MAT 160 FINAL PROJECT

1. **Challenge: Automatic Classification of images via SVD decomposition**

   You will write an algorithm in MATLAB for the classification of handwritten digits. For this you can download the following files from https://www.math.ucdavis.edu/~deloera/TEACHING/MATH160/guessdigit-project.zip

   NOTE: Data comes in compressed form, to open you type (after downloading), unzip guessdigit-project.zip. Inside the directory that will open up you will find 5 files:

```
ima2.m          -- Code Displays an image vector in the right orientation

azip.mat        -- the matrix of training digits data

dzip.mat        -- command dzip(i) tells you the (correct) digit you have in column
                   i of the matrix azip

dtest.mat       -- tells you the (correct answer) of test digits

testzip.mat     -- the test digits data
```

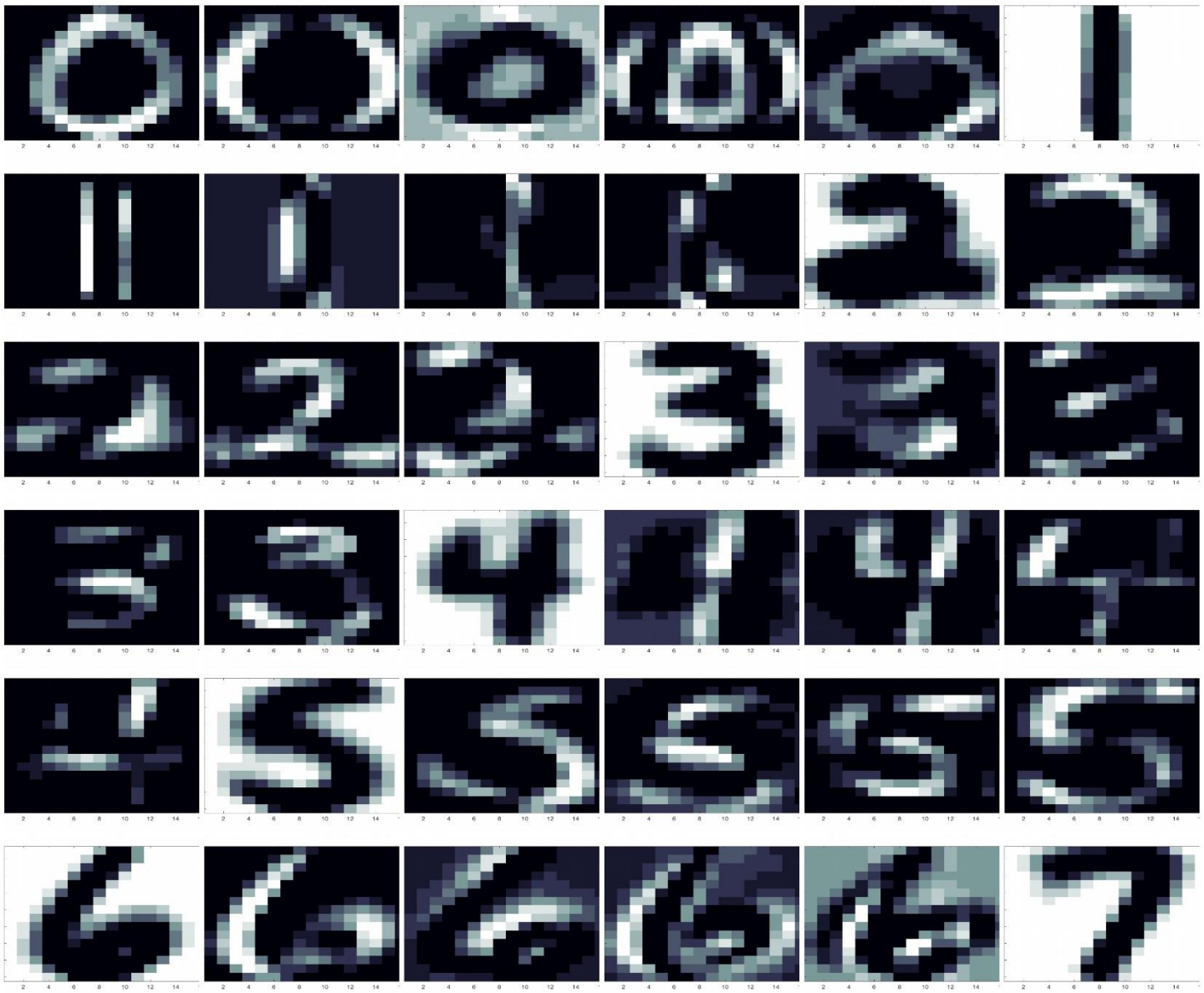
   Use the training set, and compute the SVD of each class matrix (classes are those matrices that represent the same digit). Use the first few (5, 10, 20) singular vectors as basis of a class and classify unknown test digits according to how well they can be represented in terms of the respective bases (use the relative residual vector in the least squares problem as a measure). Here are some specific tasks.

