# Assignment 3

> To run the project
>
> > python source.py

## Part 1

> Entire dataset is available via data .

Python function `gen_data()` calls functions in `numpy.random` to build several sets of points drawn from certain distribution of separately specified means and covariance matrixes, while the scale of the datasets is controlled by parameter scale.

```
"""
:param means: means for the distributions, in shape of (k, dim)
:param covs: covariances for the distributions, in shape of (k, dim, dim)
:param scale: number of points in each set, in shape of (k)
"""
```

The dataset is then shuffled and saved in a file called `data.data`, where each line stands for a point and each point is displayed in form $(x_1, x_2, \ldots, x_{dim})$. Function `load_data()` reads the data file and converts data into type `numpy.ndarray`.

## Part 2

### Gaussian Mixture Model

**Gaussian Mixture Model** is implemented in class `GMM` in file *GMM.py* .

Gaussian Mixture Model assumes that data are generated from several($\mathbf{K}$) gaussian distribution. Each gaussian distribution is called a component and has distinct mean $\mu_k$ and covariance $\Sigma_k$. A k-dim one-hot latent variable $\mathbf{z} = [z_1, \ldots, z_K]$ was introduced to imply which component sample $\mathbf{x}$ is drawn from.

The marginal distribution of $\mathbf{z}$ is given by $p(z_k = 1) = \pi_k$, that's $p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$.

Since the conditional probability of $\mathbf{x}$ given $\mathbf{z}$ is given by $p(\mathbf{x}|z_k = 1) = p(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k})$, the marginal distribution of $\mathbf{x}$ is :

$$p(\mathbf{x}) = \sum_{j=1}^{K} p(z_j = 1)\, p(\mathbf{x}|z_j = 1) = \sum_{j=1}^{K} \pi_j p(\mathbf{x}|\mu_j, \mathbf{\Sigma}_j)$$

According to Bayes' theorem, after observation of sample $\mathbf{x}$, the posterior responsibility of $\mathbf{x}$ drawn from component $k$ is:

$$\begin{aligned} p(z_k = 1|\mathbf{x}) &= \frac{p(z_k = 1)\, p(\mathbf{x}|z_k = 1)}{p(\mathbf{x})} \\ &= \frac{\pi_k p(\mathbf{x}|\mu_\mathbf{k}, \mathbf{\Sigma_k})}{\sum_{j=1}^{K} \pi_j p(\mathbf{x}|\mu_\mathbf{j}, \mathbf{\Sigma_j})} \end{aligned}$$

For the entire dataset $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x_N}\}$, the log likelihood function is:

$$\ln p(\mathbf{X}|\pi,\mu,\mathbf{\Sigma}) = \sum_{n=1}^{N} \ln \left[ \sum_{k=1}^{K} \pi_k p\left(\mathbf{x}_n|\mu_\mathbf{k},\mathbf{\Sigma_k}\right) \right]$$

So the solving of the model is converted to restricted extremum problem as follows:

$$\max_{\pi,\mu,\Sigma} \ln p(\mathbf{X}|\pi,\mu,\mathbf{\Sigma})$$
$$s.t. \sum_{k=1}^{K} \pi_k = 1$$

With Lagrange multipliers, we can solve:

$$\begin{cases} \mu_\mathbf{k} = \frac{1}{N_k} \sum_{n=1}^{N} \left[\gamma\left(z_{nk}\right) \mathbf{x_n}\right] \\ \mathbf{\Sigma_k} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma\left(z_{nk}\right) \left(\mathbf{x_n} - \mu_\mathbf{k}\right) \left(\mathbf{x_n} - \mu_\mathbf{k}\right)^T \\ \pi_k = \frac{N_k}{N} \end{cases}$$

where, $\gamma\left(z_{nk}\right) = \frac{\pi_k p(\mathbf{x_n}|\mu_\mathbf{k},\mathbf{\Sigma_k})}{\sum_{j=1}^{K} \pi_j p(\mathbf{x_a}|\mu_\mathbf{j},\mathbf{\Sigma}_j)}, \quad N_k = \sum_{n=1}^{N} \gamma\left(z_{nk}\right)$.

