Statistical Pattern Recognition

Shuhei Watanabe

March 31, 2022

1 Introduction

While many real-world tasks can be solved by rule-based algorithms, such heuristics do not generalize for complicated tasks such as digit recognition. For example, when we tilt images, heuristics have to add new rules manually. However, this is not efficient and machine learning or pattern recognition mitigates this problem. Machine learning lets the computer learn decision rules automatically from a set of examples; therefore, we do not need to add rules manually and machine learning models learn to predict outputs for unseen data from provided similar examples. Pattern recognition handles supervised learning and unsupervised learning. Supervised learning includes regression and classification tasks. Unsupervised learning includes clustering, density estimation, and subspace estimation.

In statistical pattern recognition, we often infer various probability distribution from given data or derive the the optimal decisions from the distribution or directly from data. Inferences are performed based on the following Bayes' theorem:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}},$$
$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Marginal Likelihood}}$$

where $\mathcal{D}$ is a dataset and $\boldsymbol{\theta}$ is a set of parameters for the posterior. The maximization of $p(\mathcal{D}|\boldsymbol{\theta})$ is called maximum likelihood estimation (MLE) and the maximization of $p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})$ is called maximum a posteriori (MAP) estimation. Since MLE maximizes the likelihood of data, it is likely to overfit to the dataset. On the other hand, MAP estimation considers a prior distribution of a set of parameters and thus parameters are regularized towards the prior distribution. However, when we have less data, the estimation is more uncertain. For this reason, we might need to reject the inference and analyze data manually when MAP or the maximum likelihood is smaller than a certain threshold. In this notebook, we handle the following three models:

1. **Generative model**: Learn the likelihood and prior from training data and approximate the posterior. Since the dimension of data is high in most cases, the inference is more complex and usually more difficult.

2. **Discriminative model**: Directly learn the posterior from training data. Since it does not require high dimensional inferences, this model is less complex.

3. **Mappings**: Directly learn the mapping from inputs to outputs. Probabilities do not play a role anymore and this model can be trained with less data compared to previous two models.

The examples for each model is listed in Table 1. We discuss more details from the next section. 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.

Table 1: The examples for each model

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

2 Probability distribution

2.1 Bernoulli distribution

Suppose $p(x = 1|\mu) = \mu$ represents the probability that we get $x = 1$ where $x \in \{0, 1\}$, then the Bernoulli distribution is the following:

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

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 $\mathcal{D} = \{x_i\}_{i=1}^N$ where each sample is obtained **independently**, the MLE is computed as follows:

$$\begin{aligned}
p(\mathcal{D}|\mu) &= \prod_{i=1}^{N} \mathrm{Bern}(x_i|\mu) \\
\log p(\mathcal{D}|\mu) &= \sum_{i=1}^{N} \Bigg(x_i \log \mu + (1-x_i)\log(1-\mu) \Bigg) \\
\frac{\partial}{\partial \mu} \log p(\mathcal{D}|\mu) &= \sum_{i=1}^{N} \Bigg(\frac{x_i}{\mu} - \frac{1-x_i}{1-\mu} \Bigg) = \frac{n}{\mu} - \frac{N-n}{1-\mu} \\
\mu_{\mathrm{MLE}} &= \frac{n}{N} \quad \left(\because \frac{\partial}{\partial \mu} \log p(\mathcal{D}|\mu) = 0 \right)
\end{aligned}$$

where n is the number of occurrences of $x_i = 0$.

2.2 Binomial distribution

When we get $x = 1$ n times in the N-th tries of the task in the previous section, this probability is calculated by binomial distribution and formulated as follows:

$$\mathrm{Bin}(n|N, \mu) = {}_N\mathrm{C}_n \mu^n (1-\mu)^{N-n}.$$

The mean and variance of the distribution are $\mathbb{E}[x] = N\mu$ and $\mathbb{V}[x] = N\mu(1-\mu)$. When we have less data, the MLE tends to overfit the data. For this reason, we **introduce a prior distribution** to suppress the overfitting. If we would like to infer the posterior, we use **Bayesian inference**; however, when we only need a set of parameters that achieves the maximum posterior, we use **MAP estimation**. Bayesian inference requires the marginal likelihood, which is often hard to compute. On the other hand, when the prior is so-called **conjugate prior**, the posterior distribution is easily

Table 2: The list shows the priors conjugate to the likelihoods that take specific distribution forms. The posterior and the conjugate prior take the same form and the predictive distribution is the marginal distribution.

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

obtained as **the posterior distribution belongs to the same probability distribution family as the prior distribution**. The choice of the conjugate prior depends on the form of the likelihood. Table 2 lists the correspondance of likelihoods and the conjugate prior distributions.