   - Write your code to do classification, it brakes the training data in classes, computes the SVD of each class and uses that to make predictions. It takes in a test data point and makes a prediction.
   - Tune the algorithm for accuracy of classification. Give a table or graph of the percentage of correctly classified digits as a function of the number of basis vectors. Graph the situation for 5, 10, 20 basis vectors. Display the results in a table (or tables).
   - Check the singular values of the different classes. Is it reasonable to use different numbers of basis vectors for different classes? If so, perform a few experiments to find out if it really pays off to use fewer basis vectors in one or two of the classes (i.e., do you get different/same outcome?).
   - Check if all digits are equally easy or difficult to classify. Also look at one of the difficult ones, and see that in many cases they are very badly written. What is the most difficult digit to read for the computer? Does it help to increase the number of singular vectors you used? Write comments at the very end of your code with your thoughts.

```
clc;
close all;
clear all;
load('azip.mat'); %Training Set
load('dzip.mat');  %Training Digit
load('testzip.mat'); %Testing set
load('dtest.mat');   %Testing Digit
rowNames
={'Digit0','Digit1','Digit2','Digit3','Digit4','Digit5','Digit6','Digit7','Digit8','Digit
9'};
[rowdtest coldtest] = size(dtest);
[rowazip colazip] = size(azip);
DigitAccuracy = [];%Allocate accuracy matrix
CorrectlyClassDig = [];%Allocate accuracy for each class
counter=1;
%k = [1:50];
k=[5,10,20];
for i=1:size(k')
    m = i;
    leftSingularVector = zeros(rowazip,m,10);
    for l=0:9 %find left singular Vector for 0,..,9 digits.
```

```
        leftSingularVector(:,:,l+1) = computeLeftSVD(azip(:,dzip==l),m);
    end
    DigitLabel = -1*ones(size(dtest));
    for j = 1:coldtest
        DigitLabel(j) = findDigit(leftSingularVector,testzip(:,j));
    end
    %Compare found digit with dtest
    for J=0:9
        lab = dtest == J;
        DigitAccuracy(counter, J+1)= sum(DigitLabel(lab)~=J)/sum(lab);%Find Wrong Digit
        DigitAccuracy(counter, J+1)= (1 - DigitAccuracy(counter, J+1))*100; %Calculate
percentage
    end
    totalError = 100*sum((DigitLabel - dtest) ~= 0)/coldtest;
    TotalAccuracy(counter)= 100-totalError;
    counter = counter+1;
end

[percentage,location] = max(DigitAccuracy);
for i = 1:size(location')
    for j = 1:size(k')
    if location(i) == j
        location(i) = k(j);
    end
    end
end

figure
plot(DigitAccuracy);
ylabel('Correct Classified Digits (%)');
xlabel('Basis vector');
xticks([1 2 3]);
xticklabels({'5','10','20'});
legend('Digit0','Digit1','Digit2','Digit3','Digit4','Digit5','Digit6','Digit7','Digit8','
Digit9');

Result = [percentage; location];
T = array2table(Result','RowNames',rowNames,'VariableNames', {'Accuracy','BasisVector'})

function U = computeLeftSVD(X, K)
    %compute the SVD of each class matrix and return k left singular vectors
    [U,S,V] = svds(X,K);
end
function digit = findDigit(foundApproximation, actualData)
    %Find digit with minimum error comparing testing set and left singular
    %vector
    uperbound = realmax;
     alloDigit = -1;
    for i =0:9 %Compute 2-norm Residual Error ||y - K*(K'*y||2
        E = norm(actualData -
foundApproximation(:,:,i+1)*(foundApproximation(:,:,i+1)'*actualData),2);
        if uperbound > E
            uperbound = E;
            alloDigit = i;
        end
    end
    digit = alloDigit;
end
```

For the first part, in the code above it is clear how I compute the SVD for each class. Additionally, I used ima2 to compute the image of the digits for some values of k. Which is shown in the image below.

