Statistical Pattern Recognition

Shuhei Watanabe

March 31, 2022

1 Preliminaries

In this section, we define the notations used in this notebook.

- $\boldsymbol{\theta}$, a set of parameters of a probability distribution,
- $\mathcal{D}$, the observations used for statistical inference,
- $\boldsymbol{x} \in \mathbb{R}^D$, a vector that belongs to the space, on which a probability distribution is defined,
- $\boldsymbol{X} \in \mathbb{R}^{N \times D}$, a set of observations $\boldsymbol{x}$,
- $D \in \mathbb{Z}_+$, the dimension of the space,
- $N \in \mathbb{Z}_+$, the number of observations,
- $K \in \mathbb{Z}_+$, the number of clusters or categories,
- $p(\boldsymbol{x}|\boldsymbol{\theta}) : \mathbb{R}^D \to \mathbb{R}_+$, the probability density function of a probability distribution with its parameters $\boldsymbol{\theta}$,
- $\boldsymbol{I}_n \in \mathbb{R}^{n \times n}$, the n-dimensional identity matrix, and
- $\boldsymbol{0}_n \in \mathbb{R}^n$, the n-dimensional zero vector.

We stick to the notations above throughout the notebook if not specified. Note that any famous distributions also follow the same notation. For example, while the Gaussian distribution with the mean vector $\boldsymbol{\mu}$ and the covariance matrix Σ is written as $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$, its probability density function is written as $\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \Sigma)$.

2 Introduction

While rule-based algorithms solve many real-world tasks, such heuristics do not generalize to complicated tasks such as digit recognition. For example, when we tilt images, heuristics require some more manually engineered rules. Machine learning or pattern recognition is used to address this problem. In machine learning, we let computer learn decision rules automatically from a set of examples so that we do not have to add rules manually. By extracing some patterns from the provided dataset, we can predict outcomes of unseen data. Such pattern recognition is classified mostly into either supervised or unsupervised learning. Supervised learning includes regression and classification tasks and unsupervised learning includes clustering, density estimation, and subspace estimation.

In statistical pattern recognition, we aim to infer parameters of a parametric probability distribution based on a provided dataset or derive the underlying distribution of parameters. Such inferences rely on the following Bayes' theorem:

$$
\begin{aligned}
p(\boldsymbol{\theta}|\mathcal{D}) &= \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}, \\
\text{Posterior} &= \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal Likelihood}}
\end{aligned}
\tag{1}
$$

Table 1: Concrete examples of each model.

Models	Methods
Generative models	Auto encoder
	GAN
Discriminative models	Logistic regression
	Gaussian process
Mappings	LDA
	SVM
	AdaBoost
	Decision tree

where $\boldsymbol{\theta}$ is parameters of the posterior. The frequent inferences are maximum likelihood estimation (MLE), which is the maximization of $p(\mathcal{D}|\boldsymbol{\theta})$, and maximum a posteriori (MAP) estimation, which is the maximization of $p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$. MLE is inclined to overfit the dataset and MAP estimation regularizes the estimation of the parameters $\boldsymbol{\theta}$ by a prior distribution. Both inferences are very important in statistical pattern recognition, so we handle the closed-form solutions of famous probability distributions in Section 3.

In machine learning, practitioners use the following three models as well:

1. **Generative model**: Learning of prior and likelihood from a training dataset. This inference is usually more complex and difficult compared to the following two models as we assume the dimension of $\mathcal{D}$ is very high. One can also make a decision using the approximated posterior.
2. **Discriminative model**: Direct learning of the posterior from a training dataset. Since $\boldsymbol{\theta}$ is usually defined in a low-dimensional space, this model is less complex.
3. **Mappings**: Direct learning of a mapping from an input to an output. As probabilities do not play a role here anymore, the training of such a model is achievable with less data compared to previous two models.

Concrete examples for each model is listed in Table 1. Note that since classification methods such as linear discriminant analysis, AdaBoost, decision trees, logistic regression, and support vector machines are discussed in the machine learning course, we do not discuss those models in this notebook.

3 Probability Distribution

3.1 Bernoulli Distribution

Suppose $\mathbb{P}[x = 1|\mu] = \mu$ represents the probability that we get $x = 1$ where $x \in \{0, 1\}$, then the probability mass function of the Bernoulli distribution is the following:

$$\mathrm{Bern}(x|\mu) = \mu^x (1-\mu)^{1-x}. \tag{2}$$

The mean and variance of the distribution are $\mathbb{E}[x] = 0 \times \mathrm{Bern}(x = 0|\mu) + 1 \times \mathrm{Bern}(x = 1|\mu) = \mu$ and $\mathbb{V}[x] = \mu(1-\mu)$. Additionally, given a dataset $\mathcal{D} = \{x_n\}_{n=1}^N$ sampled i.i.d., the MLE is computed as

Table 2: The list shows the priors conjugate to each likelihood function. The posterior and the conjugate prior take the same form, and the marginal distribution takes the form written in the predictive distribution column.

Likelihood	Parameters	Conjugate Prior	Predictive Distribution
Binomial	μ	Beta	Beta · binomial
Multinomial	$\boldsymbol{\mu}$	Dirichlet	Dirichlet · multinomial
Gaussian	$\boldsymbol{\mu}$	Gaussian	Gaussian
	$\boldsymbol{\Lambda}$	Wishart (Gamma for 1D)	Student's t
	$\boldsymbol{\mu}, \boldsymbol{\Lambda}$	Gaussian-Wishart (Gauss-Gamma for 1D)	Student's t

follows:

$$\begin{aligned} p(\mathcal{D}|\mu) &= \prod_{n=1}^{N} \mathrm{Bern}(x_n|\mu), \\ \log p(\mathcal{D}|\mu) &= \sum_{n=1}^{N} \biggl(x_n \log \mu + (1 - x_n) \log(1 - \mu) \biggr), \\ \frac{\partial}{\partial \mu} \log p(\mathcal{D}|\mu) &= \sum_{n=1}^{N} \biggl(\frac{x_n}{\mu} - \frac{1 - x_n}{1 - \mu} \biggr) = \frac{N_1}{\mu} - \frac{N_0}{1 - \mu}, \\ \mu_{\mathrm{MLE}} &= \frac{N_1}{N} \biggl(\because \frac{\partial}{\partial \mu} \log p(\mathcal{D}|\mu) = 0 \biggr). \end{aligned} \tag{3}$$

where N_0 and N_1 are the number of occurrences of $x_n = 0$ and $x_n = 1$, respectively, and $N_0 + N_1 = N$ holds.

3.2 Binomial Distribution

When $x = 1$ occurs N_1 times during N trials of the same task as in the previous section, the probability mass function of the binomial distribution is calculated as follows:

$$\mathrm{Bin}(N_1|N, \mu) = {}_N\mathrm{C}_{N_1} \mu^{N_1} (1 - \mu)^{N_0} \tag{4}$$

where $N_0 = N - N_1$ is the number of occurrences of $x = 0$. The mean and variance of the distribution are $\mathbb{E}[x] = N\mu$ and $\mathbb{V}[x] = N\mu(1 - \mu)$. Since the MLE is inclined to overfit with a small dataset, we may want to regularize the estimation. To do so, **Bayesian inference** is employed for the posterior inference or **MAP estimation** is employed for the optimal parameter estimation regularized by a prior distribution. As the marginal likelihood is often intractable, Bayesian inference is hard to perform. However, when we carefully choose the prior, the posterior calculation becomes much tractable. Such priors are called **conjugate prior** listed in Table 2. When we choose the conjugate prior of a likelihood function, it is known that the posterior distribution becomes the same probability distribution family as the prior distribution.

The conjugate prior of the binomial distribution is the following Beta distribution:

$$\mathrm{Beta}(\mu|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \mu^{\alpha - 1} (1 - \mu)^{\beta - 1} \tag{5}$$

where $\alpha, \beta \in \mathbb{R}_+$ are hyperparameters that control the regularization effect and $\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$ is the Gamma function. The posterior distribution will be smoothed more when α and β are larger. Given the conjugate prior, the posterior is computed as follows:

$$p(\mu|N_1, N, \alpha, \beta) \propto \mu^{N_1 + \alpha - 1} (1 - \mu)^{N_0 + \beta - 1}. \tag{6}$$

By maximizing the posterior, we obtain $\mu_{\mathrm{MAP}} = \frac{n+\alpha-1}{N+\alpha+\beta-2}$. μ_{MAP} converges to μ_{MLE} as N goes to infinity. Note that although we assume a-priori assumption, i.e. $\alpha/\beta = 1$ in most cases, we often rely on frequentism to determine those parameters to be more objective. For example, we repeat experiments for 100 times and take the probability of $x = 1$ as the ratio of α/β to reduce the degree of freedom.

3.3 Multinomial Distribution

The generalization of the binomial distribution is the multinomial distribution. Namely, the multinomial distribution considers $\boldsymbol{x} \in \{0,1\}^K$ where $\boldsymbol{x}$ is a one-hot vector and K is the number of categories. More formally, the multinomial distribution is represented as:

$$\mathrm{Multi}(\boldsymbol{x}|\boldsymbol{\mu}) = \frac{N!}{\prod_{k=1}^K N_k!} \prod_{k=1}^K \mu_k^{x_k} \tag{7}$$

where $\mu_k \geq 0$, $\sum_{k=1}^K \mu_k = 1$, $\sum_{k=1}^K N_k = N$, and N_k is the number of occurrences of $x_n = k$. Given a dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$, the likelihood is computed as:

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{n=1}^N \prod_{k=1}^K \mu_k^{x_{n,k}} = \prod_{k=1}^K \mu_k^{N_k}. \tag{8}$$

The maximization of the likelihood is solved using the Lagrangian multiplier:

$$\mathcal{L}(\boldsymbol{\mu}, \lambda) = \sum_{k=1}^K N_k \log \mu_k + \lambda\biggl(\sum_{k=1}^K \mu_k - 1\biggr). \tag{9}$$

The KKT conditions are satisfied when the derivatives with respect to $\boldsymbol{\mu}$ and λ are zero and we obtain $\boldsymbol{\mu}_{\mathrm{MLE}} = [N_1/N, \dots, N_K/N]$.

Since the conjugate prior of the multinomial distribution is the Dirichlet distribution:

$$\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma\biggl(\sum_{k=1}^K \alpha_k\biggr)}{\prod_{k=1}^K \Gamma(\alpha_k)} \prod_{k=1}^K \mu_k^{\alpha_k - 1}. \tag{10}$$

The posterior also becomes the Dirichlet distribution:

$$p(\boldsymbol{\mu}|\boldsymbol{n}, \boldsymbol{\alpha}) \propto \prod_{k=1}^K \mu_k^{N_k + \alpha_k - 1}. \tag{11}$$

Hence, we yield $\boldsymbol{\mu}_{\mathrm{MAP}} = \left[\frac{N_1+\alpha_1-1}{N+\sum_{k=1}^K(\alpha_k-1)}, \dots, \frac{N_K+\alpha_K-1}{N+\sum_{k=1}^K(\alpha_k-1)}\right]$ by the MAP estimation.

3.4 Gaussian Distribution

The D-dimensional formulation is the following:

$$\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\biggl(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\biggr). \tag{12}$$

The exponent part $(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ in the equation above. When the dimension D is large, we can opt for the covariance matrix $\boldsymbol{\Sigma}$ to be a diagonal matrix so that the inference time scales linearly to D in exchange for the representational capacity.

