Table of Contents

This code evaluates the test set.

```
% ** Important.  This script requires that:
% 1)'centroid_labels' be established in the workspace
% AND
% 2)'centroids' be established in the workspace
% AND
% 3)'test' be established in the workspace


% IMPORTANT!!:
% You should save 1) and 2) in a file named 'classifierdata.mat' as part of
% your submission.

predictions = zeros(200,1);          %Column vectors of 200
all_distances = zeros(200,1);
%outliers = zeros(200,1);
% loop through the test set, figure out the predicted number
for i = 1:200
testing_vector=test(i,:);
% Extract the centroid that is closest to the test image
[prediction_index,
vec_distance]=assign_vector_to_centroid(testing_vector,centroids);

predictions(i) = centroid_labels(prediction_index);
all_distances(i) = vec_distance;

end
```

DESIGN AND IMPLEMENT A STRATEGY TO SET THE outliers VECTOR

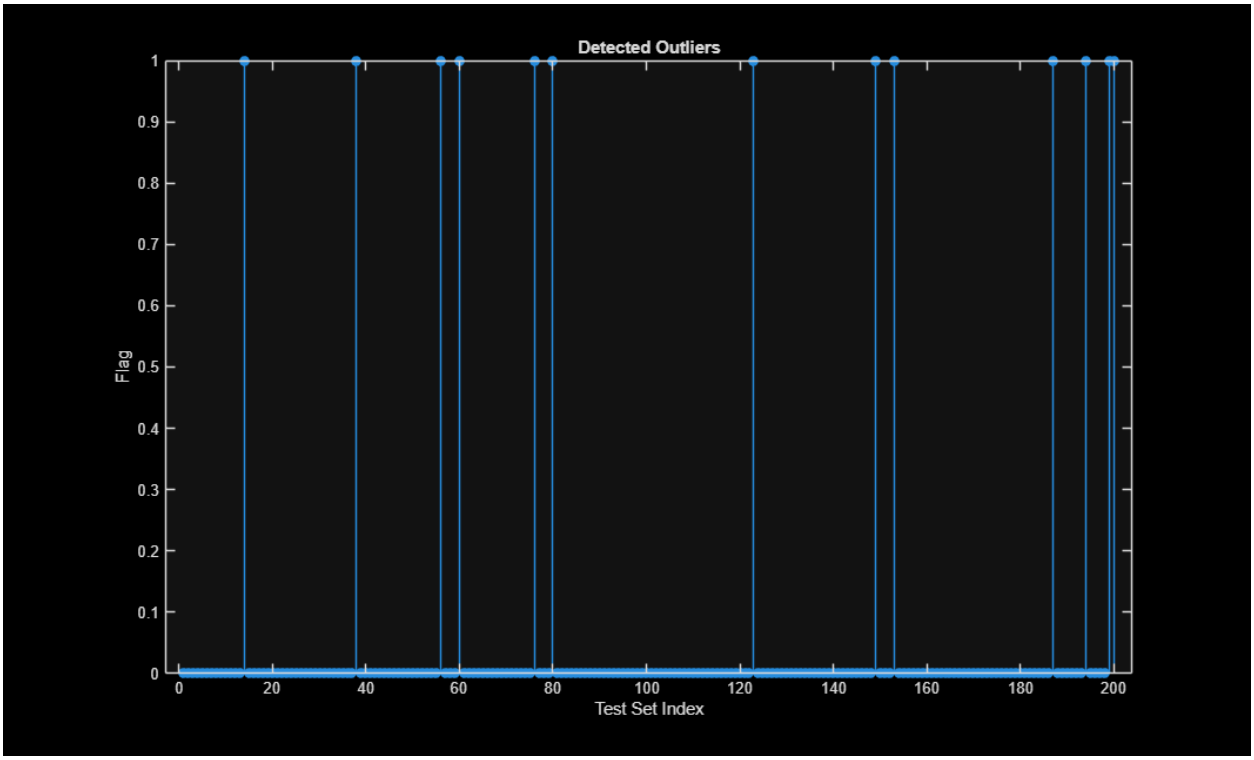
outliers(i) should be set to 1 if the i^th entry is an outlier otherwise, outliers(i) should be 0 FILL IN

```
threshold = mean(all_distances) + 1.6*std(all_distances);
outliers = all_distances > threshold;
```

MAKE A STEM PLOT OF THE OUTLIER FLAG

```
figure;
stem(outliers,'filled');
```

```
xlabel('Test Set Index');
ylabel('Flag');
title('Detected Outliers');
```



