Lecture 10: Bayesian GMM and Non-parametric Clustering

Shukai Gong

1 Bayesian GMM

Recall that in the GMM paradigm, we have the following likelihood function for a single data point $\boldsymbol{x}$:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} w_k \underbrace{p(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}_{\text{Gaussian}}$$

we make a slight modification to the model by changing the covariance matrix Σ_k into precision matrix $\Phi_k = \Sigma_k^{-1}$, and we have the following likelihood function:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} w_k \underbrace{p(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Phi}_k)}_{\text{Gaussian}}$$

Under the Bayesian point of view, the parameters of the GMM $\boldsymbol{\Theta} = \{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Phi}_k\}_{k=1}^K$ are set as random variables obeying a **prior distribution**. We take Bayesian inference of 1-dimension GMM as an example and assume that the prior distribution of the parameters is a conjugate prior, which is

- Mixing coefficients: $w_k \sim \text{Dirichlet}(\delta_1, \cdots, \delta_K), \forall k$
- Precision matrix (1-dimension): $\phi_k \sim \text{Gamma}(\frac{a_k}{2}, \frac{b_k}{2}), \forall k$
- Mean: $\mu_k|\phi_k \sim \mathcal{N}(m_k, \frac{1}{\alpha_k \phi_k}), \forall k$

Given data x and the latent variable z, suppose we have N_k data points in the k-th cluster, then the posterior distribution of the parameters is

- $w_k|x, z \sim \text{Dirichlet}(\delta_1^*, \cdots, \delta_K^*), \delta_k^* = \delta_k + N_k$
- $\phi_k|x, z \sim \text{Gamma}(\frac{a_k^*}{2}, \frac{b_k^*}{2}), a_k^* = a_k + \frac{N_k}{2}, b_k^* = b_k + \frac{1}{2} \underbrace{\sum_{n=1}^{N_k}(x_n - \mu_k)^2}_{\text{All data points within the k-th cluster}}$
- $\mu_k|x, z, \phi \sim \mathcal{N}(m_k^*, \frac{1}{\alpha_k^* \phi_k^*}), m_k^* = \frac{\alpha_k m_k + \sum_{n=1}^{N_k} x_n}{\alpha_k^*}, \phi_k^* = \phi_k + N_k$

1.1 MCMC Algorithm for Bayesian GMM

MCMC Algorithm

- **Initialization:** Initialize the parameters $\boldsymbol{\Theta} = \{w_k, \boldsymbol{\mu}_k, \boldsymbol{\Phi}_k\}_{k=1}^K$ via sampling from priors
- **Repeat following steps until convergence:**
  1. Sampling latent variables z_{nk} (the responsibility of the k-th cluster for the n-th data point) from $z_{nk} \sim z|\boldsymbol{x}_n, w_k, \boldsymbol{\mu}_k, \boldsymbol{\Phi}_k$

2. Sampling $\boldsymbol{w} \sim w|\{z_{nk}\}_{n,k}, \{\boldsymbol{x}_n\}$
3. Sampling $\phi_k \sim \phi_k|\{z_{nk}\}_{n,k}, \{\boldsymbol{x}_n\}$
4. Sampling $\boldsymbol{\mu}_k \sim \boldsymbol{\mu}_k|\{z_{nk}\}_{n,k}, \{\boldsymbol{x}_n\}, \phi_k$

The difference of EM-optimized GMM and Bayesian GMM is that EM algorithm is a **point estimation** method while Bayesian GMM is a **distribution estimation** method. EM algorithm computes the responsibility γ_{ik} (E-step) and optimizes parameters $\boldsymbol{\Theta}$ (M-step) in a deterministic way, while MCMC algorithm samples the parameters latent codes and parameters in a probabilistic way.

Bayesian GMM can be further extended to an infinite mixture model, i.e. the model learns the number of clusters K. However, the complexity of Bayesian GMM is high due to the efficiency of sampling.

2 Nonparametric Clustering

2.1 Kernel Density Estimation

Given i.i.d. samples, we wish to estimate their density function in a nonparametric way.

Kernel Density Estimation (KDE)

Given i.i.d. samples $\{\boldsymbol{x}_n\}_{n=1}^N$, the KDE of the samples is defined as

$$\hat{p}_h(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_n) := \frac{1}{N}\sum_{n=1}^{N} K_h(\boldsymbol{x} - \boldsymbol{x}_n)$$

where h is the bandwidth of the kernel.