3.4.1 Basic Properties

Gaussian distribution is closed under conditioning, multiplication, marginalization, and linear mapping. Additionally, since the covariance matrix is always symmetric and positive definite, $\boldsymbol{\Sigma}$ can be decomposed by a principal axes transformation $\boldsymbol{\Sigma} = \boldsymbol{U}^\top \mathrm{diag}(\lambda_1, \cdots, \lambda_D)\boldsymbol{U}$ where $\boldsymbol{U}$ is a unitary matrix [1] and λ_d is an eigenvalue of $\boldsymbol{\Sigma}$. Using this property, $\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}^\top \mathrm{diag}(\lambda_1^{-1}, \cdots, \lambda_D^{-1})\boldsymbol{U}$ and we obtain $(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x} - \boldsymbol{\mu}) = (U(\boldsymbol{x} - \boldsymbol{\mu}))^\top \mathrm{diag}(\lambda_1^{-1}, \cdots, \lambda_D^{-1})U(\boldsymbol{x} - \boldsymbol{\mu})$. By taking $U(\boldsymbol{x} - \boldsymbol{\mu})$ as a new coordinate system and $\mathrm{diag}(\lambda_1, \cdots, \lambda_D)$ as a covariance matrix, we can remove the correlation between each coordinate. Principal component analysis uses this property.

3.4.2 Maximum Likelihood Estimation of Parameters

Given a dataset $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$, the maximum likelihood is achieved when we take the following:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\mu}} \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \boldsymbol{0}_D, \\ \frac{\partial}{\partial \boldsymbol{\Sigma}} \log \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \boldsymbol{0}_{D \times D}. \end{aligned} \tag{13}$$

By solving the equations, we obtain $\boldsymbol{\mu}_{\mathrm{MLE}} = \frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n$ and $\boldsymbol{\Sigma}_{\mathrm{MLE}} = \frac{1}{N}\sum_{n=1}^N \boldsymbol{x}_n \boldsymbol{x}_n^\top - \boldsymbol{\mu}_{\mathrm{MLE}}\boldsymbol{\mu}_{\mathrm{MLE}}^\top$. Since the solutions are computed only using the first momentum $\sum_{n=1}^N \boldsymbol{x}_n$ and the second momentum $\sum_{n=1}^N \boldsymbol{x}_n \boldsymbol{x}_n^\top$, we do not have to store each data point $\boldsymbol{x}_n$ for the next update. As these statistics are sufficient to calculate the parameters, they are dubbed **sufficient statistics**. The parameter updates can be performed sequentially and more efficiently using the sufficient statistics. While the expectation of mean $\boldsymbol{\mu}_{\mathrm{MLE}}$ is $\boldsymbol{\mu}_{\mathrm{MLE}}$, that of the covariance matrix is:

$$\begin{aligned} N\mathbb{E}[\boldsymbol{\Sigma}_{\mathrm{MLE}}] &= \mathbb{E}\biggl[\sum_{n=1}^N (\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{x}_n)^2\biggr] \\ &= \mathbb{E}\biggl[\sum_{n=1}^N (\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}} + \boldsymbol{\mu}_{\mathrm{true}} - \boldsymbol{x}_n)^2\biggr] \\ &= \underbrace{\mathbb{E}\biggl[\sum_{n=1}^N (\boldsymbol{x}_n - \boldsymbol{\mu}_{\mathrm{true}})^2\biggr]}_{=N\boldsymbol{\Sigma}_{\mathrm{true}}} + \mathbb{E}\biggl[\sum_{n=1}^N \underbrace{(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})^2}_{\text{const w.r.t. } n}\biggr] - 2\mathbb{E}\biggl[\underbrace{\sum_{n=1}^N (\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})(\boldsymbol{x}_n - \boldsymbol{\mu}_{\mathrm{true}})}_{=N(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})^2}\biggr] \\ &= N\boldsymbol{\Sigma}_{\mathrm{true}} - N\mathbb{E}[(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})^2]. \end{aligned} \tag{14}$$

Then we transform $\mathbb{E}[(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})^2]$ as follows:

$$\mathbb{E}[(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})^2] = \mathbb{V}\biggl[\frac{\sum_{n=1}^N \boldsymbol{x}_n}{N}\biggr] = \frac{1}{N^2}\mathbb{V}\biggl[\sum_{n=1}^N \boldsymbol{x}_n\biggr] = \frac{N}{N^2}\mathbb{V}[\boldsymbol{x}] = \frac{\boldsymbol{\Sigma}_{\mathrm{true}}}{N} \tag{15}$$

where the last transformation uses the assumption that $\boldsymbol{x}$ is sampled i.i.d. By plug-in the result, we obtain $\mathbb{E}[\boldsymbol{\Sigma}_{\mathrm{MLE}}] = \frac{N-1}{N}\boldsymbol{\Sigma}_{\mathrm{true}}$. This result implies that $\boldsymbol{\Sigma}_{\mathrm{MLE}}$ is underestimated compared to the ground truth. Since the covariance matrix is biased when N is small, we debias $\boldsymbol{\Sigma}_{\mathrm{MLE}}$ by multiplying $\frac{N}{N-1}$.

3.4.3 Bayesian Inference of Mean Given Variance

When we already know the covariance $\boldsymbol{\Sigma}$, the likelihood of $\boldsymbol{\mu}$ is computed as follows:

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{n=1}^N \mathcal{N}(\boldsymbol{x}_n | \boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{16}$$

[1] A unitary matrix satisfies $\boldsymbol{U}^{-1} = \boldsymbol{U}^\top$.

Since the conjugate prior is also the Gaussian distribution, we obtain the following posterior using the prior $p(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_{\mathrm{prior}}, \boldsymbol{\Sigma}_{\mathrm{prior}})$:

$$
\begin{aligned}
p(\boldsymbol{\mu}|\mathcal{D}) &\propto \biggl(\prod_{n=1}^{N} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})\biggr)\mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_{\mathrm{prior}}, \boldsymbol{\Sigma}_{\mathrm{prior}}) \\
\log p(\boldsymbol{\mu}|\mathcal{D}) &= -\frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}) - \frac{1}{2}(\boldsymbol{\mu} - \boldsymbol{\mu}_{\mathrm{prior}})^\top \boldsymbol{\Sigma}_{\mathrm{prior}}^{-1}(\boldsymbol{\mu} - \boldsymbol{\mu}_{\mathrm{prior}}) + \mathrm{const.}
\end{aligned}
\tag{17}
$$

Let the mean and the covariance of the posterior be $\boldsymbol{\mu}_{\mathrm{post}}$ and $\boldsymbol{\Sigma}_{\mathrm{post}}$. Then the parameters take the following form by transforming Eq. (17):

$$
\begin{aligned}
\boldsymbol{\mu}_{\mathrm{post}} &= \boldsymbol{\Sigma}_{\mathrm{post}}\biggl(\boldsymbol{\Sigma}^{-1}\sum_{n=1}^{N}\boldsymbol{x}_n + \boldsymbol{\Sigma}_{\mathrm{prior}}^{-1}\boldsymbol{\mu}_{\mathrm{prior}}\biggr), \\
\boldsymbol{\Sigma}_{\mathrm{post}} &= (N\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_{\mathrm{prior}}^{-1})^{-1}.
\end{aligned}
\tag{18}
$$

Note that we can trivially obtain the case of $D = 1$ as follows:

$$
\begin{aligned}
\sigma_{\mathrm{post}}^2 &= \frac{\sigma^2\sigma_{\mathrm{prior}}^2}{N\sigma_{\mathrm{prior}}^2 + \sigma^2}, \\
\mu_{\mathrm{post}} &= \sigma_{\mathrm{post}}^2\biggl(\frac{\sum_{n=1}^{N} x_n}{\sigma^2} + \frac{\mu_{\mathrm{prior}}}{\sigma_{\mathrm{prior}}^2}\biggr).
\end{aligned}
\tag{19}
$$

3.4.4 Bayesian Inference of Variance Given Mean

First, we set $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$ for the sake of simplicity. In this case, the conjugate prior is Gamma distribution for one dimension and Wishart distribution for multi dimensions:

$$
\begin{aligned}
\mathrm{Gam}(\lambda|\alpha, \beta) &= \frac{\beta^\alpha}{\Gamma(\alpha)}\lambda^{\alpha-1}e^{-\beta\lambda}, \\
\log \mathcal{W}(\boldsymbol{\Lambda}|\nu, \boldsymbol{W}) &= \frac{\nu - D - 1}{2}\log|\boldsymbol{\Lambda}| - \frac{1}{2}\mathrm{Tr}(\boldsymbol{W}^{-1}\boldsymbol{\Lambda}) + \mathrm{const},
\end{aligned}
\tag{20}
$$

where $\alpha, \beta \in \mathbb{R}_{>0}$, $\nu > D - 1$ and $\boldsymbol{W} \in \mathbb{R}^{D\times D}$ is a positive definite matrix. Using this equation and the fact that these are the conjugate priors of this setting, we obtain the following parameters:

$$
\begin{aligned}
\text{One dimension : } & \alpha_{\mathrm{post}} = \frac{N}{2} + \alpha_{\mathrm{prior}},\ \beta_{\mathrm{post}} = \frac{1}{2}\sum_{n=1}^{N}(x_n - \mu)^2 + \beta_{\mathrm{prior}}, \\
\text{Multi dimension : } & \boldsymbol{W}_{\mathrm{post}}^{-1} = \sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^\top + \boldsymbol{W}_{\mathrm{prior}}^{-1},\ \nu_{\mathrm{post}} = N + \nu_{\mathrm{prior}}.
\end{aligned}
\tag{21}
$$

Using the posterior, the MAP estimation of $\lambda, \boldsymbol{\Lambda}$ is computed as:

$$
\begin{aligned}
\lambda_{\mathrm{MAP}} &= \operatorname*{argmax}_{\lambda}\mathrm{Gam}(\lambda|\alpha_{\mathrm{post}}, \beta_{\mathrm{post}}) = \frac{\alpha_{\mathrm{post}} - 1}{\beta_{\mathrm{post}}}, \\
\boldsymbol{\Lambda}_{\mathrm{MAP}} &= (\nu - D - 1)\boldsymbol{W}_{\mathrm{post}}.
\end{aligned}
\tag{22}
$$

For the prediction of $\boldsymbol{x}$, **student's t-distribution**, which is obtained by the marginalization of the posterior with respect to the prior, might be employed:

$$
\begin{aligned}
\mathrm{St}(x|\mu, t, \nu) &= \int_0^\infty \mathcal{N}(x|\mu, \lambda^{-1})\mathrm{Gam}(\lambda|\alpha, \beta)d\lambda = \frac{\Gamma((\nu+1)/2)}{\Gamma(\nu/2)}\biggl(\frac{t}{\pi\nu}\biggr)^{1/2}\biggl(1 + \frac{t}{\nu}(x - \mu)^2\biggr)^{-(\nu+1)/2}, \\
\mathrm{St}(x|\boldsymbol{\mu}, \boldsymbol{T}, \nu') &= \int \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})\mathcal{W}(\boldsymbol{\Lambda}|\nu, \boldsymbol{W})d\boldsymbol{\Lambda} = \frac{\Gamma((\nu'+D)/2)}{\Gamma(\nu'/2)}\frac{|\boldsymbol{T}|^{1/2}}{(\pi\nu')^{D/2}}\biggl(1 + \frac{1}{\nu'}(\boldsymbol{x} - \boldsymbol{\mu})^\top\boldsymbol{T}(\boldsymbol{x} - \boldsymbol{\mu})\biggr)^{-(\nu'+D)/2}
\end{aligned}
\tag{23}
$$

where $\nu = 2\alpha, t = \alpha/\beta, \nu' = 1 - D + \nu$, and $\boldsymbol{T} = (1 - D + \nu)\boldsymbol{W} \in \mathbb{R}^{D \times D}$ is a positive definite matrix. These distributions are called **predictive distribution**, which does not depend on the parameters of the posterior due to the marginalization, and we can use them to predict the distribution of $\boldsymbol{x}$. The student's t-distribution is advantageous because it has **long tails** compared to Gaussian distribution and it is **robust to outliers**. Note that we, of course, need to set the hyperparameters of the prior distribution somehow to yield the predictive distribution.

Another long-tail distribution is the **Laplace distribution**:

$$\mathcal{L}(\boldsymbol{x}|\boldsymbol{\mu}, b) = \frac{1}{2b}\exp\biggl(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}\|}{b}\biggr) \tag{24}$$

If $\boldsymbol{x}$ is in 1D space, the MLE is obtained analytically as $\mu = \mathrm{med}(x_1, \cdots, x_N), b = 1/N\sum_{n=1}^{N}|x_n - \mu|$. However, we do not have any closed-form for higher dimensional spaces.

