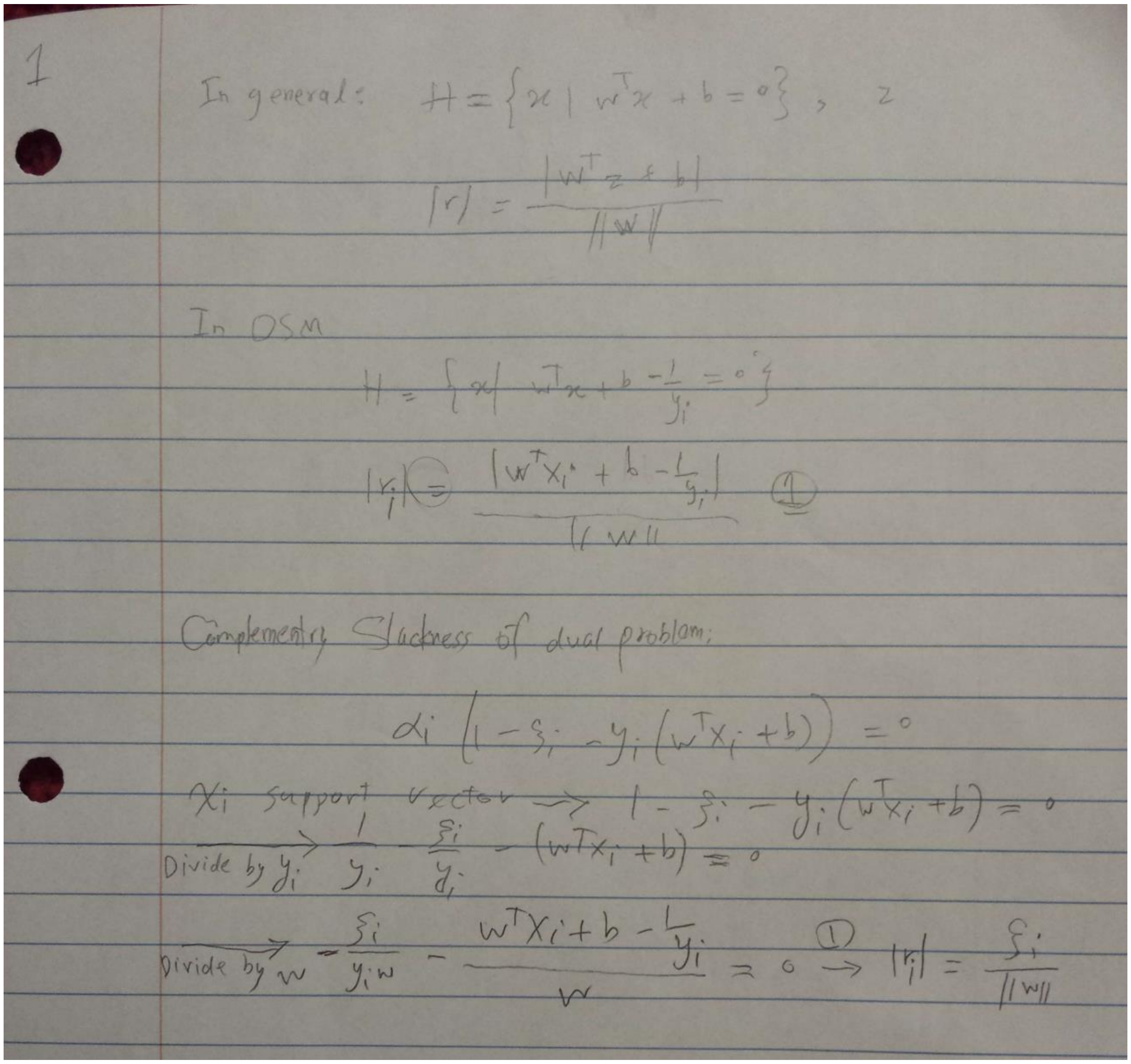
1. Margins and Slack Variables (5 points)

Let $(\mathbf{w}, b, \boldsymbol{\xi})$ be the solution of the optimal soft-margin hyperplane based on training data $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)$. Show that if $\boldsymbol{x}_i$ is a margin error (see notes), then the distance from $\boldsymbol{x}_i$ to the hyperplane

$$\{\boldsymbol{x} : y_i(\mathbf{w}^T\boldsymbol{x} + b) = 1\}$$

is proportional to ξ_i, and give the constant of proportionality.