In fact, since parameters $\pi, \mu, \Sigma$ are unknown, we cannot calculate $\gamma\left(z_{nk}\right)$ and $N_k$ to get analytical solution, EM algorithm is used.

**EM algorithm**

The **EM algorithm**(Expectation-Maximization algorithm) is implemented as follows:

1. **Initialize**: Initialize $\pi, \mu, \Sigma$ and calculate the initial value of the log likelihood function.
    - In fact, to converge faster, the results derived from first few step of k-means can be used.
2. **Expectation step**: Use current value of $\pi, \mu, \Sigma$ to calculate the responsibility $\gamma\left(z_{nk}\right)$

$$\gamma\left(z_{nk}\right) \leftarrow \frac{\pi_k p\left(\mathbf{x_n}|\mu_\mathbf{k},\mathbf{\Sigma_k}\right)}{\sum_{j=1}^{K} \pi_j p\left(\mathbf{x_n}|\mu_\mathbf{j},\mathbf{\Sigma_j}\right)}$$

3. **Maximization step**: Use current responsibility $\gamma\left(z_{nk}\right)$ to calculate $\pi, \mu, \Sigma$

$$\begin{aligned} \mu_\mathbf{k}^{\text{new}} &\leftarrow \frac{1}{N_k} \sum_{n=1}^{N} \left[\gamma\left(z_{nk}\right) \mathbf{x_n}\right] \\ \mathbf{\Sigma_k}^{\text{new}} &\leftarrow \frac{1}{N_k} \sum_{n=1}^{N} \gamma\left(z_{nk}\right) \left(\mathbf{x_n} - \mu_\mathbf{k}^{\text{new}}\right) \left(\mathbf{x_n} - \mu_\mathbf{k}^{\text{new}}\right)^T \\ \pi_k^{\text{new}} &\leftarrow \frac{N_k}{N} \end{aligned}$$

4. **Checkout**: Use the new parameter $\mu_\mathbf{k}^{\text{new}}$ , $\sum_\mathbf{k}^{\text{new}}$ , $\pi_k^{\text{new}}$ to check if the log likelihood function has converged, if not, turn to step 2 and iterate.

> In fact, expectation & maximization step ensures the rise of log likelihood function, however, as a iterating method, EM algorithm may still converges to a local maximum point.