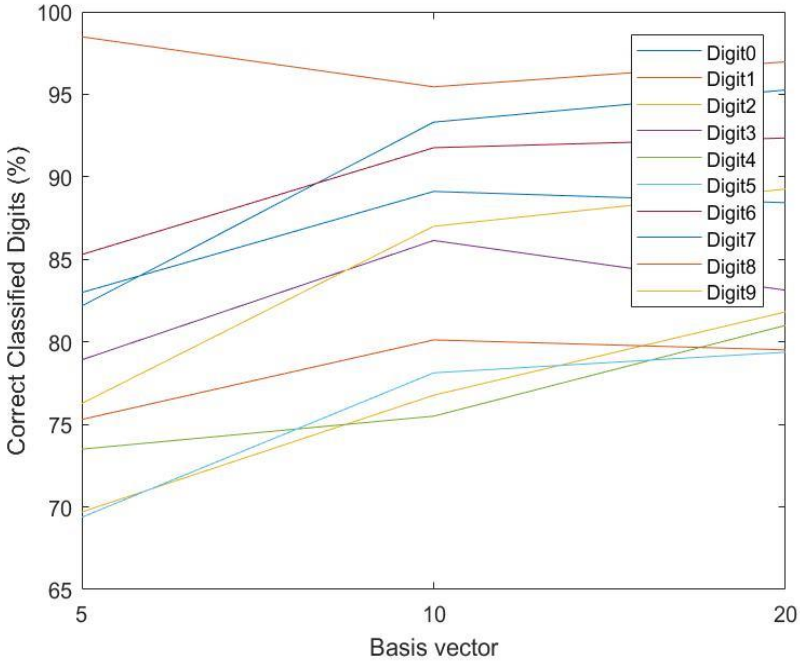


For part 2, the graph and table for the basis vector (5,10,20) are the following:

T =

10×2 table

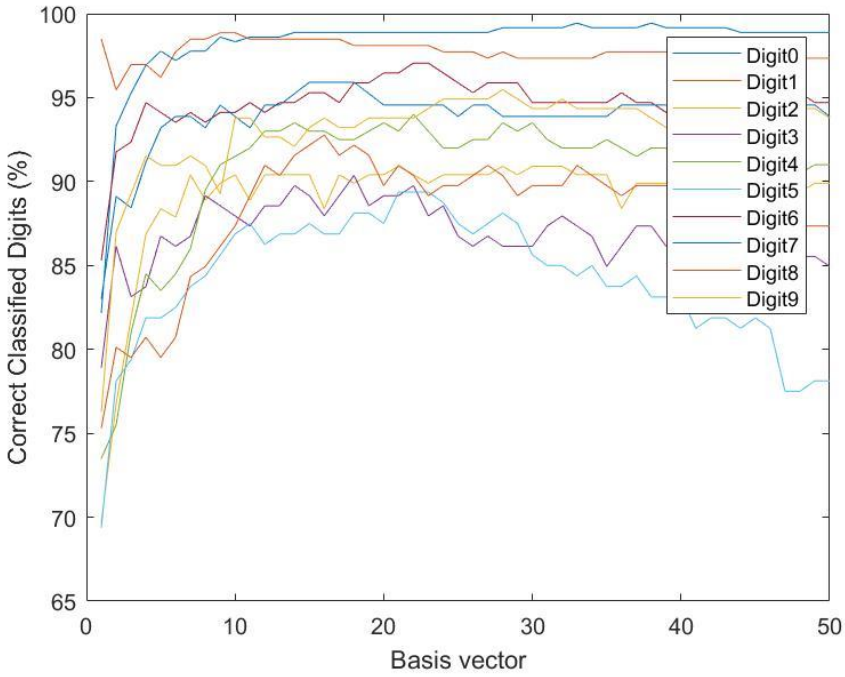
	Accuracy	BasisVector
Digit0	95.265	20
Digit1	98.485	5
Digit2	81.818	20
Digit3	86.145	10
Digit4	81	20
Digit5	79.375	20
Digit6	92.353	20
Digit7	89.116	10
Digit8	80.12	10
Digit9	89.266	20