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

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

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. As α, β becomes large, the prior will have more impact on the posterior distribution. The posterior is computed as follows:

$$p(\mu|n, N, \alpha, \beta) \propto \mu^{n+\alpha-1}(1-\mu)^{N-n+\beta-1}.$$

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

2.3 Multinomial distribution

Multinomial distribution is the general form of binomial distribution. In other words,multinomial distribution handles $\boldsymbol{x} \in \mathbb{R}^K$ where $\boldsymbol{x}$ is a one-hot vector and K is the number of categories. The formulation is as follows:

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

where $\mu_k \geq 0$, $\sum_{k=1}^K \mu_k = 1$, and $\sum_{k=1}^K n_k = N$. Given a dataset $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^N$, the likelihood is computed as:

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{i=1}^N \prod_{k=1}^K \mu_k^{x_{i,k}} = \prod_{k=1}^K \mu_k^{n_k}$$

where n_k is the number of occurrences of $x_i = k$. The maximization of the likelihood is solved using the following Lagrangian multiplier:

$$\mathcal{L}(\boldsymbol{\mu}, \lambda) = \sum_{k=1}^K n_k \log \mu_k + \lambda\left(\sum_{k=1}^K \mu_k - 1\right).$$

The KKT conditions are satisfied when the derivatives with respect to $\boldsymbol{\mu}$ and λ are zero and we obtain $\boldsymbol{\mu}_{\mathrm{MLE}} = [n_1/N, \dots, n_K/N]$.

The conjugate prior of the multinomial distribution is the following Dirichlet distribution:

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

Using the Dirichlet distribution as a conjugate prior, we obtain the following posterior:

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

The MAP estimation yields $\boldsymbol{\mu}_{\mathrm{MAP}} = \left[\frac{n_1+\alpha_1-1}{N+\sum_{k=1}^K(\alpha_k-1)}, \dots, \frac{n_K+\alpha_K-1}{N+\sum_{k=1}^K(\alpha_k-1)}, \right]$.

2.4 Gaussian distribution

The D-dimensional formulation is the following:

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

Note that $(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})$ is called mahalanobis distance. In the case of large D, we often restrict $\boldsymbol{\Sigma}$ to be a diagonal matrix so that the inference time scales linearly with respect to D in exchange of the representational capacity.

2.4.1 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_i 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 $(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) = (U(\boldsymbol{x}-\boldsymbol{\mu}))^\top \mathrm{diag}(\lambda_1^{-1}, \cdots, \lambda_D^{-1})U(\boldsymbol{x}-\boldsymbol{\mu})$. In this formulation, $U(\boldsymbol{x}-\boldsymbol{\mu})$ is viewed as a new coordinate system and $\mathrm{diag}(\lambda_1^{-1}, \cdots, \lambda_D^{-1})$ as covariance matrix with no-correlation between each coordinate. Note that this property is used for Principal component analysis.

2.4.2 Maximum likelihood estimation of parameters

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

$$\frac{\partial}{\partial \boldsymbol{\mu}} \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \boldsymbol{0}$$
$$\frac{\partial}{\partial \boldsymbol{\Sigma}} \log \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \boldsymbol{0}$$

By solving the equations, we obtain $\boldsymbol{\mu}_{\mathrm{MLE}} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i$ and $\boldsymbol{\Sigma}_{\mathrm{MLE}} = \frac{1}{N}\sum_{i=1}^N \boldsymbol{x}_i\boldsymbol{x}_i^\top - \boldsymbol{\mu}_{\mathrm{MLE}}\boldsymbol{\mu}_{\mathrm{MLE}}^\top$ Since the log-likelihood is computed using only the first momentum $\sum_{i=1}^N \boldsymbol{x}_i$ and the second momentum $\sum_{i=1}^N \boldsymbol{x}_i\boldsymbol{x}_i^\top$, we call them **sufficient statistics** and we do not have to store each data point $\boldsymbol{x}_i$ as long as we store the sufficient statistics. The sufficient statistics are updated sequentially and it reduces the

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

overall time complexity. While the expectation of mean $\boldsymbol{\mu}_{\mathrm{MLE}}$ is $\boldsymbol{\mu}_{\mathrm{MLE}}$, that of the covariance matrix is:

$$\begin{aligned}
N\mathbb{E}[\boldsymbol{\Sigma}_{\mathrm{MLE}}] &= \mathbb{E}\left[\sum_{i=1}^{N}(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{x}_i)^2\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{N}(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}} + \boldsymbol{\mu}_{\mathrm{true}} - \boldsymbol{x}_i)^2\right] \\
&= \underbrace{\mathbb{E}\left[\sum_{i=1}^{N}(\boldsymbol{x}_i - \boldsymbol{\mu}_{\mathrm{true}})^2\right]}_{=N\boldsymbol{\Sigma}_{\mathrm{true}}} + \mathbb{E}\left[\sum_{i=1}^{N}\underbrace{(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})^2}_{\text{const w.r.t. } i}\right] - 2\mathbb{E}\left[\underbrace{\sum_{i=1}^{N}(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}})(\boldsymbol{x}_i - \boldsymbol{\mu}_{\mathrm{true}})}_{=N(\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu})}\right] \\
&= N\boldsymbol{\Sigma}_{\mathrm{true}} - N\mathbb{E}[\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}}].
\end{aligned}$$

