Project 1

Name: Shrey Patel

1) Loading the dataset of handwritten digits collected by USPS

```
load usps_all.mat

% data. Dimension = 256x1100x10
% 256 pixels, 1100 instances of 10 digits(1,2,...0)
```

2) Studying the role of the split of the training vs testing data for classification

N=1000

```
tic
% Dividing each class into two sets:
% the training set consisting of N examples for each of the digits 0 through 9,
% and the testing set consisting of 1100-N examples of each of the digits 0 through 9,
% where N ranges from 100 to 1000.
% Propose and describe a selection algorithm for choosing N out of 1100 images
% for any integer value of N.

% 1000/100 split
% Algorithm: Simply picking the first 1000 as training and the rest as
%       test, since I'd like to use maximum data for training.
N = 1000
```

```
N = 1000
```

```

% first 1000 intances of each digit go into training set
train_set = zeros(256,N,10);
for digit = 1:10
    train_set(:,:,digit) = data(:,1:N,digit);
end
% converting 256x1000x10 to 256x10000 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to1000=>digit1; 1001to2000=>digit2
% and so on.
train_set = reshape(train_set,256,[]);

% last 100 intances of each digit go into test set
test_set = zeros(256,1100-N,10);
for digit = 1:10
    test_set(:,:,digit) = data(:,N+1:1100,digit);
end
% converting 256x100x10 to 256x1000 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to100=>digit1; 101to200=>digit2
% and so on.
test_set = reshape(test_set,256,[]);

% Assessing the scope of the computational work:
% total digits (256x1 column vectors) in the training set = 1000x10 = 10000
% total digits (256x1 column vectors) that need to be tested/classified = 100x10 = 1000
% k-Nearest Neighbor classification using the knnsearch function
% k = 20
% metric: Euclidean distance (L2)
    % ---we first transpose our train and test matrices so that each row in train
    % is an 'observation/digit' and each row in test will be our 'query
    % points'. We'll find the distance from each query point (in test) to each
    % observation in train. 1000x10000 = 10^7 computations.
train_set = train_set';
test_set = test_set';
%tic
[Idx, euc_dist] = knnsearch(train_set,test_set,'K',20, ...
                            'Distance','euclidean');
%toc
Idx;
euc_dist = round(euc_dist);
```

```
labels = zeros(3,10);
% creating labels (w.r.t. training set)
labels(1,:) = 1:10;
labels(2,:) = 1:1000:10000;
labels(3,:) = 1000:1000:10000;

% success rate
euc_Idx(:,:,1) = Idx(1:100,:);
euc_Idx(:,:,2) = Idx(101:200,:);
euc_Idx(:,:,3) = Idx(201:300,:);
euc_Idx(:,:,4) = Idx(301:400,:);
euc_Idx(:,:,5) = Idx(401:500,:);
euc_Idx(:,:,6) = Idx(501:600,:);
euc_Idx(:,:,7) = Idx(601:700,:);
euc_Idx(:,:,8) = Idx(701:800,:);
euc_Idx(:,:,9) = Idx(801:900,:);
euc_Idx(:,:,10) = Idx(901:1000,:);
% euc_Idx(:,:,4)  % check
count = 0;
temp = zeros(1,10);
for i = 1:10  % for ith digit
    for m = 1:100  % for each row
        for n = 1:20 % for each column (nearest neighbor)
           if (euc_Idx(m,n,i)>(i*1000) || euc_Idx(m,n,i)<=((i-1)*1000)) % incorrect
               i;
               m;
               n;
               euc_Idx(m,n,i);
               count = count + 1;
               temp(1,i) = temp(1,i) + 1;
           end
        end
    end
    i;
    temp(1,i); % incorrect nearest neighbor for each digit
    count;
```

```
end
count            % total number of incorrect nearest neighbors
```

```
count = 2632
```

```
%sum(temp,2)    % must be equal to count
idv_success_rate = 1-(temp/2000) % success% for each digit
```

```
idv_success_rate = 1×10
    0.9970    0.8115    0.9140    0.7725    0.8850    0.9075    0.9245    0.7245 ⋯
```

```
error_rate = count/20000
```

```
error_rate = 0.1316
```

```
success_rate = 1-error_rate
```

```
success_rate = 0.8684
```

```
toc
```

```
Elapsed time is 2.871164 seconds.
```