1

In general: $H = \{x \mid w^T x + b = 0\}$, z

$$|r| = \frac{|w^T z + b|}{\|w\|}$$

In OSM

$$H = \{x \mid w^T x + b - \frac{1}{y_i} = 0\}$$

$$|r_i| = \frac{|w^T x_i + b - \frac{1}{y_i}|}{\|w\|} \quad ①$$

Complementry Slackness of dual problem:

$$\alpha_i \left(1 - \xi_i - y_i(w^T x_i + b)\right) = 0$$

x_i support vector $\rightarrow 1 - \xi_i - y_i(w^T x_i + b) = 0$

$$\xrightarrow{\text{Divide by } y_i} \frac{1}{y_i} - \frac{\xi_i}{y_i} - (w^T x_i + b) = 0$$

$$\xrightarrow{\text{Divide by } w} \frac{\xi_i}{y_i w} - \frac{w^T x_i + b - \frac{1}{y_i}}{w} = 0 \xrightarrow{①} |r_i| = \frac{\xi_i}{\|w\|}$$

2. Support Vector Machines and Cross Validation (10 points)

Download the file `diabetes_scaled.mat` from Canvas. This contains variables `X` and `y` for a classification problem. There are 768 labeled feature vectors. The features are

```
1. Number of times pregnant
2. Plasma glucose concentration
3. Diastolic blood pressure (mm Hg)
4. Triceps skin fold thickness (mm)
5. 2-Hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m)^2)
7. Diabetes pedigree function
8. Age (years)
```

The binary labels are

```
Binary label:
   +1 = tested positive for diabetes
   -1 = non-positive
```

Reserve the last 268 instances for testing. The goal is to minimize the probability of error on the test data. On the 500 training examples, train an SVM using the Cauchy kernel

$$k(\mathbf{u},\mathbf{v}) = \left(1 + \frac{\|\mathbf{u}-\mathbf{v}\|^2}{\sigma^2}\right)^{-1},$$

using a 5-fold cross validation estimate of the probability of error to select the parameters C and σ. Report the selected parameters, the cross-validation error estimate at those parameters, the test error, and the number of support vectors of the final classifier. Also, submit your code.

Comments:

