Amath563: Classification for Reduced Rank Data

Helena Liu, May 25th, 2020

Abstract: This work applies various classical clustering and classification methods to the Yale Face Image dataset. SVD is performed to reduce the rank of the images before feeding them to the classifier. The results demonstrate the advantage of using rank reduced data for classification and suggest a few avenues of future work.

I. Introduction and Overview

Clustering and classification on image data for face identification has important applications in marketing and biometrics. One key challenge of these tasks is overfitting, where classification methods trained to minimize the error on high dimensional noisy data fail to generalize. This study performs classification on rank reduced data obtained using singular value decomposition (SVD) for the Yale Face Image dataset and demonstrates how controlling the rank of the data can affect the classification outcome.

II. Theoretical Background

A) Singular value decomposition (SVD)

Every matrix $A \in \mathbb{R}^{mxn}$ has a singular value decomposition $A = U\Sigma V^*$, where $U \in \mathbb{R}^{mxm}$ is unitary, $V \in \mathbb{R}^{mxn}$ is unitary and $\Sigma \in \mathbb{R}^{mxn}$ is diagonal [1]. Let each column in A represent a datapoint, so that n denotes the number of points and m denotes the dimension of each datapoint, then the columns of U consists of an orthonormal basis expansion of the data matrix A. The j^{th} column of U corresponds to the j^{th} principle component (PC) or mode of A, with the strength of that PC given by singular value σ_j, the j^{th} diagonal element in Σ, and the component (i.e. the projection) of the data along that PC is the j^{th} column of V. The indices of singular values are sorted based on their magnitude, with σ_1 denoting the strength of the most dominant mode. Therefore, the first r columns of U correspond to the top r dominating modes.

An important application of SVD is to obtain rank reduced approximation to the data. A_r, a rank r approximation of A from the first r modes, can be obtained by

$$A_r = \sum_{i=1}^{r} U_i \sigma_i V_i^*, \text{ (1)}$$

where U_i and V_i are the i^{th} columns of U and V, respectively. This has an important consequence in classification, as it can alleviate overfitting by removing noisy components of the data through dimensionality reduction. For that reason, this work performs classification on rank reduced data.

B) Classification

Classification tasks involve separating data into P discrete classes when the true class is known. These tasks fall into the broader category of supervised learning, where the true label to each data point is given, and a model is built to best capture the relationship between the data and its label. The process of building a model is called training, where a certain optimization routine is

performed to tune model parameters such that when training data is inputted to the model, the error between the model output and the training labels are minimized (assuming regularization is neglected). This model is then applied to unseen data and evaluated by comparing its predicted label with the true label of these unseen data.

To obtain such a model, different classifiers employ different model architecture and optimization routines. The classifiers used in this work are k^{th} nearest neighbours (KNN), Support Vector Machines (SVM), decision tree, Naïve Bayes and linear discriminant analysis (LDA). KNN identifies the label of a test data based a majority vote from the labels of the k training data with the smallest Euclidean distance to that test data. An advantage of KNN is that it can draw highly irregular and nonlinear boundaries around the data, but at the same time, it is prone to overfitting due to this capability. Naïve Bayes finds the prior and the likelihood of the data for each class and then identify the class with the highest posterior probability. Naïve Bayes is very simple to compute, but its outcome is limited by how accurate the distributions can be computed. Decision trees split the datapoints into successive nodes. Each node is associated with a label or the probability of a label. Decision trees, similar to KNN, can draw highly nonlinear boundaries depending on the depth of the tree used. LDA seeks to find a line of separation between classes that maximizes between class variance and minimizes within class variance. LDA is limited to linear separation of the data, and SVM, the pillar of classification methods, offer a more versatile approach by specifying a kernel that transforms nonlinear data into a linearly separable space. The choice of the kernel for SVM is important to the classification outcome, which will be demonstrated in Section IV.

C) Clustering

Unlike supervised learning, the true label is unknown in unsupervised learning, which seeks to find underlying structures and patterns in data. Clustering is a type of unsupervised learning task. In the absence of correct answers, intuition and domain knowledge would be very helpful for evaluating if the resultant clusters are reasonable. Often, unsupervised learning is applied to obtain the clusters when little is known about the data, and these clusters then provide the labels for supervised learning.

An example of clustering algorithm is K-means, which separates data into K clusters and identify the cluster for each data based on which cluster mean it is the closest to. It iteratively identifies the mean for data in each cluster and then update the cluster label for each datapoint based on which cluster mean it is the closest to using Euclidean distance. Selecting K may involve some background knowledge and trial-and-error explorations.

Another method for clustering is Gaussian Mixture Model (GMM), because in order to use this method, the dimension of each datapoint has to be less than the number of observations. The dataset investigated in this study does not satisfy this requirement.