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

$$\mathbb{E}[\boldsymbol{\mu}_{\mathrm{MLE}} - \boldsymbol{\mu}_{\mathrm{true}}] = \mathbb{V}\left[\frac{\sum_{i=1}^{N}\boldsymbol{x}_i}{N}\right] = \frac{1}{N^2}\mathbb{V}\left[\sum_{i=1}^{N}\boldsymbol{x}_i\right] = \frac{N}{N^2}\mathbb{V}[\boldsymbol{x}] = \frac{\boldsymbol{\Sigma}_{\mathrm{true}}}{N}$$

where the last transformation uses the assumption that $\boldsymbol{x}$ is sampled i.i.d. By plug-in the result, we obtain $\mathbb{E}[\boldsymbol{\Sigma}_{\mathrm{MLE}}] = \frac{N-1}{N}\boldsymbol{\Sigma}_{\mathrm{true}}$ and 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 often modify $\boldsymbol{\Sigma}_{\mathrm{MLE}}$ by multiplying $\frac{N}{N-1}$.

2.4.3 Bayesian inference of mean given variance

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

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{i=1}^{N}\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

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

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

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. (1):

$$\begin{aligned}
\boldsymbol{\mu}_{\mathrm{post}} &= \boldsymbol{\Sigma}_{\mathrm{post}}\left(\boldsymbol{\Sigma}^{-1}\sum_{i=1}^{N}\boldsymbol{x}_i + \boldsymbol{\Sigma}_{\mathrm{prior}}^{-1}\boldsymbol{\mu}_{\mathrm{prior}}\right) \\
\boldsymbol{\Sigma}_{\mathrm{post}} &= (N\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}_{\mathrm{prior}}^{-1})^{-1}
\end{aligned}$$

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

$$\begin{aligned}
\sigma_{\mathrm{post}}^2 &= \frac{\sigma^2\sigma_{\mathrm{prior}}^2}{N\sigma_{\mathrm{prior}}^2 + \sigma^2} \\
\mu_{\mathrm{post}} &= \sigma_{\mathrm{post}}^2\left(\frac{\sum_{i=1}^{N}x_i}{\sigma^2} + \frac{\mu_{\mathrm{prior}}}{\sigma_{\mathrm{prior}}^2}\right)
\end{aligned}$$

2.4.4 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:

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

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 prior of this setting, we obtain the following parameters:

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

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

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

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:

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

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. Those are called **predictive distribution**, which does not depend on parameters of the posterior due to the marginalization, and we can use it 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**:

$$\mathcal{L}(\boldsymbol{x}|\boldsymbol{\mu},b) = \frac{1}{2b}\exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}\|}{b}\right)$$

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

2.4.5 Bayesian inference of both mean and variance

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

$$\begin{aligned}p(\mu,\lambda) =& \mathcal{N}(\mu|\mu',(\beta\lambda)^{-1})\mathrm{Gam}(\lambda|\alpha,\beta)\\ p(\boldsymbol{\mu},\boldsymbol{\Lambda}) =& \mathcal{N}(\boldsymbol{\mu}|\boldsymbol{\mu}',(\beta\boldsymbol{\Lambda})^{-1})\mathcal{W}(\boldsymbol{\Lambda}|\nu,\boldsymbol{W})\end{aligned}$$

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 $\log p(\boldsymbol{x}) = \log p(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Lambda}) - \log p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\boldsymbol{x}) + \text{const}$. Note that we use the results from the posterior computation to calculate $p(\boldsymbol{\mu}, \boldsymbol{\Lambda}|\boldsymbol{x})$.

3 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 view of the EM algorithm.

3.1 Gaussian mixture models (GMM)

Since the typical parametric distributions have only limited representational capacity and many real-world problems have multi-modal distributions, the following Gaussian mixture models (GMM) is widely used:

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

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

$$\begin{aligned} p(\boldsymbol{x}|\boldsymbol{z}) &= \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k} \\ p(\boldsymbol{x}) &= \int p(\boldsymbol{x}|\boldsymbol{z}) \underbrace{p(\boldsymbol{z})}_{p(z_k=1|\boldsymbol{z})=\pi_k} d\boldsymbol{z} \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \end{aligned} \tag{2}$$

where $\boldsymbol{z} \in \{0, 1\}^K$ is a one-hot vector. From Eq. (2), GMM is viewed as the marginalized likelihood with respect to the latent variables $\boldsymbol{z}$. 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 variables and the parameters. This scheme is called expectation-maximization (**EM**) algorithm and we see the details from the next section.

