STATS 601 Lecture Notes

STATS 601: Advanced Statistical Learning

University of Michigan, Ann Arbor

Author: Ziheng Wei
Date: April 29, 2025

CONTENT

Before You Read

These notes are for *STATS 601: Advanced Statistical Learning* at the University of Michigan in the Winter 2025 semester, taught by Professor Kean Ming Tan and GSI Kihyuk Hong. The author is Ziheng Wei, a first-year Master of Applied Statistics student at UMich.

STATS 601 is a PhD-level course covering a broad array of statistical learning methods, with a particular focus on dimension reduction, classification, and clustering methods. Unlike other machine learning courses at UMich, such as STATS 503, EECS 545, EECS 553, and SI 670, this course places a strong emphasis on theory. For example, in the final exam you might spend more than half an hour deriving an EM-algorithm problem by hand. If you are primarily interested in machine-learning programming, you may prefer one of the four courses listed above.

STATS 601 is solid, especially for some master's and undergraduate students who must invest significant effort to earn a good grade. My notes include both lecture material and Ki's lab sessions. I hope they will help any student (even just one) taking this course, as well as statistical enthusiasts pursuing theoretical machine learning, to master the content more effectively.

Due to my own limitations, these notes may not always explain concepts as clearly as a textbook and may contain errors. If you find any mistakes or have suggestions for improvement, feel free to contact me at `zihwei@umich.edu`.

Ziheng Wei
Apr 28, 2025
(the day before the final exam)

Introduction

(According to the syllabus)

0.1 Course Description

This is an advanced introduction to statistical learning methods. Topics include:

(1) dimension reduction techniques, including principal component analysis, factor analysis, multidimensional scaling and manifold learning;

(2) conceptual framework of classification including cost functions, Bayes classifiers, overfitting and generalization; specific classification methods including logistic regression, naive Bayes, discriminant analysis, support vector machines, kernel-based methods, generalized additive models, tree-based methods, boosting, neural networks;

(3) clustering methods including K-means, model-based clustering algorithms, mixture models, latent variable models, hierarchical models; and algorithms such as the EM algorithm, Gibbs sampling, and variational inference methods;

(4) Additional topics that may be covered include categorical data analysis, graphical models, and deep learning.

0.2 Course Outline

- **Review of multivariate Gaussian**
- **Dimension reduction methods.**
  - data-driven linear dimensionality reduction (principal component analysis)
  - model-based linear dimensionality reduction (factor analysis)
  - nonlinear dimension reduction: multidimensional scaling and manifold learning
- **Classification methods.**
  - model-based classification: linear classification, logistic regression

  - data-driven classification
  - support vector machines, generalized additive models, tree-based methods, boosting, neural networks

- **Clustering methods.**
  - data-driven clustering: k-means, spectral clustering
  - model-based clustering: mixture of multivariate Gaussians
  - Dirichlet process mixtures and multivariate extensions

- **Selective topics**
  - graphical models and hierarchical models
  - latent variable models

0.3 Grades

There will be bi-weekly homework assignments, a midterm exam, a final exam, and an individual project.

0.4 Textbooks and References

1. *The elements of statistical learning* by Hastie, Friedman, and Tibshirani (henceforth *ESL*);
2. *Pattern recognition and machine learning* by C. Bishop (henceforth *PRML*);
3. *An Introduction to Multivariate Statistical Analysis* by T.W.Anderson;
4. *Aspects of multivariate statistical theory* by R.J. Muirhead;
5. Long Nguyen's lecture notes;
6. Michael Jordan's selected book chapters on graphical models.

Part I

DIMENSION REDUCTION

Chapter 1

Multivariate Normal Distribution (MVN)

1.1 Multivariate Normal Distribution

1.1.1 Definition

Consider a p-dimension random vector $Z = (Z_1, \ldots, Z_p)^T$, where $Z_1, \ldots, Z_p \overset{i.i.d.}{\sim} N(0,1)$. We have $EZ_i = 0$, $varZ_i = 1$ and $varZ = I_p$. The PDF of Z is

$$f(Z) = \prod_{i=1}^{p} f(Z_i) = \frac{1}{(2\pi)^{\frac{p}{2}}} \exp(-\frac{1}{2}\sum_{i=1}^{p} z_i^2) = \frac{1}{(2\pi)^{\frac{p}{2}}} \exp(-\frac{1}{2}Z^T Z),\ Z \in \mathbb{R}^p$$

Let $B \in \mathbb{R}^{p\times p}$ be any non-singular matrix and $\mu \in \mathbb{R}^p$ be any p-dimensional vector. $X = BZ + \mu \in \mathbb{R}^p$.
By Jacobi transformation, the PDF of X is given by

$$f(X|\mu,\Sigma) = (2\pi)^{-\frac{p}{2}}|BB^T|^{-\frac{1}{2}} \exp[-\frac{1}{2}(x-\mu)^T(BB^T)^{-1}(X-\mu)]$$

$$= (2\pi)^{-\frac{p}{2}}|\Sigma|^{-\frac{1}{2}} \exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(X-\mu)]$$

where $\Sigma = BB^T$. It's obvious that $\begin{cases} EX = E(BZ+\mu) = BE(Z) + \mu = \mu \\ cov(X) = cov(BZ) = Bcov(Z)B^T = BB^T = \Sigma \end{cases}$.

1.1.2 Properties

1. $\int f(x|\mu,\Sigma)dx = 1$ (property of PDF);
2. The distribution of X only depends on μ and Σ;
3. The affine transformation of a Normal distribution is Normal too: $X \sim N_p(\mu,\Sigma)$. Let $A \in \mathbb{R}^{n\times p}, b \in \mathbb{R}^{n\times 1}$, then $Y = AX + b \sim N_n(A\mu + b, A\Sigma A^T)$;

Proof sketch. Consider $X = BZ + \mu$, $Z \sim N_p(0, I_p)$, we have $Y = ABZ + A\mu + b$.

Then, $\begin{cases} E(Y) = ABE(Z) + A\mu + b = A\mu + b \\ cov(ABZ) = ABI_pB^TA^T = ABB^TA^T = A\Sigma A^T \end{cases}$. □

4. The subset of a MVN distribution is Normal too:
$A = (I_m,\ 0)_{m\times p} = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix}$, $b = 0$. Then $AX + b = \begin{pmatrix} X_1 \\ \vdots \\ X_m \end{pmatrix} \sim N_m(\mu', \Sigma')$ where $\mu' = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_m \end{pmatrix}$, $\Sigma' = \begin{pmatrix} \sigma_{11} & 0 & \cdots & 0 \\ 0 & \sigma_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_{mm} \end{pmatrix}$;

5. Standardization of a MVN is also MVN:
$X \sim N_p(\mu, \Sigma)$, then $\Sigma^{-1/2}(X - \mu) \sim N_p(0, I_p)$;

6. If (X_1, X_2) follows MVN, Then X_1 is independent of $X_2 \Leftrightarrow cov(X_1, X_2) = 0$;

7. Conditional distribution of MVN:
If $X \sim N_p(\mu, \Sigma)$, Σ_{22} is positive definite, where $X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$, $\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}$, $\Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$. The distribution of X_1 given X_2 is

$$X_1|X_2 \sim N(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}).$$

Note

Let $S \in \mathbb{R}^{n\times n}$, define

$$S = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}$$

where $S_{11} \in \mathbb{R}^{n_1\times n_1}$, $S_{12} \in \mathbb{R}^{n_1\times n_2}$, $S_{21} \in \mathbb{R}^{n_2\times n_1}$, $S_{22} \in \mathbb{R}^{n_2\times n_2}$ and $n_1 + n_2 = n$.

Consider partition

$$\begin{pmatrix} I_{n_1} & -S_{12}S_{22}^{-1} \\ 0 & I_{n_2} \end{pmatrix} \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \begin{pmatrix} I_{n_1} & 0 \\ -S_{22}^{-1}S_{21} & I_{n_2} \end{pmatrix} = \begin{pmatrix} S_{11.2} & 0 \\ 0 & S_{22} \end{pmatrix}$$

where

$$S_{11.2} = S_{11} - S_{12}S_{22}^{-1}S_{21}$$

Note

Then

$$\begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} = \begin{pmatrix} I_{n_1} & S_{12}S_{22}^{-1} \\ 0 & I_{n_2} \end{pmatrix} \begin{pmatrix} S_{11.2} & 0 \\ 0 & S_{22} \end{pmatrix} \begin{pmatrix} I_{n_1} & 0 \\ S_{22}^{-1}S_{21} & I_{n_2} \end{pmatrix}$$

Its inverse matrix is

$$\begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix}^{-1} = \begin{pmatrix} I_{n_1} & 0 \\ -S_{22}^{-1}S_{21} & I_{n_2} \end{pmatrix} \begin{pmatrix} S_{11.2}^{-1} & 0 \\ 0 & S_{22}^{-1} \end{pmatrix} \begin{pmatrix} I_{n_1} & -S_{12}S_{22}^{-1} \\ 0 & I_{n_2} \end{pmatrix}$$

So

$$X^T S^{-1} X = (X_1 - S_{12}S_{22}^{-1}X_2)^T S_{11.2}^{-1}(X_1 - S_{12}S_{22}^{-1}X_2) + X_2^T S_{22}^{-1} X_2$$

From the notes above, we know that

$$(X^T-\mu)\Sigma^{-1}(X-\mu) = (X_1-\mu_1-\Sigma_{12}\Sigma_{22}^{-1}(X_2-\mu_2))^T\Sigma_{11.2}^{-1}(X_1-\mu_1-\Sigma_{12}\Sigma_{22}^{-1}(X_2-\mu_2))$$
$$+(X_2-\mu_2)^T\Sigma_{22}^{-1}(X_2-\mu_2)$$

which means $f(X_1, X_2) = f(X_1|X_2)f(X_2)$.

A fun fact is $X_1 - \Sigma_{12}\Sigma_{22}^{-1}X_2 \perp X_2$.

Proof sketch. Use property 3 and 6 above. Define matrix A as:

$$A = \begin{pmatrix} I_{p_1} & -\Sigma_{12}\Sigma_{22}^{-1} \\ 0 & I_{p_2} \end{pmatrix}$$

Multiplying by vector:

$$AX = \begin{pmatrix} X_1 - \Sigma_{12}\Sigma_{22}^{-1}X_2 \\ X_2 \end{pmatrix}$$

Since $X \sim N(\mu, \Sigma)$, we have:

$$AX \sim N(A\mu, A\Sigma A^T)$$

which follows:

$$\mathcal{N}_p\left(\begin{pmatrix} \mu_1 - \Sigma_{12}\Sigma_{22}^{-1}\mu_2 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11.2} & 0 \\ 0 & \Sigma_{22} \end{pmatrix}\right)$$

The covariance is 0. Thus, it follows that they are independent.
So the conditional distribution:

$$X_1 - \Sigma_{12}\Sigma_{22}^{-1}X_2 \mid X_2 \sim N(\mu_1 - \Sigma_{12}\Sigma_{22}^{-1}\mu_2, \Sigma_{11.2})$$

Therefore:

$$X_1 \mid X_2 \sim N(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2), \Sigma_{11.2})$$

□

Expectation of X_1 given X_2 can be regarded as the regression mean function of X_1 on X_2. $\Sigma_{12}\Sigma_{22}^{-1}$ is the regression coefficient.

Proof sketch. Recall OLS: $X_1 \sim X_2$. The estimated regression coefficient is $\hat{B} = (X_2^T X_2)^{-1} X_2^T X_1 = \Sigma_{22}^{-1}\Sigma_{12}$. □

1.1.3 Canonical representation of MVN

The canoncial representation of MVN $X \sim N_p(\mu, \Sigma)$ is:

$$f(X|\mu, \Sigma) = \exp\{a + \frac{1}{2} X^T \Theta X\}$$

where

$$\begin{cases} a = -\frac{1}{2}[p\log(2\pi) - \log|\Theta| + \eta^T\Theta^{-1}\eta] \\ \Theta = \Sigma^{-1} \\ \eta = \Theta\mu \end{cases}$$

One application: **Gaussian Graphical Model** (GGM).
We will discuss about it in the next section.

References

1. Chapter 17. *ESL*.
2. Meinshausen, N., & Bühlmann, P. (2006). High-dimensional graphs and variable selection with the lasso.
3. Speed, T.P. and Kiiveri, H.T., 1986. Gaussian Markov distributions over finite graphs. *The Annals of Statistics*, pp.138-150.
4. Friedman, J., Hastie, T. and Tibshirani, R., 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), pp.432-441.

1.1.4 Maximum Likelihood Estimation of MVN

Let $X_1, \ldots, X_n \overset{i.i.d.}{\sim} N_p(\mu, \Sigma) \quad (n \gg p)$.

Theorem: The MLEs for μ and Σ are:

$$\begin{cases} \hat{\mu} = \bar{X} \\ \hat{\Sigma} = \frac{1}{n}\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})^T \end{cases}$$

Proof.

$$f(X \mid \mu, \Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(X-\mu)^T\Sigma^{-1}(X-\mu)\right\}$$

So the likelihood function is:

$$L(\mu, \Sigma) = \prod_{i=1}^{n} f(X_i \mid \mu, \Sigma)$$

$$= (2\pi)^{-np/2}|\Sigma|^{-n/2} \exp\left\{-\frac{1}{2}\sum_{i=1}^{n}(X_i-\mu)^T\Sigma^{-1}(X_i-\mu)\right\}$$

$$= (2\pi)^{-np/2}|\Sigma|^{-n/2} \exp\left\{-\frac{1}{2}tr[\Sigma^{-1}\sum_{i=1}^{n}(X_i-\mu)(X_i-\mu)^T]\right\}$$

The log likelihood function is:

$$\log L = c + \frac{n}{2}\log|\Sigma^{-1}| - \frac{1}{2}tr(\Sigma^{-1}A)$$

where $A = \sum_{i=1}^{n}(X_i-\mu)(X_i-\mu)^T$.

To find MLE of μ, Σ:
First compute the derivative with respect to (henceforth *"w.r.t."*) μ:

$$\frac{\partial \log L}{\partial \mu} = 0 \Rightarrow \frac{\partial \log L}{\partial \mu} = \sum_{i=1}^{n}(X_i-\mu)^T\Sigma^{-1} = 0 \Rightarrow \frac{1}{n}\sum_{i=1}^{n}(X_i-\hat{\mu}) = 0$$

So

$$\hat{\mu} = \bar{X}$$

Note

- $\frac{\partial \log|A|}{\partial A} = A^{-1}$;
- $\frac{\partial tr(AB)}{\partial A} = B^T$;
- $\frac{\partial tr(ABA^T)}{\partial A} = A(B+B^T)$;
- $\frac{\partial tr(BA^TCA)}{\partial A} = 2CAB$ if B, C are symmetric.

Calculate the derivative w.r.t. $\boldsymbol{\Sigma^{-1}}$:

$$\frac{\partial \log L}{\partial \Sigma^{-1}} = \frac{n}{2}\Sigma - \frac{1}{2}A^T = 0$$

$$\Rightarrow \hat{\Sigma} = \frac{1}{n}\hat{A} = \frac{1}{n}\sum_{i=1}^{n}(X_i-\bar{X})(X_i-\bar{X})^T$$

□

1.2 Convex Optimization

1.2.1 Norm

A. Vector Norm $\|\cdot\|_r$:
$\forall \vec{x}, \vec{y} \in \mathbb{R}^n$, vector norm of $\mathbb{R}^n(\|\cdot\|)$ satisfies:
(1) $\|\vec{x}\| \geq 0$, $\|\vec{x}\| = 0 \Leftrightarrow \vec{x} = 0$;
(2) $\forall \lambda \in \mathbb{R}$, $\|\lambda \vec{x}\| = |\lambda| \, \|\vec{x}\|$;
(3) triangle inequality: $\|\vec{x} + \vec{y}\| \leq \|\vec{x}\| + \|\vec{y}\|$.

