Clustering

Tuesday, April 16, 2019 3:00 PM

Last Week:

For plotting the decision boundary in multiple dimensions.
You could take the mean, like we did, or you could take the sum or median. It depends on what you want to do.

Completeness and Purity should be as good or better if you add another dimension.
By adding more information and increasing the dimensionality you're breaking the degeneracy.

Clustering

Grouping Objects into Sets

Unsupervised - No training set, No labeled data.

No prior information about expected number of clusters.

Good as an exploratory tool for your data.

K-means Algorithm

K : number of clusters, C_k
N_k : number of points in C_k

Chooses a centroid for each cluster, μ_k, that minimizes **the inertia (sum of squares)**

$$\sum_{k=1}^{K} \sum_{i \epsilon C_k} ||x_i - \mu_k||^2$$

$$\mu_k = \frac{1}{N_k} \sum_{i \epsilon c_k} x_i$$

For each cluster look at the points within each cluster and determine the distance from the centroid. Minimizing this is related to the moment of inertia of your data.

Steps in k-means

1. Choose number of clusters
2. Choose your initial centroids
    - In general you choose random points within the limits of the data set.
    - By this I mean within the parameter space, not like min max of your scatter points.
3. Assign each point in the sample to the centroid that is closest to it.
4. Calculate the mean of all the points assigned to each centroid.
5. Calculate the difference in position for the mean in step 3 and the current centroid. If it is greater than some threshold, make the means the new centroids.
6. Repeat 2 through 4 until the if statement lets you out.

Note that the centroids can get stuck in local minimum.
There's merit in running this multiple times with different starting positions.

Concerns

- Final centroids dependent on initial centroids.
    - Run with different initial conditions.
- Good for convex/regular data, not elongated/irregular data
- It looks like blobby not gaussian data.
- It cant figure out how many centroids to choose on it's own.

Code

Kmeans = kmeans(n_clusters = 3, init = "random", n_init=1)
Init: method for initialization,
kmeans++ : speeds up convergence.
Random: chooses random spots within data.
N_init: how many times to try with different initial conditions.

kmeans.fit(X)
kmeans.predict(X)
Centroids = kmeants.cluster_centers_
dimensions: number of clusters, x,y coordinates of each centroid.

Ways to Improve:

Maybe use a distance other than euclidian space.
Penalize gaps.

Review:

Vstack?
Kfold cross validation

Resource:

https://www.saedsayad.com/clustering_kmeans.htm