Extended Face Classification

Echo Liu

May 28, 2020

https://github.com/EchoRLiu/EigenFacesExtansion

Abstract

Using datasets with images and labels, we hope to train the computer to recognize objects of interest and classify them to give us certain information. This process comes down to build a statistical model that is capable to extract distinguishing features from the input digital images and maps them to certain labels correctly. This can be done with clustering and classification methods. Another important perspective in feature extraction is dimension reduction, which is essentially a low-dimension reconstruction of the data without noise, and we are going to see how useful Singular Value Decomposition (SVD) is for this task. In this project, using the Extended Yale Face Database B [2], we illustrate these ideas by using unsupervised and supervised learning, such as Naive Bayesian, Linear Discrimination, SVM, along with SVD to build models to conduct certain classification tasks. Each model is evaluated based on how accurately it labels the test image dataset.

1 Introduction and Overview

Here we describe how unsupervised and supervised learning can be used to construct clustering or classification models, and how SVD can be used along the models for dimension reduction for better feature extraction and efficient algorithms.

1.1 Dataset Description

The Extended Yale Face Database B includes originally uncropped version (the faces are not centered), which contains images of human faces from 14 individuals with various facial expressions and accessories, and cropped version, which contains images of human faces from 38 individuals with various lighting settings.

1.2 General Approach

First, since features usually encode knowledge on objects, which is difficult to learn from a raw finite set of input pixels [3], we first use Haar wavelet on all images to better present the encoded information. Through SVD analysis on these two versions, we can show that cropping or centering is necessary when conducting object detection or classification task, and we can also gain more information on how how many POD basis we need to reconstruct the images and how the basis are like.

Further, we train unsupervised and supervised models on the cropped dataset to first classify individuals and then classify female and male, purely based on facial features. For each loop of cross-validation, first, we shuffle the dataset and divide them into training and testing sets, with corresponding labels; then, K-means clsuter, Gaussian Mixture Model, KNN, Naive Bayesian, Linear Discrimination, Support Vector Machine, are used to build classification models and make prediction on test dataset; we then compare the prediction result and true test data labels to check the accuracy of the result.

2 Theoretical Background

As we learned from our textbook [1], wavelet are used for orthogonal bases with ability to overcome uncertainty principle, which is a core idea of multi-resolution enabling both time and frequency information. One of the simplest version of wavelet is Haar wavelet, defined as

$$\Psi(t) = \begin{cases} 1 & 0 \le t \le 1/2 \\ -1 & 1/2 \le t \le 1 \\ 0 & otherwise \end{cases} \tag{1}$$

similar to sound frequency information, Haar wavelet can also produce important and unique features of an image.

A way to decompose data is

$$A = \hat{U}\hat{\Sigma}V^* \tag{2}$$

where $\hat{U}$ have orthonormal columns, V is an unitary matrix and $\hat{\Sigma}$ is a diagonal matrix. This can be used for any matrix A. The basic idea of SVD is to "stretch", "rotate" and "compress" the data, to make the variables independent. The idea of PCA here is that in this way, we are able to rank the variables based on how much variance they have, thus reducing the variables based on how important they are to the data.

Unsupervised and supervised learning are two important ways to classify data. For the first one, during training, no labels are given, we are mainly looking for cluster in data and look for the closest groups to let the dataset decide. For supervised training, labels are needed, and several methods such as Naive Bayesian, Linear Discrimination, SVM can be used to better classify data. One classic and important unsupervised learning method is K-means clustering. Depending on the clusters specified, certain random initial points are selected and separate groups based on distance towards those initial points; center mass locations of those groups are then chosen; these two steps are repeated to find the optimal way to separate clusters. A similar method to K-means is K-Nearest Neighbours (KNN), which is actually supervised learning. With the training data with labels, the algorithm looks for k points near every test data point, and label it according to those neighbours, which makes this method is capable of drawing non-linear separation line between groups. Another unsupervised learning method is Gaussian Mixture models, which looks at the distributions of clusters data. Using equation 3, where $P(x|\theta)$ is the overall probability, based on each clusters' distribution, we can try to maximize the overall probability.

$$P(x|\theta) = \sum_{k=1}^{\mathcal{K}} \alpha_k P_k(x, \theta_k) \tag{3}$$

Naive Bayes (NB) uses the idea of conditional probability and Bayes rules. For instance, if we have two classes 0 and 1,

$$\frac{p(0|x)}{p(1|x)} = \frac{p(x|0)p(0)}{p(x|1)p(1)}. \tag{4}$$

now all the probability on the right hand can be calculated, which tells us which class the data point belongs to with a larger probability. Linear Discrimination Analysis (LDA) finds the optimal way to project the data onto a line to maximize the distance between the projected data from different classes. Support Vector Machine (SVM) is a method that tries to find the optimal line to separate the data, where the curves along each data class's boundary points are optimized first, and another set of lines are optimized to maximize the distance between every two curves we found before. An advantage of SVM is that it can produce non-linear curve separating the data. Classification Trees separate data by proposing many questions and based on the yes or no answers.