Note

frequently used vector norms:
L1 Norm: $\|\vec{x}\|_1 = \sum_{i=1}^{n} |x_i|$;
L2 Norm (Euclidean Norm): $\|\vec{x}\|_2 = \sqrt{\vec{x}^T \vec{x}} = \sqrt{\sum_{i=1}^{n} |x_i|^2}$;
L-infinity Norm (Chebyshev Norm): $\|\vec{x}\|_\infty = \max_{1 \leq i \leq n} |x_i|$.

B. Matrix Norm $\|\cdot\|_M$:
$\forall$A, B $\in \mathbb{R}^{n \times n}$, matrix norm of $\mathbb{R}^{n \times n}(\|\cdot\|)$ satisfies:
(1) $\|\mathrm{A}\| \geq 0$, $\|\mathrm{A}\| = 0 \Leftrightarrow \mathrm{A} = 0$;
(2) $\forall \lambda \in \mathbb{R}$, $\|\lambda \mathrm{A}\| = |\lambda| \, \|\mathrm{A}\|$;
(3) triangle inequality: $\|\mathrm{A} + \mathrm{B}\| \leq \|\mathrm{A}\| + \|\mathrm{B}\|$.

if it satisfies (4) $\|\mathrm{A} \cdot \mathrm{B}\| \leq \|\mathrm{A}\| \cdot \|\mathrm{B}\|$, then we call it compatible matrix norm or modulus.

Note

- **Frobenius-Norm (F-Norm)**: $\|\mathrm{A}\|_F = \sqrt{<\mathrm{A},\mathrm{A}>} = \sqrt{tr(\mathrm{A}^{\mathrm{T}}\mathrm{A})} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2}$. Often used to measure the error of matrices (L2 loss);
- **1-Norm (Column Sum Norm)**: $\|\mathrm{A}\|_1 = \max_{\|\vec{x}\|=1} \|\mathrm{A}\vec{x}\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|$, the maximum absolute column sum of the matrix;
- **2-Norm (Spectral Norm)**: $\|\mathrm{A}\|_2 = \max_{\|\vec{x}\|=1} \|\mathrm{A}\vec{x}\|_2 = \max \sqrt{\rho(\mathrm{A}^{\mathrm{T}}\mathrm{A})}$, the square root of the maximum eigenvalue of the matrix multiplied by its conjugate transpose;
- **Infinity Norm (Row Sum Norm)**: $\|\mathrm{A}\|_{\infty,\infty} = \max_{\|\vec{x}\|=1} \|\mathrm{A}\vec{x}\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|$, the maximum absolute row sum of the matrix;
- **nuclear norm**:$\|B\|_* = \sum_{i=1}^{q} \sigma_i(B)$, the sum of all the singular values of B. Often used to control the rank of a matrix in sparse low-rank modeling;
- $\|B\|_{1,2} := \sum_{i=1}^{q} \|B_i\|_2$ row sum of L2-norm of each row vectors;
- $\|X\|_{1,1} = \sum_{j,k} |X_{jk}|$ the sum of the absolute values for all elements in X;
- $\|X\|_{2,0} = \sum_{i=1}^{n} \mathbb{I}_{\{\|X_{i,:}\|_2 \neq 0\}}$ is the number of non-zero rows in X;
- $\|diag(X)\|_0 = \sum_{i=1}^{n} \mathbb{I}_{\{X_{ii}\neq 0\}}$ is the number of non-zeros diagonal entries in X.

1.2.2 Convex optimization problem

$$\begin{aligned} &\text{minimize} && f(\boldsymbol{x}) \\ &\text{subject to} && \begin{array}{ll} g_i(\boldsymbol{x}) \le 0, & i = 1,\dots,m \\ h_i(\boldsymbol{x}) = 0, & i = 1,\dots,p \end{array} \end{aligned}$$

where the optimization variable $\boldsymbol{x} \in \mathbb{R}^n$ and

- $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable.
- $g_i : \mathbb{R}^n \to \mathbb{R}$ are convex and differentiable.
- $h_i : \mathbb{R}^n \to \mathbb{R}$ are affine and differentiable.

1. Lagrangian function:

$$\mathcal{L}(\boldsymbol{x};\mu,\lambda) := f(\boldsymbol{x}) + \sum_{i=1}^{m} \mu_i\, g_i(\boldsymbol{x}) + \sum_{j=1}^{p} \lambda_j\, h_j(\boldsymbol{x})$$

2. KKT conditions

- $g_i(\mathbf{x}) \le 0, \quad i = 1,\dots,m.$ (feasibility)
- $h_i(\mathbf{x}) = 0, \quad i = 1,\dots,p.$ (feasibility)
- $\mu_i \ge 0, \quad i = 1,\dots,m.$
- $\lambda_i g_i(\mathbf{x}) = 0, \quad i = 1,\dots,m.$
- $\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x};\mu,\lambda) = 0.$

3. Slater's condition There exists $\mathbf{x}$ such that $g_i(\mathbf{x}) < 0, \quad i = 1,\dots,m$ and $h_i(\mathbf{x}) = 0, \quad i = 1,\dots,p.$

Theorem: If Slater's condition holds, then KKT conditions $\iff$ optimality.

Note that in many optimization problems, not necessarily convex, optimality implies KKT conditions.

1.2.3 Example: Projection onto a vector space

The projection of $y \in \mathbb{R}^n$ onto a subspace $\mathcal{S} \subseteq \mathbb{R}^n$ is

$$\text{Proj}_{\mathcal{S}}(y) = \arg\min_{x \in \mathcal{S}} \|x - y\|_2$$

Let $X \in \mathbb{R}^{n \times p}$ be a full rank matrix with $n \ge p$. Show that projection of y onto the column space of X is $X(X^TX)^{-1}y$.

Proof.

$$X \in \mathcal{C}(X) \Leftrightarrow X = XZ \quad \text{for some Z}$$

$$\therefore \min_{X \in C(X)} \|X - b\|_2^2 = \min_{Z \in \mathbb{R}^n} \|XZ - y\|_2^2.$$

The Lagrangian function

$$L = \|XZ - y\|_2^2 = (XZ - y)^T(XZ - y)$$

$$= Z^TX^TXZ - 2y^TXZ + y^Ty$$

Calculate the partial derivative:

$$\frac{\partial L}{\partial z} = 2X^T XZ - 2X^T y = 0$$

If $X^T X$ is inverse, then

$$Z^* = (X^T X)^{-1} X^T y$$

So the projection of y on $\mathcal{C}(X)$ is denoted by

$$\text{Proj}_{\mathcal{C}(X)}(y) = Xz^* = X(X^T X)^{-1} X^T y.$$

□

1.2.4 Example: Eigenvalue Decomposition

Note

Eigenvalue Decomposition
A symmetric matrix $A \in \mathbb{R}^{n \times n}$ can be expressed as

$$A = U\Lambda U^T$$

where $U = (\mathbf{u}_1, \ldots, \mathbf{u}_n)$ is an orthogonal matrix and $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ is diagonal. $\mathbf{u}_i$ is an eigenvector of A with eigenvalue λ_i.

Let A be a symmetric matrix with $A = U\Lambda U^T$.

- Verify that $tr(A) = \sum_{i=1}^n \lambda_i$.
- Find $\max_{x \in \mathbb{R}^n} x^T Ax$ s.t. $\|x\| = 1$ where A is symmetric.

Proof sketch. The proof is as follows.

- $tr(A) = tr(U\Lambda U^T) = tr(U^T U\Lambda) = tr(\Lambda)$.
- The Lagrangian function:

$$L = x^T Ax + \mu(1 - x^T x)$$

$$\frac{\partial L}{\partial x} = 2Ax - 2\mu x = 0$$

Therefore $Ax = \mu x$, which implies (X, μ) are eigenvectors and the corresponding eigenvalues of A.

□

1.3 Applications of MVN (I): Gaussian Graphical Models

1.3.1 Conditional Independence

Let $X_1, \ldots, X_n \overset{i.i.d.}{\sim} N(\mu, \Sigma)$, $X = (X_1, \ldots, X_n)$. The MLE for mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d \times d}$ are

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} X_i = \frac{1}{n} X^T \mathbf{1}_n$$

$$\hat{\Sigma} = \frac{1}{n} (X - \mathbf{1}_n \hat{\mu}^T)^T (X - \mathbf{1}_n \hat{\mu}^T)$$

Let $\Theta = \Sigma^{-1}$ denote the precision matrix. We have:

- $\Sigma_{ij} = 0 \Leftrightarrow X_i \perp X_j$;
- $\Theta_{ij} = 0 \Leftrightarrow X_i \perp X_j | X_{-(i,j)}$.

We estimate the dependence structure by estimating Σ and Θ.

1.3.2 Graphical Models

Consider a collection $X = X_1, \ldots, X_d$ of random variables. We want to represent dependence structure within X.
For an undirected graphical model, it has three key components:

- A graph $G = (V, E)$.
- Nodes (vertices) $V = \{X_1, \ldots, X_d\}$.
- Edges $E \subseteq V \times V$.

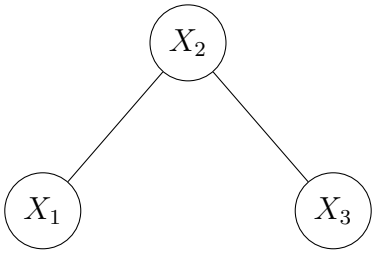
1.3.3 Pairwise Markov Property

The Pairwise Markov Property of an undirected model:

$$(i, j) \notin E \quad \Longleftrightarrow \quad X_i \perp X_j \mid X_{-(i,j)}.$$

Pairwise Markov Property $\Longleftrightarrow$ Global Markov Property $(A \perp B \mid C)$ for subsets of nodes A and B separated by C.

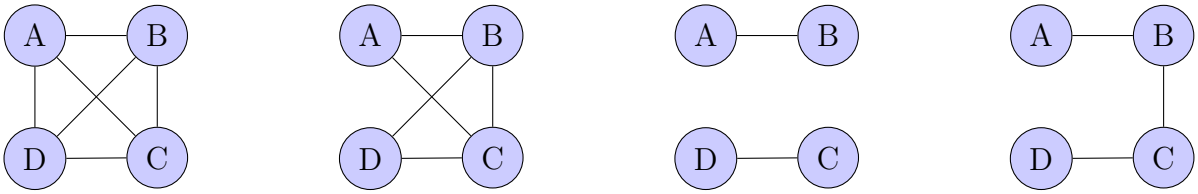
C separates A and B if no path between A and B when C is removed from the graph.



Set of nodes: $V = \{X_1, X_2, X_3\}$; Set of edges: $E = \{(X_1, X_2), (X_2, X_3)\}$.

Here we have $X_1 \perp X_3 | X_2$.

Examples 1.3.1



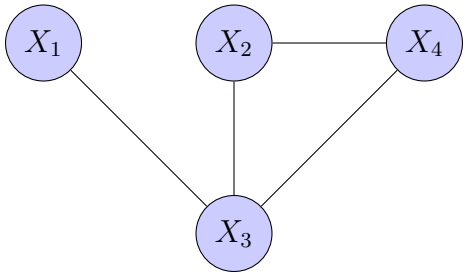
1. completely dependent;
2. $A \perp D | B, C$;
3. $A, B \perp C, D$;
4. $A \perp D$, $A \perp C | B$, $B \perp D | C$.

An easier method for the problem above

- Remove B then there is no line connecting A and C, then $A \perp C | B$;
- Remove B and C then there is no line connecting A and D, then $A \perp D | B, C$;
- If there is no line connecting A and D, then $A \perp D$.

Examples 1.3.2

Draw an undirected graphical model for X_1, X_2, X_3, X_4 with $X_1 \perp X_2 | X_3$.



1.3.4 Joint Distribution

A graph $G = (V, E)$ specifies conditional independence structure. A **clique** of G is a fully connected (complete) subgraph C of G. A **maximal clique** C of G is a clique if $C \cup \{v\}$ is not a clique for some $v \in V \setminus C$.

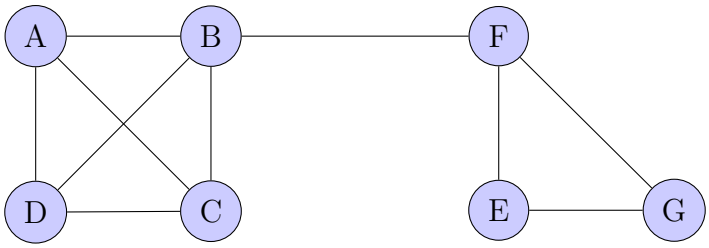
Let $\mathcal{C}$ be set of maximal cliques of G. Then

$$p(X) \propto \prod_{C \in \mathcal{C}} \phi_C(X_C)$$

where $\phi(C)$ are called **potential functions**.

Theorem
Markov properties induced by graphical representation hold $\Leftrightarrow$ joint distribution factorized by potential functions on cliques.

Examples 1.3.3
Find the maximal cliques and represent the joint distribution by the product of potential functions.



$\{A, B\}$: clique but not maximal;
$\{A, B, C\}$: clique but not maximal;
$\{A, B, C, D\}$: maximal clique;
Besides, $\{E, F, G\}$ and $\{B, F\}$ are also maximal cliques.
The joint distribution can be factorize by

$$P(X) \propto \phi_{ABCD}(A, B, C, D)\phi_{BF}(B, F)\phi_{EFG}(E, F, G)$$

1.3.5 Estimation of Gaussian Graphical Models

Consider $X = (X_1, \ldots, X_d) \sim N_d(\mu, \Sigma)$. As mentioned above, $\Theta_{ij} = 0 \Leftrightarrow X_i \perp X_j | X_{-(i,j)}$, which means there is no edge between X_i and X_j, i.e., $(i,j) \in E \Leftrightarrow \Theta_{ij} \neq 0$.
Given Θ, one can construct a Graphical model. However, estimate $\hat{\Theta}$ can be dense, leading to a complete graph. So the question is **how to get a sparse estimate of** Θ?

1. **estimation ignoring all constraints (e.g., some entries are 0)**
   Let $X_1, \ldots, X_n \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$. The Likelihood function is given by:

$$L(\boldsymbol{\mu}, \Sigma | X_1, \ldots, X_n) = \prod_{i=1}^{n} \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(X_i - \boldsymbol{\mu})^T \Sigma^{-1} (X_i - \boldsymbol{\mu})\right].$$

   Taking the logarithm:

$$\ell(\boldsymbol{\mu}, \Sigma) = -\frac{n}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{n}(X_i - \boldsymbol{\mu})^T \Sigma^{-1}(X_i - \boldsymbol{\mu}) + constant$$

   To find $\hat{\boldsymbol{\mu}}$, take the derivative with respect to $\boldsymbol{\mu}$:

$$\frac{\partial \ell}{\partial \boldsymbol{\mu}} = -\frac{n}{2}\Sigma^{-1}\sum_{i=1}^{n}(X_i - \boldsymbol{\mu}) = 0$$

   So we obtain the MLE of μ:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{i=1}^{n} X_i = \bar{X}$$

   For Σ, consider the trace identity:

$$\sum_{i=1}^{n}(X_i - \bar{X})^T \Sigma^{-1}(X_i - \bar{X}) = tr\left(\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T \Theta\right).$$

   Define the sample covariance matrix (biased):

$$S = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T.$$

   Thus:

$$\ell = -\frac{n}{2}tr(S\Theta) + \frac{n}{2}\log|\Theta|.$$

   Taking the derivative w.r.t. Θ:

$$\frac{\partial \ell}{\partial \Theta} = -\frac{n}{2}S + \frac{n}{2}\Theta^{-1} = 0$$

   Multiplying both sides by Σ, we get a dense estimate of Θ:

$$\hat{\Theta} = S^{-1} = \frac{1}{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})(X_i - \bar{X})^T}$$

2. **estimation with known structure**
   It can be considered as solving MLE considering constraints $\Theta_{ij} = 0,\ (i,j) \in E^C$ where E is the set of the known graph's edges.