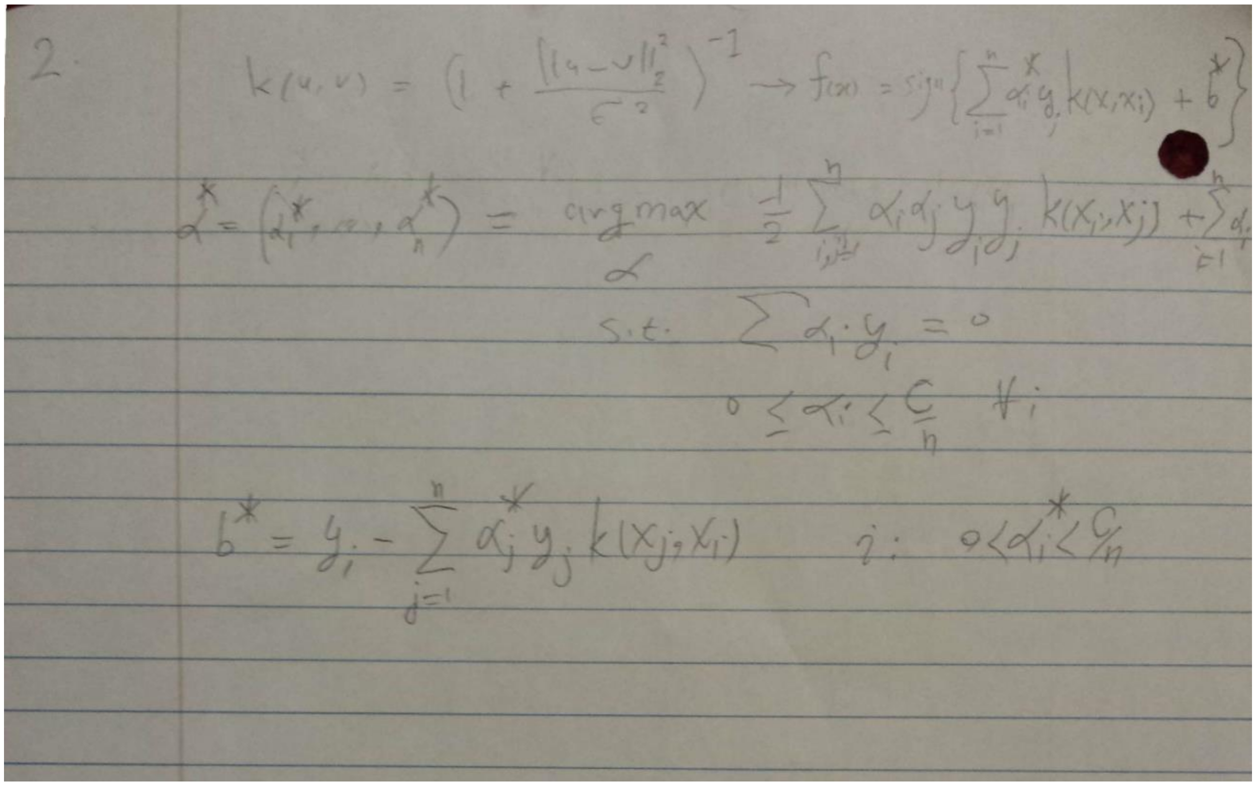
- Matlab users: we have provided you with the file `smo.m`, which implements the SMO algorithm mentioned in the notes. Type `help smo` to see how it works. If you are curious, take a look "under the hood" and see if you can understand what it is doing. Alteratively, Matlab provides an SVM solver in one of its toolboxes, or in Python, you can use the scikit-learn library and run "from sklearn.svm import SVC." Details of this function can be found at
`http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`.

- For consistency of everyone's solutions, please search over the following grid of parameter values (in Matlab notation):

  ```
  grid_sigma = 2.^(0:5);
  grid_C     = 2.^(6:11);
  ```

  So you are searching over 36 parameter combinations.

- SVM solvers often have a tolerance parameters that determines the algorithm's termination criterion. For example, the provided `smo.m` routine has a tolerance parameter that yields good results when set to 0.01. Setting the parameter too small can lead to long run times, and setting it to large can lead to poor convergence. Try to find a middle ground.
- Please don't use randomization to select the folds, even though this is generally a good idea. These data have already been randomized. This way everyone should get a similar answer.
- Please also don't worry about making sure the class proprtions are the same within each fold. Once again this is generally a good idea, but in this problem it is not necessary.



2.

$$k(u,v) = \left(1 + \frac{\|u-v\|_2^2}{\sigma^2}\right)^{-1} \rightarrow f(x) = \text{sign}\left\{\sum_{i=1}^{n} \alpha_i^* y_i k(x,x_i) + b^*\right\}$$

$$\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*) = \underset{\alpha}{\text{argmax}} \; -\frac{1}{2}\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(x_i,x_j) + \sum_{i=1}^{n} \alpha_i$$

$$\text{s.t.} \quad \sum \alpha_i \cdot y_i = 0$$

$$0 \leq \alpha_i \leq \frac{C}{n} \quad \forall i$$

$$b^* = y_i - \sum_{j=1}^{n} \alpha_j^* y_j k(x_j, x_i) \qquad i: \; 0 < \alpha_i^* < C/n$$

б=1, C=2048, cross validation error estimate=, test error=13.81%, # of support vectors=16