An important thing during training and testing is cross-validation. This allows us to see different combination of training and testing dataset and see how well the algorithm behaves generally.

3 Algorithm Implementation and Development

1. Yale Uncropped Face data set and cropped data are first loaded into `uncropped faces` and `cropped faces`

2. Haar wavelet is then applied to all images to extract edge features using Algorithm 1 and store the result into `uncropped wave` and `cropped wave`

```
Algorithm 1: Haar Wavelet
  Obtain the matrix where each column represent one image, the size of each image as m, n
  Obatin the size of the matrix as p, q
  Get the column size of colormap(gray) as nbcol
  for $j = 1 : q$ do
    Obtain column j of the matrix, reashape them to m by n, change the datatype to double and store the result in X.
    apply dwt2(X, 'haar') and store the results as cA, cH, cV, cD
    apply wcodemat(,nbcol) to both cH, cV and store results as cod cH1, cod cV1
    Store cod cH1 + cod cV1 as cod edge
    reshape cod edge into a column vector and store it into the column j of dcData
  end for
```

3. SVD is then applied to `uncropped wave` and `cropped wave` and obtain results `U, S, V`
4. visualize `U, S, V`
5. `U, S, V` are then analyzed in the next section.
6. use Algorithm 2 on `V` with cross validation to obtain average rate of correct classification.

```
Algorithm 2: Training and Testing
  Initialize avg correct as 0, cross, cross validation times, as 300.
  for $j = 1 : cross$ do
    Within each class, shuffle data and separate them into train and test
    Create labels for train as ctrain and for test as ctest
    Choose each of the following methods.
    For Naive Bayes method, apply nb = fitcnb(train, ctrain) and obtain pre using nb.predict(test)
    For Linear Discrimination method, obtain pre using classify(test, train, ctrain)
    For SVM method, apply svm = fitcecoc(train, ctrain) and obtain pre using predict(svm, test)
    For Quadratic Discrimination method, apply da = ClassificationDiscriminant.fit(train, ctrain) and obtain pre using da.predict(test)
    For KNN method, apply knn = ClassificationKNN.fit(train, ctrain), define the number of neighbours we want using knn.NumNeighbors = 2 (we use 2 here), and obtain pre using knn.predict(test)
    For Tree Classification method, apply tree = ClassificationTree.fit(train, ctrain) and obtain pre using tree.predict(tree)
    Visualize pre using bar(pre)
    calculate correct by summing the result ($ctest == pre$) and update avg correct = avg correct + correct / total num of testing data
  end for
  Obtain the true avg correct = avg correct/cross
```

modes	10	20	40	60	80	100	1000
NB	19.98%	54.86%	76.51%	82.13%	83.93%	85.62%	-
LD	22.31%	60.95%	81.57%	86.76%	89.86%	91.50%	97.58%
SVM	15.21%	42.11%	62.30%	68.04%	74.41%	80.01%	-

Table 1: Individual classification accuracy rate

4 Computational Results

4.1 SVD Analysis

After SVD, the first four modes of U for cropped images and uncropped images are visualized in figure 1. Every column of U represents a mode, which can be understood as a feature of the data. For cropped images, the first four modes successfully capture the important features of human faces, which means it's possible for us to reconstruct the data with reasonable amount of modes. However, as we can see in the right side of figure 1, the first four modes do not contain any features of human faces, but rather human shape, which means we cannot use those modes to reconstruct the data. This observation is important as it leads researchers to show we have to have aligned boxes to capture features to avoid noises caused by nonalignment, and as in [4], Viola and Jones figured out an algorithm to efficiently do so.

The diagonal of matrix S represents the variance of the data captured by each mode, and the quicker the variance drops along the modes, we can use fewer modes, thus achieving lower dimensions. Figure 2 shows the variance captured by modes for both cropped and uncropped images. The variance drops more quickly for cropped images than uncropped images, which confirms what we see in the first fours modes of U. For the cropped images, the first 10 modes capture a lot of the variance, however, as we would see later, we need more than 10 modes to reconstruct the data, and using 90% threshold for capturing the energy, we need 1666 ranks for cropped images. As we established earlier, since there is too much noise in the uncropped images, it would be very difficult to reconstruct uncropped images, thus the rank for uncropped images is not very meaningful here. But the calculation is also done to show that, to capture 90% energy, 126 ranks would be needed.

The matrix V contains information on how each mode of U should be adjusted, as shown in figure 3. We can also use V as a way to observe clustering behaviour. As in figure 4, in three-dimension, all 38 individuals of cropped images are clustered together, same for all 15 individuals of uncropped images in figure 5. We can already see it would be a difficult task to classify the individuals since they are all clustered together. However, going to higher dimensions, we might be able to classify them. The cluster behaviour is also demonstrated for female and male classification in figure 6, which seems like an easier case to classify the two compared to individual classification.