N = 1050 (more optimal in terms of computations)

```
tic
% Dividing each class into two sets:
% the training set consisting of N examples for each of the digits 0 through 9,
% and the testing set consisting of 1100-N examples of each of the digits 0 through 9.
% Propose and describe a selection algorithm for choosing N out of 1100 images
% for any integer value of N.

% 1050/50 split
% Algorithm: Simply picking the first 1050 as training and the rest as
%       test.
N = 1050
```

```
N = 1050
```

```

% first 1050 intances of each digit go into training set
train_set = zeros(256,N,10);
for digit = 1:10
    train_set(:,:,digit) = data(:,1:N,digit);
end
% converting 256x1050x10 to 256x10500 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to1050=>digit1; 1051to2100=>digit2
% and so on.
train_set = reshape(train_set,256,[]);

% last 50 intances of each digit go into test set
test_set = zeros(256,1100-N,10);
for digit = 1:10
    test_set(:,:,digit) = data(:,N+1:1100,digit);
```

```
end
% converting 256x50x10 to 256x500 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to50=>digit1; 51to100=>digit2
% and so on.
test_set = reshape(test_set,256,[]);

% Assessing the scope of the computational work:
% total digits (256x1 column vectors) in the training set = 1050x10 = 10500
% total digits (256x1 column vectors) that need to be tested/classified = 50x10 = 500

% k-Nearest Neighbor classification using the knnsearch function
% k = 20
% metric: Euclidean distance (L2)
    % ---we first transpose our train and test matrices so that each row in train
    % is an 'observation/digit' and each row in test will be our 'query
    % points'. We'll find the distance from each query point (in test) to each
    % observation in train. 10500x500 = 5.25x10^6 computations.
train_set = train_set';
test_set = test_set';
%tic
[Idx, euc_dist] = knnsearch(train_set,test_set,'K',20, ...
                            'Distance','euclidean');
%toc
Idx;
euc_dist = round(euc_dist);
```

```
labels = zeros(3,10);
% creating labels for testing purposes
labels(1,:) = 1:10;
labels(2,:) = 1:1050:10500;
labels(3,:) = 1050:1050:10500;

% success rate
euc_Idx2(:,:,1) = Idx(1:50,:);
euc_Idx2(:,:,2) = Idx(51:100,:);
euc_Idx2(:,:,3) = Idx(101:150,:);
euc_Idx2(:,:,4) = Idx(151:200,:);
euc_Idx2(:,:,5) = Idx(201:250,:);
euc_Idx2(:,:,6) = Idx(251:300,:);
euc_Idx2(:,:,7) = Idx(301:350,:);
euc_Idx2(:,:,8) = Idx(351:400,:);
euc_Idx2(:,:,9) = Idx(401:450,:);
euc_Idx2(:,:,10) = Idx(451:500,:);
%euc_Idx2(:,:,4)  % check
count = 0;
temp = zeros(1,10);
for i = 1:10  % for ith digit
    for m = 1:50  % for each row
        for n = 1:20 % for each column (nearest neighbor)
            % incorrect nearest neighbors
           if (euc_Idx2(m,n,i)>(i*1050) || euc_Idx2(m,n,i)<(1+((i-1)*1050)))
               i;
               m;
               n;
               euc_Idx2(m,n,i);
               count = count + 1;
               temp(1,i) = temp(1,i) + 1;
           end
        end
    end
    i;
    temp(1,i); % incorrect nearest neighbor for each digit
    count;
end
count           % total number of incorrect nearest neighbors
```

```
count = 1420
```

```
% sum(temp,2);    % must be equal to count
idv_success_rate = 1-(temp/1000) % success% for each digit
```

```
idv_success_rate = 1×10
    0.9950    0.7750    0.9320    0.7630    0.8790    0.8770    0.9380    0.6930 ⋯
```

```
error_rate = count/10000
```

```
error_rate = 0.1420
```

```
success_rate = 1-error_rate
```

```
success_rate = 0.8580
```

```
toc
```

```
Elapsed time is 1.659660 seconds.
```