   It is an optimization problem $\max_{\Theta,\mu} \ell(\mu, \Theta | X_1, \ldots, X_n) \quad s.t. \Theta_{ij} = 0,\ i,j \in E^C.$
   The Lagrangian function is denoted by

$$L = \log|\Theta| - tr(S\Theta) + \sum_{(i,j)\in E^C} \gamma_{ij}\Theta_{ij} = \log|\Theta| - tr(S\Theta) + tr(\Gamma\Theta)$$

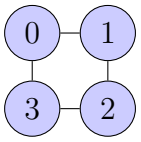
   where $\sum_{i,j} \gamma_{ij}\Theta_{ij} =< \Gamma, \Theta >= tr(\Gamma\Theta)$ and $\gamma_{ij} = 0$ when $(i,j) \in E$, i.e., no constraint on (i,j).
   Calculate the derivative:

$$\frac{\partial L}{\partial \Theta} = \Theta^{-1} - S + \Gamma = 0$$

   So the MLE of Θ is

$$\hat{\Theta} = (S - \Gamma)^{-1}$$

   Examples 1.3.4



   For this graph, since $\gamma_{ij} = 0 \Leftrightarrow \Theta_{ij} = 0$, We set $\Gamma = \begin{pmatrix} 0 & 0 & a & 0 \\ 0 & 0 & 0 & b \\ c & 0 & 0 & 0 \\ 0 & d & 0 & 0 \end{pmatrix}$.

3. **estimation with unknown structure**
   In unknown true graph situation, we use **Graphical LASSO (G-LASSO)** method here:

$$(\hat{\mu}, \hat{\Theta}) = \arg\max_{\mu,\Theta} L(\mu, \Theta) - \lambda\|\Theta\|_1$$

   where $\|\Theta\|_1 = \sum_{j,k} |\Theta_{jk}|$ and $\lambda > 0$ is the sparsity tuning parameter.

References

(a) Chapter 17.3.2 *ESL*.

(b) Yuan, M., & Lin, Y. (2007). Model selection and estimation in the Gaussian graphical model. *Biometrika*, 94(1), 19-35.

(c) Friedman, J., Hastie, T. and Tibshirani, R., 2008. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), pp.432-441.

1.4 Applications of MVN (II): other applications

1.4.1 Probabilistic PCA

(This will be discussed more specifically in the next Chapter.)
Let $Y_{i\ p\times 1} \overset{i.i.d.}{\sim} N_p(\mu, \Sigma = \Lambda_{p\times k}\Lambda^T_{k\times p} + \sigma^2 I_p), \quad i = 1, \ldots, n$ where $p \gg k$ and $\Lambda\Lambda^T$ is a low rank matrix.
The observation Y_i can then be represented by

$$Y_i = \mu + \Lambda_{p\times k} X_{k\times 1} + W_{i\ p\times 1}$$

where $X_i \perp W_i$, the latent variables $\begin{cases} W_i \overset{i.i.d.}{\sim} N_p(0, \sigma^2 I_p) \\ X_i \overset{i.i.d.}{\sim} N_k(0, \sigma^2 I_k) \end{cases}$.
To solve the PCA solution is to solve the MLE of Λ.

1.4.2 Factor Analysis Model

(This will be discussed more specifically in Chapter 3.)
Let $Y_{i\ p\times 1} \overset{i.i.d.}{\sim} N_p(\mu, \Sigma = \Lambda_{p\times k}\Lambda^T_{k\times p} + \psi), \quad i = 1, \ldots, n$ where ψ is a diagonal matrix.
Then Y_i can be represented by

$$Y_i = \mu + \Lambda_{p\times k} X_{k\times 1} + W_{i\ p\times 1}$$

where $X_i \perp W_i$, the latent variables $\begin{cases} W_i \overset{i.i.d.}{\sim} N_p(0, \psi) \\ X_i \overset{i.i.d.}{\sim} N_k(0, \sigma^2 I_k) \end{cases}$.

1.4.3 Reduced rank regression model (RRR)

For common multivariate linear regression cases (not low-rank), we have the model $Y_{i\ q\times 1} = B_{q\times p} X_{i\ p\times 1} + E_{i\ q\times 1}, \quad E_i \sim N_q(0, \Sigma)$ where B is the unknown regression parameter.

MLE for B and Σ:

$$\begin{cases} \hat{B} = (X^T X)^{-1} X^T Y \\ \hat{\Sigma} = \frac{1}{n}(Y - X\hat{B})^T (Y - X\hat{B}) \end{cases}$$

However, when $p \gg n$, $(X^T X)^{-1}$ is not available, we call it "**low-rank**". In this situation, we use reduced rank regression instead.

For simplicity, we assume $\Sigma = I_q$. The goal becomes:

$$\hat{B} = \arg\max_B L(B) \quad s.t.\ rank(B) = k,\ k \le n, p, q$$

Be careful: It's not convex! So we use nuclear norm penalty to control the rank instead:

$$\hat{B} = \arg\max_B L(B) - \lambda\|B\|_*$$

Then add sparsity into the target function, and we obtain **row-sparse reduced rank regression model**:

$$\hat{B} = \arg\max_B L(B) - \lambda_1\|B\|_* - \lambda_2\|B\|_{1,2}$$

where $L(B) = -\frac{n}{2}\log|\Sigma| - \frac{1}{2}tr\left[\Sigma^{-1}(Y - XB^T)^T(Y - XB^T)\right]$. Here we add LASSO penalty on each row of B.

Matrix norm (see in previous section 1.2.1)

- **nuclear norm**:$\|B\|_* = \sum_{i=1}^q \sigma_i(B)$, which is the sum of all the singular values of B. Used to control the rank.
- $\|B\|_{1,2} := \sum_{i=1}^q \|B_i\|_2$ row sum of L2-norm of each row vectors. Used to add row sparsity.

One of the applications of this model is **recommendation systems**, e.g., of Amazon.

1.5 Inference of MVN

1.5.1 Wishart Distribution

Theorem
Suppose we have data $X_1, \ldots, X_n \overset{i.i.d.}{\sim} N_p(\mu, \Sigma)$ with sample mean $\bar{X} = \frac{1}{n}\sum_{i=1}^n X_i$ and sample covariance matrix $S = \frac{1}{n-1}\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})^T = \frac{1}{n-1}A$. Then,

$$\begin{cases} \bar{X} \perp S \\ \bar{X} \sim N_p(\mu, \frac{1}{n}\Sigma) \\ S_{p\times 1} \sim W_p(n-1, \frac{1}{n-1}\Sigma) \end{cases}$$

$W_p(n-1, \frac{1}{n-1}\Sigma)$ means Wishart distribution with $df = n-1$. $A \sim W_p(n-1, \Sigma)$.

Definition If $X_1, \ldots, X_n \overset{i.i.d.}{\sim} N_p(0, \Sigma)$, let $A = \sum_{i=1}^n X_i X_i^T$, then $A \sim W_p(n, \Sigma)$.
(It is like "multivariate Chi-square": $X_1, \ldots, X_n \overset{i.i.d.}{\sim} N_p(0,1), Z = \sum_{i=1}^n X_i^2 \sim \chi_n^2$.)

Properties

1. For $A \sim N_p(\mu, \Sigma)$ $\quad n > p$, the PDF is

$$\frac{1}{2^{np/2}|\Sigma|^{n/2}\Gamma_p(\frac{n}{2})} \exp\{-\frac{1}{2}tr(\Sigma^{-1}A)\}|A|^{\frac{n-p-1}{2}}$$

   For MVN, the PDF is $f \propto |\Sigma|^{-n/2}\exp\{-\frac{1}{2}tr(\Sigma^{-1}A)\}$.

2. If $p = 1$, $A \sim W_1(n, \sigma^2)$, that is $\frac{1}{\sigma^2}A \sim \chi_1^2$;

3. Preservation under linear transformation:
   Let $M \in R^{k\times p}$, $A \sim W_p(n, \Sigma)$. Then $M_{k\times p}A_{p\times p}M_{p\times k}^T \sim W_k(n, M\Sigma M_{k\times k}^T)$;
   One-dimension case: if $a \in R^p$, $A \sim W_p(n, \Sigma)$, then $aAa^T \sim (a\Sigma a^T)\chi_n^2$

4. If $A \sim W_p(n, \Sigma)$, then A^{-1} has an inverse Wishart distribution.

1.5.2 Hotelling T^2 test

One-sample case Suppose $X_1, \ldots, X_n \overset{i.i.d.}{\sim} N_p(\mu, \Sigma)$. Our goal is to test $H_0 : \mu_{p\times 1} = 0$ V.S. $H_A : \mu_{p\times 1} \neq 0$.
(If $X_{11}, \ldots, X_{1p}$ independent, we can use t-test for p times. If not, we use Hotelling T^2 test.)
The test statistic is $T^2 = n\bar{X}^T S^{-1}\bar{X}$ where $S = \frac{1}{n-1}\sum_{i=1}^n (X_i - \bar{X})(X_i - \bar{X})^T$.

Under H_0, we have

$$\frac{T^2}{n-1} \cdot \frac{(n-1)-p+1}{p} \sim F_{p,(n-1)-p+1} = \frac{\frac{\chi_p^2}{p}}{\frac{\chi^2_{(n-1)-p+1}}{(n-1)-p+1}} = \frac{\frac{\chi_p^2}{p}}{\frac{\chi^2_{n-p}}{n-p}}$$

(Under H_A, we have

$$\frac{T^2}{n-1}\cdot\frac{(n-1)-p+1}{p}\sim F_{p,(n-1)-p+1}(\delta)=\frac{\frac{\chi^2_p(\delta)}{p}}{\frac{\chi^2_{(n-1)-p+1}}{(n-1)-p+1}}=\frac{\frac{\chi^2_p(\delta)}{p}}{\frac{\chi^2_{n-p}}{n-p}}$$

where $\delta=n\mu^T\Sigma^{-1}\mu$. It's a non-central F distribution.)

Theorem
Likelihood Ratio Test (**LRT**) is equivalent to Hotelling T^2 test.

$$\Lambda(X)=\frac{\sup\limits_{\Theta_0}L(0,\Sigma)}{\sup\limits_{\Theta_A}L(\mu,\Sigma)}=\frac{\sup\limits_{\mu}L(0,\Sigma)}{\sup\limits_{\mu,\Sigma}L(\mu,\Sigma)}=(\frac{1}{1+\frac{T^2}{n-1}})^{\frac{n}{2}}$$

So

$$-2\log\Lambda(X)=n\log(1+\frac{T^2}{n-1})\approx T^2$$

See the Proof in HW 1.

Two-sample case Suppose we have two samples $X_1,\dots,X_{n_X}\overset{i.i.d.}{\sim}N_p(\mu_X,\Sigma)$ and $Y_1,\dots,Y_{n_Y}\overset{i.i.d.}{\sim}N_p(\mu_Y,\Sigma)$. The goal is to test $H_0:\mu_X=\mu_Y$ V.S. $H_A:\mu_X\neq\mu_Y$. The test statistic is

$$T^2=\frac{n_Xn_Y}{n_X+n_Y}(\bar{X}-\bar{Y})^TS_p^{-1}(\bar{X}-\bar{Y})$$

where S_p is the pooled covariance matrix $S_p=\frac{1}{n_X-1+n_Y-1}[(n_X-1)S_X+(n_Y-1)S_Y]$. S_X,S_Y are sample covariance matrices.

In the test,

$$\frac{T^2}{n_X+n_Y-2}\cdot\frac{n_X+n_Y-p+1}{p}\sim F_{p,n_X+n_Y-p+1}(\delta)$$

where $\delta=\frac{n_Xn_Y}{n_X+n_Y}(\mu_X-\mu_Y)^T\Sigma^{-1}(\mu_X-\mu_Y)$. Under H_0, It follows central F distribution.

Chapter 2

Principal Component Analysis (PCA)

2.1 PCA

2.1.1 Basic knowledge about PCA

Data We have high-dimensional data $y_1, \ldots, y_n \in \mathbb{R}^q$,i.e., $Y = (y_1, \ldots, y_n) \in \mathbb{R}^{n \times q}$.

Goal Dimension reduction. In practice, q can be huge, and we want to learn low-dimensional embeddings $X_1, \ldots, X_n \in \mathbb{R}^k$ ($k \ll q$) of $y_1, \ldots, y_n \in \mathbb{R}^q$ that best represent the original data.

Assumption We believe that the data vectors lie mostly near a low-dimensional linear space (submanifold) M (k-dim, $k \ll q$). **(No distribution assumption on $y_1, \ldots, y_n$!)**

What to do Estimate M from $y_1, \ldots, y_n$.
What we try to do is to project $y_1, \ldots, y_n \in \mathbb{R}^q$ onto a k-dimensional subspace while "preserving as much information as possible".

Applications

- Data visualization;
- learn latent factors for intepretations;
- clustering;
- principal component regression;
- classification.

pre-process Before PCA, one must center the data Y such that $Y^T \mathbf{1}_n = 0$, i.e., $\sum_{i=1}^n y_i = 0$.

Terminology

1. **Data** $Y \in \mathbb{R}^{n\times q}$: $Y = (Y_1, \ldots, Y_n)$ where $y_i \in \mathbb{R}^q$ and are centered;

2. **Loadings** $U \in \mathbb{R}^{q\times k}$: $U = (u_1, \ldots, u_k)$ is a set of orthonormal directions that "matters the most". U is an orthogonal matrix, i.e., $U^TU = I$. The manifold M can be represented in terms of K basis vectors $\mu_1, \ldots, \mu_k \in \mathbb{R}^q$;

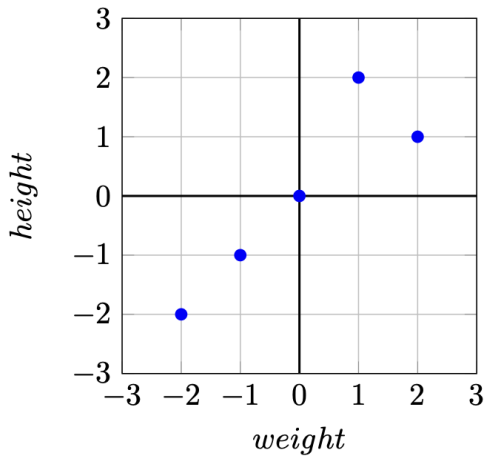
3. **Scores** $X = YU \in \mathbb{R}^{n\times k}$: $X_i = U^TY_i \in \mathbb{R}^k$ is the score of y_i. It is the low-dimensional representation of y_i and can be regarded as the coordinate along the directions in U that represents y_i.
   Each point in M can be represented as $u_1X_1 + \cdots + u_kX_k$. For $i = 1, \ldots, n$, we can represent our data as

$$y_i \approx u_1X_{i1} + \cdots + u_kX_{ik} = U_{q\times k}X_{i\ k\times 1}$$

example 2.1.1

$y_i \in \mathbb{R}^2$. Two features: height, weight. Centered. Suppose we run PCA to reduce the dimension to 1.

- Find principal loading u_1.
- What are the scores of the 5 observations?



The principal loading $u_1 = \frac{1}{\sqrt{2}}(1,1)^T$.