4.2 Test 1: Face Classification

With 300 cross-validation, table 1 show the accuracy of 38 individuals classification task, using supervised learning Naive Bayes, Linear Discrimination and SVM, with different numbers of features used. As the number of features we use increases, the accuracy of classification increases as well, and we can see with 100 features, we can already reach around 90% accuracy for all three methods. But among the three, we can see LD gives the best result, and it's also the most efficient algorithm. Because of the fast algorithm, 1000 features are also used to see how well it can behave, and it can reach a very high accuracy 97.58%. The other two methods, especially SVM, are very slow and is not as good as LD. Thus, for a successful individual classification method, we should choose Linear Discrimination method with around 1000 features or less, if needing lower-dimension to deal with.

4.3 Test 2: Gender Classification

With 300 cross-validation, table 2 show the accuracy of female and male classification task, using supervised learning Naive Bayes, Linear Discrimination and SVM, with different numbers of features used. The result is different from the result of test 1 and is very interesting. First, for NB method, the best accuracy we get

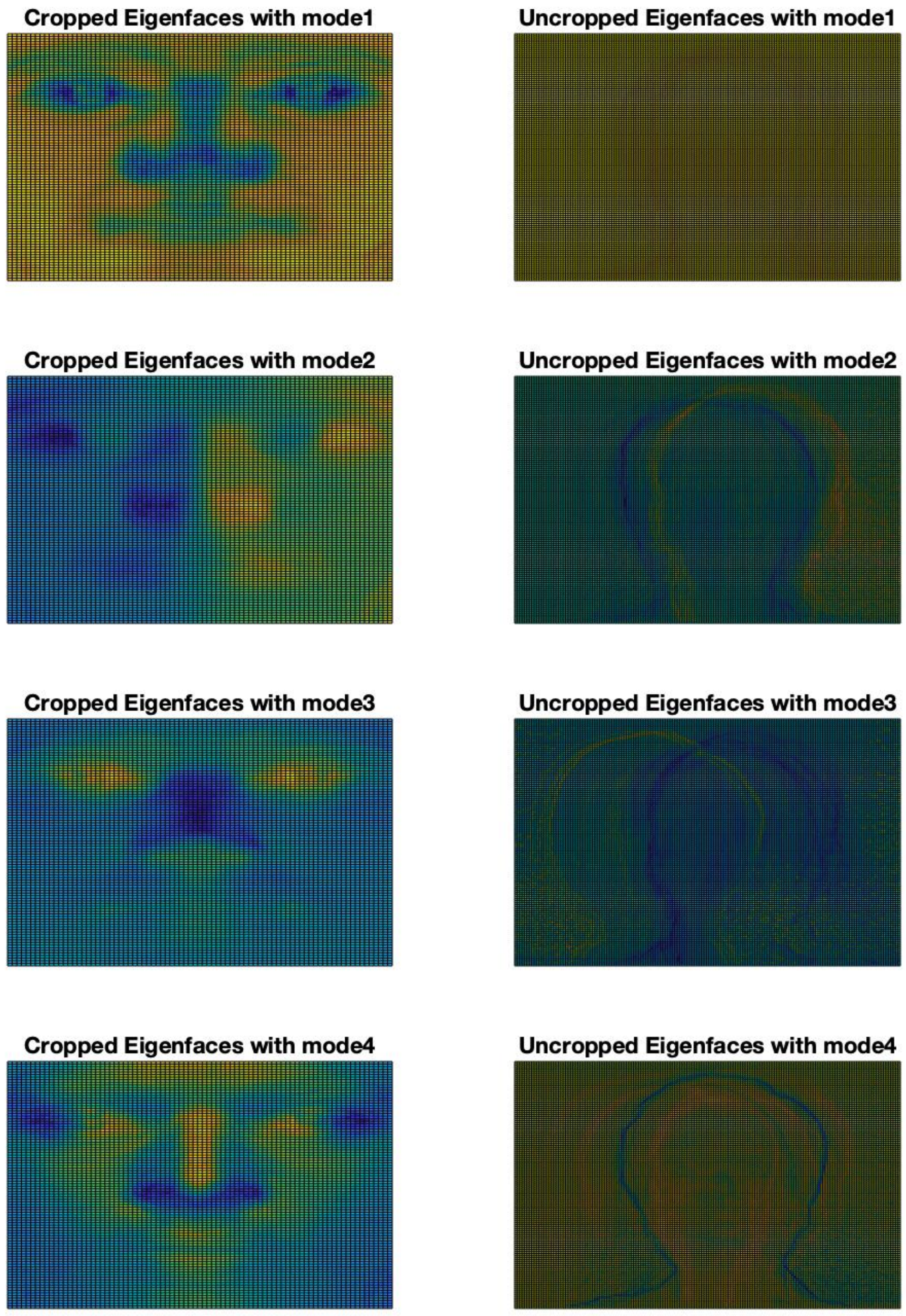

Figure 1: Eigenfaces

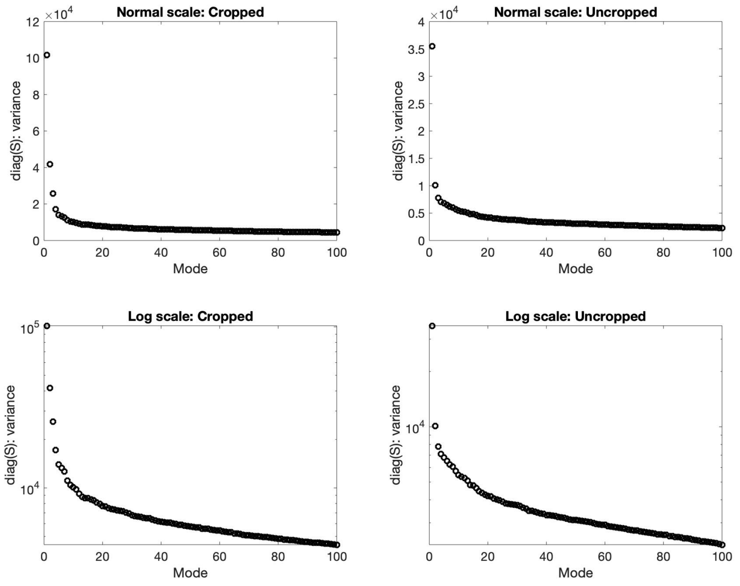

Figure 2: Variance captured by each mode

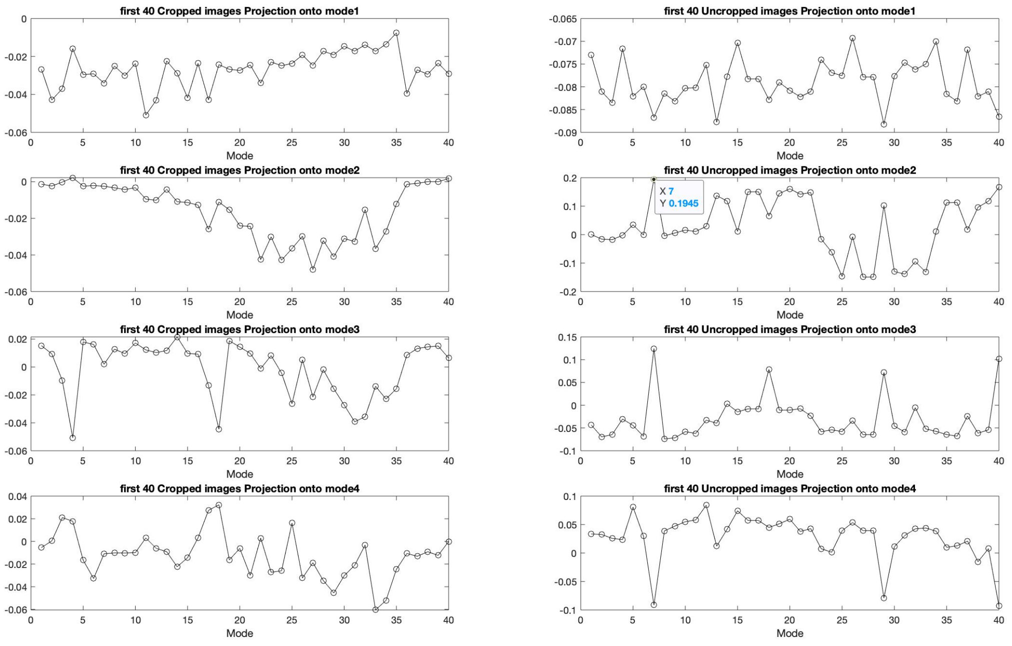

Figure 3: Mode parameters

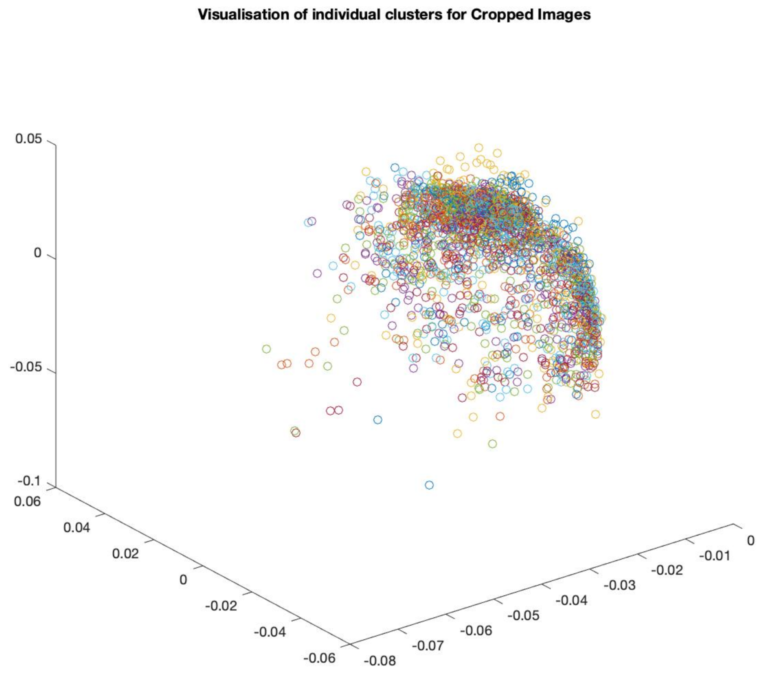

Figure 4: Individual Clusters for Cropped Images

