Programming Ex.7

ML:Programming Exercise 7:K-Means Clustering and PCA

Debugging Tip

The submit script, for all the programming assignments, does not report the line number and location of the error when it crashes. The follow method can be used to make it do so which makes debugging easier.

Open ex7/lib/submitWithConfiguration.m and replace line:

```
fprintf('!! Please try again later.\n');

```

(around 28) with:

```
fprintf('Error from file:%s\nFunction:%s\nOn line:%d\n', e.stack(1,1).file,e.stack(1,1).name, e.stack(1,1).line );

```

That top line says '!! Please try again later' on crash, instead of that, the bottom line will give the location and line number of the error. This change can be applied to all the programming assignments.

Workaround for problem in plotting routine

{CTA Note: This problem only effects certain versions of Octave} after completion of the computecentroids.m function, i ran into the following problem:

coursera

```
K-Means iteration 1/10...
error: __scatter__: A(I): index out of bounds; value 4 out of bound 3
error: called from:
error:   /Applications/Octave.app/Contents/Resources/share/octave/3.4.0/m/plot/private/__scatter__.m at line 159, column 13
error:   /Applications/Octave.app/Contents/Resources/share/octave/3.4.0/m/plot/scatter.m at line 71, column 11
error:  ?/ex7/mlclass-ex7/plotDataPoints.m at line 12, column 1
error:  ?/ex7/mlclass-ex7/plotProgresskMeans.m at line 11, column 1
error:  ?/ex7/mlclass-ex7/runkMeans.m at line 48, column 9
error:  ?/ex7/mlclass-ex7/ex7.m at line 92, column 19

```

i don't think it is caused by my solution, and found a workaround by modifying the plotDataPoints.m as follows

```
% use idx directly. It will index into the default color map.
% scatter(X(:,1), X(:,2), 15, colors);
 scatter(X(:,1), X(:,2), 15, idx);

```

The issue is a bug in the scatter() function in certain versions of Octave.

findClosestCentroids() issue with regards to the grader

If two centroids have identical distances, the submit grader wants you to select the one with the lowest index value. This situation arises when running ex7_pca.m - some of the image pixels have the same minimum distance to more than one centroid. This restriction is most easily accommodated by using the min() function to find the centroid with the minimum distance. Students have found that using the find() function does not result in the answer the grader prefers.

Selecting the initial centroids - an additional consideration

This issue was omitted from the lectures. When the initial centroids are selected, be sure that they are each unique. For example, if using K-Means to compress an image, each of the initial centroids should represent a unique color. If two initial centroids were the exact same color, then you would effectively have K-1 centroids, not K.

Using the kMeansInitCentroids() method as given in ex7.pdf, an experiment on the "bird_small.mat" data set shows that approximately 5 tries in 10,000 will result in duplicate centroids. The method given in ex7.pdf only selects unique members of the training set as the centroids - it does not verify that they are not duplicate values.

One method for preventing duplicate centroids would be as follows:

- Randomly select a set of K training examples as the initial centroids.
- Use the unique(centroids, 'rows') function to get a matrix of all of the unique centroid values.
- If the number of unique rows is not equal to K, then re-select a new set of initial centroids.

Another method would be to prevent any duplicates at all by using the unique function on the training examples (unique(X, 'rows')) **before**randomly selecting the initial centroids.

Fully vectorizing findClosestCentroids()

It is possible to fully vectorize this function by using 3D arrays for the training examples and the centroids.

Tip 1: To transform 2D arrays to 3D, you can use permute with an extra dimension index. For example, you can transform a $m \times n$ (2D) matrix **A2**to a $m \times 1 \times n$ (3D) array **A3** using A3 = permute(A2, [1 3 2]);

Tip 2: Instead of using repmat to "expand" a matrix for binary operations, it is usually faster to use bsxfun.

Errata in projectData.m

Make the following change in the "Instructions" section:

```
%      projection_k = x' * U(:, 1:k);

```

The "1:k" portion was missing the "1:" part.