The score of each point is the location of the point's projection onto the loading vector. The scores are $-2\sqrt{2}, -\sqrt{2}, 0, \frac{3}{2}\sqrt{2}, \frac{3}{2}\sqrt{2}$.

2.1.2 Optimization and Solutions for PCA

1. **For $k = 1$ case**:
   We only have one projecting direction $u_1 \in \mathbb{R}^q$ with $u_1^Tu_1 = 1$. We project

centered data $y_i \to (u_1 u_1^T) y_i = u_1 (u_1^T y_i) = u_i X_{i1}$.
The goal of PCA is to maximize the variance of $X_{i1}, \ldots, X_{ik}$, that is, to solve

$$\max_{u_1 \in \mathbb{R}^q} \frac{1}{n} \sum_{i=1}^{n} (X_i - \bar{X})^2 \quad s.t.\ u_1^T u_1 = 1$$

$$= \max_{u_1 \in \mathbb{R}^q} \frac{1}{n} \sum_{i=1}^{n} (u_1^T y_i - u_1^T \bar{y})^2 \quad s.t.\ u_1^T u_1 = 1$$

We know

$$\frac{1}{n} \sum_{i=1}^{n} (u_1^T y_i - u_1^T \bar{y})^2 = \frac{1}{n} \sum_{i=1}^{n} u_1^T (y_i - \bar{y})(y_i - \bar{y})^T u_1$$

$$= u_1^T \left(\frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})(y_i - \bar{y})^T \right) u_1 = u_1^T S u_1$$

where S is the sample covariance matrix $S = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y})(y_i - \bar{y})^T$.
Then the optimization problem becomes

$$\max_{u_1 \in \mathbb{R}^q} u_1^T S u_1 \quad s.t.\ u_1^T u_1 = 1$$

We introduce Lagrange multiplier:

$$\max_{u_1 \in \mathbb{R}^q} u_1^T S u_1 - \lambda (u_1^T u_1 - 1)$$

Take derivative w.r.t. u_1, we obtain

$$2 S u_1 - 2 \lambda u_1 = 0$$

that is $S u_1 = \lambda u_1$, which indicates λ is the eigenvalue of S and u_1 is the corresponding eigenvector.
Then replace $u_1^T S u_1 - \lambda (u_1^T u_1 - 1)$ with $S u_1 = \lambda u_1$, the problem becomes

$$\max_{u_1 \in \mathbb{R}^q} u_1^T \lambda u_1 - \lambda (u_1^T u_1 - 1) = \underset{u_1 \in \mathbb{R}^q}{\lambda}$$

This means λ is the largest eigenvalue of matrix S and u_1 is the corresponding eigenvector.

2. **For general cases** k:
The subspace is $U = [u_1, \ldots, u_k]_{q \times k}$, $U^T U = I_k$, and here we project $y_i \to U_{q \times k} U^T y_i = u_1 X_{i1} + \cdots + u_k X_{ik}$.
The optimization problem becomes

$$\max_{U \in \mathbb{R}^{q \times k}} \sum_{j=1}^{k} u_j^T S u_j \quad s.t.\ U^T U = I_k$$

that is, $u_i^T u_i = 1$ and $u_i^T u_j = 0,\ \forall i \neq j$.

examples 2.1.2
Show that maximizing the variance of scores is equivalent to minimizing projection error.

$$\begin{aligned}
&\min_U \|Y - YUU^T\|_F^2 \\
&= \min_U tr\big((Y - YUU^T)^T(Y - YUU^T)\big) \\
&= \min_U tr\big(Y^TY - Y^TYUU^T - UU^TY^TY + UU^TY^TYUU^T\big) \\
&= \min_U tr\big(Y^TY - Y^TYUU^T\big) \quad (\because tr(UU^TY^TYUU^T) = tr(UU^TY^TY)) \\
&= \max_U tr\big(Y^TYUU^T\big) = \max_U tr\big(U^TY^TYU\big) \\
&= \max_U tr\big(U^TSU\big) = \max_U \sum_{i=1}^{k} u_i^T S u_i
\end{aligned}$$

Solutions to general cases:

- **Method 1: solve them iteratively**:

  Step 1: solve $\max_{u_1 \in \mathbb{R}^q} u_1^T S u_1 \quad s.t.\ u_1^T u_1 = 1$;
  Step j ($j \geq 2$): solve $\max_{u_j \in \mathbb{R}^q} u_j^T S u_j \quad s.t.\ u_j^T u_j = 1,\ u_k^T u_j = 0,\ \forall k < j$.

- **Method 2: solve them simultaneously (Eigen-Decomposition)**:

  $$\max_{U \in \mathbb{R}^{q \times k}} \sum_{j=1}^{k} u_j^T S u_j - \sum_{j=1}^{k} \lambda(u_j^T u_j - 1) - \sum_{j \neq k} u_j^T u_k$$

  It is equivalent to

  $$\max_{U \in \mathbb{R}^{q \times k}} tr(U^TSU) - tr(\Lambda(U^TU - I_k)), \quad \Lambda \text{ symmetric}$$

  Take derivative:

  $$2SU - 2U\Lambda = 0 \Rightarrow SU = U\Lambda \Rightarrow \Lambda = U^TSU$$

  Eigen-Decomposition: let $\Lambda = ODO^T$, $O^TO = I$, where D is diagonal.
  Then $SUO = UOD$.
  If we let $U^* = UO$, we then have $SU^* = U^*D$, which means

  $$S = U^*DU^{*T}, \quad U^{*T}U^* = I$$

U^* is the set of eigenvectors of S with eigenvalues $d_1, \ldots, d_k$.
The problem becomes

$$\max_{U^* \in \mathbb{R}^{q \times k}} \sum_{j=1}^{k} u_j^{*T} D u_j^* \quad s.t.\ U^{*T} U^* = I_k$$

In computing, we first perform eigen-decomposition on S and get O and D. Then we solve the optimization problem and get U^*. The loading matrix is given by $U = U^* O^T$.

- **Method 3: Singular Value Decomposition (SVD)**:
  In method 2 we perform eigen-decomposition on sample covariance matrix S. In method 3, we directly use SVD on the centered data matrix $\bar{Y}$.

  Let $\bar{Y}$ denotes the centered data:

$$\bar{Y}_{n \times q} = \begin{pmatrix} y_1 - \bar{y} \\ \vdots \\ y_n - \bar{y} \end{pmatrix}$$

  Let $\bar{Y}_{n \times q} = U_{n \times q} D_{q \times q} V_{q \times q}^T$, where

  $U_{n \times q}$: orthogonal, $U^T U = I_q$. Columns of U are left singular vectors;

  $V_{q \times q}$: orthogonal, $V^T V = I_q$. Columns of V are right singular vectors;

$$D_{q \times q} : \text{ diagonal, } D = \begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_q \end{pmatrix} . d_1 \geq \cdots \geq d_q.$$

  Note: The D here is not the same as the one in Method 2 !
  Sample covariance

$$S = \frac{1}{n} \bar{Y}^T \bar{Y} = \frac{1}{n} V D U^T U D V^T = \frac{1}{n} V D^2 V^T$$

  Therefore, the eigenvalues of S are $\frac{d_i^2}{n}$; the eigenvectors are contained in V. The first k columns of V are the principal loadings.

2.1.3 Properties of PCA

Consider population version:

$$\max_{U \in \mathbb{R}^{q \times k}} tr(U^T \Sigma U) \quad s.t.\ U^T U = I_k$$

where U contains the top k eigenvectors of Σ.

- For the PCA scores $X = YU$,

$$E(X) = E(Y)U = 0$$

$$cov(X) = U^T cov(Y)U = U^T \Sigma U = \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_k \end{pmatrix}$$

  where $\lambda_1 \geq \cdots \geq \lambda_k$ are the top k eigenvalues.
- When $k = q$, the total variance is $tr(\Sigma) = \sum_{j=1}^{k} \lambda_j$. The j-th component accounts for $\frac{\lambda_j}{\sum_{j=1}^{k} \lambda_j}$ proportion of variance.
- Another way to look at PCA:
  Approximate data by $\tilde{y}_i = \mu + UX_i$ and our goal is to find the best linear combination that gives minimum aprroximation error.

2.1.4 Probabilistic PCA (for a spiked covariance model)

This can be thought of a MLE of latent variable model:

$$Y_{q\times 1} = \mu_{q\times 1} + \Lambda_{q\times k} X_{k\times 1} + W_{q\times 1}$$

where $W \perp X, \quad W \sim N_q(0, \sigma^2 I_q), \quad X \sim N_k(0, I_k)$ are latent variables.
Therefore, we have $Y \sim N_q(\mu, \sigma^2 I_q + \Lambda\Lambda^T)$ where $\Lambda\Lambda^T$ has low rank ($k \ll q$). The column space of Λ corresponds to the top k eigenvectors of $\Sigma = \sigma^2 I_q + \Lambda\Lambda^T$.

1. When $k = 1$, $y \sim N_q(\mu, \sigma^2 I_q + \Lambda\Lambda^T)$, where$\Lambda$ is a q-dim vector.
   The first PCA vector is $\frac{\Lambda}{\|\Lambda\|_2}$ and the variance is $\sigma^2 + \|\Lambda\|_2^2$.

   Proof sketch.
   $$(\sigma^2 I_q + \Lambda\Lambda^T)\frac{\Lambda}{\|\Lambda\|_2} = (\sigma^2 + \|\Lambda\|_2^2)\frac{\Lambda}{\|\Lambda\|_2}$$
   □

2. k-dimension:
   The log likelihood is given by

   $$\ell = \log L = -\frac{nq}{2}\log(2\pi) - \frac{n}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{n}(y_i - \mu)^T\Sigma^{-1}(y_i - \mu)$$

   We let $\frac{\partial \ell}{\partial \mu} = 0$ and get the MLE of μ is $\hat{\mu} = \bar{y}$. After we subtract $\bar{y}$ and set $S = \sum_{i=1}^{n}(y_i - \bar{y})(y_i - \bar{y})^T$, the log likelihood becomes

   $$\ell = -\frac{n}{2}[q\log(2\pi) + \log|\Sigma| + tr(\Sigma^{-1}S)]$$

   We can solve this using EM algorithm, which will be introduced in Chapter 4.

References

- Chapter 17.2, *PRML*;
- Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 61(3), 611-622.

2.1.5 PCA for High-dimensional data ($n < q$)

Same as 2.1.2 Method 3(2), we use $S_{q\times q} = \frac{1}{n}\bar{Y}^T\bar{Y}$ notation. S has q eigenvalues but at least $q-n$ are 0 because of rank. The computational cost for finding q eigenvectors is $O(q^3)$.
So we consider $G_{n\times n} = \frac{1}{n}\bar{Y}\bar{Y}^T$.
Suppose the eigenvectors of S are $u_1, \ldots, u_k$, we have $Su_j = \lambda_j u_j, \quad j = 1, \ldots, k$.
Multiply both sides by $\bar{Y}$:

$$\frac{1}{n}\bar{Y}\bar{Y}^T\bar{Y}u_j = \lambda_j \bar{Y}u_j$$

which means

$$G\bar{Y}u_j = \lambda_j \bar{Y}u_j$$

Let $\gamma_j = \bar{Y}u_j$, then

$$G\gamma_j = \lambda_j\gamma_j$$

γ_j are the eigenvectors for G. $\|\gamma_j\| = 1$. Thus the computational cost to get γ_j's is $O(n^3) < O(q^3)$.

$$u_j = \frac{\bar{Y}^T\gamma_j}{\|\bar{Y}^T\gamma_j\|_2} = \frac{1}{\sqrt{n\lambda_j}}\bar{Y}^T\gamma_j$$

Proof sketch.

$$\|\bar{Y}^T\gamma_j\|_2^2 = \gamma_j^T\bar{Y}\bar{Y}^T\gamma_j = n\lambda_j$$

□

2.2 Sparse PCA

For data $Y_{q\times 1} = (y-1,\ldots,y_q)$. Assume $u_1,\ldots,u_k$ to be sparse, e.g., $u_1 = (\frac{\sqrt{2}}{2},\frac{\sqrt{2}}{2},0,\ldots,0)$.

2.2.1 Method 1: Based on LASSO

Start with u_1:

$$u_1 = \arg\max_{u_1} u_1^T S u_1, \quad s.t.\ u_1^T u_1 = 1 \ and\ \|u_1\| \le t$$

Then solve $u_2,\ldots,u_k$ sequentially.

References

Jolliffe, I.T., Trendafilov, N.T., & Uddin, M. (2003). A Modified Principal Component Technique Based on the LASSO. *Journal of Computational and Graphical Statistics*, 12(3), 531–547.

2.2.2 Method 2: Hui Zou's method

The PCA method becomes

$$\min_{\Theta,\mu} \sum_{i=1}^{n} \|y_i - \Theta U^T y_i\|_2^2 + \lambda_1\|U\|_1 + \lambda_2\|U\|_2^2 \quad s.t.\ U^T U = I_k$$

First, we fix Θ and solve U; then we fix U and solve Θ. Repeat the procedure until they reach convergence.

References

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of computational and graphical statistics*, 15(2), 265-286.

2.2.3 Method 3: Fantope Projection for PCA

The problem becomes

$$\max\ tr(SX) - \lambda\|X\|_{1,1} \quad s.t.\ X \in \mathcal{F}^k$$

where $\|X\|_{1,1} = \sum_{j,k}|X_{jk}|$ and $\mathcal{F}^k = \{X : 0 \le X \le I, tr(X) = k\}$.

This method is **CONVEX** and all other sparse PCA methods are not! Thus we can find the optimal value.

Assumptions

- S, Σ are symmetric;
- Let $\delta(\Sigma) = \lambda_k(\Sigma) - \lambda_{k+1}(\Sigma) > 0$;
- Sparsity - projection Π: the projection of data X onto subspace spanned by eigenvectors of Σ corresponding to k largest eigenvalues of Σ.
  This satisfies
  $$\|\Pi\|_{2,0} \leq c \quad \text{or} \quad \|diag(\Pi)\|_0 \leq c$$
  where c is a constant, $\|\Pi\|_{2,0} = \sum_{i=1}^n \mathbb{I}_{\{\|\Pi_{i,:}\|_2 \neq 0\}}$ is the number of non-zero rows in Π, $\|diag(\Pi)\|_0 = \sum_{i=1}^n \mathbb{I}_{\{\Pi_{ii} \neq 0\}}$ is the number of non-zeros diagonal entries in Π.

Theorem If $\lambda > \|\Pi\|_{\infty,\infty}$ and $S \geq \|\Pi\|_{2,0}$, then

$$\|\hat{X} - \Pi\|_F \leq \frac{4S\lambda}{\delta}$$

where $\delta = \lambda_k(\epsilon) - \lambda_{k+1}(\epsilon) > 0$, $\|\mathrm{A}\|_{\infty,\infty} = \max\limits_{1 \leq i \leq m} \sum_{j=1}^n |a_{ij}|$.