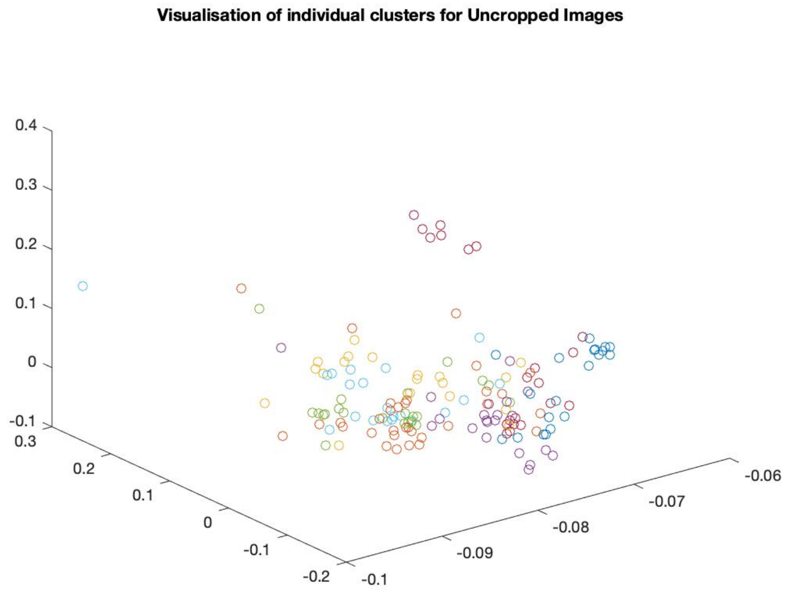

Figure 5: Individual Clusters for Uncropped Images

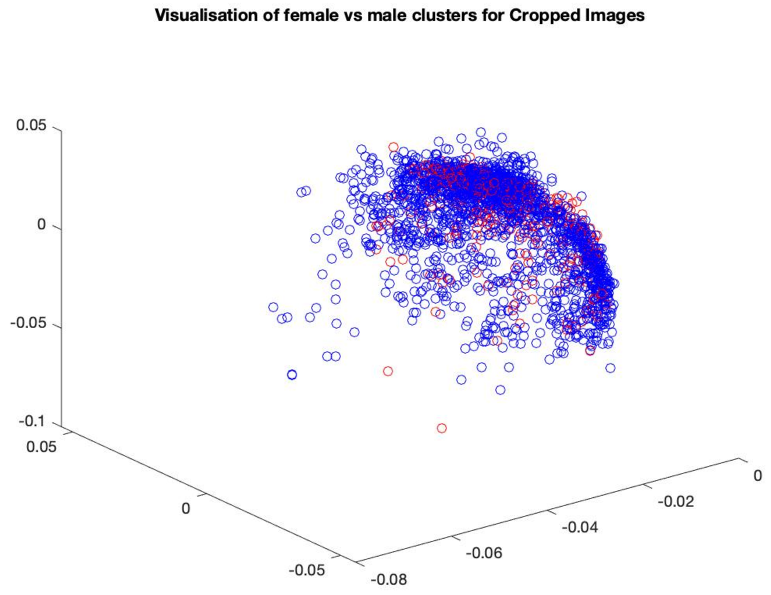

Figure 6: Female and Male Clusters for Cropped Images

modes	10	20	40	60	80	100
NB	82.61%	89.47%	87.90%	85.31%	84.40%	82.15%
LD	64.29%	84.25%	89.35%	92.25%	95.19%	95.48%
SVM	84.21%	84.21%	84.31%	84.37%	84.63%	84.60%

Table 2: Gender classification accuracy rate

is with 20 features, while for LD and SVM, as the features increase, the accuracy increases as well. LD is the fastest and best method for this task as well. Since this is an easier classification task, only 100 features can already achive 95.48% accuracy with LD. Thus, to have a successful gender classification algorithm, we can use Linear classification with around 100 features.

