STATS 601 - Statistical Learning

By Xuelin Zhu
Email: xuelin@umich.edu

September 22, 2023

This is a notes sequence that follows the lectures by Professor Gongjun Xu in 2023 Winter. Reference textbook is *The Elements of Statistical Learning* by Trevor Hastie[1] and *Pattern Recognition and Machine Learning* by Bishop[2] . This notes sequence serves as an outline since the lecture contents are not in the same order with the book. Details and supplemented examples are still left in the book, and I will leave the corresponding index in the book for reference.

BTW, there should be one more chapter about multivariate normal, but it is in the handwriting note.

Contents

Chapter 1

Principal Component Analysis

Principal Component Analysis is a powerful statistical tool for dimension reduction, which has been applied widely in many areas. §1.1 to §1.3 focus on the classical PCA, its assumption, and the mathematical foundation for estimation. §1.4 to §1.6 focus on other understandings and views of PCA, rather than the classical form. §1.7 to §1.9 is about its extensions and applications.

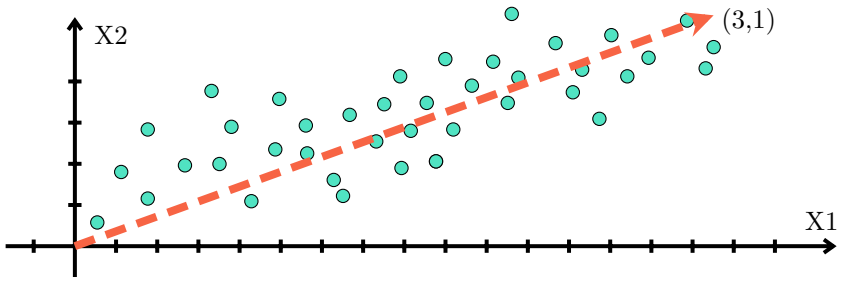
1.1 Settings and Goals

Suppose we are having such kind of data:

Table 1.1: Raw data of sample size N and dimension $(q+4)$

Patient	Response	Age	Gender	Gene 1	Gene 2	$\cdots$	Gene q
#1							
#2							
#3							
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
#N							

The number of column of a data matrix is called the *dimension*. In this case, we are having a data with sample size N and dimension $q+4$. Now we further assume that $N \ll q$, which means the row vectors could not span the $\mathbb{R}^q$ space, and they lie mostly near a low dimensional linear space $\mathbb{R}^p$ $(p \ll q)$. This is the setting of PCA. And here below is an illustration.



As we see, the data do not lie on the whole $\mathbb{R}^2$ space, but nearly concentrate on a line, which is a $\mathbb{R}^1$ linear subspace. And the similar situation may happen in the high dimensional data. Maybe for a 100000-dimensional data, the dimension of the linear space spanned by its row vectors is only 1000. With some approximation, then it could be further reduced to 200-dimension. Our goal with PCA is to find the lower dimensional space $\mathbb{R}^p$, which could characterize the original space $\mathbb{R}^q$.

The most extreme situation is: the n row vectors span a $\mathbb{R}^n$ space. In this case, no lower dimension space could be used for approximation. Therefore, we need to assume that the raw data $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^q$ lie mostly near a low dimension linear space $\mathbb{R}^p$, say $\mathcal{M}$. For this $\mathcal{M}$, there exists an orthogonal base:

$$\boldsymbol{u_1}, \ldots, \mathbf{u}_p \in \mathbb{R}^q \quad \text{such that} \quad \forall \mathbf{v} \in \mathcal{M}, \quad \mathbf{v} = x_1\mathbf{u}_1 + \cdots + x_p\mathbf{u}_p = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_p \end{bmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{pmatrix}.$$

For convenience, we denote

$$U_{q\times p} = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_p \end{bmatrix}, \quad \text{with} \quad U^\top U = I_p \ \text{(note the order!)}.$$

Then every vector in this $\mathcal{M}$ space has a representation using the basis of U's column vectors. Because we assume the raw data $\mathbf{y}$ nearly sits in this space, we have

$$\forall\, \mathbf{y}_i, \quad i \in \{1, \ldots, N\}: \quad \mathbf{y}_i \approx x_1\mathbf{u}_1 + \cdots + x_p\mathbf{u}_p = U\mathbf{x}_i,$$

where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ is the coordinates under basis U. Now we formally state our assumption of PCA.

Problem Setup. (PCA) Suppose the raw observed data are q-dimensional $\mathbf{y}_1, \ldots, \mathbf{y}_n$ from an unknown population $Y \sim \text{Dist}(\boldsymbol{\mu}, \Sigma)$, and the components of $\mathbf{y}$ nearly lie in a lower p-dimensional linear subspace ($p < q$). Our goal is to find out the p-dimensional subspace, more specifically, to find out the orthogonal basis matrix $U_{q\times p}$ to get

$$\mathbf{y}_{q\times 1} \mapsto \mathbf{x}_{p\times 1} \quad \text{such that} \quad \mathbf{x} = U^\top\mathbf{y} \quad \text{with} \quad U_{q\times p} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_p. \end{pmatrix}$$

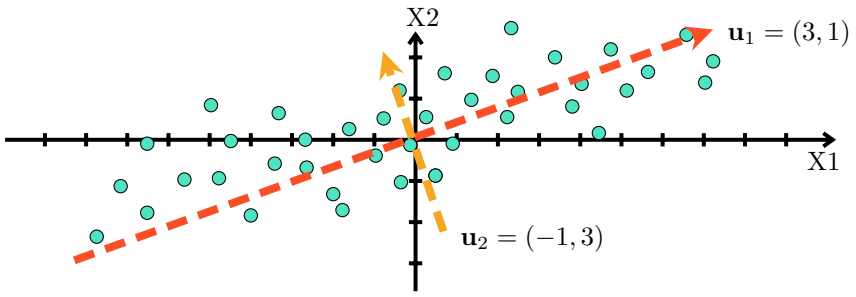
Here diverge two version. If we know the population covariance matrix Σ, then we are doing population PCA, using Σ. However, if we have the sample, we are doing the sample version PCA, using S.

1.2 Estimation of the Lower Dimensional Space

Now we suppose we are dealing with sample data, instead of the population. There are many ways to estimate the U, one of which was introduced in MA304, using sequential selection by maximization lemma. Now, we introduce another way to derive the first p PCs simultaneously, using SVD or spectral decomposition.

1.2.1 Case I: $p = 1$

We first deal with the easiest situation: all data nearly lie on a 1 dimensional space. So, our goal is to find the unit direction vector $\mathbf{u}_1$ such that $\mathbf{u}_1^\top\mathbf{u}_1 = 1$. For illustration, let's consider the following example.



If we project the sample on the $\mathbf{u}_1$ direction, the variance of the coordinates will be much larger than that if we project on the $\mathbf{u}_2$ direction. Note that we are projecting only on one direction, so we can express

$$\text{Proj}_{\mathbf{u}_1}(\mathbf{y}_i) = \mathbf{u}_1(\mathbf{u}_1^\top \mathbf{u}_1)^{-1}\mathbf{u}_1^\top \mathbf{y}_i = \mathbf{u}_1\mathbf{u}_1^\top \mathbf{y}_i = \left(\mathbf{u}_1^\top \mathbf{y}_i\right)\mathbf{u}_1 = x_{i1}\mathbf{u}_1,$$

where all x_{i1}'s are scalar, not vector, whose first index i of x_{i1} indicates the obs number, the second 1 indicates the projection on the first direction. If $\mathbf{u}_1$ direction makes the variance of projection coordinates $\{x_{11}, x_{21}, \ldots, x_{N1}\}$ largest, we say $\mathbf{u}_1$ direction captures the most variation of $\mathbf{y}$'s.

Proposition 1. *To find the first PC of the sample $\mathbb{Y}$, we optimize the following function:*

$$\max_{\mathbf{u}_1 \in \mathbb{R}^q} \frac{1}{N-1}\sum_{i=1}^{N}(x_{i1} - \bar{x}_1)^2 \quad \textit{subject to} \quad \mathbf{u}_1^\top \mathbf{u}_1 = 1, \quad \textit{where} \quad \bar{x}_1 = \frac{1}{N}\sum_{j=1}^{N} x_{i1}.$$

The solution is

$\mathbf{u}_1$ = the eigenvector of S with the largest eigenvalue.

Proof. It is derived that $x_{i1} = \mathbf{u}_1^\top \mathbf{y}_i$, and then $\bar{x}_1 = \mathbf{u}_1^\top \bar{\mathbf{y}}$. It then follows that

$$\begin{aligned}
\frac{1}{N-1}\sum_{i=1}^{N}(x_{i1} - \bar{x}_1)^2 &= \frac{1}{N-1}\sum_{i=1}^{N}\left(\mathbf{u}_1^\top \mathbf{y}_i - \mathbf{u}_1^\top \bar{\mathbf{y}}\right)^2 \\
&= \frac{1}{N-1}\sum_{i=1}^{N}\left[\mathbf{u}_1^\top (\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^\top \mathbf{u}_1\right] \\
&= \mathbf{u}_1^\top \left[\frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{y}_i - \bar{\mathbf{y}})(\mathbf{y}_i - \bar{\mathbf{y}})^\top\right]\mathbf{u}_1 \\
&= \mathbf{u}_1^\top S \mathbf{u}_1.
\end{aligned}$$

Therefore, our goal becomes to

$$\max_{\mathbf{u}_1 \in \mathbb{R}^q} \mathbf{u}_1^\top S \mathbf{u}_1 \quad \text{subject to} \quad \mathbf{u}_1^\top \mathbf{u}_1 = 1. \tag{1.1}$$

Introducing a Lagrange multiplier for $\mathbf{u}_1^\top \mathbf{u}_1 = 1$, we set

$$\frac{\partial}{\partial \mathbf{u}_1}\left\{\mathbf{u}_1^\top S \mathbf{u}_1 - \lambda(\mathbf{u}_1^\top \mathbf{u}_1 - 1)\right\}\Big|_{\hat{\mathbf{u}}_1} = 0 \quad \Rightarrow \quad S\hat{\mathbf{u}}_1 = \lambda \hat{\mathbf{u}}_1. \tag{1.2}$$

This means, the optimization is obtained only if $\hat{\mathbf{u}}_1$ is an eigenvector of S. But which eigenvector is the

solution? We need to plug (2) back to (1), and see the value of $\mathbf{u}_1^\top S\mathbf{u}_1$.

$$\begin{aligned}\max_{\mathbf{u}_1\in\mathbb{R}^q}\left\{\mathbf{u}_1^\top S\mathbf{u}_1-\lambda(\mathbf{u}_1^\top\mathbf{u}_1-1)\right\} &= \max_{\mathbf{u}_1\in\mathbb{R}^q}\left\{\mathbf{u}_1^\top\lambda\mathbf{u}_1-\lambda\left(\mathbf{u}_1^\top\mathbf{u}_1-1\right)\right\}\\ &= \max_{\mathbf{u}_1\in\mathbb{R}^q}\lambda.\end{aligned}$$

By (2), λ is the eigenvalue corresponding to the eigenvector $\mathbf{u}_1$. So, by choosing $\mathbf{u}_1$ to be the eigenvector with the largest eigenvalue, we get this optimized. □

1.2.2 Case II: $p > 1$

Now we relax the condition $p = 1$, assuming the raw data $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\} \in \mathbb{R}^q$ lie nearly on a p dimensional subspace. Suppose $(\mathbf{u}_1, \ldots, \mathbf{u}_p)$ is an orthogonal basis of this $\mathbb{R}^p$ space, and we write

$$U = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_p\end{bmatrix}_{q\times p}, \quad \text{where} \quad \mathbf{u}_k \in \mathbb{R}^q \text{ for all k} \in \{1, 2, \ldots, p\}.$$

Then every observation $\mathbf{y}_i$ has p projection coordinates now:

$$\forall i \in \{1, \ldots, N\}: \quad \mathbf{y}_i \in \mathbb{R}^q \quad \Longrightarrow \quad \mathbf{x}_i = \begin{pmatrix} x_{i1}\\ x_{i2}\\ \vdots\\ x_{ip}\end{pmatrix} = \begin{pmatrix}\mathbf{u}_1^\top\mathbf{y}_i\\ \mathbf{u}_2^\top\mathbf{y}_i\\ \vdots\\ \mathbf{u}_p^\top\mathbf{y}_i\end{pmatrix} = U^\top\mathbf{y}_i.$$

And as usual, we call $\mathbf{x}_i$ as the projection coordinates of $\mathbf{y}_i$ on the basis U. In fact, we are using the projection to approximate the true $\mathbf{y}_i$, i.e.

$$\mathbf{y}_i \approx x_{i1}\mathbf{u}_1 + x_{i2}\mathbf{u}_2 + \cdots + x_{ip}\mathbf{u}_p = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_p\end{bmatrix}\begin{pmatrix} x_{i1}\\ x_{i2}\\ \vdots\\ x_{ip}\end{pmatrix} = U_{q\times p}U_{p\times q}^\top\,\mathbf{y}_i.$$

This is an interpretation of what we are doing. Now, let's begin the estimation. Similarly, we also want the projection coordinates capture more variation of $\mathbf{y}$'s.

Proposition 2. *To find the first p PC's of the sample $\mathbb{Y}$, we optimize the following function:*

$$\max_{U\in\mathbb{R}^{q\times p}}\left(\text{sum of sample variance of }\begin{bmatrix}\{x_{11} & x_{21} & \cdots & x_{N1}\}\\ \{x_{12} & x_{22} & \cdots & x_{N2}\}\\ & & \vdots & \\ \{x_{1p} & x_{2p} & \cdots & x_{Np}\}\end{bmatrix}\right) \quad \textit{subject to} \quad U^\top U = I_p.$$

One solution is $U = \begin{bmatrix}\mathbf{u}_1 & \ldots & \mathbf{u}_p\end{bmatrix}$, *where*

$$\mathbf{u}_k = \text{the eigenvector of S with the } k-\text{th largest eigenvalue.}$$

And any rotation of the p column vectors $U_{q\times p}O_{p\times p}$, where $O^\top O = I_p$, is another solution.

Proof. Let's first simplify the optimization function. In Proposition 2, we showed

$$\text{sample variance of } \{x_{11}, \dots, x_{N1}\} = \mathbf{u}_1^\top S \mathbf{u}_1.$$

So, with the same argument, we have

$$\begin{aligned} \text{sample variance of } \{x_{11}, \dots, x_{N1}\} &= \mathbf{u}_1^\top S \mathbf{u}_1, \\ &\vdots \\ \text{sample variance of } \{x_{1p}, \dots, x_{Np}\} &= \mathbf{u}_p^\top S \mathbf{u}_p. \end{aligned}$$

This leads to

$$\text{sum of sample variance of } \begin{bmatrix} \{x_{11} & x_{21} & \cdots & x_{N1}\} \\ \{x_{12} & x_{22} & \cdots & x_{N2}\} \\ & & \vdots & \\ \{x_{1p} & x_{2p} & \cdots & x_{Np}\} \end{bmatrix} = \sum_{i=1}^{p} \mathbf{u}_i^\top S \mathbf{u}_i.$$

Our restrictions for $\mathbf{u}_i$'s are

$$U^\top U = I_p \quad \Longleftrightarrow \quad \begin{cases} \mathbf{u}_i^\top \mathbf{u}_i = 1 & \text{for } j = 1, \dots, p, \\ \mathbf{u}_i^\top \mathbf{u}_j = 1 & \text{for } i \neq j. \end{cases}$$

Introducing λ_{ij} for every restriction, our optimization function becomes

$$\max_{\mathbf{u}_1, \dots, \mathbf{u}_p \in \mathbb{R}^q} \left\{ \sum_{i=1}^{p} \mathbf{u}_i^\top S \mathbf{u}_i - \sum_{i=1}^{p} \lambda_{ii} (\mathbf{u}_i^\top \mathbf{u}_i - 1) - \sum_{1 \leq i \neq j \leq p} \lambda_{ij} (\mathbf{u}_i^\top \mathbf{u}_j - 0) \right\}. \tag{1.3}$$

If we write the Lagrange multipliers as a matrix

$$\Lambda = \begin{pmatrix} \lambda_{11} & \lambda_{12} & \cdots & \lambda_{1p} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{p1} & \lambda_{p2} & \cdots & \lambda_{pp} \end{pmatrix},$$

optimization (3) becomes (why this step holds is left in Notes 3.1.1)

$$\max_{\mathbf{u}_1, \dots, \mathbf{u}_p \in \mathbb{R}^q} \Big\{ tr\left(U^\top S U\right) - tr\big[\Lambda \left(U^\top U - I_p\right) \big] \Big\}.$$

Using matrix derivative (Notes 2.1), and setting the derivative to be zero, we get

$$\frac{\partial}{\partial U} \Big\{ tr\left(U^\top S U\right) - tr\big[\Lambda \left(U^\top U - I_p\right) \big] \Big\} \Big|_{\hat{U}} = \mathbf{0} \quad \Rightarrow \quad S\hat{U} = \hat{U}\Lambda.$$

Note that the Λ matrix is symmetric, so we use spectral decomposition, and it follows that

$$\Lambda = O \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_p \end{pmatrix} O^\top \quad \Rightarrow \quad S\hat{U}O = \hat{U}O \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_p \end{pmatrix}.$$

For convenience, denote $U^* := \hat{U}O$ and suppose $U^* = \begin{bmatrix} \mathbf{u}_1^* & \dots & \mathbf{u}_p^* \end{bmatrix}$. Then we have a very simple expression

$$SU^* = U^* \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_p \end{pmatrix} \quad \Longleftrightarrow \quad \begin{pmatrix} S\mathbf{u}_1^* & S\mathbf{u}_2^* & \cdots & S\mathbf{u}_p^* \end{pmatrix} = \begin{pmatrix} d_1\mathbf{u}_1^* & d_2\mathbf{u}_2^* & \cdots & d_p\mathbf{u}_p^* \end{pmatrix}.$$

This means, the optimization is obtained only if every $\mathbf{u}_k^*$ above is an eigenvector of S, and correspondingly, d_k is the eigenvalue. And here comes the same question: which eigenvectors should we take since S has $\min(N, q)$ eigenvectors? We also plug the possible solution into the maximization function. Before plugging, note that

$$S\hat{U} = \hat{U}\Lambda, \quad U^* = UO \quad \text{and} \quad U^\top U = O^\top (U^*)^\top (U^*) O = I_p,$$

where the last equality holds because $U^* = \begin{bmatrix} \mathbf{u}_1^* & \dots & \mathbf{u}_p^* \end{bmatrix}$ with each $\mathbf{u}_k^*$ being an eigenvectors of S (eigenvectors are orthogonal). Then we look back at the maximization:

$$\begin{aligned} \max_{\mathbf{u}_1,\dots,\mathbf{u}_p \in \mathbb{R}^q} \left\{ tr\left(U^\top S U\right) - tr\left[\Lambda\left(U^\top U - I_p\right)\right] \right\} &= \max_{\mathbf{u}_1,\dots,\mathbf{u}_p \in \mathbb{R}^q} \left\{ tr\left(U^\top U \Lambda\right) - tr\left[\Lambda\left(U^\top U - I_p\right)\right] \right\} \\ &= \max_{\mathbf{u}_1,\dots,\mathbf{u}_p \in \mathbb{R}^q} \left\{ tr\left(\Lambda\right) \right\} \\ &= \max_{\mathbf{u}_1,\dots,\mathbf{u}_p \in \mathbb{R}^q} tr \left\{ O \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_p \end{pmatrix} O^\top \right\} \\ &= \max_{\mathbf{u}_1,\dots,\mathbf{u}_p \in \mathbb{R}^q} \sum_{i=1}^{p} d_i. \end{aligned}$$

Since every d_k is the eigenvalue corresponding to the eigenvector $\mathbf{u}_k^*$ of S, we choose $d_1, \dots, d_p$ to be the largest eigenvalues, and $\mathbf{u}_k^*$ is the corresponding eigenvectors, it gets optimized. □

Note that the last word of Proposition 2 says: the first p PC's of the sample $\mathbb{Y}$ is NOT unique. The reason for it is we are doing optimization on p PC's simultaneously. For more detail for the identifiable issue, refer Notes 3.1.1. **This means, we can rotate the first p PC's, but remain at the same explained proportion of total variance.** However, the explained proportion of each PC from 1 to p may change.

1.3 Properties of the Score Vector

Definition 1. In the PCA setting, $\mathbf{x}$ is the **principal component transformation** of $\mathbf{y}$, and is called the **score vector**. The linear transform matrix U is called the **loading matrix**.

We first focus on the population version PCA, since then the score vector $\mathbf{x}$ becomes a affine transformation of $\mathbf{y}$ and is easy to study. Recall the estimation procedure (in the last note), the optimization function is

$$\max_{U:\, U^\top U = I_p} \sum_{j=1}^{p} \mathrm{Var}(X_j) = \max_{U:\, U^\top U = I_p} tr(U^\top \Sigma U).$$

If we denote that largest p eigenvalues of Σ as $\lambda_1, \dots, \lambda_p$ with corresponding eigenvectors $\mathbf{u}_1, \cdots, \mathbf{u}_p$, then

the optimization is obtained when

$$\hat{U} = \begin{pmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_p \end{pmatrix} O_{p\times p} =: \hat{U}_0 O \quad \text{where } O_{p\times p} \text{ is any orthogonal matrix,}$$

and correspondingly, the score vector is

$$\hat{\mathbf{X}} = \hat{U}^\top \mathbf{Y} = O^\top \hat{U}_0 \mathbf{Y}$$

Therefore, the solution is NOT unique, and is up to an rotation in the $\mathbb{R}^p$ subspace. But with or without the rotation O, we can derive some properties of $\mathbf{X}$.

Proposition 3. *If $\mathbf{X}$ is the score vector of the PCA for $\mathbf{Y} \sim$ Dist $(\boldsymbol{\mu}, \Sigma)$, then*

$$\mathbb{E}\mathbf{X} = \hat{U}^\top \boldsymbol{\mu} \quad \textit{and} \quad \mathrm{Var}\mathbf{X} = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p \end{pmatrix}, \quad \textit{where } \lambda' s \textit{ are eigenvalues of } \Sigma.$$

Proof of this proposition is trivial, thereby omitted. It tells us the score vectors are uncorrelated, and each variance equals to the corresponding eigenvalue. The same result holds for the sample version, which provides us with a way to choose p.

Definition 2. For the row data $\mathbf{Y}$ and its PCA scores $\mathbf{X}$, the total variance of $\mathbf{Y}$ is

$$\text{Total variance} = \sum_{j=1}^{q} \mathrm{Var}X_j = \sum_{j=1}^{q} \lambda_j = tr(\Sigma).$$

We say the j-th PC (X_j) accounts for

$$\frac{\lambda_j}{tr(\Sigma)}$$

proportion of the total variance, and the first p PC's captures

$$\frac{\lambda_1 + \cdots + \lambda_p}{\lambda_1 + \cdots + \lambda_p + \cdots + \lambda_q}$$

of the total variation of $\mathbf{Y}$. For the sample version, the notation is the same.

Using this interpretation, we can use the proportion of explained total variance to choose p.

1.4 Sequential Estimation of PCA

In §1.2, an optimization to solve $\hat{U} = (\mathbf{u}_1, \cdots, \mathbf{u}_p)$ simultaneously is proposed. But recall that in doing this way, the solution is not unique and up to a rotation. In fact, if we optimize the problem one component by one component, we can avoid the rotation and identifiable issue.

Algorithm 1. The procedure to do sequential PCA for $\mathbf{Y}$ is:

(a) solving PC1 by

$$\max_{\mathbf{u}_1 \in \mathbb{R}^q} \mathbf{u}_1^\top \Sigma \mathbf{u}_1 \quad \text{subject to} \quad \mathbf{u}_1^\top \mathbf{u}_1 = 1;$$

(b) solving PC2 by

$$\max_{\mathbf{u}_2 \in \mathbb{R}^q} \mathbf{u}_2^\top \Sigma \mathbf{u}_2 \quad \text{subject to} \quad \begin{cases} \mathbf{u}_2^\top \mathbf{u}_2 = 1 \\ \mathbf{u}_2^\top \mathbf{u}_1 = 0 \end{cases}$$

(c) Repeating (a) and (b) till we get desired explained proportion.

1.5 Other Explanations of PCA

The classical PCA does not rely on strong statistical assumptions, so it is widely applicable. But we also have some statistical ways to look at PCA such that we can use some familiar tools such as MLE, LSE and hypothesis testing.

1.5.1 Probabilistic PCA

Assumption 1. For every observation, the raw data $\mathbf{Y} \in \mathbb{R}^q$ is q-dimensional, and is generated by

$$\mathbf{Y}_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}\mathbf{X}_p + W_q \quad \text{where} \quad \begin{cases} \mathbf{X} \sim \mathrm{N}_p(\mathbf{z}, I_p), \\ W \sim \mathrm{N}_q(\mathbf{z}, \sigma^2 I_q), \\ W \perp\!\!\!\perp \mathbf{X} \ \text{ for every obs}, \\ \text{observations are independent.} \end{cases}$$

Equivalently, we are assuming the population

$$\mathbf{Y} \sim \mathrm{N}_q\left(\boldsymbol{\mu}, \sigma^2 I_q + \Lambda\Lambda^\top\right).$$

Note that the only observed data is $\mathbf{Y}$, the $(\mathbf{X}, W)$ are latent variables, $(\boldsymbol{\mu}, \sigma^2, \Lambda)$ are parameters. If we estimate this model parameters, we are actually decomposing the covariance structure of $\mathbf{Y}$ to be

$$\Sigma \mapsto \sigma^2 I_q + \Lambda\Lambda^\top.$$

In other words, we are using a low rank structure Λ to estimate a high rank structure Σ. If we accept this setting, the problem becomes: how to estimate the parameters? A usual way is the MLE. Suppose we have the sample

$$\mathbf{y}_i \overset{\text{iid}}{\sim} \mathrm{N}\left(\boldsymbol{\mu}, \sigma^2 I_q + \Lambda\Lambda^\top\right).$$

Since $\mathbf{X}$ does not appear in the explicit form of $\mathbf{Y}$'s marginal distribution, we can use the it to do MLE. The optimization problem is:

$$\max_{\boldsymbol{\mu}, \sigma^2, \Lambda} L(\mathbf{y}_1, \cdots, \mathbf{y}_n \mid \boldsymbol{\mu}, \sigma^2, \Lambda) \quad \Rightarrow \quad \left(\hat{\boldsymbol{\mu}}, \hat{\sigma}^2, \hat{\Lambda}\right).$$

The derivation is not easy, so we directly give out the explicit solution. If we have already set the lower

dimension is p, the MLE is

$$\hat{\boldsymbol{\mu}} = \bar{\mathbf{y}}, \quad \hat{\sigma}^2 = \frac{\lambda_{p+1} + \cdots + \lambda_q}{q-p}, \quad \text{and} \quad \hat{\Lambda} = \hat{U} \begin{pmatrix} \lambda_1 - \hat{\sigma}^2 & 0 & \cdots & 0 \\ 0 & \lambda_2 - \hat{\sigma}^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_p - \hat{\sigma}^2 \end{pmatrix} O_{p\times p},$$

where $\hat{U}$ and λ's are the same as classical PCA estimation, and O is some orthogonal matrix.

1.5.2 Least Square's View

In Probabilistic PCA, we view $\mathbf{X}$ as random variables. Correspondingly, we can also view it as fixed but unknown parameters, just as the difference in fixed or random effect models.

Assumption 2. For every observation, the raw data $\mathbf{Y}_i \in \mathbb{R}^q$ is independent, and is generated by

$$\mathbf{Y}_i = \boldsymbol{\mu} + U\mathbf{X}_i + \text{error}_i, \quad i \in \{1, 2, \ldots, n\}$$

In this case, since we don't have any distribution assumption for the model, MLE is not applicable. But the LSE is a good natural choice. The unknown parameters are $(\boldsymbol{\mu}, U, \mathbb{X})$, where $\mathbb{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)^\top$ is the design matrix. The optimization problem is

$$\min_{\boldsymbol{\mu}, U, \mathbb{X}} \sum_{i=1}^{n} (\mathbf{Y}_i - \boldsymbol{\mu} - U\mathbf{X}_i)^2 \quad \text{subject to} \quad U^\top U = I_p.$$

The derivation is left in Homework 2, and the solution is

$$\boldsymbol{\mu} = \bar{\mathbf{Y}}, \quad \hat{U} = \left(\mathbf{u}_1, \cdots, \mathbf{u}_p\right), \quad \text{and} \quad \hat{\mathbf{X}}_i = U^\top(\mathbf{Y}_i - \hat{\boldsymbol{\mu}}).$$

1.6 Singular Value Decomposition for PCA

In the last lecture note, we have seen the procedure to estimate U in the PCA. It is closely related to spectral decomposition of Σ. Now we introduce its relation with the SVD of Σ, which is how the computer works when we require it to do the PCA.

Suppose we have the centered sample data matrix

$$\mathbb{Y}_{n\times q} = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_n^\top \end{pmatrix}.$$

And the singular value decomposition of $\mathbb{Y}$ is

$$\mathbb{Y}_{n\times q} = U_{n\times r} D_{r\times r} V_{r\times q}^\top,$$

where $U_{n\times r}$'s column vectors are orthogonal, $D_{r\times r}$ is diagonal with singular values of $\mathbb{Y}$, and $V_{q\times r}$'s column

vectors are also orthogonal, i.e.

$$U^\top U = I_r, \quad V^\top V = I_r, \quad \text{and} \quad D = \begin{pmatrix} d_1 & & \\ & \ddots & \\ & & d_r \end{pmatrix} \quad \text{with} \quad d_1 > \cdots > d_r > 0.$$

Here the value of r is uniquely determined by $rank(\mathbb{Y}) \leq \min(n, q)$. And normally, we can rearrange the order the U, V, D so that the order of d's is decreasing by their values.

What is the relation between PCA and SVD? Since the data is centered, we have

$$S = \frac{1}{n-1}\mathbb{Y}^\top\mathbb{Y} = \frac{1}{n-1}\sum_{i=1}^{n}\mathbf{y}_i\mathbf{y}_i^\top.$$

Multiplying on both side by $n-1$, and using SVD, it follows that

$$(n-1)S = \mathbb{Y}^\top\mathbb{Y} = (VDU^\top)(UDV^\top) = VD^2V^\top.$$

Note that

$$(n-1)S = VD^2V^\top = \begin{pmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_r \end{pmatrix} \begin{pmatrix} d_1^2 & & \\ & \ddots & \\ & & d_r^2 \end{pmatrix} \begin{pmatrix} \mathbf{v}_1^\top \\ \vdots \\ \mathbf{v}_r^\top \end{pmatrix}.$$

This means v_i's are the eigenvectors of S, and d_i^2's are eigenvalues of S, and normally

$$\begin{cases} \text{if } n > q & \Rightarrow \quad r = q \text{ (low-dimension problem)}, \\ \text{if } n < q & \Rightarrow \quad r = n \text{ (high-dimension problem)}. \end{cases}$$

Be very careful that the matrix notation is not consistent with classical PCA derivation! (because we are consistent with SVD notation $UDV^\top$ now). Recall in Assumption 1, for every observation $\mathbf{y}_i$, its score vector is $\mathbf{x}_i = U^\top\mathbf{y}_i$. Because here the loading matrix's notation becomes V, using SVD, the score vector for every observation is $\mathbf{x}_i = V^\top\mathbf{y}_i$. Writing them as a score matrix, we finnaly get

$$\mathbb{X}_{n\times r} = \mathbb{Y}_{n\times q}V_{q\times r}.$$

If we further simplify this expression with $\mathbb{Y} = UDV^\top$, it becomes

$$\mathbb{X}_{n\times r} = U_{n\times r}D_{r\times r}.$$

If we don't want all PC's, we can choose the first $p < r$ vectors to do the truncated SVD, and the equality becomes approximation:

$$\mathbf{Y}_{n\times q} \approx U^*_{n\times p}D^*_{p\times p}(V^*)^\top_{p\times q}.$$

Normally, using this SVD way is quicker to do PCA, because spectral decomposition of S would not be easy if $n \ll q$.

1.7 Sparse Structure of PCA

In §1.7-§1.9, we will cover different forms of PCA extensions, see how it works for different models. However, the optimization in every setting differs a lot from each other. So, the specific details to estimate, to optimize are omitted. But the general idea of optimization is provided.

To illustrate this idea, we start with an example.

Example. Since the only constraints in the PCA optimization is that $U^\top U = I_p$, the score vector may be like the following form:

$$X_1 = 0.14 \times age - 0.22 \times weight + 0.18 \times height,$$

or the following form:

$$X_1 = 0 \times age + 0.9 \times weight + 0.44 \times height.$$

Which one would you want? Obviously, the first form is very hard to interpret or give explanation for the group $(0.14 \times age - 0.22 \times weight + 0.18 \times height)$. By contrast, the second form may indicate that X_1 is an index related to the person's body, but not related to his age.

This is the intuition of the sparse structure: we want to know which Y_i's collectively play a role. In the mathematical expression, we desire every vector $\mathbf{u}_i$ in

$$U_{q\times p} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_p \end{pmatrix}$$

to have some number significant, and others close to 0, just as $\boldsymbol{\mu}_1 = (0,\ 0,\ 0.7,\ 0.7,\ 0)^\top$. The idea to restrict some terms to be zero in estimation refers to the terminology *sparsity*. Now we have the goal, but how to achieve this in the estimation? Three ways are provided.

1.7.1 LASSO Penalty

As known, Lasso regression shrinks the coefficients to 0. So, we can borrow this idea to sequential PCA.

Algorithm 2. To get the sparse PCA structure, we do the following steps:

(a) Choose a tuning parameter $t > 0$.

(b) Find the first PC by

$$\max_{\mathbf{u}_1 \in \mathbb{R}^p} \mathbf{u}_1^\top S \mathbf{u}_1 \quad \text{subject to} \quad \begin{cases} \mathbf{u}_1^\top \mathbf{u}_1 = 1, \\ ||\mathbf{u}_1|| \le t. \end{cases}$$

(c) Find the second PC by

$$\max_{\mathbf{u}_2 \in \mathbb{R}^p} \mathbf{u}_2^\top S \mathbf{u}_2 \quad \text{subject to} \quad \begin{cases} \mathbf{u}_2^\top \mathbf{u}_2 = 1, \\ \mathbf{u}_2^\top \mathbf{u}_1 = 0, \\ ||\mathbf{u}_2|| \le t. \end{cases}$$

(d) Repeat (c) until we get PC's attaining the wanted explained proportion of total variance.

This idea is proposed by *Ian Jolliffe* in his paper in 2003. But practically, this optimization is not easy since the function is not convex.

1.7.2 Least Square in the Fixed Effect View

Remember the least square's view of PCA and the optimization function is

$$\min_{\boldsymbol{\mu},U,\mathbb{X}} \sum_{i=1}^{n} \left(\mathbf{y}_i - \boldsymbol{\mu} - U\mathbf{x}_i\right)^2 \quad \text{subject to} \quad U^\top U = I_p.$$

Treating $\mathbb{X}$ as unknown but fixed effect, we can apply least square with a shrinkage penalty to optimize this problem and get sparsity.

Algorithm 3. Suppose the observed data matrix $\mathbb{Y}_{n\times q}$ is already centered, and the desired dimension p is given.

(a) Choose the tuning parameters for LASSO penalty $\lambda \geq 0$, and ridge penalty $\alpha \geq 0$.

(b) We set (don't start) the optimization function to be

$$\min_{U,\Theta\in\mathbb{R}^{q\times p}} \sum_{i=1}^{n} \left|\left|\mathbf{y}_i - \Theta U^\top \mathbf{y}_i\right|\right|_2^2 + \lambda||U||_1 + \alpha||U||_2^2 \quad \text{subject to} \quad \Theta^\top\Theta = I_p. \tag{1.4}$$

(c) Set initial values of $U_{(0)}$ and $\Theta_{(0)}$.

(d) Fix $\Theta = \Theta_{(0)}$, minimizing (1) w.r.t. U and set the optimal value of U to be $U_{(1)}$.

(e) Fix $U = U_{(1)}$, minimizing (1) w.r.t. Θ and set the optimal value of Θ to be $\Theta_{(1)}$.

(f) Repeat (d) and (e), until it converges.

The reason to minimizing (1) instead of the original function is that we treat U as the loading matrix and $\mathbf{x}_i = U^\top\mathbf{y}_i$, and the Θ matrix is a basis for the score vectors (NOT rigorous). And the restrictions are put on the loading matrix U, not Θ. Another thing worth noticing is that simultaneous optimization of U and Θ is normally infeasible, because it involves to many parameters. For more details, refer Ch 14.4.5 of *Elements of Statistical Learning*.

1.7.3 Sparse Probabilistic PCA

Don't forget we have one more way to understand PCA, and it also corresponds to a way of sparse structure. Recall the model assumption.

Assumption 3. For every observation, the raw data $\mathbf{Y} \in \mathbb{R}^q$ is q-dimensional, and is generated by

$$\mathbf{Y}_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}\mathbf{X}_p + W_q \quad \text{where} \quad \begin{cases} \mathbf{X} \sim \mathrm{N}_p(\mathbf{z}, I_p), \\ W \sim \mathrm{N}_q(\mathbf{z}, \sigma^2 I_q), \\ W \perp\!\!\!\perp \mathbf{X} \text{ for every obs}, \\ \text{observations are independent.} \end{cases}$$

Equivalently, we are assuming the population

$$\mathbf{Y} \sim \mathrm{N}_q\left(\boldsymbol{\mu}, \sigma^2 I_q + \Lambda\Lambda^\top\right).$$

Since the marginal distribution of $\mathbf{Y}$ is explicit, we can directly do MLE with LASSO penalty:

$$\min_{\boldsymbol{\mu},\sigma^2,\Lambda} -\log L(\boldsymbol{\mu},\sigma^2,\Lambda) + \lambda||\Lambda||_1.$$

Once desired dimension p and tuning parameter λ are given, the optimization is doable. Gradient descending method, or *EM algorithm* (to be introduced) are applicable.

1.8 Non-linear PCA

Let's use another way to understand PCA. Suppose the observable $\mathbf{Y}$ is generated or affected by some latent hidden variable $\mathbf{X}$. This means, to truly understand the whole relation, we are looking for

$$\mathbf{Y} \xrightarrow[f_\theta]{\text{encoder}} \mathbf{X} \xrightarrow[g_\beta]{\text{decoder}} \mathbf{Y},$$

where the $\mathbf{Y}$ on the left is the input, the $\mathbf{X}$ in the middle is the latent representation, and the $\mathbf{Y}$ on the right is the output. We are estimating the encoder and decoder functions f_θ and g_β, and the optimization function is

$$\min_{\theta,\,\beta} \sum_{i=1}^{n} ||\mathbf{Y}_i - g_\beta(f_\theta(\mathbf{Y}_i)||^2 + \text{constraints}.$$

If the encoder and decoder functions are linear, then it is the PCA case. Otherwise, it is called non-linear PCA, which is commonly used in neural network.

1.9 PCA for Binary Data

Though we are normally using PCA with continuous data, it could be applied to the categorical data. In other words, we are summarizing continuous data from the discrete data! But this leads to complected models. Let's focus on the binary type.

Assumption 4. Suppose $\mathbf{Y} \in \mathbb{R}^q$ is one observation, and its every component is Bernoulli distributed:

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ \vdots \\ Y_q \end{pmatrix} \quad \text{and} \quad Y_k \sim \text{Ber}(p_k), \;\; k \in \{1,\dots,q\}.$$

Assume there exists a latent random vector $\mathbf{X} \in \mathbb{R}^p$ affecting $\mathbf{Y}$ through the following way:

$$\begin{cases} \text{the latent variable } \; \mathbf{X} \sim \text{N}(\mathbf{z}, I_p); \\ \text{for every component of } \mathbf{Y}: \;\; Y_k \mid \mathbf{X} \sim \text{Ber}(p_k), \quad \text{where} \log \frac{p_k}{1-p_k} = a_k + \boldsymbol{b}_k^\top \mathbf{X}; \\ \text{components are mutually independent: } \; Y_j \perp\!\!\!\perp Y_k, \quad \forall j \neq k. \end{cases}$$

Note that for one component $Y_i \in \mathbf{Y}$, there is a set of coefficients for it:

$$\text{parameters of } Y_i = \{a_i, b_{i1}, \dots, b_{ip}\}.$$

Therefore, the total number of unknown parameters of this model is $q \times (p+1)$. Though this is a parametric

model, the marginal density of $\mathbf{Y}$ could not be derived explicitly, so the MLE is not applicable. The *EM algorithm* (to be introduced) is a doable approach.

Chapter 2

Factor Analysis

2.1 From PCA to Factor Analysis

When it comes to the dimension reduction, the PCA is the most famous tool, and it provides the best linear transformation which attains the maximum explained proportion of total variance of the raw data. In the previous derivation, we know that if the goal dimension p is given, then the loading matrix is NOT unique, but up to a rotation matrix, which provides us with a way to get a better interpretation.

Example. Suppose the first 2 PC's are extracted by the raw data matrix $\mathbb{Y}_{N\times q}$, and the principal component transformation is

$$\mathbf{y}_{q\times 1} \mapsto \mathbf{x}_{2\times 1}: \quad \mathbf{x} = U^\top \mathbf{y} \quad \text{with} \quad U_{q\times 2} = \begin{pmatrix} \mathbf{u}_1 & \mathbf{u}_2 \end{pmatrix}.$$

By doing this, we are approximating $\mathbf{y}$ by

$$\mathbf{y}_{q\times 1} \approx U_{q\times 2}\mathbf{x}_{2\times 1} = x_1\mathbf{u}_1 + x_2\mathbf{u}_2.$$

Mathematically, in this case, we are using the 2-dim subspace $\text{col}(U) \subset \mathbb{R}^q$ to approximate the raw data space. Recall that we can rotate the loading matrix U and the coordinates $\mathbf{x}$ simultaneously, without losing any explained proportion of variance. This means we can choose another base of $\text{col}(U)$ with another coordinates $\mathbf{x}^*$ such that

$$x_1^*\mathbf{u}_1^* + x_2^*\mathbf{u}_2^* = x_1\mathbf{u}_1 + x_2\mathbf{u}_2,$$

to represent this 2-dim subspace $\text{col}(U)$ as illustrated in the next example.

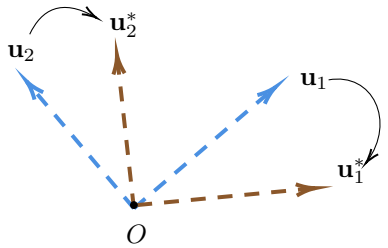


Figure 1: 2 dimension subspace: col(U) $\subset \mathbb{R}^q$

Therefore, we can choose the basis (do rotation) so that it could give some meaning such as

$$\mathbf{y}_3 = \begin{pmatrix} height \\ b.p. \\ weight \end{pmatrix} \approx x_1 \begin{pmatrix} 0.45 \\ 0.53 \\ 0.74 \end{pmatrix} + x_2 \begin{pmatrix} 0.62 \\ -0.34 \\ 0.54 \end{pmatrix}$$

$$\text{(By rotation)} \quad \Rightarrow \quad \approx x_1^* \begin{pmatrix} 0.05 \\ 0.03 \\ 0.98 \end{pmatrix} + x_2^* \begin{pmatrix} 0.02 \\ -0.94 \\ 0.14 \end{pmatrix}.$$

Using the second basis, we can see $b.p.$ is closely related to x_2^* and $weight$ is closely related to x_1^*. Though we might not know its true meaning, we can regard $\mathbf{x}^*$ as latent factors.

This goal of better interpretation is where factor analysis starts. Of course, getting factors by rotating the PC's is a good way. Another way relies on the following probabilistic model.

2.2 Exploratory Factor Analysis

We start with the basic FA model, exploratory factor analysis. Because we usually don't have any information about the data as well as the latent factors, for the simplicity in the estimation, we assume the components of the latent variable $\mathbf{X}$ (factors) are independent of each other. After using EFA to find the estimated subspace $\text{col}(U)$, we can then rotate the factors to make them match our needs like dependency or sparsity.

Assumption 5 (Exploratory factor analysis)**.** The observed data $\mathbf{Y}$ is q-dimensional. Assume there is a latent vector $\mathbf{X}$ affecting $\mathbf{Y}$, and the underlying model is

$$\mathbf{Y}_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}\mathbf{X}_{p\times 1} + W_q \quad \text{where} \quad \begin{cases} W \perp\!\!\!\perp \mathbf{X}, \\ \mathbf{X} \sim \mathbf{N}(\mathbf{z}, I_p), \\ W \sim \mathbf{N}(\mathbf{z}, \Psi_q), \end{cases}$$

where $\Psi = \text{diag}(\psi_1, \ldots, \psi_q) > 0$.

Parameters are mean vector $\boldsymbol{\mu}$, error variance Ψ_q and the loading matrix Λ. Note that this assumption contains the identifiable issue, since inserting a rotation matrix $O_{p\times p}$ and letting $\Lambda^* = \Lambda O^\top$ and $\mathbf{X}^* = O\mathbf{X}$, leads to the same model.

To interpret this model, we can regard $\mathbf{Y}$ as the scores of each aspect from the questionnaire, i.e.

$$\mathbf{Y} = \begin{pmatrix} \text{calculation score} \\ \text{algebra score} \\ \vdots \\ \text{analysis score} \end{pmatrix},$$

$\boldsymbol{\mu}$ as the common mean score among people, $\mathbf{X}$ contains some factors that have impact on the score, Λ is the loading matrix with coefficients for how factors affect the questionnaire scores, and W is the error.

Since Assumption 5 contains distribution, simple derivation leads to

$$\begin{pmatrix}\mathbf{X}\\ \mathbf{Y}\end{pmatrix} \sim \mathrm{N}\left(\begin{bmatrix}\mathbf{z}_p\\ \boldsymbol{\mu}_q\end{bmatrix}, \begin{bmatrix} I_p & \Lambda_{p\times q}^\top \\ \Lambda_{q\times p} & \Psi + \Lambda\Lambda^\top\end{bmatrix}\right) \quad \Rightarrow \quad \mathbf{Y} \sim \mathrm{N}_q(\boldsymbol{\mu}, \Psi + \Lambda\Lambda^\top).$$

how to make it identifiable? Intuitively, a model is not identifiable because it contains to many parameters. If we add some constraints on parameters, it may work (not rigorous).

Note that the identifiable issue comes from the rotation matrix O, which contains p^2 elements (free parameters). It is orthogonal, so

$$\begin{cases} O_j^\top O_j = 1 & 1 \le j \le p, \\ O_j^\top O_k = 0 & \forall j \ne k, \end{cases}$$

which gives $p + \binom{p}{2} = p(p+1)/2$ constraints. Therefore, we can add $p(p-1)/2$ more constraints, making the number of constraints equalling to then number of free parameters. One possible way is to restrict

$$\Lambda_{q\times p} = \begin{bmatrix} * & 0 & \cdots & 0 \\ * & * & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \\ * & * & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & * \end{bmatrix}_{q\times p},$$

whose restrictions are put on the right upper triangular area, and the number is $p(p-1)/2$. The fact is, by doing this way, we can make it identifiable, and it remains to estimate parameters. But we will leave the estimation to the next lecture note, and focus on other related issues in this one.