For part 3, I run the program for all the basis vectors from 1 to 50, and as a result, Yes! It is reasonable to use different basis vectors for different classes. As shown below, almost all digits are more accurate in a different basis vector. For example, from the graph, we see that digit 5 become less accurate as it increases.

```
T =

  10×2 table
```

	Accuracy	BasisVector
Digit0	99.443	33
Digit1	98.864	9
Digit2	90.909	21
Digit3	90.361	18
Digit4	94	22
Digit5	89.375	21
Digit6	97.059	22
Digit7	95.918	15
Digit8	92.771	16
Digit9	95.48	28



For part 3, as I explained above and it is shown in the graph. Digit 5 is the most difficult to predict as we increase the number of basis vectors. In addition, Digit 5 is also the one how attained the lowest maximum accuracy possible in the run 1…50 with a 89.375% accuracy at the basis vector 21. At the end, increasing the number of basis vectors does not help the system to be more accurate. It is clear that after 28, there was not a better basis vector than the ones before.

2. **Challenge: Models for predicting the quality of wines.**

In this problem we will use two types of convex-linear models to to predict wine quality (as judged by enologists) from chemical measurements. The dataset you need to use is available at `http://www.math.ucdavis.edu/~deloera/TEACHING/MATH160/winesinfo.csv`. In each line, the first 11 columns contain the results from various chemical tests performed on the wine, and the last column is the evaluation of how good the wine is (a score between 0 and 10).

First, consider a model based on Linear programming. For wine sample i, let us denote by $y_i \in \mathbb{R}$ its score and by $\mathbf{x}_i \in \mathbb{R}^{11}$ its chemical properties. Construct a linear model to predict y_i as a function of $\mathbf{x}_i$, that is, we want to find $\mathbf{a} \in \mathbb{R}^{11}$ and $b \in \mathbb{R}$ such that:

$$y_i \simeq \mathbf{a}^\intercal \mathbf{x}_i + b.$$

The quality of the model will be evaluated using the ℓ_1 norm, *i.e.*, we want to find a solution to this optimization problem:

$$\min_{\substack{\mathbf{a}\in\mathbb{R}^{11} \\ b\in\mathbb{R}}} \frac{1}{n}\sum_{i=1}^{n} |y_i - \mathbf{a}^\intercal \mathbf{x}_i - b|.$$

a. Remember from class that the above problem is equivalent to the following linear program:

$$\begin{aligned} \min_{\substack{\mathbf{a}\in\mathbb{R}^{11} \\ b\in\mathbb{R} \\ \mathbf{z}\in\mathbb{R}^n}} \ & \frac{1}{n}\sum_{i=1}^{n} z_i \\ \text{s.t. } & z_i \geq y_i - \mathbf{a}^\intercal \mathbf{x}_i - b, 1 \leq i \leq n \\ & z_i \geq \mathbf{a}^\intercal \mathbf{x}_i + b - y_i, 1 \leq i \leq n \end{aligned}$$

Explain how to rewrite this problem in matrix form:

$$\min_{\mathbf{d}\in\mathbb{R}^{12+n}} \mathbf{c}^\intercal \mathbf{d}$$
$$\text{s.t. } A\mathbf{d} \leq \mathbf{b}$$

In particular, give the dimensions and definitions of $\mathbf{c}$, A and $\mathbf{b}$.

b. Use an LP solver (again, we recommend using SCIP, MATLAB works too) to solve the above problem and report your code as well as the optimal value of the problem. Note that the value of the problem is exactly the average absolute error of the linear model on the dataset. Does it seem to be within an acceptable range?

In this next part, you will re-use the dataset to fit a different linear model to predict wine quality as a function of the chemical measurements, but we will use least-squares regression instead of ℓ_1-regression.

a. Compute the optimal solution to the least-squares regression problem. Report your code, the linear model ($\mathbf{a}$ and b) and the value of function RSS for this model.