The following plots the correct and incorrect predictions

Make sure you understand how this plot is constructed

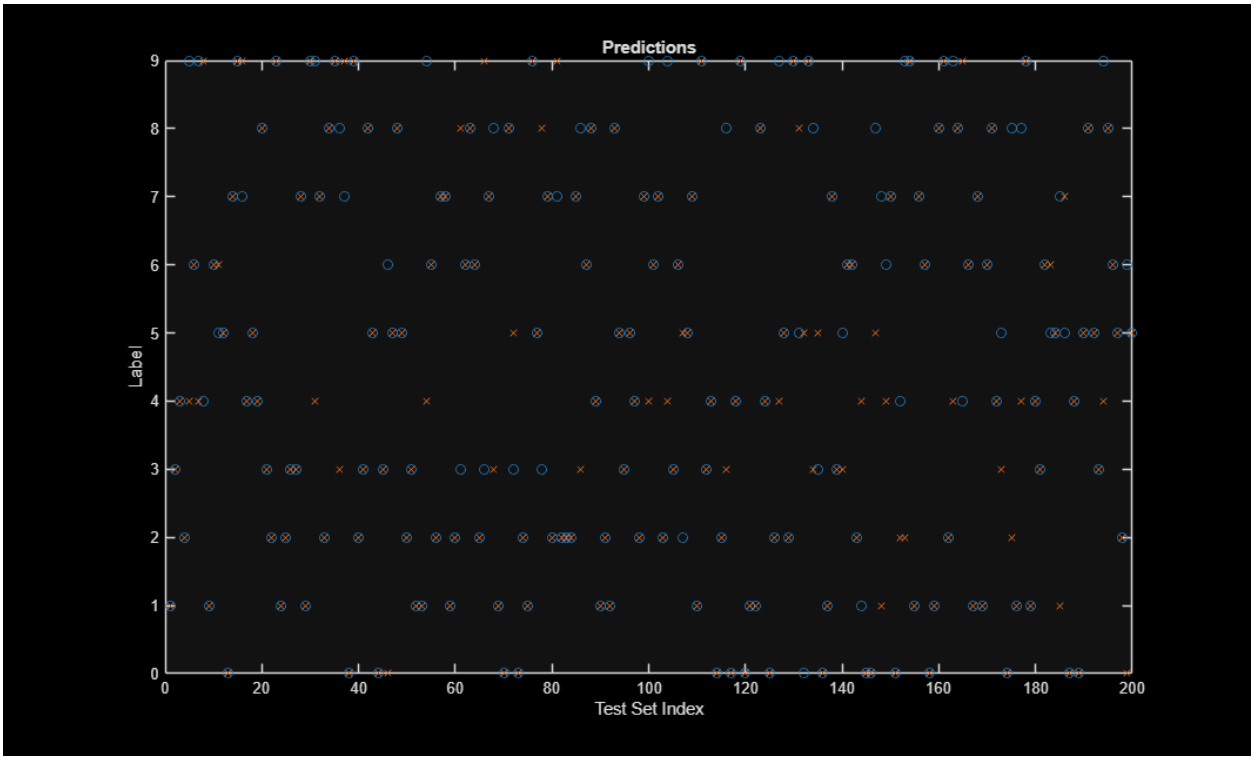
```
figure;
plot(correctlabels,'o');
hold on;
plot(predictions,'x');
title('Predictions');
xlabel('Test Set Index');
ylabel('Label');

accurate = zeros(200,1);
for i=1:200
    accurate(i) = (correctlabels(i) == predictions(i));
end

percentAccuracy = (norm(accurate)^2)/size(accurate,1)


percentAccuracy =
```

0.7850



The following line provides the number of instances where and entry in correctlabel is

equatl to the corresponding entry in prediction However, remember that some of these are outliers

```
sum(correctlabels==predictions)
```

```
function [index, vec_distance] = assign_vector_to_centroid(data,centroids)
    data_vector = reshape(data(1:784), 1, 784);

    comparison = 10000000000000;
    index = 1;

    for i = 1:size(centroids,1)          %goes from 1 to number of rows of
centroids (aka the random images)
        centroid_vector = centroids(i,1:784);
        current_distance = norm(data_vector - centroid_vector)^2;

        if current_distance < comparison
            comparison = current_distance;
            index = i;
        end
    end

    % Return the index and the squared distance
```

```
    vec_distance = comparison;
end

save('classifierdata.mat',"centroid_labels","centroids")
```

```
ans =

   157
```

Published with MATLAB® R2025a