III. Algorithm Implementation and Development

A) SVD for Face Images

To convert the Yale Face Image dataset into the form of $A \in \mathbb{R}^{mxn}$ described in Section II, each image is reshaped into a column vector. SVD is applied, with the columns of U containing the basis expansion of the face images, columns of V denoting the component of the images on that basis and diagonal elements of Σ denoting the strength of each PC. For classification with rank r data, the first r columns of V, i.e. the projection of A_r in (1) onto the PC space, is used as input data. This is used instead of A_r because LDA requires the input covariance matrix to be invertible, i.e. full rank. Classification is performed on various numbers of PCs used to investigate how the rank of the data affects the classification performance.

To visualize how many PCs are required to explain 95% of the data variance, the r^{th} cumulative relative energy:

$$E_r = \frac{\sum_i^r \sigma_i^2}{\sum_i^n \sigma_i^2}, \quad (2)$$

is computed to see what at which r the it exceeds 0.95.

B) Classification

All classifiers in Section II are implemented in MATLAB, with the input data as the first r columns of V. The MATLAB functions for these classifiers are given in Appendix A. There are two classes for gender classification and 38 classes for the task of subject classification. Gender labels were created to this author's best ability. Many of these classifier routines in MATLAB are originally designed for binary classification, so the built-in function *fitcecoc()* is used to generalize these classifiers to multi-class classification.

Unless otherwise stated, default parameters in MATLAB R2016b are used for the classifiers. For KNN, k is set to 5, which is picked to minimize classification error from trial-and-error.

C) Cross-Validation

The performance of predictions on the unseen data can depend on how the dataset is split into train and test sets. Thus, the procedure of training and testing is repeated five times, each time with the data for each class randomly split into 80% for training and 20% for testing. The split was done for each class individually so that the training data can have an adequate representation of the data distribution. Cross-validation error rate is averaged across these five runs.

D) Clustering Analysis for Face Images

To analyze which facial features tend to cluster, the face images are separated into clusters using K-means, as described in Section II. For K=2, the absolute difference between the cluster means for each pixel is computed and plotted to see which set of pixels yield the largest difference between cluster means. This is because pixels with the largest distance are those that separate the clusters. For K>2, this is procedure is repeated for pairs of clusters. Because it is not straightforward what the number of clusters should be, several values of K are tried.

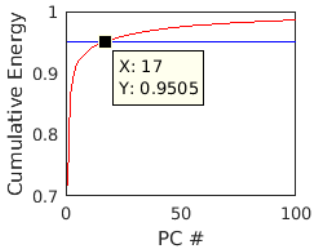
IV. Computational Results

A) SVD Analysis for Faces

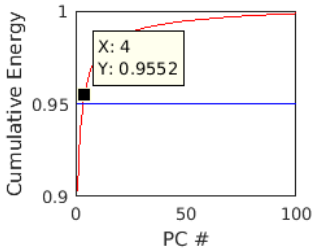
SVD analysis is performed on both the cropped face images and uncropped images, where each image is reshaped into a column vector and the image dataset forms a number of pixels by number of images matrix. Results are summarized in Figure 1. As shown in Figure 1a, the cropped images require about 17 modes to capture 95% of the total energy, whereas the uncropped images only require about 4 modes to capture about 95% of the total energy.

Rank approximations are performed on one example image for each of the cropped sets and uncropped sets and illustrated in Figure 1b and Figure 1e, respectively. The cropped example required far fewer modes than the uncropped example for the face to be recognizable. With only about 5 modes, the edges of the facial features are recognizable for the cropped photo, whereas only a rough outline of the head is recognizable for the uncropped case. With some where between 20 and 50 modes, the cropped face becomes recognizable, whereas the uncropped image is still very blurry.

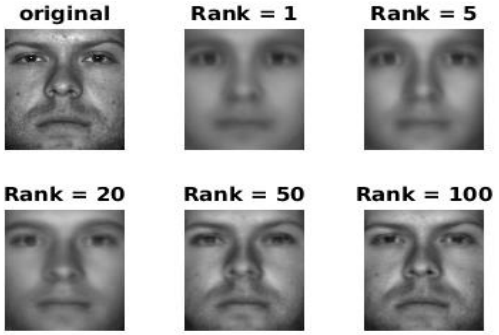
The top four modes (i.e. the first four columns of U) are plotted in Figure 1c and Figure 1f for the cropped image and uncropped image, respectively. For the cropped photo, these four modes show the edges of the facial feature, whereas the top four modes for the uncropped photo only able to capture a rough outline of the head. Hence, a significant amount of energy of the modes for the uncropped case is spent on the environment, as the top four modes dominate about 95% of the total energy, and yet they only capture a head outline but not the facial features. Thus, the cropped image set is more efficient to represent nuanced facial features and used for subject identification in the rest of this report.