3.4.5 Bayesian Inference of Both Mean and Variance

In this case, the conjugate prior for the 1D Gaussian is the product of the Gaussian distribution and the Gamma distribution, i.e. the Gauss-Gamma distribution. For the multi-dimensional Gaussian, the conjugate prior is the product of the Gaussian distribution and the Wishart distribution, i.e. the Gauss-Wishart distribution. The formulations are as follows:

$$\begin{aligned} p(\mu, \lambda) =& \mathcal{N}(\mu|\mu_{\mathrm{prior}}, (b\lambda)^{-1})\mathrm{Gam}(\lambda|\alpha, \beta), \\ p(\boldsymbol{\mu}, \boldsymbol{\Lambda}) =& \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}_{\mathrm{prior}}, (b\boldsymbol{\Lambda})^{-1})\mathcal{W}(\boldsymbol{\Lambda}|\nu, \boldsymbol{W}). \end{aligned} \tag{25}$$

Since $p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\mathcal{D}) = p(\boldsymbol{\mu}|\boldsymbol{\Lambda}, \mathcal{D})p(\boldsymbol{\Lambda}|\mathcal{D})$ holds, we first derive the posterior of the mean $p(\boldsymbol{\mu}|\boldsymbol{\Lambda}, \mathcal{D})$ and then estimate the precision matrix using $p(\boldsymbol{\Lambda}|\mathcal{D}) = p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Lambda})p(\boldsymbol{\mu}, \boldsymbol{\Lambda})/p(\boldsymbol{\mu}|\boldsymbol{\Lambda}, \mathcal{D})$. The closed forms are available in this case as well. The predictive distribution for this case is the Student's t-distribution again and computed using $\log p(\boldsymbol{x}) = \log p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}) - \log p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\boldsymbol{x}) + \mathrm{const}$. Note that we use the results from the posterior computation to calculate $p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\boldsymbol{x})$.

4 Clurstering and EM Algorithm

Clustering is an unsupervised task to divide given data points into several groups. In this section, we cover Gaussian mixture models (GMM) and K-means, and we discuss both methods from the EM algorithm perspective.

4.1 Gaussian Mixture Models (GMM)

Although the parametric distributions handled in the previous section are very convenient analytically, their representational capacity is often not sufficient to solve real-world problems, which require multi-modal distributions. To address this problem, Gaussian mixture models (GMM) is widely used:

$$p(\boldsymbol{x}) = \sum_{k=1}^{K}\pi_k\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{26}$$

where the mixture coefficients must satisfy $\pi_k \geq 0$ and $\sum_{k=1}^K \pi_k = 1$. When we introduce discrete latent variables, GMM is reformulated as:

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{z}) &= \prod_{k=1}^K \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}, \\ p(\boldsymbol{x}) &= \int p(\boldsymbol{x}|\boldsymbol{z}) \underbrace{p(\boldsymbol{z})}_{p(z_k=1|\boldsymbol{z})=\pi_k} d\boldsymbol{z} \\ &= \sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned} \tag{27}$$

where $\boldsymbol{z} \in \{0,1\}^K$ is a one-hot vector. From Eq. (27), GMM is viewed as the marginalized likelihood with respect to the latent variable $\boldsymbol{z}$. Since π_k, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ are mutually dependent, a closed-form solution for MLE is not available. Hence, we use an iterative scheme that jointly optimizes the latent variable and the parameters. This scheme is called expectation-maximization (**EM**) algorithm and we will see the details in the next section.

4.1.1 EM Algorithm for GMM

We first consider the **E step** where we take the expectation with respect to the latent variable. E step assumes that we have **complete** dataset $\boldsymbol{X}, \boldsymbol{Z}$ and compute the following using Bayes' theorem:

$$\begin{aligned} \gamma_{k,n} \coloneqq p(z_k = 1|\boldsymbol{x}_n) &= \frac{p(\boldsymbol{x}_n|z_k = 1)p(z_k = 1)}{\sum_{k'=1}^K p(z_{k'} = 1)p(\boldsymbol{x}_n|z_{k'} = 1)} \\ &= \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^K \pi_{k'} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} \end{aligned} \tag{28}$$

where $\gamma_{k,n}$ is called **responsibility** of the k-th cluster for the n-th data point and it realizes a soft-assignment to each cluster. Then the log-likelihood of given data points is computed as:

$$\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^N \log\biggl(\sum_{k=1}^K \pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\biggr). \tag{29}$$

The next step is **M step** where we consider the maximization of the log-likelihood given the responsibilities and **incomplete** dataset $\boldsymbol{X}$. To solve the optimization with the equality constraint, i.e. $\sum_{k=1}^K \pi_k = 1$, we employ the following Lagrange multiplier:

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \lambda) = -\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) - \lambda\biggl(\sum_{k=1}^K \pi_k - 1\biggr). \tag{30}$$

The KKT conditions for $\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \lambda)$ are the following:

$$\begin{aligned} \textbf{Stationarity}: &\frac{\partial \mathcal{L}}{\partial \pi_k} = 0, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = \boldsymbol{0}_D, \frac{\partial \mathcal{L}}{\partial \boldsymbol{\Sigma}_k} = \boldsymbol{0}_{D\times D}, \\ \textbf{Primal Feasibility}: &\sum_{k=1}^K \pi_k = 1. \end{aligned} \tag{31}$$

From the stationarity conditions, we obtain:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\mu}_k} \log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= -\sum_{n=1}^{N} \gamma_{k,n} \boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = \boldsymbol{0}_D, \\
\boldsymbol{\mu}_k &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{k,n}\boldsymbol{x}_n, \\
\frac{\partial}{\partial \boldsymbol{\Sigma}_k} \log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \sum_{n=1}^{N} \gamma_{k,n}\biggl(-\frac{1}{2}\boldsymbol{\Sigma}_k^{-1} + \frac{1}{2}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_k^{-1}\biggr) = \boldsymbol{0}_{D\times D}, \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{k,n}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top, \\
\frac{\partial \mathcal{L}}{\partial \pi_k} &= -\sum_{n=1}^{N} \frac{\gamma_{k,n}}{\pi_k} - \lambda = 0, \\
\pi_k &= \frac{N_k}{N},
\end{aligned} \tag{32}$$

where $N_k = \sum_{n=1}^{N} \gamma_{k,n}$. These mean vectors and covariance matrices correspond to the weighted average of those obtained from the fed data points. Note that since the global maximum of the likelihood is trivially achieved by taking the singular covariance matrices at each data point, we need to avoid such shrinkage by initializing far away from such solutions or applying a prior to suppress such shrinkage.

In summary, the EM algorithm for GMM is performed as follows after the initialization of $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}$:

1. **E step**: Given the fixed parameters, compute the **expectation** with respect to the latent variable:
$$\gamma_{k,n} \coloneqq \mathbb{E}[z_k|\boldsymbol{x}_n] = p(z_k = 1|\boldsymbol{x}_n), \text{ and} \tag{33}$$
2. **M step**: Given the responsibilities, **maximize** the log-likelihood in Eq. (32)
$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{k,n}\boldsymbol{x}_n,\ \boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{k,n}(\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^\top,\ \pi_k = \frac{N_k}{N}. \tag{34}$$

As each iteration guarantees the improvement from the last iteration, the EM algorithm always yields a local optimum.

4.2 K-Means

K-means algorithm divides a given dataset $\boldsymbol{X}$ into K clusters where K is a control parameter. This algorithm minimize the following criterion:

$$\mathcal{L}(\boldsymbol{r}, \boldsymbol{\mu}) = \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma_{k,n}\|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|^2. \tag{35}$$

K-means algorithm also follows the EM algorithm:

1. **E-step**: Minimize $\mathcal{L}$ with respect to $\boldsymbol{\gamma}$ by assigning each data point to the closest cluster center:
$$\gamma_{k,n} = \begin{cases} 1 \text{ if } k = \mathrm{argmin}_{k'} \|\boldsymbol{x}_n - \boldsymbol{\mu}_{k'}\|^2 \\ 0 \text{ otherwise} \end{cases}, \text{ and}$$

2. **M-step**: Minimize $\mathcal{L}$ with respect to the centroid $\boldsymbol{\mu}_k$ by MLE:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} &= 2\sum_{n=1}^{N} \gamma_{k,n}(\boldsymbol{x}_n - \boldsymbol{\mu}_k) = \mathbf{0}_D, \\ \boldsymbol{\mu}_k &= \frac{\sum_{n=1}^{N} \gamma_{k,n}\boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma_{k,n}}. \end{aligned} \tag{36}$$

The differences from GMM are (1) the responsibilities $\gamma_{k,n}$ are either 0 or 1, and (2) the covariance matrix is singular, i.e. hard assignment. The second point implies that GMM approaches the result of K-means as $\boldsymbol{\Sigma}_k$ goes to $\mathbf{0}_{D\times D}$. Note that the computational complexity in each iteration is $O(KND)$ and this algorithm also guarantees to converge to a local minimum. Furthermore, the parameter selection of K changes the result drastically, and $K = N$ leads to the kernel density estimator.

4.3 General EM Algorithm

In this section, we discuss the generalization of the EM algorithm. Given a dataset $\boldsymbol{X}$, we would like to maximize the following likelihood:

$$p(\boldsymbol{X}|\boldsymbol{\theta}) = \int p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})d\boldsymbol{Z}. \tag{37}$$

When the optimization of the complete-data likelihood $p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})$ is tractable while the direct optimization of the incomplete-data likelihood $p(\boldsymbol{X}|\boldsymbol{\theta})$ is intractable, we can think of the decomposition of the optimization problem into two subproblems: (1) to estimate the expectation with respect to the latent variable given a data point $p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}) = \mathbb{E}[\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}]$ (**E step**), and (2) to maximize the complete-data likelihood given the expectation with respect to the parameters $\boldsymbol{\theta}$ (**M step**). The incomplete-data log-likelihood is computed as:

$$\begin{aligned} \log p(\boldsymbol{X}|\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{Z}\sim q(\boldsymbol{Z})}[\log p(\boldsymbol{X}|\boldsymbol{\theta})] \ (\because p(\boldsymbol{X}|\boldsymbol{\theta}) \text{ does not depend on } \boldsymbol{Z}) \\ &= \mathbb{E}_{\boldsymbol{Z}\sim q(\boldsymbol{Z})}\left[\log \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})}\right] \ (\because \text{Bayes' theorem}) \\ &= \underbrace{\int q(\boldsymbol{Z}) \ \log \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} d\boldsymbol{Z}}_{=\mathcal{L}_{\text{ELBO}}(q,\boldsymbol{\theta})} + \underbrace{\int q(\boldsymbol{Z}) \ \log \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})} d\boldsymbol{Z}}_{=D_{\text{KL}}(q\|p)} \end{aligned} \tag{38}$$

where $q(\boldsymbol{Z})$ is an approximate distribution of the latent variable and $\mathcal{L}_{\text{ELBO}}(q, \boldsymbol{\theta})$ is the evidence lower bound (**ELBO**) of the likelihood. In E step, we get the parameters $\boldsymbol{\theta}_{\text{old}}$ and the approximate distribution $q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}})$. Then, we obtain the following:

$$\log p(\boldsymbol{X}|\boldsymbol{\theta}_{\text{old}}) = \mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}_{\text{old}}). \tag{39}$$

Now we know that the incomplete-data log-likelihood can achieve at least $\mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}_{\text{old}})$. It means that if $\mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}) \geq \mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}_{\text{old}})$ holds, the following also holds [2]:

$$\begin{aligned} \log p(\boldsymbol{X}|\boldsymbol{\theta}) &= \mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}) + D_{\text{KL}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}})\|p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})) \\ &\geq \mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}_{\text{old}}) = \log p(\boldsymbol{X}|\boldsymbol{\theta}_{\text{old}}). \end{aligned} \tag{40}$$

As the KL divergence is non-negative, the improvement in $\mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta})$ directly leads to a larger incomplete-data log-likelihood. Therefore, we maximize $\mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta})$ with respect to

[2] In principle, $\log p(\boldsymbol{X}|\boldsymbol{\theta}) \geq \log p(\boldsymbol{X}|\boldsymbol{\theta}_{\text{old}})$ can still happen even if $\mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}) \geq \mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}), \boldsymbol{\theta}_{\text{old}})$ does not hold. As it is much easier to handle only the ELBO, we stick to this solution.