3.1.1 EM algorithm for GMM

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

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

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

$$\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

The next step is **M step** where we consider the maximization of log likelihood given the responsibilities and **incomplete** dataset $\boldsymbol{X}$. Since we have a constraint $\sum_{k=1}^{K}\pi_k = 1$, we need to consider the following Lagrange multiplier:

$$\mathcal{L}(\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}, \lambda) = -\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) - \lambda\left(\sum_{k=1}^{K}\pi_k - 1\right)$$

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

$$\begin{aligned}\textbf{Stationarity}: \frac{\partial\mathcal{L}}{\partial\pi_k} = 0, \frac{\partial\mathcal{L}}{\partial\boldsymbol{\mu}_k} = \boldsymbol{0}, \frac{\partial\mathcal{L}}{\partial\boldsymbol{\Sigma}_k} = \boldsymbol{0},\\ \textbf{Primal feasibility}: \sum_{k=1}^{K}\pi_k = 1.\end{aligned}$$

From the stationary conditions, we obtain:

$$\begin{aligned}\frac{\partial}{\partial\boldsymbol{\mu}_k}\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= -\sum_{i=1}^{N}\gamma_{k,i}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k) = \boldsymbol{0}\\ \boldsymbol{\mu}_k &= \frac{1}{N_k}\sum_{i=1}^{N}\gamma_{k,i}\boldsymbol{x}_i\\ \frac{\partial}{\partial\boldsymbol{\Sigma}_k}\log p(\boldsymbol{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) &= \sum_{i=1}^{N}\gamma_{k,i}\left(-\frac{1}{2}\boldsymbol{\Sigma}_k^{-1} + \frac{1}{2}\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top\boldsymbol{\Sigma}_k^{-1}\right) = \boldsymbol{0}\\ \boldsymbol{\Sigma}_k &= \frac{1}{N_k}\sum_{i=1}^{N}\gamma_{k,i}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top\\ \frac{\partial\mathcal{L}}{\partial\pi_k} &= -\sum_{i=1}^{N}\frac{\gamma_{k,i}}{\pi_k} - \lambda = 0\\ \pi_k &= \frac{N_k}{N}\end{aligned} \tag{3}$$

where $N_k = \sum_{i=1}^{N}\gamma_{k,i}$. Those mean and covariance correspond to the weighted average of those obtained from the fed data points. Note that the global maximum of the likelihood is achieved by taking the singular covariance matices at each data point and thus we need to avoid shrinking to the trivial solution by **initializing far away from such solutions** or **applying a prior to the covariance to avoid 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**: Compute the **expectation** of the latent variables given fixed parameters:

$$\gamma_{k,i} \coloneqq \mathbb{E}[z_k|\boldsymbol{x}_i] = p(z_k = 1|\boldsymbol{x}_i), \text{ and}$$

2. **M step**: Given the responsibilities, **maximize** the log-likelihood as in Eq. (3)

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i=1}^{N}\gamma_{k,i}\boldsymbol{x}_i,\ \boldsymbol{\Sigma}_k = \frac{1}{N_k}\sum_{i=1}^{N}\gamma_{k,i}(\boldsymbol{x}_i - \boldsymbol{\mu}_k)(\boldsymbol{x}_i - \boldsymbol{\mu}_k)^\top,\ \pi_k = \frac{N_k}{N}.$$

Each iteration guarantees the improvement from the last iteration and thus the EM algorithm always yields a local optimum.

3.2 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:

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

We iteratively minize $\mathcal{L}$ as follows:

1. **E-step**: Minimize $\mathcal{L}$ with respect to $\boldsymbol{\gamma}$ by assigning each data point to the closest cluster center

$$\gamma_{k,i} = \begin{cases} 1 \text{ if } k = \mathrm{argmin}_{k'} \|\boldsymbol{x}_i - \boldsymbol{\mu}_{k'}\|^2 \\ 0 \text{ otherwise} \end{cases}, \text{ and}$$

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

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\mu}_k} = 2 \sum_{i=1}^{N} \gamma_{k,i} (\boldsymbol{x}_i - \boldsymbol{\mu}_k) = 0$$
$$\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_{k,i} \boldsymbol{x}_i}{\sum_{i=1}^{N} \gamma_{k,i}}.$$

The differences are (1) the responsibilities $\gamma_{k,i}$ are either 0 or 1, and (2) the covariance matrix is singular, i.e. hard assignment. In other words, the second point implies that GMM approaches the result of K-means as $\boldsymbol{\Sigma}_k$ goes to $\boldsymbol{0}$. 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.