[Note]: For 1-dimension data, under Gaussian kernel, a rule of thumb for bandwidth selection is $h = \left(\frac{4\hat{\sigma}^5}{3N}\right)^{0.2} \approx 1.06\hat{\sigma}N^{-0.2}$, where $\hat{\sigma}$ is the standard deviation of the samples.
[Note]: When K_h is a Gaussian kernel, the KDE can actually be interpreted as a GMM model with **known parameters**.

Figure 1 shows a simple example of the KDE of a 1-dimension dataset. The left histogram is essentially a KDE graph with a gate kernel ($K_h(\boldsymbol{x}, \boldsymbol{x'}) = \frac{1}{h} \cdot \mathbb{I}(\|\boldsymbol{x} - \boldsymbol{x'}\|_1 \leq h)$), and the right KDE curve is obtained by adding up the Gaussian kernels ($K_h(\boldsymbol{x}, \boldsymbol{x'}) = \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x'}\|_2^2}{2h^2}\right)$) of each sample.

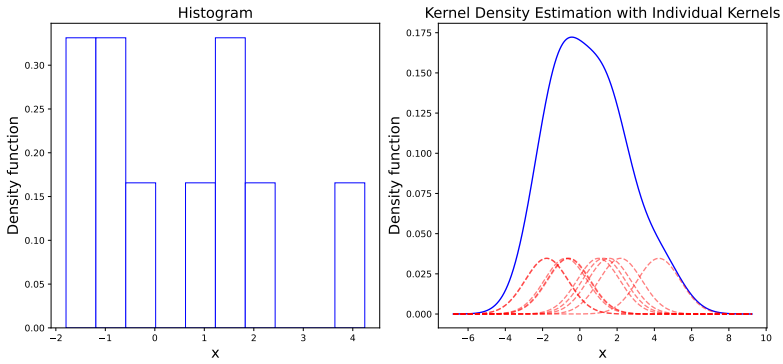

Figure 1: Kernel Density Estimation

2.2 Mean-Shift Algorithm

As an overview, the Mean-Shift clustering algorithm is a nonparametric clustering algorithm that works by iteratively shifting the data points towards the **densest** area of the data, and therefore forming clusters. Figure 2 shows a simplest 1D example of shifting process. The data points are shifted towards the peak of kernel density.

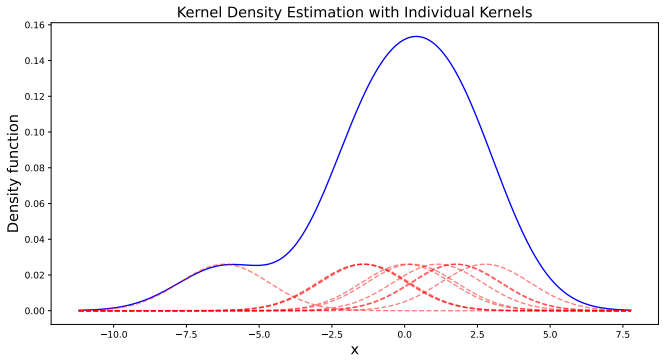

Figure 2: Shifting Data Points to the Densest Area

It's nonparametric because unlike K-means clustering, it doesn't require the number of clusters, K, to be specified beforehand. The number of clusters is determined based on the areas where the density of the data points is highest, and the densest areas are determined by the kernel density estimation introduced before.

Before we introduce the steps of Mean-Shift algorithm, we need to specify the definition of *the mean of a data point* in the algorithm. Given a set of samples $\{\boldsymbol{x}_n\}_{n=1}^N$ and a kernel function $K_h(\boldsymbol{x}, \boldsymbol{x'})$ with bandwidth h, the mean of a data point $\boldsymbol{x}$ in its neighborhood $\mathcal{N}(\boldsymbol{x})$ is defined as

$$m(\boldsymbol{x}) = \frac{\sum\limits_{\boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{x})} \boldsymbol{x}_i K_h(\boldsymbol{x}, \boldsymbol{x}_i)}{\sum\limits_{\boldsymbol{x}_i \in \mathcal{N}(\boldsymbol{x})} K_h(\boldsymbol{x}, \boldsymbol{x}_i)}$$

the neighborhood $\mathcal{N}(\boldsymbol{x})$ can be defined as a circle centered at $\boldsymbol{x}$ with radius h, or a KNN neighborhood with k nearest neighbors, or merely all data points. Here we can adopt $\mathcal{N}(\boldsymbol{x}) = \{\boldsymbol{x}_i\}_{i=1}^N$ for simplicity, that is,

$$m(\boldsymbol{x}) = \frac{\sum\limits_{i=1}^{N} \boldsymbol{x}_i K_h(\boldsymbol{x}, \boldsymbol{x}_i)}{\sum\limits_{i=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_i)}$$

one can notice that $m(\boldsymbol{x})$ coincides with the definition of Nadaraya-Watson estimator in kernel regression.