$\boldsymbol{\theta}$ in M step:

$$\begin{aligned}\mathcal{L}_{\mathrm{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}_{\mathrm{old}}),\boldsymbol{\theta}) &= \int p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}_{\mathrm{old}})\ \log\frac{p(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta})}{p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}_{\mathrm{old}})}d\boldsymbol{Z} \\ &= \underbrace{\int p(\boldsymbol{Z}|\boldsymbol{X},\boldsymbol{\theta}_{\mathrm{old}})\ \log p(\boldsymbol{X},\boldsymbol{Z}|\boldsymbol{\theta})d\boldsymbol{Z}}_{\text{M-step}}\ +\ \text{const.}\end{aligned} \tag{41}$$

By repeating these two steps, we can iteratively maximize the likelihood $p(\boldsymbol{X}|\boldsymbol{\theta})$. As discussed above, since the improvement of ELBO is guaranteed, the convergence of the EM algorithm to a stationary point is also guaranteed although it does not guarantee the convergence to a local maxima. Even if closed-form solutions for both steps are not available, **generalized EM** algorithm enables the iterative optimization of the likelihood.

5 Non-Parametric Methods

While parametric models and GMM have limited representational capacity, non-parametric methods dynamically change their capacities according to the number of data points N, and thus are unbiased as N goes to infinity. In this section, we discuss non-parametric density estimation methods.

The advantages of non-parametric methods are (1) simplicity and (2) to be able to capture general densities. On the other hand, the computational and memory complexity increase linearly in the number of data points and it often suffers from overfitting.

5.1 Density Estimation Methods

5.1.1 Histogram

The histogram is the most basic probability density estimation method. We first divide the parameter space $\mathcal{X}$ into bins and count the number of occurrences in each bin. In other words, we define each bin as Δ_i, and we assume that the parameter space $\mathcal{X}$ is covered by the union of bins $\mathcal{X} = \bigcup_i \Delta_i$ and each bin does not intersect $\Delta_i \cap \Delta_j = \emptyset$. Then the probability density $p(\boldsymbol{x})$ is computed as:

$$p(\boldsymbol{x}) = \frac{n_i}{NV(\Delta_i)}\ \left(\because \int p(\boldsymbol{x})d\boldsymbol{x} = \sum_i p(\boldsymbol{x})V(\Delta_i) = 1, \sum_i n_i = N\right) \tag{42}$$

where n_i is the number of data points that belong to the i-th bin and $V : \mathbb{R}^D \rightarrow \mathbb{R}_+$ is a volume measure. The advantages of the histogram are that (1) we can discard samples once we check to which bin they belong, and (2) the algorithm is simple. On the other hand, the choice of the bin width affects the estimation. While small bins can capture detailed information, it causes overfitting. In contrast, large bins prevent overfitting; however, it loses local details. Furthermore, although some samples might be close to boundaries, especially in high dimensions, the histogram counts each sample towards only one bin. This issue gives rise to the loss of information in some regions. To mitigate this issue, one might introduce weighted counting.

5.1.2 Kernel Density Estimation (KDE)

Kernel density estimation (KDE) represents the density as the sum of kernel functions as follows:

$$p(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} k(\boldsymbol{x}_i, \boldsymbol{x}). \tag{43}$$

Histogram is a special case of KDE and we can choose the kernel function freely as long as it satisfies $k(\boldsymbol{x},\cdot) \geq 0, \int k(\boldsymbol{x},\cdot)d\boldsymbol{x} = 1$ and is sufficiently smooth. The benefits of KDE are to (1) **not need**

EM algorithm, (2) be able to achieve **high accuracy** and (3) require **only one hyperparameter** (bandwidth). On the other hand, we have to store all the training samples and the bandwidth cannot locally adapt to the data. Additionally, since it is a low-biased model, it is likely to overfit easily. For this reason, we need to perform cross validation to choose the robust bandwidth. One example of loss measure is the following expected leave-one-out negative log-likelihood:

$$\begin{aligned}\mathbb{E}[\mathcal{L}(h|\boldsymbol{x})] &= \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(h|\boldsymbol{x}_i) \\ \text{where } \mathcal{L}(h|\boldsymbol{x}_i) &= -\log\biggl(\frac{1}{N-1}\sum_{j\neq i}k(\boldsymbol{x}_i,\boldsymbol{x}_j;h)\biggr).\end{aligned} \tag{44}$$

We optimize this metric either via direct search or gradient descent. Since the Epanechnikov kernel has short tails and assures convergence, we often use this kernel.

An alternative loss measure is the following integrated squared error:

$$\begin{aligned}\mathcal{L}(h) &= \int (p(\boldsymbol{x}) - \hat{p}(\boldsymbol{x}))^2 d\boldsymbol{x} \\ &= \underbrace{\int p(\boldsymbol{x})^2 d\boldsymbol{x}}_{\text{const}} - 2\underbrace{\int \hat{p}(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}}_{\mathbb{E}[\hat{p}(\boldsymbol{x})]\simeq\frac{1}{N}\sum_{i=1}^{N}\hat{p}(\boldsymbol{x}_i)} + \int \hat{p}(\boldsymbol{x})^2 d\boldsymbol{x}.\end{aligned} \tag{45}$$

This loss measure guarantees the convergence.

Additionally, in most real-world applications, we are interested in the maximum density rather than the density function. The mean-shift algorithm realizes the identification of the local maxima. Considering the Gaussian kernel and the learning rate $\alpha = h^2$, then we obtain the following update:

$$\boldsymbol{x}_{t+1} = \frac{\sum_{i=1}^{N}\boldsymbol{x}_i k(\boldsymbol{x}_i,\boldsymbol{x}_t)}{\sum_{i=1}^{N}k(\boldsymbol{x}_i,\boldsymbol{x}_t)} \tag{46}$$

where the equation is derived by the gradient ascent using $\frac{\partial \log p(\boldsymbol{x}_t)}{\partial \boldsymbol{x}}$.

5.1.3 K-Nearest Neighbors Method (KNN)

As discussed, KDE does not adapt the bandwidth, and thus the density is not optimal. In dense regions, it does not show the optimal density due to over-smoothing. In sparse regions, it does not show the optimal density due to overfitting. The following KNN is a remedy for this problem.

$$p(\boldsymbol{x}) = \frac{K}{NV(\boldsymbol{x})}\left(\because p(\boldsymbol{x}|\mathcal{C}_i) = \frac{K_i}{N_iV(\boldsymbol{x})}, p(\mathcal{C}_i) = \frac{N_i}{N}\right) \tag{47}$$

where $V(\boldsymbol{x})$ is the minimum hypersphere volume which includes the K-nearest neighbors and K plays a role of smoothing. In contrast to KDE, KNN yields **noisy estimates in dense regions** and large K leads to more smoothing effect and biased estimate. Bayesian formulation for KNN is the following:

$$\begin{aligned}\text{posterior} &= \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \\ p(\mathcal{C}_i|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}|\mathcal{C}_i)p(\mathcal{C}_i)}{\sum_j p(\boldsymbol{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{K_i/N_iV(\boldsymbol{x}) \times N_i/N}{K/NV(\boldsymbol{x})} = \frac{K_i}{K}\end{aligned} \tag{48}$$

where K_i is the number of *neighbors* that belong to the i-th class and N_i is the number of *data points* that belong to the i-th class. From this equation, the posterior of the i-th class given the data point $\boldsymbol{x}$ is $p(\mathcal{C}_i|\boldsymbol{x}) = K_i/K$. KNN is a straightforward way of classification; however, the performance is sensitive to the parameter selection of K and KNN often suffers from noisy estimates in dense regions.

5.1.4 Adaptive KDE

While KDE provides accurate density in dense regions, it underestimates density in sparse regions due to the fixed bandwidth. On the other hand, KNN provides smoothing effect in sparse regions. This fact gives rise to the following adaptive KDE:

$$\begin{aligned} p(\boldsymbol{x}) &= \frac{1}{N(2\pi)^{D/2}|\boldsymbol{\Sigma}(\boldsymbol{x})|^{1/2}} \sum_{i=1}^{N} \exp\biggl(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x})\boldsymbol{\Sigma}(\boldsymbol{x})^{-1}(\boldsymbol{x}_i - \boldsymbol{x})^\top\biggr), \\ \boldsymbol{\Sigma}(\boldsymbol{x}) &= \frac{1}{K}\sum_{\boldsymbol{x}_i \in \mathcal{S}} (\boldsymbol{x}_i - \boldsymbol{x})(\boldsymbol{x}_i - \boldsymbol{x})^\top \text{ ($\mathcal{S}$ is a set of K-nearest neighbors).} \end{aligned} \tag{49}$$

Since the variance is calculated based on the K-nearest neighbors, this KDE provides more stable estimates both in sparse and dense regions. Note that we call the kernel in the adaptive KDE Anisotropic kernel and determine the optimal K via cross validation.

5.2 Space Subdivision

Since KNN requires the comparison of distances to each data point, the time complexity increases linearly to the number of data points. However, if we subdivide the parameter space beforehand, we can achieve sublinear time complexity. We list major space subdivision methods:

1. **K-d trees**: Subdivide the space along each coordinate using CART algorithm and optionally allocate weights to points close to boundaries (**Spill trees**),

2. **Tree-structured vector quantization (TSVQ)**: Subdivide the space with linear lines along arbitrary directions using K-means with $K = 2$, and

3. **Randomized trees**: Build multiple trees and consider the union of all points obtained from each tree as neighbors.

Each method aims to filter a group of points that are close to a point of interest. Obviously, if we increase the accuracy of the inference, the searching takes more time. While the quickest algorithm is the K-d tree, it suffers from the curse of dimensionality. The other two algorithms behave better in high dimensions. Another solution is the spill trees that consider a margin from each boundary and view points in the margin belonging to both subspaces; however, if we make this margin too large, the memory requirement grows exponentially. Notice that each tree requires $O(N \log N)$ to build and $O(\log N)$ for inference if we view the split procedure as $O(1)$.

6 Regression

6.1 Preliminaries (Causal Relation of Variables)

First, we consider three types of causal relations (called **triple**):

1. **Head-to-tail** ($A \rightarrow B \rightarrow C$): $p(A, B, C) = p(A)p(B|A)p(C|B)$,

2. **Tail-to-tail** ($A \leftarrow B \rightarrow C$): $p(A, B, C) = p(A|B)p(C|B)p(B)$, and

3. **Head-to-head** ($A \rightarrow B \leftarrow C$): $p(A, B, C) = p(B|A, C)p(A)p(C)$.

Then if $p(A, C|B) = p(A|B)p(C|B)$ or $p(B|A, C) = p(B|A)$ holds, A and C are said to be *conditionally independent* given B. When we consider $p(A, C|B)$, head-to-tail and tail-to-tail type exhibit conditional independence while head-to-head type does not. If two nodes are connected with an edge, those two nodes are obviously conditionally dependent. The test of the conditional independence of two nodes is called **d-separation**. The test between Nodes u and v is performed by checking the following:

1. Enumerate all (**undirected**) paths from u to v,
2. Divide each path into a set of triples and check whether all triples are active, and
3. Return "true" if there is at least one path; otherwise "false".

Note that a triple is active if and only if:

1. **Head-to-tail** ($A \to B \to C$): B is unobserved,
2. **Tail-to-tail** ($A \leftarrow B \to C$): B is unobserved, and
3. **Head-to-head** ($A \to B \leftarrow C$): B or one of its descendants (in the directed graph) is observed.

