PRML Assignment III

Generation of Data

The training data is generated by ***createDataset*** in **data.py**. The ***createDataset*** function uses the means and covariations of Gaussian distributions initialized by ***initializeGaussianDistributions*** in **data.py** to generate samples which compile to Gaussian distributions. Command line options are provided for specifying the number of samples, the path of saving the dataset and corresponding labels, and the number, dimension and separation degree of Gaussian distributions.

Description of Models

GAUSSIAN MIXTURE MODEL

The ***GaussianMixtureModel*** is defined in **model.py** in **handout** directory. It contains an **K-means model** to initialize the means and covariations of gaussian distributions. When training the Gaussian mixture model on a certain dataset, the K-means model is first trained on the dataset, and Gaussian mixture model initialize its means, covariations and proportions of each Gaussian distributions according to the trained K-means model.

The ***fit*** method of ***GaussianMixtureModel*** provides **two ways** to train the Gaussian mixture model. The first is to **calculate the logarithm likelihood function and judge whether it converges**, the second is to **complete certain epochs of training**. Command line options are provided for specifying the training method. In both ways of training, I use the **Expectation Maximization Algorithm** to update the parameters of Gaussian mixture model. The **E-step** fixes parameters and calculates the posterior probability function of distributions. The **M-step** find parameters to maximize the posterior probability function. By applying the **Expectation Maximization Algorithm** iteratively, the posterior probability function will be maximized.

The ***plot*** method of ***GaussianMixtureModel*** will plot the dataset and ellipse representing distributions dynamically. Colours are randomly specified for each Gaussian distribution and command line options are provided for specifying whether or not to plot the training process of Gaussian mixture model.

K-MEANS MODEL

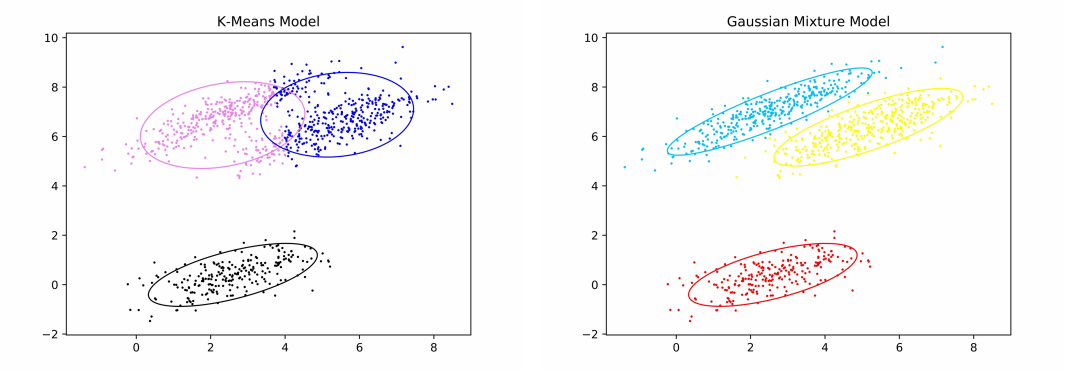
The ***kMeansModel*** is defined in **model.py** in **handout** directory. It uses **K-means++ algorithm** to choose the initial values of cluster centers. The workflow of **K-means++ algorithm** is to choose the first center randomly and add a new center whose distance to the choose centers is farthest to the class of chosen centers until k cluster enters are chosen.

The ***fit*** method of ***kMeansModel*** train the k-means model on certain dataset.

The ***plot*** method of ***kMeansModel*** will plot the dataset and ellipse representing distributions dynamically. Colours are randomly specified for each Gaussian distribution and command line options are provided for specifying whether or not to plot the training process of k-means model.

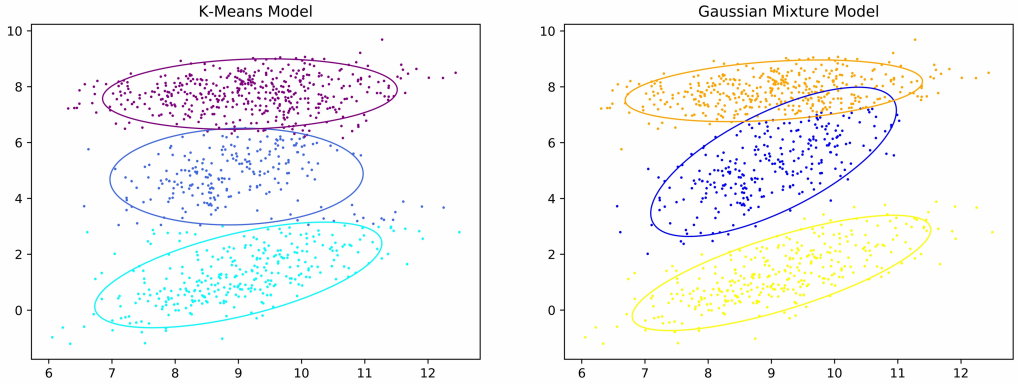
Performance of Models

Create a dataset containing 3 Gaussian distributions and train the Gaussian mixture model. The k-means model and Gaussian mixture model perform differently.



According to the picture, the k-means model assign some samples to the blue class, which evidently belongs to the pink class, while in the result of Gaussian mixture model, it correctly assign them to the sky blue class.

In the following example, the k-means model reaches an accuracy of 89.6%, while the Gaussian mixture model reaches an accuracy of 96.0%.



Both of the examples show that the k-means model considers only the distance between samples and cluster centers, while the Gaussian mixture model considers not only the distance but also the posterior probabilities.

Appendix

Run the programme by typing **“python source.py”** and specify command line options referring to **source.py**.