Proof. Let $\hat{X}$ be the optimal solution of

$$\min \ -tr(SX) + \lambda\|X\|_{1,1} \quad s.t.\ X \in \mathcal{F}^k$$

$$-tr(S\hat{X}) + \lambda\|\hat{X}\|_{1,1} \leq -tr(S\Pi) + \lambda\|\Pi\|_{1,1}$$

$$\Rightarrow 0 \leq tr(S(\hat{X} - \Pi)) + \lambda\|\Pi\|_{1,1} - \lambda\|\hat{X}\|_{1,1}$$

Let $\hat{\Delta} = \hat{X} - \Pi$, then:

$$0 \leq tr(S\hat{\Delta}) + \lambda\|\Pi\|_{1,1} - \lambda\|\hat{\Delta} + \Pi\|_{1,1}$$

$$0 \leq tr(S\hat{\Delta}) - \lambda(\|\hat{\Delta} + \Pi\|_{1,1} - \|\Pi\|_{1,1})$$

By Lemma in Vu et al. (2013), we have $\frac{\delta}{2}\|\hat{\Delta}\|_F^2 \leq -tr(\Sigma\hat{\Delta})$. Then:

$$\frac{\delta}{2}\|\hat{\Delta}\|_F^2 + tr(\Sigma\hat{\Delta}) \leq tr(S\hat{\Delta}) - \lambda(\|\hat{\Delta} + \Pi\|_{1,1} - \|\Pi\|_{1,1})$$

$$\frac{\delta}{2}\|\hat{\Delta}\|_F^2 \leq tr(S - \Sigma)\hat{\Delta} - \lambda(\|\hat{\Delta} + \Pi\|_{1,1} - \|\Pi\|_{1,1})$$

Using Hölder's inequality:

$$\frac{\delta}{2}\|\hat{\Delta}\|_F^2 \leq \|S - \Sigma\|_{\infty,\infty}\|\hat{\Delta}\|_{1,1} - \lambda(\|\hat{\Delta} + \Pi\|_{1,1} - \|\Pi\|_{1,1})$$

$$\leq \lambda(\|\hat{\Delta}\|_{1,1} - \|\hat{\Delta} + \Pi\|_{1,1} + \|\Pi\|_{1,1})$$

Consider

$$\|\hat{\Delta}\|_{1,1} - \|\hat{\Delta} + \Pi\|_{1,1} + \|\Pi\|_{1,1}$$
$$= \|\hat{\Delta}_J\|_{1,1} - \|\hat{\Delta}_J + \Pi_J\|_{1,1} + \|\Pi_J\|_{1,1}$$

where J is the subset of indices of nonzero entries of Π, hence

$$\leq 2\|\hat{\Delta}_J\|_{1,1}$$

Therefore,

$$\frac{\delta}{2}\|\hat{\Delta}\|_F^2 \leq 2\lambda\|\hat{\Delta}_J\|_{1,1}$$

Because J has at most S^2 entries,

$$\frac{\delta}{2}\|\hat{\Delta}\|_F^2 \leq 2\lambda\|\hat{\Delta}_J\|_{1,1} \leq 2S\lambda\|\hat{\Delta}_J\|_F \leq 2S\lambda\|\hat{\Delta}\|_F$$

So

$$\frac{\delta}{2}\|\hat{\Delta}\|_F \leq 2S\lambda$$

which indicates

$$\|\hat{X} - \Pi\|_F \leq \frac{4S\lambda}{\delta}$$

□

References

Vu, V. Q., Cho, J., Lei, J., & Rohe, K. (2013). Fantope projection and selection: A near-optimal convex relaxation of sparse PCA. *Advances in neural information processing systems*, 26.

2.2.4 Method 4: Truncated power method

In this method, the problem becomes

$$\arg\max_{u\in\mathbb{R}^q} u^T S u, \quad s.t.\ u^T u = 1,\ \|u\|_0 \leq S$$

Here we have $\|\hat{u} - u\|_2 \leq c\sqrt{S\log(\frac{q}{n})}$, which is better the theorem's result above ($\|\hat{u} - u\|_2 \leq \frac{4S}{\delta}\sqrt{\log(\frac{q}{n})}$) when we choose $\lambda = \sqrt{\log(\frac{q}{n})}$ because here S lies inside the square root.

References

Yuan, X. T., & Zhang, T. (2013). Truncated power method for sparse eigenvalue problems. *The Journal of Machine Learning Research*, 14(1), 899-925.

Chapter 3

Factor Analysis

3.1 Factor Model

3.1.1 Model

It is a model-based approach for dimension reduction.
Assume data lie in a low-dimension linear space. The model is

$$Y_{q\times 1} = \mu_{q\times 1} + \Lambda_{q\times k} X_{k\times 1} + W_{q\times 1}$$

where $W \perp X$, $X \sim N_k(0, I_k)$, $W \sim N_q(0, \Psi)$ and $\Psi = \begin{pmatrix} \psi_1 & 0 & \cdots & 0 \\ 0 & \psi_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \psi_q \end{pmatrix}$ where $\psi_i > 0, \quad i = 1, \ldots, q$.
In this model, Λ is the loading matrix, X is the factors and W is the random noise.

Under the Gaussian assumption,

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} I & \Lambda \\ \Lambda^T & \Lambda\Lambda^T + \Psi \end{pmatrix})$$

where X is latent and Y can be observed. It is equivalent to $Y \sim N(0, \Lambda\Lambda^T + \Psi)$.

The GOAL of Factor model is to estimate μ, Λ, Ψ using only $Y_1, \ldots, Y_n$.

3.1.2 Estimation

The log likelihood function is given by

$$\ell(\theta|y) = -\frac{n}{2}\log\det(\Lambda\Lambda^T + \Psi) - \frac{1}{2}\sum_{i=1}^{n}(y_i - \mu)^T(\Lambda\Lambda^T + \Psi)^{-1}(y_i - \mu) + c$$

1. μ:

$$\frac{d\ell}{d\mu} = -\sum_{i=1}^{n}(\Lambda\Lambda^T + \Psi)^{-1}(y_i - \mu) = 0$$

So the MLE of μ is easy to calculate, which is $\hat{\mu} = \bar{y}$.

However, the MLE's for Λ and Ψ are challenging. (they are not identifiable!)

2. Λ:
No unique Λ since $\ell(\theta|y)$ depends on Λ only through $\Lambda\Lambda^T$. If we have an orthogonal matrix R where $RR^T = I$, then $(\Lambda R)(\Lambda R)^T = \Lambda\Lambda^T$.

To make Λ identifiable:

if Λ is constrained to satisfy $\Lambda_{q\times k} = \begin{pmatrix} 1 & \cdots & 0 \\ & \ddots & \vdots \\ \star & & 1 \\ & \star & \end{pmatrix}$ (add $\frac{k(k+1)}{2}$ constraints),

then Ψ is diagonal, Λ, μ, Ψ are all identifiable.

References

- Chapter 14, *An Introduction to Multivariate Statistical Analysis* by T.W.Anderson;
- Bai, J., & Li, K. (2012). Statistical analysis of factor models of high dimension.

3.1.3 Select number of factors k

Upper bound of k The degree of freedom is given by

$$df = \frac{q(q+1)}{2} - [q(k+1) - \frac{k(k-1)}{2}] = \frac{1}{2}[(q-k)^2 - (q+k)]$$

in which the first one is from symmetric covariance matrix of Y $\Sigma_{q\times q}$ and the second one is from $\Psi + \Lambda\Lambda^T$ while q is from Ψ, qk is from Λ, $\frac{k(k-1)}{2}$ means the number of elements that constrained to be 0 in Λ.
Because we need $df > 0$, we get the upper bound of k.

methods to select k

- scree plot;
- parallel analysis;
- mode-based approach: AIC, BIC, ...;
- Likelihood Ratio Test (LRT).

We will then discuss LRT specifically.

LRT framework

- Step 1: $H_0 : k = 0$ V.S. $H_1 : k > 0$, i.e., $H_0 : \Sigma = \Psi$ V.S. $H_1 : \Sigma \neq \Psi$.

  If we reject H_0,

- Step 2: $H_0 : k = 1$ V.S. $H_1 : k > 1$,
  i.e., $H_0 : \Sigma = \Psi + \Lambda\Lambda^T$ V.S. $H_1 : \Sigma \neq \Psi + \Lambda\Lambda^T$.

  If we reject H_0,

- Step 3: $H_0 : k = 2$ V.S. $H_1 : k > 2$.

  Stop when we fail to reject H_0.

  Test statistics:

$$LRT = \frac{\sup_{\Theta_0} L(\mu, \Sigma)}{\sup_{\Theta_1} L(\mu, \Sigma)}$$

  and

$$-2 \log LRT \xrightarrow{d} \chi^2_{df}$$

3.1.4 Estimate latent factors X

Suppose $\hat{\mu}, \hat{\lambda}, \hat{\Psi}$, we want to estimate the latent factors X.

Lemma If

$$\begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N(\begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix})$$

Then the conditional distribution is

$$X_1 | X_2 \sim N(\mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(X_2 - \mu_2), \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21})$$

In this case, $X|Y \sim N(E(X|Y), var(X|Y))$, where

$$E(X|Y) = 0 + \Lambda^T(\Psi + \Lambda\Lambda^T)^{-1}(Y - \mu) = (I + \Lambda^T\Psi^{-1}\Lambda)^{-1}\Lambda^T\Psi^{-1}(Y - \mu)$$

which is like Ridge regression.

Note

Sherman-Morrison identity:

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^TA^{-1}}{1 + v^TA^{-1}u}$$

Woodbury identity:

$$(A + UCV^T)^{-1} = A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1}$$

3.2 Comparison between Probabilistic PCA and Factor Analysis

3.2.1 Model

Probabilistic PCA The model is

$$Y = \Lambda X + W$$

where $W \sim N(0, \sigma^2 I)$, which indicates the same variance;

Factor Analysis The model is

$$Y = \Lambda X + W$$

where $W \sim N(0, \Psi)$, and Ψ is diagonal, which means we have different variance for each dimension.

3.2.2 Solution

Probabilistic PCA The MLE's of PCA can be obtained by eigen-decomposition on the sample covariance matrix or SVD on the centered data.

Factor Analysis The MLE's of Factor model are often calculated by EM algorithm.

3.2.3 Application

Probabilistic PCA Used more in dimension reduction, data visualization and reconstruction. The goal is to find the direction with the maximum variance.

Factor Analysis Used more for discovering latent factors for the observed data.

Note: For both PCA and FA, Λ can only be identified up to an orthonormal transformation, i.e., the solutions remains invariant to the rotation of Λ. Their solutions are not unique.

Chapter 4

EM Algorithm

4.1 Motivation and Intuition for EM Algorithm

4.1.1 Motivation

Now we have latent variables $X_1, \ldots, X_n$ and observed data $Y_1, \ldots, Y_n$. Suppose (Y_i, X_i) is sampled from $p(y, x|\theta)$ where θ is unknown.
The **GOAL** is to find the MLE of θ given only the observed data $Y_1, \ldots, Y_n$, which is to maximize the marginal log likelihood

$$\ell(\theta|y) = \sum_{i=1}^{n} \log p(y_i|x_i, \theta) = \sum_{i=1}^{n} \log \int_{\mathcal{X}_i} p(y_i, x_i|\theta) dx_i$$

$$= \sum_{i=1}^{n} \log \int_{x_i} p(x_i) p(y_i|x_i, \theta) dx_i$$

and we use iterative approach to solve this problem.

Example 4.1.1: Finite mixture model

- Let $f(y|\psi)$ be a distribution parameterized by ψ.Y is sampled from one of $f(y|\psi_k)$, $k = 1, \ldots, K$.
- $X \in \{1, \ldots, K\}$ indicates which distribution Y is sampled from. $X = k$ with probability π_k.

$$X \sim \text{Multinomial}(\pi_1, \ldots, \pi_K), \quad Y|X = k \sim f(y|\psi_k).$$

- Unknown parameters we want to estimate: $\theta = (\pi, \psi)$.
- Consider $(y_1, x_1), \ldots, (y_n, x_n)$ i.i.d. sampled from $p(y, x|\theta)$. We only observe $y_1, \ldots, y_n$.

Express the marginal log likelihood in terms of (π, ψ).

$$\ell(\theta|y) = \sum_{i=1}^{n} \log p(y_i|\theta)$$

$$= \sum_{i=1}^{n} \log \sum_{k=1}^{K} p(y_i, x_i = k|\theta)$$

$$= \sum_{i=1}^{n} \log \sum_{k=1}^{K} p(x_i = k|\theta) p(y_i|x_i = k, \theta)$$

$$= \sum_{i=1}^{n} \log(\sum_{k=1}^{K} \pi_k f(y_i|\psi_k))$$

4.1.2 Intuition

E-Step Since maximizing nonconvex $\ell(\theta|y)$ is hard computationally, we optimize an easy-to-work-with surrogate function $\mathcal{F}(\theta, q)$ instead of $\ell(\theta|y)$.
Here the surrogate function $\mathcal{F}(\theta, q)$ is the lower bound for $\ell(\theta|y)$:

$$\forall \theta, \mathcal{F}(\theta, q) \leq \ell(\theta|y), \forall q$$

$$\max_{\theta} \ell(\theta|y) = \max_{\theta} \max_{q} \mathcal{F}(\theta, q)$$

In t-th iteration, the function is based on $\theta^{(t-1)}$. The E-step can be viewed as solving

$$q^{(t)} = \arg\max_{q \in Q} \mathcal{F}(\theta^{(t-1)}, q)$$

which is solving the optimal q with θ fixed.

M-Step we maximize the surrogate function $\mathcal{F}(\theta, q)$ to update $\theta^{(t-1)}$ to $\theta^{(t)}$. This step can be written as

$$\theta^{(t)} = \arg\max_{\theta} \mathcal{F}(\theta, q^{(t)})$$

which is solving the optimal θ with q fixed.

Then the constraints on the surrogate function above become

$$\mathcal{F}(\theta^{(t-1)}, q) \leq \ell(y|\theta^{(t-1)})$$

$$\mathcal{F}(\theta^{(t-1)}, q^{(t)}) = \ell(y|\theta^{(t-1)})$$

Note EM Algorithm guarantees that $\ell(\theta^{(t)}|y)$ never decreases. Each iteration pushes up the lower bound.

Proof Sketch.

$$\begin{aligned}\ell(\theta^{(t-1)}|y) &= \mathcal{F}(\theta^{(t-1)}, q^{(t)}) \\ &\leq \mathcal{F}(\theta^{(t)}, q^{(t)}) \\ &= \mathcal{F}(\theta^{(t)}, q^{(t+1)}) \\ &= \ell(\theta^{(t)}|y)\end{aligned}$$

□

4.2 Interpretation for EM Algorithm

Next we prove that the marginal likelihood has a lower bound.

$$\ell(\theta|y) = \log p(y_1, \ldots, y_n|\theta)$$

$$= \log p(y|\theta) \int_{\mathcal{X}} q(x)dx \quad q(x) \text{ is a pdf}$$

$$= \int_{\mathcal{X}} \log p(y|\theta) q(x)dx$$

$$= \int_{\mathcal{X}} \log[\frac{q(x)}{q(x)} \cdot \frac{p(y,x|\theta)}{p(x|y,\theta)}]q(x)dx$$

$$= \int_{\mathcal{X}} \log[\frac{p(y,x|\theta)}{q(x)}]q(x)dx + \int_{\mathcal{X}} \log[\frac{q(x)}{p(x|y,\theta)}]q(x)dx$$

$$= \mathcal{F}(\theta, q) + D_{KL}(q\|p(\cdot|y,\theta))$$

KL Divergence

KL Divergence is represented by $D_{KL}(q\|p) = \int_{\mathcal{X}} \log(\frac{q(x)}{p(x)})q(x)dx$.
Next we will prove KL Divergence is positive:

$$D_{KL}(q\|p) = \int_{\mathcal{X}} \log(\frac{q(x)}{p(x)})q(x)dx$$

$$= E_{X\sim q(\cdot)} \log(\frac{q(x)}{p(x)}) = -E_{X\sim q(\cdot)} \log(\frac{p(x)}{q(x)})$$

$$= -\int_{\mathcal{X}} \log(\frac{p(x)}{q(x)})q(x)dx$$

$$\geq -\log \int_{\mathcal{X}} \frac{p(x)}{q(x)} \cdot q(x)dx \quad (Jensen's\ Inequality)$$

$$= -\log \int_{\mathcal{X}} p(x)dx = -\log(1) = 0$$

and "=" holds $\iff$ $p(x) = q(x)$.