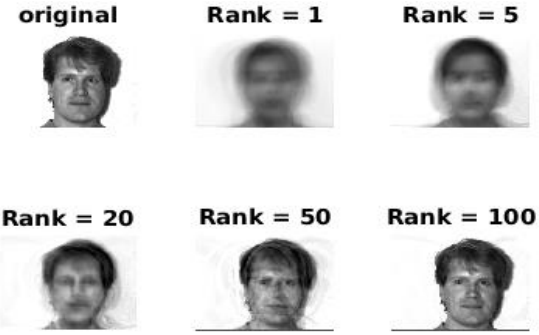
(a) Cumulative energy % (cropped)



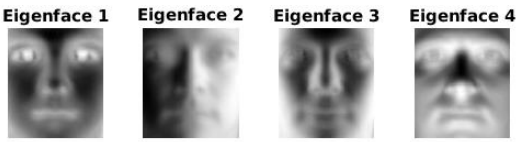
(d) cumulative energy % (uncropped)



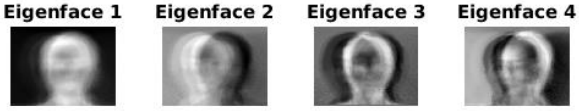
(b) Rank approximation (cropped)



(e) Rank approximation (uncropped)



(c) Modes (columns of U) (cropped)



(f) Modes (columns of U) (uncropped)

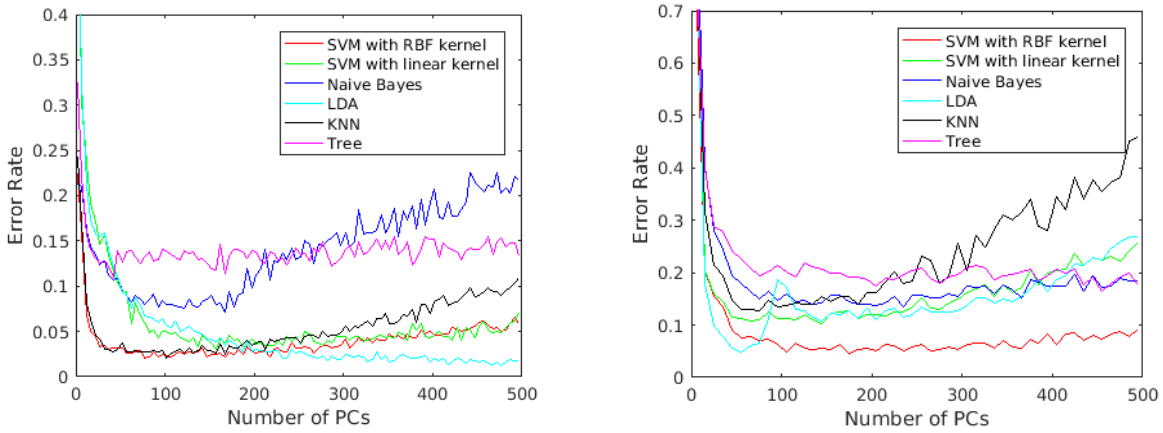
***Figure 1**: SVD Analysis of the cropped and uncropped Yale face image datasets*

B) Subject and Gender Identification

All classification methods discussed in Section III are applied to the cropped images for gender classification and subject classification, with the average cross-validation error rate over the number of modes used plotted in Figure 2.

In general, lower error rate is achieved with gender classification than subject classification, which is expected since the former is a binary classification, whereas the latter involves 38 distinct classes. When less than 100 modes are used, SVM with Gaussian kernel (RBF) and KNN achieve lower error rates than the other methods. This is probably because of their versatility in drawing nonlinear classification boundaries. Moreover, it should be noted that the choice of kernel for SVM is crucial for the performance, as SVM with a Gaussian kernel achieves a much lower error rate than linear SVM.

As the number of modes increases, the cross-validation performance of KNN start to degrade for both tasks, possibly due to overfitting. An intuitive explanation is that as the rank increases, the data becomes noisier and the identity of the nearest neighbours become more spurious. Similar trend is seen with a few other methods as well, such as Naïve Bayes. SVM with Gaussian kernel seems more robust to overfitting than other methods. These results show that dimensionality reduction on input data can be used as a tool to avoid overfitting.