3.3 General EM algorithm

Now we consider the more general form of EM algorithm. Given a dataset $\boldsymbol{X}$, we would like to maximize the following likelihood:

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

We assume that 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. Therefore, we decompose the optimization problem into two subproblems: (1) to estimate the expectation of the latent variables given a data point $\mathbb{E}[\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}}]$ (**E step**), and (2) to maximize complete-data likelihood given the expectation and the fixed parameters (**M step**). More formally, we compute:

$$\begin{aligned} \log p(\boldsymbol{X}|\boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{Z} \sim q(\boldsymbol{Z})}[\log p(\boldsymbol{X}|\boldsymbol{\theta})] \ (\because p(\boldsymbol{X}|\boldsymbol{\theta}) \text{ does not depend on } \boldsymbol{Z}) \\ &= \mathbb{E}_{\boldsymbol{Z} \sim q(\boldsymbol{Z})}\left[\log \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})}\right] \ (\because \text{Bayes' theorem}) \\ &= \underbrace{\int q(\boldsymbol{Z}) \ \log \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\boldsymbol{\theta})}{q(\boldsymbol{Z})} d\boldsymbol{Z}}_{=\mathcal{L}_{\text{ELBO}}(q, \boldsymbol{\theta}) \ \to \ \log p(\boldsymbol{X}|\boldsymbol{\theta})} + \underbrace{\int q(\boldsymbol{Z}) \ \log \frac{q(\boldsymbol{Z})}{p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})} d\boldsymbol{Z}}_{=D_{\text{KL}}(q \| p) \ \to \ 0} \end{aligned} \tag{4}$$

where $q(\boldsymbol{Z})$ is an approximate distribution of the latent variables and $\mathcal{L}_{\text{ELBO}}(q, \boldsymbol{\theta})$ is the evidence lower bound (**ELBO**) of likelihood. Since the LHS of Eq. (4) is constant and the KL-divergence takes zero when $q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta})$, we obtain $\log p(\boldsymbol{X}|\boldsymbol{\theta}) = \mathcal{L}_{\text{ELBO}}(p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}), \boldsymbol{\theta})$. Hence, the maximization of

ELBO given the distribution $q(\boldsymbol{Z}) = p(\boldsymbol{Z}|\boldsymbol{X}, \boldsymbol{\theta}_{\text{old}})$ yields the MLE of $p(\boldsymbol{X}|\boldsymbol{\theta})$ under this condition. Then ELBO is reformulated as follows:

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

By repeating these two steps, we can iteratively maximize the likelihood $p(\boldsymbol{X}|\boldsymbol{\theta})$. The EM algorithm guarantees the improvement of ELBO and converges to a stationary point (, but no guarantee of a local maximum in general). Even when closed-form solutions for both steps are not available, **generalized EM** algorithm enables the iterative optimization of the likelihood.

4 Non-parameteric methods

While parametric models and GMM have limited representational capacity, non-parametric methods dynamically change their capacity 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.

4.1 Density estimation methods

4.1.1 Histogram

Histogram is the most basic probability density estimation method. We first divide the parameter space $\mathcal{X}$ into bins and counts 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:

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

where n_i is the number of data points that belong to the i-th bin and $V : \mathbb{R}^D \to \mathbb{R}_+$ is a volume measure. The major advantages of histogram are that (1) we can throw away samples once we check to which bin they belong, and (2) the algorithm is simple. On the other hand, since we have to store all the bin information, the memory requirements increase exponentially and histogram will not be a realistic choice as the dimension becomes large. Additionally, the choice of the bin width affects the estimation. While small bins can capture detailed information, it causes overfitting. On the other hand, large bins prevent overfitting; however, it loses local details. Furthermore, although some samples might be close to boundaries, especially in high dimensions, 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.

4.1.2 Kernel density estimation (KDE)

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

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

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:

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

This metrics can be optimized via either direct search or gradient descent. Since the Epanechnikov kernel has short-tail and it assures the convergence, we often use this kernel.

An alternative loss measure is the following integrated squared error:

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

This loss measure guarantees the convergence.

Additionally, in most real-world applications, we are intereted 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:

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

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

4.1.3 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 because of the over-smoothing. In sparse regions, it does not show the optimal density because of the overfitting. The following KNN is a remedy for this problem.

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

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 estimate in dense regions** and large K leads to more smoothing effect and biased estimate. Bayesian formulation for KNN is the following:

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

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 for classification; however, the performance is sensitive to the parameter selection of K and KNN often suffers from noisy estimate in dense regions.