Because KL Divergence is positive, the lower bound of log likelihood is $\mathcal{F}(\theta, q) = \int_{\mathcal{X}} \log[\frac{p(y,x|\theta)}{q(x)}]q(x)dx = E_{X\sim q(\cdot)} \log(\frac{p(x,y|\theta)}{q(x)})$, called **Evidence Lower Bound (ELBO)**.

Usually, taking max over $\mathcal{F}(\theta, q)$ is easier than $\ell(\theta|y)$, and we often calculate $E_{X\sim q(\cdot)} \log p(x, y|\theta)$ instead because $E_{X\sim q(\cdot)} \log q(x)$ is independent of θ.

References

- Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1), 1-22;
- Chapter 9, *PRML*.

4.3 EM Algorithm

E-Step In this step we need to maximize $\mathcal{F}(\theta^{(t-1)}, q)$:

- First we calculate the maximized surrogate function q: $q(x)^{(t)} = p(x|y, \theta^{(t-1)})$;

  Proof sketch. maximize $\ell(\theta|y) \iff D_{KL} = 0 \iff q(x)^{(t)} = p(x|y, \theta^{(t-1)})$. □

- Then we calculate $E_{X\sim q^{(t)}(\cdot)} \log p(x, y|\theta) = \int \log p(x, y|\theta) q^{(t)}(x) dx$.

M-Step In this step we need to maximize $\mathcal{F}(\theta, q^{(t)})$:
We need to calculate $\theta^{(t)} = \arg\max_{\theta} E_{X\sim q^{(t)}(\cdot)} \log p(x, y|\theta)$.

Note

EM Algorithm is a **Coordinate Ascent** algorithm. For maximizing the log likelihood:

E-Step: update $q^{(t+1)} = \arg\max_{q} \mathcal{F}(\theta^{(t)}, q)$;

M-Step: update $\theta^{(t+1)} = \arg\max_{\theta} \mathcal{F}(\theta, q^{(t+1)})$

Example 4.3.1: Bernoulli Mixture Model Consider a mixture model with K components. Each component k is parameterized by a Bernoulli distribution with success probabilities

$$\psi_k = (\psi_{k1}, \ldots, \psi_{kd})$$

That is, a random vector $\mathbf{Y} \in \{0, 1\}^d$ from component k has independent entries, and each entry Y_j follows a Bernoulli distribution with success probability ψ_{kj}. Mixing weight

$$\pi = (\pi_1, \ldots, \pi_K) \quad \text{where} \quad \sum_k \pi_k = 1$$

Each data point $y_i \in \{0, 1\}^d$, $i = 1, \ldots, n$ is generated by:

1. Selecting a component $x_i \sim$ Multinomial(π).
2. Sampling $y_i \sim$ Bernoulli(ψ_{x_i}).

Applications: black-white images with $d = 64 \times 64$ pixels of $K = 10$ digits.

Estimate $\theta = (\pi, \psi)$ using EM algorithm.

The full likelihood of (y_i, x_i), $i = 1, \ldots, n$ is

$$p(y_i, x_i|\theta) = p(y_i|x_i, \theta)p(x_i|\theta)$$

$$= \prod_{j=1}^{d} \psi_{x_i,j}^{y_{ij}}(1 - \psi_{x_i,j})^{1-y_{ij}} \cdot \pi_{x_i}$$

E-Step: Recall $q_i = p(\cdot|y_i, \theta)$,

$$q_i(k) = p(x_i = k|y_i, \theta)$$

$$= \frac{p(x_i = k, y_i|\theta)}{p(y_i|\theta)} = \frac{p(x_i = k, y_i|\theta)}{\sum_{k'=1}^{K} p(x_i = k', y_i|\theta)}$$

$$= \frac{\prod_{j=1}^{d} \psi_{kj}^{y_{ij}}(1 - \psi_{kj})^{1-y_{ij}} \cdot \pi_k}{\sum_{k'=1}^{K} \prod_{j=1}^{d} \psi_{k'j}^{y_{ij}}(1 - \psi_{k'j})^{1-y_{ij}} \cdot \pi_{k'}}$$

Recall for optimizing $\mathcal{F}(\theta, q)$, we only need to optimize $E_{X\sim q(\cdot)} \log p(x, y|\theta)$, which is

$$\sum_{i=1}^{n} E_{X\sim q_i} \log p(x, y_i|\theta)$$

$$= \sum_{i=1}^{n} E_{X\sim q_i}\{\log \pi_X + \sum_{j=1}^{d}[y_{ij} \log \psi_{x,j} + (1 - y_{ij}) \log(1 - \psi_{x,j})]\}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} q_i(k)\{\log \pi_k + \sum_{j=1}^{d}[y_{ij} \log \psi_{kj} + (1 - y_{ij}) \log(1 - \psi_{kj})]\} \quad (\star)$$

M-Step:

- π: Note that $\sum_k \pi_k = 1$, so we use Lagrangian method:

$$L = (\star) + \lambda(1 - \sum_{k=1}^{K} \pi_k)$$

Take derivative:

$$\frac{\partial L}{\partial \pi_k} = \frac{1}{\pi_k} \sum_{i=1}^{n} q_i(k) - \lambda = 0 \quad (\star\star)$$

So

$$\pi_k \lambda = \sum_{i=1}^{n} q_i(k), \quad k = 1, \ldots, K$$

Sum over k:

$$\lambda = \sum_{i=1}^{n} \sum_{k=1}^{K} q_i(k) = \sum_{i=1}^{n} 1 = n$$

because $q_i(k)$ is a pmf. Plug into $(\star\star)$:

$$\pi_k = \frac{1}{n} \sum_{i=1}^{n} q_i(k)$$

- ψ_{kj}:

$$\frac{\partial L}{\partial \psi_{kj}} = \sum_{i=1}^{n} [q_i(k) \frac{y_{ij}}{\psi_{kj}} - q_i(k) \frac{1 - y_{ij}}{1 - \psi_{kj}}] = 0$$

So

$$(1 - \psi_{kj}) \sum_{i=1}^{n} q_i(k) y_{ij} = \psi_{kj} \sum_{i=1}^{n} q_i(k)(1 - y_{ij})$$

Therefore

$$\sum_{i=1}^{n} q_i(k) y_{ij} = \psi_{kj} \sum_{i=1}^{n} q_i(k)$$

So

$$\psi_{kj} = \frac{\sum_{i=1}^{n} q_i(k) y_{ij}}{\sum_{i=1}^{n} q_i(k)}$$

In summary, in t-th E-Step, we update

$$q_i(k)^{(t)} = \frac{\prod_{j=1}^{d} \psi_{kj}^{(t)\ y_{ij}} (1 - \psi_{kj}^{(t)})^{1 - y_{ij}} \cdot \pi_k^{(t)}}{\sum_{k'=1}^{K} \prod_{j=1}^{d} \psi_{k'j}^{(t)\ y_{ij}} (1 - \psi_{k'j}^{(t)})^{1 - y_{ij}} \cdot \pi_{k'}^{(t)}}$$

in t-th M-Step, we update

$$\pi_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} q_i^{(t)}(k)$$

and

$$\psi_{kj}^{(t+1)} = \frac{\sum_{i=1}^{n} q_i^{(t)}(k) y_{ij}}{\sum_{i=1}^{n} q_i^{(t)}(k)}$$

4.4 EM Algorithm and Factor Analysis

As mentioned in the last Chapter, EM Algorithm can help solve the numerical solutions for Factor models.
We want to estimate μ, Λ, Ψ from

$$\begin{pmatrix} X \\ Y \end{pmatrix} \sim N(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} I & \Lambda \\ \Lambda^T & \Lambda\Lambda^T + \Psi \end{pmatrix})$$

Let the true value $\mu = 0$ for simplicity. We have

$$x_i|y_i \sim (E(x_i|y_i), var(x_i|y_i))$$

where

$$E(x_i|y_i) = (I + \Lambda^T\Psi^{-1}\Lambda)^{-1}\Lambda^T\Psi^{-1}y_i$$
$$cov(x_i|y_i) = (I + \Lambda^T\Psi^{-1}\Lambda)^{-1}$$

E-Step

$$E_{x_i|y_i,\theta^{(t)}}\ell(\theta|x,y) = E\left[\sum_i \log p(x_i, y_i|\theta)\right]$$

$$= E\left[\sum_i \log p(y_i|x_i,\theta)p(x_i|\theta)\right] \quad \text{omit the constant}$$

$$= E\left[\sum_i \log p(x_i|\theta) - \frac{n}{2}\log|\Psi| - \frac{1}{2}\sum_{i=1}^n tr\left((y_i - \Lambda x_i)^T\Psi^{-1}(y_i - \Lambda x_i)\right)\right] + c$$

$$= E_{x_i|y_i,\theta^{(t)}}\left[-\frac{n}{2}\log|\Psi| - \frac{n}{2}tr\left(\Psi^{-1}S\right)\right] + c$$

$$= -\frac{n}{2}\log|\Psi| - \frac{n}{2}tr\left(\Psi^{-1}E(S|Y,\theta^{(t)})\right) + c.$$

which is the surrogate function $Q(\theta)$ where

$$S = \frac{1}{n}\sum_{i=1}^n (y_i - \Lambda x_i)(y_i - \Lambda x_i)^T$$

$$E_{X|Y}(S) = E(S|Y,\theta^{(t)}) = \frac{1}{n}\sum_{i=1}^n \left[y_i y_i^T - 2y_i\Lambda E(x_i|y_i,\theta^{(t)}) + \Lambda E(x_i x_i^T|y_i,\theta^{(t)})\Lambda^T\right]$$

in which $E(x_i|y_i,\theta^{(t)})$ is given above before the E-Step and $E(x_i x_i^T|y_i,\theta^{(t)})$ can be calculated by $cov(x_i|y_i,\theta^{(t)}) + E(x_i|y_i,\theta^{(t)})E(x_i|y_i,\theta^{(t)})^T$.

M-Step: Calculate $\theta^{(t+1)}$

Λ:

$$\frac{dQ(\theta|\theta^{(t)})}{d\Lambda} = -\frac{n}{2}\frac{d}{d\Lambda}\left[tr\left(\Psi^{-1}E(S|Y,\theta^{(t)})\right)\right]$$

Note: $Q(\theta|\theta^{(t)}) = E_{X\sim q^{(t)}(\cdot)} \log p(x,y|\theta^{(t)}) = E_{X\sim p(x|y,\theta^{(t)})} \log p(x,y|\theta^{(t)})$.

$$= -\frac{n}{2} tr\left(\Psi^{-1}\frac{d}{d\Lambda}E(S|Y,\theta^{(t)})\right).$$

$$= -\frac{n}{2} tr\left(\Psi^{-1}[-\frac{2}{n}\frac{d}{d\Lambda}(\sum_i y_i \Lambda E(x_i|y_i,\theta^{(t)}) + \sum_i \Lambda E(x_i x_i^T|y_i,\theta^{(t)})\Lambda^T)]\right)$$

Note: although $E(x_i|y_i,\theta^{(t)})$ and $E(x_i x_i^T|y_i,\theta^{(t)})$ depend on Λ too, they are in the previous iteration and will not affect $\Lambda^{(t+1)}$.

$$= tr\left(\Psi^{-1}\sum_i [y_i E(x_i|y_i,\theta^{(t)})^T - \Lambda E(x_i x_i^T|y_i,\theta^{(t)})]\right) = 0$$

Therefore

$$\Lambda^{(t+1)} = \left(\sum_i y_i E(x_i|y_i,\theta^{(t)})^T\right)\left(\sum_i E(x_i x_i^T|y_i,\theta^{(t)})\right)^{-1}$$

Ψ:

$$\frac{d\mathcal{Q}(\theta)}{d\Psi^{-1}} = \frac{n}{2}\Psi - \frac{n}{2}E(S|y,\theta^{(t)}) = 0$$

Therefore

$$\Psi^{(t+1)} = E(S|y,\theta^{(t)})$$

Use $\Lambda^{(t+1)}$ inside E!

Chapter 5

Multidimensional Scaling (MDS)

5.1 MDS

We have data $Y_1, \ldots, Y_n \in \mathbb{R}^q$. Let matrix $D = \{d_{ij}\}_{n\times n}$ be the distance between Y_i and Y_j $(i, j = 1, \ldots, n)$.
For example, Euclidean distance $d_{ij} = \|y_i - y_j\|_2$.

Main idea optimization approach for dimension reduction.

Note:

- No randomness assumption on $y_1, \ldots, y_n$;
- Easily extendable to **nonlinear** manifold setting.

GOAL To find $X_1, \ldots, X_n \in \mathbb{R}^k$ $(k \ll q)$, which means to find a lower-dimensional representation of the data that preserves the pairwise distances as well as possible..

Method Minimize the **stress function** $S(X) = \sum_{1\le i,j\le n}(d_{ij} - \|X_i - X_j\|_2)^2$.

Remark

- No closed form solution is available;
- The optimization problem is not convex;
- we can use some optimization tools to perform local optimization (e.g., gradient descent algorithm);
- The stress function $S(X)$ can be easily modified to handle different situations by changing the scale of d_{ij} or $S(X)$:

1. Assign weights so that small distances are preserved:

$$S(X) = \sum_{1 \le i,j \le n} \frac{1}{d_{ij}} (d_{ij} - \|X_i - X_j\|_2)^2$$

2. Assign weights if large distances matter more:

$$S(X) = \sum_{1 \le i,j \le n} d_{ij} (d_{ij} - \|X_i - X_j\|_2)^2$$

Note

1. We actually don't need individual data points $y_1, \ldots, y_n$. All we need is d_{ij} which is the sufficient statistics for MDS;
2. If we want MDS to capture nonlinear relationships, we can replace d_{ij} by kernel functions, e.g., sigmoid kernels.

References

Chapter 14 section 8 & 9, *ESL*.

5.2 Classical MDS

5.2.1 Model and solutions

Instead of presenting the distance, we work with the inner product.
Let $K = (k_{ij})_{n\times n} = (y_i^T y_j)_{n\times n}$.

GOAL To find $X_1, \ldots, X_n \in \mathbb{R}^k$ *s.t.* $\min_X S(X) = \sum_{1\le i,j\le n}(y_i^T y_j - x_i^T x_j)^2$

where $X_{n\times k} = \begin{pmatrix} X_1^T \\ \vdots \\ X_n^T \end{pmatrix}$ and $Y_{n\times q} = \begin{pmatrix} Y_1^T \\ \vdots \\ Y_n^T \end{pmatrix}$.

The optimization problem is equivalent to $\min_X \|YY^T_{n\times n} - XX^T_{n\times n}\|_F^2$. Let $G = YY^T$ with $rank(G) = q$, the problem reduces to finding the low-rank approximation to $G = YY^T$ since XX^T has at most rank $k \ll q$:

$$\min_X \|G_{n\times n} - XX^T_{n\times n}\|_F^2$$

Eckart-Young Theorem

The best rank-k approximation of $G = \bar{Y}\bar{Y}^T$ is:

$$G' = \sum_{j=1}^{k} \lambda_j u_j u_j^T$$

where $\bar{Y}$ means **centered data** Y; $\lambda_1 \ge \cdots \ge \lambda_k$ are the top-k eigenvalues of G and u_j's are the corresponding eigenvectors.

$$G' = [\sqrt{\lambda_1}u_1, \ldots, \sqrt{\lambda_k}u_k][\sqrt{\lambda_1}u_1, \ldots, \sqrt{\lambda_k}u_k]^T$$

where $X = [\sqrt{\lambda_1}u_1, \ldots, \sqrt{\lambda_k}u_k]$.