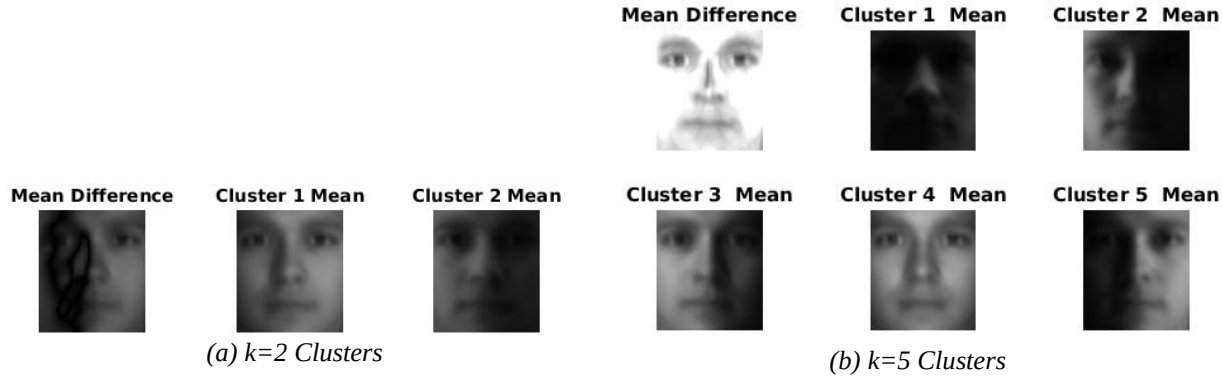


(a) Gender classification *(b) Subject classification*

***Figure 2**: Mean cross-validation error rate over the number of PCs used for various classifiers*

C) Clustering Analysis

The k-means algorithm is applied to the cropped face images to identify clusters, and the absolute difference between the centroid means for each pixel is plotted in Figure 3 and titled as “mean difference”. The higher the difference for between the centroid means for a pixel, the more important is that pixel in separating the clusters relative to other pixels. When two clusters are used (Figure 3a), the pixels that are the most important in separating the clusters are those along the edges of facial parts, capturing features such as nose width. Looking at the cluster means, these two clusters indeed have distinct edges of facial features, such as nose width and distance between the nose and mouth. Moreover, another feature that separates the clusters is the lighting of the photo, which has been neglected in this work. K-means is repeated for five clusters and these trends are again observed.



(a) k=2 Clusters *(b) k=5 Clusters*

***Figure 3**: Illustration of distance between cluster means for each pixel (mean difference) and the cluster means for each cluster obtained using k-means*

V. Summary and Conclusions

This work applied various classification methods to identify subject and gender from the Yale Face Image dataset. Cropped face images were used because they were able to represent more nuanced facial features with fewer PCs. Classification was performed on rank reduced data obtained from SVD. Cross-validation error is suboptimal when too little or too many modes are used due to underfitting or overfitting. The error is the lowest when the number of modes used sit in a good balance between the two. This report demonstrates the use of SVD to control the rank of the input data to balance between underfitting and overfitting.

Moreover, various classifier settings beyond the default ones were explored. For SVM, the choice of kernel plays an important role in determining the outcome of SVM, as it determines the complexity of the boundary that can be formed between two classes. For KNN, the number of neighbours is key to the balance between overfitting and underfitting. Future work may involve exploring different settings of decision tree beyond the default ones. Another direction would be to examine the mislabeled cases and identify possible reasons of misclassifications for improvements.

Clustering analysis suggests that images tend to cluster based on edges of facial parts, which are pertinent to subject and gender identification. However, clustering analysis also shows that images can also cluster based on their degree of lighting, which is not explored in this work. Future work may take lighting into considerations, such as normalize the pixel intensity with respect to average lighting of that photo, and/or ensuring the training set consists of an adequate representation of the lighting distribution.

VI. References:

[1] Jose Nathan Kutz. Data-driven modeling & scientific computation: methods for complex systems & big data. Oxford University Press, 2013.

Appendix A: MATLAB Functions Used

A) Built-in functions in MATLAB R2016b.

Functions for preprocessing:

- Imshow(X): displays an image with pixel intensity given by each element in the matrix X
- [U,S,V] = svd(A,0): returns the reduced singular value decomposition of A
- randperm(N): returns a random permutation of integers from 1 to N

Functions for classification:

- mdl = fitcecoc(X, Y): returns a multi-class classifier based on data given in X and labels given in Y; the default learner is SVM.
    - An example to use other learners such as LDA:
      lda = fitcecoc(X, Y, 'Learners', 'discriminant');
    - An example to specify settings in a learner:
      t = templateKNN('NumNeighbors', 5
      knn = fitcecoc(X, Y, 'Learners', t);
- pre = mdl.predict(X_{test}): predicts the labels for X_{test} using the classifier obtained above

Functions for clustering:

- [idx, C] = kmeans(X,k): performs k-means clustering to the data matrix X into k clusters, and returns idx containing the cluster indices of each datapoint and C containing the cluster means.

B) Subroutines written by the author of this report

```
function gender = get_gender(subnum)
% Return the gender of the subject specified by the subject number
% 0 - female, 1 – male

function [xtrain, ctrain, xtest, ctest, Vtrain, Vtest] = get_train_test(xdata_cell, V, rnk, task)
% Split data for 80% for training, 20% for validation. The split is done
% for every subject, which allow the training data to represent the general
% data distribution and perform better than the 80-20 split for the
% entire dataset as a whole.

% Input: all cropped face image data; task can be 'gender' or 'subject'
% Output: 80% of the data for every subject is used for training, Also
% returns the split for PC coordinate V with specified rank
% 20% of the data for every subject is used for testing.
```

Appendix B – MATLAB CODE

The MATLAB code used in this assignment have been uploaded to GitHub: https://github.com/Helena-Yuhan-Liu/Inferring-Structure-of-Complex-Systems.