4.1.4 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:

$$p(\boldsymbol{x}) = \frac{1}{N(2\pi)^{D/2}|\boldsymbol{\Sigma}(\boldsymbol{x})|^{1/2}} \sum_{i=1}^{N} \exp\left(-\frac{1}{2}(\boldsymbol{x}_i - \boldsymbol{x})\boldsymbol{\Sigma}(\boldsymbol{x})^{-1}(\boldsymbol{x}_i - \boldsymbol{x})^\top\right),$$

$$\boldsymbol{\Sigma}(\boldsymbol{x}) = \frac{1}{K} \sum_{\boldsymbol{x}_i \in \mathcal{S}} (\boldsymbol{x}_i - \boldsymbol{x})(\boldsymbol{x}_i - \boldsymbol{x})^\top \ (\mathcal{S} \text{ is a set of } K\text{-nearest neighbors}).$$

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

4.2 Space subdivision

Since KNN requires the comparison of distances to each data point, the time complexity increases linearly with respect 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 by 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 K-d tree, it suffers from the curse of dimensionality. Other two algorithms better-behave 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 requirements grows exponentially. Notice that each tree requires $O(N \log N)$ for build and $O(\log N)$ for inference if we view the split procedure as $O(1)$.

5 Regression

5.1 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:

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

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 [2]. In the MAP estimation, we maximize the following:

$$\begin{aligned} \text{posterior} &\propto \text{likelihood} \times \text{prior}, \\ p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y}) &\propto p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w})p(\boldsymbol{w}). \end{aligned} \tag{5}$$

Since the conjugate prior for the likelihood $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{X}\boldsymbol{w}, \sigma^2\boldsymbol{I})$ [3] with Gaussian distribution with unknown mean is Gaussian distribution, the following holds:

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

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

$$\boldsymbol{\mu}_{\text{post}} = \frac{1}{\sigma^2}\boldsymbol{\Sigma}_{\text{post}}\boldsymbol{X}^\top\boldsymbol{y}, \boldsymbol{\Sigma}_{\text{post}} = \left(\frac{1}{\sigma^2}\boldsymbol{X}^\top\boldsymbol{X} + \boldsymbol{\Sigma}_{\text{prior}}^{-1}\right)^{-1} \tag{6}$$

are the parameters of the posterior. This result is directly derived by transforming Eq. (5). Using weights sampled from the posterior, the prediction of Bayesian linear regression is computed as follows:

$$\begin{aligned} p(y|\boldsymbol{x}, \boldsymbol{X}, \boldsymbol{y}) &= \int p(y|\boldsymbol{x}, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{X}, \boldsymbol{y})d\boldsymbol{w} \\ &= \mathcal{N}(\boldsymbol{\mu}_{\text{post}}^\top\boldsymbol{x}, \boldsymbol{x}^\top\boldsymbol{\Sigma}_{\text{post}}\boldsymbol{x}). \end{aligned} \tag{7}$$

Note that $\boldsymbol{X}$ can be replaced 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.

5.2 The evidence approximation

Since Bayesian linear regression requires control parameters and they obviously affect the prediction, they need to be determined via cross validation. However, the 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**. First, let us 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:

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

However, this marginalization is not feasible analytically without any assumptions. For this reason, we introduce the following two assumptions:

[2] MLE is equivalent to MAP estimation with the uniformly distributed marginal likelihood.
[3] σ^2 is estimated via MLE, i.e. $\sigma^2 \coloneqq \sigma^2_{\text{MLE}} = \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2/N$

Assumption 1

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

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

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

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

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

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{x}, \alpha, \beta) &= \int p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w}, \beta) p(\boldsymbol{w}|\alpha) d\boldsymbol{w} \\ &= \left(\frac{\beta}{2\pi}\right)^{N/2} \left(\frac{\alpha}{2\pi}\right)^{D/2} \int \exp\left(-\frac{\beta}{2}\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 - \frac{\alpha}{2}\|\boldsymbol{w}\|^2\right) d\boldsymbol{w}. \end{aligned} \tag{8}$$

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:

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

where $\mu_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}}$ are identical to those in Eq. (6). Since the following holds,

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

we can transform the integral in Eq. (8) as follows:

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

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

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

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

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

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.

6 Kernel methods

6.1 The properties of the 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 over- or underestimation. For example, **more smoothing leads to generalization and biased estimates**. Note that if a kernel functions 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 represented as 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 [4]. Isotropic kernel can handle graph and image well. The major drawbacks of kernel methods are (1) poor extrapolation, (2) shrinkage to zero-mean prediction in sparse regions.

6.2 Kernel regression

When we reformulate linear regression from the Bayesian view as in Eq. (7), we obtain the following:

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

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

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

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 $N \ll D$ where D might be potentially infinity.

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

6.3 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:

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

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

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

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.