4.4 Test 3: Unsupervised learning

With 300 cross-validation, table 3 show the accuracy of 38 individuals classification task and gender classification task, using unsupervised learning Gaussian Mixture method with different numbers of features used (here low number of features is used for individual classification task because of the limitation of the function). The unsupervised learning has very low accuracy for individual classification task, and it might be because 38 clusters are very hard to distinguish since as we can see in figure 4, they are all very tightly clustered together. The gender classification task is significantly better since the number of clusters the algorithm needs to identify is reduced. It's interesting to see that the best result happens with only 10 features for both of the tasks. The accuracy for gender classification is generally higher than individual classification since it's easier. However, we can see the unsupervised learning method generally does not behave as well as with supervised learning methods. Thus, for these specific two tasks, we probably should not use unsupervised learning method GM.

5 Summary and Conclusions

Through this project, we can see again how useful SVD is. We can also see how wavelet is used to obtain features on images and how supervised training and unsupervised learning can be used to classify data.

modes	5	10	20	40	80	200	500
Face Classification	2.59%	2.58%	-	-	-	-	-
Gender Classification	49.19%	52.06%	51.05%	51.58%	49.79%	50.02%	51.42%

Table 3: classification accuracy rate using GM

References

[1] Steven L. Brunton and J. Nathan Kutz. *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge University Press, 2019.

[2] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman. "From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose". In: *IEEE Trans. Pattern Anal. Mach. Intelligence* 23.6 (2001), pp. 643–660.

[3] Rainer Lienhart and Jochen Maydt. "An Extended Set of Haar-like Features for Rapid Object Detection". In: vol. 1. Feb. 2002, pp. I–900. DOI: 10.1109/ICIP.2002.1038171.

[4] P. Viola and M. Jones. "Rapid object detection using a boosted cascade of simple features". In: (2001).

Appendix A MATLAB Functions

- `dwt2(X, 'haar')` computes the single-level 2-D discrete wavelet transform (DWT) of the input data X using the haar wavelet.
- `wcodemat(X,nbcol)` rescales the input X as integers in the range [1, nbcol].
- `randperm(n)` returns shuffled number from 1 to n.