Throughout this section, we assume $p(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) = p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{X})p(\boldsymbol{w})$ (head-to-head), and thus the posterior of $\boldsymbol{w}$ given $\boldsymbol{y}$ is not conditionally independent of $\boldsymbol{X}$ although it is in a prior.

6.2 Bayesian Linear Regression

Bayesian linear regression gives an uncertainty measure to the linear regression. The linear regression assumes that the output $\boldsymbol{y}$ follows the Gaussian distribution with a fixed variance σ and performs MLE to estimate $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})$. On the other hand, Bayesian linear regression computes the posterior distribution of weights $p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})$ using Bayes' theorem as follows:

$$\begin{aligned} \text{posterior} &= \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \\ p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) &= \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})}{\int p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w})d\boldsymbol{w}}. \end{aligned} \tag{50}$$

Since the denominator $p(\boldsymbol{y}|\boldsymbol{X})$ does not depend on the weight $\boldsymbol{w}$ and the denominator generally requires a complicated integral, we use MAP estimation [3]. In the MAP estimation, we maximize the following:

$$\begin{aligned} \text{posterior} &\propto \text{likelihood} \times \text{prior}, \\ p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) &\propto p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w}). \end{aligned} \tag{51}$$

Since the conjugate prior for the likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{w}, \sigma^2 \boldsymbol{I})$ [4] with Gaussian distribution with unknown mean is Gaussian distribution, the following holds:

$$\begin{aligned} \text{Prior}: &\ \ \boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma}_{\text{prior}}), \\ \text{Posterior}: &\ \ p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) \sim \mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}}) \end{aligned} \tag{52}$$

where $\boldsymbol{\Sigma}_{\text{prior}}$ is a covariance matrix of the prior $p(\boldsymbol{w})$ that controls the regularization effect and are chosen via cross validation, and

$$\boldsymbol{\mu}_{\text{post}} = \frac{1}{\sigma^2}\boldsymbol{\Sigma}_{\text{post}}\boldsymbol{X}^\top \boldsymbol{y}, \boldsymbol{\Sigma}_{\text{post}} = \left(\frac{1}{\sigma^2}\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{\Sigma}_{\text{prior}}^{-1}\right)^{-1} \tag{53}$$

are the parameters of the posterior. This result is directly derived by transforming Eq. (51). Using weights sampled from the posterior, the prediction of Bayesian linear regression is computed as follows:

$$\begin{aligned} p(y|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) &= \int p(y|\boldsymbol{x}, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})d\boldsymbol{w} \\ &= \mathcal{N}(\boldsymbol{\mu}_{\text{post}}^\top \boldsymbol{x}, \boldsymbol{x}^\top \boldsymbol{\Sigma}_{\text{post}}\boldsymbol{x}). \end{aligned} \tag{54}$$

Note that we can replace $\boldsymbol{X}$ with a set of non-linear mapping Φ such as $\Phi = [1, x, x^2, ..., x^d]$. When we use the Laplace prior, we can promote **the sparsity** of the model although a closed-form solution will not be available anymore.

[3] MLE is equivalent to MAP estimation with the uniformly distributed marginal likelihood.

[4] σ^2 is estimated via MLE, i.e. $\sigma^2 := \sigma^2_{\text{MLE}} = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2/N$

6.3 Evidence Approximation

Since Bayesian linear regression requires control parameters and they obviously affect the prediction, we need to determine those parameters via cross validation. However, cross validation is demanding in this case. For this reason, we consider the marginalization of the control parameters and the (approximated) marginalization is performed via so-called **evidence approximation**. We define $\boldsymbol{\Sigma}_{\text{prior}}^{-1} = \alpha \boldsymbol{I}$, $1/\sigma^2 = \beta$, and $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$ in this section. Then the marginalization is computed as:

$$p(y|\boldsymbol{x}, \mathcal{D}) = \int p(y|\boldsymbol{x}, \boldsymbol{w}, \beta) p(\boldsymbol{w}|\alpha, \beta, \mathcal{D}) p(\alpha, \beta|\mathcal{D}) d\boldsymbol{w} d\alpha d\beta. \tag{55}$$

Note that we use $p(y|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}, \mathcal{D}) = p(y|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}, \mathcal{D}) p(\boldsymbol{w}|\boldsymbol{\theta}, \mathcal{D}) p(\boldsymbol{\theta}|\mathcal{D})$ where $\boldsymbol{\theta}$ is a set of hyperparameters for a likelihood and a prior, and the independence of $\boldsymbol{x}$ and the conditional independence of $\mathcal{D}$ [5] for the derivation of the equation above. However, this marginalization is not feasible analytically without any assumptions. For this reason, we introduce the following two assumptions:

Assumption 1

1. *The posterior $p(\alpha, \beta|\mathcal{D})$ is sharply peaked around optimal values $\alpha^\star, \beta^\star$*

$$p(y|\boldsymbol{x}, \mathcal{D}) \simeq p(y|\boldsymbol{x}, \alpha^\star, \beta^\star, \mathcal{D}) = \int p(y|\boldsymbol{x}, \boldsymbol{w}, \beta^\star) p(\boldsymbol{w}|\alpha^\star, \beta^\star, \mathcal{D}) d\boldsymbol{w}. \tag{56}$$

2. *The prior distribution $p(\alpha, \beta)$ is a non-informative, i.e. the uniform distribution*

$$p(\alpha, \beta|\mathcal{D}) \propto p(\mathcal{D}|\alpha, \beta) p(\alpha, \beta) \propto p(\mathcal{D}|\alpha, \beta). \tag{57}$$

From those assumptions, we can reformulate the estimation as MLE of $\alpha^\star, \beta^\star$ via the maximization of the follwoing **evidence function**:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}, \alpha, \beta) &= \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}, \beta) p(\boldsymbol{w}|\alpha) d\boldsymbol{w} \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2} \left(\frac{\alpha}{2\pi}\right)^{D/2} \int \exp\left(-\frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 - \frac{\alpha}{2}\|\boldsymbol{w}\|^2\right) d\boldsymbol{w}. \end{aligned} \tag{58}$$

The optimization of the evidence function is achieved by EM algorithm that takes $\boldsymbol{w}$ as latent variables. In E step, we first compute the posterior as follows:

$$p(\boldsymbol{w}|\alpha, \beta, \mathcal{D}) \propto \mathcal{N}(\boldsymbol{w}|\boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}}) \coloneqq \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \tag{59}$$

where $\mu_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}}$ are identical to those in Eq. (53). Since the following holds,

$$\frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \frac{\alpha}{2}\|\boldsymbol{w}\|^2 = \frac{1}{2}(\boldsymbol{w} - \boldsymbol{\mu})^\top \boldsymbol{\Lambda}(\boldsymbol{w} - \boldsymbol{\mu}) + \underbrace{\frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}\|^2 + \frac{\alpha}{2}\|\boldsymbol{\mu}\|^2}_{\text{const w.r.t. } \boldsymbol{w}}, \tag{60}$$

we can transform the integral in Eq. (58) as follows:

$$\begin{aligned} \int \exp\left(-\frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 - \frac{\alpha}{2}\|\boldsymbol{w}\|^2\right) d\boldsymbol{w} &= \exp\left(-\frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}\|^2 - \frac{\alpha}{2}\|\boldsymbol{\mu}\|^2\right) \int \exp\left(-\frac{1}{2}(\boldsymbol{w} - \boldsymbol{\mu})^\top \boldsymbol{\Lambda}(\boldsymbol{w} - \boldsymbol{\mu})\right) d\boldsymbol{w} \\ &= \frac{(2\pi)^{D/2}}{|\boldsymbol{\Lambda}|^{1/2}} \exp\left(-\frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}\|^2 - \frac{\alpha}{2}\|\boldsymbol{\mu}\|^2\right). \end{aligned} \tag{61}$$

[5] $p(\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{w}, \mathcal{D}) = p(\boldsymbol{x}) p(\boldsymbol{\theta}, \boldsymbol{w}, \mathcal{D})$ and $p(y|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}) = p(y|\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{\theta}, \mathcal{D})$.

Therefore, the log-likelihood in E step is computed as follows:

$$\log p(\boldsymbol{y}|\boldsymbol{x}, \alpha, \beta) = \frac{N}{2} \log \frac{\beta}{2\pi} + \frac{D}{2} \log \alpha - \frac{1}{2} \log |\boldsymbol{\Lambda}| - \frac{\beta}{2} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}\|^2 - \frac{\alpha}{2} \|\boldsymbol{\mu}\|^2. \tag{62}$$

In M step, we maximize the log-likelihood and the solution for MLE is the following:

$$\alpha^\star = \frac{\gamma}{\|\boldsymbol{\mu}\|^2},\ \beta^\star = \left(\frac{1}{N - \gamma} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\mu}\|^2 \right)^{-1} \text{ where } \ \gamma = \sum_{i=1}^{D} \frac{\lambda_i}{\lambda_i + \alpha_{\text{old}}}, \tag{63}$$

and λ_i is the eigenvalue of $\beta \boldsymbol{X}^\top \boldsymbol{X}$. Since λ_i are close to the eigenvalues of the posterior covariance matrix $\boldsymbol{\Sigma}_{\text{post}}$, λ_i is viewed as the variance of principle axes. For this reason, we can measure **the effective number of well-determined dimensions** or the dimensions not dominated by prior via γ. For example, when we have sufficient training data points, the covariance becomes large and thus γ becomes large and dominates the noise control factor α. On the other hand, when $|\lambda_i| \ll |\alpha_{\text{old}}|$, the regularization effects dominate the prediction.

7 Kernel Methods

7.1 Properties of Kernel Function

The definition of kernel function is as follows:

Definition 1
kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a similarity measure of given points $\boldsymbol{x}, \boldsymbol{x}'$. This function must satisfy the following properties:

1. **Symmetric**: $k(\boldsymbol{x}, \boldsymbol{x}') = k(\boldsymbol{x}', \boldsymbol{x})$
2. **Semi-positive definite**: $\forall n \in \mathbb{N}_{\geq 1}, \forall \boldsymbol{a} \in \mathbb{R}^n, \sum_{i=1}^{n} \sum_{j=1}^{n}, a_i a_j k(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$

We must design kernel functions to satisfy the properties. The following operations over kernel functions always yield another kernel function:

1. **Additive**: $k_1(\boldsymbol{x}, \boldsymbol{x}') + k_2(\boldsymbol{x}, \boldsymbol{x}'),\ k_1(\boldsymbol{x}_a, \boldsymbol{x}_a') + k_2(\boldsymbol{x}_b, \boldsymbol{x}_b')$
2. **Multiplication**: $ck_1(\boldsymbol{x}, \boldsymbol{x}'),\ f(\boldsymbol{x})k_1(\boldsymbol{x}, \boldsymbol{x}')f(\boldsymbol{x}'),\ k_1(\boldsymbol{x}, \boldsymbol{x}')k_2(\boldsymbol{x}, \boldsymbol{x}'),\ k_1(\boldsymbol{x}_a, \boldsymbol{x}_a')k_2(\boldsymbol{x}_b, \boldsymbol{x}_b')$
3. **Others**: $\exp(k_1(\boldsymbol{x}, \boldsymbol{x}')), \boldsymbol{x}^\top A \boldsymbol{x}$

where $c \in \mathbb{R}_{>0}$, $f(\boldsymbol{x}) : \mathcal{X} \to \mathbb{R}$ and $A \in \mathbb{R}^{D \times D}$ is a positive semi-definite matrix. Kernel functions often assume smoothness, and it controls the overestimation or underestimation. For example, **more smoothing leads to generalization and biased estimates**. Note that if a kernel function is represented as a function of $\boldsymbol{x} - \boldsymbol{x}'$, it is called **stationary kernel** and it is invariant to translation and if a kernel function is a function of $\|\boldsymbol{x} - \boldsymbol{x}'\|$, it is called **isotropic kernel** or **radial basis function** and it is invariant to translation and rotation. Stationary kernels are often not sufficiently flexible, and thus we combine other kernels and optimize all hyperparameters via the maximization of log-likelihood using gradient ascent [6]. An isotropic kernel can handle graphs and images well. The drawbacks of kernel methods are (1) poor extrapolation, (2) shrinkage to zero-mean prediction in sparse regions.