2.3 General Factor Analysis

Sometimes, we may want the factors to be NOT independent, but correlated. This leads to the generalization of the FA model.

Assumption 6 (General factor analysis)**.** The observed data $\mathbf{Y}$ is q-dimensional. Assume there is a latent vector $\mathbf{X}$ affecting $\mathbf{Y}$, and the underlying model is

$$\mathbf{Y}_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}\mathbf{X}_{p\times 1} + W_q \quad \text{where} \quad \begin{cases} W \perp\!\!\!\perp \mathbf{X}, \\ \mathbf{X} \sim \mathbf{N}(\mathbf{z}, \Phi_p), \\ W \sim \mathbf{N}(\mathbf{z}, \Psi_q), \\ Y_i \perp\!\!\!\perp Y_j \mid \mathbf{X}, \ \ \forall i \ne j, \end{cases}$$

where Φ_p is not necessarily diagonal, and $\Psi = \mathrm{diag}(\psi_1, \ldots, \psi_q) > 0$.

Despite correlation between $\mathbf{X}$, the covariance structure of observable variable $\mathbf{Y}$ is all captured by $\mathbf{X}$, since once given $\mathbf{X}$, Y_i's are independent. And in general, we can relax the distribution assumption since we

only care about the covariance structure:

$$\mathrm{Cov}(\mathbf{Y}) = \Psi + \Lambda\Phi\Lambda^\top.$$

Similar to the EFA, this model is also not identifiable, since we can insert an orthogonal matrix $O_{p\times p}$:

$$\mathbf{Y}_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}O^\top_{p\times p}O_{p\times p}\mathbf{X}_p + W_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}\mathbf{X}_p + W_q.$$

And the number of free parameters is p^2. Expect for the restriction in EFA, we can also restrict

$$\Lambda_{q\times p} = \begin{pmatrix} I_{p\times p} \\ *_{(q-p)\times p} \end{pmatrix},$$

to solve it since I_p gives p^2 constraints. (More constrains on the Λ, we will have less constraints on the Φ.) For more constraints methods, refer *Bai & Li, 2012, Annals of Statistics*.

2.4 Practical Issues

Before we move on to the estimation, we now focus on three issues:

(a) after EFA, how to rotate the factor?

(b) given data matrix $\mathbb{Y}$ and after estimation $\hat{\Lambda}$, how to get the factors $\mathbb{X}$?

(c) how to choose the number of p?

2.4.1 Factor Rotation

Generally, if we do not have convincing information, it is suggested to do EFA first, assuming factors are mutually independent. After we get the estimated $\hat{\Lambda}$ in the EFA, now it is the time to consider rotation. Normally speaking, the pattern we expect is

$$\Lambda = \begin{pmatrix} 1.10 & -1.15 & 0.14 & \cdots & 0.35 \\ 0.68 & 1.35 & 0.61 & \cdots & -0.25 \\ -0.63 & 0.98 & -0.22 & \cdots & 0.25 \\ -0.64 & -2.95 & 0.24 & \cdots & -1.28 \\ -1.37 & -0.15 & -1.30 & \cdots & -1.02 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0.41 & 0.37 & -0.99 & \cdots & 0.32 \\ 1.49 & -0.72 & 1.41 & \cdots & 0.16 \end{pmatrix} \quad \Rightarrow \quad \Lambda^* = \Lambda O^\top = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

It is obvious that we want sparsity structure in the loading matrix. Three common ways are briefly introduced:

(a) L_1 or L_2 loss function:

$$\min_{O:O^\top O=I_p} \mathrm{loss}(\hat{\Lambda}O^\top).$$

(b) varimax is another loss function.

(c) oblique rotation. (not orthogonal rotation, and it brings dependency of $\mathbf{X}$).

2.4.2 Estimation of Factors

Now we consider the EFA setting ($\mathrm{Cov}\mathbf{X} = \Phi = I_p$) and assume the loading matrix is estimated already as $\hat{\Lambda}$, the error variance is estimated as $\hat{\Psi}$. Note that the assumption is

$$\mathbf{Y} = \boldsymbol{\mu} + \Lambda\mathbf{X} + W, \quad \text{where } W \sim \mathbf{N}(0, \Psi),$$

where $\hat{\boldsymbol{\mu}}, \hat{\Lambda}, \hat{\Psi}$ are already estimated, and $\mathbb{Y}$ is observed. So, we can treat $\mathbf{X}$ as regression coefficients, $\hat{\Lambda}$ as known covariates, and $\hat{\Psi}$ as the weights to get the WLS estimate of $\hat{\mathbf{x}}$:

$$\hat{\mathbf{x}}_i = \left(\hat{\Lambda}^\top \hat{\Psi} \hat{\Lambda}\right)^{-1} \hat{\Lambda}^\top \hat{\Psi} \mathbf{y}_i.$$

We can even further use the Bayesian idea, treat $\mathbf{X} \sim \mathbf{N}(0, I_p)$ as the prior, and $\hat{\mathbf{x}}_i$ as the data. By Ridge regression, it follows that

$$\hat{\mathbf{x}}_i = \left(I + \hat{\Lambda}^\top \hat{\Psi} \hat{\Lambda}\right)^{-1} \hat{\Lambda}^\top \hat{\Psi} \mathbf{y}_i.$$

2.4.3 Selection of p

This is a serious issue, and it could be applied not only in FA, but also PCA. Simple methods are known as

(a) scree plot;

(b) explained proportion of total variance.

But none of them are supported by statistical theory. Now we propose two theoretical methods.

Parallel Analysis

Horn's parallel analysis (1965) uses the idea of permutation test. Suppose the observed data matrix is

$$\mathbb{Y}_{N\times q} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1q} \\ y_{21} & y_{22} & \cdots & y_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nq} \end{bmatrix} = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_N^\top \end{pmatrix},$$

and the sample covariance matrix of $\mathbb{Y}$ is S with eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_q$. Under EFA assumptions,

$$S \xrightarrow{a.s.} \Psi + \Lambda\Lambda^\top.$$

If we assume the covariance structure is captured by the p factors, then $\Lambda\Lambda^\top$ part should accounts for a large proportion in the diagonal of S, and the noise variance, Ψ accounts for a small proportion, i.e. the first p terms in the diagonal of

$$\Psi + \Lambda\Lambda^\top = \begin{pmatrix} \lambda_1 & & & & & & \\ & \ddots & & & & & \\ & & \lambda_p & & & & \\ & & & \lambda_{p+1} & & & \\ & & & & \ddots & & \\ & & & & & \lambda_q & \end{pmatrix}$$

should be large. Therefore, we can use the following algorithm to test

$$H_0 : \text{there is no signal} \quad \text{vs} \quad H_1 : \text{there is at least one signal.}$$

But note that this is not a strict hypothesis testing as we shall see. Under H_0, there is no significant factor, so λ's, the eigenvalues of S, come from the noise Ψ, but not $\Lambda\Lambda^\top$. Now, we utilize this idea to regenerate data.

Algorithm 4 (Parallel analysis)**.** Set a threshold $0 < \alpha < 1$ for when to stop. Do random permutation for each column of $\mathbb{Y}$:

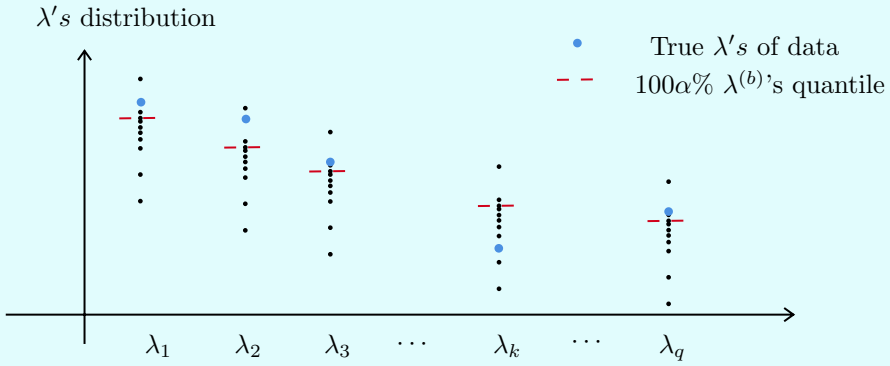
$$\mathbb{Y}_{N\times q} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1q} \\ y_{21} & y_{22} & \cdots & y_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nq} \end{bmatrix}.$$

(a) After permuting all columns, we get $\mathbb{Y}^{(1)}$. Calculate the eigenvalues $\lambda_1^{(1)} > \cdots > \lambda_q^{(1)}$ of $S^{(1)}$.

(b) Repeat B times of step (a).

(c) Calculate the distribution and quantile of each re-sampled λ's.

(d) Choose the $k \in \{1, 2, \ldots, q\}$ when the true λ_k falls below the threshold in the re-sampled distribution.



Why does this algorithm work? When we permute the data matrix $\mathbb{Y}$ by column, the sample variance matrix S does not change, so the covariance structure of $\mathbf{Y}$'s components is preserved. However, since every observation is mutually independent, i.e. $\mathbf{y}_i \perp\!\!\!\perp \mathbf{y}_j$ and $\mathbf{x}_i \perp\!\!\!\perp \mathbf{x}_j$ for $i \neq j$, when we switch the order inside the column, the covariance structure of $\mathbf{X}$ gets killed. Therefore, the re-sampled data could be regarded from the structure

$$\mathrm{Cov}(\mathbf{y}_i^{(b)}) = \Psi.$$

And under H_0 : there is no signal, the true λ should not be significantly greater than the re-sampled $\lambda's$, otherwise, we have the confidence to claim there is a signal. For more details, refer *Edgar Dobriban, 2020, Annals of Statistics.*

Hypothesis Testing

The last method relies on the probabilistic assumption. It could also be applied to probabilistic PCA. In essence, we are doing sequential hypothesis testing for

$$H_0 : \Sigma \text{ is diagonal} \quad \text{v.s.} \quad H_1 : \Sigma \text{ is not diagonal.}$$

For simplicity, we consider the setting of probabilistic PCA.

Assumption 7. For every observation, the raw data $\mathbf{Y} \in \mathbb{R}^q$ is q-dimensional, and is generated by

$$\mathbf{Y}_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}\mathbf{X}_p + W_q \quad \text{where} \quad \begin{cases} \mathbf{X} \sim \mathrm{N}_p(\mathbf{z}, I_p), \\ W \sim \mathrm{N}_q(\mathbf{z}, \Psi_q), \\ W \perp\!\!\!\perp \mathbf{X} \text{ for every obs}, \\ \text{observations are independent.} \end{cases}$$

Equivalently, we are assuming the population

$$\mathbf{Y} \sim \mathrm{N}_q\left(\boldsymbol{\mu}, \Psi + \Lambda\Lambda^\top\right).$$

For the hypothesis

$$H_0 : \Sigma \text{ is diagonal} \quad \iff \quad p = 0 \quad \iff \quad \mathrm{Var}\mathbf{Y} = \Psi,$$

the likelihood ratio test is

$$\mathrm{LRT} = \frac{\sup_{\boldsymbol{\mu},\Psi} L(\boldsymbol{\mu}, \Psi \mid \mathbb{Y})}{\sup_{\boldsymbol{\mu},\Sigma} L(\boldsymbol{\mu}, \Sigma \mid \mathbb{Y})}, \quad \text{where } \Psi \text{ is diagonal and } \Sigma \text{ is symmetric.}$$

By the large sample theory, as N increase and p is any fixed number

$$2\log(\mathrm{LRT}) \xrightarrow{d} \chi^2\left(\frac{q(q+1)}{2} - \left(q(p+1) - \frac{p(p-1)}{2}\right)\right).$$

Setting $p = 0$, if H_0 is rejected, then we can set $p = 1, \ldots$ and so on. Sequentially, we will stop at some point k. But since this is a sequential hypothesis testing, the total type II error is the sum of k-th type II error and type I error of all previous hypothesis testings.

Chapter 3

Expectation Maximization

In Chapter 2, we discussed sparse PCA, probabilistic PCA and factor analysis. Direct MLE of the parameters in them are sometimes not easy. In this note, we introduce the EM algorithm, which is a powerful tool for latent variable model, to help.

3.1 Latent Variable Models

Many probabilistic models such as linear model, time series, and generalized linear model, do not contain latent variables. They have parameters, and once we get the observed data, the MLE could be done (may not be explicitly solved like GLM). However, these simple models do not satisfy our needs. Sometimes, we know the observed data are affected by some un-observable variables with unknown parameters. This kind of models are called *latent variable models*.

Generally, without considering the hierarchical structure, the model is specified in the following way.

Assumption 8 (Latent variable models)**.** Suppose there exists a latent random vector $\mathbf{X} \in \mathbb{R}^p$ follows some distribution with parameters $\boldsymbol{\theta}_x$, write

$$\mathbf{X} \sim p(\mathbf{x} \mid \boldsymbol{\theta}_x).$$

And the observable data $\mathbf{Y} \in \mathbb{R}^q$ is from the conditional distribution

$$\mathbf{Y} \mid \mathbf{X} = \mathbf{x} \sim p(\mathbf{y} \mid \mathbf{x}, \boldsymbol{\theta}_y).$$

For every observation $\mathbf{y}_i$, the parameters $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$ are the same. But due to the variation of $\mathbf{x}_i$, the distribution of $\mathbf{y}_i$ may vary.

We can simply understand as the fact that the latent variable $\mathbf{x}$ generates $\mathbf{y}$, and our goal is to estimate $\boldsymbol{\theta}_x$ and $\boldsymbol{\theta}_y$. Both $\mathbf{X}$ and $\mathbf{Y}$ could be discrete or continuous. This depends on the scenario. A few models in the previous lecture notes are latent variable models, such as

(a) factor analysis model

$$\mathbf{X} \sim \mathbf{N}(\mathbf{z}, \Phi) \quad \text{and} \quad \mathbf{Y} \mid \mathbf{X} = \mathbf{x} \sim \mathbf{N}\left(\boldsymbol{\mu}, \Psi + \Lambda\Phi\Lambda^\top\right),$$

where parameters are $\boldsymbol{\theta}_x = \{\Phi\}$ and $\boldsymbol{\theta}_y = \{\boldsymbol{\mu}, \Psi, \Lambda\}$.

(b) probabilistic PCA and PCA for binary data.

(c) Gaussian mixture model and other clustering models (to be introduced).

3.2 EM Algorithm

3.2.1 Optimization Goal

Under Assumption 1, suppose we observed a random sample

$$\mathbb{Y}_{N\times q} = \begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1q} \\ y_{21} & y_{22} & \cdots & y_{2q} \\ \vdots & \vdots & \ddots & \vdots \\ y_{N1} & y_{N2} & \cdots & y_{Nq} \end{bmatrix} = \begin{pmatrix} \mathbf{y}_1^\top \\ \mathbf{y}_2^\top \\ \vdots \\ \mathbf{y}_N^\top \end{pmatrix},$$

but the latent realizations $\mathbf{x}_1, \ldots, \mathbf{x}_N$ are not known. For one observation i, we have

$$\text{joint density of } \mathbf{X} \text{ and } \mathbf{Y}: \ p(\mathbf{x}_i, \mathbf{y}_i \mid \boldsymbol{\theta}) \quad \Rightarrow \quad \text{marginal density of } \mathbf{Y}: \ \int_{\mathbf{x}_i} p(\mathbf{x}_i, \mathbf{y}_i \mid \boldsymbol{\theta}) d\mathbf{x}_i.$$

By the conditional distribution assumption of $\mathbf{Y}$ given $\mathbf{X} = \mathbf{x}$, it could be further decompose as

$$\text{marginal density of } \mathbf{Y}: \ \int_{\mathbf{x}_i} p(\mathbf{x}_i \mid \boldsymbol{\theta}_x) p(\mathbf{y}_i \mid \mathbf{x}_i, \boldsymbol{\theta}_y) d\mathbf{x}_i.$$

Then we can write the *marginal likelihood* of the sample $\mathbb{Y}$, which is also called the *likelihood for observed data*:

$$L(\boldsymbol{\theta}; \mathbb{Y}) = \prod_{i=1}^{N} p(\mathbf{y}_i \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \int_{\mathbf{x}_i} p(\mathbf{x}_i \mid \boldsymbol{\theta}_x) p(\mathbf{y}_i \mid \mathbf{x}_i, \boldsymbol{\theta}_y) d\mathbf{x}_i,$$

and the *marginal log-likelihood*:

$$l(\boldsymbol{\theta}; \mathbb{Y}) = \sum_{i=1}^{N} \log p(\mathbf{y}_i \mid \boldsymbol{\theta}) = \sum_{i=1}^{N} \left\{ \log \int_{\mathbf{x}_i} p(\mathbf{x}_i \mid \boldsymbol{\theta}_x) p(\mathbf{y}_i \mid \mathbf{x}_i, \boldsymbol{\theta}_y) d\mathbf{x}_i \right\}, \tag{3.1}$$

which is the goal of maximization with respect to $\boldsymbol{\theta}$. If the model does not contain latent variable, then we don't have an integration, leading to a simple optimization. However, without an explicit expression of this integration, the optimization is challenging most of the time. And this is the usual case we will deal with, because we can arbitrarily choose the distribution of $\mathbf{X}$ and $\mathbf{Y} \mid \mathbf{X} = \mathbf{x}$.

Remark. Recall the models we have introduced. Probabilistic PCA and factor analysis have an explicit expression for the integration since the distribution assumption is normal. However, sometimes EM algorithm is even faster than the direct optimization when the explicit expression exists. And for the binary PCA, we don't have an explicit expression.

Since observations $\mathbf{y}_i$'s are independent of each other, we will start our derivation using marginal log-likelihood of the whole sample

$$l(\boldsymbol{\theta} \mid \mathbb{Y}) = \log p(\mathbb{Y} \mid \boldsymbol{\theta})$$

instead of a single observation.

3.2.2 EM Procedures

For simplicity, we first suppose we can have the explicit form of the joint distribution $p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})$ and the conditional distribution $p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})$. To summary the EM algorithm in one sentence, it is an iterative algorithm with

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \int_{\mathbb{X}} \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \cdot p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}) d\mathbb{X},$$

or equivalently,

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbb{X}|\mathbb{Y}, \boldsymbol{\theta}^{(t)}} \Big[\log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big]. \tag{3.2}$$

For a formal expression, the EM algorithm is as follows.

Algorithm 5 (EM algorithm)**.** Set an initial value for $\hat{\boldsymbol{\theta}}^{(0)}$.

- Expectation step: Set distribution q_t for $\mathbb{X}$ of this step to be the posterior of $\mathbb{X}$ given $\mathbb{Y}$

  $$q_t(\mathbb{X}) = p(\mathbb{X} \mid \mathbb{Y}, \hat{\boldsymbol{\theta}}^{(t)}),$$

  and calculate the expectation expression with the unknown parameter $\boldsymbol{\theta}$:

  $$\mathbb{E}_{q_t} \Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big\}.$$

- Maximization step: Update $\hat{\boldsymbol{\theta}}^{(t+1)}$ by the value that maximizes the expectation above:

  $$\hat{\boldsymbol{\theta}}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \ \mathbb{E}_{q_t} \Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big\}.$$

Repeat EM steps until

$$\Big| \hat{\boldsymbol{\theta}}^{(t+1)} - \hat{\boldsymbol{\theta}}^{(t)} \Big| \quad \text{or} \quad \Big| \max \mathbb{E}_{q_{t+1}} \Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big\} - \max \mathbb{E}_{q_t} \Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big\} \Big|$$

is smaller than a previously set threshold.

We leave the question, why this algorithm works, for now and focus on some remarks about this algorithm.

Remark. There are few topics worth discussion.

(a) Even the data $\mathbb{Y}$ is given and the parameter $\hat{\boldsymbol{\theta}}^{(0)}$ is set, the posterior distribution $q_0(\mathbb{X}) = p(\mathbb{X} \mid \mathbb{Y}, \hat{\boldsymbol{\theta}}^{(0)})$ is not guaranteed to get a closed form or easy to calculate.

(b) Perfect goal is q_0 has a closed form and the expectation

$$\mathbb{E}_{q_0} \Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big\}$$

has a closed expression as a function of $\boldsymbol{\theta}$. However, with the difficulty in (a), we may not get a closed form in (b).

For example, the binary PCA model has both difficulties (a) and (b) mentioned in the last remark. At the source, the problem comes from the unknown distribution of $\mathbb{X} \mid \mathbb{Y} = \mathbf{y}$. If this is the case, how do we

implement the EM algorithm? The answer is by simulation such as Monte Carlo. Since

$$q_0(\mathbb{X}) = p(\mathbb{X} \mid \mathbb{Y}, \hat{\boldsymbol{\theta}}^{(0)}) = \frac{p(\mathbb{X}, \mathbb{Y} \mid \hat{\boldsymbol{\theta}}^{(0)})}{p(\mathbb{Y} \mid \hat{\boldsymbol{\theta}}^{(0)})} = \frac{p(\mathbb{Y} \mid \mathbb{X}, \hat{\boldsymbol{\theta}}^{(0)}) \times p(\mathbb{X} \mid \hat{\boldsymbol{\theta}}^{(0)})}{p(\mathbb{Y} \mid \hat{\boldsymbol{\theta}}^{(0)})}$$

is a known function, though its expression is so complicated that we can not know the distribution of q_0, we can use simulation method to regenerate samples from the known distribution function q_0. Then we try different $\boldsymbol{\theta}$ to estimate (below is a function of $\boldsymbol{\theta}$)

$$\mathbb{E}_{q_0}\Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big\} = \frac{1}{B} \sum_{i=1}^{B} \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}),$$

and use the $\boldsymbol{\theta}$ that maximizes this expectation and proceed the EM algorithm.

3.2.3 The Ascent Likelihood of EM

In this subsection, we focus on why the EM algorithm works. Specifically speaking, what guarantees it to converge.

Proposition 4. *Under Assumption 1 and Algorithm 2, it follows that*

$$\log p(\mathbb{Y} \mid \boldsymbol{\theta}^{(t)}) \leq \log p(\mathbb{Y} \mid \boldsymbol{\theta}^{(t+1)}),$$

i.e., with iterations, the marginal log-likelihood is ascending.

Proof. First, let's forget about t, and the following identity for all distributions:

$$\log p(\mathbb{Y} \mid \boldsymbol{\theta}) = \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) - \log p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}).$$

Now, consider we are at time t, and the distribution $p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)$ is known. Integrating the above equation on both sides w.r.t. this distribution, we get

$$\begin{aligned}
\text{LHS} &= \int_{\mathbb{X}} \log p(\mathbb{Y} \mid \boldsymbol{\theta}) \cdot p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} \\
&= \log p(\mathbb{Y} \mid \boldsymbol{\theta}) \int_{\mathbb{X}} p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} \\
&= \log p(\mathbb{Y} \mid \boldsymbol{\theta}),
\end{aligned}$$

which means the LHS does not change after integration. And we also have

$$\begin{aligned}
\text{RHS} &= \int_{\mathbb{X}} \Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) - \log p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}) \Big\} \cdot p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} \\
&= \int_{\mathbb{X}} \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \cdot p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} - \int_{\mathbb{X}} \log p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}) \cdot p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} \\
&= \mathbb{E}_{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)} \Big[\log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big] + \mathbb{E}_{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)} \Big[\log p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}) \Big].
\end{aligned}$$

And LHS=RHS means

$$\log p(\mathbb{Y} \mid \boldsymbol{\theta}) = \mathbb{E}_{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)} \Big[\log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big] + \mathbb{E}_{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)} \Big[\log p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}) \Big].$$

For simplicity, we use the notation:

$$Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}\right) = \mathbb{E}_{p\left(\mathbb{X}|\mathbb{Y},\boldsymbol{\theta}^{(t)}\right)}\Big[\log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})\Big] \quad \text{and} \quad H\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}\right) = \mathbb{E}_{p\left(\mathbb{X}|\mathbb{Y},\boldsymbol{\theta}^{(t)}\right)}\Big[\log p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})\Big].$$

Then the equality could be expressed as

$$\log p(\mathbb{Y} \mid \boldsymbol{\theta}) = Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}\right) + H\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}\right) \tag{3.3}$$

Here comes the critical point of EM algorithm. In the procedure (3.2) of EM algorithm, we set

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbb{X}|\mathbb{Y},\boldsymbol{\theta}^{(t)}}\Big[\log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})\Big].$$

Therefore, we have the inequality

$$Q\left(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}\right) \geq Q\left(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}\right), \quad \forall \boldsymbol{\theta} \in \boldsymbol{\Theta}.$$

In particular, it means

$$Q\left(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}\right) \geq Q\left(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)}\right). \tag{3.4}$$

Besides, we can have an inequality for H as well:

$$\begin{aligned}
&H\left(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}\right) - H\left(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)}\right) \\
&\quad= \mathbb{E}_{p\left(\mathbb{X}|\mathbb{Y},\boldsymbol{\theta}^{(t)}\right)}\Big[\log p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t+1)}\right)\Big] - \mathbb{E}_{p\left(\mathbb{X}|\mathbb{Y},\boldsymbol{\theta}^{(t)}\right)}\Big[\log p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)\Big] \\
&\quad= \int_{\mathbb{X}} p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) \cdot \log p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t+1)}\right) d\mathbb{X} - \int_{\mathbb{X}} p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) \cdot \log p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} \\
&\quad= \int_{\mathbb{X}} p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) \cdot \log \frac{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t+1)}\right)}{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)} d\mathbb{X} \\
&\quad= -\int_{\mathbb{X}} p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) \cdot \log \frac{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)}{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t+1)}\right)} d\mathbb{X} \\
&\quad= -\mathrm{KL}\Big\{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) \parallel p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t+1)}\right)\Big\} \\
&\quad\leq 0 \qquad \text{(KL divergence's property, in the next page)}
\end{aligned} \tag{3.5}$$

By (3.3), (3.4) and (3.5), we conclude that

$$\begin{aligned}
\log p\left(\mathbb{Y} \mid \boldsymbol{\theta}^{(t+1)}\right) &= Q\left(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}\right) + H\left(\boldsymbol{\theta}^{(t+1)}, \boldsymbol{\theta}^{(t)}\right) \\
&\geq Q\left(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)}\right) + H\left(\boldsymbol{\theta}^{(t)}, \boldsymbol{\theta}^{(t)}\right) = \log p\left(\mathbb{Y} \mid \boldsymbol{\theta}^{(t)}\right).
\end{aligned}$$

□

3.2.4 The Intuition behind EM

We already know the mechanism of EM algorithm, including how to use it as well as how it works. In this subsection, we will explore how the creators of EM algorithm come up with it. More specifically, how do they know the choice of

$$q_t(\mathbb{X}) = p(\mathbb{X} \mid \mathbb{Y}, \hat{\boldsymbol{\theta}}^{(t)}),$$

makes it work.

As usual, let's first forget about t, and we assume $q(\mathbb{X})$ is any given density. Then it follows that

$$\begin{aligned}
\log p(\mathbb{Y} \mid \boldsymbol{\theta}) &= \int_{\mathbb{X}} \log p(\mathbb{Y} \mid \boldsymbol{\theta}) \times q(\mathbb{X}) d\mathbb{X} \\
\Big(\because p(x,y) = p(y)p(x \mid y) \Big) \quad &= \int_{\mathbb{X}} \log \left\{ \frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})} \cdot \frac{q(\mathbb{X})}{q(\mathbb{X})} \right\} \times q(\mathbb{X}) d\mathbb{X} \\
&= \int_{\mathbb{X}} \left\{ \log \frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{q(\mathbb{X})} + \log \frac{q(\mathbb{X})}{p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})} \right\} \times q(\mathbb{X}) d\mathbb{X} \\
&= \int_{\mathbb{X}} \log \left[\frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{q(\mathbb{X})} \right] \times q(\mathbb{X}) d\mathbb{X} + \int_{\mathbb{X}} \log \left[\frac{q(\mathbb{X})}{p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})} \right] \times q(\mathbb{X}) d\mathbb{X}.
\end{aligned}$$

Now we will consider the two terms one by one. For simplicity, we write

$$\text{(I)} = \int_{\mathbb{X}} \log \left[\frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{q(\mathbb{X})} \right] \times q(\mathbb{X}) d\mathbb{X} \quad \text{and} \quad \text{(II)} = \int_{\mathbb{X}} \log \left[\frac{q(\mathbb{X})}{p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})} \right] \times q(\mathbb{X}) d\mathbb{X}.$$

Term (II)

Let's consider term (II) first.

Definition 3 (Kullback-Leibler divergence)**.** Suppose Q and P are two probability measures with density q and p, then the KL divergence from Q to P is

$$\mathrm{KL}(q \| p) = \int_x \log \frac{q(x)}{p(x)} \times q(x) dx = \int_x \log \frac{q(x)}{p(x)} dQ = \mathbb{E}_Q \left[\log \frac{q(X)}{p(X)} \right].$$

Jensen's inequality says

$$\text{when } \phi \text{ is convex: } \varphi(\mathbb{E}X) \leq \mathbb{E}\varphi(X) \quad \text{and} \quad \text{when } \phi \text{ is concave: } \varphi(\mathbb{E}X) \geq \mathbb{E}\varphi(X).$$

Therefore, we can apply it here and get

$$\mathrm{KL}(q \| p) = -\mathbb{E}_Q \left[\log \frac{p(X)}{q(X)} \right] \geq -\log \left(\mathbb{E}_Q \left[\frac{p(X)}{q(X)} \right] \right) = -\log \int_x \frac{p(x)}{q(x)} \times q(x) dx = -\log \int_x p(x) dx = 0.$$

Therefore, KL divergence of two p.m.s are always non-negative, and they are equal if and only if $\phi(x)$ is linear or $p(X)/q(X)$ is constant a.s. Since log is not linear, we have the equality only when $p(X) = q(X)$ a.s., which means the distributions $P = Q$ weakly.

Now, let's focus on term (II). Using the definition of KL divergence, we can see

$$\text{(II)} = \int_{\mathbb{X}} \log \left[\frac{q(\mathbb{X})}{p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})} \right] \times q(\mathbb{X}) d\mathbb{X} = \mathrm{KL}\Big(q(\cdot) \parallel p(\cdot \mid \mathbb{Y}, \boldsymbol{\theta}) \Big) \geq 0.$$

Therefore, term (I) is always a lower bound for $l(\boldsymbol{\theta} \mid \mathbb{Y})$, i.e.

$$\log p(\mathbb{Y} \mid \boldsymbol{\theta}) = \text{(I)} + \text{(II)} \geq \text{(I)}.$$

Term (I)

In fact, term (I) has a formal name, *evidence lower bound* (ELBO), in variational analysis. Suppose we are at the time t, and the conditional distribution $p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)$ and joint distribution $p\left(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}^{(t)}\right)$ are known. From the former derivation, we know for any distribution $q(\mathbb{X})$:

$$\log p(\mathbb{Y} \mid \boldsymbol{\theta}) = \int_{\mathbb{X}} \log \left[\frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{q(\mathbb{X})}\right] \times q(\mathbb{X}) d\mathbb{X} + \int_{\mathbb{X}} \log \left[\frac{q(\mathbb{X})}{p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})}\right] \times q(\mathbb{X}) d\mathbb{X}.$$

and the equality holds if and only if, in the KL divergence, we take

$$q(\mathbb{X}) = p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}).$$

Since we don't know the true distribution of $p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})$, we can only use the one from the knowledge in the previous step $p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)$ to replace the $q(\mathbb{X})$. Then the equality becomes

$$\log p(\mathbb{Y} \mid \boldsymbol{\theta}) = \int_{\mathbb{X}} \log \left[\frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)}\right] \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} + \int_{\mathbb{X}} \log \left[\frac{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)}{p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})}\right] \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X}.$$

If we can create an algorithm that guarantees $\boldsymbol{\theta}^{(t)} \to \boldsymbol{\theta}$, the second term goes to zero as we iterate. Therefore, the optimization could be restricted on the first term, i.e., set

$$\boldsymbol{\theta}^{(t+1)} = \arg\max_{\boldsymbol{\theta}} \int_{\mathbb{X}} \log \left[\frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)}\right] \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X}.$$

If this guarantees $\boldsymbol{\theta}^{(t)} \to \boldsymbol{\theta}$, then the algorithm is completed. Let's expand it to see.

$$\begin{aligned}
\boldsymbol{\theta}^{(t+1)} &= \arg\max_{\boldsymbol{\theta}} \int_{\mathbb{X}} \log \left[\frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)}\right] \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} \\
&= \arg\max_{\boldsymbol{\theta}} \left\{ \int_{\mathbb{X}} \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} - \int_{\mathbb{X}} \log p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X} \right\}
\end{aligned}$$

Note that the second term has nothing to do with $\boldsymbol{\theta}$,

$$= \arg\max_{\boldsymbol{\theta}} \int_{\mathbb{X}} \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X}.$$

This expression shows what we are doing is exactly the same as we did in the EM algorithm. And the previous page have shown $l\left(\boldsymbol{\theta}^{(t+1)}\right) \geq l\left(\boldsymbol{\theta}^{(t)}\right)$. So, we are done.

Explanation of EM

Note that we transformed the MLE from:

$$\begin{aligned}
\max_{\boldsymbol{\theta}} \Big\{ \log p(\mathbb{Y} \mid \boldsymbol{\theta}) \Big\} &= \max_{\boldsymbol{\theta}} \left\{ \log \int_{\mathbb{X}} p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) d\mathbb{X} \right\} \\
\Rightarrow \quad \max_{\boldsymbol{\theta}} \mathbb{E}_Q \Big[\log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big] &= \max_{\boldsymbol{\theta}} \left\{ \int_{\mathbb{X}} \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) Q(d\mathbb{X}) \right\}.
\end{aligned}$$

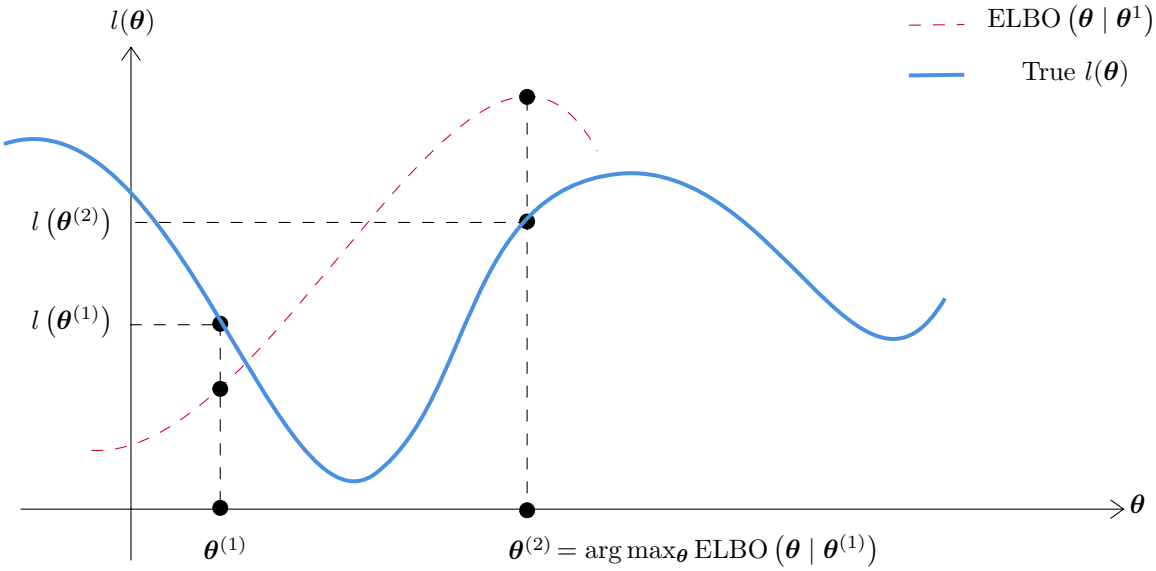
Basically, we switch the order of integration and log, which is critical for easy calculation. But the price is introducing another distribution q for iteration.

To make it more vivid, we can understand the E-step and M-step as follows.

- E-step: we fix the known parameter information in $\boldsymbol{\theta}^{(t)}$, then ELBO becomes a function of $\boldsymbol{\theta}$:

$$\text{ELBO}(\boldsymbol{\theta}) = \int_{\mathbb{X}} \log \left[\frac{p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta})}{p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right)} \right] \times p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) d\mathbb{X}.$$

- M-step: we maximize the ELBO($\boldsymbol{\theta}$) w.r.t. $\boldsymbol{\theta}$, and this gives us more information about $\boldsymbol{\theta}$ so that we can do the next iteration.



However, the only guarantee is $l\left(\boldsymbol{\theta}^{(t+1)}\right) \geq l\left(\boldsymbol{\theta}^{(t)}\right)$, which means we may get to a local maximum instead of the global one. So, the common way is to start from several initial points, and find the possible global one.

Alternatives to distribution $q(\mathbb{X})$

In the above algorithm, the distribution $q(\mathbb{X})$ is chosen to be the posterior of $\mathbb{X}$ given $\mathbb{Y}$ and $\hat{\boldsymbol{\theta}}^{(t)}$, i.e.

$$q_0(\mathbb{X}) = p(\mathbb{X} \mid \mathbb{Y}, \hat{\boldsymbol{\theta}}^{(0)}),$$

and this provides us with the convenience to ignore the term (II) in the maximization since it vanishes by the convergence $q_n(\mathbb{X}) \to p(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta})$. However, using this q may result in difficulty in deriving the closed expression as discussed in the last remark (a). For the convenience to derive the closed form of q, we can choose

$$q(\mathbf{x}_i) \overset{\text{ind}}{\sim} \mathbf{N}(\boldsymbol{\mu}_i, \Sigma_i),$$

and this allows us to derive many closed form. However, using this q, the term (II) may not vanish in the iteration. This idea of replacing q is used in variational inference.

3.3 Case Study: EM for Factor Analysis

Theoretically, the derivation of the EM algorithm seems abstract. Let's use it for exploratory factor analysis estimation, and we will see its power. The model assumption is

Assumption 9 (Exploratory factor analysis)**.** The observed data $\mathbf{Y}$ is q-dimensional. Assume there is a latent vector $\mathbf{X}$ affecting $\mathbf{Y}$, and the underlying model is

$$\mathbf{Y}_q = \boldsymbol{\mu}_q + \Lambda_{q\times p}\mathbf{X}_{p\times 1} + W_q \quad \text{where} \quad \begin{cases} W \perp\!\!\!\perp \mathbf{X}, \\ \mathbf{X} \sim \mathbf{N}(\mathbf{z}, I_p), \\ W \sim \mathbf{N}(\mathbf{z}, \Psi_q), \end{cases}$$

where $\Psi = \operatorname{diag}(\psi_1, \ldots, \psi_q) > 0$.

For simplicity in deriving the conditional probability, we can write

$$\begin{pmatrix} \mathbf{X}_i \\ \mathbf{Y}_i \end{pmatrix} \overset{\text{iid}}{\sim} \mathbf{N}\left(\begin{bmatrix} \mathbf{z} \\ \boldsymbol{\mu} \end{bmatrix}, \begin{bmatrix} I_{p\times p} & \Lambda^\top_{p\times q} \\ \Lambda_{q\times p} & \Psi + \Lambda\Lambda^\top \end{bmatrix} \right).$$

Since the marginal likelihood has an explicit form, we can directly solve its MLE. But for now, let's use EM for a practice. By the conditional property of multivariate normal (Lecture 1 Section 2), it follows that

$$\mathbf{X}_i \mid \mathbf{Y}_i = \mathbf{y}_i \sim \mathbf{N}\Big(\Lambda^\top \left(\Lambda\Lambda^\top + \Psi\right)^{-1} (\mathbf{y}_i - \boldsymbol{\mu}), \;\; I_p - \Lambda^\top \left(\Psi + \Lambda\Lambda^\top\right)^{-1} \Lambda \Big).$$

If the inversion of $\left(\Lambda\Lambda^\top + \Psi\right)$ is difficult, probably when q is large, we can use the following alternatives

$$\begin{cases} \mathbb{E}(\mathbf{X}_i \mid \mathbf{Y}_i = \mathbf{y}_i) = \left(I_p + \Lambda^\top \Psi^{-1} \Lambda\right)^{-1} \Lambda^\top \Psi^{-1} (\mathbf{y}_i - \boldsymbol{\mu}), \\ \operatorname{Var}(\mathbf{X}_i \mid \mathbf{Y}_i = \mathbf{y}_i) = \left(I_p + \Lambda^\top \Psi \Lambda\right)^{-1}. \end{cases}$$

Note this is equality, not an approximation. (One can show it by exercise) And it makes the q-by-q inverse to be p-by-p. With this conditional distribution, we can return to the EM algorithm. Since the MLE for normal mean would be the sample mean, we can assume $\mu = 0$, otherwise we center this data. So, we don't need to deal with the estimation of μ in the EM algorithm.

3.3.1 E-Step of Factor Analysis

Recall that in this step, we are setting

$$q_t(\mathbb{X}) = p\left(\mathbb{X} \mid \mathbb{Y}, \boldsymbol{\theta}^{(t)}\right) \quad \text{and calculating} \quad \mathbb{E}_{q_t}\Big\{ \log p(\mathbb{X}, \mathbb{Y} \mid \boldsymbol{\theta}) \Big\}.$$

Since every observation is independent of each other, we can expand the sample density to be

$$\begin{aligned}
\mathbb{E}_{q_t}\Big\{\log p(\mathbb{X},\mathbb{Y}\mid\boldsymbol{\theta})\Big\} &= \mathbb{E}_{q_t}\left\{\sum_{i=1}^{N}\log p(\mathbf{x}_i,\mathbf{y}_i\mid\boldsymbol{\theta})\right\}\\
&= \mathbb{E}_{q_t}\left\{\sum_{i=1}^{N}\Big[\log p(\mathbf{y}_i\mid\mathbf{x}_i,\boldsymbol{\theta})+\log p(\mathbf{x}_i\mid\boldsymbol{\theta})\Big]\right\}\\
(\because \mathbf{X}_i\sim\mathbf{N}(\mathbf{z},I_p) \text{ is not related to } \boldsymbol{\theta}) \quad &= \mathbb{E}_{q_t}\left\{\sum_{i=1}^{N}\log p(\mathbf{x}_i)+\sum_{i=1}^{N}\log p(\mathbf{y}_i\mid\mathbf{x}_i,\boldsymbol{\theta})\right\}. \qquad (3.6)
\end{aligned}$$

Using the assumption $\mathbf{Y}_i\mid\mathbf{X}_i=\mathbf{x}_i\sim\mathbf{N}(\Lambda\mathbf{x}_i,\Psi)$ and the log density of multivariate normal, we have

$$\log p(\mathbf{y}_i\mid\mathbf{x}_i,\boldsymbol{\theta}) = -\frac{q}{2}\log(2\pi)-\frac{1}{2}\log|\Psi|-\frac{1}{2}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)^\top\Psi^{-1}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right).$$

Plugging it in (3.6), it continues to be

$$\begin{aligned}
&\mathbb{E}_{q_t}\Big\{\log p(\mathbb{X},\mathbb{Y}\mid\boldsymbol{\theta})\Big\}\\
&= \mathbb{E}_{q_t}\left\{\sum_{i=1}^{N}\log p(\mathbf{x}_i)-\frac{Nq}{2}\log(2\pi)-\frac{N}{2}\log|\Psi|-\frac{1}{2}\sum_{i=1}^{N}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)^\top\Psi^{-1}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)\right\}\\
&= \sum_{i=1}^{N}\mathbb{E}_{q_t}\log p(\mathbf{x}_i)-\frac{Nq}{2}\log(2\pi)-\frac{N}{2}\log|\Psi|-\frac{1}{2}\sum_{i=1}^{N}\mathbb{E}_{q_t}\Big\{\mathrm{tr}\Big[\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)^\top\Psi^{-1}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)\Big]\Big\}\\
&= \sum_{i=1}^{N}\mathbb{E}_{q_t}\log p(\mathbf{x}_i)-\frac{Nq}{2}\log(2\pi)-\frac{N}{2}\log|\Psi|-\frac{1}{2}\sum_{i=1}^{N}\mathbb{E}_{q_t}\Big\{\mathrm{tr}\Big[\Psi^{-1}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)^\top\Big]\Big\}\\
&= \sum_{i=1}^{N}\mathbb{E}_{q_t}\log p(\mathbf{x}_i)-\frac{Nq}{2}\log(2\pi)-\frac{N}{2}\log|\Psi|-\frac{1}{2}\mathbb{E}_{q_t}\Big\{\mathrm{tr}\Big[\Psi^{-1}\sum_{i=1}^{N}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)^\top\Big]\Big\}.
\end{aligned}$$

For simplicity, we use the notation

$$S=\frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)\left(\mathbf{y}_i-\Lambda\mathbf{x}_i\right)^\top.$$

And plug it back to the main equation, it follows that

$$\begin{aligned}
\mathbb{E}_{q_t}\Big\{\log p(\mathbb{X},\mathbb{Y}\mid\boldsymbol{\theta})\Big\} &= \sum_{i=1}^{N}\mathbb{E}_{q_t}\log p(\mathbf{x}_i)-\frac{Nq}{2}\log(2\pi)-\frac{N}{2}\log|\Psi|-\frac{N}{2}\mathbb{E}_{q_t}\Big\{\mathrm{tr}\Big[\Psi^{-1}S\Big]\Big\}\\
&= \sum_{i=1}^{N}\mathbb{E}_{q_t}\log p(\mathbf{x}_i)-\frac{Nq}{2}\log(2\pi)-\frac{N}{2}\log|\Psi|-\frac{N}{2}\mathrm{tr}\Big[\Psi^{-1}\times\mathbb{E}_{q_t}S\Big], \qquad (3.7)
\end{aligned}$$

where q_t is a given distribution from the last step, (Ψ,Λ) are the parameters that need updating in this step. And be careful, the matrix S is a quadratic form containing the parameter Λ, which needs updating.

Now we have an explicit form of $\mathbb{E}_{q_t}\Big\{\log p(\mathbb{X},\mathbb{Y}\mid\boldsymbol{\theta})\Big\}$. Then we can proceed to the maximization step.

3.3.2 M-Step of Factor Analysis

Note that we are maximizing (3.7), some of whose terms do not contain $\boldsymbol{\theta}$. Ignoring those terms, the maximization problem, in essence, is

$$\max_{\Lambda,\Psi} \left\{ -\frac{N}{2} \log |\Psi| - \frac{N}{2} \mathrm{tr}\Big[\Psi^{-1} \times \mathbb{E}_{q_t} S \Big] \right\}, \tag{3.8}$$

where Λ is inside the matrix S. So, we set the maximizers to be $\Lambda^{(t+1)}$ and $\Psi^{(t+1)}$, and we are done. Since the FA model has explicit solution under normal assumption, let's continue to see what the optimizer are.