3) Studying the role of the structure of the split of the training vs testing data for classification

```
% New selection algorithm to pick our training set with N=1050, which
% was more optimal than N=1000
% I've decided to pick the 1050 nearest neighbors to the mean vector of
% each digit as the train-set. In other words, 50 farthest instances from the mean vector of
% a digit will go into the test-set. This is a good way to see how our
% model would perform in the worst case scenario

mean_data = zeros(256,10);
for i = 1:10
    xsize = size(data(:,:,i));
    mean_data(:,i) = sum(data(:,:,i),2)/xsize(2);
end
mean_data; % 256x10 dataset with each column being the mean vector of each digit class
```

```
% splitting
new_idx = zeros(10,1100);
for i = 1:10
    [Idx, euc_dist] = knnsearch(double(data(:,:,i)'),mean_data(:,i)','K',1100, ...
                            'Distance','euclidean');
    new_idx(i,:) = Idx;
end
new_idx;

% 256x1050x10 training sets
training_set = zeros(256,1050,10);
for k = 1:10
    training_set(:,:,k) = data(:,new_idx(k,1:1050),k);
end
%training_set(:,:,1);

% 256x50x10 testing sets
testing_set = zeros(256,50,10);
for k = 1:10
    testing_set(:,:,k) = data(:,new_idx(k,1051:1100),k);
end
%testing_set(:,:,1);
```

```
size(training_set);
size(testing_set);
% check
figure;
n = cell(1,50);
for k = 1:50
    k; % printing k or the digit
    reshaped_data = reshape(training_set(:,k,6),[16 16]); % reshape to 16x16
    n(:,k) = {reshaped_data};
end
% display
% imshow(imtile(n, "GridSize",[10 5]));
```

Finding the nearest neighbors and the global success rate

```
tic

training_set = reshape(training_set,256,[]);
testing_set = reshape(testing_set,256,[]);

training_set = training_set';
testing_set = testing_set';
%tic
[Idx, euc_dist] = knnsearch(training_set,testing_set,'K',20, ...
                            'Distance','euclidean');
%toc
Idx;
euc_dist = round(euc_dist);
```

```
% success rate
euc_Idx2(:,:,1) = Idx(1:50,:);
euc_Idx2(:,:,2) = Idx(51:100,:);
euc_Idx2(:,:,3) = Idx(101:150,:);
```

```
euc_Idx2(:,:,4) = Idx(151:200,:);
euc_Idx2(:,:,5) = Idx(201:250,:);
euc_Idx2(:,:,6) = Idx(251:300,:);
euc_Idx2(:,:,7) = Idx(301:350,:);
euc_Idx2(:,:,8) = Idx(351:400,:);
euc_Idx2(:,:,9) = Idx(401:450,:);
euc_Idx2(:,:,10) = Idx(451:500,:);
%euc_Idx2(:,:,4)  % check
count = 0;
temp = zeros(1,10);
for i = 1:10  % for ith digit
    for m = 1:50  % for each row
        for n = 1:20 % for each column (nearest neighbor)
            % incorrect nearest neighbors
           if (euc_Idx2(m,n,i)>(i*1050) || euc_Idx2(m,n,i)<(1+((i-1)*1050)))
               i;
               m;
               n;
               euc_Idx2(m,n,i);
               count = count + 1;
               temp(1,i) = temp(1,i) + 1;
           end
        end
    end
    i;
    temp(1,i); % incorrect nearest neighbor for each digit
    count;
end
count            % total number of incorrect nearest neighbors
```

```
count = 2654
```

```
%sum(temp,2);    % must be equal to count
idv_success_rate = 1-(temp/1000) % success% for each digit
```

```
idv_success_rate = 1×10
    0.9750    0.7390    0.7070    0.7140    0.6420    0.8310    0.7160    0.4900 ⋯
```

```
error_rate = count/10000
```

```
error_rate = 0.2654
```

```
success_rate = 1-error_rate
```

```
success_rate = 0.7346
```

```
toc
```

```
Elapsed time is 1.584654 seconds.
```

4) Analyzing the role of the metric in the classification process using the optimal value of N and the best training set selection method. Euclidean vs Manhattan distances.