6.4 Gaussian process (GP) regressor

6.4.1 Formulation

The Gaussian process (GP) drops the intermediate estimation of weights $\boldsymbol{w} \in \mathbb{R}^{\infty}$ and directly estimates priors in the function space $\Phi = \{\phi_i\}_{i=1}^{\infty}$ where $\phi_i : \mathcal{X} \to \mathbb{R}$ is a (usually non-linear) mapping. First, we assume that the observation y given a data point $\boldsymbol{x}$ follows:

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

In other words, the likelihood is $p(\boldsymbol{y}|\boldsymbol{f}) = p(\boldsymbol{y}|\boldsymbol{\Phi}, \boldsymbol{w}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{\Phi}\boldsymbol{w}, \lambda\boldsymbol{I})$ where we define $\boldsymbol{\Phi} \coloneqq \Phi(\boldsymbol{X}) \in \mathbb{R}^{N\times\infty}$ and $\boldsymbol{w} = \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I})$. Then the moments of the likelihood are the following:

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

Therefore, the prior is $p(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K})$ where $\boldsymbol{f} = \boldsymbol{\Phi}\boldsymbol{w} \in \mathbb{R}^{N\times N}$ and we obtain the following marginal likelihood:

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

Using this result, we obtain the following predictive distribution given a new data point:

$$p\left(\begin{bmatrix}\boldsymbol{y} \\ y_{N+1}\end{bmatrix}\right) \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}\boldsymbol{K} + \lambda\boldsymbol{I} & \boldsymbol{k}_{N+1} \\ \boldsymbol{k}_{N+1}^{\top} & k(\boldsymbol{x}_{N+1}, \boldsymbol{x}_{N+1}) + \lambda\end{bmatrix}\right) \coloneqq \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix}\boldsymbol{C}_N & \boldsymbol{k}_{N+1} \\ \boldsymbol{k}_{N+1}^{\top} & c_{N+1}\end{bmatrix}\right) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{C}_{N+1}),$$

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

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

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 $p(\boldsymbol{x}_a) = \mathcal{N}(\boldsymbol{\mu}_a, \boldsymbol{\Sigma}_a), p(\boldsymbol{x}_b) = \mathcal{N}(\boldsymbol{\mu}_b|\boldsymbol{\Sigma}_b)$. Note that if all non-diagonal elements of $\boldsymbol{K}$ are close to zero, the prediction overfits the training data and if most elements in $\boldsymbol{K}$ have similar values, it implies that the predictions are biased due to over-smoothing. From the representor theorem, we can compute the predictive mean as follows:

$$\mu(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) \tag{9}$$

where the weights are $\boldsymbol{\alpha} = \boldsymbol{C}_N^{-1}\boldsymbol{y}$.

This formulation can be derived from the linear-combination perspective as well. Suppose we would like to optimize the following loss function:

$$\begin{aligned}
\mathcal{L}(\boldsymbol{y}, \boldsymbol{\Phi w}) &= \frac{1}{2}(\boldsymbol{y} - \boldsymbol{\Phi w})^\top(\boldsymbol{y} - \boldsymbol{\Phi w}) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2, \\
\frac{\partial \mathcal{L}(\boldsymbol{y}, \boldsymbol{\Phi w})}{\partial \boldsymbol{w}} &= -\boldsymbol{\Phi}^\top\boldsymbol{y} + \boldsymbol{\Phi}^\top\boldsymbol{\Phi w} + \lambda \boldsymbol{w}, \\
\textbf{Stationarity: } \boldsymbol{w}^\star &= \boldsymbol{\Phi}^\top \underbrace{\frac{1}{\lambda}(\boldsymbol{y} - \boldsymbol{\Phi w})}_{\text{Dual variable: } \boldsymbol{\alpha}} = \boldsymbol{\Phi}^\top\boldsymbol{\alpha}.
\end{aligned}$$

Therefore, the dual problem of this optimization is the following:

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

This result is identical to that in Eq. (9).

The computational complexity for the training is dominated by matrix inversion and that costs $O(N^3)$ and the inference costs $O(N^2)$. If the feature dimension D is smaller than N, linear regression of non-linear feature will be more efficient. Since the matrix inversion is not feasible when N is large, we often exploit the fact that most elements in $\boldsymbol{K}$ are close to zero and consider $\boldsymbol{K}$ as a sparse matrix in order to reduce the computational complexity. 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 approximations of $\boldsymbol{K}_{N\times N} \simeq \boldsymbol{K}_{N\times M}\boldsymbol{K}_{M\times M}^{-1}\boldsymbol{K}_{M\times N}$ and it is guaranteed that there is a kernel matrix $\boldsymbol{K}_{M\times M}$ that satisfies this approximation. 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. Overall, it costs $O(M^3 + NMP)$.