Appendix B MATLAB Code

```
close all; clear all; clc

% Load cropped faces dataset.
m1=192; n1=168; % Resolution of images.
nw1=96*84; % SVD resolutions.
cropped_faces=zeros(32256, 2432);
% we got 2432 images in croppedYale folder, we are missing yaleB14 folder.

file='/Users/yuhongliu/Downloads/CroppedYale/yaleB';
pos=0;

for i=1:39
    if i<=9
        dirt=append(file,string(0),string(i));
    else
        dirt=append(file,string(i));
    end
    file_ls=dir(dirt);
    NF = length(file_ls);
    for j = 3:NF
        pos=pos+1;
        cropped_faces(:, pos) = reshape(imread(fullfile(dirt, file_ls(j).name)), 32256,1);
    end
end
```

```
% We have 64 images from 38 people.

%%

% Load uncropped faces dataset.
m2=243; n2=320; nw2=122*160;% remember the shape of images.
uncropped_faces=zeros(77760,165); % we have 15 subjects with 11 different faces.

file='/Users/yuhongliu/Downloads/yalefaces_uncropped/yalefaces/subject';
subtitle=[".centerlight",".glasses",".happy",".leftlight",".noglasses",".normal",".rightlig

for i=1:15
    for j=1:11
        if i<=9
            file_title=append(file,string(0),string(i),subtitle(j));
        else
            file_title=append(file,string(i),subtitle(j));
        end
        pos=11*(i-1)+j;
        uncropped_faces(:,pos)=reshape(imread(file_title), 77760, 1);
    end
end

%%

% Get the wavelet representations.
cropped_faces_wave = dc_wavelet(cropped_faces, m1, n1, nw1);
uncropped_faces_wave = dc_wavelet(uncropped_faces, m2, n2, nw2);

feature = 60;

%%

[U1,S1,V1]=svd(cropped_faces_wave,0);
U1=U1(:,1:feature);
[U2,S2,V2]=svd(uncropped_faces_wave,0);
U2=U2(:,1:feature);

%%

% The first four POD modes.
figure(1);
for j = 1:4
    % Cropped faces.
    subplot(4,2,2*j-1);
    ut1=reshape(U1(:,j), 96, 84);
    ut2=ut1(96:-1:1,:);
    pcolor(ut2);
    set(gca,'Xtick',[],'Ytick',[]);
    title(['Cropped Eigenfaces with mode' num2str(j)]);
    % Uncropped faces.
    subplot(4,2,2*j);
    ut1=reshape(U2(:,j), 122, 160);
    ut2=ut1(122:-1:1,:);
    pcolor(ut2);
    set(gca,'Xtick',[],'Ytick',[]);
    title(['Uncropped Eigenfaces with mode' num2str(j)]);
end

%%

figure(2);
% Cropped faces.
subplot(2,2,1); % normal scale.
plot(diag(S1)/sum(diag(S1)),'ko','Linewidth',[2]);
set(gca,'Fontsize',[14],'Xlim',[0 100]);
xlabel('Mode'), ylabel('Energy')
title('Normal scale: Cropped')
subplot(2,2,3);
semilogy(diag(S1)/sum(diag(S1)),'ko','Linewidth',[2]); % Log scale.
set(gca,'Fontsize',[14],'Xlim',[0 100]);
xlabel('Mode'), ylabel('Energy')
title('Log scale: Cropped')
% Uncropped faces.
subplot(2,2,2); % normal scale.
plot(diag(S2)/sum(diag(S2)),'ko','Linewidth',[2]);
set(gca,'Fontsize',[14],'Xlim',[0 100]);
xlabel('Mode'), ylabel('Energy')
title('Normal scale: Uncropped')
subplot(2,2,4);
semilogy(diag(S2)/sum(diag(S2)),'ko','Linewidth',[2]); % Log scale.
set(gca,'Fontsize',[14],'Xlim',[0 100]);
xlabel('Mode'), ylabel('Energy')
title('Log scale: Uncropped')

%%

energy1 = diag(S1)/sum(diag(S1));
thresh90 = 0;
for i = 1:2432
    thresh90 = thresh90 + energy1(i);
    if thresh90 > .9
        disp(i) % rank can be 1666.
        break
    end
end

energy2 = diag(S2)/sum(diag(S2));
thresh90 = 0;
for i = 1:165
    thresh90 = thresh90 + energy2(i);
    if thresh90 > .9
        disp(i) % rank can be 126.
        break
    end
end

%%
```

```
% Projection of the first 40 face images onto the first four POD modes.
figure(3);
for j=1:4
    % Cropped faces.
    subplot(4,2,2*j-1);
    plot(1:40,V1(1:40,j),'ko-');
    xlabel('Mode')
    title(['first 40 Cropped images Projection onto mode' num2str(j)])
    % Uncropped faces.
    subplot(4,2,2*j);
    plot(1:40,V2(1:40,j),'ko-');
    xlabel('Mode')
    title(['first 40 Uncropped images Projection onto mode' num2str(j)])
end

% should we use u or v as our projection.

%%

% Inviduals.
figure(4)
for i = 1:38
    plot3(V1((i-1)*64+1:i*64,1), V1((i-1)*64+1:i*64, 2), V1((i-1)*64+1:i*64, 3), 'o'),
    hold on
    title('Visualisation of individual clusters for Cropped Images');
end

figure(5)
for i = 1:15
    plot3(V2((i-1)*11+1:i*11,1), V2((i-1)*11+1:i*11, 2), V2((i-1)*11+1:i*11, 3),'o'),
    hold on
    title('Visualisation of individual clusters for Uncropped Images');
end

%%

fm = [5 26 27 31 33 36];
% Female vs Male.
figure(6)
for i = 1:38
    if ismember(i, fm)
        plot3(V1((i-1)*64+1:i*64,1), V1((i-1)*64+1:i*64, 2), V1((i-1)*64+1:i*64, 3), 'ro')
        hold on
    else
        plot3(V1((i-1)*64+1:i*64,1), V1((i-1)*64+1:i*64, 2), V1((i-1)*64+1:i*64, 3), 'bo')
        hold on
    end
    title('Visualisation of female vs male clusters for Cropped Images');
end

%% Test 1. Classify individuals.

avg_correct=0;
```

```
cross=300; % Cross-validation is important!!
ctest = [];
ctrain = [];
for j = 1:38
    ctrain=[ctrain; j*ones(57,1)];
    ctest=[ctest; j*ones(7,1)];
end

% un = 1; feature = 20; % GM method.
un = 0; feature=40; % Other supervised methods.