[6] Since the objective is often non-convex, we aim to find a local maximum.

7.2 Kernel Regression

When we reformulate linear regression from the Bayesian view as in Eq. (54), we obtain the following:

$$
\begin{aligned}
p(y|\boldsymbol{x}, \mathcal{D}) &= \mathcal{N}(y|\boldsymbol{\mu}_{\mathrm{post}}^\top \boldsymbol{x}, \boldsymbol{x}^\top \boldsymbol{\Sigma}_{\mathrm{post}} \boldsymbol{x}), \\
\mathbb{E}[y] &= \frac{1}{\sigma^2} \boldsymbol{y}^\top \underbrace{\boldsymbol{X} \boldsymbol{\Sigma}_{\mathrm{post}}^\top \boldsymbol{x}}_{\text{Dot product}} .
\end{aligned} \tag{64}
$$

By replacing the dot product $1/\sigma^2 \boldsymbol{X}\boldsymbol{\Sigma}_{\mathrm{post}}^\top \boldsymbol{x}$ with a summation of a kernel function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$, the expression is reformulated as follows:

$$
y(\boldsymbol{x}) = \sum_{i=1}^{n} y_i k(\boldsymbol{x}_i, \boldsymbol{x}) \tag{65}
$$

where $(\boldsymbol{x}_i, y_i)$ is the i-th training data point. Note that this procedure works for non-linearly mapped feature space $\Phi(\boldsymbol{X}) \in \mathbb{R}^{N \times D}$ instead of $\boldsymbol{X}$ and the kernel function corresponds to the feature set is called the **equivalent kernel**. Since this representation does not have the basis function and the number of operations depends on the number of data points, we can reduce the computational complexity (**kernel trick**) in the case of $N < D$ where D might be potentially infinity.

7.3 Regression Using Kernel Density Estimation

Since the goal of regression tasks is to estimate the conditional distribution $p(y|\boldsymbol{x})$, we can calculate it from the following:

$$
\begin{aligned}
p(y|\boldsymbol{x}) &= \frac{p(\boldsymbol{x}, y)}{\int p(\boldsymbol{x}, y) dy} \\
p(\boldsymbol{x}, y) &= \frac{1}{N} \sum_{i=1}^{N} k(\{\boldsymbol{x}, y\}, \{\boldsymbol{x}_i, y_i\})
\end{aligned} \tag{66}
$$

For the sake of simplicity, we define $\boldsymbol{u} = \{\boldsymbol{x}, y\}$. Using this formulation, the predictive mean is computed as:

$$
\begin{aligned}
\hat{y}(\boldsymbol{x}) = \int y p(y|\boldsymbol{x}) dy &= \frac{\sum_{i=1}^N \int y k(\boldsymbol{u}, \boldsymbol{u}_i) dy}{\sum_{i=1}^N \int k(\boldsymbol{u}, \boldsymbol{u}_i) dy} \ (\because \text{The denominator does not depend on } y) \\
&= \frac{\sum_{i=1}^N \biggl(\int (y - y_i) k(\boldsymbol{u}, \boldsymbol{u}_i) dy + \int y_i k(\boldsymbol{u}, \boldsymbol{u}_i) dy \biggr)}{\sum_{i=1}^N \int k(\boldsymbol{u}, \boldsymbol{u}_i) dy} \\
&= \frac{\sum_{i=1}^N y_i \int k(\boldsymbol{u}, \boldsymbol{u}_i) dy}{\sum_{i=1}^N \int k(\boldsymbol{u}, \boldsymbol{u}_i) dy} \ (\because \text{Assume zero mean kernel}) \\
&= \frac{\sum_{i=1}^N y_i g(\boldsymbol{x}, \boldsymbol{x}_i)}{\sum_{i=1}^N g(\boldsymbol{x}, \boldsymbol{x}_i)} \ \biggl(\text{Define } g(\boldsymbol{x}, \boldsymbol{x}') \coloneqq \int k(\boldsymbol{u}, \boldsymbol{u}_i) dy \biggr) \\
&= \sum_{i=1}^N w(\boldsymbol{x}, \boldsymbol{x}_i) y_i \ \biggl(\text{Define } w(\boldsymbol{x}, \boldsymbol{x}_i) \coloneqq \frac{g(\boldsymbol{x}, \boldsymbol{x}_i)}{\sum_{j=1}^N g(\boldsymbol{x}, \boldsymbol{x}_j)} \biggr).
\end{aligned} \tag{67}
$$

This model is called **Nadaraya-Watson model**. Intuitively, this model weights each y_i with a weight $w(\boldsymbol{x}, \boldsymbol{x}_i)$ that measures the similarity between $\boldsymbol{x}$ and $\boldsymbol{x}_i$. Although the conditional distribution $p(y|\boldsymbol{x})$ is a multimodal distribution, we assume that the prediction $\hat{y}(\boldsymbol{x})$ follows a unimodal Gaussian noise.

7.4 Gaussian Process (GP) Regressor

7.4.1 Formulation

First, we assume that the observation $y \in \mathbb{R}$ given a data point $\boldsymbol{x} \in \mathbb{R}^D$ follows:

$$y = f(\boldsymbol{x}) + \epsilon = \boldsymbol{w}^\top \Phi(\boldsymbol{x}) + \epsilon,\ \epsilon \sim \mathcal{N}(0, \lambda). \tag{68}$$

where $\boldsymbol{w} \in \mathbb{R}^\infty$ is weights of the non-linear regression, $\Phi = \{\phi_i\}_{i=1}^\infty$ is a set of functions, and $\phi_i : \mathbb{R}^D \rightarrow \mathbb{R}$ is a (non-linear) mapping. Since it is, in principle, impossible to estimate the optimal $\boldsymbol{w}$, Gaussian process (GP) instead estimates the posterior $p(y|\boldsymbol{x}, \mathcal{D})$ using kernel trick. In this section, we show the transformation to derive the posterior formula.

Suppose we have a set of observations $\mathcal{D} := \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$, and we define $\boldsymbol{X} := [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]^\top$ and $\boldsymbol{y} := [y_1, \dots, y_N]^\top$. From Eq. (68) and , the likelihood is defined as $p(\boldsymbol{y}|\boldsymbol{f}) = p(\boldsymbol{y}|\boldsymbol{\Phi}, \boldsymbol{w}, \lambda) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{\Phi}\boldsymbol{w}, \lambda \boldsymbol{I}_N)$ where $\boldsymbol{\Phi} := \Phi(\boldsymbol{X}) \in \mathbb{R}^{N \times \infty}$, $\boldsymbol{w}$ is sampled from $\mathcal{N}(\boldsymbol{0}_\infty, \sigma^2 \boldsymbol{I}_\infty)$, and $\sigma^2 \in \mathbb{R}_+$ is variance of the prior. The moments of the likelihood are computed as follows:

$$\begin{aligned}
&\mathbb{E}[\boldsymbol{\Phi}\boldsymbol{w}] = \boldsymbol{\Phi}\mathbb{E}[\boldsymbol{w}] = \boldsymbol{0}_N, \\
&\mathbb{V}[\boldsymbol{\Phi}\boldsymbol{w}] = \mathbb{E}[\boldsymbol{\Phi}\boldsymbol{w}(\boldsymbol{\Phi}\boldsymbol{w})^\top] = \boldsymbol{\Phi}\mathbb{E}[\boldsymbol{w}\boldsymbol{w}^\top]\boldsymbol{\Phi}^\top = \boldsymbol{\Phi}(\sigma^2 \boldsymbol{I}_\infty)\boldsymbol{\Phi}^\top = \sigma^2 \boldsymbol{\Phi}\boldsymbol{\Phi}^\top = \boldsymbol{K}_N.
\end{aligned} \tag{69}$$

Note that replacing $\sigma^2 \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$, which is the product of infinite matrices, with a kernel matrix $\boldsymbol{K}_N \in \mathbb{R}^{N \times N}$ is called kernel trick. From Eq. (69) and the definition of GP, i.e. $\boldsymbol{f}$ follows the multivariate Gaussian distribution, the prior is $p(\boldsymbol{f}) = p(\boldsymbol{\Phi}\boldsymbol{w}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}_N, \boldsymbol{K}_N)$. Using the likelihood and the prior, the marginal likelihood $p(\boldsymbol{y})$ can be computed as:

$$p(\boldsymbol{y}) = \int p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f})d\boldsymbol{f} = \int \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \lambda \boldsymbol{I}_N)\mathcal{N}(\boldsymbol{f}|\boldsymbol{0}_N, \boldsymbol{K}_N)d\boldsymbol{f} = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}_N, \boldsymbol{K}_N + \lambda \boldsymbol{I}). \tag{70}$$

In the same fashion, the predictive distribution of y_{N+1} at a new data point $\boldsymbol{x}_{N+1}$ can be calculated:

$$\begin{aligned}
p\left(\begin{bmatrix} \boldsymbol{y} \\ y_{N+1} \end{bmatrix}\right) &= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_{N+1} \end{bmatrix} \;\middle|\; \boldsymbol{0}_{N+1}, \begin{bmatrix} \boldsymbol{K}_N & \boldsymbol{k}_{N+1} \\ \boldsymbol{k}_{N+1}^\top & k(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+1}) \end{bmatrix} + \lambda \boldsymbol{I}_{N+1}\right) \\
&:= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_{N+1} \end{bmatrix} \;\middle|\; \boldsymbol{0}_{N+1}, \begin{bmatrix} \boldsymbol{C}_N & \boldsymbol{k}_{N+1} \\ \boldsymbol{k}_{N+1}^\top & c_{N+1} \end{bmatrix}\right) \\
&= \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_{N+1} \end{bmatrix} \;\middle|\; \boldsymbol{0}_{N+1}, \boldsymbol{C}_{N+1}\right),
\end{aligned} \tag{71}$$

where $\boldsymbol{k}_{N+1} = [k(\boldsymbol{x}_1, \boldsymbol{x}_{N+1}), \cdots, k(\boldsymbol{x}_N, \boldsymbol{x}_{N+1})]^\top$. Since both $p(\boldsymbol{y})$ and $p(y_{N+1})$ follow the Gaussian distribution, the conditional distribution is computed as follows:

$$p(y_{N+1}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{k}_{N+1}^\top \boldsymbol{C}_N^{-1}\boldsymbol{y}, c_{N+1} - \boldsymbol{k}_{N+1}^\top \boldsymbol{C}_N^{-1}\boldsymbol{k}_{N+1}) \tag{72}$$

where the transformation uses the formula $p(\boldsymbol{x}_a|\boldsymbol{x}_b) = \mathcal{N}(\boldsymbol{\mu}_a + \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}(\boldsymbol{x}_b - \boldsymbol{\mu}_b), \boldsymbol{\Sigma}_{aa} - \boldsymbol{\Sigma}_{ab}\boldsymbol{\Sigma}_{bb}^{-1}\boldsymbol{\Sigma}_{ba})$ given $p(\boldsymbol{x}_a) = \mathcal{N}(\boldsymbol{x}_a|\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a)$, $p(\boldsymbol{x}_b) = \mathcal{N}(\boldsymbol{x}_b|\boldsymbol{\mu}_b, \boldsymbol{\Sigma}_b)$, and

$$p\left(\begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_a \\ \boldsymbol{x}_b \end{bmatrix} \;\middle|\; \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{aa}\boldsymbol{\Sigma}_{ab} \\ \boldsymbol{\Sigma}_{ba}\boldsymbol{\Sigma}_{bb} \end{bmatrix}\right). \tag{73}$$