Code of problem 2

```
close all; clear all; clc
load diabetes_scaled.mat
global sigma

n=500;
d=2;

x_train=X(1:n,:);
y_train=y(1:n);
x_test=X(n+1:end,:);
y_test=y(n+1:end);

grid_sigma=2.^(0:5);
grid_C=2.^(6:11);

k=5;
tol=1e-2;

sigma_count=0;
for sigma=grid_sigma
    sigma_count=sigma_count+1;
    C_count=0;
    for C=grid_C
        C_count=C_count+1;
        for l=1:5
            x_traink=x_train;
            y_traink=y_train;
            x_testk=x_traink((n/k)*(l-1)+1:(n/k)*(l-1)+(n/k),:);
            y_testk=y_traink((n/k)*(l-1)+1:(n/k)*(l-1)+(n/k),:);
            x_traink((n/k)*(l-1)+1:(n/k)*(l-1)+(n/k),:)=[];
            y_traink((n/k)*(l-1)+1:(n/k)*(l-1)+(n/k),:)=[];
            for i=1:size(x_traink,1)
                for j=1:size(x_traink,1)
                    K(i,j)=Cauch_ker(x_traink(i),x_traink(j));
                end
            end
            [alpha,bias]=smo(K,y_traink',C,tol);
            for i=1:size(x_traink,1)
                if alpha(i)>0 && alpha(i)<C/n
                    sum1=0;
                    for j=1:size(x_traink,1)

sum1=sum1+alpha(j)*y_traink(j)*Cauch_ker(x_traink(j),x_traink(i));
                    end
                    b=y_traink(i)-sum1;
%                     break;
                end
            end
            for i=1:size(x_testk,1)
                sum1=0;
                for j=1:size(x_traink,1)

sum1=sum1+alpha(j)*y_traink(j)*Cauch_ker(x_testk(i),x_traink(j));
                end
                y_pred(i)=sign(sum1+b);
            end
            R(l)=norm(y_pred'-y_testk);
        end
        Rcv(sigma_count,C_count)=1/k*sum(R);
    end
end
[M,I]=min(Rcv(:))
[I_row,I_col]=ind2sub(size(Rcv),I)

sigma=grid_sigma(I_row)
C=grid_C(I_col)

for i=1:size(x_train,1)
    for j=1:size(x_train,1)
        K(i,j)=Cauch_ker(x_train(i),x_train(j));
    end
end
[alpha,bias]=smo(K,y_train',C,tol);
sv_count=0;
for i=1:size(x_train,1)
    if alpha(i)>0 && alpha(i)<C/n
        sv_count=sv_count+1;
        sum1=0;
        for j=1:size(x_train,1)
            sum1=sum1+alpha(j)*y_train(j)*Cauch_ker(x_train(j),x_train(i));
        end
        b1(sv_count)=y_train(i)-sum1;
    end
end
sv_count
b=mean(b1);
for i=1:size(x_test,1)
    sum1=0;
    for j=1:size(x_train,1)
        sum1=sum1+alpha(j)*y_train(j)*Cauch_ker(x_test(i),x_train(j));
    end
    y_pred(i)=sign(sum1+b);
end
error_test=sum((y_pred'-y_test)/2)/numel(y_test)*100
```

Code for Cauchy kernel

```
function k=Cauch_ker(u,v)
global sigma
k=(1+(norm(u-v)^2/sigma^2))^-1;
end
```

3. PCA (5 points each)

Read over the proof of PCA based on the generalized Rayleigh quotient. Then solve the following problems. The first problem motivates a method for selecting k, as discussed at the end of the first set of notes on PCA.

a. Let $k \in \{0, 1, \dots, d\}$ be arbitrary. Show that

$$\min_{\boldsymbol{\mu}, A, \{\boldsymbol{\theta}_i\}} \sum_{i=1}^{n} \|\boldsymbol{x}_i - \boldsymbol{\mu} - A\boldsymbol{\theta}_i\|^2 = n \sum_{j=k+1}^{d} \lambda_j,$$

where A ranges over all $d \times k$ matrices with orthonormal columns.

Hint: This is easy if you use properties of the trace operator.

b. Give a condition involving the spectral decomposition of the sample covariance matrix that is both necessary and sufficient for the subspace $\langle A \rangle$ in PCA to be unique.