To deal with this, we need to simplify the term $\mathbb{E}_{q_t} S$. Recall that

$$\begin{aligned} S &= \frac{1}{N} \sum_{i=1}^{N} (\mathbf{y}_i - \Lambda \mathbf{x}_i) (\mathbf{y}_i - \Lambda \mathbf{x}_i)^\top \\ &= \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{y}_i \mathbf{y}_i^\top - \mathbf{y}_i \mathbf{x}_i^\top \Lambda^\top - \Lambda \mathbf{x}_i \mathbf{y}_i^\top + \Lambda \mathbf{x}_i \mathbf{x}_i^\top \Lambda^\top \right), \end{aligned}$$

and the conditional distribution q_t of each observation $\mathbf{x}_i$ is

$$\begin{aligned} \mathbf{X}_i \mid \mathbf{Y}_i = \mathbf{y}_i \sim \mathbf{N}\left(\mathbf{m}_i^{(t)}, V_t \right), \\ \text{where } \ \mathbf{m}_i^{(t)} = V_t \Lambda_t^\top \Psi_t^{-1} (\mathbf{y}_i - \boldsymbol{\mu}) \quad \text{and} \quad V_t = \left(I_p + \Lambda_t^\top \Psi_t \Lambda_t \right)^{-1}. \end{aligned}$$

Therefore, we can examine the expectation of each term in S one by one:

$$\mathbb{E}_{q_t} \left(\mathbf{y}_i \mathbf{x}_i^\top \Lambda^\top \right) = \mathbf{y}_i \left(\mathbf{m}_i^{(t)} \right)^\top \Lambda^\top \quad \text{and} \quad \mathbb{E}_{q_t} \left(\Lambda \mathbf{x}_i \mathbf{y}_i^\top \right) = \Lambda \mathbf{m}_i^{(t)} \mathbf{y}_i^\top.$$

Then we can write them in the matrix form:

$$\begin{aligned} \sum_{i=1}^{N} \mathbb{E}_{q_t} \left(\mathbf{y}_i \mathbf{x}_i^\top \Lambda^\top \right) = \mathbb{Y}^\top M_t \Lambda^\top \quad \text{and} \quad \sum_{i=1}^{N} \mathbb{E}_{q_t} \left(\Lambda \mathbf{x}_i \mathbf{y}_i^\top \right) = \Lambda M_t^\top \mathbb{Y}, \\ \text{where } \ M_t = \begin{bmatrix} \left(\mathbf{m}_1^{(t)} \right)^\top \\ \vdots \\ \left(\mathbf{m}_N^{(t)} \right)^\top \end{bmatrix} = \mathbb{Y}_c \left(\Psi^{-1} \right)^\top \Lambda V_t^\top \quad \text{and} \quad \mathbb{Y}_c \ \text{ is the centered data matrix.} \end{aligned}$$

The expectation of the last term with respect to q_t is

$$\begin{aligned} \mathbb{E}_{q_t} \left(\Lambda \mathbf{x}_i \mathbf{x}_i^\top \Lambda^\top \right) &= \Lambda \mathbb{E}_{q_t} \left[\mathbf{x}_i \mathbf{x}_i^\top \right] \Lambda^\top \\ &= \Lambda \Big\{ \mathrm{Var}_{q_t}(\mathbf{x}_i) + (\mathbb{E}_{q_t} \mathbf{x}_i) (\mathbb{E}_{q_t} \mathbf{x}_i)^\top \Big\} \Lambda^\top \\ &= \Lambda \Big\{ V_t + \left(\mathbf{m}_i^{(t)} \right) \left(\mathbf{m}_i^{(t)} \right)^\top \Big\} \Lambda^\top. \end{aligned}$$

Writing it into matrix form, it follows

$$\sum_{i=1}^{N} \mathbb{E}_{q_t} \left(\Lambda \mathbf{x}_i \mathbf{x}_i^\top \Lambda^\top \right) = N \Lambda V_t \Lambda^\top + \Lambda M_t^\top M_t \Lambda^\top.$$

Therefore, we can rewrite Equation (3.8) into

$$
\begin{aligned}
&-\frac{N}{2}\log|\Psi| - \frac{N}{2}\mathrm{tr}\Big[\Psi^{-1}\times\mathbb{E}_{q_t}S\Big] \\
&\qquad = -\frac{N}{2}\log|\Psi| - \frac{1}{2}\mathrm{tr}\left\{\Psi^{-1}\cdot\left[\mathbb{Y}^\top\mathbb{Y} - \mathbb{Y}^\top M_t\Lambda^\top - \Lambda M_t^\top\mathbb{Y} + N\Lambda V_t\Lambda^\top + \Lambda M_t^\top M_t\Lambda^\top\right]\right\}. \qquad (3.9)
\end{aligned}
$$

Now we maximize this function (3.9). Taking derivative, it follows

$$
\begin{aligned}
\frac{\partial}{\partial\Lambda}(3.9) &= \frac{1}{2}\frac{\partial}{\partial\Lambda}\mathrm{tr}\left\{\Psi^{-1}\mathbb{Y}^\top M_t\Lambda^\top + \Psi^{-1}\Lambda M_t^\top\mathbb{Y} - N\Psi^{-1}\Lambda V_t\Lambda^\top - \Psi^{-1}\Lambda M_t^\top M_t\Lambda^\top\right\} \\
&= \frac{1}{2}\left[\Psi^{-1}\mathbb{Y}^\top M_t + \Psi^{-1}\mathbb{Y}^\top M_t - 2N\Psi^{-1}\Lambda V_t - 2\Psi^{-1}\Lambda M_t^\top M_t\right] \\
&= \Psi^{-1}\mathbb{Y}^\top M_t - N\Psi^{-1}\Lambda V_t - \Psi^{-1}\Lambda M_t^\top M_t,
\end{aligned}
$$

and

$$
\frac{\partial}{\partial\left(\Psi^{-1}\right)}(3.9) = \frac{N}{2}\Psi - \frac{1}{2}\left[\mathbb{Y}^\top\mathbb{Y} - \mathbb{Y}^\top M_t\Lambda^\top - \Lambda M_t^\top\mathbb{Y} + N\Lambda V_t\Lambda^\top + \Lambda M_t^\top M_t\Lambda^\top\right].
$$

Setting two derivatives to be zero, we get

$$
\begin{aligned}
\Lambda_{t+1} &= \mathbb{Y}^\top M_t\left(NV_t + M_t^\top M_t\right)^{-1} \\
\Psi_{t+1} &= \frac{1}{N}\left[\mathbb{Y}^\top\mathbb{Y} - \mathbb{Y}^\top M_t\Lambda_{t+1}^\top - \Lambda_{t+1}M_t^\top\mathbb{Y} + N\Lambda_{t+1}V_t\Lambda_{t+1}^\top + \Lambda_{t+1}M_t^\top M_t\Lambda_{t+1}^\top\right] \\
&= \frac{1}{N}\left[\mathbb{Y}^\top\mathbb{Y} - \Lambda_{t+1}M_t^\top\mathbb{Y}\right].
\end{aligned}
$$

One may note that the Ψ matrix is required to be diagonal. How do we guarantee this in every updates? We provide one feasible way: first optimizing it without constraints, and then removing all un-diagonal terms, haha, surprisingly easy.

3.3.3 EM Algorithm with Lasso Penalty

We can also add the Lasso penalty to get the sparsity structure of Λ in the EM algorithm. The original optimizatin problem changes a little from

$$
\max_{\boldsymbol{\theta}}\log p(\mathbb{Y}\mid\boldsymbol{\theta}) \quad\Rightarrow\quad \max_{\boldsymbol{\theta}}\Big\{\log p(\mathbb{Y}\mid\boldsymbol{\theta}) - \lambda||\Lambda||_1\Big\}.
$$

How to implement this in the EM algorithm? Now we will use a different technique instead of plugging S. Recall that original expectation in E-step is

$$
\begin{aligned}
&\mathbb{E}_{q_t}\Big\{\log p(\mathbb{X},\mathbb{Y}\mid\boldsymbol{\theta})\Big\} \\
&\qquad = \mathbb{E}_{q_t}\left\{\sum_{i=1}^N\log p(\mathbf{x}_i) - \frac{Nq}{2}\log(2\pi) - \frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_{i=1}^N\left(\mathbf{y}_i - \Lambda\mathbf{x}_i\right)^\top\Psi^{-1}\left(\mathbf{y}_i - \Lambda\mathbf{x}_i\right)\right\}.
\end{aligned}
$$

With Lasso penalty, we need to subtract the L_1 norm from the penalty, i.e.

$$
\mathbb{E}_{q_t}\left\{\sum_{i=1}^N\log p(\mathbf{x}_i) - \frac{Nq}{2}\log(2\pi) - \frac{N}{2}\log|\Psi| - \frac{1}{2}\sum_{i=1}^N\left(\mathbf{y}_i - \Lambda\mathbf{x}_i\right)^\top\Psi^{-1}\left(\mathbf{y}_i - \Lambda\mathbf{x}_i\right) - \lambda||\Lambda||_1\right\}.
$$

With the similar argument that some terms are not related to parameters, so the maximization goal is

$$\max_{\Lambda,\Psi} \mathbb{E}_{q_t} \left\{ -\frac{N}{2} \log |\Psi| - \frac{1}{2} \sum_{i=1}^{N} (\mathbf{y}_i - \Lambda \mathbf{x}_i)^\top \Psi^{-1} (\mathbf{y}_i - \Lambda \mathbf{x}_i) - \lambda ||\Lambda||_1 \right\}.$$

With the Lasso penalty, optimization step needs modification. Since Ψ is diagonal, write

$$\sum_{i=1}^{N} (\mathbf{y}_i - \Lambda \mathbf{x}_i)^\top \Psi^{-1} (\mathbf{y}_i - \Lambda \mathbf{x}_i) = \sum_{i=1}^{N} \sum_{j=1}^{q} \frac{(\mathbf{y}_{ij} - \Lambda_{j\cdot} \mathbf{x}_i)^2}{\psi_j} = \sum_{j=1}^{q} \sum_{i=1}^{N} \frac{(\mathbf{y}_{ij} - \Lambda_{j\cdot} \mathbf{x}_i)^2}{\psi_j}.$$

So, the matrix Λ in the second term is separated into rows. And we can do the same for the third term:

$$\lambda ||\Lambda||_1 = \sum_{j=1}^{q} \lambda ||\Lambda_{j\cdot}||_1.$$

Then note that the maximization problem could be written in the form

$$\begin{aligned}
&\max_{\Lambda,\Psi} \mathbb{E}_{q_t} \left\{ -\frac{N}{2} \log |\Psi| - \frac{1}{2} \sum_{i=1}^{N} (\mathbf{y}_i - \Lambda \mathbf{x}_i)^\top \Psi^{-1} (\mathbf{y}_i - \Lambda \mathbf{x}_i) - \lambda ||\Lambda||_1 \right\} \\
&\qquad = \max_{\Lambda,\Psi} \mathbb{E}_{q_t} \left\{ -\frac{N}{2} \log |\Psi| - \frac{1}{2} \sum_{j=1}^{q} \sum_{i=1}^{N} \frac{(\mathbf{y}_{ij} - \Lambda_{j\cdot} \mathbf{x}_i)^2}{\psi_j} - \sum_{j=1}^{q} \lambda ||\Lambda_{j\cdot}||_1 \right\} \\
&\qquad = \max_{\Lambda,\Psi} \mathbb{E}_{q_t} \left\{ -\frac{N}{2} \log |\Psi| - \frac{1}{2} \sum_{j=1}^{q} \left[\sum_{i=1}^{N} \frac{(\mathbf{y}_{ij} - \Lambda_{j\cdot} \mathbf{x}_i)^2}{\psi_j} - \lambda ||\Lambda_{j\cdot}||_1 \right] \right\} \\
\left(\text{Replace } \mathbf{x}_i \text{ by } \mathbf{m}_i^{(t)} \right) &\approx \max_{\Lambda,\Psi} \left\{ -\frac{N}{2} \log |\Psi| - \frac{1}{2} \sum_{j=1}^{q} \left[\sum_{i=1}^{N} \frac{\left(\mathbf{y}_{ij} - \Lambda_{j\cdot} \mathbf{m}_i^{(t)} \right)^2}{\psi_j} - \lambda ||\Lambda_{j\cdot}||_1 \right] \right\},
\end{aligned}$$

where we can recognize this term

$$\sum_{i=1}^{N} \frac{\left(\mathbf{y}_{ij} - \Lambda_{j\cdot} \mathbf{m}_i^{(t)} \right)^2}{\psi_j} - \lambda ||\Lambda_{j\cdot}||_1 = \sum_{i=1}^{N} \left(\frac{\mathbf{y}_{ij}}{\sqrt{\psi_j}} - \Lambda_{j\cdot} \frac{\mathbf{m}_i^{(t)}}{\sqrt{\psi_j}} \right)^2 - \lambda ||\Lambda_{j\cdot}||_1 \tag{3.10}$$

is a Lasso LSE problem of j-th row.

Therefore, the EM algorithm with Lasso penalty for FA model is as follows.

Algorithm 6. Set an initial value for (Λ_0, Ψ_0).

- Calculate the conditional mean for every observation $\mathbf{x}_i$ at this step:

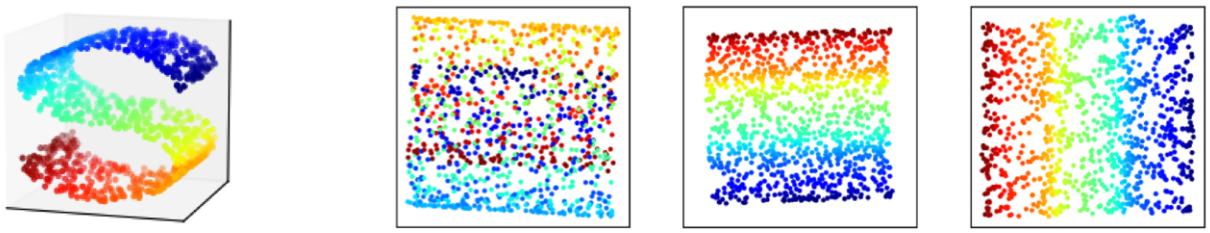
$$\mathbf{m}_i^{(t)} = V_t \Lambda_t^\top \Psi_t^{-1} (\mathbf{y}_i - \boldsymbol{\mu}) \quad \text{and} \quad V_t = \left(I_p + \Lambda_t^\top \Psi_t \Lambda_t \right)^{-1}$$

- Update of Λ: Fixing $j = 1, 2, \ldots, q$, using the known quantity $\mathbf{m}_i^{(t)}, \mathbf{y}_{ij}$ and $\psi_j^{(t)}$, run Lasso regression (3.10) and solve the j-the row updates $\Lambda_{j\cdot}^{(t+1)}$. Repeat until all rows are updated.
- Update of Ψ: Update $\Psi^{(t+1)} = \frac{1}{N} \left[\mathbb{Y}^\top \mathbb{Y} - \Lambda^{(t+1)} M_t^\top \mathbb{Y} \right]$.

Chapter 4

MDS and Kernel Methods

When it comes to dimension reduction, PCA is the most famous tool, which projects the raw data to a linear subspace to get dimension reduction. However, linear reduction may not capture the characters of the raw data such as the example in Figure 4.1.



Raw Data (a) PCA Projection (b) LLE Projection (c) IsoMap Projection

Figure 4.1: Manifold raw data space

As we see, the raw data lie in a curved surface in $\mathbb{R}^3$ with the shape of the letter 'S'. Under this circumstance, no matter which linear subspace we choose, PCA would not behave good enough as we wee in Figure 4.1(a). This is why we need the non-linear dimension reduction, as 4.1(b) and 4.1(c).

4.1 Multidimensional Scaling

We start with a linear method called *multidimensional scaling* (MDS), which is the source of non-linear method. Though it is a linear method, unlike the PCA utilizing the covariance structure, MDS aims to find a lower dimensional space, which minimizes the loss of similarity of raw data.

Problem Setup (MDS)**.** Suppose the raw observed data are $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{R}^q$, with a distance (dissimilarity) measure on that space:

$$d_1(\cdot\,,\,\cdot) : \mathbb{R}^q \times \mathbb{R}^q \mapsto \mathbb{R}.$$

The goal is to find a lower dimensional representation of the data $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^p$, also with a distance

measure $d_2(\cdot\ ,\ \cdot)$ on that space such that minimizes the dissimilarity differences, i.e.

$$\sum_{1\leq i,j\leq N}\Big\{d_1\left(\mathbf{y}_i,\mathbf{y}_j\right)-d_2\left(\mathbf{x}_i,\mathbf{x}_j\right)\Big\}^2.$$

Note that the distances d_1, d_2 should be previously set, so the optimization is with respect to $\mathbf{x}$'s.

The distance could be defined in many ways, such as

$$\text{Euclidean distance:}\quad d_{ij}=||\mathbf{y}_i-\mathbf{y}_j||\quad \text{for each pair } \mathbf{y}_i,\mathbf{y}_j\in\mathbb{R}^q.$$

No matter which distances we use, our goal is the same: dimension reduction with the similarity structure of observations preserved. And with some complex choices of d, we may recover non-linear structure, as we shall see in the kernel PCA topics. Let's start with the classical MDS.

4.1.1 The Classical MDS

In fact, to minimize the dissimilarity difference is equivalent to minimize the similarity difference. So, the classical MDS works with a similarity measure for $\mathbf{y}$'s and $\mathbf{x}$'s, letting

$$d_1(\mathbf{y}_i,\mathbf{y}_j)=\mathbf{y}_i^\top\mathbf{y}_j\quad\text{and}\quad d_2(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^\top\mathbf{x}_j.$$

The formal definition is as follow.

Definition 4 (Classical MDS)**.** Suppose the raw observed data are $\mathbf{y}_1,\ldots,\mathbf{y}_N\in\mathbb{R}^q$. The goal is to find a lower dimensional representation of the data $\mathbf{x}_1,\ldots,\mathbf{x}_N\in\mathbb{R}^p$ such that

$$\min_{\mathbf{x}_1,\ldots,\mathbf{x}_N}\sum_{1\leq i,j\leq N}\left(\mathbf{y}_i^\top\mathbf{y}_j-\mathbf{x}_i^\top\mathbf{x}_j\right)^2.$$

As usual, we assume the data is centered for simplicity. If we write the design matrices as

$$\mathbb{Y}_{N\times q}=\begin{pmatrix}\mathbf{y}_1^\top\\ \mathbf{y}_2^\top\\ \vdots\\ \mathbf{y}_N^\top\end{pmatrix}\quad\text{and}\quad\mathbb{X}_{N\times p}=\begin{pmatrix}\mathbf{x}_1^\top\\ \mathbf{x}_2^\top\\ \vdots\\ \mathbf{x}_N^\top\end{pmatrix},$$

the minimization problems could be written as

$$\min_{\mathbb{X}}\left|\left|\mathbb{Y}\mathbb{Y}^\top-\mathbb{X}\mathbb{X}^\top\right|\right|_F,$$

where the norm is Frobenius's norm, which is defined as

$$||A||_F:=\sqrt{\mathrm{Tr}(A^\top A)}=\sqrt{\sum_{i,j}a_{ij}^2}\quad\text{(sum of square of all terms)}.$$

If we SVD design matrices as (suppose $q < N$)

$$\mathbb{Y} = U_{N\times q} D_{q\times q} V_{q\times q}^\top \quad \text{and} \quad \mathbb{X} = U^*_{N\times p} D^*_{p\times p} (V^*)^\top_{p\times p},$$

we can see

$$\mathbb{Y}\mathbb{Y}^\top = U_{N\times q} D^2_{q\times q} U^\top_{q\times N} \quad \text{and} \quad \mathbb{X}\mathbb{X}^\top = U^*_{N\times p} (D^*)^2_{p\times p} (U^*)^\top_{p\times N}.$$

In essence, we are finding a low rank approximation $\mathbb{X}\mathbb{X}^\top$ for $\mathbb{Y}\mathbb{Y}^\top$. And we will call

$$\text{Gram matrix:} \quad \mathbb{Y}\mathbb{Y}^\top.$$

Fortunately, the classical MDS's optimization is completed by the following theorem.

Theorem 5 (Eckart–Young Theorem)**.** *The best p-rank approximation of* $\mathbb{Y}\mathbb{Y}^\top$ *is*

$$\mathbb{Y}\mathbb{Y}^\top \approx \sum_{j=1}^{p} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top, \quad \textit{where } \lambda_j \textit{ and } \mathbf{u}_j \textit{ are the first } p \textit{ eigens of } \mathbb{Y}\mathbb{Y}^\top.$$

4.1.2 Comparison between MDS and PCA

The Eckart–Young Theorem states a very similar result to PCA:

- In the classical MDS, we do spectral decomposition of Gram matrix $\left(\mathbb{Y}\mathbb{Y}^\top\right)_{N\times N}$
- In the classical PCA, we do spectral decomposition of the sample covariance matrix $\left(\mathbb{Y}^\top\mathbb{Y}\right)_{q\times q}$

In fact, we shall see they are mathematically the same in the view of SVD of $\mathbb{Y}$.

Same Procedures by SVD

Now suppose

$$\mathbb{Y}_{N\times q} = U_{N\times r} D_{r\times r} V^\top_{r\times q},$$

by simple algebra, leading to

$$\mathbb{Y}^\top\mathbb{Y} = V D^2 V^\top \quad \text{and} \quad \mathbb{Y}\mathbb{Y}^\top = U D^2 U^\top,$$

meaning that

- the columns of V are the eigenvectors of $(n-1)S = \mathbb{Y}^\top\mathbb{Y}$,
- the columns of U are the eigenvectors of Gram matrix $\mathbb{Y}\mathbb{Y}^\top$,
- and the two kinds of decomposition share the same eigenvalues D^2 (normally ordered $d_1 > \cdots > d_r$).

Same Projection Scores

Additionally, the classical MDS has the same projection scores $\mathbb{X}$ as the classical PCA. W.L.O.G., we assume we take $p = r$, and the proof for other case is the same. Recall that the score vector $\mathbf{x}_i$ and score matrix $\mathbb{X}$ are (PCA loading matrix now is denoted by V)

$$\mathbf{x}_i = V^\top \mathbf{y}_i \quad \Rightarrow \quad \mathbb{X} = \mathbb{Y}V = UDV^\top V = UD.$$

It remains to show that MDS has the same score matrix. Recall that in the last section, when we minimizing

$$\min_{\mathbb{X}} \left|\left|\mathbb{Y}\mathbb{Y}^\top - \mathbb{X}\mathbb{X}^\top\right|\right|_F,$$

Eckart and Young give the best approximation

$$\mathbb{Y}\mathbb{Y}^\top \approx \sum_{j=1}^{p} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top, \quad \text{where } \lambda_j \text{ and } \mathbf{u}_j \text{ are the first } p \text{ eigens of } \mathbb{Y}\mathbb{Y}^\top,$$

i.e., the best approximates $\mathbb{X}\mathbb{X}^\top$ is (when $p = r$)

$$\mathbb{X}\mathbb{X}^\top = \sum_{j=1}^{r} \lambda_j \mathbf{u}_j \mathbf{u}_j^\top = U \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_r \end{pmatrix} U^\top = UD^2U^\top = (UD)(UD)^\top.$$

Therefore, the score matrix $\mathbb{X} = UD$, is the same as PCA!

4.2 Kernel PCA

Though the kernel PCA is named after PCA, the idea is more like the MDS, i.e. calculating the similarity measure of observations $\mathbb{Y}\mathbb{Y}^\top$, NOT the covariance structure $\mathbb{Y}^\top\mathbb{Y}$. And by choosing different kernels, we are 'projecting' raw data, which is not linear separable, to higher dimensions, where they become separable. Here is an example of $\mathbb{R}^2$ non-separable but $\mathbb{R}^3$ separable data.

Example. Originally, data sit in the $\mathbb{R}^2$ plane, circle-like. In this case, no matter which linear subspace we choose, the two classes could not be separated. However, if we choose the *higher dimension projection* function

$$\forall \mathbf{y} \in \mathbb{R}^2: \quad \phi(\mathbf{y}) = \begin{pmatrix} \mathbf{x}_1 = \mathbf{y}_1 \\ \mathbf{x}_2 = \mathbf{y}_2 \\ \mathbf{x}_3 = \mathbf{y}_1^2 + \mathbf{y}_2^2 \end{pmatrix},$$

then it becomes possible to separate them now by a linear subspace in $\mathbb{R}^3$.

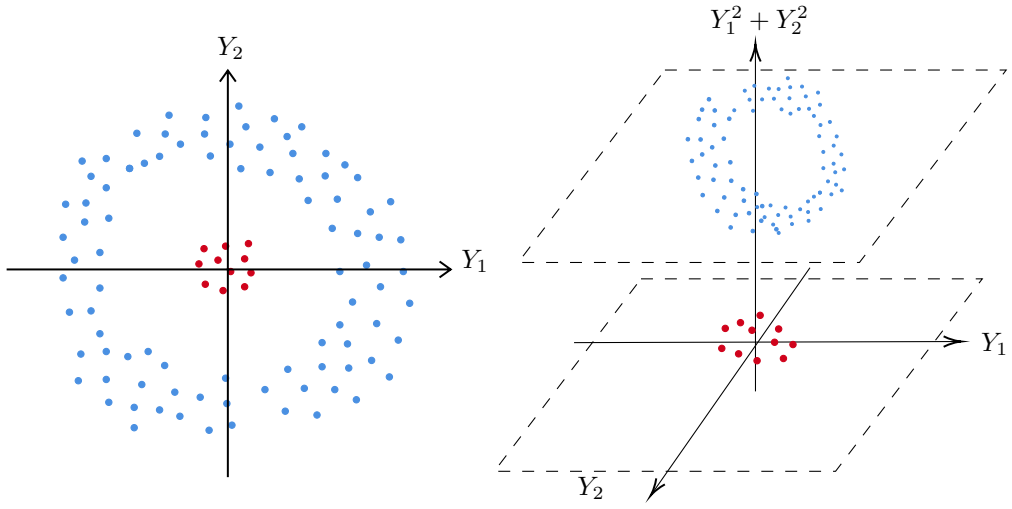


Figure 2: Higher dimension projection

So, we can conclude the steps for kernel PCA now. First, find a function ϕ which maps the data to a higher dimension space. And then do the classical MDS on the new data.

4.2.1 Kernel Functions

By the idea of the last paragraph, how to choose the ϕ function becomes a problem. By the magic of mathematics, we don't need to choose the function $\phi : \mathbb{R}^q \mapsto \mathbb{R}^p$, but a kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$.

Definition 5 (Kernel function)**.** A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is called kernel function if it is

- symmetric, i.e. $k(\mathbf{y}_1, \mathbf{y}_2) = k(\mathbf{y}_2, \mathbf{y}_1)$;
- positive semi-definite, i.e. $\forall n \in \mathbb{N}$, $\forall \mathbf{y}_1, \ldots, \mathbf{y}_n \in \mathbb{R}^q$ and $\forall \alpha_1, \ldots, \alpha_n \in \mathbb{R}$,

$$\sum_{i,j} \alpha_i \alpha_j k(\mathbf{y}_i, \mathbf{y}_j) \geq 0.$$

The reason to introduce the kernel functions is the following theorem.

Theorem 6 (Kernel implies embedding)**.** *A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$ is a kernel if and only if there exists a Hilbert space $\mathscr{H}$ (with an inner product) and a feature map $\phi : \mathcal{X} \mapsto \mathscr{H}$ such that*

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

Remark. To simply understand it, we can use the following idea. In practice, we don't need to pick the feature map $\phi(\cdot) : \mathbb{R}^q \mapsto \mathbb{R}^p$, but pick the kernel function $k(\cdot, \cdot) : \mathbb{R}^q \times \mathbb{R}^q \mapsto \mathbb{R}$, because once the kernel k is chosen, the feature map ϕ is automatically chosen as well.

Proof. The if part is easy by checking that $k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle$ is indeed a symmetric kernel function. Now we prove the only if part, assuming that function k is a kernel. Our goal is to construct a Hilbert space $\mathscr{H}$ with an inner product $\langle \cdot, \cdot \rangle$, and a mapping $\phi : \mathcal{X} \mapsto \mathbb{R}$ such that

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

First we consider the feature map $\phi : \mathcal{X} \mapsto \mathbb{R}^{\mathcal{X}}$, where $\mathcal{X}$ is the sample space and $\mathbb{R}^{\mathcal{X}}$ is the space of all real-valued functions from $\mathcal{X} \mapsto \mathbb{R}$, defined as

$$\mathbf{x} \mapsto \phi(\mathbf{x}) := k_{\mathbf{x}} := k(\mathbf{x}, \cdot).$$

That is, the point $\mathbf{x} \in \mathcal{X}$ is mapped to the function $k_{\mathbf{x}} : \mathcal{X} \mapsto \mathbb{R}$, $k_{\mathbf{x}}(\mathbf{y}) = k(\mathbf{x}, \mathbf{y})$. Now we collect all images of this kind of functions

$$\mathcal{G} := \left\{ \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \cdot) : \alpha_i \in \mathbb{R},\ n \in \mathbb{N},\ \mathbf{x}_i \in \mathcal{X} \right\},$$

i.e. the linear span of all feature maps $k_{\mathbf{x}}$. So, $\mathcal{G}$ contains all finite linear combinations of feature functions, and we can verify it is a vector space. As for the inner product on $\mathcal{G}$, we define

$$\begin{cases} \langle k_{\mathbf{x}}, k_{\mathbf{y}} \rangle = \langle k(\mathbf{x}, \cdot), k(\mathbf{y}, \cdot) \rangle := k(\mathbf{x}, \mathbf{y}) & \text{if } \mathbf{x}, \mathbf{y} \in \mathcal{X}, \\ \langle f, g \rangle := \sum_{ij} \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{y}_j) & \text{if } f = \sum_i \alpha_i k(\mathbf{x}_i, \cdot) \text{ and } g = \sum_j \alpha_j k(\mathbf{y}_i, \cdot). \end{cases}$$

Defining this way, we can check (left as exercises) that this operation is well defined (the same function $f \in \mathcal{G}$ could have different linear combination presentation, verify they lead to the same result), and satisfies all properties that needed to be an inner product (this is where positive definiteness comes to use).

Finally, to make $\mathcal{G}$ a Hilbert space, we take its topological completion $\bar{\mathcal{G}}$, adding all limits of Cauchy sequences into this space. This result space $\mathcal{H} := \bar{\mathcal{G}}$ is the wanted space, and called the *reproducing kernel Hilbert space* (RKHS). And by construction, we have for all $\mathbf{x}, \mathbf{y} \in \mathcal{X}$,

$$k(\mathbf{x}, \mathbf{y}) = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

□

Why it is called reproducing kernel Hilbert space? Because of the following property.

Proposition 7. *Suppose $\mathscr{H}$ is a reproducing kernel Hilbert space and $f \in \mathscr{H}$ has the presentation of $f = \sum_i \alpha_i k(\mathbf{x}_i, \cdot)$, then for all $\mathbf{x} \in \mathcal{X}$,*

$$\langle f, k(\mathbf{x}, \cdot) \rangle = f(\mathbf{x}).$$

Proof. The result follows from

$$\langle f, k(\mathbf{x}, \cdot) \rangle = \left\langle \sum_i \alpha_i k(\mathbf{x}_i, \cdot), k(\mathbf{x}, \cdot) \right\rangle = \sum_i \alpha_i \langle k(\mathbf{x}_i, \cdot), k(\mathbf{x}, \cdot) \rangle = \sum_i \alpha_i k(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x}).$$

□

By this idea, what we need to do in practice, is choosing a kernel k, and calculating the Gram matrix $K = \left\{ k(\mathbf{y}_i, \mathbf{y}_j) \right\}_{ij}$, and implementing MDS. Doing this way, we even gain a benefit as in the next examples.

Example. Let's see few examples of kennel functions, some of whose feature map could be derived, but the others are not.

(a) Linear PCA chooses the kernel to be $k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y}$. So for a fixed point $\mathbf{x}$ in the sample $\mathcal{X}$, its feature map is defined by

$$\phi(\mathbf{x}) := k(\mathbf{x}, \cdot) := k_{\mathbf{x}}(\cdot) \quad \text{with inner product} \quad k(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y} = \langle \phi(\mathbf{x}), \phi(\mathbf{y}) \rangle .$$

Therefore, we can tell the feature map $\phi(\mathbf{x}) = k_{\mathbf{x}}(\cdot)$ is an identity function that

$$\forall \mathbf{y} \in \mathcal{X} : \quad k_{\mathbf{x}}(\mathbf{y}) = \mathbf{x}.$$

(b) Polynomial kernel with order d chooses

$$k(\mathbf{y}_i, \mathbf{y}_j) = \left(1 + \mathbf{y}_i^\top \mathbf{y}_j\right)^d .$$

For example, when $d = 2$, $q = 2$, it is

$$\begin{aligned} k(\mathbf{y}_i, \mathbf{y}_j) = \left(1 + \mathbf{y}_i^\top \mathbf{y}_j\right)^2 &= (1 + y_{11}y_{21} + y_{21}y_{22})^2 \\ &= 1 + 2y_{11}y_{21} + 2y_{12}y_{22} + y_{11}^2 y_{21}^2 + y_{21}^2 y_{22}^2 + 2y_{11}y_{12}y_{21}y_{22}. \end{aligned}$$

After simplification, we can also find the feature map $\mathbb{R}^2 \mapsto \mathbb{R}^6$:

$$\phi(\mathbf{y}) = k_{\mathbf{y}}(\cdot) = \begin{pmatrix} 1 & \sqrt{2}y_1 & \sqrt{2}y_2 & y_1^2 & y_2^2 & \sqrt{y_1 y_2} \end{pmatrix}^\top \quad \text{and} \quad k(\mathbf{y}_i, \mathbf{y}_j) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle .$$

(c) Gaussian kernel with dispersion parameter σ^2 is

$$k(\mathbf{y}_i, \mathbf{y}_j) = \exp\left\{ -\frac{||\mathbf{y}_i - \mathbf{y}_j||^2}{\sigma^2} \right\}.$$

However, we can not find an explicit expression for the feature map ϕ under this kernel, but its existence is guaranteed by the last theorem.

These examples show that specifying kernel function is sometimes more powerful than specifying the feature map since some kernel, such as the Gaussian, conveys a good meaning of distance but has no explicit feature map expression.

4.2.2 Procedures of Kernel PCA

Kernel Matrix Method

Here we give the general procedures of Kernel PCA implementation.

Algorithm 7. The procedure to do Kernel PCA for $\mathbb{Y}$ is:

(a) Pick a kernel function $k(\mathbf{y}_i, \mathbf{y}_j) : \mathbb{R}^q \times \mathbb{R}^q \mapsto \mathbb{R}$.

(b) Calculate the Gram matrix of the kernel data

$$K = \begin{pmatrix} k(\mathbf{y}_1, \mathbf{y}_2) & k(\mathbf{y}_1, \mathbf{y}_3) & \cdots & k(\mathbf{y}_1, \mathbf{y}_N) \\ k(\mathbf{y}_2, \mathbf{y}_2) & k(\mathbf{y}_2, \mathbf{y}_3) & \cdots & k(\mathbf{y}_2, \mathbf{y}_N) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{y}_N, \mathbf{y}_2) & k(\mathbf{y}_N, \mathbf{y}_3) & \cdots & k(\mathbf{y}_N, \mathbf{y}_N) \end{pmatrix}.$$

(c) Center the kernel matrix

$$\tilde{K} = \left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top \right) K \left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top \right).$$

(d) Implement the classical MDS to the centered kernel matrix $\tilde{K}$, in essence, spectral decompose

$$\tilde{K}\mathbf{u}_i = \lambda \mathbf{u}_i \quad \Longleftrightarrow \quad \tilde{K} = Q \Lambda Q^\top.$$

(e) For any new data $\mathbf{y}^*$, we can project it to each new dimension j

$$\mathbf{x}_j^* = \sum_{i=1}^{N} k(\mathbf{y}_i, \mathbf{y}^*) \mathbf{u}_{ij}.$$

Dissimilarity Matrix Method

Since the inner product in $\mathscr{H}$ uses the view of similarity measure, as discussed in the classical MDS, the practical case is sometimes different. Specifically, we are given the dissimilarity matrix, or distance matrix. If the distance is defined as the Euclidean distance, i.e.

$$d_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_j||^2,$$

we can transform the distance matrix $D^2 = (d_{ij})^2$ into kernel gram matrix K by

$$K = -\frac{1}{2}\left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top\right) D^2 \left(I - \frac{1}{N}\mathbf{1}\mathbf{1}^\top\right).$$

And more generally, if the distance is other type rather than the Euclidean one, we can also apply this formula (but the kernel will be unknown).

4.2.3 Mathematical View of Kernel PCA

Theorem 6 says if we choose a kernel k to do kernel PCA, we are actually doing the classical PCA on the new feature data $\phi(\mathbf{y}_i)$'s. But recall that $\phi(\mathbf{y})$ is a function! What is the PCA of functions? In Example 6, we see that the polynomial kernel with order 2 has the feature map:

$$\phi(\mathbf{y}) := k_{\mathbf{y}}(\cdot) = \begin{pmatrix} 1 & \sqrt{2}y_1 & \sqrt{2}y_2 & y_1^2 & y_2^2 & \sqrt{y_1 y_2} \end{pmatrix}^\top.$$

This is a constant function when $\mathbf{y}$ is specified, so it is understandable. What if this function $k_{\mathbf{y}}(\cdot)$ is NOT constant?

To avoid too much involved in functional analysis, we assume conditions

- feature data are centered, i.e. $\sum_{i=1}^N \phi(\mathbf{y}_i) = 0$;
- the RKHS $\mathscr{H}$ is M-dim (in the true case, it should be ∞) and $M > q$.

Solution of Feature Data PCA

Now suppose the kernel k is pre-specified. Though we don't know what the feature functions $\phi(\mathbf{y}_i)$'s are, we know it exists and we denote the sample covariance matrix of features as

$$C_{M\times M} = \frac{1}{N}\sum_{i=1}^N \phi(\mathbf{y}_i)\phi(\mathbf{y}_i)^\top$$

with spectral decomposition is

$$j = 1, \ldots, M: \quad C\mathbf{v}_j = \lambda_j \mathbf{v}_j.$$

The key idea is that though C is unknown, the kernel Gram matrix K is known. If we somehow relate λ's and $\mathbf{v}$'s to the known matrix K, then we can solve the PCA of C without working directly in $\mathscr{H}$. Recall

$$k(\mathbf{y}_i, \mathbf{y}_j) = \langle \phi(\mathbf{y}_i), \phi(\mathbf{y}_j) \rangle,$$

and this $K = (k)_{ij}$ matrix is known. And for the matrix C, by its spectral decomposition, we know

$$\left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{y}_i)\phi(\mathbf{y}_i)^\top\right]\mathbf{v}_j = C\mathbf{v}_j = \lambda_j\mathbf{v}_j,$$

where the LHS could be simplified as

$$\left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{y}_i)\phi(\mathbf{y}_i)^\top\right]\mathbf{v}_j = \left[\frac{1}{N}\sum_{i=1}^{N}\phi(\mathbf{y}_i)\left\langle\phi(\mathbf{y}_i),\mathbf{v}_j\right\rangle\right] = \lambda_j\mathbf{v}_j.$$

Note that $\langle\phi(\mathbf{y}_i),\mathbf{v}_j\rangle$ is a scalar, so it means $\mathbf{v}_j$ is a linear combination of $\phi(\mathbf{y}_i)$'s, write

$$\mathbf{v}_j = \sum_{i=1}^{N} a_{ji}\phi(\mathbf{y}_i) = \Phi^\top\mathbf{a}_j, \quad \text{where} \quad \Phi = \begin{pmatrix}\phi(\mathbf{y}_1)^\top\\ \vdots \\ \phi(\mathbf{y}_N)^\top\end{pmatrix} \text{ and } \mathbf{a}_j = \begin{pmatrix}a_{j1}\\ \vdots \\ a_{jN}\end{pmatrix}. \tag{4.1}$$

Using this notation, the matrix C could be rewritten as

$$C = \frac{1}{N}\Phi^\top\Phi \quad \Rightarrow \quad C\mathbf{v}_j = \frac{1}{N}\Phi^\top\Phi\mathbf{v}_j = \lambda_j\mathbf{v}_j \quad \Rightarrow \quad \frac{1}{N}\Phi\Phi^\top\Phi\mathbf{v}_j = \lambda_j\Phi\mathbf{v}_j. \tag{4.2}$$

Note that $\Phi\Phi^\top$ is the kernel Gram matrix:

$$\left(\Phi\Phi^\top\right)_{ij} = \langle\phi(\mathbf{y}_i),\phi(\mathbf{y}_j)\rangle = (K)_{ij},$$

and by construction, $\mathbf{v}_j = \Phi^\top\mathbf{a}_j$. So, plugging these two expressions in to (4.2), it follows that

$$\frac{1}{N}K\Phi\Phi^\top\mathbf{a}_j = \lambda_j\Phi\Phi^\top\mathbf{a}_j \quad \Rightarrow \quad \frac{1}{N}K^2\mathbf{a}_j = \lambda_j K\mathbf{a}_j \quad \Rightarrow \quad \frac{1}{N}K\mathbf{a}_j = \lambda_j\mathbf{a}_j \text{ (assume } K \text{ is invertible)}$$

i.e. the constructed $\{\mathbf{a}_j : j = 1,\ldots,N\}$ and $\{\lambda_j : j = 1,\ldots,N\}$ are the eigens of K/N, which is solvable! Now, once we solve $\mathbf{v}_j$'s, we are done (goal is to decompose C, i.e. finding $\mathbf{v}_j$ and λ_j). However, this is not feasible. What we could do is only standardize $\mathbf{v}_j$. Recall the assumption that

$$j = 1,\ldots,M: \quad C\mathbf{v}_j = \lambda_j\mathbf{v}_j.$$

We only assume directions for $\mathbf{v}_j$, but not require it is unit vector. So, $\mathbf{v}_j$'s are not orthogonal basis now. But by construction

$$\mathbf{v}_j = \sum_{i=1}^{N} a_{ji}\phi(\mathbf{y}_i) = \Phi^\top\mathbf{a}_j.$$

Therefore it follows that

$$\begin{aligned} ||\mathbf{v}_j||^2 = \mathbf{a}_j^\top\Phi\Phi^\top\mathbf{a}_j &= \mathbf{a}_j^\top K\mathbf{a}_j \\ \text{(by construction (4.1) of } \mathbf{a}) \quad &= N\lambda_j\mathbf{a}_j^\top\mathbf{a}_j \\ (\mathbf{a} \text{ is orthogonal basis}) \quad &= N\lambda_j. \end{aligned}$$

So, the orthogonal basis for C should be

$$U_{M\times N} = \begin{bmatrix} \frac{\mathbf{v}_1}{\sqrt{N\lambda_1}} & \frac{\mathbf{v}_2}{\sqrt{N\lambda_2}} & \cdots & \frac{\mathbf{v}_N}{\sqrt{N\lambda_N}} \end{bmatrix} = \begin{bmatrix} \frac{\Phi^\top \mathbf{a}_1}{\sqrt{N\lambda_1}} & \frac{\Phi^\top \mathbf{a}_2}{\sqrt{N\lambda_2}} & \cdots & \frac{\Phi^\top \mathbf{a}_N}{\sqrt{N\lambda_N}} \end{bmatrix}, \tag{4.3}$$

where $\mathbf{a}_j$ is computable by K, but $\mathbf{v}_j$ remains unknown since Φ is not.

Computation of Feature Projection

One use of the decomposition of C is to compute the feature projection for any new or old observation. Suppose we now have the training data $\mathbf{y}_1, \ldots, \mathbf{y}_N$ and a new observation $\mathbf{y} \in \mathbb{R}^q$. And we want to compute the feature projection of the new observation $\mathbf{y}$.

Since now we project $\mathbf{y}$ into the feature space $\phi(\mathbf{y})$, and the basis of that space is (4.2), the kernel PC scores (feature projection) of $\mathbf{y}$ is

$$\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_N \end{pmatrix} = U^\top \phi(\mathbf{y}).$$

If we consider the first kernel PC component, it is

$$\begin{aligned} x_1 &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{v}_1^\top \phi(\mathbf{y}) \\ \text{(by construction (4.1) of } \mathbf{a}) \quad &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1^\top \Phi \phi(\mathbf{y}) \\ &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1^\top \begin{pmatrix} \phi(\mathbf{y}_1)^\top \\ \vdots \\ \phi(\mathbf{y}_N)^\top \end{pmatrix} \phi(\mathbf{y}), \quad \text{where } \mathbf{y}_1, \ldots, \mathbf{y}_N \text{ are training data} \\ &= \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1^\top \begin{pmatrix} k(\mathbf{y}_1, \mathbf{y}) \\ \vdots \\ k(\mathbf{y}_N, \mathbf{y}) \end{pmatrix}. \end{aligned}$$

Therefore, all kernel PC scores of the new observation $\mathbf{y}$ are

$$\mathbf{x}^\top = \begin{bmatrix} k(\mathbf{y}_1, \mathbf{y}) & \cdots & k(\mathbf{y}_N, \mathbf{y}) \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1 & \cdots & \frac{1}{\sqrt{N\lambda_N}} \mathbf{a}_N \end{bmatrix}.$$

As for the kernel PC scores for the training data, we use the notation $\mathbb{Y}$ to denote the design matrix of training data, and put the first p kernel PC scores of $\mathbf{y}$ into a matrix:

$$\mathbb{X}_{N\times p} = \begin{bmatrix} \mathbf{x}_1^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix} = \begin{bmatrix} k(\mathbf{y}_1, \mathbf{y}_1) & \cdots & k(\mathbf{y}_1, \mathbf{y}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{y}_N, \mathbf{y}_1) & \cdots & k(\mathbf{y}_N, \mathbf{y}_1) \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{N\lambda_1}} \mathbf{a}_1 & \cdots & \frac{1}{\sqrt{N\lambda_N}} \mathbf{a}_N \end{bmatrix}.$$

4.3 The General Kernel Method

4.3.1 From Kernel PCA to Kernel Method

Recall the kernel PCA. Specifying the kernel function k, we project the raw data $\mathbf{y}_1, \ldots, \mathbf{y}_N \in \mathbb{R}^q$ into the reproducing kernel Hilbert space (RKHS) $\mathscr{H}_k$ to get $\phi(\mathbf{y}_1), \ldots, \phi(\mathbf{y}_N)$, and do the PCA for $\phi(\mathbf{y}_1), \ldots, \phi(\mathbf{y}_N)$, whose each element is a function. Though we did not work directly on the $\mathscr{H}_k$, what we derive is, in essence, find the PC function $v_1 \in \mathscr{H}_k$ such that

$$\max_{v_1 \in \mathscr{H}_k} \frac{1}{N} \sum_{i=1}^{N} [v_1(\mathbf{y}_i) - \bar{v}_1]^2 \quad \text{subject to} \quad ||v_1||_{\mathscr{H}_k} = 1.$$

Introducing a Lagrange multiplier, we are doing

$$\min_{v_1 \in \mathscr{H}_k} -\frac{1}{N} \sum_{i=1}^{N} [v_1(\mathbf{y}_i) - \bar{v}_1]^2 + \lambda(||v_1||_{\mathscr{H}_k} - 1),$$

and from the derivation in the last lecture note, we know the first PC function of a new observation $\mathbf{y}$ is

$$\mathbf{x} = v_1(\mathbf{y}) = \sum_{i=1}^{N} \frac{a_{1i}}{\sqrt{N\lambda_1}} \tilde{k}(\mathbf{y}_i, \mathbf{y}).$$

This is where we start to generalize the kernel method.