A) Load data

```
% Recursively go through the folders and load all data

%% Cropped

clear;

ldir = './CroppedYale';
subdirs = dir(ldir);

xdata = []; subnum = []; gender=[]; first_im = 1;
xdata_cell=cell(39,1); % each cell contains all images pertaining to one subject
for ii = 1:length(subdirs)
   subdir = subdirs(ii);
   if ~strcmp(subdir.name, '.') && ~strcmp(subdir.name, '..')
      subsubdirs = dir([subdir.folder '/' subdir.name]);
      for jj = 1:length(subsubdirs)
         subsubdir = subsubdirs(jj);
         if ~strcmp(subsubdir.name, '.') && ~strcmp(subsubdir.name, '..')
            % Load data
            importfile([subsubdir.folder '/' subsubdir.name]);
            % Stack the image and subject number
            var_name = split(subsubdir.name,'.');
            if ~strcmp(var_name(end),'bad')
               var_name=char(var_name(1));
               var_name = strrep(var_name,'+','_');
               var_name = strrep(var_name,'-','_');
               eval(['x = ' var_name ';']);
               xdata = [xdata reshape(x, [length(x(:)) 1])];
               sn = str2double(subsubdir.name(6:7));
               subnum = [subnum; sn];
               gender = [gender; get_gender(sn)];
               xdata_cell{sn} = [xdata_cell{sn} reshape(x, [length(x(:)) 1])];
               % Get the image dimension
               if first_im
                  nrow = size(x,1);
                  ncol = size(x,2);
                  first_im = 0;
               end
            end
         end
      end
   end
end

save('cropped_data.mat', 'xdata', 'xdata_cell', 'subnum', 'gender', 'nrow', 'ncol');

%% Uncropped

clear;
```

```
ldir = './yalefaces';
subdirs = dir(ldir);

xdata = []; subnum = []; emotion=[]; first_im = 1;
for ii = 1:length(subdirs)
   subdir = subdirs(ii);
   if ~strcmp(subdir.name, '.') && ~strcmp(subdir.name, '..')
      subsubdirs = dir([subdir.folder '/' subdir.name]);
      for jj = 1:length(subsubdirs)
         subsubdir = subsubdirs(jj);
         if ~strcmp(subsubdir.name, '.') && ~strcmp(subsubdir.name, '..')
            % Load data
            importfile_uncropped([subsubdir.folder '/' subsubdir.name]);
            % Stack the image, subject number and emotion
            var_name = split(subsubdir.name,'.');
            emotion = [emotion; var_name(2)];
            subnum = [subnum; num2str(subsubdir.name(8:9))];
            x = cdata;
            xdata = [xdata reshape(x, [length(x(:)) 1])];
            if first_im
               nrow = size(x,1);
               ncol = size(x,2);
               first_im = 0;
            end
         end
      end
   end
end

save('uncropped_data.mat', 'xdata', 'subnum', 'emotion', 'nrow', 'ncol');
```