3 (a)

$$\min_{\mu, A, \{\theta_i\}} \sum_{i=1}^{n} \|x_i - \mu - A\theta_i\|_2^2 = n \sum_{j=k+1}^{d} \lambda_j$$

$A \in \mathbb{R}^{d \times k}$ with orthonormal cols.

$$\min_{\mu, A, \{\theta_i\}} \sum_{i=1}^{n} \|x_i - \mu - A\theta_i\|_2^2 = \min_{P \in P_k} \|X - PX\|_F$$

where,

$$X = [x_1, \dots, x_n]$$

$$\xrightarrow{\text{prop } 4} \|X - PX\|_F = tr\left[(X-PX)^T(X-PX)\right] = tr(X^TX) - tr(X^TPX) - tr(X^T$$
$$- tr(X^TP^TP$$

$$P = P^T, \quad P^2 = P$$

$$\rightarrow \|X - PX\|_F = tr(X^TX) - tr(X^TPX), \quad P = AA^T, A \in A$$

$$tr(X^TPX) = tr(X^TAA^TX) \overset{\text{Prop2}}{=} tr(A^TXX^TA)$$

$$\max_{A \in A_k} tr(A^TXX^TA)$$

$$XX^T = U\Lambda U^T, \quad U = [u_1 \dots u_d], \quad \Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ \emptyset & & \lambda_d \end{bmatrix}$$

GRQ: $A = [u_1 \dots u_k] \rightarrow A^TXX^TA =$

$$\begin{bmatrix} u_1^T \\ \vdots \\ u_k^T \end{bmatrix} [u_1 \dots u_d] \begin{bmatrix} \lambda_1 & & \emptyset \\ & \ddots & \\ 0 & & \lambda_d \end{bmatrix} \begin{bmatrix} u_1^T \\ \vdots \\ u_d^T \end{bmatrix} [u_1 \dots u_k] \rightarrow tr(A^TXX^TA) = \sum_{i=1}^{k} \lambda_i$$

$$tr(X^TX) \overset{\text{Prop2}}{=\!=} tr(XX^T) \overset{\text{Prop3}}{=\!=} \sum_{i=1}^{d} \lambda_i$$

$$\|X - PX\|_F = tr(X^TX) - tr(A^TXX^TA) = \sum_{i=k+1}^{d} \lambda_i$$

3(b)

$$A = \begin{bmatrix} a_1^T \\ \vdots \\ a_k^T \end{bmatrix}$$

$$\theta_i^{(j)} = a_j^T x_i \rightarrow \mathrm{Var}(\theta^{(j)}) = \frac{1}{n}\sum_{i=1}^{n} (a_j^T x_i)^2$$

$$= \frac{1}{n}\sum_{i=1}^{n} a_j^T x_i x_i^T a_j = a_j^T \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T a_j$$

$$= a_j^T U \Lambda U^T a_j = (U^T a_j)^T \Lambda (U^T a_j) = c^T \Lambda c$$

$$\max_{\|c\|=1} c^T \Lambda c = \max_{\|c\|=1} c_1^2 \lambda_1 + \cdots + c_d^2 \lambda_d$$

$$\text{if } \lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_d \longrightarrow c^* = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

4. Eigenfaces (5 points each)

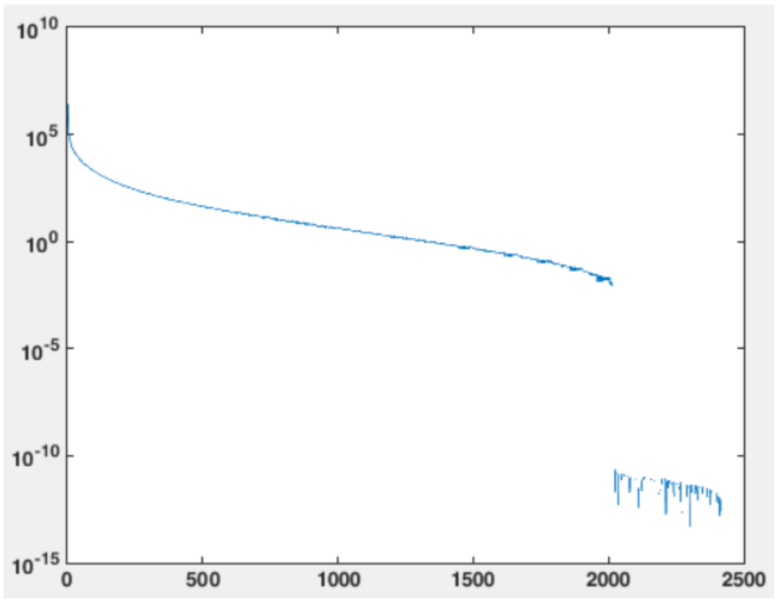
In this exercise you will apply PCA to a modified version of the Extended Yale Face Database B. The modified database is available in the file `yalefaces.mat` on Canvas. The modification I made was simply to reduce the resolution of each image by a factor of $4 \times 4 = 16$ to hopefully avoid computational and memory bottlenecks.

