ex7 tutorial

findClosestCentroids()

Tom MosherMentorWeek 8 · 2 years ago · Edited

This tutorial gives a method for findClosestCentroids() by iterating through the centroids. This runs considerably faster than looping through the training examples.

- Create a "distance" matrix of size (m x K) and initialize it to all zeros. 'm' is the number of training examples, K is the number of centroids.
- Use a for-loop over the 1:K centroids.
- Inside this loop, create a column vector of the distance from each training example to that centroid, and store it as a column of the distance matrix. One method is to use the bsxfun) function and the sum() function to calculate the sum of the squares of the differences between each row in the X matrix and a centroid.
- When the for-loop ends, you'll have a matrix of centroid distances.
- Then return idx as the vector of the indexes of the locations with the minimum distance. The result is a vector of size (m x 1) with the indexes of the closest centroids.

==========

Additional implementation tips on how to use bsxfun():

bsxfun() means "broadcast function". It applies whatever function you specify (such as @minus) between the 1st argument (a matrix) and the second argument (typically a row vector).

Here's an example you can enter in your console:

```
1
2
```

```
Q = magic(3)
D = bsxfun(@minus, Q, [1 2 3])
```

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

It returns a value that is the same size as the first argument.

As implemented for this function, each row in the matrix returned by bsxfun() is the differences between that row of X and each of the features of one of the centroids.

Then, for each row, you compute the sum of the squares of those differences.

Here is code that does it.

sum(D.^2,2)

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
Note that if you leave off the ',2' part, then sum() will sum over the rows, and you'll instead get a (1xn) vector. Using ',2' tells sum() to work over the columns, so you'll get a (m x 1) result.
We repeat this process for each of the centroids. Once we have the squared distance between each example and each centroid, we then return the index where the min() value was found for each example.

computeCentroids()

Tom MosherMentorWeek 8 · 10 months ago
The method in this tutorial iterates over the centroids.
The function computeCentroids is called with parameters "X, "idx" and "K".
"K" is the number of centroids.
"idx" is a vector with one entry for each example in "X", which tells you which centroid each example is assigned to. The values range from 1 to K, so you will need a for-loop over that range.
You can get a selection of all of the indexes for each centroid with:
sel = find(idx == i) % where i ranges from 1 to K
Now we want to compute the mean of all these selected examples, and assign it as the new centroid value:
centroids(i,:) = mean(X(sel,:))
(Note that this method breaks if sel is a null vector (i.e. if no examples are assigned to centroid 'i'. In that case, probably should only update centroid(i,:) if sel is a non-null vector.
Repeat this procedure all each centroid. The function returns the computed centroid values.

pca

Tom MosherMentorAssignment: K-Means Clustering and PCA · 2 years ago · Edited
These are tutorials for all three functions in the ex7 PCA exercise. All of these functions have a vectorized implementation in one or two lines of code.
==========
pca()
Compute the transpose of X times X, scale by 1/m, and use the svd() function to return the U, S, and V matrices.

X is size (m x n), so "X transpose X" and U are both size (n x n)
(note: the feature matrix X has already been normalized, see ex7_pca.m)
==========
projectData()
Errata:
In projectData.m, make the following change in the Instructions section:
% projection_k = x' * U(:, **1:k**);

Return Z, the product of X and the first 'K' columns of U.
X is size (m x n), and the portion of U is (n x K). Z is size (m x K).
==========
recoverData()
Return X_rec, the product of Z and the first 'K' columns of U.
Z is size (m x K), and the portion of U is (n x K). Use transposition so the result X_rec is size (m x n).
==========