B) SVD Analysis

```
%% SVD for Cropped Faces

% Load data
% xdata, # pixels by # images, contains all the cropped faces image data
clear; close all; clc;
load('cropped_data.mat');
xdata = double(xdata);

N = size(xdata,2); % number of images
x_0mean = xdata - mean(xdata,2); % Remove baseline (mean) for each pixel

%% Run SVD
% U: columns contain the PCs, i.e. the basis for the data
% V: projection of data onto U, i.e. the component of X on the basis
% S: singular value denoting the strength of each modes

% SVD for cropped images
[Uc,Sc,Vc] = svd(xdata,0);
[Uc_0,Sc_0,Vc_0] = svd(x_0mean,0);

% Singular values

figure; semilogy(1:length(Sc),Sc,'.');
```

```
xlabel('PC #'); ylabel('Singular Value');

% Cumulative energy plot
Ek=diag(Sc).^2;
Hk_c=Ek./sum(Ek);
figure; plot(1:length(Hk_c), cumsum(Hk_c), 'r'); hold on;
plot(1:length(Hk_c),0.95*ones(size(1:length(Hk_c))),'b'); hold off;
ylabel('Cumulative Energy'); xlabel('PC #'); xlim([0 100]);
set(gcf, 'position', [100 100 250 200]); set(gcf,'color','w');

% turns out w/o de-mean, 20 modes needed for 95% of the energy
% with mean removed, 75 modes needed for 95% of the energy

%% Rank approximation to one example face
imnum=1; % example image idx

% original xdata
rnk = [1 5 20 50 100];
figure;
subplot(2,3,1)
imshow(reshape(uint8(xdata(:,imnum)),[nrow,ncol]))
title('original')
for k = 1:length(rnk)
   Xc_approx = Uc(:,1:rnk(k))*Sc(1:rnk(k),1:rnk(k))*Vc(:,1:rnk(k))';
   C = reshape(uint8(Xc_approx(:,imnum)),[nrow,ncol]);
   subplot(2,3,k+1); imshow(C);
   title(['Rank = ', num2str(rnk(k))])
end
set(gcf, 'position', [100 100 450 300]); set(gcf,'color','w');

%% Plot the top four dominant modes
m = 4;

figure;
for k = 1:m
   subplot(1,m,k);
   imshow(reshape(Uc(:,k),[nrow,ncol]),[min(Uc(:,k)) max(Uc(:,k))])
   title(['Eigenface ', num2str(k)])
end
set(gcf, 'position', [100 100 600 150]); set(gcf,'color','w');

%% SVD for Cropped Faces
% Load data
% xdata, # pixels by # images, contains all the cropped faces image data
load('uncropped_data.mat');
xdata = double(xdata);

N = size(xdata,2); % number of images
x_0mean = xdata - mean(xdata,2); % Remove baseline (mean) for each pixel

%% Run SVD for uncropped images

% SVD for cropped images
[Uu,Su,Vu] = svd(xdata,0);
[Uu_0,Su_0,Vu_0] = svd(x_0mean,0);

% Singular values
```

```
figure;semilogy(1:length(Su),Su,'.');
xlabel('PC #'); ylabel('Singular Value');

% Cumulative energy plot
Ek=diag(Su).^2;
Hk_u=Ek./sum(Ek);
figure; plot(1:length(Hk_u), cumsum(Hk_u), 'r'); hold on;
plot(1:length(Hk_u),0.95*ones(size(1:length(Hk_u))),'b'); hold off;
ylabel('Cumulative Energy'); xlabel('PC #'); xlim([0 100]);
set(gcf, 'position', [100 100 250 200]); set(gcf,'color','w');

%% Rank approximation to the faces.
imnum=1; % example image idx

% original xdata
rnk = [1 5 20 50 100];
figure;
subplot(2,3,1)
imshow(reshape(uint8(xdata(:,imnum)),[nrow,ncol]))
title('original')
for k = 1:length(rnk)
   Xc_approx = Uu(:,1:rnk(k))*Su(1:rnk(k),1:rnk(k))*Vu(:,1:rnk(k))';
   C = reshape(uint8(Xc_approx(:,imnum)),[nrow,ncol]);
   subplot(2,3,k+1); imshow(C);
   title(['Rank = ', num2str(rnk(k))])
end
set(gcf, 'position', [100 100 450 300]); set(gcf,'color','w');


%% Plot the top four dominant modes
m = 4;

figure;
for k = 1:m
   subplot(1,m,k);
   imshow(reshape(Uu(:,k),[nrow,ncol]),[min(Uu(:,k)) max(Uu(:,k))])
   title(['Eigenface ', num2str(k)])
end
set(gcf, 'position', [100 100 600 150]); set(gcf,'color','w');

% Comparison:
% Uncropped images require fewer modes to capture 95% of the energy,
% but those dominant modes are focused on capturing edges of the head
% position, rather than edges of the facial features.
% Hence, it is not surprising that cropped figure achieves a better
% approximation with fewer modes.
```