Manhattan distance metric for N=1050:

```
tic
% Dividing each class into two sets:
% the training set consisting of N examples for each of the digits 0 through 9,
% and the testing set consisting of 1100-N examples of each of the digits 0 through 9.
% Propose and describe a selection algorithm for choosing N out of 1100 images
% for any integer value of N.

% 1050/50 split
% Algorithm: Simply picking the first 1050 as training and the rest as
%       test.
N = 1050
```

```
N = 1050
```

```
% first 1050 intances of each digit go into training set
train_set = zeros(256,N,10);
for digit = 1:10
    train_set(:,:,digit) = data(:,1:N,digit);
end
% converting 256x1050x10 to 256x10500 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to1050=>digit1; 1051to2100=>digit2
% and so on.
train_set = reshape(train_set,256,[]);

% last 50 intances of each digit go into test set
test_set = zeros(256,1100-N,10);
for digit = 1:10
    test_set(:,:,digit) = data(:,N+1:1100,digit);
end
% converting 256x50x10 to 256x500 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to50=>digit1; 51to100=>digit2
% and so on.
test_set = reshape(test_set,256,[]);

% Assessing the scope of the computational work:
% total digits (256x1 column vectors) in the training set = 1050x10 = 10500
% total digits (256x1 column vectors) that need to be tested/classified = 50x10 = 500

% k-Nearest Neighbor classification using the knnsearch function
% k = 20
% metric: Euclidean distance (L2)
    % ---we first transpose our train and test matrices so that each row in train
    % is an 'observation/digit' and each row in test will be our 'query
    % points'. We'll find the distance from each query point (in test) to each
    % observation in train. 10500x500 = 5.25x10^6 computations.
train_set = train_set';
test_set = test_set';
%tic
[Idx, euc_dist] = knnsearch(train_set,test_set,'K',20, ...
                            'Distance','cityblock');
%toc
Idx;
euc_dist = round(euc_dist);
```

```
labels = zeros(3,10);
% creating labels for testing purposes
labels(1,:) = 1:10;
labels(2,:) = 1:1050:10500;
labels(3,:) = 1050:1050:10500;

% success rate
euc_Idx2(:,:,1) = Idx(1:50,:);
euc_Idx2(:,:,2) = Idx(51:100,:);
euc_Idx2(:,:,3) = Idx(101:150,:);
euc_Idx2(:,:,4) = Idx(151:200,:);
euc_Idx2(:,:,5) = Idx(201:250,:);
euc_Idx2(:,:,6) = Idx(251:300,:);
euc_Idx2(:,:,7) = Idx(301:350,:);
euc_Idx2(:,:,8) = Idx(351:400,:);
euc_Idx2(:,:,9) = Idx(401:450,:);
euc_Idx2(:,:,10) = Idx(451:500,:);
%euc_Idx2(:,:,4)  % check
count = 0;
temp = zeros(1,10);
for i = 1:10  % for ith digit
    for m = 1:50  % for each row
        for n = 1:20 % for each column (nearest neighbor)
            % incorrect nearest neighbors
           if (euc_Idx2(m,n,i)>(i*1050) || euc_Idx2(m,n,i)<(1+((i-1)*1050)))
               i;
               m;
               n;
               euc_Idx2(m,n,i);
               count = count + 1;
               temp(1,i) = temp(1,i) + 1;
           end
        end
    end
    i;
    temp(1,i); % incorrect nearest neighbor for each digit
    count;
end
count           % total number of incorrect nearest neighbors
```

```
count = 1594
```

```
% sum(temp,2);    % must be equal to count
idv_success_rate = 1-(temp/1000) % success% for each digit
```

```
idv_success_rate = 1×10
    0.9980    0.7590    0.9180    0.7560    0.8800    0.8650    0.9100    0.6350 ···
```

```
error_rate = count/10000
```

```
error_rate = 0.1594
```

```
success_rate = 1-error_rate
success_rate = 0.8406
```

```
toc
```

```
Elapsed time is 1.603741 seconds.
```

