K-Means Gotchas!

MOD29

It's easy to understand the K-Means algorithm, and extremely fast to execute. So fast that it's often ran several times over as you saw earlier. Since each successive run of isn't dependent on the results of earlier runs, the execution process lends itself to parallelization, each centroid seeding trial being ran independently. If the clustering job at hand is still taking too long, SciKit-Learn's MiniBatchKMeans further optimizes the process for you.

Considering how basic of an algorithm it is, K-Means performs pretty well, and its implementation is the basis for a few more advanced clustering algorithms, such as learning vector quantization and Gaussian mixture. Having a solid understanding of K-Means will help you understand those better when you study them.

K-Means is only really suitable when you have a good estimate of the number clusters that exist in your unlabeled data. There are many estimation techniques for approximating the correct number of clusters, but you'll have to get that number before running K-Means. Even if you *do* have the right number of clusters selected, the result produced by K-Means can vary depending on the initial centroid placement. So if you need the same results produced each time, your centroid seeding technique also needs to be able to reliably produce the same placement given the same data. Due to the centroid seed placement having so much of an effect on your clustering outcome, you have to be careful since it is possible to have centroids with only a single sample assigned to them, or even *no* samples assigned to them in the worst case scenario.

Two other key characteristics of K-Means are that it assumes your samples are length normalized, and as such, is sensitive to feature scaling. It also assumes that the cluster sizes are roughly spherical and similar; this way, the nearest centroid is always the correct assignment.