4.3.2 The Usefulness of Kernel Method

The goal of kernel method is to solve the problems of the following form.

Problem Setup (Kernel method)**.** Suppose the paired observed data, $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N) \in (\mathbb{R}^p, \mathbb{R}^q)$, are observed. Moreover, a kernel function k, a loss function L, and a penalty function P are specified. Our goal is to find out the best predictive function $\hat{f}$ such that

$$\hat{f} = \arg\min_{f \in \mathscr{H}_k} \sum_{i=1}^{N} L\Big(\mathbf{y}_i, f(\mathbf{x}_i)\Big) + \lambda P(f).$$

In fact, this setting is used in many models that we have already seen as in the next example.

Example. In non-parametric regression, we model $\mathbb{E}(Y_i \mid \mathbf{x}_i) = f(\mathbf{x})$ by the loss function

$$L(y_i, f(\mathbf{x}_i)) = [y_i - f(\mathbf{x}_i)]^2.$$

But this optimization has infinite many solutions since function space is ∞ dimensional. Therefore, we need restrictions as in kernel ridge regression (to be introduced).

In general, we can not solve this problem easily. However, if we choose the penalty function to be $||\cdot||_{\mathscr{H}_k}$, the next powerful theorem guarantees a unique solution in a linear form.

Theorem 8 (Representer theorem)**.** *Given the data $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)$, a kernel function k and a*

loss function L are specified. The unique solution to

$$\min_{f\in\mathscr{H}_k}\sum_{i=1}^{N}L\Big(\mathbf{y}_i,f(\mathbf{x}_i)\Big)+\lambda||f||_{\mathscr{H}_k}$$

is given by the finite dimensional linear form

$$\hat{f}(\mathbf{x})=\sum_{i=1}^{N}\alpha_i k(\mathbf{x}_i,\mathbf{x})\quad with\quad ||\hat{f}||^2_{\mathscr{H}_k}=\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j),\quad where\quad \boldsymbol{\alpha}=\Big(\alpha_1,\dots,\alpha_N\Big)\in\mathbb{R}^N.$$

By Representer theorem, the optimization problem in ∞-dim $\mathscr{H}_l$ space reduces to a N-dim problem. And it suffices to solve all α's, i.e.

$$\begin{aligned}\arg\min_{f\in\mathscr{H}_k}\sum_{i=1}^{N}L\Big(\mathbf{y}_i,f(\mathbf{x}_i)\Big)+\lambda||f||_{\mathscr{H}_k}&=\arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N}\sum_{i=1}^{N}L\Big(\mathbf{y}_i,\sum_{j=1}^{N}\alpha_j k(\mathbf{x}_j,\mathbf{x}_i)\Big)+\lambda\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j k(\mathbf{x}_i,\mathbf{x}_j)\\&=\arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N}\sum_{i=1}^{N}L\left(\mathbf{y}_i,\boldsymbol{\alpha}^\top\begin{bmatrix}k(\mathbf{x}_1,\mathbf{x}_i)\\\vdots\\k(\mathbf{x}_N,\mathbf{x}_i)\end{bmatrix}\right)+\lambda\boldsymbol{\alpha}^\top K\boldsymbol{\alpha}.\end{aligned}$$

4.4 Kernel Ridge Regression

4.4.1 Recap: Ridge Regression

Suppose the the observed centered data are $(\mathbf{x}_1,y_1),\dots,(\mathbf{x}_N,y_N)$ and assume for $i=1,\dots,N$

$$y_i=\boldsymbol{\theta}^\top\mathbf{x}_i+\epsilon_i.$$

By basic regression, we know the solution to LSE and MLE, i.e.

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\sum_{i=1}^{N}\left(y_i-\boldsymbol{\theta}^\top\mathbf{x}_i\right)^2\quad\text{and}\quad\max_{\boldsymbol{\theta}\in\mathbb{R}^p}L(\boldsymbol{\theta};\mathbb{X},\mathbb{Y})$$

are both $\hat{\boldsymbol{\theta}}=\left(\mathbb{X}^\top\mathbb{X}\right)^{-1}\mathbb{X}^\top\mathbb{Y}$. And if we add an L_2 penalty, the solution to

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\sum_{i=1}^{N}\left(y_i-\boldsymbol{\theta}^\top\mathbf{x}_i\right)^2+\lambda||\boldsymbol{\theta}||^2\right\}=\min_{\boldsymbol{\theta}\in\mathbb{R}^p}\left\{\left(\mathbb{Y}-\mathbb{X}\boldsymbol{\theta}\right)^\top\left(\mathbb{Y}-\mathbb{X}\boldsymbol{\theta}\right)+\lambda\boldsymbol{\theta}^\top\boldsymbol{\theta}\right\}$$

is the ridge regression coefficient, which is

$$\hat{\boldsymbol{\theta}}=\left(\mathbb{X}^\top\mathbb{X}+\lambda I_p\right)^{-1}\mathbb{X}^\top\mathbb{Y},$$

where $\mathbb{X}^\top\mathbb{X}$ is the sample covariance matrix of the data. By Woodbury matrix identity, we can further simplify it as

$$\hat{\boldsymbol{\theta}}=\mathbb{X}^\top\left(\mathbb{X}\mathbb{X}^\top+\lambda I_N\right)^{-1}\mathbb{Y}=\mathbb{X}^\top\left(G+\lambda I_N\right)^{-1}\mathbb{Y},$$

where G is the Gram matrix of the data. We may wonder, if we replace the Gram matrix by Kernel matrix, is there a corresponding kernel explanation for it as we have in kernel PCA? The answer is yes!

4.4.2 Mathematical Derivation

Since we are no longer looking for a linear function $\boldsymbol{\theta}^\top \mathbf{x}$, the model assumption needs a little modification. Specifying a kernel function k, assume

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \text{where} \quad f \in \mathscr{H}_k.$$

And our goal is to find

$$\hat{f}(\mathbf{x}) = \arg\min_{f \in \mathscr{H}_k} \left\{ \sum_{i=1}^{N} \Big[y_i - f(\mathbf{x}_i) \Big]^2 + \lambda ||f||^2_{\mathscr{H}_k} \right\}. \tag{4.4}$$

By Representer theorem, the solution is in the form of

$$\forall \mathbf{x} \in \mathbb{R}^p : \quad \hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i k(\mathbf{x}_i, \mathbf{x}) \quad \text{where} \quad \mathbf{x}_1, \ldots, \mathbf{x}_N \ \text{ are training data.}$$

Then this optimization (4.4) becomes linear optimization

$$\begin{aligned} \hat{\boldsymbol{\alpha}} = \begin{pmatrix} \hat{\alpha}_1 \\ \vdots \\ \hat{\alpha}_N \end{pmatrix} &\leftarrow \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \left\{ \sum_{i=1}^{N} \left[y_i - \sum_{j=1}^{N} \alpha_j k(\mathbf{x}_j, \mathbf{x}_i) \right]^2 + \lambda \boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \right\} \quad \text{where} \quad (K)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \\ &= \arg\min_{\boldsymbol{\alpha} \in \mathbb{R}^N} \left(\mathbb{Y} - K\boldsymbol{\alpha} \right)^\top \left(\mathbb{Y} - K\boldsymbol{\alpha} \right) + \lambda \boldsymbol{\alpha}^\top K \boldsymbol{\alpha}. \end{aligned}$$

Since this is a linear convex optimization, taking derivative with respect to $\boldsymbol{\alpha}$, we set

$$\frac{\partial}{\partial \boldsymbol{\alpha}} \Big[\left(\mathbb{Y} - K\boldsymbol{\alpha} \right)^\top \left(\mathbb{Y} - K\boldsymbol{\alpha} \right) + \lambda \boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \Big] \Big|_{\alpha = \hat{\alpha}} = 0 \quad \Rightarrow \quad K \left(\mathbb{Y} - K\hat{\boldsymbol{\alpha}} \right) = \lambda K \hat{\boldsymbol{\alpha}}.$$

Practically, the data and kernel function guarantees kernel Gram matrix K is invertible, which leads to

$$\hat{\boldsymbol{\alpha}} = \left(K + \lambda I \right)^{-1} \mathbb{Y}.$$

And plugging $\hat{\boldsymbol{\alpha}}$ into $\hat{f}$, we get

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i k(\mathbf{x}_i, \mathbf{x}) = \hat{\boldsymbol{\alpha}}^\top \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_N, \mathbf{x}) \end{bmatrix} = \mathbb{Y}^\top (K + \lambda I)^{-1} \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_N, \mathbf{x}) \end{bmatrix}.$$

Now we can compare the coefficients of kernel ridge $\hat{\boldsymbol{\alpha}}$ and conventional ridge $\hat{\boldsymbol{\theta}}$:

$$\hat{\boldsymbol{\alpha}} = \left(K + \lambda I \right)^{-1} \mathbb{Y} \quad \text{and} \quad \hat{\boldsymbol{\theta}} = \mathbb{X}^\top \left(G + \lambda I \right)^{-1} \mathbb{Y},$$

and the predicting function of kernel ridge $\hat{f}$ and conventional ridge $\hat{\boldsymbol{\theta}}^\top \mathbf{x}$:

$$\hat{f}(\mathbf{x}) = \mathbb{Y}^\top (K + \lambda I)^{-1} \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}) \\ \vdots \\ k(\mathbf{x}_N, \mathbf{x}) \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{\theta}} \mathbf{x} = \mathbb{Y} (G + \lambda I)^{-1} \mathbb{X} \mathbf{x} = \mathbb{Y}^\top (G + \lambda I)^{-1} \begin{bmatrix} \langle \mathbf{x}_1, \mathbf{x} \rangle \\ \vdots \\ \langle \mathbf{x}_N, \mathbf{x} \rangle \end{bmatrix},$$

so we can see the conventional ridge is just a special case of the kernel ridge.

4.4.3 Commonly Used Kernels

Commonly used kernel functions are the same as those used in kernel PCA:

- polynomial kernel,
- Gaussian kernel.

And in this section, we will focus on one special kernel and its extension: smoothing splines.

Smoothing Spline

The *spline* method could be also classified into a special type of kernel method. But to avoid computation complexity, this method is commonly used when we have only one predictor. When the number of predictors gets more, we use the *additive model* (to be introduced). Suppose the observed data are $(x_1, y_1), \ldots, (x_N, y_N)$, and we assume

$$y_i = f(x_i) + \epsilon_i \quad \text{where} \quad f \text{ is a smooth function with the existence of } f''.$$

In other word, we now specify the function space to be $\mathscr{H}_c = \{f : \exists f''\}$, and search a best predicting function with smallest loss in it, instead of specifying a kernel function k. The goal is to find

$$\hat{f}(x) \leftarrow \arg\min_{f \in \mathscr{H}_c} \left\{ \sum_{i=1}^{N} [y_i - f(x_i)]^2 + \lambda \int (f'')^2 \right\}. \tag{4.5}$$

We can view this $\mathscr{H}_c$ as a RKHS generated by a certain kernel function k. The only difference is we now directly specify this space.

Remark. Note that the linear function, such as $f = 2x + 1$, is not penalized by the integral norm. So, the whole function space could be separated into two part, one of which is linear and not penalized, the other is non-linear and penalized.

Also by the Representer theorem, the solution to (4.5) is in the form of

$$\hat{f}(x) = \sum_{j=1}^{N} \theta_j N_j(x) \quad \text{where} \quad \begin{cases} N_1(x) = 1, \\ N_2(x) = x, \\ N_{k+2}(x) = d_k(x) - d_{N-1}(x) \quad \forall k = 1, \ldots, N-2, \end{cases}$$
$$\text{where} \quad d_k(x) = \frac{(x - x_k)_+^3 + \cdots + (x - x_N)_+^3}{x - x_k}.$$

Using the same trick, since the Representer theorem guarantees the above form, the optimization problem

becomes linear and convex optimization:

$$\begin{aligned}\hat{\boldsymbol{\theta}} = \begin{pmatrix}\hat{\theta}_1 \\ \vdots \\ \hat{\theta}_N\end{pmatrix} &\leftarrow \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^N} \left\{ \sum_{i=1}^N \left[y_i - \sum_{j=1}^N \theta_j N_j(x_i) \right]^2 + \lambda \int (f'')^2 \right\} \\ &= \arg\min_{\boldsymbol{\theta}\in\mathbb{R}^N} \left\{ (\mathbb{Y} - N\boldsymbol{\theta})^\top (\mathbb{Y} - N\boldsymbol{\theta}) + \lambda \boldsymbol{\theta}^\top K \boldsymbol{\theta} \right\}, \\ &\qquad\qquad \text{where} \quad (N)_{ij} = N_j(x_i) \quad \text{and} \quad K_{ij} = \int N_i'' N_j''.\end{aligned}$$

Taking derivative and solving this optimization, we get

$$\hat{\boldsymbol{\theta}} = \left(N^\top N + \lambda K\right)^{-1} N\mathbb{Y}.$$

But let me repeat, we do not directly use smoothing spline in p predictors problem, since the computation is complex. Instead, we use the following additive model, which could be also considered as a generalization of splines, but with less computation complexity.

Additive Model

Now we suppose $\mathbf{x} \in \mathbb{R}^p$ and assume $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ follows

$$y_i = \alpha + f_1(x_{i1}) + \cdots + f_p(x_{ip}) + \epsilon_i \quad \text{where} \quad f_i \in \{f : \exists f'' \text{ with } f(0) = 0\}.$$

There are three points worth noticing:

- the intercept α is universal, i.e. does not vary among i;
- the requirement $f_i(0) = 0$ is to guarantee the model to be identifiable;
- for the estimation work, we further require $\mathbb{X}$ is full of column rank.

Our goal is find the best predicting function:

$$\hat{\alpha}, \hat{f}_i \leftarrow \arg\min_{\alpha, f_i} \left\{ \sum_{i=1}^N \left[y_i - \alpha - \sum_{j=1}^p f_j(x_{ij}) \right]^2 + \sum_{j=1}^p \lambda_j \int (f'')^2 \right\} \quad \text{where} \quad \forall j = 1, \ldots, p: \quad \sum_{i=1}^N f_j(x_{ij}) = 0.$$

Note the form of optimization separate the estimation of α and f_i. So, simply taking derivative w.r.t. α, we can get

$$\hat{\alpha} = \bar{y}.$$

Now, it remains to estimate the functions $f_1, \ldots, f_p$. In fact, it is hard to optimize simultaneously, so we introduce the following Backfitting algorithm.

Algorithm 8 (Backfitting)**.** Suppose we are estimating the additive model:

$$y_i = \alpha + f_1(x_{i1}) + \cdots + f_p(x_{ip}) + \epsilon_i \quad \text{where} \quad f_i \in \{f : \exists f'' \text{ with } f(0) = 0\}.$$

- **Initialize**: Set $\hat{\alpha} = \bar{y}$ and $\hat{f}_j(x) \equiv 0$ for $j = 1, \ldots, p$.

- **Iterate**: Cycle for $j = 1, \ldots, p, 1, \ldots, p, 1, \ldots, p$ until converge:

  (a) **Backfitting Step**: Fix j, do spline fit:

$$\hat{f}_j \leftarrow \text{Splines}\left[\left\{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik})\right\} \sim x_i, \quad \forall i = 1, \ldots, N\right].$$

  (b) **Centering**: Set the function to be

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N} \sum_{i=1}^{N} \hat{f}_j\left(x_{ij}\right).$$

Chapter 5

Classification

5.1 General Framework of Classification

Classification is another type of problem we will encounter very often, and it is closely related to machine learning and statistical theory. The problem is generally as follow.

Problem Setup (Classification)**.** Suppose the observed data $(\mathbf{x}_1, y_i), \ldots, (\mathbf{x}_N, y_N)$ are paired with

$$\text{information: } \mathbf{x}_i \in \mathcal{X} \quad \text{and} \quad \text{label: } y_i \in \{0, 1, \ldots, k\} =: \mathcal{Y}.$$

Our goal is to find a classifier function $f : \mathcal{X} \mapsto \mathcal{Y}$ such that

$$\hat{y} = f(\mathbf{x}) \quad \text{is a predictor for } y \text{ given a new observation } \mathbf{x}.$$

According to the number of labels, we have

$$\text{binary: } \mathcal{Y} = \{0, 1\} \quad \text{and} \quad \text{multi-class: } \mathcal{Y} = \{1, 2, \ldots, k\}.$$

Since different statisticians come up with different classifiers, we need a criterion concept to judge which classifier is better, which leads to the following definition.

Definition 6 (Loss function)**.** Given a classifier $f : \mathcal{X} \mapsto \mathcal{Y}$, a loss function $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is a performance criterion used as

$$l(y, \hat{y}), \quad \text{where } \hat{y} = \mathbf{x}.$$

For example, the very basic loss function is 0-1 loss function:$l(y, \hat{y}) = \mathbb{1}(y \neq \hat{y})$.

Definition 7 (Prediction risk)**.** Suppose $(\mathbf{X}_i, Y_i) \overset{\text{iid}}{\sim} p(\mathbf{x}, y)$ for $i = 1, \ldots, N$ is a random sample, $f : \mathcal{X} \mapsto \mathcal{Y}$ is a classifier, and $l : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ is a loss function. Define prediction risk $R(f)$ of the classifier f under the loss function l to be

$$R(f) = \mathbb{E}\Big[l\Big(y, f(\mathbf{x})\Big)\Big].$$

The Definition 7, prediction risk, is also called *Bayes risk*, or *expected classification error*.

5.2 Optimal Bayes Classifier

In this section, we assume $(\mathbf{X}_i, Y_i) \overset{\text{iid}}{\sim} p(\mathbf{x}, y)$ for $i = 1, \ldots, N$ whose joint distribution p is known, and derive the optimal classifier under 0-1 loss function, i.e. for all $i = 1, \ldots, N$,

$$(\mathbf{X}_i, Y_i) \overset{\text{iid}}{\sim} p(\mathbf{x}, y) \quad \text{and} \quad l\Big(y, f(\mathbf{x})\Big) = \mathbb{1}\Big[y = f(\mathbf{x})\Big],$$

our goal is to find the function $f^* : \mathcal{X} \mapsto \mathcal{Y}$ such that

$$f^* = \arg\min_f \mathbb{E}\Big[l\Big(y, f(\mathbf{x})\Big)\Big].$$

5.2.1 Classifier Derivation

Binary Case

For simplicity, we assume $\mathcal{Y} = \{0, 1\}$ and p is known. It follows that

$$\begin{aligned}
R(f) &= \mathbb{E}\Big[l\Big(y, f(\mathbf{x})\Big)\Big] \\
\text{(Conditioning)} \quad &= \mathbb{E}_{\mathbf{x}}\Big\{ \mathbb{E}_y\Big[l\Big(y, f(\mathbf{x})\Big) \;\Big|\; \mathbf{x}\Big]\Big\} \\
\text{(Y is discrete)} \quad &= \mathbb{E}_{\mathbf{x}}\Big\{ l\Big(1, f(\mathbf{x})\Big) \cdot \mathbb{P}(Y = 1 \mid \mathbf{x}) + l\Big(0, f(\mathbf{x})\Big) \cdot \mathbb{P}(Y = 0 \mid \mathbf{x})\Big\} \\
\text{(0-1 loss)} \quad &= \mathbb{E}_{\mathbf{x}}\Big\{ \mathbb{1}\Big(f(\mathbf{x}) \neq 1\Big) \cdot \mathbb{P}(Y = 1 \mid \mathbf{x}) + \mathbb{1}\Big(f(\mathbf{x}) \neq 0\Big) \cdot \mathbb{P}(Y = 0 \mid \mathbf{x})\Big\} \\
&= \mathbb{E}_{\mathbf{x}}\Big\{ \mathbb{1}\Big(f(\mathbf{x}) \neq 1\Big) \cdot \Big[2\mathbb{P}(Y = 1 \mid \mathbf{x}) - 1\Big] + \Big[1 - \mathbb{P}(Y = 1 \mid \mathbf{x})\Big]\Big\}. \qquad (5.1)
\end{aligned}$$

Recall that our goal is to minimize this risk function $R(f)$ by f, and we can ignore the latter term since

$$1 - \mathbb{P}(Y = 1 \mid \mathbf{x}) \quad \text{is not related to optimization.}$$

And also note that $\mathbb{1}\Big(f(\mathbf{x}) \neq 1\Big)$ only takes values 0 or 1, indicating that

$$\mathbb{1}\Big(f(\mathbf{x}) \neq 1\Big) \cdot \Big[2\mathbb{P}(Y = 1 \mid \mathbf{x}) - 1\Big] \begin{cases} \leq 0 & \text{if } \mathbb{P}(Y = 1 \mid \mathbf{x}) < 1/2, \\ \geq 0 & \text{if } \mathbb{P}(Y = 1 \mid \mathbf{x}) > 1/2. \end{cases}$$

Therefore, to minimize the loss, we should take

$$f^*(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbb{P}(Y = 1 \mid \mathbf{x}) < 1/2 \\ 1 & \text{if } \mathbb{P}(Y = 1 \mid \mathbf{x}) > 1/2 \end{cases} \quad \Longleftrightarrow \quad f^*(\mathbf{x}) = \arg\max_{k \in \{0,1\}} \mathbb{P}(Y = k \mid \mathbf{x}), \qquad (5.2)$$

which is also called *optimal Bayes classifier*.

Multi-class Case

Similar derivation leads to the optimal Bayes classifier of multi-class case, $\mathcal{Y} = \{1, \ldots, k\}$:

$$f^*(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k \mid \mathbf{x}). \qquad (5.3)$$

This means, under 0-1 loss and a known the joint density $p(\mathbf{x}, y)$, the optimal classifier is clear.

5.2.2 Risk Upper Bound

As stated, if the joint density $p(\mathbf{x}, y)$, then f^* derived above is the best classifier, minimizing the risk. However, the assumption of known $p(\mathbf{x}, y)$ may fail in most of the cases, meaning that we need to estimate the density first. The classification steps are (binary case)

(a) obtaining a good estimator $\hat{\eta}(\mathbf{x}) := \hat{P}(Y = 1 \mid \mathbf{x})$,

(b) plugging in and getting the classifier $\hat{f}(\mathbf{x}) = \mathbb{1}\Big[\hat{P}(Y = 1 \mid \mathbf{x}) \geq 1/2\Big]$.

If we can estimate the conditional mass function $\hat{\eta}(\mathbf{x}) = \hat{\mathbb{P}}(Y = 1 \mid \mathbf{x})$ well, then the empirical classifier risk $R(\hat{f})$ approaches the optimal one $R(f^*)$, otherwise $R(\hat{f}) > R(f^*)$. The next theorem guarantees an upper bound for this risk difference in the binary case.

Theorem 9. *For any estimated mass function $\hat{\eta}(\mathbf{x}) = \hat{\mathbb{P}}(Y = 1 \mid \mathbf{x})$, the risk of empirical classifier*

$$\hat{f}(\mathbf{x}) = \arg\max_{k \in \{0,1\}} \hat{\mathbb{P}}(Y = k \mid \mathbf{x})$$

has an upper bound

$$0 \leq R(\hat{f}) - R(f^*) \leq 2\mathbb{E}_{\mathbf{x}} \left| \eta(\mathbf{x}) - \hat{\eta}(\mathbf{x}) \right|, \quad \textit{where} \quad \eta(\mathbf{x}) = \mathbb{P}(Y = 1 \mid \mathbf{x}).$$

This theorem indicates that if we can learn the conditional mass function $\eta(\mathbf{x}) = \mathbb{P}(Y = 1 \mid \mathbf{x})$ very well by $\hat{\eta}(\mathbf{x})$, then the empirical classifier $\hat{f}$ could attain the same risk as the optimal classifier f^*.

Proof. Since f^* is the minimizer of the risk function,

$$0 \leq R(\hat{f}) - R(f^*)$$

holds naturally. It remains to show the other inequality. By equality (5.1), it follows that

$$\begin{aligned}
R(\hat{f}) - R(f^*) &= \mathbb{E}_{\mathbf{x}}\Big\{ \mathbb{1}\Big(\hat{f}(\mathbf{x}) \neq 1\Big) \cdot \Big[2\mathbb{P}(Y = 1 \mid \mathbf{x}) - 1\Big] + \Big[1 - \mathbb{P}(Y = 1 \mid \mathbf{x})\Big] \Big\} \\
&\qquad - \mathbb{E}_{\mathbf{x}}\Big\{ \mathbb{1}\Big(f^*(\mathbf{x}) \neq 1\Big) \cdot \Big[2\mathbb{P}(Y = 1 \mid \mathbf{x}) - 1\Big] + \Big[1 - \mathbb{P}(Y = 1 \mid \mathbf{x})\Big] \Big\} \\
&= \mathbb{E}_{\mathbf{x}}\Big\{ \mathbb{1}\Big(\hat{f}(\mathbf{x}) \neq 1\Big) \cdot \Big[2\mathbb{P}(Y = 1 \mid \mathbf{x}) - 1\Big] - \mathbb{1}\Big(f^*(\mathbf{x}) \neq 1\Big) \cdot \Big[2\mathbb{P}(Y = 1 \mid \mathbf{x}) - 1\Big] \Big\} \\
&= \mathbb{E}_{\mathbf{x}}\Big\{ \Big[\mathbb{1}\Big(\hat{f}(\mathbf{x}) \neq 1\Big) - \mathbb{1}\Big(f^*(\mathbf{x}) \neq 1\Big)\Big] \cdot \Big[2\mathbb{P}(Y = 1 \mid \mathbf{x}) - 1\Big] \Big\} \\
&= 2\mathbb{E}_{\mathbf{x}} \left\{ \mathbb{1}\Big(\hat{f}(\mathbf{x}) \neq f^*(\mathbf{x})\Big) \cdot \Big[\eta(\mathbf{x}) - \frac{1}{2}\Big] \right\}. \qquad (5.4)
\end{aligned}$$

Note that when $\hat{f}(\mathbf{x}) \neq f(\mathbf{x})$, i.e. the classification results are different, one probability is greater than $1/2$ and the other is less than $1/2$, meaning that

$$\Big(\hat{\eta}(\mathbf{x}) - \frac{1}{2}\Big) \cdot \Big(\eta(\mathbf{x}) - \frac{1}{2}\Big) \leq 0.$$

And it follows that

$$|\eta(\mathbf{x}) - \hat{\eta}(\mathbf{x})| = \left|\eta(\mathbf{x}) - \frac{1}{2}\right| + \left|\hat{\eta}(\mathbf{x}) + \frac{1}{2}\right| \geq \left|\eta(\mathbf{x}) - \frac{1}{2}\right|.$$

Using this identity in (2), we further get

$$\begin{aligned} R(\hat{f}) - R(f^*) &= 2\mathbb{E}_{\mathbf{x}} \left\{ \mathbb{1}\Big(\hat{f}(\mathbf{x}) \neq f^*(\mathbf{x})\Big) \cdot \left[\eta(\mathbf{x}) - \frac{1}{2}\right] \right\} \\ &\leq 2\mathbb{E}_{\mathbf{x}} \left|\eta(\mathbf{x}) - \hat{\eta}(\mathbf{x})\right|, \end{aligned}$$

concluding the result. □

5.2.3 Conditional Mass Estimation

By the above argument, our goal becomes estimating the conditional mass function $p(y \mid \mathbf{x})$, more specifically in the binary case, estimating $\eta(\mathbf{x}) = \mathbb{P}(Y = 1 \mid \mathbf{x})$. If this is achieved with accuracy, we can get nearly the optimal Bayes estimator.

Note that all methods we discuss in this note are model-based classification (data-driven to be introduced), and model-based can be divided in to two ideas:

- generative model: given joint distribution $p(\mathbf{x}, y)$ and compute $p(y \mid \mathbf{x})$ such as
  (a) linear/quadratic discriminant analysis (LDA/QDA);
  (b) naive Bayes classifier.
- conditional model: given conditional distribution $p(y \mid \mathbf{x})$ such as
  (a) logistic regression.

The goal is the same: using conditional distribution $p(y \mid \mathbf{x})$ to derive a good classifier $f(\mathbf{x})$. Now, let's focus on a generative model, LDA and QDA.

5.3 Binary Case LDA and QDA

For simplicity, let's start with binary case.

Assumption 10 (LDA)**.** Suppose $(X_i, Y_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Y \sim \text{Ber}(\pi) \quad \text{and} \quad \begin{cases} X \mid Y = 0 \sim \mathbf{N}_p(\boldsymbol{\mu}_0, \Sigma) \\ X \mid Y = 1 \sim \mathbf{N}_p(\boldsymbol{\mu}_1, \Sigma) \end{cases}, \quad \text{where} \quad \boldsymbol{\mu}_0 \neq \boldsymbol{\mu}_1 \quad \text{but the same variance.}$$

Further assume all parameters, $\boldsymbol{\theta} = (\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \Sigma, \pi)$, are known (or already estimated). We shall see the linear part in the following derivation. Recall our goal is to calculate $\mathbb{P}(Y = 1 \mid \mathbf{x})$, which follows from

$$\begin{aligned} \mathbb{P}(Y = 1 \mid \mathbf{x}) &= \frac{\mathbb{P}(\mathbf{x} \mid Y = 1)\mathbb{P}(Y = 1)}{\mathbb{P}(\mathbf{x} \mid Y = 0)\mathbb{P}(Y = 0) + \mathbb{P}(\mathbf{x} \mid Y = 1)\mathbb{P}(Y = 1)} \\ &= \frac{1}{1 + \frac{\mathbb{P}(\mathbf{x}|Y=0)\mathbb{P}(Y=0)}{\mathbb{P}(\mathbf{x}|Y=1)\mathbb{P}(Y=1)}} \\ \Big(\mathbf{x} \mid Y = k \sim \mathbf{N}(\boldsymbol{\mu}_k, \Sigma)\Big) \quad &= \frac{1}{1 + \frac{1-\pi}{\pi} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_0) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^\top \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_1)\right\}} \\ &= \frac{1}{1 + \exp(-\beta^\top \mathbf{x} - \gamma)}, \end{aligned} \tag{5.5}$$

where

$$\gamma = -\frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \Sigma^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0) + \log\frac{1-\pi}{\pi} \quad \text{and} \quad \beta = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0). \tag{5.6}$$

Remark. One interesting thing is, by (5.5),

$$\log\left[\frac{\mathbb{P}(Y=1 \mid \mathbf{x})}{\mathbb{P}(Y=0 \mid \mathbf{x})}\right] = \gamma + \beta^\top \mathbf{x},$$

which takes the form of logistic regression even if we are working on a different model assumption.

Now, we successfully derived $\eta(\mathbf{x}) = \mathbb{P}(Y=1 \mid \mathbf{x})$. Therefore, by the form of optimal Bayes classifier, it follows that

$$f^*(\mathbf{x}) = \mathbb{1}\left[\mathbb{P}(Y=1 \mid \mathbf{x}) \geq \frac{1}{2}\right] = \mathbb{1}\left(\beta^\top \mathbf{x} + \gamma \geq 0\right), \tag{5.7}$$

where

$$\gamma = -\frac{1}{2}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \Sigma^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_0) + \log\frac{1-\pi}{\pi} \quad \text{and} \quad \beta = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0).$$

The form (5.7) is the reason why it is called LDA, since we are in fact judging by a linear function of $\mathbf{x}$.

5.3.1 Parameter Estimation

As in §5.2, the form (5.7) is just the optimal classifier f^* when all parameters are known. Since in practical, it is commonly impossible to known the true parameters, we need to estimate them before the classification.

In essence, this is Gaussian mixture model, i.e., the observed data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$ are a random sample from $p(\mathbf{x}, y)$. It follows that

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta} \mid \mathbb{X}, \mathbf{y}) &= \prod_{i=1}^N p(y_i) p(\mathbf{x}_i \mid y_i) \\ &= \prod_{i=1}^N \prod_{k=0}^1 p(\mathbf{x}_i \mid y_i = k)^{\mathbb{1}(y_i=k)} p(y_i = k)^{\mathbb{1}(y_i=k)}.\end{aligned}$$

Then log-likelihood function follows that

$$\begin{aligned}l(\boldsymbol{\theta} \mid \mathbb{X}, \mathbf{y}) &= \sum_{i=1}^N \sum_{k=0}^1 \Big\{ \mathbb{1}(y_i = k) \log p(\mathbf{x}_i \mid y_i = k) + \mathbb{1}(y_i = k) \log p(y_i = k) \Big\} \\ &= \sum_{i:y_i=0} \log p(\mathbf{x}_i \mid y_i = 0) + \sum_{i:y_i=1} \log p(\mathbf{x}_i \mid y_i = 1) + \sum_{i:y_i=0} \log p(y_i = 0) + \sum_{i:y_i=1} \log p(y_i = 1) \\ &= \sum_{i:y_i=0} \log \mathbf{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_0, \Sigma) + \sum_{i:y_i=1} \log \mathbf{N}(\mathbf{x}_i \mid \boldsymbol{\mu}_1, \Sigma) + \sum_{i:y_i=0} \pi_0 + \sum_{i:y_i=0} \pi_1.\end{aligned}$$

Since the optimization of each parameter is separated, simple algebra leads to the MLE:

$$\hat{\pi} = \frac{\sum_{i=1}^N \mathbb{1}(y_i = 1)}{N}, \quad \hat{\mu}_k = \frac{\sum_{i=1}^N \mathbf{x}_i \mathbb{1}(y_i = k)}{\sum_{i=1}^N \mathbb{1}(y_i = k)}, \quad \text{and} \quad \hat{\Sigma} = \frac{1}{N} \sum_{k=0}^1 \sum_{i:y_i=k} (\mathbf{x}_i - \hat{\mu}_k)(\mathbf{x}_i - \hat{\mu}_k)^\top.$$

Plugging this estimation to the optimal classifier (5.7), we are done in LDA.

5.3.2 Quadratic Discriminant Analysis

QDA is just a little modification on LDA, assuming that different clusters have different variance.

Assumption 11 (QDA)**.** Suppose $(X_i, Y_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Y \sim \text{Ber}(\pi) \quad \text{and} \quad \begin{cases} X \mid Y = 0 \sim \mathbf{N}_p(\boldsymbol{\mu}_0, \Sigma_1) \\ X \mid Y = 1 \sim \mathbf{N}_p(\boldsymbol{\mu}_1, \Sigma_2) \end{cases}, \quad \text{where} \quad \boldsymbol{\mu}_0 \neq \boldsymbol{\mu}_1 \quad \text{and} \quad \Sigma_1 \neq \Sigma_2.$$

Exactly the same derivation leads to

$$\begin{aligned} \mathbb{P}(Y = 1 \mid \mathbf{x}) &= \frac{\mathbb{P}(\mathbf{x} \mid Y = 1)\mathbb{P}(Y = 1)}{\mathbb{P}(\mathbf{x} \mid Y = 0)\mathbb{P}(Y = 0) + \mathbb{P}(\mathbf{x} \mid Y = 1)\mathbb{P}(Y = 1)} \\ &= \frac{1}{1 + \frac{\mathbb{P}(\mathbf{x}|Y=0)\mathbb{P}(Y=0)}{\mathbb{P}(\mathbf{x}|Y=1)\mathbb{P}(Y=1)}}. \end{aligned}$$

By the optimal classification rule, if $\mathbb{P}(Y = 1 \mid \mathbf{x}) > 1/2$, we choose $k = 1$, and this condition is equivalent to

$$\mathbb{P}(Y = 1 \mid \mathbf{x}) > \frac{1}{2} \quad \Longleftrightarrow \quad \log\left[\frac{\mathbb{P}(\mathbf{x} \mid Y = 0)\mathbb{P}(Y = 0)}{\mathbb{P}(\mathbf{x} \mid Y = 1)\mathbb{P}(Y = 1)}\right] > 0.$$

And it follows that

$$\begin{aligned} &\log\left[\frac{\mathbb{P}(Y = 1 \mid \mathbf{x}) \cdot \mathbb{P}(Y = 1)}{\mathbb{P}(Y = 0 \mid \mathbf{x}) \cdot \mathbb{P}(Y = 0)}\right] \\ &= \log\left[\frac{(2\pi)^{-p/2}|\Sigma_0|^{-1/2}}{(2\pi)^{-p/2}|\Sigma_1|^{-1/2}} \cdot \frac{\pi_1}{\pi_0}\right] - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_0)^\top |\Sigma_0|^{-1}(\mathbf{x} - \boldsymbol{\mu}_0) + \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_1)^\top |\Sigma_1|^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) \\ &= \delta_1(\mathbf{x}) - \delta_2(\mathbf{x}), \end{aligned}$$

where for $k = 0, 1$,

$$\delta_k(\mathbf{x}) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top |\Sigma_k|^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) - \log \pi_k$$

Therefore, if $\delta_1(\mathbf{x}) > \delta_2(\mathbf{x})$, we choose $k = 1$; otherwise, we choose $k = 0$. Since the two δ functions depend on a quadratic form of $\mathbf{x}$, it is called QDA.

5.4 Multi-class LDA and QDA

Now we can consider the multi-class LDA or QDA setting.

Assumption 12 (LDA)**.** Suppose $(X_i, Y_i), i = 1, \ldots, N$ with $\mathcal{Y} = \{1, 2, \ldots, K\}$ is a random sample from the following model:

$$Y \sim \text{Categorical}(\boldsymbol{\pi}) \quad \text{and} \quad X \mid Y = k \sim \mathbf{N}_p(\boldsymbol{\mu}_k, \Sigma_k) \quad \text{with} \quad \boldsymbol{\mu}_j \neq \boldsymbol{\mu}_k.$$

If we further assume $\Sigma_1 = \cdots = \Sigma_K$, then it is the multi-class LDA, otherwise is the QDA.

5.4.1 Multi-class LDA

Since it is multi-class problem, we no longer use (5.5) but rather compute for $1 \leq k \neq l \leq K$,

$$\begin{aligned}\log\left[\frac{\mathbb{P}(Y=k \mid \mathbf{x})}{\mathbb{P}(Y=l \mid \mathbf{x})}\right] &= \log\left[\frac{\mathbb{P}(\mathbf{x} \mid Y=k)\mathbb{P}(Y=k)}{\mathbb{P}(\mathbf{x} \mid Y=l)\mathbb{P}(Y=l)}\right] \\ &= \delta_k(\mathbf{x}) - \delta_l(\mathbf{x}),\end{aligned} \tag{5.8}$$

where for all $k = 1, 2, \ldots, K$,

$$\delta_k(\mathbf{x}) = \mathbf{x}^\top \Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k^\top \Sigma^{-1}\boldsymbol{\mu}_k + \log \pi_k.$$

Recall that under 0-1 loss, the optimal classifier is

$$f^*(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y=k \mid \mathbf{x}).$$

Therefore by log odds (5.8), we can tell

$$f^*(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y=k \mid \mathbf{x}) = \arg\max_{1 \leq k \leq K} \delta_k(\mathbf{x}).$$

Parameter Estimation

Similarly, we need to estimate all parameters $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K, \Sigma)$ since in practice, they are not known. Suppose we already have the training data $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$. The same trick (weighted Gaussian estimation) leads to

$$\hat{\pi}_k = \frac{\sum_{i=1}^N \mathbb{1}(y_i = k)}{N}, \quad \hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^N \mathbb{1}(y_i = k)\mathbf{x}_i}{\sum_{i=1}^N \mathbb{1}(y_i = k)}, \quad \text{and} \quad \hat{\Sigma} = \frac{1}{N}\sum_{k=1}^K \sum_{i:y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top.$$

5.4.2 Multi-class QDA

Using the same idea of (5.8), a similar derivation leads to

$$\log\left[\frac{\mathbb{P}(Y=k \mid \mathbf{x})}{\mathbb{P}(Y=l \mid \mathbf{x})}\right] = \delta_k(\mathbf{x}) - \delta_l(\mathbf{x}),$$

where for all $k = 1, \ldots, K$,

$$\delta_k(\mathbf{x}) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\mathbf{x} - \boldsymbol{\mu}_k) + \log \pi_k.$$

And also the optimal Bayes classifier under 0-1 loss is

$$f^*(\mathbf{x}) = \arg\max_{1 \leq k \leq K} \mathbb{P}(Y=k \mid \mathbf{x}) = \arg\max_{1 \leq k \leq K} \delta_k(\mathbf{x}).$$

Parameter estimation

If parameters are all unknown, the weighted Gaussian estimation leads to

$$\hat{\pi}_k = \frac{\sum_{i=1}^N \mathbb{1}(y_i = k)}{N}, \quad \hat{\boldsymbol{\mu}}_k = \frac{\sum_{i=1}^N \mathbb{1}(y_i = k)\mathbf{x}_i}{\sum_{i=1}^N \mathbb{1}(y_i = k)}, \quad \text{and} \quad \hat{\Sigma}_k = \frac{1}{\sum_{i=1}^N \mathbb{1}(y_i = k)} \sum_{i:y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top.$$

Remark. In the LDA/QDA setting, if the label y's are not observed, then this model is also estimable (by EM) and becomes a famous clustering model, the Gaussian mixture model.

5.5 Naive Bayes

In the previous note, LDA and QDA are introduced to deal with those data such that $\mathbf{x} \in \mathbb{R}^p$ and y is the label. What if the information $\mathbf{x}$'s are categorical instead of continuous? In this case, the *Naive Bayes classifier* (multiview model) is introduced. The model assumption is

Assumption 13 (Naive Bayes)**.** Suppose $(X_i, Y_i), i = 1, \dots, N$ is a random sample from the following model:

$$Y \sim \text{Categorical}(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{Y} = \{1, 2, \dots, K\}$$

and for $\mathbf{X} \in \mathbb{R}^p$,

$$\mathbf{X} = \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \quad \text{with each component } \; X_i \in \{1, \dots, L\} \; \text{ being } L \text{ categorical.}$$

Furthermore, we assume the conditional independence between components of $\mathbf{X}$,

$$X_i \mid Y \perp\!\!\!\perp X_j \mid Y, \quad \forall\, 1 \le i \ne j \le p,$$

and the conditional distribution $p_{ljk} := \mathbb{P}(X_j = l \mid Y = k)$

$$X_j \mid Y = k \sim \text{Categorical}_L(p_{ljk}), \quad \begin{cases} \forall \text{ categories of } X_j\text{:} \;\; l = 1, \dots, L. \\ \forall \text{ components of } \mathbf{X}\text{:} \;\; j = 1, \dots, p. \\ \forall \text{ categories of } Y\text{:} \;\; k = 1, \dots, K. \end{cases}$$

Remark. If Y is further latent, this assumption is called latent class model, which is another famous model for clustering.

Similar to the previous structure, two main parts are included. Firstly, we suppose all parameters are known, and solve the optimal Bayes classifier. Second, we estimate all parameters since, in practical, the parameters are not known.

5.5.1 Optimal Bayes Classifier

In the last note, we have derived that under 0-1 loss function, the optimal Bayes classifier is

$$f^*(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k \mid \mathbf{x}).$$

Therefore, it remains to compare $\mathbb{P}(Y = k \mid \mathbf{x})$ for different $k = 1, \dots, K$. Under the model assumption, it follows that

$$\mathbb{P}(Y = k \mid \mathbf{x}) = \frac{\pi_k \mathbb{P}(\mathbf{x} \mid Y = k)}{\sum_{k=1}^K \pi_k \mathbb{P}(\mathbf{x} \mid Y = k)},$$

where we can further compute by the conditional independence assumption that

$$\begin{aligned}\mathbb{P}(\mathbf{x} \mid Y=k) &= \prod_{j=1}^{p} \mathbb{P}(X_j = x_j \mid Y=k) \\ &= \prod_{j=1}^{p}\prod_{l=1}^{L} \mathbb{P}(X_j = l \mid Y=k)^{\mathbb{1}(X_j=l)} \\ &= \prod_{j=1}^{p}\prod_{l=1}^{L} p_{ljk}.\end{aligned}$$

Therefore, it can be simplified that

$$\begin{aligned}\mathbb{P}(Y=k \mid \mathbf{x}) &= \frac{\exp\left\{\log \pi_k + \sum_{j=1}^{p}\sum_{l=1}^{L} \mathbb{1}(X_j=l)\log p_{ljk}\right\}\}}{\sum_{k=1}^{K}\exp\left\{\log \pi_k + \sum_{j=1}^{p}\sum_{l=1}^{L} \mathbb{1}(X_j=l)\log p_{ljk}\right\}} \\ &=: \frac{\exp\left(\boldsymbol{\beta}_k^\top \boldsymbol{z}\right)}{\sum_{k=1}^{K}\exp\left(\boldsymbol{\beta}_k^\top \boldsymbol{z}\right)},\end{aligned}$$

where we define for $k = 1, \ldots, K$,

$$\boldsymbol{\beta}_k^\top = \begin{pmatrix}\log \pi_k & \log p_{11k} & \cdots & \log p_{L1k} & \cdots \log p_{Lpk}\end{pmatrix}_{1+pL}$$

and

$$\boldsymbol{z} = \begin{bmatrix}1 & \mathbb{1}(X_1=1) & \cdots & \mathbb{1}(X_1=L) & \cdots & \mathbb{1}(X_p=L)_{1+pL}.\end{bmatrix}$$

Note that if all parameters $\boldsymbol{\theta} = (\boldsymbol{\pi},\ p_{ljk})$ are known, then the optimal Bayes classifier is clear. Therefore, our first part in done. It remains to estimate all parameters.

5.5.2 Parameter Estimation

Suppose we have the training sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$. The estimation method is MLE. The derivation is left as an exercise, and the result is for every l, j, k,

$$\hat{\pi}_k = \frac{\sum_{i=1}^{N}\mathbb{1}(y_i=k)}{N} \quad \text{and} \quad \hat{p}_{ljk} = \frac{\sum_{i=1}^{N}\mathbb{1}(y_i=k, x_{ij}=l)}{\sum_{i=1}^{N}\mathbb{1}(y_i=k)}.$$

5.5.3 Compound Model

Now we know LDA/QDA is for continuous $\mathbf{x}$ classification, and naive Bayes is for discrete $\mathbf{x}$. What if the information $\mathbf{x}$ is partly continuous and partly discrete? In this case, the model assumption needs a little modification.

