTrain	500
reg_tree	1x1 RegressionTree





Section 6

Support Vector Regression (SVR) and Support Vector Machines (SVM) are very similar (SVM). The goal of SVMs is to find the best hyperplane that separates all training samples while providing the greatest separation between class boundaries. This is accomplished by minimising the regression model's weighting coefficients, which maximises the margin between the ground truth and mode predictions within an error defined by epsilon. The constraints and minimization are as follows:

$$\min \frac{1}{2} ||w||^2$$

$$|\boldsymbol{Y} - \boldsymbol{X}\,\boldsymbol{\beta}| \leq \varepsilon$$

<u>Conclusion from the table *'T':*</u>

A regression problem is age estimation. These models estimate an unknown individual's age based on AAM features of human faces. It is extremely difficult to determine an individual's exact age. MAE can be used to calculate the difference in age between an individual and the regressed age. However, MAE is insufficient in and of itself because it only reports the mean prediction error across all samples. The majority of the faces have distinct facial features, poses, expressions, or lighting conditions that may influence age estimation.

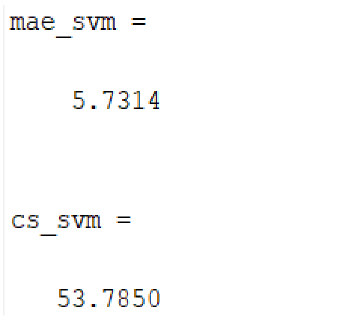
By stating, given a threshold on the estimation error that an algorithm can make, what is the percentage of samples that are within the error threshold that is deemed acceptable, cumulative score becomes a complementary metric to MAE (internally using MAE within itself). The higher the cumulative score at lower error levels, the less prediction error is tolerated, and the percentage of samples that are within acceptable error limits is calculated. In addition to MAE, cumulative score can be used as a metric to evaluate model performance in estimating an individual's age.

Code

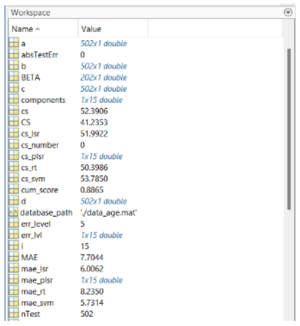
```
%% Compute the MAE and CS value (with cumulative error level of 5) for
Support Vector Regression by using LIBSVM toolbox

mae_svm = [];
cs_svm = [];

svm = fitrsvm(xtrain, ytrain);
yhat_test_svm = predict(svm, xtest);
mae_svm = [sum(abs(yhat_test_svm - ytest))/(size(ytest,1))]

d = abs(yhat_test_svm - ytest);
cs_svm = size(find(d<= err_level))/size(d) * 100

% table displaying MAE and CS values and Algorithms

values = {'Linear Regression' MAE CS; 'PLS Regression' mae_lsr cs_lsr;
'Regression Tree' mae_rt cs_rt; 'Support Vector Regression' mae_svm
cs_svm};
T = cell2table(values, 'VariableNames',{'Algorithm' 'MAE' 'CS'})
```

Outputs

Support Vector Regression results

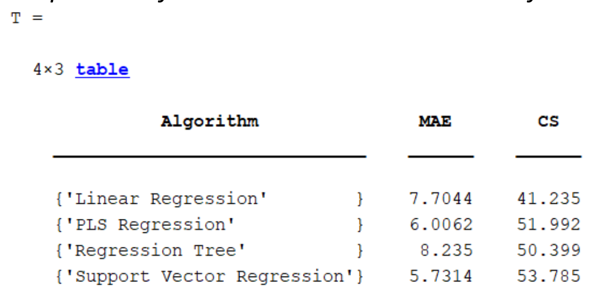


```
mae_svm =

    5.7314


cs_svm =

   53.7850
```



```
Workspace
Name ▲          Value
a               502x1 double
absTestErr      0
b               502x1 double
BETA            202x1 double
c               502x1 double
components      1x15 double
cs              52.3906
CS              41.2353
cs_lsr          51.9922
cs_number       0
cs_plsr         1x15 double
cs_rt           50.3986
cs_svm          53.7850
cum_score       0.8865
d               502x1 double
database_path   './data_age.mat'
err_level       5
err_lvl         1x15 double
i               15
MAE             7.7044
mae_lsr         6.0062
mae_plsr        1x15 double
mae_rt          8.2350
mae_svm         5.7314
nTest           502
```

Comparison of MAE and Cumulative Scores of MLR, PLS-R, Regression Tree and SVR Models:



```
T =

  4×3 table

             Algorithm                MAE       CS
    ______________________________    ______    ______

    {'Linear Regression'         }    7.7044    41.235
    {'PLS Regression'            }    6.0062    51.992
    {'Regression Tree'           }     8.235    50.399
    {'Support Vector Regression'}    5.7314    53.785
```

References:

- https://en.wikipedia.org/wiki/Linear_regression#Formulation
- https://www.datatechnotes.com/2019/02/regression-model-accuracy-mae-mse-rmse.html
- https://en.wikipedia.org/wiki/Decision_tree_learning#Further_reading
- https://www.csie.ntu.edu.tw/~cjlin/libsvm/
- https://www.mathworks.com/help/stats/partial-least-squares.html