5) Individual success rates for each digit

Already calculated in part 2, 3, and 4

-------------------- *Further analysis (extra)* ------------------------

n = 1000 with Manhattan dist. metric.

```
tic
% Dividing each class into two sets:
% the training set consisting of N examples for each of the digits 0 through 9,
% and the testing set consisting of 1100-N examples of each of the digits 0 through 9,
% where N ranges from 100 to 1000.
% Propose and describe a selection algorithm for choosing N out of 1100 images
% for any integer value of N.

% 1000/100 split
% Algorithm: Simply picking the first 1000 as training and the rest as
%       test, since I'd like to use maximum data for training.
N = 1000
```

```
N = 1000
```

```

% first 1000 intances of each digit go into training set
train_set = zeros(256,N,10);
for digit = 1:10
    train_set(:,:,digit) = data(:,1:N,digit);
end
% converting 256x1000x10 to 256x10000 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to1000=>digit1; 1001to2000=>digit2
% and so on.
train_set = reshape(train_set,256,[]);

% last 100 intances of each digit go into test set
test_set = zeros(256,1100-N,10);
for digit = 1:10
    test_set(:,:,digit) = data(:,N+1:1100,digit);
end
% converting 256x100x10 to 256x1000 (concatenating all the digits into
% single 2d matrix).. Hence, columns 1to100=>digit1; 101to200=>digit2
% and so on.
test_set = reshape(test_set,256,[]);
```

```
% Assessing the scope of the computational work:
% total digits (256x1 column vectors) in the training set = 1000x10 = 10000
% total digits (256x1 column vectors) that need to be tested/classified = 100x10 = 1000

% k-Nearest Neighbor classification using the knnsearch function
% k = 20
% metric: Euclidean distance (L2)
    % ---we first transpose our train and test matrices so that each row in train
    % is an 'observation/digit' and each row in test will be our 'query
    % points'. We'll find the distance from each query point (in test) to each
    % observation in train. 1000x10000 = 10^7 computations.
train_set = train_set';
test_set = test_set';
%tic
[Idx, euc_dist] = knnsearch(train_set,test_set,'K',20, ...
                            'Distance','cityblock');
%toc
Idx;
euc_dist = round(euc_dist);
```

```
labels = zeros(3,10);
% creating labels (w.r.t. training set)
labels(1,:) = 1:10;
labels(2,:) = 1:1000:10000;
labels(3,:) = 1000:1000:10000;

% success rate
euc_Idx(:,:,1) = Idx(1:100,:);
euc_Idx(:,:,2) = Idx(101:200,:);
euc_Idx(:,:,3) = Idx(201:300,:);
euc_Idx(:,:,4) = Idx(301:400,:);
euc_Idx(:,:,5) = Idx(401:500,:);
euc_Idx(:,:,6) = Idx(501:600,:);
euc_Idx(:,:,7) = Idx(601:700,:);
euc_Idx(:,:,8) = Idx(701:800,:);
euc_Idx(:,:,9) = Idx(801:900,:);
euc_Idx(:,:,10) = Idx(901:1000,:);
% euc_Idx(:,:,4)  % check
count = 0;
temp = zeros(1,10);
for i = 1:10  % for ith digit
    for m = 1:100  % for each row
        for n = 1:20 % for each column (nearest neighbor)
           if (euc_Idx(m,n,i)>(i*1000) || euc_Idx(m,n,i)<=((i-1)*1000)) % incorrect
               i;
               m;
               n;
               euc_Idx(m,n,i);
               count = count + 1;
               temp(1,i) = temp(1,i) + 1;
           end
        end
    end
    i;
    temp(1,i); % incorrect nearest neighbor for each digit
    count;
end
count           % total number of incorrect nearest neighbors
```

```
count = 2975
```

```
% sum(temp,2);    % must be equal to count
idv_success_rate = 1-(temp/2000) % success% for each digit
```

```
idv_success_rate = 1×10
    0.9980    0.7945    0.9080    0.7525    0.8830    0.8990    0.9095    0.6740 ⋯
```

```
error_rate = count/20000
```

```
error_rate = 0.1488
```

```
success_rate = 1-error_rate
```

```
success_rate = 0.8513
```

```
toc
```

```
Elapsed time is 2.713110 seconds.
```