a. By viewing each image as a vector in a high dimensional space, perform PCA on the full dataset. Hand in a plot the sorted eigenvalues (use the `semilogy` command in Matlab; `plt.semilogy` in Python) of the sample covariance matrix. How many principal components are needed to represent 95% of the total variation? 99%? What is the percentage reduction in dimension in each case? Useful commands in Matlab:`reshape, eig, svd, mean, diag`, and Python: `np.reshape, np.linalg.eig, np.linalg.svd, np.mean, np.diag`.

b. Hand in a 4×5 array of subplots showing principal eigenvectors ('eigenfaces') 0 through 19 as images, treating the sample mean as the zeroth order principal eigenvector. Comment on what facial or lighting variations some of the different principal components are capturing. Useful commands in Matlab: `subplot, imagesc, colormap(gray)`, in Python: `plt.imshow(x, cmap=plt.get_cmap('gray'))`, and for subplots a useful link is
`http://matplotlib.org/examples/pylab_examples/subplots_demo.html`

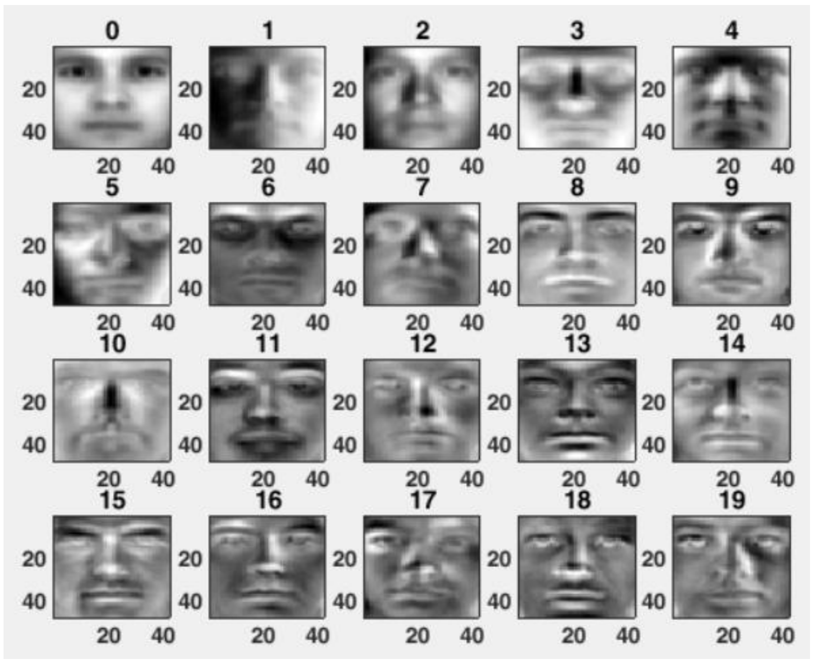
c. Submit your code.

(a) plot of sorted eigenvalues of data covariance matrix



44 (97.8175% reduction of dimension) and 168 (91.6667% reduction of dimension) PCs are needed to represent 95% and 99% of total variations, respectively.

(b) plot of principle eignevectors



The results suggests that variations in contrast of left and right side of face (1st order principle eigenvector) can be used as best parameter to distinguish between images

(c) code

```
close all; clear all; clc
load yalefaces

for i=1:size(yalefaces,3)
X(:,i)=reshape(double(yalefaces(:,:,i)),size(yalefaces,1)*size(yalefaces,2),
1);
end
S=cov1(X);
[V,D]=eig(S);
semilogy(real(diag(D)))
sum=0;
k=1;
while sum/trace(D)<.99
    sum=sum+D(k,k);
    k=k+1;
end
k
subplot(451)
imagesc(reshape(mean(X,2),size(yalefaces,1),size(yalefaces,2)));
colormap(gray)
title('0');

for i=1:19
subplot(4,5,i+1);
imagesc(reshape(V(:,i),size(yalefaces,1),size(yalefaces,2)));
colormap(gray)
title(num2str(i));
end
```