Assumption 14. Suppose $(X_i, Y_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Y \sim \text{Categorical}(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{Y} = \{1, 2, \ldots, K\}$$

and for $\mathbf{X} \in \mathbb{R}^p$,

$$\mathbf{X} = \begin{pmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{pmatrix} \quad \text{where } \mathbf{X}_1 \text{ is all continuous and } \mathbf{X}_2 \text{ is all discrete.}$$

Furthermore, we assume the conditional independence between components of $\mathbf{X}$,

$$X_i \mid Y \perp\!\!\!\perp X_j \mid Y, \quad \forall\, 1 \leq i \neq j \leq p.$$

Note that our goal is to find the optimal Bayes classifier:

$$f^*(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \mathbb{P}(Y = k \mid \mathbf{x}) = \arg\max_{k \in \mathcal{Y}} \pi_k \mathbb{P}(\mathbf{x} \mid Y = k).$$

Since we have the conditional independence assumption, it follows that

$$\mathbb{P}(\mathbf{x} \mid Y = k) = \mathbb{P}(X_1 = x_1 \mid Y = k) \cdots \mathbb{P}(X_p = x_p \mid Y = k).$$

We can split the problem into every component (or continuous part and discrete part), and solve the classifier as we did in previous single step.

In the whole model-based classification setting, recall that the optimal Bayes classifier under 0-1 loss function is

$$f^*(\mathbf{x}) = \arg\max_{k\in\mathcal{Y}} \mathbb{P}(Y = k \mid \mathbf{x}).$$

We have two types of idea to estimate the conditional mass function $p(y \mid \mathbf{x})$:

- generative model: given joint distribution $p(\mathbf{x}, y)$ and compute $p(y \mid \mathbf{x})$ such as
  (a) linear/quadratic discriminant analysis (LDA/QDA);
  (b) naive Bayes classifier.
- conditional model: given conditional distribution $p(y \mid \mathbf{x})$ such as
  (a) logistic regression.

Remark. The comparison between LDA/QDA/naive Bayes v.s. logistic regression could be found in §4.4.5 of *Elements of Statistical Learning*.

In this note, the followings are covered:

1. the logistic regression model and its estimation;
2. the kernel logistic regression model and its estimation;
3. the *generalized additive model* (GAM) and its estimation;
4. the extension of logistic regression to multi-class;
5. the *projection pursuit*.

5.6 Logistic Regression

Let's start with the most simple binary logistic regression.

Assumption 15 (Logistic regression)**.** Suppose $(\mathbf{x}_i, Y_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Y_i \overset{\text{ind}}{\sim} \text{Ber}(\pi_i) \text{ with } \log\left(\frac{\pi_i}{1-\pi_i}\right) = \boldsymbol{\theta}^\top \mathbf{x}_i,$$

where

$$\boldsymbol{\theta} = \begin{pmatrix}\theta_0 & \theta_1 & \cdots & \theta_p\end{pmatrix}^\top \text{ and } \mathbf{x}_i = \begin{pmatrix}1 & x_{i1} & \cdots & x_{ip}\end{pmatrix}.$$

Or equivalently, we can assume

$$Y_i \overset{\text{ind}}{\sim} \text{Ber}(\pi_i) \text{ where } \pi_i = \frac{\exp\left(\boldsymbol{\theta}^\top \mathbf{x}_i\right)}{1 + \exp\left(\boldsymbol{\theta}^\top \mathbf{x}_i\right)}.$$

Since the model is clear, we can directly use MLE to estimate the coefficients of regression.

5.6.1 Parameter Estimation

Note that we are treating $\mathbf{x}$'s as fixed number rather than random variables. Then we can write the likelihood function of this sample

$$\mathcal{L}(\boldsymbol{\theta}) = \prod_{i=1}^{N} p(y_i \mid \mathbf{x}_i) = \prod_{i=1}^{N} \pi_i^{y_i}[1-\pi_i]^{1-y_i},$$

and the log-likelihood function is

$$l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \Big\{ y_i \boldsymbol{\theta}^\top \mathbf{x}_i - \log\left[1 + \exp\left(\boldsymbol{\theta}^\top \mathbf{x}_i\right)\right] \Big\}. \tag{5.9}$$

We can view the negative log-likelihood (5.9) as the loss function (not 0-1 loss now). Simple algebra leads to

$$\frac{\partial l(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{i=1}^{N} y_i \mathbf{x}_i - \frac{\exp\left(\boldsymbol{\theta}^\top \mathbf{x}_i\right)}{1 + \exp\left(\boldsymbol{\theta}^\top \mathbf{x}_i\right)} \mathbf{x}_i = \sum_{i=1}^{N} (y_i - \pi_i)\mathbf{x}_i.$$

We all know that this MLE has no closed form solution of $\hat{\boldsymbol{\theta}}$. A common way to solve it is *iterative reweighted least square* (IRLS). The derivation can be referred in Homework 4 of STATS 600, and the algorithm is as follow.

Algorithm 9 (IRLS for logistic regression)**.** Given the design matrix $\mathbb{X}$ and the initial value $\boldsymbol{\theta}_0$.

(a) Set for all $i = 1, \ldots, N$:

$$\pi_i^{(t)} = \frac{\exp\left(\boldsymbol{\theta}_{(t)}^\top \mathbf{x}_i\right)}{1 + \exp\left(\boldsymbol{\theta}_{(t)}^\top \mathbf{x}_i\right)}, \quad v_i^{(t)} = \pi_i^{(t)}\left[1 - \pi_i^{(t)}\right] \quad w_i^{(t)} = \frac{1}{\pi_i^{(t)}\left[1 - \pi_i^{(t)}\right]},$$

and

$$\boldsymbol{v}^{(t)} = \begin{pmatrix} v_1^{(t)} \\ \vdots \\ v_N^{(t)} \end{pmatrix}, \text{ and } W^{(t)} = \begin{pmatrix} w_1^{(t)} & & \\ & \ddots & \\ & & w_N^{(t)} \end{pmatrix}.$$

(b) Set $\mathbf{Z}^{(t)} = W^{(t)}\left[\mathbb{X}\boldsymbol{\theta}^{(t)} + \boldsymbol{v}^{(t)}\right]$, and solve for $\boldsymbol{\theta}^{(t+1)}$ in

$$\mathbb{X}^\top W^{(t)} \mathbb{X} \boldsymbol{\theta}^{(t+1)} = \mathbb{X}^\top \boldsymbol{Z}^{(t)}.$$

(c) Repeat (a) and (b) until convergence.

5.6.2 Revisited IRLS

When the scale of data is large, the IRLS is time-consuming. We here introduce two ways to enhance the computation speed.

Gradient Ascent Updates

This method saves us from computing the second order derivative. We first choose a step size ρ, and set

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \rho \sum_{i=1}^{N} \left(y_i - \pi_i^{(t)} \right) \mathbf{x}_i.$$

Stochastic Gradient Ascent Algorithm

If the scale is really large, even the above updates converge slow. In this case, we previously choose a sequence of step size $\rho^{(t)}$ with

$$\sum_{t=0}^{\infty} \rho^{(t)} = \infty \text{ and } \sum_{t=0}^{\infty} \left[\rho^{(t)} \right]^2 < \infty$$

and random subset $I^{(t)}$. Then we update by

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \rho^{(t)} \sum_{i \in I^{(t)}} \left(y_i - \pi_i^{(t)} \right) \mathbf{x}_i.$$

For theoretical details of this algorithm, refer *Robbins-Monro* algorithm published on *Annals of Statistics*, 1950.

5.7 Kernel Logistic Regression

Like the kernel ridge, prespecifying a kernel function $k : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$, the kernel logistic regression modifies the assumption. (Please refer Note 6.2 - Kernel Ridge for similarity)

Assumption 16 (Kernel logistic regression)**.** Suppose $(\mathbf{x}_i, Y_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Y_i \overset{\text{ind}}{\sim} \text{Ber}(\pi_i) \text{ with } \log\left(\frac{\pi_i}{1-\pi_i}\right) = f(\mathbf{x}_i),$$

where $f \in \mathscr{H}_k$. Or equivalently, we can assume

$$Y_i \overset{\text{ind}}{\sim} \text{Ber}(\pi_i) \text{ where } \pi_i = \frac{\exp\left[f(\mathbf{x}_i)\right]}{1 + \exp\left[f(\mathbf{x}_i)\right]}.$$

5.7.1 Estimation

Our optimization goal naturally changes to (unknown parameter now is the f)

$$\hat{f}(\mathbf{x}) = \arg\min_{f \in \mathscr{H}_k} \Big\{ -l(f \mid \mathbb{X}, \mathbf{Y}) + \frac{\lambda}{2} ||f||^2_{\mathscr{H}_k} \Big\},$$

where we are choosing $-l(f)$ as the loss function. By the same derivation as in (5.9), we are doing

$$\hat{f}(\mathbf{x}) = \arg\min_{f \in \mathscr{H}_k} \left(\sum_{i=1}^{N} \Big\{ -y_i f(\mathbf{x}_i) + \log\left[1 + e^{f(\mathbf{x}_i)}\right] \Big\} + \frac{\lambda}{2} ||f||^2_{\mathscr{H}_k} \right). \tag{5.10}$$

By Representer theorem, the solution $\hat{f}$ have the following form:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \hat{\alpha}_i k(\mathbf{x}_i, \mathbf{x}) \;\text{ with }\; ||\hat{f}||^2_{\mathscr{H}_k} = \sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j).$$

Therefore we optimize

$$\begin{aligned}\hat{\boldsymbol{\alpha}} = \begin{pmatrix}\hat{\alpha}_1 \\ \vdots \\ \hat{\alpha}_N\end{pmatrix} &\leftarrow \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \left(\sum_{i=1}^{N} \Big\{ -y_i f(\mathbf{x}_i) + \log\Big[1 + e^{f(\mathbf{x}_i)}\Big] \Big\} + \frac{\lambda}{2}\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \right) \\ &= \arg\max_{\boldsymbol{\alpha}\in\mathbb{R}^N} \left(\sum_{i=1}^{N} \Big\{ y_i f(\mathbf{x}_i) + \log\Big[1 + e^{f(\mathbf{x}_i)}\Big] \Big\} - \frac{\lambda}{2}\boldsymbol{\alpha}^\top K \boldsymbol{\alpha} \right), \;\text{ where }\; (K)_{ij} = k(\mathbf{x}_i, \mathbf{x}_j).\end{aligned}$$

This optimization can be done using the IRLS algorithm of the typical logistic regression. Note that

$$\frac{\partial}{\partial \boldsymbol{\alpha}}\text{object} = K^\top(\mathbb{Y} - \boldsymbol{\mu}) - \lambda K\boldsymbol{\alpha}, \;\text{ where }\; \mathbb{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_N \end{pmatrix} \;\text{ and }\; \boldsymbol{\mu} = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_N \end{pmatrix} = \begin{pmatrix} \frac{e^{f(\mathbf{x}_1)}}{1+e^{f(\mathbf{x}_1)}} \\ \vdots \\ \frac{e^{f(\mathbf{x}_N)}}{1+e^{f(\mathbf{x}_N)}} \end{pmatrix},$$

and the Hessian matrix is

$$\frac{\partial^2}{\partial \boldsymbol{\alpha}\partial \boldsymbol{\alpha}^\top}\text{object} = -K^\top W K - \lambda K, \;\text{ where }\; W = \begin{pmatrix} \mu_1(1-\mu_1) & & \\ & \vdots & \\ & & \mu_N(1-\mu_N) \end{pmatrix}.$$

Therefore, we can conclude the IRLS algorithm for kernel logistic regression.

Algorithm 10 (IRLS for kernel logistic regression)**.** Given the design matrix $\mathbb{X}$ and a initial value $\boldsymbol{\alpha}^{(0)}$.

(a) Compute $\boldsymbol{\mu}^{(t)}, W^{(t)}$ by plugging $f^{(t)} = K\boldsymbol{\alpha}^{(t)}$ and set $\boldsymbol{Z}^{(t)} = K\boldsymbol{\alpha}^{(t)} + \big[W^{(t)}\big]^{-1}(\mathbf{Y} - \boldsymbol{\mu}^{(t)})$.

(b) Update coefficients vectors $\boldsymbol{\alpha}$ by

$$\boldsymbol{\alpha}^{(t+1)} = \Big(K^\top W^{(t)} K + \lambda K\Big)^{-1} K^\top W^{(t)} \boldsymbol{Z}^{(t)}.$$

(c) Repeat (a) and (b) until convergence.

5.7.2 Loss Function Comparison (Logistic v.s. Opt Bayes)

Now its time to consider the difference between the LDA/QDA and the logistic regression. The key difference is that they use different loss function. When doing LDA, we use 0-1 loss function, and the optimization is

$$f^*(\mathbf{x}) \leftarrow \arg\min_f \mathbb{E}\Big[\mathbb{1}(y \neq f(\mathbf{x}))\Big].$$

So, we can regard it as

$$\text{LDA/QDA:}\quad \text{loss}\Big(y_i, f(\mathbf{x}_i)\Big) = \mathbb{1}\Big(y_i \neq f(\mathbf{x}_i)\Big).$$

And in the logistic regression, we are optimizing

$$\hat{f}(\mathbf{x}) \leftarrow \arg\min_{f\in\mathscr{H}_k}\Big[-l(f \mid \mathbb{X}, \mathbf{Y}) + \frac{\lambda}{2}||f||^2_{\mathscr{H}_k}\Big],$$

which means the loss function is actually $-l(f)$. And for simple notation, let's denote $\tilde{y}_i = 2y_i - 1 \in \{-1, 1\}$. Then the log-likelihood (5.10) becomes

$$-l(f) = \sum_{i=1}^{N} \log\left(1 + e^{-\tilde{y}_i f(\mathbf{x}_i)}\right).$$

Therefore we can think as

$$\text{Logistic reg:} \quad \text{loss}\Big(y_i, f(\mathbf{x}_i)\Big) = \frac{\log\left(1 + e^{-\tilde{y}_i f(\mathbf{x}_i)}\right)}{\log 2}.$$

If we plot the three loss functions: 0-1, logistic, and Boosting together with respect to $\tilde{y}f(\mathbf{x})$, they behave like the following.

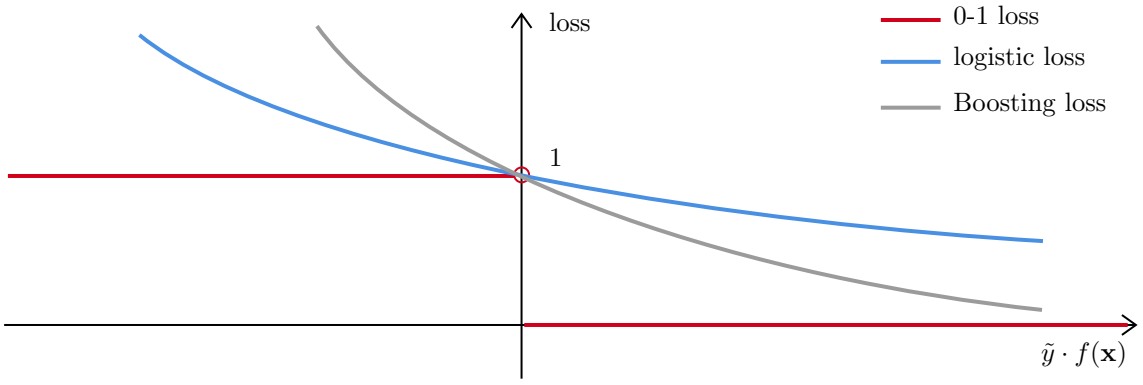


Figure 1: Different loss functions

Remark. We can conclude few things from Figure 1.

(a) In kernel logistic regression, $\hat{\pi} = f(\mathbf{x}_i)$ is the predicted probability instead of the exact predicted class. So, it is continuous.

(b) The result of logistic regression gets highly penalized if $f(\mathbf{x}) \to 1$ but $\tilde{y} = -1$, and vise versa.

(c) We can view logistic loss as a continuous correction of the discrete 0-1 loss.

(d) Boosting loss(to be introduced) is sharp compared with the logistic loss.

5.8 Generalized Additive Model

The generalized additive model is a classification tool, using the same idea of the additive model. For simplicity, let's consider the binary case $\mathcal{Y} = \{0, 1\}$.

Assumption 17 (GAM)**.** Suppose $(\mathbf{x}_i, Y_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Y_i \overset{\text{ind}}{\sim} \text{Ber}(\pi_i) \text{ with } \log\left(\frac{\pi_i}{1-\pi_i}\right) = f(\mathbf{x}_i),$$

where for every observation $\mathbf{x} \in \mathbb{R}^p$, the function f takes the following form:

$$f(\mathbf{x}) = \alpha + f_1(x_1) + \cdots + f_p(x_p) \quad \text{with } f_i'' \text{ exists for all } i = 1, \ldots, p.$$

Similar to the estimation of the additive model (Section 2.3.2 in Note 6.2), our goal should be

$$\hat{\alpha}, \hat{f}_i \leftarrow \arg\min_{\alpha, f_i} \left\{ -l(\mathbf{f}, \alpha) + \sum_{j=1}^{p} \lambda_j \int \left(f''\right)^2 \right\} \quad \text{where} \quad \forall j = 1, \ldots, p: \quad \sum_{i=1}^{N} f_j\left(x_{ij}\right) = 0.$$

And we directly provide an algorithm for the estimation (no proof or derivation is provided).

Algorithm 11 (Local scoring algorithm)**.** Suppose we are estimating the above GAM.

(a) **Initialize**: Set the starting values to be

$$\hat{\alpha} = \log\left(\frac{\bar{y}}{1-\bar{y}}\right) \text{ and } \hat{f}_j(x) \equiv 0, \quad \forall j = 1, \ldots, p.$$

(b) **Iterate**: Define

$$\hat{f}(\mathbf{x}_i) = \hat{\alpha} + \sum_{j=1}^{p} \hat{f}_j(x_{ij}) \text{ and } \hat{\mu}_i = \frac{e^{\hat{f}(\mathbf{x}_i)}}{1 + e^{\hat{f}(\mathbf{x}_i)}}.$$

(b1) Construct the working variable z_i and weights w_i for all $i = 1, \ldots, N$

$$z_i = \hat{f}(\mathbf{x}_i) + \frac{y_i + \hat{\mu}_i}{\hat{\mu}_i\left(1 - \hat{\mu}_i\right)} \text{ and } w_i = \hat{\mu}_i\left(1 - \hat{\mu}_i\right).$$

(b2) Fit a weighted linear additive model of

$$z_i = \alpha + f_1(x_{i1}) + \cdots + f_p(x_{ip}) + \epsilon_i$$

of weights w_i by *backfitting* algorithm.

(c) By repeating (b), we can update $\hat{f}_1, \ldots, \hat{f}_p$ until convergence.

We close the GAM topic by a short discussion. In general, AM and GAM provide us with a more flexible and interpretive model than the linear regression and logistic regression respectively. However, they also bring the difficulties in estimation. Especially in the GAM case, when the p or N is large, the computation is a problem, which leads us to the forward stagewise approach, such as the Boosting (to be introduced).

5.9 Multi-class Logistic Regression

Now we consider the multi-class problem, and we still want to use the model-based method as the logistic regression. The model assumption is generalized as follow.

Assumption 18 (Generalized logistic regression)**.** Suppose $(\mathbf{x}_i, Y_i), i = 1, \dots, N$ is a random sample from the following model:

$$Y_i \overset{\text{ind}}{\sim} \text{Categorical}(\boldsymbol{\pi}_K) \text{ with } \pi_{ik} := \mathbb{P}(Y_i = k \mid \mathbf{x}_i) = \frac{\exp\left[f_k(\mathbf{x}_i)\right]}{\sum_{l=1}^{K} \exp\left[f_l(\mathbf{x}_i)\right]},$$

where for the identifiability we assume $f_K(\mathbf{x}) \equiv 0$ or $\sum_{k=1}^{K} f_k(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbb{R}^p$.

Remark. This model is the combination of generalized logits model and kernel logistic model:

- If we choose $f_k(\mathbf{x}) = \boldsymbol{\theta}_k^\top \mathbf{x}$, then it becomes the generalized logit model.
- If we specify a kernel function k, and choose $f_l(\mathbf{x}) \in \mathscr{H}_k$ for all $j = 1, \dots, K$, it is a generalization of kernel logistic regression.

And if we want the label Y to be ordinal, we can choose the *proportional odds model*. For more, please refer the course *Categorical Data Analysis*.

5.10 Projection Pursuit

Projection pursuit(PP) is another statistical classification method. Let consider the binary case, $\mathcal{Y} = \{0, 1\}$

Assumption 19 (Projection pursuit)**.** Suppose $(\mathbf{x}_i, Y_i), i = 1, \dots, N$ is a random sample from the following model:

$$Y_i \overset{\text{ind}}{\sim} \text{Ber}(\pi_i) \text{ with } \log\left(\frac{\pi_i}{1-\pi_i}\right) = f(\mathbf{x}_i),$$

where for every observation $\mathbf{x} \in \mathbb{R}^p$, the function f takes the following form:

$$f(\mathbf{x}) = \sum_{m=1}^{M} g_m(\mathbf{w}_m^\top \mathbf{x}),$$

where for all $m = 1, \dots, M$, the function g_m is unknown but smooth, and the coefficient $\mathbf{w}$ is unknown.

Remark. There are few things worth comments about the projection pursuit.

- Using the assumption of PP, we are assuming that $\mathbf{w}_m^\top \mathbf{x}$, the M linear combinations of $\mathbf{x}$ are affecting the model, instead of each component of $\mathbf{x}$.
- And this idea could be borrowed to the typical linear model by assuming

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \text{ where the function } f \text{ takes the above form.}$$

- When $M = 1$ in the PP, this model is also called the *single index model* (SIM).
- And PP is closely related to the neural network (to be introduced), which models $f(\mathbf{x})$ as some non-linear but specified multi-layer.

However, the estimation of PP is not easy in general (if the function g is not specified, unlike the neural network). We provide an algorithm to estimate this model.

Algorithm 12 (Projection pursuit estimation)**.** Suppose we are estimating the above PP.

(a) Choose a penalty for functions of g_i's, and define the loss function by

$$\text{loss}(\mathbf{g}, \mathbf{w}) = -l(\mathbf{g}, \mathbf{w}) + \text{penalty}(\mathbf{g}).$$

(b) Fixing $\mathbf{w}$'s, update functions $\mathbf{g}$ by Backfitting algorithm (used in AM).

(c) Fixing $\mathbf{g}$, update $\mathbf{w}$'s by MLE.

Repeat (b) and (c) until convergence. Furthermore, if M is unknown, we can use CV to choose M.

We can see that this estimation is not easy, and its convergence is not guaranteed. This is partly the reason why in neural network, we prespecify the function g's.

In the model-based setting, there is another type of idea, tree-based methods. In this note, we focus on four ways of tree-based ideas:

1. *Classification and regression trees* (CART),
2. *Bootstrap aggregating* also called *bagging*,
3. *Random forest*,
4. *Boosting*(In the next note).

But all in all, these tree-based methods start from decision trees.

5.11 The CART

Here we consider mainly on the classification tree, however the regression tree uses the same idea and is also called the *local averaging method*. To summarize, one classification tree is a way to partition the feature space $\mathcal{X}$ into different hypercubes, and assign a class to each region. For example, a tree T could classify the feature space $\mathcal{X} = [0, 100]^2$ into $R_1, \ldots, R_4$ by the following partition:

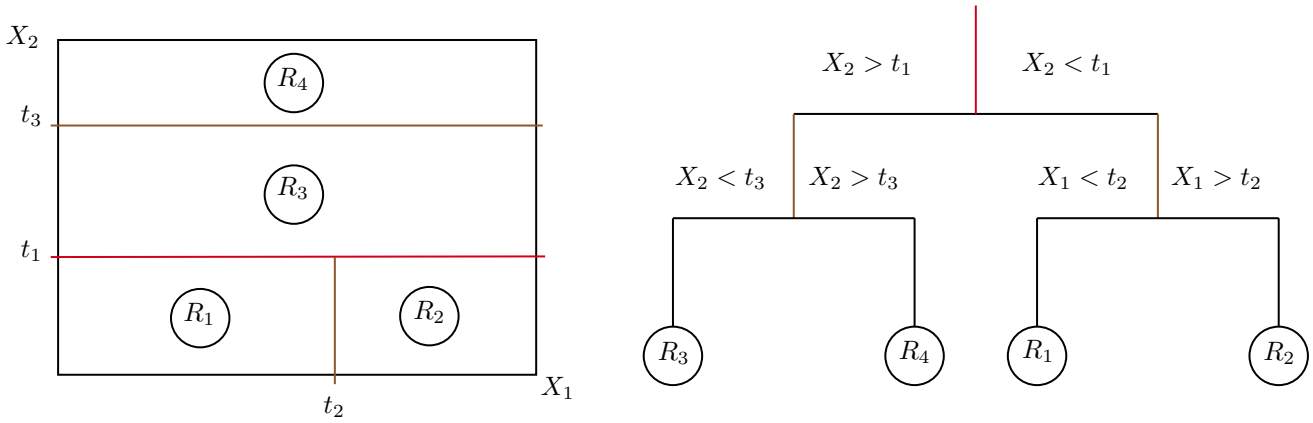


Figure 1: An example of classification tree

5.11.1 Model Assumptions

In fact, the classification tree can be viewed as the following model.

Assumption 20 (Classification tree)**.** Given the number of region M and the regions $R_1, \ldots, R_M \in \mathbb{R}^p$, suppose $(\mathbf{x}_i, Y_i)$, $i = 1, \ldots, N$ where $\mathbf{x}_i \in \mathbb{R}^p$ and $Y_i \in \{0, 1\}$ is a random sample from the following model:

$$\mathbb{P}(Y = 1 \mid \mathbf{x}) = \sum_{m=1}^{M} p_m \mathbb{1}(\mathbf{x} \in R_m), \quad \text{where} \quad p_m := \mathbb{P}(Y = 1 \mid \mathbf{x} \in R_m).$$

Or equivalently, assume

$$\log\left[\frac{\mathbb{P}(Y = 1 \mid \mathbf{x})}{\mathbb{P}(Y = 0 \mid \mathbf{x})}\right] = \sum_{m=1}^{M} \log\left(\frac{p_m}{1 - p_m}\right) \mathbb{1}(\mathbf{x} \in R_m).$$

For multi-class trees, i.e. $\mathcal{Y} = \{1, \dots, K\}$, we assume

$$\mathbb{P}(Y = k \mid \mathbf{x}) = \sum_{m=1}^{M} p_{mk} \mathbb{1}(\mathbf{x} \in R_m) \text{ where } p_{mk} := \mathbb{P}(Y = k \mid \mathbf{x} \in R_m).$$

Note that here, we assume M and $R_1, \dots, R_m$ are prespecified. Then the only thing needs estimation is p_{mk}, whose natural estimator is

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{i=1}^{N} \mathbb{1}(Y_i = k, \mathbf{x}_i \in \mathbb{R}_m), \text{ where } N_m = \sum_{i=1}^{N} \mathbb{1}(\mathbf{x}_i \in R_m),$$

with which, the predicted class (optimal Bayes classifier) of any new observation $\mathbf{x}^* \in R_m$ should be

$$k(m) := \arg\max_{k \in \mathcal{Y}} \hat{p}_{mk}.$$

Remark. The regression tree uses the same idea of the above, by assuming that

$$\mathbb{E}(Y \mid \mathbf{X} = \mathbf{x}) = \sum_{m=1}^{M} c_m \mathbb{1}(\mathbf{x} \in R_m),$$

and a natural estimator of c_m is

$$\hat{c}_m = \frac{1}{N_m} \sum_{i=1}^{N} y_i \mathbb{1}(\mathbf{x}_i \in R_m).$$

5.11.2 Estimation of Regions and Number of Regions

In the previous argument, we see that p_{mk} for all m, k are estimable given all regions $R_1, \dots, R_M$. Now it remains to estimate these regions from our training data since, practically, the regions are unknown. Same as the previous methods, different loss functions lead to different classifiers. Here, in decision tree, three loss functions are widely used.

Misclassification Error Loss

Given a tree T (the number of regions M, and all regions $R_1, \dots, R_M$), we have previously defined

$$N_m = \sum_{i=1}^{N} \mathbb{1}\left(\mathbf{x}_i \in R_m\right) \text{ and } k(m) = \arg\max_{k \in \mathcal{Y}} \hat{p}_{mk}.$$

And the misclassification error loss in the region m is defined as

$$\forall m = 1, \dots, M: \quad Q_m(T) = \frac{1}{N_m} \sum_{i: \mathbf{x}_i \in R_m} \mathbb{1}\Big[y_i \neq k(m)\Big].$$

It intuitively means the fraction of misclassification in this region. Simple calculation leads to

$$Q_m(T) = 1 - \frac{1}{N_m} \sum_{i: \mathbf{x}_i \in R_m} \mathbb{1}\Big[y_i = k(m)\Big] = 1 - \hat{p}_{m,k(m)}.$$

Gini Index Loss

Given a tree T (the number of regions M, and all regions $R_1, \ldots, R_M$), define the *Gini index loss* function by

$$Q_m(T) = \sum_{k=1}^{K} \hat{p}_{mk}\,(1 - \hat{p}_{mk}) = \sum_{1 \le k \ne k' \le K} \hat{p}_{mk}\hat{p}_{mk'}.$$

Cross Entropy

The *cross entropy* loss or *deviance* is defined by

$$Q_m(T) = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.$$

Comparison between Different Loss

It is more easier to see the different in the binary case. Now let's suppose $\mathcal{Y} = \{0, 1\}$, $p = \hat{\mathbb{P}}(Y = 1 \mid \mathbf{x} \in R_m)$, and three loss functions for region m above can be simplified to be

$$\begin{cases} \text{Misclassification error:} & Q_m(T) = 1 - \hat{p}_{m,k(m)} = 1 - \max(p, 1-p) \\ \text{Gini index:} & Q_m(T) = 2p(1-p) \\ \text{Cross entropy:} & Q_m(T) = -p\log p - (1-p)\log(1-p). \end{cases}$$

Or we can see the plots:

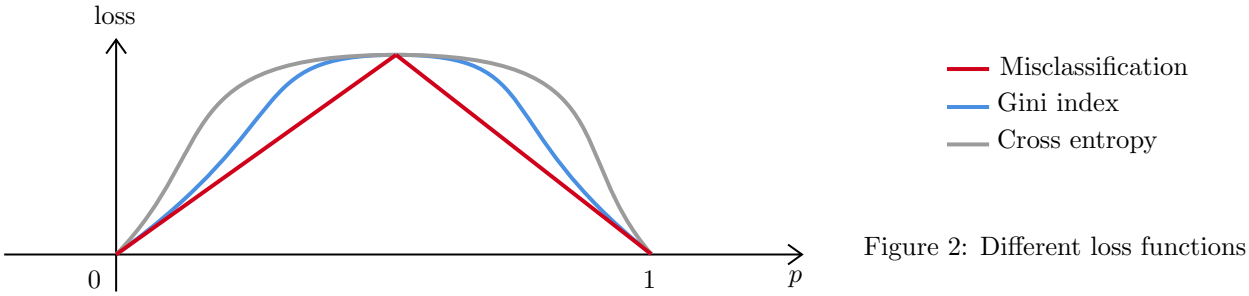


Figure 2: Different loss functions

Cross entropy loss needs a scale to stand at the same vertex, and then Gini index and cross entropy losses are numerically very similar. And they are both smooth losses, so we can use gradient decent algorithm for the optimization. However, misclassification error has no derivative at 0.5, which brings a little difficulty in optimization. For more discussion, we can refer §9.2.3, *Elements of Statistical Learning*.

Estimation Procedures

The algorithm for decision tree is well known as follows.

Algorithm 13 (Decision tree)**.** Suppose we have the sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$.

(a) **Recursive partition**: Select the splitting variable $j \in \{1, \ldots, p\}$ and the splitting point $s \in \mathbb{R}$ for $R_1 + R_2 = \{X_j < s\} + \{X_j > s\}$ by minimizing the loss function:

$$\hat{j}, \hat{s} \leftarrow \arg\min_{j,s}(N_1 Q_1 + N_2 Q_2),$$

where

$$N_m = \sum_{i=1}^{N} \mathbb{1}\left(\mathbf{x}_i \in R_m\right) \text{ and } Q_m \text{ is the loss in region } m.$$

(b) Repeat (a) recursively until we get a very large tree T.

(c) **Pruning**: Prespecify a tuning parameter $\alpha > 0$, get a pruned tree T_0 by

$$T_0 = \min_T \sum_{m=1}^{M(T)} N_m Q_m(T) + \alpha M(T), \text{ where } M(T) = \#\text{ of terminal nodes / regions.}$$

We close this part by a discussion of the advantages and disadvantages of CART. A significant benefit of it is good interpretability, since every node is clear. However, the tree may have large variance due to the hierarchical nature, i.e. an error in a top note have impact on all splits below. So, it is not robust as the following methods.

5.12 Bagging

The general idea of bootstrap aggregating is to generate perturbed versions of the training data (usually by bootstrap), train a tree on each perturbed dataset, and aggregate results from these trees by majority voting. The procedures are as follows.

Algorithm 14 (Bootstrap aggregating)**.** Suppose we have the sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$.

(a) Create B bootstrapped training dataset $D_1, \ldots, D_B$.

(b) Train a classification tree T_b on each dataset $D_1, \ldots, D_B$.

(c) For a new observation $\mathbf{x}$, classify its label to be the majority class of these B trees.

The reason to do this way is, intuitively, bagging reduces the variance of classifier while not affect the bias. In other words, if the original classification tree method is biased, the bagging would still be biased and vise versa.

We give a rough intuition of why it induces the variance of the classifier. Suppose the feature is $\mathbf{x} \in \mathbb{R}^p$, fixed, and we have grown B trees, $T_1, \ldots, T_B$. Since these trees are grown on the bootstrapped dataset, which may overlap, we assume the classification results $T_1(\mathbf{x}), \ldots, T_B(\mathbf{x})$ are identically distributed but correlated with

$$\forall i \neq j: \quad \text{Corr}\Big[T_i(\mathbf{x}), T_j(\mathbf{x})\Big] = \rho, \text{ and } \text{Var}T_i(\mathbf{x}) = \sigma^2.$$

Then simple derivation leads to

$$\text{Var}\left[\frac{T_1(\mathbf{x}) + \cdots + T_B(\mathbf{x})}{B}\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \to \rho\sigma^2, \quad \text{as } B \to \infty.$$

But the expectation of $\bar{T}$ remains the same as T_i.

Though the wonderful reduction in variance, Bagging inevitably leads the classification rule unclear, i.e., given a new $\mathbf{x}$, we don't know the result unless plug it into the computer. But a good news is that we have

the following way to measure the importance of each variable X_j, $j = 1, \ldots, p$.

5.12.1 Variable Importance Measure

Recall that when growing each tree, we are dealing with the loss function at each node $Q_m(T)$. And the variable importance measure of X_j is the total decrease amount in the loss function by splitting over X_j. Consider the following example for $B = 2$, among which the red splits denote the usage of X_2.

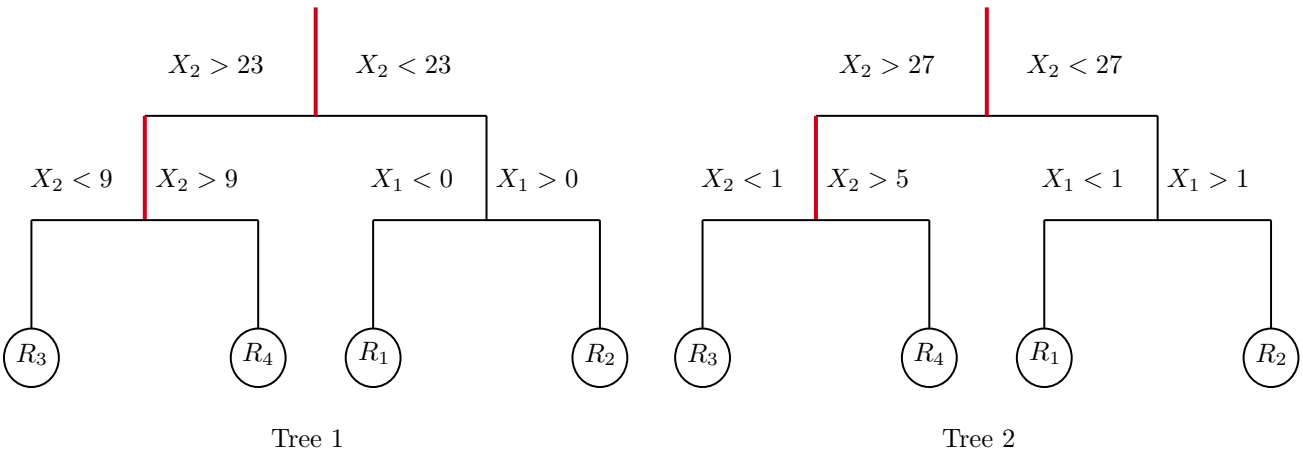


Figure 3: An example of variable importance

X_2's importance is the total amount of decrease of loss function at the four red splits.

5.13 Random Forest

Random forest is a further improvement based on Bagging. The key idea is to further reduce the variance of $\bar{T}$

$$\text{Var}\left[\frac{T_1(\mathbf{x}) + \cdots + T_B(\mathbf{x})}{B}\right] = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \to \rho\sigma^2, \quad \text{as } B \to \infty,$$

by reducing the correlation ρ, which results from the overlap in the bootstrap dataset.

Algorithm 15 (Random forest)**.** Suppose we have the sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$.

(a) Create B bootstrapped training dataset $D_1, \ldots, D_B$.

(b) Train a classification tree T_b on each dataset $D_1, \ldots, D_B$ as follows.

b1) At each node, consider a random subset of m variables for the next split, such as $m = \lfloor \sqrt{p} \rfloor$.

b2) Trees are pruned to some minimum node size (e.g. 1).

(c) For a new observation $\mathbf{x}$, classify its label to be the majority class predicted by $T_1, \ldots, T_B$.

When $B \to \infty$, in fact, the correlation between trees ρ gets decreased, however, the variance of every single classifier σ^2 increases as we only use a subset of variables. The good news is that the whole thing $\rho\sigma^2$

decreases practically.

5.13.1 Variable Importance

Similar to Bagging, the classification rule is not clear in random forest. And we can also compute the variable importance (the same definition) for each variable.

5.13.2 OOB Sample

At the bootstrap step, those observations which are not covered in D_b form a ọut-ọf-ḅag sample (OOB). And we can use them to get estimated validation error (OOB error) as follows. For each $(\mathbf{x}_i, y_i)$, construct its error by averaging only those trees from bootstrapped data without $(\mathbf{x}_i, y_i)$.

5.14 Boosting Method

Boosting is a supervised machine learning method, whose first version, *Adaboost*, is proposed by computer scientists Freund and Schapire in 1997. For simplicity, let's first consider binary case $\mathcal{Y} = \{-1, 1\}$.

5.14.1 Adaptive Boosting

The general idea of boosting is to sequentially combine a sequence of weak classifiers (slightly better than random guess, such as trees) through a weighted majority vote. To be specific, if we have M classifiers $T_1, \ldots, T_M$, we can define the final classifier as

$$G(\mathbf{x}) = \operatorname{sign}\left[\sum_{m=1}^{M} \alpha_m T_m(\mathbf{x})\right], \quad \text{where } \alpha\text{'s are weights computed from data.}$$

Remark. Before the formal procedures, we outlook two difference between Boosting and previous methods.

(a) Each tree is pruned using the information from previous trees as we shall see.

(b) We are no longer using bootstrap for dataset now. Instead, each tree is fitted on a modified version (something like residuals) of the original data as we shall see.

Algorithm 16 (AdaBoost-1)**.** Suppose we have the sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$.

(a) Initialize the weights for each observation as $w_i = 1/N$ for $i = 1, \ldots, N$.

(b) Grow the m-th tree based on the previous ones for $m = 1, \ldots, M$ as follows.

b1) Fit T_m on the training dataset using weights w_i's.

b2) Compute error rate of the m-th tree as

$$\mathrm{err}_m := \frac{\sum_{i=1}^{N} w_i \mathbb{1}\left[y_i \neq T_m(\mathbf{x}_i)\right]}{\sum_{i=1}^{N} w_i}.$$

b3) Compute the weight for m-th tree as

$$\alpha_m = \log\left(\frac{1 - \mathrm{err}_m}{\mathrm{err}_m}\right).$$

b4) Update weights for each observation as

$$w_i \leftarrow w_i \cdot \exp\Big\{\alpha_m \mathbb{1}\Big[y_i \neq T_m(\mathbf{x}_i)\Big]\Big\}.$$

(c) Output the final classifier as

$$G(\mathbf{x}) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m T_m(\mathbf{x})\right].$$

Remark. We can intuitively see some points from this algorithm.

- We can see err_m is the misclassification rate of T_m, and the weight of each tree depends on this error rate.

- Every time we start a new tree, the new weight of each observation depend on the previous trees. If the previous one is correctly predicted, the weight of this observation does not change; otherwise, the weight gets increased.

Statistical Meaning of AdaBoost

AdaBoost is equivalent to a type of forward additive model, the *additive logistic regression*.

Assumption 21 (Additive logistic regression)**.** Suppose $(\mathbf{x}_i, Y_i)$ for $i = 1, \ldots, N$ is a random sample from the following model:

$$Y_i \in \{-1, 1\} \text{ with } \log\left[\frac{\mathbb{P}(Y_i = 1 \mid \mathbf{x})}{\mathbb{P}(Y_i = -1 \mid \mathbf{x})}\right] = f(\mathbf{x}_i),$$

where for each observation $\mathbf{x} \in \mathbb{R}^p$, the function f takes the following form:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m T_m(\mathbf{x})$$

where weights α_m and weak classifiers $T_m(\mathbf{x})$ need estimating from the data.

If we estimate the additive logistic regression model under the exponential loss:

$$L\Big(y, f(\mathbf{x})\Big) = \exp\Big\{-yf(\mathbf{x})\Big\}, \tag{5.11}$$

then we can use the following *forward estimation* algorithm.

Algorithm 17 (AdaBoost-2)**.** Suppose we have the sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$.

(a) Initialize the classifier $f_0(\mathbf{x}) \equiv 0$.

(b) Update the classifier for $m = 1, \ldots, M$ as follows.

b1) Compute the weight α_m and the weak classifier T_m by

$$(\alpha_m, T_m) \leftarrow \arg\min_{\alpha,T} \sum_{i=1}^{N} L\Big(y_i, f_{m-1}(\mathbf{x}) + \alpha T(\mathbf{x}_i)\Big)$$

b2) Update the classifier by

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m T_m(\mathbf{x}).$$

(c) Repeat until we get the final classifier f_m.

Though we start from the AdaBoost by computer scientists' idea, the following proposition shows the statistical model could be the underlying assumption for that algorithm, AdaBoost-1.

Proposition 10. *For the additive logistic regression model, AdaBoost-1 and AdaBoost-2 Algorithms are equivalent, i.e. they return the same estimates.*

Proof. In the first step, we show the solution to AdaBoost-2 (b1) for T_m is

$$T_m(\mathbf{x}) \leftarrow \arg\min_{T} \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}\Big(y_i \neq T(\mathbf{x}_i)\Big), \quad \text{where} \quad w_i^{(m)} = \exp\Big\{ - y_i f_{m-1}(\mathbf{x}_i)\Big\}. \tag{5.12}$$

Plugging the required exponential loss (5.11), it follows that

$$\begin{aligned}\sum_{i=1}^{N} L\Big(y_i, f_{m-1}(\mathbf{x}) + \alpha T(\mathbf{x})\Big) &= \sum_{i=1}^{N} \exp\Big\{ - y_i f_{m-1}(\mathbf{x}_i) - \alpha y_i T(\mathbf{x}_i)\Big\} \\ \Big(\text{By def of } w_i^{(m)}\Big) \quad &= \sum_{i=1}^{N} w_i^{(m)} \exp\Big\{ - \alpha y_i T(\mathbf{x}_i)\Big\}.\end{aligned}$$

Note that $T(\mathbf{x}_i)$ is a classification result, and takes value -1 or 1. So, we can further simplify

$$\begin{aligned}\sum_{i=1}^{N} L\Big(y_i, f_{m-1}(\mathbf{x}) + \alpha T(\mathbf{x})\Big) &= e^{-\alpha} \sum_{i:y_i = T(\mathbf{x}_i)} w_i^{(m)} + e^{\alpha} \sum_{i:y_i \neq T(\mathbf{x}_i)} w_i^{(m)} \\ &= e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}\Big(y_i = T(\mathbf{x}_i)\Big) + e^{\alpha} \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}\Big(y_i \neq T(\mathbf{x}_i)\Big) \\ &= \big(e^{\alpha} - e^{-\alpha}\big) \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}\Big(y_i \neq T(\mathbf{x}_i)\Big) + e^{-\alpha} \sum_{i=1}^{N} w_i^{(m)}. \end{aligned} \tag{5.13}$$

Recall the goal optimization is

$$(\alpha_m, T_m) \leftarrow \arg\min_{\alpha,T} \sum_{i=1}^{N} L\Big(y_i, f_{m-1}(\mathbf{x}) + \alpha T(\mathbf{x}_i)\Big), \tag{5.14}$$

By (5.13), we split the optimization of α and T, and the solution for T_m should be

$$T_m \leftarrow \arg\min_{T} \sum_{i=1}^{N} w_i^{(m)} \mathbb{1}\Big(y_i \neq T(\mathbf{x}_i)\Big), \tag{5.15}$$

which completes the first part of our proof.

The second step is to solve (5.14) for α in AdaBoost-2 (b1). Plugging the solution of T (5.15) into (5.14), it follows that

$$\alpha_m \leftarrow \arg\min_{\alpha} \sum_{i=1}^{N} w_i^{(m)} \exp\Big\{ -\alpha y_i T_m(\mathbf{x}_i) \Big\}.$$

We just do usual step, taking derivative with respect to α and setting it to be 0:

$$\begin{aligned}\frac{\partial}{\partial \alpha} \sum_{i=1}^{N} w_i^{(m)} \exp\Big\{ -\alpha y_i T_m(\mathbf{x}_i) \Big\} &= \sum_{i=1}^{N} w_i^{(m)} \exp\Big\{ -\alpha y_i T_m(\mathbf{x}_i) \Big\} \cdot y_i T_m(\mathbf{x}_i) \\ &= \sum_{i: y_i \neq T_m(\mathbf{x}_i)} e^{-\alpha} w_i^{(m)} - \sum_{i: y_i = T_m(\mathbf{x}_i)} e^{-\alpha} w_i^{(m)}.\end{aligned}$$

Simple algebra leads to

$$\hat{\alpha} = \frac{1}{2} \log\left[\frac{1-\text{err}_m}{\text{err}_m}\right], \quad \text{where } \text{err}_m := \frac{\sum_{i=1}^{N} w_i \mathbb{1}\Big[y_i \neq T_m(\mathbf{x}_i)\Big]}{\sum_{i=1}^{N} w_i}.$$

Recall AdaBoost-2 (b2). Since we now know α_m and T_m, we can update the classifier to f_m. And by the definition (5.12), we can compute the updated weights for each observation to be

$$\begin{aligned} w_i^{(m+1)} &= \exp\Big\{ -y_i f_m(\mathbf{x}_i) \Big\} \\ &= \exp\Big\{ -y_i f_{m-1}(\mathbf{x}_i) - y_i \alpha_m T_m(\mathbf{x}) \Big\} \\ &= w_i^{(m)} e^{2\alpha_m \mathbb{1}\Big(y_i \neq T_m(\mathbf{x}_i)\Big)} \cdot e^{-\alpha_m}, \end{aligned}$$

which is exactly a scaled version of weights for observations in AdaBoost-1, completing the proof. □

Why the Exponential Loss?