b. Now sparsify the model to find the key 4 features of the wine that make it a good wine. What are the top 4 features for deciding quality of wines according to a LASSO model?

```
clc;
close all;
clear all;
x = dlmread('winesinfo.csv',';',1,0);
y = x(:,12); %Read chemical properties
x = x(:,1:11); %Read score
[m,n] = size(x);
rowNames
={'fixedAcidity','volatileAcidity','citricAcid','residualSugar','chlorides','freeSulfurDi
oxide','totalSulfurDioxide','density','pH','sulphates','alcohol','OptimalValue'};
t = [];
LA = [];
disp('Part b) using 1-norm')
for i = 1:11
cvx_begin quiet
    variable w2(n);
    variable b;
    minimize(norm(x*w2 + b -y,1));
cvx_end
w2(12) = cvx_optval;
T(:,2) = w2;
end
HyperplaneW2 = array2table(w2,'RowNames',rowNames);
display(HyperplaneW2);
disp('Part 2b) least squares regression')
for i = 1:11
cvx_begin quiet
    variable w3(n);
    variable b;
    minimize(norm(x*w3 +b - y,2));
cvx_end
w3(12) = cvx_optval;
T(:,3) = w3;
end
HyperplaneW3 = array2table(w3,'RowNames',rowNames);
display(HyperplaneW3);
disp('Part 2b) LASSO Model')
lambda = 1;
for i = 1:5
cvx_begin quiet% LASSO implementation
    variable w4(n);
    variable b;
    minimize norm(y-x*w4+b,2) + lambda * norm(w4+b,1);
cvx_end
```

```
w4(12) = cvx_optval;
LA(:,i) = w4;
lambda = lambda + 2*i; %Try diffent values for lambda
end
Z = abs(LA);
for i= 1:12
 Rowsum(i,1) = sum(Z(i,:));
end
LA(:,6) = Rowsum;
Total = array2table(LA,...
    'VariableNames',{'lamda1','lamda3','lamda5','lamda7','lamda9','AbsSum'},...
    'RowNames',rowNames);
HyperplanesW4 = sortrows(Total,6, 'descend')
display('Top 4 Features For deciding quality of wines as shown in the table are')
for i = 2:5
display(HyperplanesW4(i,6));
end
```

Part a). Remember that (1) $\min_{\substack{a\in\mathbb{R}^n \\ b\in\mathbb{R}}} \frac{1}{n}\sum_{i=1}^{n} |y_i - a^T x_i - b|$ is equivalent to the following,

$\min_{\substack{a\in\mathbb{R}^n \\ b\in\mathbb{R} \\ z\in\mathbb{R}^n}} \frac{1}{n}\sum_{i=1}^{n} z_i$ (2) subject to $z_i \geq y_i - a^T x_i - b$, $1 \leq i \leq n$

$z_i \geq a^T x_i + b - y_i$, $1 \leq i \leq n$.

Explain how to rewrite this problem in matrix form.

$\min_{d\in\mathbb{R}^{2+n}} c^T d$ Subject to $Ad \leq a$.

Now considering, (1) we define since $z_i := y_i - a^T x_i - b$

then it follows that $\min_{\substack{a\in\mathbb{R}^n \\ b\in\mathbb{R} \\ z\in\mathbb{R}^n}} \frac{1}{n}\sum z_i$ s.t. $z_i \geq y_i - a^T x_i - b$, $z_i \geq a^T x_i + b - y_i$

Now we associate λ_i with the ith constant, thus we have the Langragian is simply

$L(x) = \lambda_1^T (y_i - a^T x_i - b) + \lambda_2^T (a^T x_i + b - y_i) = \mu^T x$

grouping by x, b, y.

$(c^T - (\lambda_2^T - \lambda_1)^T a - \mu^T) x + (\lambda_2 - \lambda_1)^T b + (\lambda_2 - \lambda_1)^T y = L$

$\frac{dL}{dx} = 0 \Rightarrow g(\lambda_1, \lambda_2, \mu) = \begin{cases} (\lambda_2 - \lambda_1)^T (b + y) & \Leftrightarrow c^T - (\lambda_2 - \lambda_1)^T a \leq 0 \\ -\infty & \text{otherwise.} \end{cases}$

Therefore, the primal is

$\min_{\lambda_1, \lambda_2} \sum (b+y)^T (\lambda_2 - \lambda_1)$ s.t. $\sum_i a_i^T (\lambda_2 - \lambda_1) \leq c^T \; \forall i$

$\Rightarrow \min_{d\in\mathbb{R}^{2+n}} c^T d$ s.t $Ad \leq b$

$d = (\lambda_1 - \lambda_2)$

$c = (b+y)^T$

c. Repeating the steps you went through to find expression of L as a function of $q, w,$ and λ,