So we should perform eigen-decomposition on G and get the top-k eigenvalues and eigenvectors. Then $X = [\sqrt{\lambda_1}u_1, \ldots, \sqrt{\lambda_k}u_k]$; or we can use SVD on **centered** $\bar{Y}$.

5.2.2 Convert distances into Inner products

As we mentioned, in practice we work with distances more than the raw data Y. So one thing that matters is how to convert distances into inner products and perform Classical MDS.

Let $D = (d_{ij})_{n\times n}$, $\|y_i - y_j\|_2^2 = \|y_i\|_2^2 - 2y_i^T y_j + \|y_j\|_2^2$. We need to perform double recentering of the distance matrix D.

1. Rencenter the row (let the sum of row to be 0):
   $D \to \widetilde{D}$ where $\tilde{d}_{ij} = d_{ij} - \frac{1}{n}\sum_{i=1}^{n} d_{ij}$;

2. Rencenter the column (let the sum of column to be 0):
   $\widetilde{D} \to \widetilde{\widetilde{D}}$ where $\tilde{\tilde{d}}_{ij} = \tilde{d}_{ij} - \frac{1}{n}\sum_{j=1}^{n} \tilde{d}_{ij}$.

Use matrix notion:

$$-\frac{1}{2}(I_n - J)D^2(I_n - J) = \bar{Y}\bar{Y}^T = G$$

where $J = \frac{1}{n}\mathbf{1}\mathbf{1}^T$.

After transformation, we can apply Classical MDS onto $G = \bar{Y}\bar{Y}^T$.

5.2.3 Classical MDS & PCA (assume $\bar{Y}$ centered)

1. **Classical MDS**: perform eigen-decomposition on $G_{n\times n} = \bar{Y}\bar{Y}^T$. The computational complexity is $O(qn^2)$;
   PCA: perform eigen-decomposition on $C_{q\times q} = \bar{Y}^T\bar{Y}$ (sample covariance matrix is $\frac{1}{n-1}\bar{Y}^T\bar{Y}$). The computational complexity is $O(nq^2)$;

2. Both Classical MDS and PCA are related to SVD. Let $\bar{Y}_{n\times q} = U_{n\times q}D_{q\times q}V^T_{q\times q}$, where D is positive definite, $U^TU = I$, $V^TV = I$:

   Columns of U are eigenvectors of $G = \bar{Y}\bar{Y}^T = UD^2U^T$;
   Columns of V are eigenvectors of $C = \bar{Y}^T\bar{Y} = VD^2V^T$;

3. Classical MDS and PCA gives the same linear projection on lower dimension space:

   Classical MDS: solves $G = \bar{Y}\bar{Y}^T = UD^2U^T \Rightarrow$ gives $X = UD$ where $D_{k\times k} = diag(\sqrt{\lambda_1}, \ldots, \sqrt{\lambda_k})$and $XX^T = G$;
   PCA: solves $C = \bar{Y}^T\bar{Y} = VD^2V^T \Rightarrow$ gives $X = \bar{Y}V = UDV^TV = UD$.

Chapter 6

Kernel PCA

6.1 Kernel PCA

6.1.1 Motivation

How can we deal with this type of data? What if we perform PCA on the data above?

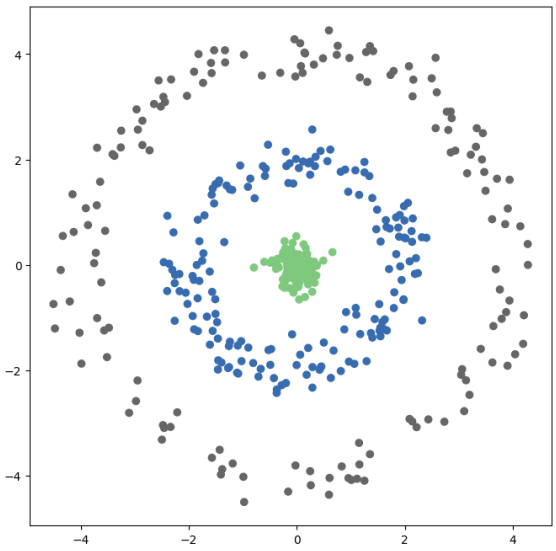


Figure 6.1

PCA finds the principal linear directions of the data, i.e., directions that linearly separate the data. For this concentric data, where there are no dominant directions, PCA fails to separate the classes.

Therefore, we project the data onto a higher dimensional space in which the data is sparse and easily separated, then we work on the projection.

6.1.2 PCA with basis expansion

Example 6.1.1 For data point $y = (y_1, y_2) \in \mathbb{R}^2$, use the mapping

$$\phi(y) = (y_1, y_2, y_1^2, y_2^2) \in \mathbb{R}^4$$

The idea is to induce the linear method PCA to pick up the representation $y_1^2 + y_2^2$, which is the square of the distance from the origin.

Run PCA on the data expanded with ϕ given above and plot the first two principal components.

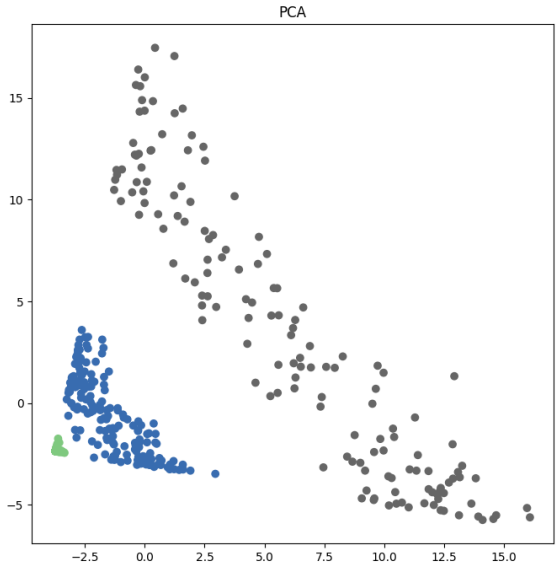


Figure 6.2: Result for PCA with basis expansion

Idea Map the data $y = (y_1, y_2) \in \mathbb{R}^q$ to a more complicated space via a nonlinear mapping $\phi : \mathbb{R}^q \to \mathbb{R}^M$.
Let $\Phi \in \mathbb{R}^{n \times M}$ be expanded data, we then apply regular PCA to reduce the dimensionality to get a good representation of the data, which computes the SVD of the covariance $\frac{1}{n}\Phi^T\Phi \in \mathbb{R}^{M \times M}$.
$\phi(y_i)$ is a M-dim vector.

Note: this is computationally expensive for large M.

Suppose that you have an "oracle" that computes $K(y, y') = \langle \phi(y), \phi(y') \rangle$ for any $y, y' \in \mathbb{R}^q$ **very quickly**. How to run PCA on expanded data Φ with $M \gg n$ efficiently using the oracle?

Regular PCA on Φ needs top eigenvectors of $\Phi^T\Phi_{M \times M}$:

$$\Phi^T\Phi v = \lambda v$$

The score is $\alpha = \Phi v \in \mathbb{R}^n$. So

$$\Phi^T\Phi(\Phi^T\Phi)v = \lambda\Phi^T\Phi v$$

$$\Phi^T \Phi \Phi^T \alpha = \lambda \Phi^T \alpha$$

$$\Phi^T (\Phi \Phi^T \alpha - \lambda \alpha) = 0$$

Let $K_{n\times n} = \Phi\Phi^T$, we have

$$\Phi^T (K\alpha - \lambda\alpha) = 0$$

Next we will prove that this is equivalent to $K\alpha - \lambda\alpha = 0$:
Because

$$K\alpha = \Phi\Phi^T\alpha, \quad \lambda\alpha = \lambda\Phi v$$

We know that $K\alpha - \lambda\alpha \in col(\Phi)$;
From the equation above ($\Phi^T(\Phi\Phi^T\alpha - \lambda\alpha) = 0$), we know that $K\alpha - \lambda\alpha \in col(\Phi)^\perp$.
Therefore, $K\alpha - \lambda\alpha = 0$.

6.1.3 Kernel functions

For performing PCA with basis expansion $\phi : \mathbb{R}^q \to \mathbb{R}^M$, we only need access to the Gram matrix:

$$K = \Phi\Phi^T = \begin{pmatrix} K(y_1, y_1) & K(y_1, y_2) & \cdots & K(y_1, y_n) \\ K(y_2, y_1) & K(y_2, y_2) & \cdots & K(y_2, y_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(y_n, y_1) & K(y_n, y_2) & \cdots & K(y_n, y_n) \end{pmatrix}$$

where $K(y, y') = \langle \phi(y), \phi(y') \rangle$ and $\langle \cdot, \cdot \rangle$ is inner product.

Instead of specifying the basis expansion map ϕ, Kernel PCA specifies which kernel K to use.

Kernel function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, generally $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$ where $\mathcal{X}$ is the sample space. In Kernel PCA we need to construct K that is **symmetric** and **p.s.d.** (positive semidefinite).

p.s.d. Kernel

A function $K : \mathbb{R}^q \times \mathbb{R}^q \to \mathbb{R}$ is a p.s.d. kernel function if $\forall n \in \mathbb{N}$, $X_1, \ldots, X_n \in \mathcal{X}$ and $a_1, \ldots, a_n \in \mathbb{R}$, the matrix K obtained by $K_{ij} = K(X_i, X_j)$ is symmetric and $a^T K a \geq 0$.

Mercer's Theorem For any p.s.d. kernel K, there exists a feature map Φ which induces a Gram matrix given by K. In particular, K introduces a space of functions generated by the linear space of $\{K(\cdot, y), \forall y \in \mathbb{R}^q\} = \mathcal{H}_K$ with the inner product $\langle K(\cdot, y), K(x, \cdot) \rangle = K(x, y)$, where $\mathcal{H}_K$ represents a RKHS which will be introduced in the next section.

To simplify, for any p.s.d. K, there exists a map

$$\phi : \mathbb{R}^q \to \mathcal{H}_K$$

$$y \to \phi(y)$$

where $K(x, y) = \langle \phi(x), \phi(y) \rangle$.

examples:

1. **Linear PCA**:
   $x, y \in \mathbb{R}^q$, we have
   $$K(x, y) = x^T y, \quad \phi(x) = x, \quad \mathcal{H}_K = \mathbb{R}^q$$

2. **Polynomial kernel**:
   $x, y \in \mathbb{R}^q$, we have
   $$K(x, y) = (1 + x^T y)^d$$
   Particularly, if $q = 2$, $x, y \in \mathbb{R}^2$, then
   $$K(x, y) = (1+x^T y)^2 = 1+2x_1y_1+2x_2y_2+x_1^2y_1^2+x_2^2y_2^2+2x_1x_2y_1y_2 = \phi(x)\phi(y)$$
   where $\phi(x) = (1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2)^T \in \mathbb{R}^6$, $\phi : \mathbb{R}^2 \to \mathbb{R}^6$ and $\phi(x) \in \mathcal{K} = \mathbb{R}^6$.
   However, Φ does not often have closed forms.

3. **Gaussian kernel (a.k.a. RBF kernel)**:
   $x, y \in \mathbb{R}^q$, we have
   $$K(x, y) = \exp\{-\frac{1}{\sigma^2}\|x - y\|_2^2\}$$
   where σ^2 is a user-specified scaling parameter.
   Here $\phi(x)$ is infinite-dimension and has no closed-form expression. One can use the Taylor expansion to prove this fact.

   Proof sketch.
   $$K(x, y) = \exp\{-\frac{1}{\sigma^2}\|x - y\|_2^2\}$$
   $$= \exp\{-\frac{1}{\sigma^2}\|x\|_2^2\} \exp\{-\frac{1}{\sigma^2}\|y\|_2^2\} \exp\{-\frac{2}{\sigma^2}x^T y\}$$
   $$= c(x)c(y) \sum_{k=0}^{\infty} \frac{(x^T y)^k}{k!} \ (Taylor\ Expansion)$$
   $$= \sum_{k=0}^{\infty} c \cdot \langle \frac{1}{\sqrt{k!}} c(x) x^k, \frac{1}{\sqrt{k!}} c(y) y^k \rangle$$

   □

6.1.4 Kernel PCA method

Kernel PCA can be viewed as doing PCA with possibly infinite-dimensional feature mapping ϕ. There are two methods for Kernel PCA.

1. **Method 1**:
   If we have successfully calculated Φ, then we can just calculate the principal components of Φ.

2. **Method 2**:
   More often, as we mentioned before, Φ does not have closed forms, e.g., it may have infinite dimensions. Then we work on the Gram matrix $K = (\langle \phi(x_i), \phi(x_j) \rangle)_{n \times n}$.

References

Chapter 12, Section 3, *PRML*.

Consider the case where Φ has been centered, i.e., $\sum_{i=1}^n \phi(y_i) = 0$ and $\mathcal{H}_k$ is m-dim ($m \gg q$).

Let $C = \frac{1}{n} \sum_{i=1}^n \phi(y_i)\phi(y_i)^T$, then $Cv_j = \lambda_j v_j$, $j = 1, \ldots, m$ where λ_j and v_j are the eigenvalues and corresponding eigenvectors.

GOAL Solve the problem without exploring $\mathcal{H}_K$ directly because Φ can be difficult to compute.

From the equations above, we have

$$Cv_j = \frac{1}{n} \sum_{i=1}^n \phi(y_i)\phi(y_i)^T v_j = \frac{1}{n} \sum_{i=1}^n \phi(y_i) \langle \phi(y_i), v_j \rangle = \lambda_j v_j$$

In other words, v_j can be given by the linear combinations of $\phi(y_i)$:

$$v_j = \sum_{i=1}^n a_{ji} \phi(y_i) = \Phi^T a_j$$

where a_{ji} are some nonzero numbers.
Matrix notation:

$$a_j = \begin{pmatrix} a_{j1} \\ \vdots \\ a_{jn} \end{pmatrix} \text{ and } \Phi_{n \times m} = \begin{pmatrix} \phi(y_i)^T \\ \vdots \\ \phi(y_n)^T \end{pmatrix}.$$

Note: a_j is a n-dim vector, and v_j is a M-dim vector.

Thus, $C_{M \times M} = \frac{1}{n} \Phi^T \Phi$, so $\frac{1}{n} \Phi \Phi^T \Phi v_j = \lambda \Phi v_j$. Because $v_j = \Phi^T a_j$, the equation becomes

$$\frac{1}{n} \Phi \Phi^T \Phi \Phi^T a_j = \lambda \Phi \Phi^T a_j$$

$$\frac{1}{n}K_nK_na_j = \lambda K_na_j$$

So we do not need to calculate Φ in fact. Instead, we only need K.

Similarly to 6.1.2, we have

$$\frac{1}{n}K_na_j = \lambda a_j$$

and the solution differs from the above equation only when a_j are the eigenvectors of $\frac{1}{n}K_n$ corresponding to $\lambda_j = 0$.

Then we can perform eigen-decomposition on $\frac{1}{n}K_n$. a_j are the eigenvectors. For $C = \frac{1}{n}\Phi^T\Phi$, its eigenvectors $v_j = \frac{1}{\sqrt{n\lambda_j}}\Phi^T a_j$ with $\|v_j\| = 1$.

Note: $\sqrt{n\lambda_j} = \|v_j\| = \sqrt{v_j^T v_j}$ is used for standardization.

6.1.5 Summary of Kernel PCA

For data $y_1, \ldots, y_n \in \mathbb{R}^q$, we have a kernel K mapping it to $\phi(y_1), \ldots, \phi(y_n) \in \mathcal{H}_K$ where K is symmetric and p.s.d.