Recall the assumption of the additive logistic regression model.

Assumption 22 (Additive logistic regression)**.** Suppose $(\mathbf{x}_i, Y_i)$ for $i = 1, \ldots, N$ is a random sample from the following model:

$$Y_i \in \{-1, 1\} \text{ with } \log\left[\frac{\mathbb{P}(Y_i = 1 \mid \mathbf{x})}{\mathbb{P}(Y_i = -1 \mid \mathbf{x})}\right] = f(\mathbf{x}_i), \tag{5.16}$$

where for each observation $\mathbf{x} \in \mathbb{R}^p$, the function f takes the following form:

$$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m T_m(\mathbf{x})$$

where weights α_m and weak classifiers $T_m(\mathbf{x})$ need estimating from the data.

In the last subsection, we know doing AdaBoost algorithm, we are, in essence, doing stagewise fitting of

the following model under exponential loss

$$L\Big(y, f(\mathbf{x})\Big) = e^{-yf(\mathbf{x})}.$$

Why do we choose this loss function? In the consideration of the population form, there are benefits.

Now consider the population $(\mathbf{X}, Y)$ where the label $Y \in \{-1, 1\}$ and we assume a conditional distribution $Y \mid \mathbf{X}$ for the label (we don't care the distribution of $\mathbf{X}$). Our goal is to find the optimal Bayes classifier

$$\begin{aligned}\arg\min_f \mathbb{E}_{Y|\mathbf{X}}\Big[L\Big(y, f(\mathbf{x})\Big)\Big] &= \arg\min_f \mathbb{E}_{Y|\mathbf{X}}\Big[e^{-yf(\mathbf{x})}\Big] \\ (\because Y \in \{-1,1\}) \quad &= \arg\min_f \Big[e^{-f(\mathbf{x})}\mathbb{P}(Y=1 \mid \mathbf{x}) + e^{f(\mathbf{x})}\mathbb{P}(Y=-1 \mid \mathbf{x})\Big].\end{aligned}$$

Since f is a classifier, and $f(\mathbf{x}) \in \mathbb{R}$, we can take derivative of the objective function w.r.t. $f(\mathbf{x})$, and set

$$\frac{\partial}{\partial f(\mathbf{x})}\text{object}\Big|_{\hat{f}(\mathbf{x})} = 0 \quad \Rightarrow \quad \hat{f}(\mathbf{x}) = \frac{1}{2}\log\left[\frac{\mathbb{P}(Y=1 \mid \mathbf{x})}{\mathbb{P}(Y=-1 \mid \mathbf{x})}\right].$$

This means under exponential loss, the form of optimal Bayes classifier is consistent with the true model (5.16):

$$\text{sign}\Big[\hat{f}(\mathbf{x})\Big] = \text{sign}(5.16).$$

Remark. In fact, if we choose the logistic regression, which means the negative log-likelihood loss:

$$L\Big(y, f(\mathbf{x})\Big) = \log\left(1 + e^{yf(\mathbf{x})}\right),$$

then the optimal Bayes classifier would be (the same derivation)

$$\arg\min_f \mathbb{E}_{Y|\mathbf{X}} L\Big(y, f(\mathbf{x})\Big) = \log\left[\frac{\mathbb{P}(Y=1 \mid \mathbf{x})}{\mathbb{P}(Y=-1 \mid \mathbf{x})}\right],$$

and is also in a consistent form with the true model (5.16).

Loss Comparisons

We end the topic about AdaBoost by comparing different loss functions in the binary case $\mathcal{Y} = \{-1, 1\}$.

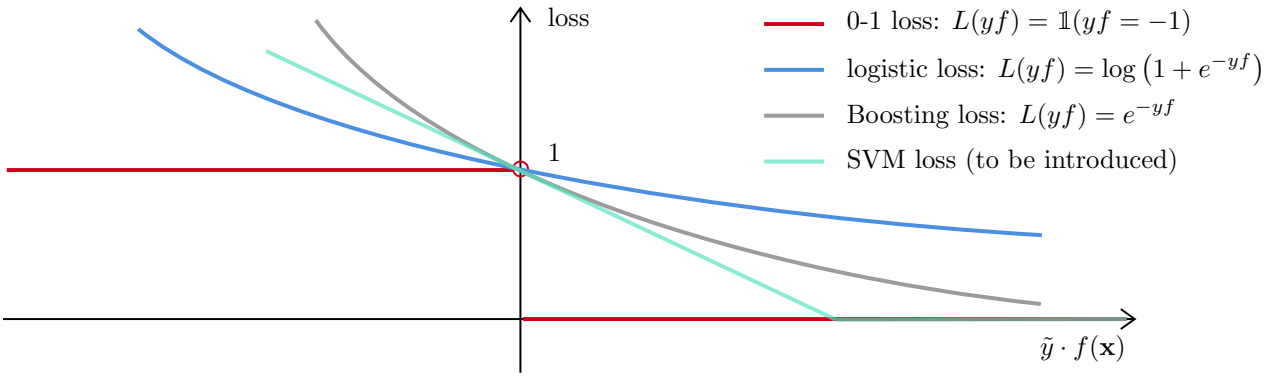


Figure 4: Different loss functions

There are several things worth noticing.

- Both logistic loss (negative log-likelihood) and boosting loss (exponential) can be viewed as monotone and continuous approximation to 0-1 loss.
- When $(-yf)$ is negatively large, the exponential loss increases linearly, however the the exponential loss increases exponentially. This means the logistic regression, and other methods using this negative log-likelihood loss (also called deviance), are much robust than the boosting, especially when there exists label misclassification in the training set (or other noisy setting). This is because the exponential loss puts heavy penalty when $(-yf)$ is negatively large.
- But there is an apparent benefit of exponential loss: AdaBoost under it is very easy to fit, no matter we use AdaBoost-1 or AdaBoost-2 algorithm, since the estimation of $\boldsymbol{\alpha}$ and T are separable as we see in the previous proof.

If we do not use exponential loss, the estimation of the model becomes complicated since the estimation of $\boldsymbol{\alpha}$ and T are separable, which leads us the an alternative solution, the *gradient boosting* technique.

5.14.2 Gradient Boosting

To solve the difficulty in estimation under different loss functions, Friedman (*Annals of Statistics*, 2001) first proposed the gradient boosting technique, borrowing the idea of gradient descent (steepest descent) optimization.

The General Idea

The multi-class model assumption is exactly as the generalized logistic regression.

Assumption 23 (Gradient boosting for multi-class problem)**.** Suppose $(\mathbf{x}_i, Y_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Y_i \overset{\text{ind}}{\sim} \text{Categorical}(\boldsymbol{\pi}_K) \;\text{ with }\; \pi_{ik} := \mathbb{P}(Y_i = k \mid \mathbf{x}_i) = \frac{\exp\left[f_k(\mathbf{x}_i)\right]}{\sum_{l=1}^{K} \exp\left[f_l(\mathbf{x}_i)\right]}, \tag{5.17}$$

where for the identifiability we assume $f_K(\mathbf{x}) \equiv 0$ or $\sum_{k=1}^{K} f_k(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbb{R}^p$, and each function f_k from $k = 1, \ldots, K$ takes the following form

$$f_k(\mathbf{x}) = \sum_{m=1}^{M} T_{km}(\mathbf{x}, \boldsymbol{\theta}_{km}) \;\text{ where }\; T_{km} = \sum_{j=1}^{J_m} \gamma_{kmj} \mathbb{1}(\mathbf{x} \in R_j).$$

For the fixed label type k and the tree m, the parameters are $\boldsymbol{\theta}_{km} = \{\gamma_{kmj},\ R_j,\ j = 1, \ldots, J_m\}$ and J_m is the prespecified number of nodes in the m-th tree.

Before the formal algorithm elaboration, the general idea to estimate $\boldsymbol{f} = (f_1, \ldots, f_k)$ by a following forward stagewise algorithm given the data and a smooth loss function. By forward stagewise, we mean, given all parameters at step $m-1$, we want to update for all $k = 1, \ldots, K$ by

$$\hat{\boldsymbol{\theta}}_{km} \leftarrow \arg\min_{\boldsymbol{\theta}_{km}} \sum_{i=1}^{N} L\Big(y_i, f_{k,m-1}(\mathbf{x}_i) + T(\mathbf{x}_i, \boldsymbol{\theta}_{km})\Big),$$

where, by the idea of learning tree by tree,

$$f_{k,m-1}(\mathbf{x}_i) := \sum_{l=1}^{m-1} T_{kl}(\mathbf{x}_i, \hat{\boldsymbol{\theta}}_{kl}).$$

Remark. It is easy to check that, for the binary case $\mathcal{Y} = \{-1, 1\}$, we don't need the subscript k for different label. And further under the exponential loss, if we restrict $\gamma_{jm} \in \{-\alpha_m, \alpha_m\}$, then this is called the scaled classification trees, and returns to AdaBoost.

Different Loss Examples

In gradient boosting, we can choose any smooth loss function as we shall see. For example, if we choose the negative log-likelihood loss, it means for a single observation $(\mathbf{x}, y)$,

$$\begin{aligned} L\Big(y, \boldsymbol{f}(\mathbf{x})\Big) &= -\log\left[\prod_{k=1}^{K} \mathbb{P}(y = k \mid \mathbf{x})^{\mathbb{1}(y=k)}\right] \\ &= -\sum_{k=1}^{K} \mathbb{1}(y = k) \log\Big[\mathbb{P}(y = k \mid \mathbf{x})\Big] \\ \Big(\text{By assumption (5.17)}\Big) \quad &= -\sum_{k=1}^{K} \mathbb{1}(y = k) f_k(\mathbf{x}) + \log\left[\sum_{l=1}^{K} e^{f_l(\mathbf{x})}\right]. \end{aligned} \tag{5.18}$$

We also still have the exponential loss function for the multi-class setting. But we don't elaborate it here, and for details, refer §10.5 of *Elements of Statistical Learning*.

Gradient Boosting Algorithm

As we noted, the gradient boosting algorithm borrows the idea of gradient descent (steepest descent) algorithm. More specifically, we optimize by

$$f_m(\mathbf{x}_i) \leftarrow f_{m-1}(\mathbf{x}_i) + \rho_m g_{mi} \;\text{ with }\; \rho_m = \arg\min_{\rho} L\Big(y, f_{m-1} + \rho g_m\Big),$$

where ρ_m is a step-size and g_{mi} is the gradient of loss function with respect to $f(\mathbf{x}_i)$. And the gradient boosting algorithm is formally as follows. For more details, refer §10.3 & 10.4 of *Elements of Statistical Learning*.

Algorithm 18 (Gradient boosting)**.** Suppose we have the sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$, and prespecify the number of tree M as well as the number of nodes J_m of each tree $m = 1, \ldots, M$. Our goal is optimize for all trees $m = 1 \ldots, M$ and all labels $k = 1 \ldots, K$

$$\hat{\boldsymbol{\theta}}_{km} \leftarrow \arg\min_{\boldsymbol{\theta}_{km}} \sum_{i=1}^{N} L\Big(y_i, f_{k,m-1}(\mathbf{x}_i) + T(\mathbf{x}_i, \boldsymbol{\theta}_{km})\Big).$$

(a) Initialize the classifier $\boldsymbol{f}^{(0)} = \left(f_1^{(0)} \ \dots \ f_K^{(0)}\right)$ by

$$\text{Binary case: set } \ f_0(\mathbf{x}) = \arg\min_{\gamma \in \mathbb{R}^1} \sum_{i=1}^{N} L(y_i, \gamma),$$

$$\text{Multi-class: set } \ \boldsymbol{f}_0(\mathbf{x}) = (f_{01} \ \dots \ f_{0J_0}) \equiv \mathbf{0}.$$

(b) Update the classifier for $m = 1, \dots, M$ as follows.

b1) Compute the negative gradient of m-th tree at each sample point $\mathbf{x}_i$ for $i = 1, \dots, N$

$$\boldsymbol{g}_{im} := (g_{i1m} \ \dots \ g_{iKm}) := -\left.\frac{\partial L\Big(y_i, \boldsymbol{f}(\mathbf{x}_i)\Big)}{\partial \boldsymbol{f}(\mathbf{x}_i)}\right|_{\boldsymbol{f}(\mathbf{x}_i)=\boldsymbol{f}_{m-1}(\mathbf{x}_i)}.$$

(Note the loss function $L\Big(y_i, \boldsymbol{f}(\mathbf{x}_i)\Big)$ is a function of $\boldsymbol{f}$, and the partial derivative would be a vector function. When we plug in $\boldsymbol{f}_{m-1}(\mathbf{x}_i)$, it is a known vector. We will see an example after the whole algorithm.)

b2) Fit a regression tree to the gradient vector $\boldsymbol{g}_{im} \sim \mathbf{x}_i$ for $i = 1, \dots, N$, and get terminal regions R_{jkm} for $j = 1, \dots, J_m$. (J_m is prespecified, and it means how many regions we want to use to approximate the gradient.)

b3) Compute for each region $j = 1, \dots, J_m$ and for each label $k = 1, \dots, K$

$$\gamma_{jkm} = \arg\min_{\gamma} \sum_{i:\mathbf{x}_i \in R_{jkm}} L\Big(y_i, f_{m-1}(\mathbf{x}_i) + \gamma\Big).$$

(Optimizing the parameter in each region separately makes this algorithm doable.)

b4) Update the m-th classifier $\boldsymbol{f}_m = (f_{1m} \ \dots \ f_{Km})$

$$f_{k,m}(\mathbf{x}) = f_{k,m-1}(\mathbf{x}) + v \sum_{j=1}^{J_m} \gamma_{jkm} \mathbb{1}(\mathbf{x} \in R_{jkm}),$$

where v is a prespecified learning rate (normally < 0.1).

(c) Output the final classifier $\hat{\boldsymbol{f}}(\mathbf{x}) = \boldsymbol{f}_M(\mathbf{x})$.

For illustration purpose, let's see an example of gradient calculation (b1).

Example (Negative log-likelihood loss)**.** Recall in (b1), the gradient vector of sample point i and m-th tree would be

$$\boldsymbol{g}_{im} := (g_{i1m} \ \dots \ g_{iKm}) := -\left.\frac{\partial L\Big(y_i, \boldsymbol{f}(\mathbf{x}_i)\Big)}{\partial \boldsymbol{f}(\mathbf{x}_i)}\right|_{\boldsymbol{f}(\mathbf{x}_i)=\boldsymbol{f}_{m-1}(\mathbf{x}_i)}.$$

If we use negative log-likelihood loss, using the derivation (5.18), each component g_{ikm} would be

$$g_{ikm} = \mathbb{1}(y_i = k) - \frac{\exp\Big\{f_{k,m-1}(\mathbf{x}_i)\Big\}}{\sum_{l=1}^{k}\exp\Big\{f_{l,m-1}(\mathbf{x}_i)\Big\}},$$

which is a number that could be calculated by the information in $m-1$ step.

Remarks on Gradient Boosting

Similar to stochastic version of IRLS in the (kernel) logistic regression, to avoid complicated computation, we can substitute the goal

$$\hat{\boldsymbol{\theta}}_{km} \leftarrow \arg\min_{\boldsymbol{\theta}_{km}} \sum_{i=1}^{N} L\Big(y_i, f_{k,m-1}(\mathbf{x}_i) + T(\mathbf{x}_i, \boldsymbol{\theta}_{km})\Big).$$

to be done in a fraction $\eta(< 1/2)$ of training data (not bootstrap, but direct draw without replacement).

The second point worth mentioning is the case when $J_m = 2$ for all $m = 1, \ldots, M$ and $M \gg p$. In this setting, every tree has only one split of one variable and most of the variables could be selected in some trees. And we call this model, the *main-effect model*, since

$$f(\mathbf{x}) = \sum_{m=1}^{M} T_m(\mathbf{x}) = g_1(x_1) + \cdots + g_p(x_p),$$

where $g_i(\cdot)$ is a function that aggregates all splits by x_i, and therefore a function only related to x_i. Since g_i is an aggregation of trees, it is still a step function (a little different from the AM and GAM which requires g's to be smooth).

More generally, when $J_m = 3$ or higher, the classification function $f(\mathbf{x})$ has an ANOVA-type expansion with interactions:

$$f(\mathbf{x}) = \sum_{m=1}^{M} T_m(\mathbf{x}) = \sum_{j=1}^{p} g_j(x_j) + \sum_{0 \le j \ne k \le p} g_{jk}(x_j, x_k) + \text{higher terms}.$$

This note focus on the classification method - neural network. For more reference, check §11 of *Elements of Statistical Learning*, §5 of Bishop, or the book *Deep Learning*.

5.15 Single Hidden Layer Network

Neural networks are classification methods that utilize many layers as we shall see. So, we start from the very simple one, the *single hidden layer network* model. In essence, it is a non-linear statistical model.

Assumption 24 (Single layer network)**.** We can use the neural network to deal with two kinds of problems: regression and classification.

- **Regression**: Suppose $(\mathbf{x}_i, \mathbf{y}_i) \in \mathbb{R}^p \times \mathbb{R}^k$ is a random sample from the following model:

$$\mathbb{E}(y_k \mid \mathbf{x}) = f_k(\mathbf{x}), \quad \forall k = 1, \ldots, K.$$

- **Classification**: Suppose $(\mathbf{x}_i, y_i) \in \mathbb{R}^p \times \{1, \ldots, K\}$ is a random sample from the following model:

$$\mathbb{P}(Y = k \mid \mathbf{x}) = \frac{\exp\left[f_k\left(\mathbf{x}_i\right)\right]}{\sum_{l=1}^{K} \exp\left[f_l\left(\mathbf{x}_i\right)\right]} \quad \text{subject to } f_K \equiv 0 \text{ or } \sum_{k=1}^{K} f_k \equiv 1.$$

And we further suppose the relation between $\mathbf{x}$ and $\mathbf{y}$ is as follows (some lines neglected).

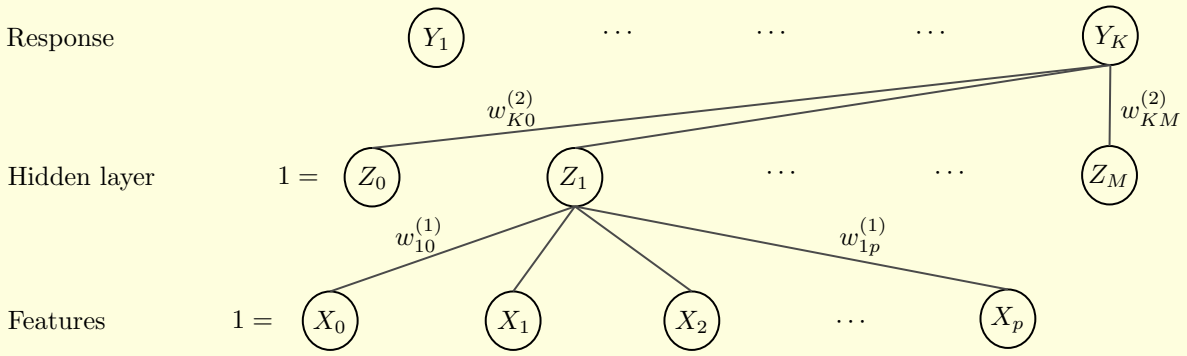


Figure 1: An example of single layer network

Specifically, we assume there exists a hidden layer of M variables (M is prespecified), $Z_1, \ldots, Z_M$, which are determined by the following functions:

$$\forall m = 1, \ldots, M: \quad Z_m = \sigma\left(w_{m0}^{(1)} + \mathbf{x}^\top \boldsymbol{w}_m^{(1)}\right),$$

where $\mathbf{x} = \begin{pmatrix} x_1 & \cdots & x_p \end{pmatrix}$ and $\boldsymbol{w}_m^{(1)} = \begin{pmatrix} w_{m1}^{(1)} & \cdots & w_{mp}^{(1)} \end{pmatrix}$ constitute unknown linear combinations of observed features, and $\sigma(\cdot)$ is a specified function, called the *activation function*. And the latent variables Z_m's are connected with responses Y_k's as follows

$$\forall k = 1, \ldots, K: \quad f_k(\mathbf{x}) = w_{k0}^{(2)} + \boldsymbol{z}^\top \boldsymbol{w}_k^{(2)},$$

where $\boldsymbol{z} = \begin{pmatrix} z_1 & \cdots & z_M \end{pmatrix}$ and $\boldsymbol{w}_m^{(2)} = \begin{pmatrix} w_{k1}^{(2)} & \cdots & w_{kM}^{(2)} \end{pmatrix}$ constitute unknown linear combinations of

latent variables.

Remark. There are few things worth noticing in the model.

(a) The activation function $\sigma(\cdot)$ can be different for different layers or different variables. But this would not affect the model much because the true functions are not known anyway. So, a common choice for the function is the following ones, called the *sigmoid function* and *rectified linear unit* (ReLu) :

$$\sigma(\nu) = \frac{e^{\nu}}{1+e^{\nu}} \text{ and } \sigma(\nu) = \max(0, \nu).$$

(b) In general, we choose all activation functions to be the same for the simplicity in the following estimation. Though this reduces the flexibility of the model, we can enlarge M to compensate.

(c) Though $\mathbf{x}^{\top}\boldsymbol{w}$ is a linear combination of observed features, it is not essentially PCA because the coefficients learning of PCA only depends on $\mathbb{X}$ (unsupervised learning), however the learning of neural network or projection pursuit depends on not only $\mathbb{X}$ but also $\mathbb{Y}$ (supervised learning).

(d) To add the hidden layer, we can add another layer $Z_1^{(2)}, \ldots, Z_{M_2}^{(2)}$ which is determined by $\sigma(\cdot)$ of the first layer $\boldsymbol{Z}^{(1)}$ as input.

5.15.1 Comparisons between Models

As we see, different models assume the same on the top, i.e.

$$\mathbb{E}(y_k \mid \mathbf{x}) = f_k(\mathbf{x}) \quad \text{or} \quad \mathbb{P}(Y = k \mid \mathbf{x}) = \frac{\exp\left[f_k\left(\mathbf{x}_i\right)\right]}{\sum_{l=1}^{K} \exp\left[f_l\left(\mathbf{x}_i\right)\right]},$$

depending on its a classification setting or regression setting. What differs models is the assumptions on f_k's. Supposing

- f_k's are from a RKHS, the model is kernel ridge/logistic regression;
- f_k's are additive smooth functions, the model is AM/GAM;
- f_k's are step functions of $\mathbf{x}$, the model is decision tree;
- f_k's are weighted average of trees, the model is Boosting.
- there exists a hidden layer $Z'm$ influencing the model, this is neural network/projection pursuit.

5.15.2 Estimation

We take single hidden layer neural network as an example, and suppose the number of latent variables M and activation function $\sigma(\cdot)$ are prespecified. Then unknown parameters include

$$\boldsymbol{\theta} = \begin{cases} \forall m = 1, \ldots, M: & w_{m0}^{(1)} \quad \cdots \quad w_{mp}^{(1)}, \\ \forall k = 1, \ldots, K: & w_{k0}^{(2)} \quad \cdots \quad w_{kM}^{(2)}. \end{cases}$$

Suppose we have the sample $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \ldots, N$. Normally, the loss function are chosen to be

$$\text{regression:} \quad L(\boldsymbol{\theta}) = \sum_{k=1}^{K} \sum_{i=1}^{N} \Big[y_{ik} - f_k(\mathbf{x}_i) \Big]^2,$$

$$\text{classification:} \quad L(\boldsymbol{\theta}) = -\sum_{k=1}^{K} \sum_{i=1}^{N} y_{ik} \log \Big[\mathbb{P}(y_i = k \mid \mathbf{x}_i) \Big].$$

The general idea is to minimize the above loss function with respect to the parameters by gradient descent methods, which is also called *backpropagation* in machine learning.

Case Study: Regression Neural Network

Now, let's delve into the derivation of the regression setting in the single layer neural network. Since the loss function can viewed as a sum of loss in every single observation, we can denote the loss of i-th observation by

$$L_i(\boldsymbol{\theta}) = \sum_{k=1}^{K} \Big[y_{ik} - f_k(\mathbf{x}_i) \Big]^2, \text{ where } f_k(\mathbf{x}_i) = w_{k0}^{(2)} + \boldsymbol{z}_i^\top \boldsymbol{w}_k^{(2)} \text{ and } z_{im} = \sigma\left(w_{m0}^{(1)} + \mathbf{x}_i^\top \boldsymbol{w}_m^{(1)} \right).$$

Therefore, if we fix the $\underline{i\text{-th}}$ observation, $\underline{k\text{-th}}$ component response, and $\underline{m\text{-th}}$ latent variable z_{im}, the partial derivative w.r.t. the coefficients on the second layer is

$$\frac{\partial L_i(\boldsymbol{\theta})}{\partial w_{km}^{(2)}} = \frac{\partial L_i(\boldsymbol{\theta})}{\partial f_k(\mathbf{x}_i)} \cdot \frac{\partial f_k(\mathbf{x}_i)}{\partial w_{km}^{(2)}} = -2\Big[y_{ik} - f_k(\mathbf{x}_i) \Big] \cdot z_{im}, \quad \forall i, k, m. \tag{5.19}$$

Similarly if we fix the $\underline{i\text{-th}}$ observation, $\underline{m\text{-th}}$ latent variable z_{im}, and $\underline{l\text{-th}}$ feature (observed) variable x_{ml}, the partial derivative w.r.t. the coefficients on the first layer is

$$\begin{aligned} \frac{\partial L_i(\boldsymbol{\theta})}{\partial w_{ml}^{(1)}} &= \sum_{k=1}^{K} \frac{\partial L_i(\boldsymbol{\theta})}{\partial f_k(\mathbf{x}_i)} \cdot \frac{\partial f_k(\mathbf{x}_i)}{\partial w_{ml}^{(1)}} \\ &= \sum_{k=1}^{K} -2\Big[y_{ik} - f_k(\mathbf{x}_i) \Big] \cdot w_{km}^{(2)} \sigma'\Big(w_{m0}^{(1)} + \mathbf{x}_i^\top \boldsymbol{w}_m^{(1)} \Big) x_{il}, \quad \forall i, m, l, \end{aligned} \tag{5.20}$$

where the $\sum_{k=1}^{K}$ and the second equality is from the model structure. For convenience, denote

$$\begin{aligned} \delta_{ki} &= \frac{\partial L_i(\boldsymbol{\theta})}{\partial f_k(\mathbf{x}_i)} = -2\Big[y_{ik} - f_k(\mathbf{x}_i) \Big], \\ s_{mi} &= \sum_{k=1}^{K} \frac{\partial L_i(\boldsymbol{\theta})}{\partial f_k(\mathbf{x}_i)} w_{km}^{(2)} \sigma'\Big(w_{m0}^{(1)} + \mathbf{x}_i^\top \boldsymbol{w}_m^{(1)} \Big) = \sum_{k=1}^{K} -2\Big[y_{ik} - f_k(\mathbf{x}_i) \Big] w_{km}^{(2)} \sigma'\Big(w_{m0}^{(1)} + \mathbf{x}_i^\top \boldsymbol{w}_m^{(1)} \Big). \end{aligned}$$

The two partial derivatives (5.19), (5.20) then follow that

$$\frac{\partial L_i(\boldsymbol{\theta})}{\partial w_{km}^{(2)}} = \delta_{ki} z_{mi} \text{ and } \frac{\partial L_i(\boldsymbol{\theta})}{\partial w_{ml}^{(1)}} = s_{mi} x_{il}.$$

It remains to utilize the gradient descent methods to optimize it, which is the next algorithm.

Algorithm 19 (Backpropagation)**.** Given the sample $(\mathbf{x}_i, \mathbf{y}_i)$, $i = 1, \ldots, N$ and initial parameter $\boldsymbol{\theta}^{(0)}$.

(a) Compute the gradients at t-th step by plugging $\boldsymbol{\theta}^{(t)}$ in the RHS of

$$\delta_{ki}^{(t)} = -2\Big[y_{ik} - f_k^{(t)}(\mathbf{x}_i)\Big],$$
$$s_{mi}^{(t)} = \sum_{k=1}^{K} -2\Big[y_{ik} - f_k^{(t)}(\mathbf{x}_i)\Big] w_{km}^{(2)\sim(t)} \sigma'\Big(w_{m0}^{(1)\sim(t)} + \mathbf{x}_i^\top \boldsymbol{w}_m^{(1)\sim(t)}\Big).$$

(b) Update $\boldsymbol{\theta}^{(t+1)}$ by gradient descent methods:

$$w_{km}^{(2)\sim(t+1)} \leftarrow w_{km}^{(2)\sim(t)} - \rho^{(t)} \sum_{i=1}^{N} \left.\frac{\partial L_i(\boldsymbol{\theta})}{\partial w_{km}^{(2)}}\right|_{(t)} = w_{km}^{(2)\sim(t)} - \rho^{(t)} \sum_{i=1}^{N} \delta_{ki}^{(t)} z_{mi}^{(t)},$$
$$w_{ml}^{(1)\sim(t+1)} \leftarrow w_{ml}^{(1)\sim(t)} - \rho^{(t)} \sum_{i=1}^{N} \left.\frac{\partial L_i(\boldsymbol{\theta})}{\partial w_{ml}^{(1)}}\right|_{(t)} = w_{km}^{(2)\sim(t)} - \rho^{(t)} \sum_{i=1}^{N} s_{mi}^{(t)} x_{il},$$

where $\rho^{(t)}$ is prespecified learning rate at step t.

(c) Repeat until convergence.

Remark. There are few things worth improved as follows.

(a) To avoid complicated computation, we can use the stochastic gradient descent idea, replacing $\sum_{i=1}^{N}$ by $\sum_{i \in \text{subset}^{(t)}}$.

(b) To avoid overfitting, we can add ridge penalties for every parameters, also called *weight decay*:

$$\min_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) + \lambda J(\boldsymbol{\theta}), \quad \text{where } J(\boldsymbol{\theta}) = \sum_{k,m} \Big[w_{km}^{(2)}\Big]^2 + \sum_{m,l} \Big[w_{ml}^{(1)}\Big]^2,$$

and $L(\boldsymbol{\theta})$ is the loss function according to the setting, λ is a tuning parameter. Or, we can use the early stopping idea.

5.16 Support Vector Machine

This note focus on the classification method - *Support Vector Machine*. For more reference, check §4.5, 12 of *Elements of Statistical Learning*, §7 of *Pattern Recognition and Machine Learning*.

We focus on the derivation of binary case now, $\mathcal{Y} = \{-1, 1\}$ with $\mathbf{x} \in \mathbb{R}^p$. The key idea of SVM is to find the optimal separating hyperplane, which maximizes the distance (also called *margin*) to the closest point from either class. And the two notions, optimal and hyperplane, are worth further explanation.

Definition 8 (Hyperplane)**.** A separating (linear) hyperplane $\mathcal{L}$ in $\mathbb{R}^p$ is a collection of points in the following form

$$\mathcal{L} = \{\mathbf{x} \in \mathbb{R} : \beta_0 + \boldsymbol{\beta}^\top \mathbf{x} = 0\}, \quad \text{where } \beta_0 \in \mathbb{R} \text{ and } \boldsymbol{\beta} \in \mathbb{R}^p.$$

In mathematics. it is also referred as the *affine* hyperplane.

Definition 9 (Margin)**.** For a hyperplane $\mathcal{L}$, the its margin M is defined to be the sum of smallest distances to the closest point from either class, i.e.

$$M = d(\mathbf{x}_1, \mathcal{L}) + d(\mathbf{x}_2, \mathcal{L}),$$

where $\mathbf{x}_1, \mathbf{x}_2$ are two closest points from two classes.

Example. Here are two examples of given hyperplanes, sample, and margins in $\mathbb{R}^2$ and $\mathbb{R}^3$.

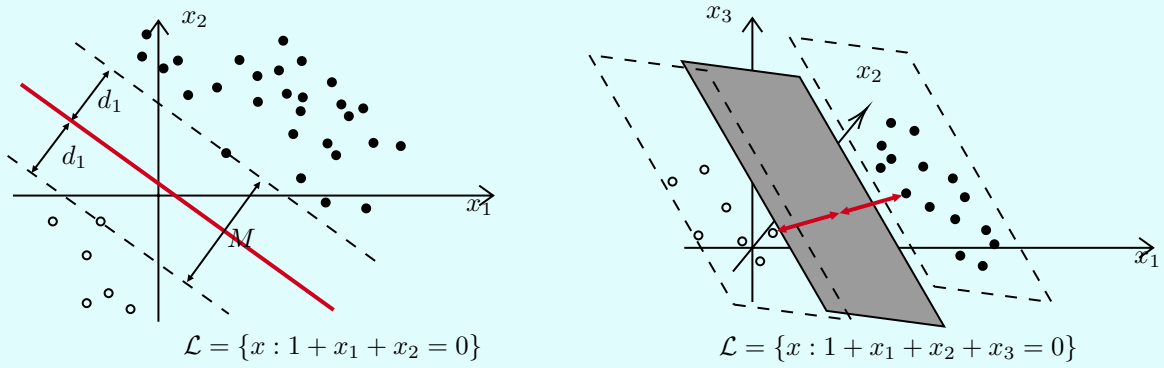


Figure 1: Examples of hyperplane and margin

Remark. There are several things worth noticing in the examples.

- A hyperplane of $\mathbb{R}^2$ is a line, and of $\mathbb{R}^3$ is a plane. But it is usually not a subspace, since it does not contain the origin.
- Given $\boldsymbol{\beta} \in \mathbb{R}^p$, we can translate parallel the hyperplane by varying the intercept $\beta_0 \in \mathbb{R}$. But we choose the one that makes the distance to two classes the same as in the $\mathbb{R}^2$ example, i.e. $d(\mathbf{x}_1, \mathcal{L}) = d(\mathbf{x}_2, \mathcal{L})$.
- Once the hyperplane is given, denoted by $f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}$ for convenience, the points on either side would have different sign when plugging in $\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}$. So, the classification rule is

$$y = \begin{cases} 1 & \text{if } f(\mathbf{x}) > 0, \\ -1 & \text{if } f(\mathbf{x}) < 0. \end{cases}$$

With these preliminaries, we can introduce SVM formally now, starting with the ideal case, which means the 2 classes could be separated by a hyperplane. But note that, sometimes they cannot.

5.17 Separable Case SVM

5.17.1 Optimization Formation

For any hyperplanes $\mathcal{L}$, first we calculate what it the distance of any point to it. Suppose there are two point $\mathbf{x}_1, \mathbf{x}_2$ in it. Then by definition, we know

$$\begin{cases} \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_1 = 0 \\ \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_2 = 0 \end{cases} \quad \Rightarrow \quad \boldsymbol{\beta}^\top (\mathbf{x}_1 - \mathbf{x}_2) = 0 \quad \Rightarrow \quad \boldsymbol{\beta} \perp \mathcal{L},$$

i.e. $\boldsymbol{\beta} \in \mathbb{R}^p$ (not including β_0) is a vector in $\mathbb{R}^p$ perpendicular to the hyperplane $\mathcal{L}$, as shown below.

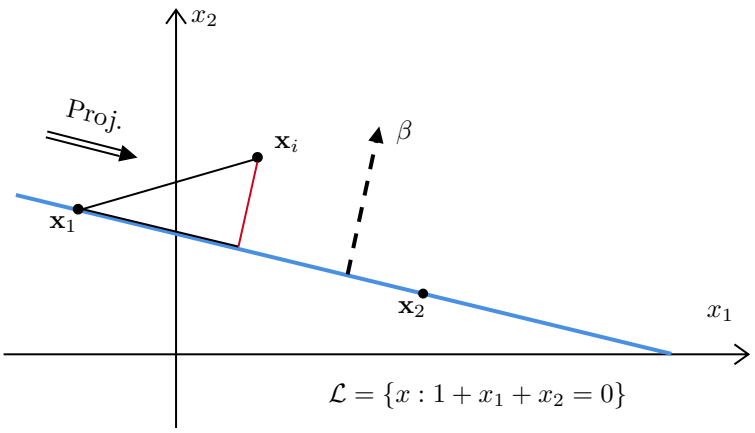


Figure 2: Illustration in a $\mathbb{R}^2$ hyperplane

Then the signed distance of a sample point $\mathbf{x}_i$ to $\mathcal{L}$ should be the length of the projection of $(\mathbf{x}_i - \mathbf{x}_1)$ on $\boldsymbol{\beta}$, i.e.

$$d_{\text{sign}}(\mathbf{x}_i, \mathcal{L}) = ||\text{Proj}_{\boldsymbol{\beta}}(\mathbf{x}_i - \mathbf{x}_1)|| = \left|\left| \frac{\boldsymbol{\beta}\boldsymbol{\beta}^\top(\mathbf{x}_i - \mathbf{x}_1)}{||\boldsymbol{\beta}||^2} \right|\right| = \frac{1}{||\boldsymbol{\beta}||}(\boldsymbol{\beta}^\top \mathbf{x}_i - \boldsymbol{\beta}^\top \mathbf{x}_1).$$

With $\mathbf{x}_1$ on $\mathcal{L}$, we can use $\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_1 = 0$ to replace $\boldsymbol{\beta}^\top \mathbf{x}_1$ in the signed distance, and it follows that

$$d_{\text{sign}}(\mathbf{x}_i, \mathcal{L}) = \frac{1}{||\boldsymbol{\beta}||}(\boldsymbol{\beta}^\top \mathbf{x}_i + \beta_0) = \frac{1}{||\boldsymbol{\beta}||} f(\mathbf{x}_i),$$

where the sign comes from $f(\mathbf{x}_i)$. If we use y_i to denote its classification result, i.e.

$$y = \begin{cases} 1 & \text{if } f(\mathbf{x}) > 0, \\ -1 & \text{if } f(\mathbf{x}) < 0, \end{cases}$$

we can derive an expression for the distance of any point $\mathbf{x}_i \in \mathbb{R}^p$ to $\mathcal{L}$:

$$d(\mathbf{x}_i, \mathcal{L}) = \frac{f(\mathbf{x}_i) y_i}{||\boldsymbol{\beta}||}.$$

Recall the notation of margin M, and we can define the SVM formally now.

Definition 10 (Support vector machine)**.** Suppose $(\mathbf{x}_i, Y_i)$, $i = 1, \ldots, N$ is a random sample in $\mathbb{R}^p \times \{-1, 1\}$. The support vector machine is a hyperplane in $\mathbb{R}^p$ solving the following optimization problem:

$$\max_{\beta_0, \boldsymbol{\beta}} M \quad \text{subject to} \quad \frac{y_i(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0)}{||\boldsymbol{\beta}||} \geq M, \quad \forall i = 1, \ldots, N.$$

Intuitively, this means we are looking for a hyperplane such that maximizes the margin, and it corresponds to what we want in the beginning. Since $\boldsymbol{\beta}$ is a direction vector for the hyperplane, we do not care about its

length. For convenience, we further set $||\boldsymbol{\beta}|| = 1/M$, and we want

$$\max_{\beta_0,\boldsymbol{\beta}} \frac{1}{||\boldsymbol{\beta}||} \iff \min_{\beta_0,\boldsymbol{\beta}} ||\boldsymbol{\beta}|| \iff \min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2,$$

which leads us to the **primal problem** of SVM:

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 \quad \text{subject to} \quad y_i(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1, \quad \forall i = 1, \ldots, N.$$

Remark. Though the function to be minimized $||\boldsymbol{\beta}||$ does not contain β_0, the constraints contain β_0, which plays a role in the optimization. And this is a standard convex problem (definition by Wikipedia). This is why we can optimize it easily.

5.17.2 Convex Optimization

Primal Problem

Using Lagrange function, we can simplify our primal problem to a conventional optimization form, which helps solve it using optimization methods (for more, refer *STATS - 606*).

Problem Setup (SVM Primal problem)**.** The primal problem of SVM,

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 \quad \text{subject to} \quad y_i(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1, \quad \forall i = 1, \ldots, N,$$

introduces the primal Lagrange function

$$L_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) := \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \Big[y_i \left(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0 \right) - 1 \Big],$$

where $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \cdots & \alpha_N \end{pmatrix}^\top$ with all non-negative elements. Then solving the primal problem is equivalent to solve the following problem:

$$\min_{\beta_0,\boldsymbol{\beta},\boldsymbol{\alpha}} L_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) \quad \text{subject to} \quad \alpha_i \geq 0, \quad \forall i = 1, \ldots, N.$$

The reason to do the above way is a focus of §1.2.3 of this note, thereby omitted here. Now, we continue further simplify this primal problem to the dual problem. Taking derivative of L_P w.r.t. β_0 and $\boldsymbol{\beta}$ leads to

$$\frac{\partial L_P}{\partial \beta_0} = -\sum_{i=1}^{N} \alpha_i y_i \quad \Rightarrow \quad \sum_{i=1}^{N} \alpha_i^* y_i = 0, \tag{5.21}$$

$$\frac{\partial L_P}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \quad \Rightarrow \quad \boldsymbol{\beta}^* = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i. \tag{5.22}$$

Plugging the two expression back into the L_P, leads to

$$\begin{aligned} L_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) &= \frac{1}{2}\left|\left|\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i\right|\right|^2 - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i^\top \left[\sum_{j=1}^{N} \alpha_j y_j \mathbf{x}_j\right] - \sum_{i=1}^{N} \alpha_i y_i \beta_0 + \sum_{i=1}^{N} \alpha_i \\ &= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{subject to} \quad \alpha_i \geq 0 \ \text{ and } \ \sum_{i=1}^{N} \alpha_i y_i = 0. \end{aligned}$$

Dual Problem

And this is what we will handle in the end, the dual problem.

Problem Setup (SVM Dual problem)**.** The primal problem of SVM,

$$\min_{\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}} L_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) \quad \text{subject to} \quad \alpha_i \geq 0, \quad \forall i = 1, \ldots, N,$$

leads to the dual Lagrange function

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{subject to} \quad \alpha_i \geq 0 \ \text{ and } \ \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Then solving the solving the primal problem is equivalent to solve

$$\max_{\boldsymbol{\alpha}} L_D(\boldsymbol{\alpha}).$$

We already can solve for the best hyperplane $\hat{f}(\mathbf{x})$ by the above, but we also want to know the reason of switch max and min, and better interpret the results. These lead us to the following discussion.

Constrained Optimization

In this subsection, we forget the notation of β_0 in the SVM, and suppose all parameters are contained in $\boldsymbol{\beta} \in \mathbb{R}^p$, which is a more general case. We consider a primal problem

$$p^* = \min_{\boldsymbol{\beta} \in \mathbb{R}^p} f(\boldsymbol{\beta}) \quad \text{subject to} \quad \mathbf{g}(\boldsymbol{\beta}) = \begin{cases} g_1(\boldsymbol{\beta}) \leq 0 \\ \vdots \\ g_m(\boldsymbol{\beta}) \leq 0 \end{cases} \tag{5.23}$$

i.e., minimizing the scalar $f(\boldsymbol{\beta})$ w.r.t. a p-dimensional vector subject to m inequality constraints. If we introduce the Lagrange function and define

$$L_P(\boldsymbol{\beta}, \boldsymbol{\lambda}) = f(\boldsymbol{\beta}) + \boldsymbol{\lambda}^\top \mathbf{g}(\boldsymbol{\beta}),$$

where $\boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 & \cdots & \lambda_m \end{pmatrix}^\top$ with all non-negative elements, we can transfer the primal problem to be

$$p^* = \min_{\boldsymbol{\beta}: \mathbf{g}(\boldsymbol{\beta}) \leq 0} f(\boldsymbol{\beta}) = \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left[f(\boldsymbol{\beta}) + \max_{\boldsymbol{\lambda} \geq 0} \boldsymbol{\lambda}^\top \mathbf{g}(\boldsymbol{\beta}) \right] = \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \max_{\boldsymbol{\lambda} \geq 0} \left[f(\boldsymbol{\beta}) + \boldsymbol{\lambda}^\top \mathbf{g}(\boldsymbol{\beta}) \right] = \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \max_{\boldsymbol{\lambda} \geq 0} L_P(\boldsymbol{\beta}, \boldsymbol{\lambda}).$$

Remark. We can simply explain the reason for the second equality. Suppose some $g_i(\boldsymbol{\beta}) > 0$. Then this $\boldsymbol{\beta}$ could not be a optimizer in the LHS. But since $\lambda_i > 0$, in the RHS, the maximization of $\lambda_i g_i(\boldsymbol{\beta})$ can go to ∞. So, it is also not included in any possible solution of RHS.

As we shall see, under some regularity conditions, we can switch the order of min and max:

$$p^* = \min_{\boldsymbol{\beta}\in\mathbb{R}^p} \max_{\boldsymbol{\lambda}\geq 0} L_P(\boldsymbol{\beta}, \boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq 0} \min_{\boldsymbol{\beta}\in\mathbb{R}^p} L_P(\boldsymbol{\beta}, \boldsymbol{\lambda}).$$

If we define

$$L_D(\boldsymbol{\lambda}) = \min_{\boldsymbol{\beta}\in\mathbb{R}^p} L_P(\boldsymbol{\beta}, \boldsymbol{\lambda}),$$

under the regularity conditions, the primal problem is equivalent to the dual problem:

$$d^* = \max_{\boldsymbol{\lambda}\geq 0} L_D(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq 0} \min_{\boldsymbol{\beta}\in\mathbb{R}^p} L_P(\boldsymbol{\beta}, \boldsymbol{\lambda}) = p^*.$$

It remains to consider the order of optimizations. In general, $p^* \geq d^*$, and we call $(p^* - d^*)$, the *duality gap*. For simplicity, in the following discussion, we write $\mathbf{g}(\boldsymbol{\beta}) < 0$ or $\boldsymbol{\lambda} \geq 0$ to mean the inequalities for all elements in this vector.

Theorem 11. *In the above setting and notations, if $f(\cdot)$ and $\mathbf{g}(\cdot)$ are both convex, and there exists $\boldsymbol{\beta} \in \mathbb{R}^p$ such that $\mathbf{g}(\boldsymbol{\beta}) < 0$, then $p^* = d^*$.*

The above theorem guarantees the order switch of max and min in our SVM problem. Furthermore, we have a stronger theorem to help the optimization.