Mean-Shift Algorithm

- **Initialization:** Initialize the mean of each data point to its own value.
- **Mean-Shift Vector:** For each $\boldsymbol{x}_i$, the mean-shift vector is $m(\boldsymbol{x}_i) - \boldsymbol{x}_i = \dfrac{\sum\limits_{i=1}^{N} \boldsymbol{x}_i K_h(\boldsymbol{x}, \boldsymbol{x}_i)}{\sum\limits_{i=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_i)} - \boldsymbol{x}_i$.
- **Shifting:** Updating by $\boldsymbol{x}_i \leftarrow m(\boldsymbol{x}_i)$
- **Iteration:** Repeat steps 2 and 3 until convergence or maximum iteration is reached.

Figure 3 shows the Mean-Shift algorithm in action. Now we will justify why by updating $\boldsymbol{x}_i \leftarrow m(\boldsymbol{x}_i)$, the data points will eventually converge to their nearest peakw of the kernel density. Suppose the kernel function is a Gaussian kernel $K_h(\boldsymbol{x}, \boldsymbol{x'}) = \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x'}\|_2^2}{2h^2}\right)$, we compute the gradient of kernel density $\hat{p}_h(\boldsymbol{x}) = \frac{1}{N}\sum_{n=1}^{N} K_h(\boldsymbol{x} - \boldsymbol{x}_n)$ as

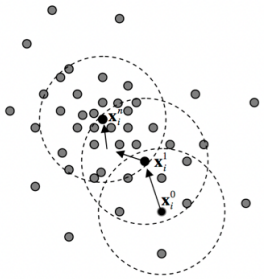

Figure 3: Mean-shift Algorithm

$$
\begin{aligned}
\nabla_{\boldsymbol{x}} \hat{p}_h(\boldsymbol{x}) &= \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{x}} K_h(\boldsymbol{x} - \boldsymbol{x}_n) = \frac{1}{N}\sum_{n=1}^{N} \nabla_{\boldsymbol{x}} \left(\frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2}{2h^2}\right)\right) \\
&= \frac{1}{N}\sum_{n=1}^{N} \frac{1}{\sqrt{2\pi}h} \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}_n\|_2^2}{2h^2}\right) \frac{\boldsymbol{x} - \boldsymbol{x}_n}{h^2} = -\frac{1}{Nh^2}\sum_{n=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_n)(\boldsymbol{x} - \boldsymbol{x}_n) \\
&\propto \sum_{n=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_n)(\boldsymbol{x} - \boldsymbol{x}_n) \propto \frac{\sum_{n=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_n)(\boldsymbol{x}_n - \boldsymbol{x})}{\sum_{n=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_n)\boldsymbol{x}} = \frac{\sum_{n=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_n)\boldsymbol{x}_n}{\sum_{n=1}^{N} K_h(\boldsymbol{x}, \boldsymbol{x}_n)} - \boldsymbol{x} \\
&= m(\boldsymbol{x}) - \boldsymbol{x}
\end{aligned}
$$

The gradient of the kernel density is proportional to the mean-shift vector, which means that by updating $\boldsymbol{x}_i \leftarrow m(\boldsymbol{x}_i)$, we are essentially doing a gradient ascent on the data points: $\boldsymbol{x}_n^{(t+1)} = \boldsymbol{x}_n^{(t)} + \eta \nabla_{\boldsymbol{x}} \hat{p}_h(\boldsymbol{x}_n^{(t)})$.

The Mean-Shift algorithm can also be considered as a variant of EM algorithm, where the **E-step** is computing $m(\boldsymbol{x}_i), \forall i$, and the **M-step** is updating $\boldsymbol{x}_i \leftarrow m(\boldsymbol{x}_i)$. Under the EM algorithm setup:

- Data: $\{\boldsymbol{x}_n\}_{n=1}^N$
- Latent variable: means / centroids $\{\boldsymbol{m}_k\}_{k=1}^K$
- E-step: estimating the mean $m(\boldsymbol{x}_i)$ conditioned on $\boldsymbol{x}_i$.
- M-step: maximizng $p(\boldsymbol{x}_i)$ (finding the nearest peak)

References

- Latent Variable Model: Gaussian Mixture Model
- EM Algorithm for GMM and its Bayesian interpretation
- Mean Shift
- Machine Learning - Mean-Shift Clustering
- Wikipedia - Mean shift