Note that if all non-diagonal elements of $\boldsymbol{K}_N$ are close to zero, the prediction overfits the training data and if most elements in $\boldsymbol{K}_N$ have similar values, it implies that the predictions are biased due to over-smoothing. Furthermore, the following fact about the predictive mean of a given point $\boldsymbol{x}$ is known to be the representor theorem:

$$\mu(\boldsymbol{x}) = \sum_{n=1}^N \alpha_n k(\boldsymbol{x}, \boldsymbol{x}_n) \tag{74}$$

where the weights are $\boldsymbol{\alpha} = \boldsymbol{C}_N^{-1}\boldsymbol{y}$. Eq. (74) can be derived also based on the least squares estimate perspective. Suppose we would like to minimize the following loss function:

$$\mathcal{L}(\boldsymbol{y}, \boldsymbol{\Phi w}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{\Phi w})^\top(\boldsymbol{y} - \boldsymbol{\Phi w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2. \tag{75}$$

As the optimal point exists at stationary points, we take the derivative of the loss function with respect to $\boldsymbol{w}$ and substitute its value with zero:

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\Phi w})}{\partial \boldsymbol{w}} &= -\boldsymbol{\Phi}^\top \boldsymbol{y} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi w} + \lambda \boldsymbol{w}, \\ \textbf{Stationarity: } \boldsymbol{w}^\star &= \boldsymbol{\Phi}^\top \frac{1}{\lambda}(\boldsymbol{y} - \boldsymbol{\Phi w}). \end{aligned} \tag{76}$$

Furthermore, let $\frac{1}{\lambda}(\boldsymbol{y} - \boldsymbol{\Phi w})$ be a dual variable $\boldsymbol{\alpha}$. Then, the dual problem of the original optimization problem is to maximize the following function with respect to $\boldsymbol{\alpha}$:

$$\begin{aligned} \mathcal{L}(\boldsymbol{y}, \boldsymbol{\alpha}) &= \frac{1}{2}(\boldsymbol{y} - \underbrace{\boldsymbol{\Phi\Phi}^\top}_{=\boldsymbol{K}_N}\boldsymbol{\alpha})^\top(\boldsymbol{y} - \boldsymbol{\Phi\Phi}^\top\boldsymbol{\alpha}) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top\boldsymbol{\Phi\Phi}^\top\boldsymbol{\alpha} \\ &= \frac{1}{2}(\boldsymbol{y} - \boldsymbol{K}_N\boldsymbol{\alpha})^\top(\boldsymbol{y} - \boldsymbol{K\alpha}) + \frac{\lambda}{2}\boldsymbol{\alpha}^\top\boldsymbol{K}_N\boldsymbol{\alpha}. \end{aligned} \tag{77}$$

In the same fashion, the optimal $\boldsymbol{\alpha}$ is computed as:

$$\begin{aligned} \frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} &= -\boldsymbol{K}_N\boldsymbol{y} + \boldsymbol{K}_N^2\boldsymbol{\alpha} + \lambda\boldsymbol{K}_N\boldsymbol{\alpha}, \\ \textbf{Stationarity: } \boldsymbol{\alpha}^\star &= (\boldsymbol{K}_N + \lambda\boldsymbol{I}_N)^{-1}\boldsymbol{y} = \boldsymbol{C}_N^{-1}\boldsymbol{y}. \end{aligned} \tag{78}$$

This result agrees with $\boldsymbol{\alpha}$ in Eq. (74).

7.4.2 Bottleneck of Training and Its Solutions

The bottleneck of the training is the matrix inversion that incurs the time complexity of $O(N^3)$. For inference, each point requires the time complexity of $O(N^2)$. If the dimension D is less than N, non-linear regression will be more efficient. Since the matrix inversion is not feasible when N is large, we reduce the time complexity by considering $\boldsymbol{K}_N$ as a sparse matrix. Other options are **Bayesian committees** and **Nyström approximation**. Bayesian committees combine estimates on different subsets of size $M(< N)$ and assume that $\boldsymbol{K}_{M\times M} \simeq \boldsymbol{K}_{M\times N}\mathrm{diag}[\theta_1, \cdots, \theta_N]\boldsymbol{K}_{N\times M} \in \mathbb{R}^{M\times M}$. The estimation of $\boldsymbol{\theta}$ costs $O(M^3)$ and the inference costs $O(NM^2)$ due to the matrix multiplication. Nyström approximation exploits the low-rank approximation $\boldsymbol{K}_{N\times N} \simeq \boldsymbol{K}_{N\times M}\boldsymbol{K}_{M\times M}^{-1}\boldsymbol{K}_{M\times N}$ and it is guaranteed that there is a kernel matrix $\boldsymbol{K}_{M\times M}$ that satisfies this approximation if $\mathrm{rank}(\boldsymbol{K}_{N\times N}) = M$. The eigendecomposition of $K_{M\times M}$ costs $O(M^3)$ and the computations of eigenvectors costs $O(NMP)$ where P is the number of eigenvectors to use. The overall time complexity is $O(M^3 + NMP)$.

7.4.3 Automatic Relevance Determination (ARD)

Although each dimension usually has different importances in function approximation, stationary kernels handle all the dimensions equally due to the same bandwidth for all the dimensions. By using automatic relevance determination (ARD), we can address this problem. For example, when we use the Gaussian kernel, we can think of the following formulation for ARD:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \theta_0 \exp\biggl(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^\top \mathrm{diag}[\theta_1, \cdots, \theta_D](\boldsymbol{x} - \boldsymbol{x}')\biggr). \tag{79}$$

In ARD, we would like to optimize $\boldsymbol{\theta} = [\theta_1, \dots, \theta_D]$ by maximizing the following marginal log-likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}) = -\frac{1}{2}\log|\boldsymbol{C}_N(\boldsymbol{\theta})| - \frac{1}{2}\boldsymbol{y}^\top\boldsymbol{C}_N(\boldsymbol{\theta})^{-1}\boldsymbol{y} - \frac{N}{2}\log 2\pi. \tag{80}$$

Recall that the marginal likelihood $p(\boldsymbol{y})$ is defined in Eq. (70). We can maximize the marginal log-likelihood by using the gradient with respect to the hyperparameters:

$$\frac{\partial}{\partial \theta_d} \log p(\boldsymbol{y}|\boldsymbol{\theta}) = -\frac{1}{2}\mathrm{Tr}\left(\boldsymbol{C}_N^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}_N(\boldsymbol{\theta})}{\partial \theta_d}\right) + \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{C}_N^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}_N(\boldsymbol{\theta})}{\partial \theta_d}\boldsymbol{C}_N^{-1}(\boldsymbol{\theta})\boldsymbol{y}. \tag{81}$$

The computational complexity of the optimization is dominated by the inversion of $\boldsymbol{C}_N(\boldsymbol{\theta})$, which costs $O(N^3)$. Once the optimization completes and $\boldsymbol{C}_N^{-1}(\boldsymbol{\theta})$ is computed, the inference requires $O(N^2)$.

7.5 Mixture of Regressors

Regression task usually supports only the Gaussian distribution and cannot handle multimodal distributions; however, many real-world applications, such as the prediction of traffic at a junction have multimodal distributions. Such prediction is realized by the mixture of regressors:

$$p(y|\boldsymbol{f}, \boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(y|\boldsymbol{w}_k^\top \Phi(\boldsymbol{x}), \lambda) \tag{82}$$

where $\boldsymbol{\theta}$ is a set of all hyperparameters $\lambda, \boldsymbol{w}_k, \pi_k$ and $\boldsymbol{f} = [\boldsymbol{w}_1^\top \Phi(\boldsymbol{x}), \dots, \boldsymbol{w}_K^\top \Phi(\boldsymbol{x})]$. As in GMM, we apply the EM algorithm, which again takes $\boldsymbol{z}$ as latent variables, to infer the optimal parameters. We first define $f_{k,i} \coloneqq \boldsymbol{w}_k^\top \Phi(\boldsymbol{x}_i)$. E-step evaluates the posterior of the latent variables:

$$\gamma_{k,i} = \mathbb{E}[z_{k,i}|\boldsymbol{x}_i] = p(z_{k,i} = 1|\boldsymbol{x}_i, \boldsymbol{\theta}_{\mathrm{old}}) = \frac{\pi_k \mathcal{N}(y_i|f_{k,i}, \lambda)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(y_i|f_{k,i}, \lambda)} \tag{83}$$

M-step maximizes the expectation of the complete-data likelihood given definitions $\boldsymbol{F} \in \mathbb{R}^{N\times K}$ and $\boldsymbol{F}_{i,k} = f_{k,i} \coloneqq \boldsymbol{w}_k^\top \Phi(\boldsymbol{x}_i)$:

$$\begin{aligned}
&\log p(\boldsymbol{y}|\boldsymbol{F}, \boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{z}}[\log p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{F}, \boldsymbol{\theta})] = \sum_{i=1}^{N} \log\left(\sum_{k=1}^{K} \pi_k \mathcal{N}(y_i|f_{k,i}, \lambda)\right),\\
&\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{F}, \boldsymbol{\theta})}{\partial \boldsymbol{w}_k} = \sum_{i=1}^{N} \frac{\gamma_{k,i}}{\lambda}(y_i - f_{k,i})\Phi(\boldsymbol{x}_i) = \boldsymbol{0},\\
&\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{F}, \boldsymbol{\theta})}{\partial \lambda} = \sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{k,i}\left(\frac{1}{2\lambda^2}(f_{k,i} - y_i)^2 - \frac{1}{2\lambda}\right) = 0 \Rightarrow \lambda = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} \gamma_{k,i}\|\boldsymbol{y} - \Phi(\boldsymbol{X})\boldsymbol{w}_k\|^2,\\
&\quad \frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \pi_k} = 0 \Rightarrow \pi_k = \frac{1}{N}\sum_{i=1}^{N} \gamma_{k,i} \text{ (Same KKT conditions in Eq. (32)).}
\end{aligned} \tag{84}$$

When we define a responsibility matrix as $\boldsymbol{R}_k = \mathrm{diag}[\gamma_{k,1}, \dots, \gamma_{k,N}]$ and a feature matrix $\Phi(\boldsymbol{X}) \coloneqq \boldsymbol{\Phi} \in \mathbb{R}^{N\times\infty}$, the derivative with respect to $\boldsymbol{w}_k$ is transformed as follows:

$$\boldsymbol{\Phi}^\top \boldsymbol{R}_k(\boldsymbol{y} - \boldsymbol{\Phi}\boldsymbol{w}_k) = \boldsymbol{0} \Rightarrow \boldsymbol{w}_k = (\boldsymbol{\Phi}^\top \boldsymbol{R}_k \boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^\top \boldsymbol{R}_k \boldsymbol{y}. \tag{85}$$

We iteratively optimize and obtain the local maxima. However, the mixture coefficients are still fixed for all $\boldsymbol{x}$ and it potentially shows overestimation of probability densities in sparse regions because it assumes the same multi-modality over the whole space. **Mixture of experts model** circumvents this issue. This model defines π_k as a function of $\boldsymbol{x}$, i.e. $\pi_k \coloneqq \pi_k(\boldsymbol{x})$ (**Gating function**), so that the modalities dynamically changes based on regions. The optimization of the gating function is solved by separately applying the EM algorithm.

8 Sampling Methods

8.1 Monte-Calro (MC) Sampling for Expectation

The basic usage of Monte-Carlo (MC) sampling is to take an expectation of $f(\boldsymbol{x})$ given a distribution $p(\boldsymbol{x})$:

$$\begin{aligned}\mathbb{E}[f] &= \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} \\ &\simeq \frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{x}^{(i)})\end{aligned} \tag{86}$$

where M is the number of samples. Although MC works nicely with very few samples regardless of the dimensionality of $\boldsymbol{x}$ if $\boldsymbol{x}^{(i)}$ is i.i.d, MC suffers from (1) **difficulties to get independent samples**, and (2) **dominant samples**, i.e. large $|f(\boldsymbol{x})|$, **from regions with small probability**.

8.2 Sampling from Standard Distribution

Suppose we would like to obtain 1-dimensional samplings x from a certain distribution $f(x)$. We consider to convert a uniformly distributed random variable $z \sim \mathcal{U}(0,1)$ to a sample from the target distribution $f(x)$. Since the value range of the cumulated probability distribution $F(X \leq x) = \int_{-\infty}^{x} f(x')dx'$ is always $[0,1]$, the conversion is achieved by $x = F^{-1}(z)$ as long as $F^{-1}(z)$ has an analytical form or a special library. The obvious issue of this method is no generalization to a distribution that does not have an analytical F^{-1}.