if un == 1

    i = 1;
    while i <= cross

        % figure(4+i)

        % Seperate training and testing dataset.
        % use around 90% of images for training.

        qs = zeros(64, 38);
        for j = 1:38
            qs(:,j) = randperm(64);
        end

        xtrain = zeros(2166,feature);
        xtest = zeros(266,feature);
        for k = 1:38
            individual = V1((64*(k-1)+1):(64*k),1:feature);
            q = qs(:,k);
            xtrain((57*(k-1)+1):(57*k),:) = individual(q(1:57),:);
            xtest((7*(k-1)+1):(7*k),:) = individual(q(58:end),:);
        end

        % GM has very low accuracy. 2.58%.
        try
            gm=fitgmdist(xtrain,38);
            pre=cluster(gm, xtest);
            correct=sum((pre == ctest));
            avg_correct = avg_correct + correct/266;
            disp(i)
            i = i+1;
        catch
            i = i;
        end
    end

else

    for i = 1:cross

        % figure(4+i)

        % Seperate training and testing dataset.
        % use around 90% of images for training.

        qs = zeros(64, 38);
        for j = 1:38
            qs(:,j) = randperm(64);
        end
        xtrain = zeros(2166,feature);
        xtest = zeros(266,feature);
        for k = 1:38
            individual = V1((64*(k-1)+1):(64*k),1:feature);
            q = qs(:,k);
            xtrain((57*(k-1)+1):(57*k),:) = individual(q(1:57),:);
            xtest((7*(k-1)+1):(7*k),:) = individual(q(58:end),:);
        end

    %      % Naive Bayesian method. 82.11%
        nb=fitcnb(xtrain, ctrain);
        pre=nb.predict(xtest);

        % Linear Discrimination behave the best with spectrogram.
        % 86.48%.
        % pre=classify(xtest, xtrain, ctrain);

        % SVM. 68.16%
%          svm=fitcecoc(xtrain,ctrain);
%          pre=predict(svm, xtest);

        % bar(pre);
        correct=sum((pre == ctest));
        avg_correct = avg_correct + correct/266;
    end
end

disp(avg_correct/cross);

%% Test 2. Gender Classification.

avg_correct=0;
cross=300; % Cross-validation is important!!

fm = [5 26 27 31 33 36]; % [5 27-1 28-1 32-1 34-1 37-1];
% Missing folder 14. Hence the -1.
% Label Male as 1, female as 2.
ctest = ones(266,1);
ctrain = ones(2166,1);
for j = 1:6
    ctrain((57*(fm(j)-1)+1):(57*fm(j)), 1) = 2;
    ctest((7*(fm(j)-1)+1):(7*fm(j)), 1)= 2;
end
feature = 100;

un = 1; % GM method.
%un = 0; % Other supervised methods.
```

```
if un == 1

    i = 1;
    while i <= cross

        % figure(4+i)

        % Seperate training and testing dataset.
        % use around 90% of images for training.
        qs = zeros(64, 38);
        for j = 1:38
            qs(:,j) = randperm(64);
        end
        xtrain = zeros(2166,feature);
        xtest = zeros(266,feature);
        for k = 1:38
            individual = V1((64*(k-1)+1):(64*k),1:feature);
            q = qs(:,k);
            xtrain((57*(k-1)+1):(57*k),:) = individual(q(1:57),:);
            xtest((7*(k-1)+1):(7*k),:) = individual(q(58:end),:);
        end

        % GM has low accuracy. 51.62%.
        try
            gm=fitgmdist(xtrain,2); % Female vs Male.
            pre=cluster(gm, xtest);
            correct=sum((pre == ctest));
            avg_correct = avg_correct + correct/266;
            disp(i)
            i = i+1;
        catch
            i = i;
        end
    end

else

    for i = 1:cross

        % figure(4+i)

        % Seperate training and testing dataset.
        % use around 90% of images for training.

        qs = zeros(64, 38);
        for j = 1:38
            qs(:,j) = randperm(64);
        end

        xtrain = zeros(2166,feature);
        xtest = zeros(266,feature);
        for k = 1:38
            individual = V1((64*(k-1)+1):(64*k),1:feature);
            q = qs(:,k);
            xtrain((57*(k-1)+1):(57*k),:) = individual(q(1:57),:);
            xtest((7*(k-1)+1):(7*k),:) = individual(q(58:end),:);
        end

        % KNN.
        [ind, D] = knnsearch(xtrain, xtest, 'k', 2);

    %     % Naive Bayesian method. 85.39%
%           nb=fitcnb(xtrain, ctrain);
%           pre=nb.predict(xtest);

        % Linear Discrimination behave the best with spectrogram.
        % 92.27%.
        pre=classify(xtest, xtrain, ctrain);

        % SVM. % 84.36%
        svm=fitcecoc(xtrain,ctrain);
        pre=predict(svm, xtest);

        % bar(pre);
        correct=sum((pre == ctest));
        avg_correct = avg_correct + correct/266;
    end
end

disp(avg_correct/cross);

%%

function dcData = dc_wavelet(dcfile, m, n, nw)
    [p,q]=size(dcfile);
    nbcol = size(colormap(gray),1);
    dcData=zeros(nw,q);

    for i = 1:q
        X=double(reshape(dcfile(:,i),m,n));
        [cA,cH,cV,cD]=dwt2(X,'haar');
        cod_cH1 = wcodemat(cH,nbcol);
        cod_cV1 = wcodemat(cV,nbcol);
        cod_edge=cod_cH1+cod_cV1;
        dcData(:,i)=reshape(cod_edge,nw,1);
    end
end
```