Theorem 12 (Karush-Kuhn-Tucker condition)**.** *If f and g are both differentiable and convex, and there exists $\boldsymbol{\beta} \in \mathbb{R}^p$ such that $\mathbf{g}(\boldsymbol{\beta}) < 0$, then the solution to the optimization $(\boldsymbol{\beta}^*, \boldsymbol{\lambda}^*)$ satisfies*

(a) $\mathbf{g}(\boldsymbol{\beta}^*) \leq 0$ and $\boldsymbol{\lambda}^* \geq 0$,

(b) for all $j = 1, \ldots, m$, we have $\lambda_j^* g_j(\boldsymbol{\beta}^*) = 0$,

(c) $\nabla_{\boldsymbol{\beta}} f(\boldsymbol{\beta}^*) + (\boldsymbol{\lambda}^*)^\top \nabla_{\boldsymbol{\beta}} \mathbf{g}(\boldsymbol{\beta}^*) = \mathbf{0}$, in other words, $\nabla_{\boldsymbol{\beta}} L_P(\boldsymbol{\beta}^*, \boldsymbol{\lambda}^*) = \mathbf{0}$.

Solution Properties to SVM

Now in this section, we focus on the properties of the solution. Recall the SVM primal problem:

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 \quad \text{subject to} \quad y_i\left(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0\right) \geq 1, \quad \forall i = 1, \ldots, N,$$

and comparing to (5.23), we can find

$$f(\beta_0, \boldsymbol{\beta}) = \frac{1}{2}||\boldsymbol{\beta}||^2 \text{ and } g_i(\beta_0, \boldsymbol{\beta}) = 1 - y_i\left(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0\right) \leq 0.$$

Both f and $\mathbf{g}$ are convex and differentiable, satisfying the KKT condition. Therefore, we can apply the solution properties to the SVM problem. By Theorem 5,

(a) for all $i = 1, \ldots, N$, we have $y_i\left(\mathbf{x}_i^\top \boldsymbol{\beta}^* + \beta_0^*\right) \geq 1$ and $\alpha_i^* \geq 0$,

(b) for all $i = 1, \ldots, N$, we have $\alpha_i^* \left\{ y_i \left[(\boldsymbol{\beta}^*)^\top \mathbf{x}_i + \beta_0^* \right] - 1 \right\} = 0$,

(c) recalling the derivatives (5.21) and (5.22), we have

$$\sum_{i=1}^{N} \alpha_i^* y_i = 0 \text{ and } \boldsymbol{\beta}^* = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i.$$

Now we can have some discussions concerning these results. Fix the observation i. From (b), if $\alpha_i^* > 0$, then

$$y_i \left[(\boldsymbol{\beta}^*)^\top \mathbf{x}_i + \beta_0^* \right] = 1$$

And recall that the original optimization is

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2} ||\boldsymbol{\beta}||^2 \quad \text{subject to} \quad y_i(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1, \quad \forall i = 1, \ldots, N,$$

with the equality holding when $\mathbf{x}_i$ is on the boundary of classification.

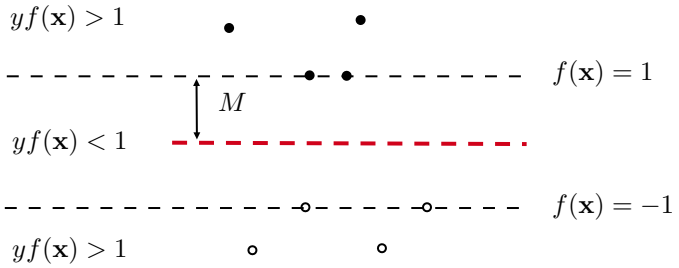


Figure 3: SVM Boundary

Similarly, by (c), if $y_i \left[(\boldsymbol{\beta}^*)^\top \mathbf{x}_i + \beta_0^* \right] > 1$, then $\mathbf{x}_i$ is not on the boundary and at the same time $\alpha_i = 0$. This means whether an observation $\mathbf{x}_i$ is on the boundary or not can be judged by its corresponding α_i^*, i.e. for all observations $i = 1, \ldots, N$,

$$\alpha_i = \begin{cases} 0 & \text{if } \mathbf{x}_i \text{ is not on the boundary,} \\ 1 & \text{if } \mathbf{x}_i \text{ is on the boundary.} \end{cases}$$

Then combining the (c), it follows that

$$\boldsymbol{\beta}^* = \sum_{i=1}^{N} \alpha_i^* y_i \mathbf{x}_i = \sum_{i: \alpha_i^* > 0} \alpha_i^* y_i \mathbf{x}_i = \sum_{i \in \text{Boundary}} \alpha_i^* y_i \mathbf{x}_i.$$

Formally speaking, SVM classification decision is decided by a linear combination of observations on the boundary. And we call those points on the boundary, the *support points/vectors*, denoted by a set $\mathbb{S}$.

Estimation of the Intercept

We can understand the hyperplane $f(\mathbf{x}) = \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}$ as a direction vector $\boldsymbol{\beta} \in \mathbb{R}$ and a location parameter β_0. By previous sections, we have estimated the direction $\boldsymbol{\beta}^*$, and now it remains the location. Since

$$\forall i \in \mathbb{S}: \quad y_i \left(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i \right) = 1 \;\; \Rightarrow \;\; y_i^2 \left(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i \right) = y_i \;\; \Rightarrow \;\; \sum_{i \in \mathbb{S}} \left(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i \right) = \sum_{i \in \mathbb{S}} y_i,$$

which leads to a natural estimator of β_0^* by plugging in $\boldsymbol{\beta}^*$ and the following algorithm:

$$\beta_0^* = \frac{\sum_{i\in\mathbb{S}} \left(y_i - (\boldsymbol{\beta}^*)^\top \mathbf{x}_i\right)}{|\mathbb{S}|}.$$

Algorithm 20. The procedure to do SVM is:

(a) Optimize the dual problem for $\boldsymbol{\alpha}^*$:

$$\max_{\boldsymbol{\alpha}} L_D(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{subject to } \alpha_i \geq 0 \text{ and } \sum_{i=1}^N \alpha_i y_i = 0.$$

(b) Calculate the estimated hyperplane by

$$\beta^* = \sum_{i=1}^N \alpha_i^* y_i \mathbf{x}_i \text{ and } \beta_0^* = \frac{\sum_{i\in\mathbb{S}} \left[y_i - (\boldsymbol{\beta}^*)^\top \mathbf{x}_i\right]}{|\mathbb{S}|}$$

(c) The final classification rule is

$$f^*(\mathbf{x}) = \beta_0^* + (\boldsymbol{\beta}^*)^\top \mathbf{x} \begin{cases} > 0 & \Rightarrow \quad \hat{y} = 1 \\ < 0 & \Rightarrow \quad \hat{y} = -1 \end{cases}$$

5.17.3 Kernelization of SVM

You may wonder, if we only care about the solution but not the properties or interpretations, why don't we directly solve the primal problem by minimizing

$$L_P\left(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}\right) := \frac{1}{2}\|\boldsymbol{\beta}\|^2 - \sum_{i=1}^N \alpha_i \left[y_i\left(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0\right) - 1\right], \tag{5.24}$$

but rather transfer to another dual problem by maximizing

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{i=1}^N\sum_{j=1}^N \alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{subject to} \quad \alpha_i \geq 0 \quad \text{and} \sum_{i=1}^N \alpha_i y_i = 0. \tag{5.25}$$

Are we making things more complicated? The answer is, if we only care about the solution, either optimization is fine. But the dual problem naturally leads us to the kernelization of SVM with $\mathbf{x}_i^\top \mathbf{x}_j$.

Specifying a kernel function $k(\cdot,\cdot) : \mathbb{R}^p \times \mathbb{R}^p \mapsto \mathbb{R}$, similarly to the kernel PCA/Ridge, we project

$$\forall i = 1, \ldots, N: \quad \mathbf{x}_i \mapsto \phi(\mathbf{x}) := k_{\mathbf{x}_i} := k(\mathbf{x}_i, \cdot), \text{ where } k_{\mathbf{x}_i}(\cdot) : \mathbb{R}^p \mapsto \mathbb{R},$$

and find a linear hyperplane on $\mathscr{H}_k$, which can be understood as a non-linear hyperplane in $\mathbb{R}^p$. So, in the primal problem, if we replace

$$\mathbf{x}_i \Rightarrow \phi(\mathbf{x}_i) \text{ and } \mathbf{x}_i^\top \mathbf{x}_j \Rightarrow \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\rangle = k(\mathbf{x}_i, \mathbf{x}_j),$$

the primal problem (5.24) needs modification to

$$L_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\alpha}) := \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{N} \alpha_i \Big[y_i \Big(\langle \phi(\mathbf{x}_i), \boldsymbol{\beta} \rangle + \beta_0 \Big) - 1 \Big]$$

Treating all $\phi(\mathbf{x}_i)$ as constants, by the same derivation, the dual problem (5.25) needs modification to

$$L_D(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{subject to} \quad \alpha_i \geq 0 \quad \text{and} \sum_{i=1}^{N} \alpha_i y_i = 0.$$

Similar to the Algorithm 6, once $\boldsymbol{\alpha}$ is estimated, everything else is done. But one needs to note that though

$$\boldsymbol{\beta}^* = \sum_{i=1}^{N} \alpha_i^* y_i \phi(\mathbf{x}_i), \text{ where } \phi \text{ is an element in } \mathscr{H}_k,$$

and we don't know its true form, this would not bring any trouble since the estimation of the intercept is

$$\begin{aligned} \beta_0^* &= \frac{\sum_{i \in \mathbb{S}} \left[y_i - (\boldsymbol{\beta}^*)^\top \phi(\mathbf{x}_i) \right]}{|\mathbb{S}|} = \frac{\sum_{i \in \mathbb{S}} \Big(y_i - \langle \boldsymbol{\beta}^*, \phi(\mathbf{x}_i) \rangle \Big)}{|\mathbb{S}|} \\ &= \frac{\sum_{i \in \mathbb{S}} \Big(y_i - \Big\langle \sum_{i=1}^{N} \alpha_j^* y_j \phi(\mathbf{x}_j), \phi(\mathbf{x}_i) \Big\rangle \Big)}{|\mathbb{S}|} \\ &= \frac{\sum_{i \in \mathbb{S}} \Big(y_i - \sum_{i=1}^{N} \alpha_j^* y_j k(\mathbf{x}_i, \mathbf{x}_j) \Big)}{|\mathbb{S}|}, \end{aligned}$$

and for any new observation $\mathbf{x} \in \mathbb{R}^p$, the classification rule is

$$\begin{aligned} f(\mathbf{x}) &= \beta_0^* + (\boldsymbol{\beta}^*)^\top \phi(\mathbf{x}) = \beta_0^* + \langle \boldsymbol{\beta}^*, \phi(\mathbf{x}) \rangle = \beta_0^* + \left\langle \sum_{i=1}^{N} \alpha_i^* y_i \phi(\mathbf{x}_i), \phi(\mathbf{x}) \right\rangle \\ &= \beta_0^* + \sum_{i=1}^{N} \alpha_i^* y_i \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle = \beta_0^* + \sum_{i=1}^{N} \alpha_i^* y_i k(\mathbf{x}_i, \mathbf{x}). \end{aligned}$$

5.18 Non-separable Case SVM

The SVM searches for a linear hyperplane in $\mathbb{R}^p$, but it is possible that there does not exist any hyperplane that could perfectly split two classes (do not consider kernel SVM). In the algorithm, it behaves no solution to the optimization, both L_P and L_D. In this case, we search a hyperplane which maximizes the margin by allowing some points on the wrong side of the margin, as shown in Figure 4.

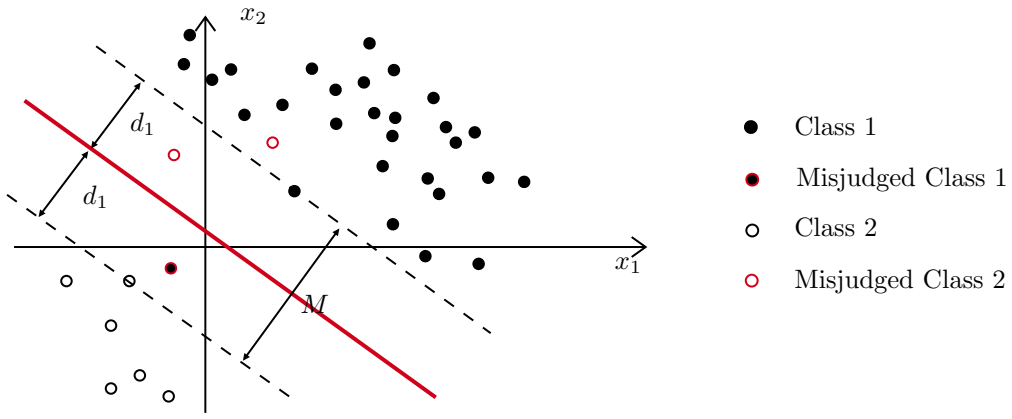


Figure 4: Non-separable SVM

In practical, we do the following primal optimization.

Definition 11 (Non-separable SVM)**.** Suppose $(\mathbf{x}_i, Y_i)$, $i = 1, \ldots, N$ is a random sample in $\mathbb{R}^p \times \{-1, 1\}$. The non-separable SVM,

$$f^*(\mathbf{x}) = \beta_0^* + (\boldsymbol{\beta}^*)^\top \mathbf{x},$$

is a hyperplane in $\mathbb{R}^p$ solving the following optimization problem:

$$\min_{\beta_0, \boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 \quad \text{subject to} \quad y_i(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0) \geq 1 - \xi_i, \quad \forall i = 1, \ldots, N,$$

where the slack variable is defined to be

$$\boldsymbol{\xi}_{N\times 1} = \begin{pmatrix} \xi_1 \\ \vdots \\ \xi_N \end{pmatrix} \text{ with } \forall i = 1, \ldots, N \quad \xi_i \geq 0 \text{ and } \sum_{i=1}^N \xi_i \leq K.$$

Recall that in the separable SVM, we know the two cases

$$y_i f(\mathbf{x}_i) = y_i \left(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0\right) \geq 1 \text{ with } \begin{cases} = 1 & \mathbf{x}_i \text{ in on the boundary,} \\ > 1 & \mathbf{x}_i \text{ in not on it and correctly classified.} \end{cases}$$

Now in the non-separable case, we allow $y_i f(\mathbf{x}_i) \geq 1 - \xi_i$, it allows three cases

$$y_i f(\mathbf{x}_i) = y_i \left(\mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0\right) \begin{cases} = 1 & \mathbf{x}_i \text{ is on the boundary,} \\ > 1 & \mathbf{x}_i \text{ is not on it and correctly classified,} \\ < 1 & \mathbf{x}_i \text{ is inside the margin of its class.} \end{cases}$$

And ξ_i can be viewed as a tolerance of how much $\mathbf{x}_i$ is allowed to be wrong, and $\sum_{i=1}^N \xi_i$ is a total wrong flexibility.

5.18.1 Optimization and Intuition behind

By convex optimization, we can equivalently do

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\sum_{i=1}^{N}\xi_i \quad \text{subject to} \quad \xi_i \geq 0 \text{ and } y_i\left(\mathbf{x}_i^\top\boldsymbol{\beta}+\beta_0\right) \geq 1-\xi_i, \quad \forall i = 1,\ldots,N. \tag{5.26}$$

And this is exactly what we do in the programming.

Remark. There are few things we can notice from this form.

(a) The constant $C \geq 0$ is a cost/tuning parameter to be chosen, which controls the trade-off between maximizing margin and minimizing error.

(b) The separable case SVM corresponds to $C = \infty$, and leading to $\xi_i = 0$.

(c) Large C tends to minimize the training error, but the margin may be narrow. But note that, if C is too large, it may has no solution as the separable case.

(d) Small C tends to maximize the margin distance, but the error may be more.

In fact, the above optimization has one equivalence, which helps us understand the intuition behind the SVM. But remember, what we do in the programming is (5.26).

Proposition 13. *The optimization (5.26) is equivalent to*

$$\min_{\beta_0,\boldsymbol{\beta}} \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N} L\Big(y_i, f(\mathbf{x}_i)\Big), \quad \textit{where} \quad L\Big(y_i, f(\mathbf{x}_i)\Big) := \max\Big(1-y_i f(\mathbf{x}_i), 0\Big)$$

is the Hinge loss function.

Proof. We consider two cases. If $1 - y_i f(\mathbf{x}_i) \leq 0$, then constraints

$$\begin{cases} \xi_i \geq 0 \\ \xi_i \geq 1 - y_i f(\mathbf{x}_i) \end{cases} \quad \Rightarrow \quad \xi_i \geq 0 \quad \Rightarrow \quad \text{to make } C\sum_{i=1}^{N}\xi_i \text{ min}, \quad \text{take } \xi_i = 0.$$

If $1 - y_i f(\mathbf{x}_i) > 0$, then the above argument leads to $\xi_i \geq 1 - y_i f(\mathbf{x}_i)$. To attain the min, we take $\xi_i = 1 - y_i f(\mathbf{x}_i)$. The two cases leads to

$$\xi_i = \max\Big(1-y_i f(\mathbf{x}_i), 0\Big) = L\Big(y_i, f(\mathbf{x}_i)\Big).$$

□

Changing some constants, the SVM optimization we do in essence is equivalent to the following:

$$\hat{f}(\mathbf{x}) \leftarrow \arg\min_{\beta_0,\boldsymbol{\beta}} \sum_{i=1}^{N} L\Big(y_i, f(\mathbf{x}_i)\Big) + \lambda||\boldsymbol{\beta}||^2,$$

where

$$L\Big(y_i, f(\mathbf{x}_i)\Big) = \max\Big(1-y_i f(\mathbf{x}_i), 0\Big) \text{ and } f(x) = \mathbb{P}(Y = 1 \mid \mathbf{x}) - \frac{1}{2}.$$

Recall that the (kernel) logistic regression is doing

$$\hat{f}(\mathbf{x}) \leftarrow \arg\min_{f\in\mathscr{H}_k} \sum_{i=1}^{N} L\Big(y_i, f(\mathbf{x}_i)\Big) + \lambda||f||^2_{\mathscr{H}_k},$$

where

$$L\Big(y_i, f(\mathbf{x}_i)\Big) = \frac{\log\left(1+e^{-\tilde{y}_i f(\mathbf{x}_i)}\right)}{\log 2} \quad \text{and} \quad \hat{f}(x) = \log\frac{\mathbb{P}(Y=1\mid \mathbf{x})}{\mathbb{P}(Y=-1\mid \mathbf{x})}.$$

What we are trying to understand is, different classification methods are doing the same thing: modeling the probability by some function f, and minimizing w.r.t. this function f by some loss function L and the data.

5.18.2 Estimation

In the previous subsection, we mentioned the optimization is based on (5.26). Introducing Lagrange multipliers, we reach the primal problem.

Problem Setup (Non-separable SVM Primal problem)**.** The primal Lagrange function of non-separable SVM is

$$L_P(\beta_0, \boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}, \boldsymbol{\mu}) := \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i\Big[y_i\left(\beta_0+\boldsymbol{\beta}^\top\mathbf{x}_i\right) - (1-\xi_i)\Big] - \sum_{i=1}^{N}\mu_i\xi_i,$$

where $\boldsymbol{\alpha}_{N\times 1}$ and $\boldsymbol{\mu}_{N\times 1}$ are Lagrange multipliers with $\alpha_i \geq 0$ and $\mu_i \geq 0$ for all $i = 1, \ldots, N$.

The derivation is almost the same as in the separable SVM. Taking partial derivatives, we get

$$\frac{\partial L_P}{\partial \beta_0} = -\sum_{i=1}^{N}\alpha_i y_i \quad \Rightarrow \quad \sum_{i=1}^{N}\alpha_i^* y_i = 0 \tag{5.27}$$

$$\frac{\partial L_P}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i \quad \Rightarrow \quad \boldsymbol{\beta}^* = \sum_{i=1}^{N}\alpha_i^* y_i \mathbf{x}_i \tag{5.28}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i \quad \Rightarrow \quad C = \alpha_i^* + \mu_i^* \quad \forall i = 1, \ldots, N.$$

Plugging them back leads to the dual problem.

Problem Setup (Non-separable SVM Dual problem)**.** The dual Lagrange function of non-separable SVM is

$$L_D(\boldsymbol{\alpha}) := \sum_{i=1}^{N}\alpha_i - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \quad \text{subject to} \quad \sum_{i=1}^{N}\alpha_i y_i = 0 \ \text{ and } \ 0 \leq \alpha_i \leq C, \ \ \forall i.$$

Using the same procedure in separable SVM, we estimate $\boldsymbol{\alpha}^*$ first, then calculate $\boldsymbol{\beta}^*$ and β_0^*.

5.18.3 Solution Properties

The above optimization is also a convex one with KKT conditions. So, the following properties follows:

(a) for all $i = 1, \ldots, N$, we have

$$\begin{cases} y_i \left(\beta_0^* + \mathbf{x}_i^\top \boldsymbol{\beta}^*\right) \geq 1 - \xi_i^*, \\ \xi_i \geq 0, \end{cases} \quad \text{and} \quad \begin{cases} \alpha_i^* \geq 0, \\ \mu_i^* \geq 0. \end{cases}$$

(b) for all $i = 1, \ldots, N$, we have

$$\alpha_i^* \Big[y_i(\beta_0^* + \mathbf{x}_i^\top \boldsymbol{\beta}^*) - (1 - \xi_i^*) \Big] = 0 \text{ and } \mu_i \xi_i = 0.$$

(c) recalling the derivatives (5.27) and (5.28), we have for all $i = 1, \ldots, N$

$$\sum_{i=1}^N \alpha_i^* y_i = 0, \quad \boldsymbol{\beta}^* = \sum_{i=1}^N \alpha_i^* y_i \mathbf{x}_i, \text{ and } \mu_i^* + \alpha_i^* = C.$$

Now we will explore the relation between α_i and boundary conditions as we did in separable SVM. Starting the discussion of α_i.

(α-i) If $\alpha_i = 0$, meaning that this observation $\mathbf{x}_i$ has no contribution to $\boldsymbol{\beta}^*$, then

$$\alpha_i = 0 \overset{\text{(c3)}}{\Longrightarrow} \mu_i = C \overset{\text{(b2)}}{\Longrightarrow} \xi_i = 0 \overset{\text{(a1)}}{\Longrightarrow} y_i \left(\beta_0^* + \mathbf{x}_i^\top \boldsymbol{\beta}^*\right) \geq 1,$$

we conclude that $\mathbf{x}_i$ is either on the margin, or on the right side.

(α-ii) If $0 < \alpha_i < C$, then

$$0 < \alpha_i < C \overset{\text{(c3)}}{\Longrightarrow} \mu_i > 0 \overset{\text{(c2)}}{\Longrightarrow} \xi_i = 0 \overset{\text{(a1)}}{\Longrightarrow} y_i \left(\beta_0^* + \mathbf{x}_i^\top \boldsymbol{\beta}^*\right) = 1,$$

we conclude that $\mathbf{x}_i$ is on the margin.

(α-iii) If $\alpha_i = C$, then

$$\alpha_i = C \overset{\text{(c3)}}{\Longrightarrow} \mu_i = 0 \overset{\text{(b2)}}{\Longrightarrow} \xi_i \geq 0 \begin{cases} > 0 \overset{\text{(b1)}}{\Longrightarrow} y_i \left(\beta_0^* + \mathbf{x}_i^\top \boldsymbol{\beta}^*\right) < 1 \Rightarrow \text{ wrong side}, \\ = 0 \overset{\text{(b1)}}{\Longrightarrow} y_i \left(\beta_0^* + \mathbf{x}_i^\top \boldsymbol{\beta}^*\right) = 1 \Rightarrow \text{ on the margin}. \end{cases}$$

Conversely, let's discuss by $y_i f(\mathbf{x}_i)$.

($yf(\mathbf{x})$−i) If $y_i f(\mathbf{x}_i) > 1$, then

$$y_i f(\mathbf{x}_i) > 1 \overset{\text{(b1)}}{\Longrightarrow} \alpha_i = 0 \Rightarrow \text{ no contribution to } \boldsymbol{\beta}^*.$$

($yf(\mathbf{x})$−ii) If $y_i f(\mathbf{x}_i) = 1$, then $\mathbf{x}_i$ is on the margin, but we know nothing about α_i:

$$y_i f(\mathbf{x}_i) = 1 \Rightarrow 0 \leq \alpha_i \leq C.$$

There might exist $\mathbf{x}_i$ on the margin, but $\alpha_i = 0$, i.e. it still has no contribution to $\boldsymbol{\beta}^*$.

($yf(\mathbf{x})-$iii) If $y_i f(\mathbf{x}_i) < 1$, then

$$y_i f(\mathbf{x}_i) < 1 \overset{\text{(a1)}}{\Longrightarrow} \xi_i > 0 \overset{\text{(b2)}}{\Longrightarrow} \mu_i = 0 \overset{\text{(c3)}}{\Longrightarrow} \alpha_i^* = C,$$

we conclude $\mathbf{x}_i$ contributes to the SVM.

By the discussion of $yf(\mathbf{x})$, to sum up, partial points on the margin ($yf(\mathbf{x}) - ii$) and all points inside the margin ($yf(\mathbf{x}) - iii$) are supports points, i.e. $\alpha_i^* > 0$.

5.18.4 Kernalization of SVM

Using the same idea, we can generalized the non-separable SVM by kernel methods, assuming

$$\log \frac{\mathbb{P}_x(Y=1)}{\mathbb{P}_x(Y=-1)} = f(\mathbf{x}), \text{ where } f \in \mathcal{H}_k.$$

If we further use Hinge loss function on the sample, the primal problem is

$$f^* \leftarrow \arg\min_{f \in \mathcal{H}_k} \sum_{i=1}^N L\Big(y_i, f(\mathbf{x}_i)\Big) + \lambda ||f||_{\mathcal{H}_k},$$

and the dual problem is

$$\boldsymbol{\alpha}^* \leftarrow \arg\max_{\boldsymbol{\alpha}} \sum_{i=1}^N \alpha_i - \frac{1}{2}\sum_{ij} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j), \quad \text{subject to} \quad \begin{cases} 0 \le \boldsymbol{\alpha} \le C, \\ \sum_{i=1}^N \alpha_i y_i = 0. \end{cases}$$

5.19 Generalization of Hinge Loss

Since the Hinge loss is introduced, we can utilize it in the regression setting. Given the sample $(\mathbf{x}_i, y_i)$ for $i = 1, \ldots, N$, the model assumption is

$$\mathbb{E}_{\mathbf{x}_i} Y_i = f(\mathbf{x}_i) = \beta_0 + \boldsymbol{\beta}^\top \mathbf{x}_i,$$

and we estimate the parameters by the Hinge loss

$$\min_{\beta_0, \boldsymbol{\beta}} \sum_{i=1}^N L_\epsilon\Big(y_i, f(\mathbf{x}_i)\Big) + \lambda ||\boldsymbol{\beta}||^2, \text{ where } L_\epsilon\Big(y_i, f(\mathbf{x}_i)\Big) = \begin{cases} 0 & \text{if } |y_i - f(\mathbf{x}_i)| \le \epsilon \\ |y_i - f(\mathbf{x}_i)| - \epsilon & \text{o.w.} \end{cases}$$

When $\epsilon \to 0$, the loss function becomes L_1 loss, which gives the median regression.

Remark. Note the difference between L_1 loss(median reg) and L_1 penalty(lasso). For more, refer §7 of *Pattern Recognition and Machine Learning*.

Chapter 6

Clustering

Clustering is another statistical problem, where we want to group a collection of objects $\mathbf{x}_i$'s into clusters, assuming each cluster represents a subpopulation within a heterogeneous population. But different from classification, the cluster labels $z_i \in \{1, 2, \ldots, K\}$'s are unknown in clustering analysis. In this note, we focus on data-driven method, more specifically the K-means, and in the next note, we will turn to model-based clustering, more specifically the Gaussian mixture model and latent class model.

From now on we assume the feature $\mathbf{x} \in \mathbb{R}^p$ and the label $z \in \{1, \ldots, K\}$, where the total number of clusters K is assumed to be known.

6.1 K-means Formulation

6.1.1 Distance Measure

A key part of data-driven classification is the choice of loss function. Similarly, a central part in data-driven clustering methods is the notion of dissimilarity/similarity measure between $\mathbf{x}$'s. A natural choice is the squared Euclidean distance

$$D(\mathbf{x}_i, \mathbf{x}_j) = ||\mathbf{x}_i - \mathbf{x}_j||_2^2.$$

If we want more generality, we can define different distances for each dimension of the feature, and use their weighted sum as the total difference:

$$D(\mathbf{x}_i, \mathbf{x}_j) = \sum_{l=1}^{p} w_l d_l(x_{il}, x_{jl}), \text{ with } \sum_{l=1}^{p} w_l = 1 \text{ and } w_l \geq 0.$$

Now we assume the distance measure and the total number of clusters are given, and start our derivation.

6.1.2 Estimating Labels

As a natural idea for clustering, we want observations in the same cluster to be close. So, treating the labels $\mathbf{z} = (z_1, \ldots, z_N)$ as input variables, we want to minimize

$$W(\mathbf{z}) := \frac{1}{2} \sum_{k=1}^{K} \left[\frac{1}{N_k} \sum_{i:z_i=k} \sum_{j:z_j=k} D(\mathbf{x}_i, \mathbf{x}_j) \right], \text{ where } N_k = \sum_{i=1}^{N} \mathbb{1}(z_i = k).$$

In particular, if we consider the squared Euclidean distance, then

$$W(\mathbf{z}) = \frac{1}{2}\sum_{k=1}^{K}\left[\frac{1}{N_k}\sum_{\substack{i:z_i=k\\ j:z_j=k}}||(\mathbf{x}_i-\boldsymbol{\mu}_k)-(\mathbf{x}_j-\boldsymbol{\mu}_k)||^2\right], \text{ where } \boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i:z_i=k}\mathbf{x}_i.$$

Note that $\boldsymbol{\mu}_k$ also depends on our input $\mathbf{z}$. After simplification, we can show under squared Euclidean distance becomes

$$W(\mathbf{z}) = \sum_{k=1}^{K}\sum_{i:z_i=k}||\mathbf{x}_i-\boldsymbol{\mu}_k||^2.$$

This is why $W(\mathbf{z})$ is also called the *Within cluster Sum of Squared errors* (WSS).

Definition 12 (Sum of Squares)**.** Given a sample and their labels $(\mathbf{x}_i, z_i)$, $i = 1,\dots,N$, the within cluster sum of squares is

$$W(\mathbf{z}) = \sum_{k=1}^{K}\sum_{i:z_i=k}||\mathbf{x}_i-\boldsymbol{\mu}_k||^2,$$

the Between cluster Sum of Squared errors (BSS) is

$$B(\mathbf{z}) = \sum_{k=1}^{K}N_k||\boldsymbol{\mu}_k-\bar{\mathbf{x}}||^2,$$

and the Total Sum of Squared errors is

$$T = W(\mathbf{z}) + B(\mathbf{z}) = \sum_{i=1}^{N}||\mathbf{x}_i-\bar{\mathbf{x}}||^2.$$

Remark. The BSS and WSS depend on the label input $\mathbf{z}$, however the TSS does not and it only depends on the data. One popular algorithm is as follows.

Algorithm 21 (K-means)**.** Given the sample $\mathbf{x}_i$, $i = 1,\dots,N$.

(a) Initialize cluster means $\boldsymbol{\mu}_k$ for $k = 1,\dots,K$ at some positions.

(b) Repeat the following until convergence:

b1) For $i = 1,\dots,N$, update the label z_i by

$$z_i \leftarrow \arg\min_k ||\mathbf{x}_i-\boldsymbol{\mu}_k||^2.$$

b2) For $k = 1,\dots,K$, update the label center $\boldsymbol{\mu}_k$ by

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i:z_i=k}\mathbf{x}_i, \text{ where } N_k \text{ also depends on updated } z_i.$$

Unfortunately, this optimization is not convex, so even the above algorithm depends greatly on the initial values and have multiple local solutions. There are two feasible ways to solve this problem.

- In practice, we may consider multiple random initial values and choose the best one giving the minimized $W(\mathbf{z})$.
- There are some proposed methods to choose the initial value, such as K-means ++.

6.2 Selection of K

In this section, we no longer treat the total number of clusters K as unknown, and introduce several ways to choose it.

6.2.1 Elbow Method

A very exploratory and intuitive way is to draw the WSS v.s. k plot as follows.

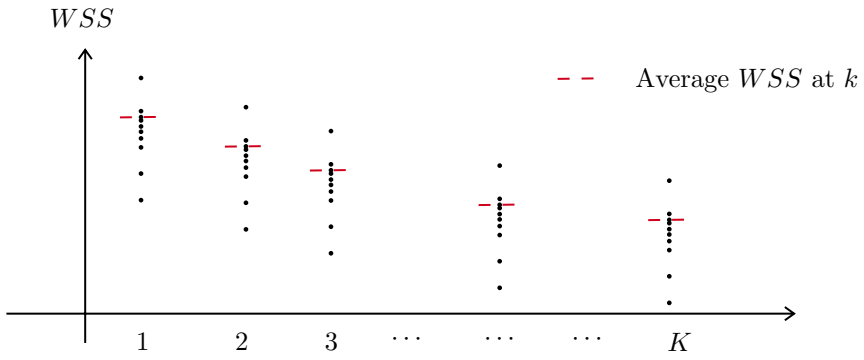


Figure 1: Elbow plot over different k

We choose a k such that we think after that there is no significant decrease in WSS.

6.2.2 Silhouette Method

For every observation $\mathbf{x}_i$, we define a similarity measure for $\mathbf{x}_i$ to its own cluster as

$$a_i = \frac{1}{|c_i| - 1} \sum_{j:z_j=z_i} D(\mathbf{x}_i, \mathbf{x}_j), \quad \text{where } c_i \text{ is } i\text{-th obs's cluster size,}$$

a dissimilarity measure for $\mathbf{x}_i$ to other clusters as

$$b_i = \min_{k \neq z_i} \frac{1}{|c_k|} \sum_{j:z_j=k} D(\mathbf{x}_i, \mathbf{x}_j),$$

and the Silhouette value of $\mathbf{x}_i$ as

$$s(i) = \begin{cases} \frac{b_i - a_i}{\max(a_i, b_i)} & \text{if } |c_i| > 1 \\ 0 & \text{if } |c_i| = 1. \end{cases}$$

In this way, we can use $s_i \in [-1, 1]$ to somehow reflect whether $\mathbf{x}_i$ is closer its own cluster, or other clusters. The more closer to 1 means $\mathbf{x}_i$ is more closer to its partners and farther to other clusters.

Then we compute the mean Silhouette value

$$\bar{S}_k = \frac{\sum_{i=1}^{N} s_k(i)}{N} \quad \Rightarrow \quad \hat{k} = \arg\min_k \bar{S}_k$$

as the best number of clusters.

6.2.3 Gap Statistic

The *gap statistic* is proposed by Tibshirani in 2001, which has a theoretical guarantee to choose the number of clusters. For more, refer *Estimating the Number of Clusters in a Data Set via the Gap Statistic* .

6.3 Hierarchical Clustering

Hierarchical clustering is another method to estimate the labels. It is calculated using a greedy algorithm, which does not require the specification of K, and produces hierarchical representation in which the clusters at each level are created by merging clusters at lower levels.

Intuitively, there are two ways of hierarchical.

- Bottom-up (Agglomerative): start with N clusters (i.e. each $\mathbf{x}_i$ being a single cluster), and merge recursively the closest two clusters into one.
- Top-down (Divisive): start with one cluster with all $\mathbf{x}_i$'s, recursively divide one existing cluster into two.

6.3.1 Optimization View

A great benefit of hierarchical clustering is that it(bottom-up) can be viewed as

$$\min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_N} \sum_{i=1}^{N} ||\mathbf{x}_i - \boldsymbol{\mu}_i||^2 + \lambda \sum_{i<j} \mathbb{1}(\boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j),$$

each $\boldsymbol{\mu}_i$ represents the center of cluster which $\mathbf{x}_i$ belongs to. If $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2$, it means $\mathbf{x}_1$ and $\mathbf{x}_2$ are in the same cluster. And we can note that

- when $\lambda = 0$, the minimization has no constraints, so every $\boldsymbol{\mu}_i = \mathbf{x}_i$ (N clusters) gets it minimized at 0.
- when $\lambda \to \infty$, then the penalty dominates the whole term if it is not 0, forcing all observations to be in the same one cluster.

Since this is also a non-convex penalty, we can use a convex fusion penalty to replace $\mathbb{1}(\boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j)$:

$$\min_{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_N} \frac{1}{2} \sum_{i=1}^{N} ||\mathbf{x}_i - \boldsymbol{\mu}_i||^2 + \lambda \sum_{i<j} ||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_q,$$

where $||\cdot||_q$ is the L_q penalty. For example,

- when $q = 1$, it is $\lambda \sum_{i<j} |\boldsymbol{\mu}_i - \boldsymbol{\mu}_j|$, and also called the *fused lasso penalty*.
- when $q = 2$, it is $\lambda \sum_{i<j} ||\boldsymbol{\mu}_i - \boldsymbol{\mu}_j||_2$, and also called the *fused group lasso penalty*.

6.4 Kernel K-means

Just as the kernel PCA setting, in the following Figure 2, it is not possible to use normal K-means to get the blue and red clusters. The only way is to project it to a higher dimensional feature space.

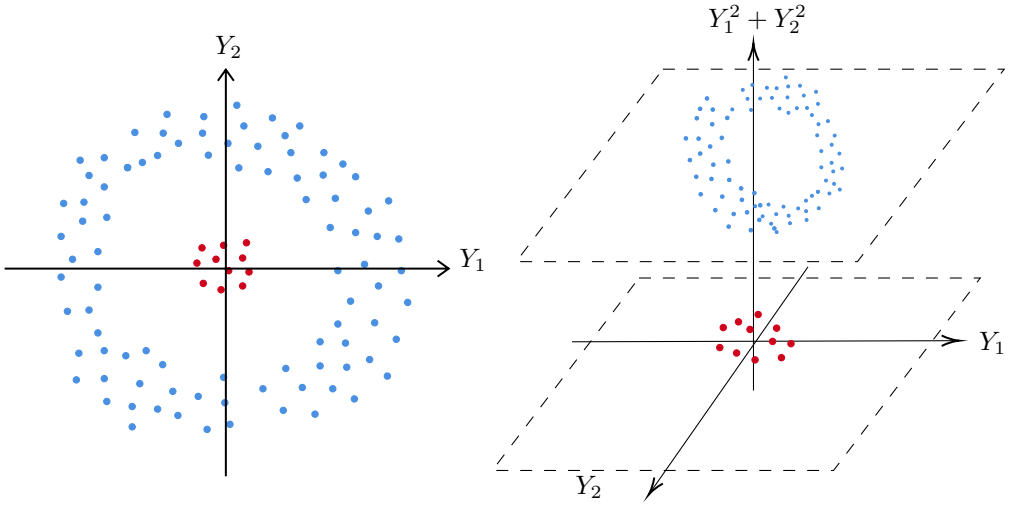


Figure 2: Higher dimension projection

So, what we want to do is K-means for $\phi(\mathbf{x}_i)$'s. A key part is how to choose the distance measure in $\mathcal{H}_k$. A common but not rigorously proved way is to use

$$d^2_{\mathcal{H}_k}\Big(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)\Big) = k(\mathbf{x}_i, \mathbf{x}_i) - 2k(\mathbf{x}_i, \mathbf{x}_j) + k(\mathbf{x}_j, \mathbf{x}_j).$$

Based on the discussion of the common K-means, we do

$$\mathbf{z} \leftarrow \arg\min_{\mathbf{z}} \sum_{k=1}^{K} \sum_{i:z_i=k} ||\phi(\mathbf{x}_i) - \boldsymbol{\mu}_k||^2_{\mathcal{H}_k}, \quad \text{where } \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i:z_i=k} \phi(\mathbf{x}_i).$$

After expanding the inner product, we can get the simplified version:

$$\sum_{k=1}^{K} \sum_{i=1}^{N} \mathbb{1}(z_i = k) \left[k(\mathbf{x}_i, \mathbf{x}_i) - \frac{2}{N_k} \sum_{j=1}^{N} \mathbb{1}(z_j = k) k(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{N_k^2} \sum_{j=1}^{N} \sum_{l=1}^{N} \mathbb{1}(z_j = k, z_l = k) k(\mathbf{x}_j, \mathbf{x}_l) \right]. \quad (6.1)$$

Algorithm 22 (Kernel K-means)**.** We can solve the above optimization by

(a) Initialize $\mathbf{z} = (z_1, \ldots, z_N)$.

(b) For each $\mathbf{x}_i$, find its new cluster label z_i by

$$z_i \leftarrow \arg\min_k ||\phi(\mathbf{x}_i) - \boldsymbol{\mu}_k||^2_{\mathcal{H}_k}, \quad \text{which can be expanded as the form (6.1).}$$

(c) Repeat until convergence.

Compared with for the data-driven clustering, the model-based clustering can provide us with better uncertainty measure. In this note, we focus on common model-based clustering methods. For more, refer *Model-Based Clustering and Classification for DS* book.

6.5 Model Assumption

In general, clustering can be considered as finite mixture models, assuming the data are from a mixture of distributions.

Assumption 25 (Mixture models)**.** Suppose $\mathbf{X} \in \mathbb{R}^p$ is one observation, and there exists a latent label $Z \in \{1, 2, \ldots, K\}$ such that

$$\begin{cases} Z \sim \text{Categorical}(\boldsymbol{\pi}_K), \\ \mathbf{X} \mid Z = k \sim \mathbb{P}(\mathbf{x} \mid Z = k, \boldsymbol{\theta}_k), \end{cases}$$

where the conditional probability $\mathbb{P}(\mathbf{x} \mid Z = k, \boldsymbol{\theta}_k)$ would be specified according to the problem setting.

Two famous and common examples are the *Gaussian mixture model* for continuous $\mathbf{x}$ and the *latent class model* for discrete $\mathbf{x}$.

(a) **Gaussian mixture model**: take $\mathbb{P}(\mathbf{x} \mid Z = k, \boldsymbol{\theta}_k) = \mathbf{N}_p(\mathbf{x} \mid \boldsymbol{\mu}_k, \Sigma_k)$.

(b) **Latent class model**: assume

$$\mathbf{X} = \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \quad \text{with} \quad \text{each component } X_i \in \{1, \ldots, L\} \text{ being } M \text{ categorical.}$$

Furthermore, we assume the conditional independence between components of $\mathbf{X}$,

$$X_i \,|Y \perp\!\!\!\perp X_j| \, Y, \quad \forall 1 \leq i \neq j \leq p.$$

and the conditional distribution $p_{ljk} := \mathbb{P}\left(X_j = l \mid Y = k\right)$

$$X_j \mid Z = k \sim \text{Categorical}_M\left(p_{ljk}\right), \quad \begin{cases} \forall \text{ categories of } X_j : l = 1, \ldots, M \\ \forall \text{ components of } \mathbf{X} : j = 1, \ldots, p \\ \forall \text{ categories of } Z : k = 1, \ldots, K \end{cases}$$

(c) For the mixed type data, which contains partly discrete and partly continuous features, we can assume conditional independence between components and do the same thing.

6.6 Estimation

6.6.1 Gaussian Mixture Model by EM

Suppose $\mathbf{x}_i, \ldots, \mathbf{x}_N$ is a random sample from the Gaussian mixture model, i.e.

Assumption 26 (Gaussian mixture model)**.** Suppose $\mathbf{X} \in \mathbb{R}^p$ is one observation, and there exists a latent label $Z \in \{1, 2, \dots, K\}$ such that

$$\begin{cases} Z \sim \text{Categorical}(\boldsymbol{\pi}_K), \\ \mathbf{X} \mid Z = k \sim \mathbf{N}_p(\boldsymbol{\mu}_k, \Sigma_k). \end{cases}$$

Then all parameters include $\boldsymbol{\theta} = \{\boldsymbol{\pi}_K, (\boldsymbol{\mu}_k, \Sigma_k), k = 1, \dots, K\}$ to be estimated. The marginal likelihood (which means the unwanted r.v. is integrated out) of the observed data is

$$\begin{aligned} \mathcal{L}(\boldsymbol{\theta}) &= \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_i, z_i = k \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) \\ &= \prod_{i=1}^{N} \sum_{k=1}^{K} p(\mathbf{x}_i \mid z_i = k, \boldsymbol{\mu}, \Sigma) p(z_i = k \mid \boldsymbol{\pi}) \\ &= \prod_{i=1}^{N} \sum_{k=1}^{K} \pi_k N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k). \end{aligned}$$

And the marginal log-likelihood of observed data is

$$\ell(\boldsymbol{\theta} \mid \mathbb{X}) = \sum_{i=1}^{N} \log \left\{ \sum_{k=1}^{K} \pi_k N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \right\}.$$

We can find the MLE using the above log-likelihood function by

- gradient ascent algorithm,
- EM algorithm.

And we mainly focus on the EM as an practice of Notes 5.1. First, we note that the complete data likelihood function is

$$\mathcal{L}_c(\boldsymbol{\theta} \mid \mathbb{X}, \mathbf{z}) = \prod_{i=1}^{N} p(\mathbf{x}_i, z_i \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid z_i, \boldsymbol{\theta}) p(z_i \mid \boldsymbol{\theta}) = \prod_{i=1}^{N} \prod_{k=1}^{K} \Big[\pi_k N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \Big]^{\mathbb{1}(z_i = k)},$$

where the last equality is the very common trick to deal with categorical variable. Then log-likelihood is

$$\log p(\mathbb{X}, \mathbf{z}) = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \Big[\log \pi_k + \log N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \Big]. \tag{6.2}$$

E-Step

Denote the distribution of the latent variable $\mathbf{z}$ at t-step by, $q_t(\mathbf{z}) = p(\mathbf{z} \mid \mathbb{X}, \hat{\boldsymbol{\theta}}^{(t)})$, which we will derive soon, and suppose it is known. The E-step is to compute by (6.2)

$$\begin{aligned} \mathbb{E}_{q_t} \Big\{ \log p(\mathbb{X}, \mathbf{z} \mid \boldsymbol{\theta}) \Big\} &= \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{E}_{q_t} \Big\{ \mathbb{1}(z_i = k) \Big[\log \pi_k + \log N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \Big] \Big\} \\ &= \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{P} \Big[z_i = k \mid \mathbb{X}, \boldsymbol{\theta}^{(t)} \Big] \Big[\log \pi_k + \log N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \Big]. \end{aligned} \tag{6.3}$$

The reason for the second equality is that the expectation is taken w.r.t. $\mathbf{z}$ conditioning on $\mathbb{X}, \boldsymbol{\theta}^{(t)}$, and $\Big[\log \pi_k + \log N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k)\Big]$ has nothing to do with $\mathbf{z}$, thereby is out the expectation.