8.3 Rejection Sampling

Rejection sampling is another sampling method used when F^{-1} is difficult. In rejection sampling, we assume that we know the analytical form of $p(\boldsymbol{x})$ (or we must know the shape, i.e. $\tilde{p}(\boldsymbol{x}) \coloneqq Cp(\boldsymbol{x})$, at least) and we have a standard distribution $q(\boldsymbol{x})$ from which we can sample easily. We first determine a fixed factor k such that $p(\boldsymbol{x}) \leq kq(\boldsymbol{x})$ holds for all $\boldsymbol{x}$. Then the rejection sampling is performed as follows:

1. Draw a sample $\boldsymbol{x}_0$ from $q(\boldsymbol{x})$,
2. Accept the sample with the probability of $p(\boldsymbol{x}_0)/kq(\boldsymbol{x}_0)$; otherwise reject it and back to 1.

The acceptance probability is generally computed as follows:

$$p_{\text{accept}} = \int \frac{p(\boldsymbol{x})}{kq(\boldsymbol{x})}q(\boldsymbol{x})d\boldsymbol{x} = \frac{1}{k} \tag{87}$$

Since the choices of k and $q(\boldsymbol{x})$ are hard for high dimensions and high dimensions typically leads to large k, this algorithm is not used for high-dimensional distribution $p(\boldsymbol{x})$. Note that the efficiency of this algorithm depends on the acceptance probability and the ideal k is 1.

8.4 Importance Sampling

If the goal is to estimate an expectation value and the analytical form of $p(\boldsymbol{x})$ is available, we can use the following importance sampling:

$$\begin{aligned}\int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} &= \int f(\boldsymbol{x})\frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}q(\boldsymbol{x})d\boldsymbol{x} \\ &\simeq \frac{1}{M}\sum_{i=1}^{M}\frac{p(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})}f(\boldsymbol{x}^{(i)})\end{aligned} \tag{88}$$

Algorithm 1 Metropolis Hastings

```
    $q(\boldsymbol{x}|\boldsymbol{x}')$        ▷ Proposal distribution. This is typically Gaussian distribution with mean = $\boldsymbol{x}$.
function METROPOLIS HASTINGS
    for $t = 0, 1, \dots, T$ do
        $\boldsymbol{x} \sim q(\cdot|\boldsymbol{x}^{(t)})$
        $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}$ with the probability of $\min\left(1, \frac{p(\boldsymbol{x})q(\boldsymbol{x}^{(t)}|\boldsymbol{x})}{p(\boldsymbol{x}^{(t)})q(\boldsymbol{x}|\boldsymbol{x}^{(t)})}\right)$ otherwise $\boldsymbol{x}^{(t)}$
```

where each point is sampled from $q(\boldsymbol{x})$ and $p(\boldsymbol{x})/q(\boldsymbol{x})$ is called **importance weight**. In contrast to the rejection sampling, the bound k is not required; however, if $q(\boldsymbol{x})$ is not close to $p(\boldsymbol{x})$, **importance weights will be biased**, and thus the expectation value will be biased as well. Since when we have many samples with a large weight, those will have more impact on the expectation value and those samples change each time. and vice versa, the importance sampling will yield high variance. Since the ideal density ratio is 1, the following **effective sample size** is checked:

$$\mathcal{L}_{\mathrm{eff}} = \sum_{i=1}^{M} \frac{p(\boldsymbol{x}^{(i)})}{q(\boldsymbol{x}^{(i)})}. \tag{89}$$

When we get only small importance weights, $\mathcal{L}_{\mathrm{eff}}$ becomes much smaller than M and the expectation value will have a totally different scale. In summary, although importance sampling is a useful algorithm, since results are likely to be biased or to have high variance easily, we need to pay attention to the distribution of weights.

8.5 Markov Chain Monte-Carlo (MCMC) Sampling

MCMC samples each weight according to a **proposal distribution** or **transition distribution** $q(\boldsymbol{x}|\boldsymbol{x}')$ and moves around the space while accepting or rejecting the proposal. Since the next state depends only on the current state, it is called Markov chain. For the final sampling, we use every t-th sample from the history to avoid the correlation between samples close to each other in terms of time steps. Ideally, this sample approximates the target distribution $p(\boldsymbol{x})$, i.e. **stationary distribution**.

8.5.1 Metropolis Hastings

The major algorithm for MCMC is Metropolis-Hasting in Algorithm 1. The sufficient condition for a stationary distribution to exist is that the **detailed balance** $p(\boldsymbol{x}_{(t)})q(\boldsymbol{x}^{(t+1)}|\boldsymbol{x}^{(t)}) = p(\boldsymbol{x}_{(t+1)})q(\boldsymbol{x}_{(t)}|\boldsymbol{x}_{(t+1)})$. In practice, even when the detailed balance is satisfied, it may still take time to reach the stationary distribution. This time (from the beginning) is called **mixing time**. Intuitively, the distribution reaches the stationary distribution when the chain forgets the beginning states. Therefore, we take samples after a burning-in phase, and the length of the burning-in phase is a hyperparameter. Although MCMC often fails if each peak of modalities is far away from each other, MCMC can sample from multimodal distributions.

8.5.2 Gibbs Sampling

Another variant of the Metropolis-hastings algorithm is the Gibbs sampling. Gibbs sampling is an efficient algorithm that samples each dimension separately conditioned on other dimensions. The fomulation is the following:

$$\begin{aligned} x_1^{(t+1)} &\sim p(x_1 | x_2^{(t)}, x_3^{(t)}, \cdots, x_D^{(t)}) \\ x_2^{(t+1)} &\sim p(x_2 | x_1^{(t+1)}, x_3^{(t)}, \cdots, x_D^{(t)}) \\ &\vdots \\ x_D^{(t+1)} &\sim p(x_D | x_1^{(t+1)}, x_2^{(t+1)}, \cdots, x_{D-1}^{(t+1)}) \end{aligned} \tag{90}$$

where D is the dimension of $\boldsymbol{x}$ and the conditional distribution is usually computed as Gaussian distribution.

9 Dimension Reduction Methods

In practice, even when data has high dimensions, intrinsic dimensions are usually fewer than the actual dimension size. Since high dimensional data usually causes the curse of dimensionality and it is hard to visualize, dimension reduction is sometimes necessary. In this section, we discuss several methods for dimension reduction.

9.1 Principal Component Analysis (PCA)

Assuming we have a dataset $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$, we would like to reduce the dataset to $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$ such that $d < D$. PCA uses linear mapping to project onto another space so that **the variance in the projected space will be maximized** and **the projection error will be minimized**. For the sake of simplicity, we first consider $d = 1$. Suppose we map the dataset to 1D space by a unit vector $\boldsymbol{u} \in \mathbb{R}^D$, then the objective is the following:

$$\max_{\boldsymbol{u} \in \mathbb{R}^D} \boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u} \text{ subject to } \|\boldsymbol{u}\|^2 = 1 \text{ where } \boldsymbol{\Sigma} = \frac{1}{N} \sum_{i=1}^N (\boldsymbol{x}_i - \boldsymbol{\mu})(\boldsymbol{x}_i - \boldsymbol{\mu})^\top, \boldsymbol{\mu} = \frac{1}{N} \sum_{i=1}^N \boldsymbol{x}_i. \tag{91}$$

Since this is a constraint optimization, the formulation results in the following Lagrange multiplier:

$$\mathcal{L}(\boldsymbol{u}, \lambda) = \boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u} - \lambda(\|\boldsymbol{u}\|^2 - 1). \tag{92}$$

From the KKT conditions, we obtain $\boldsymbol{\Sigma}\boldsymbol{u} = \lambda \boldsymbol{u}$ and the solution of this equation is obviously $\boldsymbol{u}$ to be an eigenvector of $\boldsymbol{\Sigma}$. Since $\boldsymbol{\Sigma}\boldsymbol{u} = \lambda \boldsymbol{u}$ and $\|\boldsymbol{u}\| = 1$, the variance in the new space will be $\boldsymbol{u}^\top \boldsymbol{\Sigma} \boldsymbol{u} = \lambda$. The objective is to maximize the variance, so the eigenvector $\boldsymbol{u}$ with the largest eigenvalue λ will be the solution. When $d > 1$, we just need to take d eigenvectors with the eigenvalues till the d-th largest and we define a mapping as $\boldsymbol{U} = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_d] \in \mathbb{R}^{D \times d}$. Then the projection is computed as:

$$\boldsymbol{x}_p = \boldsymbol{\mu} + \sum_{i=1}^d ((\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{u}_i) \boldsymbol{u}_i \tag{93}$$

where $(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{u}_i$ is an orthographic projection of $\boldsymbol{x} - \boldsymbol{\mu}$ onto $\boldsymbol{u}_i$. Note that the projection error is computed as:

$$\begin{aligned} \|\boldsymbol{x}_p - \boldsymbol{x}\|^2 &= \left(\sum_{i=d+1}^D ((\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{u}_i) \boldsymbol{u}_i \right)^2 \\ &= \sum_{i=d+1}^D \|((\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{u}_i) \boldsymbol{u}_i\|^2 \ (\because \boldsymbol{u}_i^\top \boldsymbol{u}_j = 0 \text{ if } i \neq j) \\ &= \sum_{i=d+1}^D \boldsymbol{u}_i^\top (\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^\top \boldsymbol{u}_i. \end{aligned} \tag{94}$$

Plugging this result into the definition of $\boldsymbol{\Sigma}$, we obtain the projection error of $\sum_{i=d+1}^{D} \lambda_i$. When $D > N$, $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{u}) = \lambda(\boldsymbol{X}\boldsymbol{u})$ is more efficient rather than $\frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{u} = \lambda\boldsymbol{u}$ as $\boldsymbol{X}\boldsymbol{X}^\top \in \mathbb{R}^{N \times N}$. When we define $\boldsymbol{v} \coloneqq \boldsymbol{X}\boldsymbol{u}$, we can reconstruct the eigenvector for the original problem as $\boldsymbol{u} = 1/(N\lambda)^{1/2}\boldsymbol{X}^\top\boldsymbol{v}$ where the coefficient will be different depending on whether we have the constraint $\|\boldsymbol{v}\|^2 = 1$ or not.

PCA is known to be a special case of multi-dimensional scaling (**MDS**). While PCA preserves Euclid distances between each data point as much as possible, MDS does so for an arbitrary distance metric. Since MDS handles an arbitrary distance metric, MDS can map features non-linearly, unlike PCA. MDS includes kernel PCA and Isomap.

9.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

We use t-SNE mostly for visualizations of a high-dimensional space and it preserves the local structure. This method matches the pair-wise similarities in both the original and the reduced spaces as follows:

1. **Similarities in the original space**: Compute similarities using the Gauss kernel

$$p_{i,j} = \frac{\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_i^2)}{\sum_{k \neq i}\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2/2\sigma_i^2)}, \tag{95}$$

2. **Similarities in the reduced space**: Compute similarities using the t-distribution

$$q_{i,j} = \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_{k \neq i}(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_k\|^2)^{-1}}, \tag{96}$$

3. **Calculation of mismatching measure**: Build perplexity matrices $(p_{i,j} + p_{j,i})/2N$ and $(q_{i,j} + q_{j,i})/2N$ and compute the mismatch measure via:

$$D_{\mathrm{KL}}(P\|Q) = \sum_{i \neq j} p_{i,j} \log \frac{p_{i,j}}{q_{i,j}} \tag{97}$$

4. **Minimize the mismatching measure**: Gradient descent of $D_{\mathrm{KL}}(P\|Q)$ with respect to $\boldsymbol{y}$.

Note that we compute σ_i using K-nearest neighbors as in adaptive KDE and the reduced space uses t-distribution because lower-dimensional spaces are crowded and a long-tailed distribution is desirable.