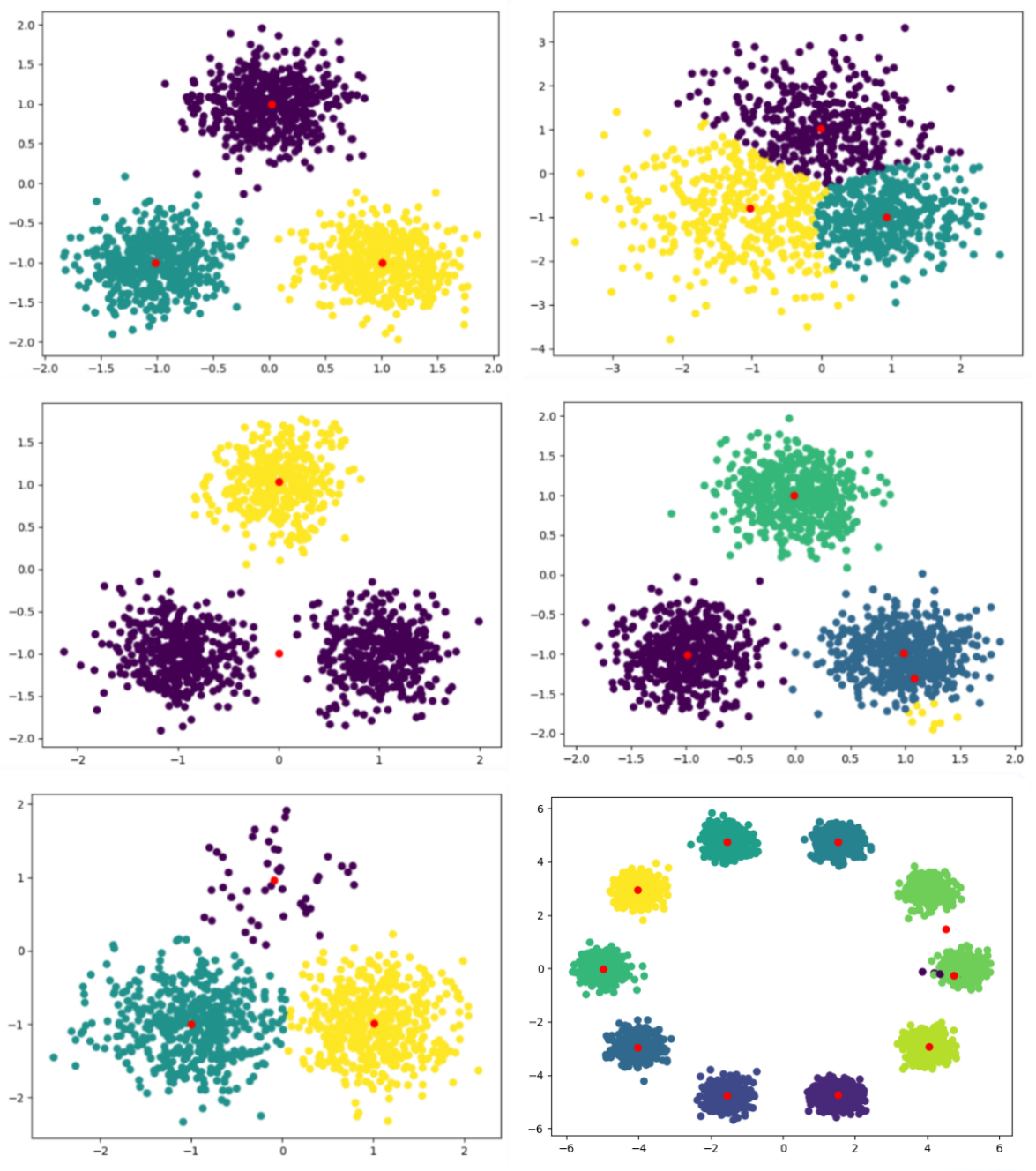
**Exploration on the GMM Model**

Several situations are considered to test the performance of the GMM Model.

- **Data generated from gaussian distributions.**

1. Default situation: $\mathbf{K} = 3$, $dim = 2$, obvious(large) overlap, 400 samples each set.
2. Small overlap: $\mathbf{K} = 3$, $dim = 2$, small overlap, 400 samples each set.
3. Less clusters: $\mathbf{K} = 3$, $dim = 2$, obvious overlap, 400 samples each set, 2 clusters.
4. More clusters: $\mathbf{K} = 3$, $dim = 2$, obvious overlap, 400 samples each set, 4 clusters.
5. Biased samples: $\mathbf{K} = 3$, $dim = 2$, obvious overlap, (40, 500, 500) samples each set.
6. High-dimension: $\mathbf{K} = 3$, $dim = 10$, obvious overlap, 400 samples each set.(can't visualize)
7. Multiple component: $\mathbf{K} = 10$, $dim = 2$, obvious overlap, 400 samples each set.

Results are visualized below(in the sequence of 1,2; 3,4; 5, 7)



We can find that:

- If the dataset is generated from several gaussian distribution, as correct priori is added, the model usually performs well(both in speed and accuracy).
- When $\mathbf{K}$ goes up, model requires far more time to converge, using results driven from k-means may reduce the number of iterations needed before converge. Higher $\mathbf{K}$ also means more likely to be stuck in local maximum points.

For 2-dim data, we can visualize and more easily spot a false cluster number $\mathbf{K}$ or converged to local maximum. The initialize for the parameters in GMM can greatly affect the final result of the model, so different initialize can be used to ensure a better result. Besides, we may use quality metrics to examine whether there has been some abnormal cluster.

- For data generated from other distributions, due to the approximability of gaussian distribution, the GMM model also has a good performance.