Gaussian Mixture Model estimation via Expectation-Maximization

Expectation-Maximization algorithm as been proved to be a really powerfull technique to estimate parameters values of a Gaussian Mixture model. The following document will presents the techniques used in my implementation and also the results with the provided data.

Initialization

The result of the Expectation Maximization is tightly bound to the **initial state**. The initial choice of the parameters for each components of the gaussian mixture, will have important consequences on the accuracy of the trained model. Because the log-likelihood is not most of the time a convex function, the algorithm can converge to some local-minimums. We discuss in this part what are the prossible options in order to improve the accuracy of the resulting Gaussian Mixture model.

Because no information are provided about the data set, the best way to initialize the model is to set the same weight and a identity matrix as variance-covariance matrix for each component of the mixture. However the mean of each component can be estimated before executing the EM algorithm.

Two ways of choosing the means will be describe:

- **Random**: Choose randomly K samples as initial means
- **K-Means**: Use K-Means for create K clusters of samples and use the centroid of each as initial means

However we have assumed so far that the number of components (K) for the gaussian mixture is already known. A simple scatter plot on the training data allow us to distinguish 4 components for each mixture. We will discuss in the next part how to estimate K.

K Estimation

Getting the most optimal number of cluster out of a data set is a well-know problem, an many algorithm already exist in order to solve it. In 2004, *Pham et al.* proposed a straight forward algorithm which has been implemented in this project. The algorithm simulates K-Means for several K and estimate the best K by comparing the improvement of the partitioning between two consecutive K.

This algorithm also return the centroids generated for the best K. Those centroids can be used to initialize the EM algorithm.

Results

In order to test the script, we train the model with 4000 labelled samples (`data/train.txt`), generating 2 gaussian mixtures models. The trained models are tested against 400 samples (`data/dev.txt`).

Training the model by using the **Random** initialization method is strait forward. The means execution time to train the model is *0.23 seconds*, however the affectation error-rate is ragging from *2.25%* to *10%*.

Using the **K-Means**, the error-rate is always *2.25%*, however it take *1.5 seconds* to run several time the K-Means algorithm. Because K-Means algorithm is also very sensitive to the initials centroids, the algorithm has to run several times in order to find the best cluster repartition. Using 1 time the K-Means algorithm is not enough to offer the EM good enough initial means values. K-Means should at least run 3 times to ensure that the centroids are good enough to provide a low error-rate.

Concerning the **K-Estimation** algorithm, more than 20 seconds are needed to find the right K value. But this time is related to the maximum K to test. Because the time complexity of K-Means is linearly bound with the number of K to test, by increasing the maximum K to check it will double the K-Estimation algorithm time.

Improvement

The result show that the minimum error rate on the test data is **2.25%**. This result can be explain by the fact that the component from each other mixture are close, and a sample generated by a mixture can by easily assigned to another.

Concerning the performance, the K-Means initialization method is the one which provides the best results. Furthermore, because the training of the **2 models can be done in parallel**, we can envision to reduce the training time by training each of the mixture independently in 2 theards. It is also possible to improve the K-Means by reducing the number of iteration before convergence. This can be done by doing a smart selection of the initials points for the K-Means by using for instance the **K-Means++** algorithm.