C) Classification and Clustering

```
%% Identification for Cropped Faces
% As seen from SVD, cropped images yield better approximations at
% lower rank, so they will be used for the identification tasks.

% Data are not separated based on lighting, so that the classifier can
% be trained and tested for images with varying degree of lighting

% TODO:
```

```
% analyze error cases, imshow the image, look at the subject number
% gscatter for PC

% setup
clear; close all; clc;
plot_en = 0;

% Load data
% xdata, # pixels by # images, contains all the cropped faces image data
load('cropped_data.mat');
xdata = double(xdata);

N = size(xdata,2); % number of images
x_0mean = xdata - mean(xdata,2); % Remove baseline (mean) for each pixel

[Uc,Sc,Vc] = svd(xdata,0);

%% Gender identification using supervised learning
% Various classifiers are tried to identify gender from face images

% Cross-validation by random split of data into training and test sets,
% repeat for Ntrials times, and then take the average error rate

rnks = 2:5:500; % rank to keep
Ntrials = 5; % number of cross-validation runs
error_svm = ones(length(rnks),Ntrials);
error_svm_lin = ones(length(rnks),Ntrials);
error_nb = ones(length(rnks),Ntrials);
error_lda = ones(length(rnks),Ntrials);
error_knn = ones(length(rnks),Ntrials);
error_tree = ones(length(rnks),Ntrials);

for r = 1:length(rnks)
   rnk = rnks(r);
   for ntrial = 1:Ntrials
      % Split data into train and test, also get the PC components for train
      % and test
      [xtrain, ctrain, xtest, ctest, Vtrain, Vtest]= ...
         get_train_test(xdata_cell, Vc, rnk, 'gender');

      % Run SVM with RBF kernel
      % selecting a good kernel is crucial, for example, high error rate with
      % mlp or linear kernel,
      svm = fitcsvm(Vtrain,ctrain','KernelFunction','gaussian',...
         'KernelScale','auto');
      pre_svm = svm.predict(Vtest);
      error_svm(r, ntrial) = sum(pre_svm~=ctest')/length(ctest');
      % plot
      if plot_en
         figure; subplot(2,1,1); bar(pre_svm,'r'); ylabel('predicted labels');
         subplot(2,1,2); bar(ctest','b'); ylabel('actual labels');
         xlabel('test instance'); title('SVM with RBF');
      end

      % SVM with linear kernel
      try
         svm = svmtrain(Vtrain, ctrain','kktviolationlevel',0.01);
      catch
```

```
            svm = svmtrain(Vtrain, ctrain','kktviolationlevel',0.1);
        end
        pre_svm_lin = svmclassify(svm, Vtest);
        error_svm_lin(r, ntrial) = sum(pre_svm_lin~=ctest')/length(ctest');

        % KNN, k found by trial and error
        neighb_idx = knnsearch(Vtrain,Vtest,'K',5);
        neighb_class = ctrain(neighb_idx);
        pre_knn = mode(neighb_class,2);
        error_knn(r, ntrial) = sum(pre_knn~=ctest')/length(ctest');
    %     neighb_idx = knnsearch(double(xtrain'),double(xtest'),'K',5);
    %     neighb_class = ctrain(neighb_idx);
    %     pre_knn = mode(neighb_class,2);
    %     error_knn(ntrial) = sum(pre_knn~=ctest')/length(ctest');
        % plot
        if plot_en
            figure; subplot(2,1,1); bar(pre_knn,'r'); ylabel('predicted labels');
            subplot(2,1,2); bar(ctest','b'); ylabel('actual labels');
            xlabel('test instance'); title('KNN');
        end

        % Run LDA
        pre_lda = classify(Vtest,Vtrain, ctrain');
        error_lda(r, ntrial) = sum(pre_lda~=ctest')/length(ctest');
        % plot
        if plot_en
            figure; subplot(2,1,1); bar(pre_lda,'r'); ylabel('predicted labels');
            subplot(2,1,2); bar(ctest','b'); ylabel('actual labels');
            xlabel('test instance'); title('LDA');
        end

        % Run Naive Bayes
        nb = fitNaiveBayes(Vtrain, ctrain');
        pre_nb = nb.predict(Vtest);
        error_nb(r, ntrial) = sum(pre_nb~=ctest')/length(ctest');
        % plot
        if plot_en
            figure; subplot(2,1,1); bar(pre_nb,'r'); ylabel('predicted labels');
            subplot(2,1,2); bar(ctest','b'); ylabel('actual labels');
            xlabel('test instance'); title('Naive Bayes');
        end

        % Run Decision tree
        tree = fitctree(Vtrain, ctrain');
        pre_tree = predict(tree, Vtest);
        error_tree(r, ntrial) = sum(pre_tree~=ctest')/length(ctest');
    end
end

% Plot the error rate versus rank
figure;
plot(rnks, mean(error_svm,2), 'r'); hold on;
plot(rnks, mean(error_svm_lin,2), 'g');
plot(rnks, mean(error_nb,2), 'b');
plot(rnks, mean(error_lda,2), 'c');
plot(rnks, mean(error_knn,2), 'k');
plot(rnks, mean(error_tree,2), 'm'); hold off;
xlabel('Number of PCs'); ylabel('Error Rate');
```

```
legend({'SVM with RBF kernel', 'SVM with linear kernel', 'Naive Bayes', ...
    'LDA', 'KNN', 'Tree'});
ylim([0 0.4]);
set(gcf, 'position', [100 100 450 350]); set(gcf,'color','w');
% For KNN and SVM RBF, better results when the rank is low
% For LDA and SVM linear, better results when the rank is high
% The choice of kernel function matters a lot for SVM

save('class_gender.mat','rnks','error_svm','error_svm_lin','error_nb',...