Note that at t-th step, we know $\boldsymbol{\theta}^{(t)}$ and the distribution q_t, so we know for every $i = 1, \ldots, N$ and $k = 1, \ldots, K$ (we shall see the iteration of π has nothing to do with $\mathbf{x}_i$, so subscript i is omitted)

$$\pi_k^{(t)} := \mathbb{P}(z_i = k \mid \mathbf{x}_i, \boldsymbol{\theta}^{(t)}). \tag{6.4}$$

Hence, we denote by

$$\tau_i^{k(t)} := \mathbb{P}\Big[z_i = k \mid \mathbb{X}, \boldsymbol{\theta}^{(t)}\Big] = \mathbb{P}\Big(z_i = k \mid \mathbf{x}_i, \boldsymbol{\theta}^{(t)}\Big) = \frac{\mathbb{P}\left[\mathbf{x}_i, z_i = k \mid \boldsymbol{\theta}^{(t)}\right]}{\mathbb{P}\left[\mathbf{x}_i \mid \boldsymbol{\theta}^{(t)}\right]} = \frac{\mathbb{P}\left[z_i = k \mid \mathbf{x}_i, \boldsymbol{\theta}^{(t)}\right] \cdot p(\mathbf{x}_i \mid \boldsymbol{\theta}^{(t)})}{\sum_{k=1}^K \mathbb{P}\left[\mathbf{x}_i, \mathbf{z}_i = k \mid \boldsymbol{\theta}^{(t)}\right]},$$

where the second equality is due to independence between observations, and the last equality is calculating the marginal of the denominator. Further calculation leads to

$$\tau_i^{k(t)} = \frac{\mathbb{P}\left[z_i = k \mid \mathbf{x}_i, \boldsymbol{\theta}^{(t)}\right] \cdot p(\mathbf{x}_i \mid \boldsymbol{\theta}^{(t)})}{\sum_{k=1}^K \mathbb{P}\left[\mathbf{z}_i = k \mid \mathbf{x}_i, \boldsymbol{\theta}^{(t)}\right] \cdot p(\mathbf{x}_i \mid \boldsymbol{\theta}^{(t)})} = \frac{\pi_k^{(t)} N_p\Big(\mathbf{x}_i \mid \boldsymbol{\mu}_k^{(t)}, \Sigma_k^{(t)}\Big)}{\sum_{l=1}^K \pi_l^{(t)} N_p\Big(\mathbf{x}_i \mid \boldsymbol{\mu}_l^{(t)}, \Sigma_l^{(t)}\Big)},$$

where the second equality is from the notion (6.4). Now our E-step (6.3) becomes

$$\mathbb{E}_{q_t}\Big\{\log p(\mathbb{X}, \mathbf{z} \mid \boldsymbol{\theta})\Big\} = \sum_{i=1}^N \sum_{k=1}^K \tau_i^{k(t)} \Big[\log \pi_k + \log N_p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k)\Big].$$

M-Step

Keep in mind that τ's are all numbers when we are at t-th step, and what we want to do is

$$\begin{bmatrix} \boldsymbol{\pi}^{(t+1)} \\ \boldsymbol{\mu}^{(t+1)} \\ \Sigma^{(t+1)} \end{bmatrix} = \arg\max_{\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma} \sum_{i=1}^N \sum_{k=1}^K \tau_i^{k(t)} \Big[\log \pi_k - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) - \frac{1}{2}\log|\Sigma_k| - \frac{p}{2}\log(2\pi)\Big].$$

Note that the maximization w.r.t. π's are separated from the $\boldsymbol{\mu}$'s and Σ's. Now let's first focus on $\boldsymbol{\pi}$, where our goal is

$$\max_{\boldsymbol{\pi}} \sum_{i=1}^N \sum_{k=1}^K \tau_i^{k(t)} \log \pi_k = \max_{\boldsymbol{\pi}} \sum_{k=1}^K \sum_{i=1}^N \tau_i^{k(t)} \log \pi_k = \sum_{k=1}^K c_k \log \pi_k,$$

where for convenience, we denote by $c_k := \sum_{i=1}^N \tau_i^{k(t)}$, which is a known number for every $k = 1, \ldots, K$, and do not forget the construction

$$\sum_{k=1}^K \pi_k = 1 \;\text{ and }\; \sum_{k=1}^K c_k = \sum_{k=1}^K \sum_{i=1}^N \tau_i^{k(t)} = \sum_{i=1}^N \sum_{k=1}^K \mathbb{P}_{q_t}(Z_i = k) = \sum_{i=1}^N 1 = N.$$

By the constraints of π's, we solve

$$\min_{\boldsymbol{\pi}} -\sum_{k=1}^K c_k \log \pi_k \quad \text{subject to} \quad \boldsymbol{\pi} \succeq 0 \;\text{ and }\; \mathbf{1}^\top \boldsymbol{\pi} = 1.$$

Writing in Lagrangian function, it becomes

$$L(\boldsymbol{\pi}, \boldsymbol{\lambda}, \nu) = -\sum_{k=1}^{K} c_k \log \pi_k - \boldsymbol{\lambda}^\top \boldsymbol{\pi} + \nu \left(\mathbf{1}^\top \boldsymbol{\pi} - 1\right) \quad \text{where } \boldsymbol{\lambda} \in \mathbb{R}^K \succeq 0 \text{ and } \nu \in \mathbb{R} \geq 0.$$

Taking derivatives leads to for all $k = 1, \ldots, K$

$$\frac{\partial L}{\partial \pi_k} = -\frac{c_k}{\pi_k} - \lambda_k + \nu \quad \Rightarrow \quad \lambda_k = \nu - \frac{c_k}{\pi_k}. \tag{6.5}$$

And by KKT condition (b), the results must satisfy for all $k = 1, \ldots, K$

$$\begin{cases} \lambda_k \pi_k = 0 \\ \mathbf{1}^\top \boldsymbol{\pi} = 1 \end{cases} \quad \Rightarrow \quad \sum_{k=1}^{K} \left(\nu - \frac{c_k}{\pi_k}\right) \pi_k = \sum_{k=1}^{K} \lambda_k \pi_k = 0 \quad \Rightarrow \quad \sum_{k=1}^{K} c_k = \nu.$$

Now, to solve the equation, we need to note for all $k = 1, \ldots, K$, either $\lambda_k = 0$ or $\pi_k = 0$. If the solution for some k is $\pi_k^* = 0$, it happens only if $c_k = 0$ otherwise the objective function explodes. So, if for some $k = 1, \ldots, K$, we observe $c_k = 0$, then we can directly claim $\pi_k^* = 0$. Now we exclude those $c_k = 0$, then the solutions for all left $\pi_k^* > 0$, forcing $\lambda_k = 0$. By (6.5), we have for all $k =, 1 \ldots, K$

$$\frac{c_1}{\pi_1^*} = \frac{c_2}{\pi_2^*} = \cdots = \frac{c_K}{\pi_K^*} \quad \Rightarrow \quad \pi_k^* = \frac{c_k}{\sum_{k=1}^K c_k}.$$

In fact, the above optimization can be regarded as the **Multinomial MLE**, and we can remember this conclusion and use it when needed. And getting back to our setting, the solution is

$$\forall k = 1, \ldots, K: \quad \pi_k^{(t+1)} = \frac{\sum_{i=1}^N \tau_i^{k(t)}}{\sum_{k=1}^K \sum_{i=1}^N \tau_i^{k(t)}} = \frac{1}{N} \sum_{i=1}^{N} \tau_i^{k(t)},$$

which does not depend on $\mathbf{x}_i$ or index i, completing the claim before.

Then it remains to optimize w.r.t. the second part, $\boldsymbol{\mu}_k$'s and Σ_k's. Note that the problem

$$\max \sum_{i=1}^{N} \sum_{k=1}^{K} \tau_i^{k(t)} \Big[-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) - \frac{1}{2} \log |\Sigma_k| - \frac{p}{2} \log(2\pi) \Big].$$

can be separated into K optimizations, each of which only concerns $\boldsymbol{\mu}_k$ and Σ_k for $k = 1, \ldots, K$. Then, we fix one $k = 1, \ldots, K$, denoting by $c_i = \tau_i^{k(t)}$, optimizing

$$\max_{\boldsymbol{\mu}_k, \Sigma_k} \sum_{i=1}^{N} c_i \Big[-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_k)^\top \Sigma_k^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) - \frac{1}{2} \log |\Sigma_k| - \frac{p}{2} \log(2\pi) \Big],$$

which is a weighted Gaussian MLE type problem, the general result is

$$\hat{\boldsymbol{\mu}} = \frac{\sum_{i=1}^N c_i \mathbf{x}_i}{\sum_{i=1}^N c_i} \text{ and } \hat{\Sigma} = \frac{\sum_{i=1}^n c_i \left[\mathbf{x}_i - \hat{\boldsymbol{\mu}}\right]^\top \left[\mathbf{x}_i - \hat{\boldsymbol{\mu}}\right]}{\sum_{i=1}^N c_i}.$$

This result is intuitive and we can remember it. Back to our GMM setting, the solution is

$$\forall k = 1, \ldots, K: \quad \boldsymbol{\mu}_k^{(t+1)} = \frac{\sum_{i=1}^N \tau_i^{k(t)} \mathbf{x}_i}{\sum_{i=1}^N \tau_i^{k(t)}} \text{ and } \Sigma_k^{(t+1)} = \frac{\sum_{i=1}^N \tau_i^{k(t)} \left[\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)}\right]^\top \left[\mathbf{x}_i - \boldsymbol{\mu}_k^{(t+1)}\right]}{\sum_{i=1}^N \tau_i^{k(t)}}.$$

Viewing EM of GMM as Soft K-means

In fact, EM is a 'soft' version of K-means. making probabilistic assignment of $\mathbf{x}$'s to each cluster. We can see this if we fix $\Sigma_k = \sigma^2 \mathbf{I}$ and set $\sigma \to 0$, then EM updates becomes K-means as follows.

$$\begin{aligned}
\tau_i^{k(t)} &:= \mathbb{P}\left[z_i = k \mid \mathbb{X}, \boldsymbol{\theta}^{(t)}\right] \\
&= \frac{\pi_k^{(t)} N_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k^{(t)}, \Sigma_k^{(t)}\right)}{\sum_{l=1}^K \pi_l^{(t)} N_p\left(\mathbf{x}_i \mid \boldsymbol{\mu}_l^{(t)}, \Sigma_l^{(t)}\right)} \\
&= \frac{\pi_k^{(t)} \exp\left\{-\frac{1}{2\sigma^{(t)}} ||\mathbf{x}_i - \boldsymbol{\mu}_k^{(t)}||^2\right\}}{\sum_{l=1}^K \pi_l^{(t)} \exp\left\{-\frac{1}{2\sigma^{(t)}} ||\mathbf{x}_i - \boldsymbol{\mu}_l^{(t)}||^2\right\}} \\
&\to \begin{cases} 1 & \text{if } \ k = \arg\min_l ||\mathbf{x}_i - \boldsymbol{\mu}_l^{(t)}||^2 \\ 0 & \text{o.w..} \end{cases}
\end{aligned}$$

6.6.2 Latent Class Model by EM

To be formal, the assumption of latent class model is as follows.

Assumption 27 (Latent class model)**.** Suppose $(X_i, Z_i), i = 1, \ldots, N$ is a random sample from the following model:

$$Z \sim \text{Categorical}(\boldsymbol{\pi}) \quad \text{with} \quad \mathcal{Z} = \{1, 2, \ldots, K\} \text{ and is latent,}$$

and for $\mathbf{X} \in \mathbb{R}^p$,

$$\mathbf{X} = \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \quad \text{with each component } X_i \in \{1, \ldots, L\} \text{ being } L \text{ categorical.}$$

Furthermore, we assume the conditional independence between components of $\mathbf{X}$,

$$X_i \mid Z \perp\!\!\!\perp X_j \mid Z, \quad \forall\, 1 \le i \ne j \le p,$$

and the conditional distribution $p_{ljk} := \mathbb{P}(X_j = l \mid Z = k)$

$$X_j \mid Z = k \sim \text{Categorical}_L(p_{ljk}), \quad \begin{cases} \forall \text{ categories of } X_j\text{:} & l = 1, \ldots, L. \\ \forall \text{ components of } \mathbf{X}\text{:} & j = 1, \ldots, p. \\ \forall \text{ categories of } Z\text{:} & k = 1, \ldots, K. \end{cases}$$

The complete data likelihood function for one observation $(\mathbf{x}_i, z_i)$ is

$$\begin{aligned}\mathcal{L}_c(\boldsymbol{\theta} \mid \mathbf{x}_i, z_i) &= \mathbb{P}(\mathbf{X}_i = \mathbf{x}_i, Z_i = z_i \mid \boldsymbol{\theta}) \\ &= \mathbb{P}(\mathbf{X}_i = \mathbf{x}_i \mid Z_i = z_i, \boldsymbol{\theta}) \cdot \mathbb{P}(Z_i = z_i \mid \boldsymbol{\theta}) \\ &= \mathbb{P}(X_{i1} = x_{i1} \mid Z_i = z_i, \boldsymbol{\theta}) \cdots \mathbb{P}(X_{ip} = x_{ip} \mid Z_i = z_i, \boldsymbol{\theta}) \cdot \mathbb{P}(Z_i = z_i \mid \boldsymbol{\theta}) \\ &= \prod_{j=1}^{p} \mathbb{P}(X_{ij} = x_{ij} \mid Z_i = z_i, \boldsymbol{\theta}) \cdot \mathbb{P}(Z_i = z_i \mid \boldsymbol{\theta}), \end{aligned} \tag{6.6}$$

where we can further compute by indicator function that for every $j = 1, \ldots, p$:

$$\begin{aligned}\mathbb{P}(X_{ij} = x_{ij} \mid Z_i = z_i, \boldsymbol{\theta}) &= \prod_{k=1}^{K} \mathbb{P}(X_{ij} = x_{ij} \mid Z_i = k, \boldsymbol{\theta})^{\mathbb{1}(z_i = k)} \\ &= \prod_{k=1}^{K} \left(\prod_{l=1}^{L} \mathbb{P}(X_{ij} = l \mid Z_i = k, \boldsymbol{\theta})^{\mathbb{1}(x_{ij} = l)} \right)^{\mathbb{1}(z_i = k)} \\ &= \prod_{k=1}^{K} \prod_{l=1}^{L} p_{ljk}^{\mathbb{1}(x_{ij}=l)\mathbb{1}(z_i=k)} \end{aligned} \tag{6.7}$$

Then plugging (6.7) into (6.6), the complete data likelihood function is

$$\mathcal{L}_c(\boldsymbol{\theta} \mid \mathbf{x}_i, z_i) = \prod_{j=1}^{p} \prod_{k=1}^{K} \prod_{l=1}^{L} p_{ljk}^{\mathbb{1}(x_{ij}=l)\mathbb{1}(z_i=k)} \cdot \prod_{k=1}^{K} \pi_k^{\mathbb{1}(z_i=k)}.$$

Then we can compute the log-likelihood of complete data of all observations

$$\ell_c(\boldsymbol{\theta} \mid \mathbb{X}, \mathbf{z}) = \sum_{i=1}^{N} \sum_{j=1}^{p} \sum_{k=1}^{K} \sum_{l=1}^{L} \mathbb{1}(x_{ij} = l)\mathbb{1}(z_i = k) \log p_{ljk} + \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{1}(z_i = k) \log \pi_k. \tag{6.8}$$

E-Step

Suppose we know the t-th step parameters $\boldsymbol{\theta}^{(t)} = \left\{ \pi_k^{(t)}, p_{ljk}^{(t)}, \forall l, j, k \right\}$. Then we can compute the conditional distribution q_t of the latent variable $\mathbf{z}$, i.e.

$$\begin{aligned}\mathbb{P}(Z_i = k \mid \mathbb{X}, \boldsymbol{\theta}^{(t)}) &= \mathbb{P}(Z_i = k \mid \mathbf{x}_i, \boldsymbol{\theta}^{(t)}) \\ &= \frac{\mathbb{P}(Z_i = k, \mathbf{X} = \mathbf{x}_i \mid \boldsymbol{\theta}^{(t)})}{\sum_{k=1}^{K} \mathbb{P}(Z_i = k, \mathbf{X} = \mathbf{x}_i \mid \boldsymbol{\theta}^{(t)})} \\ \text{(Since } Z \text{ only depends on } \pi) \quad &= \frac{\mathbb{P}(\mathbf{X} = \mathbf{x}_i \mid Z_i = k, \boldsymbol{\theta}^{(t)}) \cdot \mathbb{P}(Z_i = k \mid \boldsymbol{\pi}^{(t)})}{\sum_{k=1}^{K} \mathbb{P}(\mathbf{X} = \mathbf{x}_i \mid Z_i = k, \boldsymbol{\theta}^{(t)}) \cdot \mathbb{P}(Z_i = k \mid \boldsymbol{\pi}^{(t)})}, \end{aligned}$$

which is a computable number by the parameters in t-th step. For convenience, we denote by

$$\tau_i^{k(t)} := \mathbb{P}(Z_i = k \mid \mathbb{X}, \boldsymbol{\theta}^{(t)}) = \mathbb{P}_{q_t}(Z_i = k \mid \mathbb{X}, \boldsymbol{\theta}^{(t)}).$$

Then we take expectation of (6.8) at t-th step w.r.t. $\mathbf{z}$ under q_t:

$$\begin{aligned}\mathbb{E}_{q_t}\Big[\ell_c(\boldsymbol{\theta}^{(t)} \mid \mathbb{X}, \mathbf{z})\Big] &= \sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{K}\sum_{l=1}^{L} \mathbb{1}(x_{ij}=l)\mathbb{P}_{q_t}(Z_i=k)\log p_{ljk} + \sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{P}_{q_t}(Z_i=k)\log\pi_k \\ &= \sum_{i=1}^{N}\sum_{j=1}^{p}\sum_{k=1}^{K}\sum_{l=1}^{L} \mathbb{1}(x_{ij}=l)\tau_i^{k(t)}\log p_{ljk} + \sum_{i=1}^{N}\sum_{k=1}^{K}\tau_i^{k(t)}\log\pi_k. \end{aligned} \tag{6.9}$$

M-Step

From E-step (6.9), we can see the optimization with respect to p_{ljk}'s and π_k's can be separated, and each of them is a multinomial MLE type problem. Using the result of multinomial MLE in GMM, we conclude for all j, l, l:

$$\pi_k^{(t+1)} = \frac{\sum_{i=1}^{N}\tau_i^{k(t)}}{N} \text{ and } p_{ljk}^{(t+1)} = \frac{\sum_{i=1}^{N}\mathbb{1}(x_{ij}=l)\tau_i^{k(t)}}{\sum_{l=1}^{L}\sum_{i=1}^{N}\mathbb{1}(x_{ij}=l)\tau_i^{k(t)}}.$$

6.6.3 The Selection of K

Either Gaussian mixture model and latent class model are model-based, so a natural way to choose the number of clusters K is information-based criterion, such as AIC ad BIC.

6.7 Bayesian GMM

Instead of computing the MLE and doing inference by large sample theory in the Gaussian mixture model (frequentist), in Bayesian's view, we now treat for all $k = 1, \ldots, K$, $(\pi_k, \boldsymbol{\mu}_k, \Sigma_k)$ along with $\mathbf{Z}$ all as latent variables, and infer them by computing the posterior distribution only given the observed data $\mathbb{X}$.

Note that, in Bayesian view, we naturally allow the parameters to be latent random variables. So, it is very intuitive to use the Bayesian hierarchical structure for our GMM, whose model structure is as illustrated in Figure 1. The circled variables are random, but the uncircled variables are hyperparameters with details to be introduced in §3.1 conjugate priors.

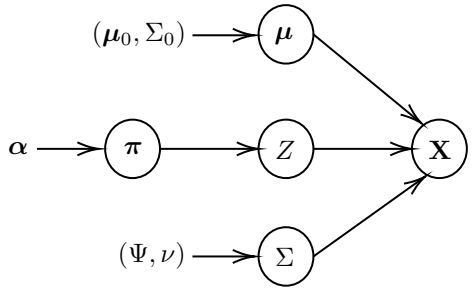


Figure 1: Hierarchical Gaussian mixture model

6.7.1 Conjugate Priors

From Figure 1, we can see there are three latent variables $\boldsymbol{\mu}, \boldsymbol{\pi}, \Sigma$ needing priors, after which we can determine the whole model. And in this section, we will explore the three one by one.

Dirichlet Prior for Multinomial Likelihood

By the GMM setting, we already prespecify that

$$Z_i \mid \boldsymbol{\pi} \overset{\text{iid}}{\sim} \text{Categorical}_K(\boldsymbol{\pi}), \quad \forall i = 1, \dots, N.$$

Then by summing up the category numbers, it follows that

$$\begin{pmatrix} N_1 \\ \vdots \\ N_K \end{pmatrix} := \begin{bmatrix} \sum_{i=1}^N \mathbb{1}(Z_i = 1) \\ \vdots \\ \sum_{i=1}^N \mathbb{1}(Z_i = K) \end{bmatrix} \sim \text{Multi}_K(N, \boldsymbol{\pi})$$

with the likelihood function

$$\mathbb{P}(\mathbf{Z} = \mathbf{z} \mid \boldsymbol{\pi}) = \prod_{i=1}^N \mathbb{P}(Z_i = z_i \mid \boldsymbol{\pi}) = \prod_{k=1}^K \pi_k^{N_k}.$$

A very natural choice for the prior is the following Dirichlet distribution.

Definition 13 (Dirichlet distribution)**.** The Dirichlet distribution of order $K \geq 2$ with parameters $\boldsymbol{\alpha} \succ 0$, denoted by $\text{Dirichlet}_K(\boldsymbol{\alpha})$, has a probability density function with respect to Lebesgue measure on the Euclidean space $\mathbb{R}^{K-1}$ given by

$$f(\pi_1, \dots, \pi_K \mid \alpha_1, \dots \alpha_K) = \frac{1}{\text{B}(\boldsymbol{\alpha})} \prod_{i=1}^K \pi_i^{\alpha_i - 1} \text{ with } \sum_{k=1}^K \pi_i = 1 \text{ and } \mathbb{E}(\pi_k) = \frac{\alpha_k}{\sum_{k=1}^K \alpha_k},$$

where the normalizing constant $B(\boldsymbol{\alpha})$ is the multivariate Beta function.

If we set the prior $\boldsymbol{\pi} \sim \text{Dirichlet}_K(\boldsymbol{\alpha})$, with the above likelihood, the posterior distribution of $\boldsymbol{\pi}$ follows

$$\boldsymbol{\pi} \mid \mathbf{z} \sim \text{Dirichlet}_K(N_1 + \alpha_1, \dots, N_K + \alpha_K) \text{ with } \mathbb{E}(\pi_k).$$

Remark. Note that when $K = 2$, the Dirichlet is the $\text{Beta}(\alpha_1, \alpha_2)$. And here, we give some special examples of $\text{Beta}(\alpha_1, \alpha_2)$ and see how the hyperparameters affect the distribution.

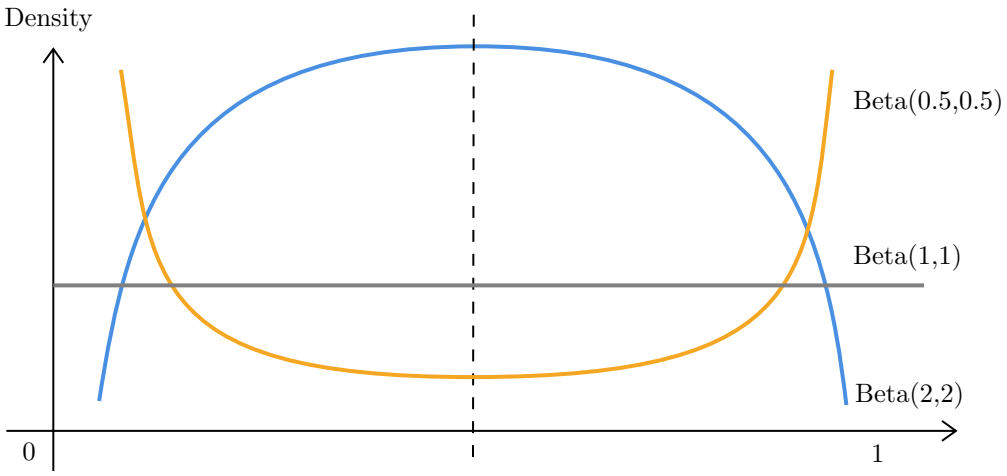


Figure 2: Beta distribution examples

One thing that is worth mentioning is when $\alpha < 0$, especially when close to 0, this prior encourages sparsity in the likelihood function since

$$\mathcal{L}(\text{all parameters}) = \mathcal{L}(\text{other structure}) \times \prod_{k=1}^{K} \pi_k^{\alpha_k - 1}.$$

If we maximize log-likelihood function w.r.t. parameters, we will see

$$\max_{\boldsymbol{\theta}} \left\{ \ell(\text{other structure}) + \sum_{k=1}^{K} (\alpha_k - 1) \log \pi_k \right\}.$$

When some $\alpha_k < 1$, it is possible for some $\pi_k \to 0$ in the optimization since the last term will explode to infinity. And we do not accept those clusters with $\pi_k = 0$, which helps us select the number K. For more about this idea, we can refer *Rousseau and Mengersen, 2011 JRSSB* or *Gu and Xu, 2019 JMLR*.

Normal Prior for the Mean in Normal Likelihood

Now we turn to the prior for the mean in a more general setting. Consider $\mathbf{x}_1, \ldots, \mathbf{x}_N \overset{\text{iid}}{\sim} \mathbf{N}_p(\boldsymbol{\mu}, \Sigma)$ where Σ is treated as known here. The log-likelihood function is (only consider $\boldsymbol{\mu}$ term)

$$\log p(\mathbb{X} \mid \boldsymbol{\mu}, \Sigma) \underset{(\boldsymbol{\mu})}{\propto} -\frac{1}{2} \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}),$$

a quadratic function of $\boldsymbol{\mu}$. If the prior of $\boldsymbol{\mu}$ is another normal, introducing another quadratic function, then the posterior of $\boldsymbol{\mu}$ is also a normal.

Using this idea, we set $\boldsymbol{\mu} \sim \mathbf{N}_p(\boldsymbol{\mu}_0, \Sigma_0)$ with $\boldsymbol{\mu}_0, \Sigma_0$ being specified hyperparameters. Then we have

$$\begin{aligned} \log p(\boldsymbol{\mu} \mid \mathbb{X}, \Sigma) &\propto -\frac{1}{2} \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}) - \frac{1}{2} (\boldsymbol{\mu} - \boldsymbol{\mu}_0)^\top \Sigma_0^{-1} (\boldsymbol{\mu} - \boldsymbol{\mu}_0) \\ &\sim \mathbf{N}_p(\tilde{\boldsymbol{\mu}}, \tilde{\Sigma}), \quad \text{where } \tilde{\Sigma}^{-1} = N\Sigma^{-1} + \Sigma_0^{-1} \text{ and } \tilde{\boldsymbol{\mu}} = \tilde{\Sigma} \left(\sum_{i=1}^{N} \Sigma^{-1} \mathbf{x}_i + \Sigma_0^{-1} \boldsymbol{\mu}_0 \right). \end{aligned}$$

Inverse-Wishart Prior for Covariance in Normal Likelihood

Let's consider the general case for $\mathbf{x}_1, \ldots, \mathbf{x}_N \overset{\text{iid}}{\sim} \mathbf{N}_p(\boldsymbol{\mu}, \Sigma)$ where $\boldsymbol{\mu}$ is treated as known here. The log-likelihood function is (only consider Σ term)

$$\log p(\Sigma \mid \mathbb{X}, \boldsymbol{\mu}) \underset{(\Sigma)}{\propto} \frac{N}{2} \log |\Sigma|^{-1} - \frac{1}{2} \sum_{i=1}^{N} (\mathbf{x}_i - \boldsymbol{\mu})^\top \Sigma^{-1} (\mathbf{x}_i - \boldsymbol{\mu}),$$

a determinant term and a centering inverse term of Σ. Fortunately, the inverse-Wishart distribution also has the same form as follows.

Definition 14 (Wishart and Inverse-Wishart distribution)**.** Suppose $\mathbf{y}_1, \ldots, \mathbf{y}_v \overset{\text{iid}}{\sim} \mathbf{N}_p(0, \Psi)$, then

$$\sum_{i=1}^{v} \mathbf{y}_i \mathbf{y}_i^\top \sim \text{Wishart}(\Psi, v).$$

Suppose $\Sigma \sim \text{Wishart}(\Psi, v)$, then define $\Sigma^{-1} \sim \text{Inv-Wishart}\left(\Psi^{-1}, v\right)$. Or conversely, suppose $\Sigma \sim \text{Inv-Wishart}(\Psi, v)$, then define $\Sigma^{-1} \sim \text{Wishart}\left(\Psi^{-1}, v\right)$, where the PDF of $\Sigma \sim$Inv-Wishart(Ψ, v)

$$f(\Sigma) \underset{(\Sigma)}{\propto} |\Sigma|^{-\frac{v+p+1}{2}} \exp\left\{-\frac{1}{2}\text{tr}\left(\Psi\Sigma^{-1}\right)\right\}.$$

Therefore, if we specify $\mathbf{x}_1, \ldots, \mathbf{x}_N \overset{\text{iid}}{\sim} \mathbf{N}_p(\boldsymbol{\mu}, \Sigma)$ where $\boldsymbol{\mu}$ is known, and $\Sigma \sim \text{Inv-Wishart}(\Psi, v)$, then the posterior of Σ follows

$$\Sigma \mid \mathbb{X}, \boldsymbol{\mu} \sim \text{Inv-Wishart}\left(\Psi + \sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^\top, v + N\right).$$

6.7.2 Model Setup and Conditional Posteriors

With priors in the last section, we can specify the model setup of Figure 1.

Assumption 28 (Bayesian Gaussian mixture model)**.** Suppose $\mathbf{X} \in \mathbb{R}^p$ is one observation, and there exists a latent label $Z \in \{1, 2, \ldots, K\}$ such that

$$\begin{cases} Z \mid \boldsymbol{\pi} \sim \text{Categorical}_K(\boldsymbol{\pi}), \\ \mathbf{X} \mid (Z = k, \boldsymbol{\mu}_k, \Sigma_k) \sim \mathbf{N}_p(\boldsymbol{\mu}_k, \Sigma_k), \end{cases}$$

where the parameters $\boldsymbol{\pi}, \boldsymbol{\mu}_k, \Sigma_k$ for $k = 1, \ldots, K$ have the independent priors

$$\begin{cases} \boldsymbol{\pi} \sim \text{Dirichlet}_K(\boldsymbol{\alpha}), \\ \boldsymbol{\mu}_k \overset{\text{iid}}{\sim} \mathbf{N}_p(\boldsymbol{\mu}_0, \Sigma_0), \\ \Sigma_k \overset{\text{iid}}{\sim} \text{Inv-Wishart}(\Psi, v), \end{cases}$$

where $\boldsymbol{\alpha}, \boldsymbol{\mu}_0, \Sigma_0, \Psi, v$ are all prespecified hyperparameters.

With this setup, we can derive the posterior distributions for all parameters now.

Note that the joint distribution of all random things is $(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma, \mathbf{Z}, \mathbb{X})$

$$\begin{aligned} p(\boldsymbol{\theta}, \mathbf{z}, \mathbb{X}) &= \prod_{i=1}^{N} p(\mathbf{x}_i, z_i, \boldsymbol{\theta}) = \prod_{i=1}^{N} p(\mathbf{x}_i \mid z_i, \boldsymbol{\theta}) p(z_i \mid \boldsymbol{\theta}) p(\boldsymbol{\theta}) \\ &= \prod_{i=1}^{N} p(\mathbf{x}_i \mid z_i, \boldsymbol{\mu}, \Sigma) p(z_i \mid \boldsymbol{\pi}) \cdot p(\boldsymbol{\pi} \mid \boldsymbol{\alpha}) \prod_{k=1}^{K} p(\boldsymbol{\mu}_k \mid \boldsymbol{\mu}_0, \Sigma_0) p(\Sigma_k \mid \Psi, v) \\ &= \prod_{i=1}^{N} \left\{ \prod_{k=1}^{K} p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k)^{\mathbb{1}(z_i = k)} \right\} p(z_i \mid \boldsymbol{\pi}) \cdot p(\boldsymbol{\pi} \mid \boldsymbol{\alpha}) \prod_{k=1}^{K} p(\boldsymbol{\mu}_k \mid \boldsymbol{\mu}_0, \Sigma_0) p(\Sigma_k \mid \Psi, v). \end{aligned} \tag{6.10}$$

Our next goal should be finding the posterior distribution of parameters, such as $p(\boldsymbol{\pi} \mid \mathbb{X})$. However, we may find in (6.10), the parameters are not separable, so what we do next is actually find the conditional distribution given all other things

$$p(\boldsymbol{\pi} \mid \mathbb{X}, \mathbf{z}, \boldsymbol{\mu}, \Sigma), \quad p(\boldsymbol{\mu}_k \mid \mathbb{X}, \mathbf{z}, \boldsymbol{\pi}, \Sigma), \quad p(\Sigma_k \mid \mathbb{X}, \mathbf{z}, \boldsymbol{\pi}, \boldsymbol{\mu}), \text{ and } p(\mathbf{z} \mid \mathbb{X}, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma),$$

instead of only given the observed data $\mathbb{X}$.

Conditional Posterior of Label Probability

To find the conditional posterior of $(\boldsymbol{\pi} \mid \text{others})$, we need to pick out all terms that concerns $\boldsymbol{\pi}$ from (6.10). And we can tell

$$\begin{aligned} p(\boldsymbol{\pi} \mid \text{others}) &\propto p(\boldsymbol{\pi} \mid \boldsymbol{\alpha}) \prod_{i=1}^{N} p(z_i \mid \boldsymbol{\pi}) \\ &= \prod_{k=1}^{K} \pi_k^{\alpha_k + N_k} \sim \text{Dirichlet}_K(\alpha_k + N_k), \end{aligned}$$

where $N_k = \sum_{i=1}^{N} \mathbb{1}(z_i = k)$, which is **random**!!

Conditional Posterior of Mean

Similarly, we can do for any fixed $k = 1, \ldots, K$,

$$\begin{aligned} p(\boldsymbol{\mu}_k \mid \text{others}) &\propto p(\boldsymbol{\mu}_k \mid \boldsymbol{\mu}_0, \Sigma_0) \prod_{i:z_i=k}^{N} p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \\ &\sim \mathbf{N}_p(\tilde{\boldsymbol{\mu}}_k, \tilde{\Sigma}_k), \text{ where } \tilde{\Sigma}_k^{-1} = \Sigma_0^{-1} + N_k \Sigma_k^{-1} \text{ and } \tilde{\boldsymbol{\mu}}_k = \tilde{\Sigma}_k \left[\Sigma_0^{-1} \boldsymbol{\mu}_0 + \Sigma_k^{-1} \sum_{i:z_i=k} \mathbf{x}_i \right]. \end{aligned}$$

And we can also note that $(\boldsymbol{\mu}_k \mid \text{others})$ depends on Σ_k and $\mathbf{z}$, and it should also be **random**!!

Conditional Posterior of Covariance

For $k = 1, \ldots, K$, we find

$$p(\Sigma_k \mid \text{others}) \propto p(\Sigma_k \mid \Psi, v) \prod_{i=1}^{N} p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k) \sim \text{Inv-Wishart} \left(\Psi + \sum_{i:z_i=k} (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^\top, N_k + v \right).$$

Conditional Posterior of Labels

For $i = 1, \ldots, N$, we find

$$p(z_i \mid \text{others}) \propto p(z_i \mid \boldsymbol{\pi}) \prod_{k=1}^{K} p(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k)^{\mathbb{1}(z_i=k)},$$

and by the same derivation in GMM (τ's derivation), we further conclude

$$\mathbb{P}(Z_i = k \mid \text{others}) = \frac{\pi_k N(\mathbf{x}_i \mid \boldsymbol{\mu}_k, \Sigma_k)}{\sum_{l=1}^{K} \pi_l N(\mathbf{x}_i \mid \boldsymbol{\mu}_l, \Sigma_l)},$$

whose distribution also depends on other parameters, and hence random.

6.7.3 MCMC Sample from Posterior Distribution

Now, it remains to connect

$$\text{Conditional on all others: } \begin{cases} p(\boldsymbol{\pi} \mid \text{others}) \\ p(\boldsymbol{\mu}_k \mid \text{others}) \\ p(\Sigma_k \mid \text{others}) \\ p(\mathbf{z} \mid \text{others}) \end{cases} \Rightarrow \text{ Conditional on observed data: } \begin{cases} p(\boldsymbol{\pi} \mid \mathbb{X}) \\ p(\boldsymbol{\mu}_k \mid \mathbb{X}) \\ p(\Sigma_k \mid \mathbb{X}) \\ p(\mathbf{z} \mid \mathbb{X}) \end{cases}.$$

The Markov Chain Monte Carlo (MCMC) is a general method based on drawing values of $\boldsymbol{\theta}$ from approximated distributions and then correlating those draws to better approximate the target distribution $p(\boldsymbol{\theta} \mid \text{Data})$. We here introduce three MCMC samplers, *Gibbs sampler, Metropolis algorithm* and *Metropolis-Hasting algorithm* to help, which is not limited to this Bayesian GMM, but sample from any given distribution, even only the kernel. For more and details, refer the book *Bayesian Data Analysis*.

Gibbs Sampler

A very natural intuition leads to the Gibbs sampler. In fact, it can be used to sample for not only the posterior but any distribution.

Algorithm 23 (Gibbs sampler)**.** Suppose the posterior distribution $p(\boldsymbol{\theta} \mid \text{data})$ of the random parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ is given.

(a) At each iteration t, sample for $j = 1, \ldots, d$ from

$$\theta_j^{(t)} \sim p(\theta_j \mid \boldsymbol{\theta}_{-j}^{(t-1)}, \text{data}).$$

(b) Repeat until some large T, which comes from some convergence analysis.

Example. For exercises or illustration purpose, we use Gibbs sample for the bivariate normal distribution:

$$\begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} \mid \boldsymbol{\theta} \sim \mathbf{N}_2 \left(\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right) \quad \text{with non-informative prior:} \quad p(\boldsymbol{\theta}) \propto c.$$

Simple calculation leads to the posterior distribution of $\boldsymbol{\theta}$ to be

$$\begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} \mid \mathbf{y} \sim \mathbf{N}_2 \left(\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right) \Rightarrow \begin{cases} \theta_1 \mid (\theta_2, \mathbf{y}) \sim \mathrm{N}\left(y_1 + \rho(\theta_2 - y_2), 1 - \rho^2\right), \\ \theta_2 \mid (\theta_1, \mathbf{y}) \sim \mathrm{N}\left(y_2 + \rho(\theta_1 - y_1), 1 - \rho^2\right). \end{cases}$$

Starting by some initial value $\boldsymbol{\theta}^{(0)}$, we sample from the conditional posterior distributions above and update until a large T.

Remark. Here are two things worth mentioning, the estimations of the posterior mean and variance. Suppose we already have $\boldsymbol{\theta}^{(0)}, \ldots, \boldsymbol{\theta}^{(T_0)}, \ldots, \boldsymbol{\theta}^{(T)}$ from the Gibbs sampler, where from T_0 on the distribution sequence already achieves stationary.

(a) To estimate or inference anything, we ignore the unstationary part, $t = 1, \ldots, T_0$, which is also called burn-in part.

(b) To estimate the posterior mean, we can use the sample average

$$\hat{\boldsymbol{\theta}} = \frac{1}{T - T_0} \sum_{t=T_0+1}^{T} \boldsymbol{\theta}^{(t)}.$$

(c) After convergence, the draws $\boldsymbol{\theta}^{(T_0)}, \ldots, \boldsymbol{\theta}^{(T)}$ can be viewed as identically distributed from $p(\boldsymbol{\theta} \mid \text{data})$, but correlated, NOT independent, and with the limit

$$\lim_{T \to \infty} (T - T_0) \cdot \text{Var}(\bar{\boldsymbol{\theta}}) = \left(1 + 2 \sum_{s=1}^{\infty} \rho_s \right) \text{Var}(\boldsymbol{\theta} \mid \mathbf{y}),$$

where ρ_s is the autocorrelation at lag s, i.e. $\rho_s = \text{corr}(\boldsymbol{\theta}_0, \boldsymbol{\theta}_s)$.

(d) Therefore, to estimate the variance, we need to use the sample far apart, such as $\boldsymbol{\theta}^{(T_0)}, \boldsymbol{\theta}^{(T_0+1000)}, \ldots$.

Example (Bayesian GMM)**.** From previous discussion, the joint distribution of $(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma, \mathbf{Z}, \mathbb{X})$ is (6.10), and we have derived the posterior distribution for each parameter conditioning on all others and data. Therefore, the Gibbs sampler for Bayesian GMM is as follows.

(a) Initialize $\boldsymbol{\pi}, \boldsymbol{\mu}_k, \Sigma_k, z_i$ for all $k = 1, \ldots, K$ and $i = 1, \ldots, N$.

(b) For the t-th sampling, set $N_k^{(t-1)} = \sum_{i=1}^{N} \mathbb{1}(z_i^{(t-1)} = k)$. And we updates all parameters by followings.

b1) Sample $\left(\boldsymbol{\pi}^{(t)} \mid \text{others}\right) \sim \text{Dirichlet}_K \left(\alpha_k + N_k^{(t-1)}\right)$.

b2) For $k = 1, \ldots, K$, sample

$$\left(\boldsymbol{\mu}_k^{(t)} \mid \text{others}\right) \sim \mathbf{N}_p \left(\tilde{\boldsymbol{\mu}}_k^{(t-1)}, \tilde{\Sigma}_k^{(t-1)}\right), \quad \text{where} \quad \left(\tilde{\Sigma}_k^{(t-1)}\right)^{-1} = \Sigma_0^{-1} + N_k^{(t-1)} \left(\Sigma_k^{(t-1)}\right)^{-1}$$

$$\text{and} \quad \tilde{\boldsymbol{\mu}}_k^{(t-1)} = \tilde{\Sigma}_k^{(t-1)} \left[\Sigma_0^{-1} \boldsymbol{\mu}_0 + \left(\Sigma_k^{(t-1)}\right)^{-1} \sum_{i : z_i^{(t-1)} = k} \mathbf{x}_i \right].$$

b3) For $k = 1, \ldots, K$, sample

$$\left(\Sigma_k^{(t)} \mid \text{others}\right) \sim \text{Inv-Wishart} \left(\Psi + \sum_{i : z_i^{(t-1)} = k} (\mathbf{x}_i - \boldsymbol{\mu}_k^{(t-1)})(\mathbf{x}_i - \boldsymbol{\mu}_k^{(t-1)})^\top, N_k^{(t-1)} + v \right).$$

b4) For $i = 1, \ldots, N$, sample $z_i^{(t)}$ from

$$\mathbb{P}\left(Z_i^{(t)} = k \mid \text{others}\right) = \frac{\pi_k^{(t-1)} N\left(\mathbf{x}_i \mid \boldsymbol{\mu}_k^{(t-1)}, \Sigma_k^{(t-1)}\right)}{\sum_{l=1}^{K} \pi_l^{(t-1)} N\left(\mathbf{x}_i \mid \boldsymbol{\mu}_l^{(t-1)}, \Sigma_l^{(t-1)}\right)}.$$

Metropolis Algorithm

Metropolis algorithm is another MCMC way to draw a sample from a distribution. But its power is, only a known kernel is enough.

Algorithm 24 (Metropolis)**.** Suppose the posterior density (or kernel) $p(\boldsymbol{\theta} \mid \text{data})$ is given.

(a) Draw a starting value $\boldsymbol{\theta}^{(0)}$ from a starting distribution $p_0(\boldsymbol{\theta})$, such as normal.

(b) At each iteration t, update the parameter as follows.

b1) Sample a proposal $\boldsymbol{\theta}^*$ from a proposal distribution $q_t\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(t-1)}\right)$, which is required to be symmetric, i.e.

$$q_t\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(t-1)}\right) = q_t\left(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^*\right) \quad \text{such as Normal.}$$

b2) Set

$$\boldsymbol{\theta}^{(t)} = \begin{cases} \boldsymbol{\theta}^* & \text{with probability } \min\left(1, \frac{p(\boldsymbol{\theta}^* \mid \text{data})}{p\left(\boldsymbol{\theta}^{(t-1)} \mid \text{data}\right)}\right), \\ \boldsymbol{\theta}^{(t-1)} & \text{otherwise,} \end{cases}$$

where $p(\cdot \mid \text{data})$ is the posterior density or kernel.

Remark. We can see, once the proposal sample increases the likelihood, we definitely accept it. Otherwise, we accept it with the *acceptance ratio*

$$\gamma_t := \frac{p(\boldsymbol{\theta}^* \mid \text{data})}{p\left(\boldsymbol{\theta}^{(t-1)} \mid \text{data}\right)}.$$

But there might exists cases that the bad initial value leads the sample to stay where it is. One way to solve it is to choose a proper q_t. For example, if we choose normal, we can adjust the σ^2 as a tuning parameter. Enlarging this variance makes it more possible to sample away from the current one.

Metropolis-Hasting Algorithm

A modified version of Metropolis is to relax the symmetric requirement on the proposal distribution, and we call it the *Metropolis-Hasting* algorithm.

Algorithm 25 (Metropolis-Hasting)**.** Suppose the posterior density (or kernel) $p(\boldsymbol{\theta} \mid \text{data})$ is given.

(a) Draw a starting value $\boldsymbol{\theta}^{(0)}$ from a starting distribution $p_0(\boldsymbol{\theta})$, such as normal.

(b) At each iteration t, update the parameter as follows.

b1) Sample a proposal $\boldsymbol{\theta}^*$ from a proposal distribution $q_t\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(t-1)}\right)$.

b2) Set

$$\boldsymbol{\theta}^{(t)} = \begin{cases} \boldsymbol{\theta}^* & \text{with probability } \min\left(1, \frac{p(\boldsymbol{\theta}^* \mid \text{data}) / q_t\left(\boldsymbol{\theta}^* \mid \boldsymbol{\theta}^{(t-1)}\right)}{p\left(\boldsymbol{\theta}^{(t-1)} \mid \text{data}\right) / q_t\left(\boldsymbol{\theta}^{(t-1)} \mid \boldsymbol{\theta}^*\right)}\right), \\ \boldsymbol{\theta}^{(t-1)} & \text{otherwise,} \end{cases}$$

where $p(\cdot \mid \text{data})$ is the posterior density or kernel.

Practical Sampling

In practice, we can combine the Gibbs sampler and MH algorithm for the sample from $\boldsymbol{\theta}_d \mid \text{data}$ as

$$\text{Gibbs sampler: } \begin{cases} \theta_1 \mid \text{others}, & \text{easy to sample}, \\ \theta_2 \mid \text{others}, & \text{easy to sample}, \\ \quad \vdots & \quad \vdots \\ \boldsymbol{\theta}_j \mid \text{others}, & \text{hard to sample by MH}. \end{cases}$$

And we can generalize these sampling methods to MCMC EM, such as for the binary PCA model. Recall that before the E-step, we will derive the conditional distribution q_t for the latent variable $\mathbf{Z} \mid$ others. So, we can use sampling methods here to get sample $\mathbf{z}$'s, and use it in the E-step, instead of doing integration. In this way, even the integration is not doable, the whole EM algorithm is doable, making the model estimable.

Bibliography

[1] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[2] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.