6.4.2 Automatic relevance determination (ARD)

Stationary kernels have the same influence from each axis because we use the same bandwidth for all dimensions; however, some features often have more impact than the others. To address this issue,

automatic relevance determination (ARD) considers the following kernel:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \theta_0 \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^\top \mathrm{diag}[\theta_1, \theta_2, \cdots, \theta_d](\boldsymbol{x}-\boldsymbol{x}')\right)$$

where θ_i is a hyperparameter. These hyperparameters are optimized via the maximization of the following log-likelihood:

$$\begin{aligned}
\log p(\boldsymbol{y}|\boldsymbol{\theta}) &= -\frac{1}{2}\log|\boldsymbol{C}_N(\boldsymbol{\theta})| - \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{C}_N(\boldsymbol{\theta})^{-1}\boldsymbol{y} - \frac{N}{2}\log 2\pi \\
\frac{\partial}{\partial\theta_i}\log p(\boldsymbol{y}|\boldsymbol{\theta}) &= -\frac{1}{2}\mathrm{Tr}\left(\boldsymbol{C}_N^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}_N(\boldsymbol{\theta})}{\partial\theta_i}\right) + \frac{1}{2}\boldsymbol{y}^\top \boldsymbol{C}_N^{-1}(\boldsymbol{\theta})\frac{\partial \boldsymbol{C}_N(\boldsymbol{\theta})}{\partial\theta_i}\boldsymbol{C}_N^{-1}(\boldsymbol{\theta})\boldsymbol{y}.
\end{aligned}$$

We can find a local maxima of $\boldsymbol{\theta}$ via gradient ascent. 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 $O(N^2)$.

6.5 Mixture of regressors

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

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

where $\boldsymbol{\theta}$ is a set of all hyperparameters $\lambda, \boldsymbol{w}_k, \pi_k$ and $\boldsymbol{f} = [\boldsymbol{w}_1^\top\Phi(\boldsymbol{x}), \ldots, \boldsymbol{w}_K^\top\Phi(\boldsymbol{x})]$. As in GMM, we apply 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:

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

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

$$\begin{aligned}
\log p(\boldsymbol{y}|\boldsymbol{F}, \boldsymbol{\theta}) &= \mathbb{E}_{\boldsymbol{z}}[\log p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{F}, \boldsymbol{\theta})] = \sum_{i=1}^{N}\log\left(\sum_{k=1}^{K}\pi_k\mathcal{N}(y_i|f_{k,i}, \lambda)\right), \\
\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{F}, \boldsymbol{\theta})}{\partial \boldsymbol{w}_k} &= \sum_{i=1}^{N}\frac{\gamma_{k,i}}{\lambda}(y_i - f_{k,i})\Phi(\boldsymbol{x}_i) = \boldsymbol{0}, \\
\frac{\partial \log p(\boldsymbol{y}|\boldsymbol{F}, \boldsymbol{\theta})}{\partial \lambda} &= \sum_{i=1}^{N}\sum_{k=1}^{K}\gamma_{k,i}\left(\frac{1}{2\lambda^2}(f_{k,i}-y_i)^2 - \frac{1}{2\lambda}\right) = 0 \Rightarrow \lambda = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}\gamma_{k,i}\|\boldsymbol{y} - \Phi(\boldsymbol{X})\boldsymbol{w}_k\|^2, \\
\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \pi_k} &= 0 \Rightarrow \pi_k = \frac{1}{N}\sum_{i=1}^{N}\gamma_{k,i} \text{ (Same KKT conditions in Eq. (3)).}
\end{aligned}$$

When we define a responsibility matrix as $\boldsymbol{R}_k = \mathrm{diag}[\gamma_{k,1}, \ldots, \gamma_{k,N}]$ 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:

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

The optimization is iteratively performed and we 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 EM algorithm.

7 Sampling methods

7.1 Monte-Calro (MC) sampling for the expectation

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

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

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**.

7.2 Sampling from standard distributions

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}.

7.3 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:

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

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.

Algorithm 1 Metropolis Hastings

$q(\boldsymbol{x}|\boldsymbol{x}')$ ▷ Proposal distribution. This is typically Gaussian distribution with mean $= \boldsymbol{x}$.

1: **function** Metropolis Hastings
2: **for** $t = 0, 1, \ldots, T$ **do**
3: $\boldsymbol{x} \sim q(\cdot|\boldsymbol{x}^{(t)})$
4: $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}$ 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 $\boldsymbol{x}^{(t)}$

7.4 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:

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

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:

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

When we get only small importance weights, $\mathcal{L}_{\text{eff}}$ 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 biased or have high variance easily, we need to pay attention to the distribution of weights.

7.5 Markov chain Monte-Carlo (MCMC) sampling

MCMC samples each weights 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**.

7.5.1 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 far away from each other, MCMC can sample from multi-modal distributions.

7.5.2 Gibbs sampling

Another variant of Metropolis 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:

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

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

8 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.

8.1 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 $\boldsymbol{X}' \in \mathbb{R}^{N \times d}$ where $d < D$. 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:

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

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

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

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:

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

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:

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

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 feature non-linearly unlike PCA and kernel PCA and Isomap are also included in MDS.

8.2 t-distributed stochastic neighbor embedding (t-SNE)

t-SNE is mostly used 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

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

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

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

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:

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

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

Note that σ_i is computed 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.