    'error_lda','error_knn','error_tree');

%% Subject identification using supervised learning
% fitcecoc() is used to apply the above classifiers to multi-classes case

rnks = [2 5:10:500]; % rank to keep
Ntrials = 3; % number of cross-validation runs
error_svm = ones(length(rnks),Ntrials);
error_svm_lin = ones(length(rnks),Ntrials);
error_nb = ones(length(rnks),Ntrials);
error_lda = ones(length(rnks),Ntrials);
error_knn = ones(length(rnks),Ntrials);
error_tree = ones(length(rnks),Ntrials);

for r = 1:length(rnks)
    rnk = rnks(r);
    for ntrial = 1:Ntrials
        % Split data into train and test, also get the PC components for train
        % and test
        [xtrain, ctrain, xtest, ctest, Vtrain, Vtest]= ...
            get_train_test(xdata_cell, Vc, rnk, 'subject');

        % SVM with Gaussian kernel
        t = templateSVM('Standardize',true,'KernelFunction','gaussian',...
            'KernelScale','auto');
        svm = fitcecoc(Vtrain,ctrain','Learners',t);
        pre_svm = svm.predict(Vtest);
        error_svm(r, ntrial) = sum(pre_svm~=ctest')/length(ctest');

        % SVM with linear kernel
        t = templateSVM('Standardize',true,'KernelFunction','linear');
        svm = fitcecoc(Vtrain,ctrain','Learners',t);
        pre_svm_lin = svm.predict(Vtest);
        error_svm_lin(r, ntrial) = sum(pre_svm_lin~=ctest')/length(ctest');

        % KNN, k found by trial and error
        t = templateKNN('NumNeighbors',5,'Standardize',1);
        knn = fitcecoc(Vtrain,ctrain','Learners',t);
        pre_knn = knn.predict(Vtest);
        error_knn(r, ntrial) = sum(pre_knn~=ctest')/length(ctest');

        % Run LDA
        lda = fitcecoc(Vtrain,ctrain','Learners','discriminant');
        pre_lda = lda.predict(Vtest);
        error_lda(r, ntrial) = sum(pre_lda~=ctest')/length(ctest');
        % plot
        if plot_en
            figure; subplot(2,1,1); bar(pre_lda,'r'); ylabel('predicted labels');
            subplot(2,1,2); bar(ctest','b'); ylabel('actual labels');
```

```
        xlabel('test instance'); title('LDA');
    end

    % Run Naive Bayes
    nb = fitcecoc(Vtrain,ctrain','Learners','naivebayes');
    pre_nb = nb.predict(Vtest);
    error_nb(r, ntrial) = sum(pre_nb~=ctest')/length(ctest');
    % plot
    if plot_en
        figure; subplot(2,1,1); bar(pre_nb,'r'); ylabel('predicted labels');
        subplot(2,1,2); bar(ctest','b'); ylabel('actual labels');
        xlabel('test instance'); title('Naive Bayes');
    end

    % Run Decision tree
    tree = fitcecoc(Vtrain,ctrain','Learners','tree');
    pre_tree = tree.predict(Vtest);
    error_tree(r, ntrial) = sum(pre_tree~=ctest')/length(ctest');
  end
end

% Plot the error rate versus rank
figure;
plot(rnks, mean(error_svm,2), 'r'); hold on;
plot(rnks, mean(error_svm_lin,2), 'g');
plot(rnks, mean(error_nb,2), 'b');
plot(rnks, mean(error_lda,2), 'c');
plot(rnks, mean(error_knn,2), 'k');
plot(rnks, mean(error_tree,2), 'm'); hold off;
xlabel('Number of PCs'); ylabel('Error Rate');
legend({'SVM with RBF kernel', 'SVM with linear kernel', 'Naive Bayes', ...
  'LDA', 'KNN', 'Tree'});
ylim([0 0.7]);
set(gcf, 'position', [100 100 450 350]); set(gcf,'color','w');

save('class_subject.mat','rnks','error_svm','error_svm_lin','error_nb',...
  'error_lda','error_knn','error_tree');

%% K-means Clustering
% For unsupervised learning, there is no teaching signal, so no
% precise way of evaluating the performance. Usually domain knowledge
% would be useful in evaluating if the clusters make sense

% GMM was skipped b/c the dimension of the data far exceeds the
% number of observations, making the cov matrix non-invertible

% Try k=2 clusters
[~, C2] = kmeans(xdata',2);
mean_diff = abs(C2(1,:)-C2(2,:));
figure;
subplot(1,3,1); imshow(reshape(uint8(mean_diff),[nrow,ncol]));
title('Mean Difference');
subplot(1,3,2); imshow(reshape(uint8(C2(1,:)),[nrow,ncol]));
title('Cluster 1 Mean');
subplot(1,3,3); imshow(reshape(uint8(C2(2,:)),[nrow,ncol]));
title('Cluster 2 Mean');
set(gcf, 'position', [100 100 600 150]); set(gcf,'color','w');
% Pixels that really separate the clusters turn out to be edges of
```

```
% facial parts, capturing features such as nose width
% Lighting also got picked up, as seen in the two different cluster means

% Try k=5 clusters
[~, C5] = kmeans(xdata',5);
mean_diff = abs(C5(1,:)-C5(5,:));
for ii=2:5
   mean_diff = mean_diff+abs(C5(ii,:)-C5(ii-1,:));
end
figure;
subplot(2,3,1); imshow(reshape(uint8(mean_diff),[nrow,ncol]));
title('Mean Difference');
for ii=1:5
   subplot(2,3,ii+1); imshow(reshape(uint8(C5(ii,:)),[nrow,ncol]));
   title(['Cluster ' num2str(ii) '  Mean']);
end
set(gcf, 'position', [100 100 600 300]); set(gcf,'color','w');
% Again, pixels that really matter are those that picks up edges
% Also clustering picks up lighting, as seen in different cluster means
% These most important pixels can be used for more efficient
% classification, which is skipped in this work

save('kmeans_data.mat','C2','C5');
```