To compute the principal components of Φ, let $C = \frac{1}{n}\sum_{i=1}^n \phi(y_i)\phi(y_i)^T$, $Cv_j = \lambda v_j$, $j = 1, \ldots, M$. Because closed form expressions for Φ are often unavailable, we work on the Gram matrix $K_{n\times n}$.

We perform eigen-decomposition on $\frac{1}{n}K$ and get eigenvalues λ_j and eigenvectors a_j. Then we choose the top p a_j's and return a $n \times p$ matrix $A_p = [a_1, \ldots, a_p] \in \mathbb{R}^{n\times p}$.

The original principal components are given by $v_j = \frac{1}{\sqrt{n\lambda_j}}\Phi^T a_j$.

However, we could not compute v_j because we do not work with $\phi(\cdot)$, but we can compute the projection of data onto the lower-dimension principal components' directions:

For data point y_i, its projection on v_j is given by

$$x_{ij} = \langle \phi(y_i), v_j\rangle = \langle \phi(y_i), \frac{1}{\sqrt{n\lambda_j}}\Phi^T a_j\rangle$$

$$= \frac{1}{\sqrt{n\lambda_j}}\phi(y_i)^T\Phi^T a_j$$

$$= \frac{1}{\sqrt{n\lambda_j}}K_i a_j$$

where K_i is the i-th **row** of K.

The projection of all data on v_j is

$$x_j = \frac{1}{\sqrt{n\lambda_j}}Ka_j$$

Once we find a_j, we can compute j-th scores for any given new data $y \in \mathbb{R}^q$ using the equation above:

$$\langle \phi(y), v_j\rangle = \langle \phi(y), \Phi^T a_j\rangle$$

$$= \phi^T(y)\Phi^T a_j = \langle \Phi\phi(y), a_j \rangle$$
$$= \langle \begin{pmatrix} \phi(y_1)^T\phi(y) \\ \vdots \\ \phi(y_n)^T\phi(y) \end{pmatrix}, a_j \rangle$$
$$= \langle \begin{pmatrix} K(y_1, y) \\ \vdots \\ K(y_n, y) \end{pmatrix}, a_j \rangle$$

where $y_1, \ldots, y_n$ are previous data and Φ is calculated by them.

Note

One must **center** the data before Kernel PCA!
We set $\tilde{\phi}(y_i) = \phi(y_i) - \frac{1}{n}\sum_{i=1}^n \phi(y_i)$, $i = 1, \ldots, n$. Then we have

$$\widetilde{\Phi}_{n\times M} = (I - J)\Phi$$

where $J_{n\times n} = \frac{1}{n}\mathbf{1}\mathbf{1}^T = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix}$.
In practice, Φ is not often available, so we center K instead: Then

$$\widetilde{K}_{n\times n} = \widetilde{\Phi}\widetilde{\Phi}^T = (I - J)K(I - J)$$

We should run this step before Kernel PCA each time!

6.1.6 Another view of Kernel PCA

The first PC solves

$$\max_{v_1 \in \mathcal{H}_K} \frac{1}{n}\sum_{i=1}^n (v_1(y_i) - \bar{v})^2, \quad s.t.\ \|v_1\|_{\mathcal{H}_K} = 1$$

where $v_1(y_i) = \phi(y_i)^T v_1 = x_{i1}$, additionally $\bar{v} = \frac{1}{n}\sum_{i=1}^n v_1(y_i) = \frac{1}{n}\sum_{i=1}^n \phi(y_i)^T v_1 = 0$ because $\phi(y)$ has been centered before.
So the optimization means to find the first principal direction on which the variance of projection of the data is maximized.

6.2 Related Topics (I): Kernel Ridge Regression (KRR)

6.2.1 Review: Ridge Regression

The model is

$$Y = \theta^T X + W$$

where the mean function is $E(Y|X) = \theta^T X$, $y \in \mathbb{R}$, $X \in \mathbb{R}^p$, W is noise.

The estimate of θ is

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \sum_{i=1}^{n} (y_i - \theta^T X_i)^2 + \lambda \|\theta\|_2^2$$

$$= \arg\min_{\theta \in \mathbb{R}^p} (Y - X\theta)^T (Y - X\theta) + \lambda \theta^T \theta$$

$$= (X^T X + \lambda I_p)^{-1} X^T Y$$

By Woodbury identity, we have

$$(X^T X + \lambda I_p)^{-1} X^T Y = X^T (X X^T + \lambda I_n)^{-1} Y$$

Given a new data x, the prediction of response

$$\hat{Y} = \hat{\theta}^T x = Y^T (X X^T + \lambda I_n)^{-1} X x$$

where XX^T is the Gram matrix. This suggests that the estimator of Y depends on the Gram matrix. So we replace it with kernel.

6.2.2 Reproducing Kernel Hilbert Space (RKHS)

Let $\mathcal{H}$ be a Hilbert space of $\mathbb{R}$-valued functions on domain $\mathcal{X}$. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a reproducing kernel of $\mathcal{H}$, and $\mathcal{H}$ is a **reproducing kernel Hilbert space** if

- $\forall x \in \mathcal{X}, k(\cdot, x) \in \mathcal{H}$;
- $\forall x \in \mathcal{X}, \forall f \in \mathcal{H}$, $\langle f(\cdot), k(\cdot, x) \rangle_{\mathcal{H}} = f(x)$.

Review: Hilbert Space

Hilbert Space is a complete vector space equipped with an inner product $\langle \cdot, \cdot \rangle$ that follows:
Let $\mathcal{H}$ be a vector space,

1. **Inner product structure**:
   For any vectors $f, g, h \in \mathcal{H}$ and scalars α, β, the inner product satisfies
   - linearity: $\langle \alpha f + \beta g, h \rangle = \alpha \langle f, h \rangle + \beta \langle g, h \rangle$;
   - symmetry: $\langle f, g \rangle = \langle g, f \rangle$;
   - positivity: $\langle f, f \rangle \geq 0, \quad \langle f, f \rangle = 0 \iff f = 0.$
2. **completeness**:
   Every Cauchy sequence in $\mathcal{H}$ converges to an element within $\mathcal{H}$.

Note: Hilbert space is more than a finite-dimension vector space. It can be an infinite-dimension function space.
e.g., the L^2 space:

$$L^2(\mathbb{R}) = \left\{ f : \mathbb{R} \to \mathbb{R} \mid \int_{-\infty}^{\infty} |f(x)|^2 dx < \infty \right\}$$

where $\langle f, g \rangle = \int_{-\infty}^{\infty} f(x) g(x) dx$.

For RKHS, the elements in a RKHS are functions induced by kernel $k(\cdot, \cdot)$. They can be represented by the linear combination of kernel functions.

In particular, for any $x, y \in \mathcal{X}$,

$$k(x, y) = \langle k(\cdot, x), k(\cdot, y) \rangle_{\mathcal{H}}$$

6.2.3 Representer Theorem

Let function $f \in \mathcal{H}_K$ with kernel K. Consider

$$\hat{f} = \arg \min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} L(y_i, f(X_i)) + \lambda \|f\|_{\mathcal{H}_K}^2$$

where L is a loss function and $\| \cdot \|_{\mathcal{H}_K}$ is a norm induced by RKHS. The solution to this optimization problem may have infinite dimensions.

However, according to the Representer theorem, the solution to this problem must have a finite-dimension form:

$$\hat{f}(x) = \sum_{i=1}^{n} \alpha_i K(x, X_i)$$

and

$$\|f\|_{\mathcal{H}_K}^2 = \alpha^T K \alpha$$

where $x, X_i \in \mathbb{R}^p$, $\alpha_i \in \mathbb{R}$.

This allows the original problem to be reduced to be a finite-dimension problem:

$$\hat{\alpha} = \arg\min_{\alpha \in \mathbb{R}^n} \sum_{i=1}^n L(y_i, \sum_{j=1}^n \alpha_j K(X_j, X_i)) + \lambda \alpha^T K \alpha$$

Proof. For a function $f \in \mathcal{H}_K$, consider the decomposition

$$f = f_S + f_\perp$$

where f_S is the projection onto the subspace

$$\{f : f(x) = \sum_{i=1}^n \alpha_i K(x, X_i), \quad \alpha_1, \ldots, \alpha_n \in \mathbb{R}\}$$

Then, $\langle f_S, f_\perp \rangle_{\mathcal{H}_K} = 0$ and it follows that

$$\|f\|_{\mathcal{H}_K}^2 = \|f_S\|_{\mathcal{H}_K}^2 + \|f_\perp\|_{\mathcal{H}_K}^2 \geq \|f_S\|_{\mathcal{H}_K}^2$$

Additionally,

$$f(X_i) = \langle f, K(\cdot, X_i) \rangle_{\mathcal{H}_K} = \langle f_S + f_\perp, K(\cdot, X_i) \rangle_{\mathcal{H}_K}$$

$$= \langle f_S, K(\cdot, X_i) \rangle_{\mathcal{H}_K} = f_S(X_i)$$

So

$$L(y_i, f(X_i)) = L(y_i, f_S(X_i))$$

It follows that for any f,

$$\sum_{i=1}^n L(y_i, f(X_i)) + \lambda \|f\|_{\mathcal{H}_K}^2 \geq \sum_{i=1}^n L(y_i, f_S(X_i)) + \lambda \|f_S\|_{\mathcal{H}_K}^2$$

So an optimal solution f^* is in the subspace, which completes the proof. □

6.2.4 Kernel Ridge Regression (KRR)

For a regression model $y = f(X) + W$, where $f \in \mathcal{H}_K$ equipped with kernel K. Kernel Ridge Regression aims to solve

$$\hat{f} = \arg\min_{f \in \mathcal{H}_K} \sum_{i=1}^n (y_i - f(X_i))^2 + \lambda \|f\|_{\mathcal{H}_K}^2$$

By representer theorem, we can restrict the search space to functions of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, X_i)$$

What is the solution for α?

The problem is

$$\arg\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} (y_i - \langle f, K(\cdot, X_i)\rangle_{\mathcal{H}_K})^2 + \lambda \|f\|_{\mathcal{H}_K}^2 \quad s.t.\ f(x) = \sum_{i=1}^{n} \alpha_i K(x, X_i)$$

where we can prove that

$$\langle f, K(\cdot, X_i)\rangle_{\mathcal{H}_K} = \sum_{j=1}^{n} \alpha_j \langle K(\cdot, X_j), K(\cdot, X_i)\rangle_{\mathcal{H}_K} = \sum_{j=1}^{n} \alpha_j K(X_j, X_i) = f(X_i)$$

and

$$\|f\|_{\mathcal{H}_K}^2 = \langle f, f\rangle_{\mathcal{H}_K} = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j K(x_i, x_j) = \alpha^T K \alpha$$

So the problem is equivalent to

$$\arg\min_{\alpha} \|y - K\alpha\|_2^2 + \lambda \alpha^T K \alpha$$

Take derivative w.r.t. α:

$$K(y - K\alpha) + \lambda K\alpha = 0$$

and we obtain **the estimate of α:**

$$\hat{\alpha} = (K(K + \lambda I))^{-1} K y = (K + \lambda I)^{-1} y$$

Given data x, we can predict the values of y using KRR:

$$\hat{f} = \sum_{i=1}^{n} \hat{\alpha}_i K(X_i, x) = \hat{\alpha}^T \begin{pmatrix} K(X_1, x) \\ \vdots \\ K(X_n, x) \end{pmatrix} = y^T (K + \lambda I)^{-1} \begin{pmatrix} K(X_1, x) \\ \vdots \\ K(X_n, x) \end{pmatrix}$$

6.3 Related Topics (II): Nonparametric Regression

6.3.1 Smoothing Splines

For a regression model $y = f(X) + W$, where $X \in \mathbb{R}^p, y \in \mathbb{R}$. $f(\cdot)$ unknown and W is the noise.

Assumption f'' exists and f is in Sobolev Space.

Sobolev Space

A space S is a Sobolev space if S satisfies

$$S = \{f | f^{(v)} \text{ is absolutely continuous for } 0 \leq v \leq m-1\}$$

where $f^{(m)} \in L^2([0,1])$.

$$\|f\|_S^2 = \sum_{v=0}^{m} \int_0^1 [f^{(v)}(u)]^2 du$$

Estimation For data (y_i, x_i), we aims to solve

$$\hat{f} = \arg\min_{f \in S} \sum_{i=1}^{n} (y_i - f(X_i))^2 + \lambda \int (f''(t))^2 dt$$

where $\lambda \int (f''(t))^2 dt$ is the penalty function for smoothness.

The solution is a natural cubic spline:

$$f(X) = \sum_{j=1}^{n} \theta_j N_j(X)$$

where $\theta_1, \ldots, \theta_n$ are coefficients to be estimated and

$$\begin{cases} N_1(X) = 1 \\ N_2(X) = X \\ \cdots \\ N_{k+2}(X) = d_k(X) - d_{k-1}(X) \end{cases}$$

with $d_k(X) = \frac{(X - X_k)_+^3 - (X - X_n)_+^3}{X_n - X_k}$ where $a_+ = \max(a, 0)$.

Therefore, the problem becomes solving

$$\arg\min_{\theta \in \mathbb{R}} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \theta_j N_j(X_i))^2 + \lambda \int (f''(t))^2 dt$$

$$= \arg\min_{\theta\in\mathbb{R}} (Y - N\theta)^T (Y - N\theta) + \lambda\theta^T K\theta$$

where $K_{n\times n} = (K_{ij})_{n\times n} = (\int N_i''(t)N_j''(t)dt)_{n\times n}$.

The estimates of θ and f are:

$$\hat{\theta} = (N^T N + \lambda K)^{-1} N y$$

$$\hat{f} = \sum_{j=1}^{n} \hat{\theta}_j N_j(X)$$

If $X \in R^2$, it is called bivariate smoothing spline.

References

Chapter 5 Section 4, *ESL*.

When p is large (here we do not mean $p > n$, we mean $p = 10, 20, \dots$), how to solve nonparametric regression?

We can use Additive models.

6.3.2 Additive Model

Suppose the model is

$$Y = \alpha + f_1(X_1) + f_2(X_2) + \cdots + f_p(X_p) + W$$

where $f_1, \dots, f_p$ are unknown functions and $f_1'', \dots, f_p''$ are well defined.

Identifiability:

$$\sum_{i=1}^{n} f_j(X_{ij}) = 0, \quad \forall j = 1, \dots, p$$

Besides, X needs to be full rank so that f_j can be estimated.

We need to estimate α, f_j by solving

$$\arg\min_{\alpha, f_j} \sum_{i=1}^{n} (y_i - \alpha - \sum_{j=1}^{p} f_j(X_j))^2 + \sum_{j=1}^{p} \lambda_j \int (f_j''(t))^2 dt$$

where λ_j's are smoothness tuning parameters.

We use backfitting algorithm:

Backfitting Algorithm for Additive Models

Initialize $\hat{\alpha} = \bar{y}$;
Repeat for $j = 1, \ldots, p$:

$$\text{apply a cubic spline to } \{y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(X_{ik}),\ X_{ij}\}$$

The function is $f_j = \hat{f}_j - \frac{1}{n}\sum_{i=1}^{n} \hat{f}_j(X_{ij})$.

References

Chapter 9, *ESL*.

Sparse Additive Model When $p > n$, we consider sparse additive model.

References

Ravikumar, P., Lafferty, J., Liu, H., & Wasserman, L. (2009). Sparse additive models. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 71(5), 1009-1030.

Part II

CLASSIFICATION

Chapter 7

Classification

7.1 Classification Problems

7.1.1 Problem Setting

Problem Given $(X_i, y_i), \quad i = 1, \ldots, n, \quad X, y \in \mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y}$ is a discrete space and y_i is a class label.
Our **Goal** is to find a classifier $f : \mathcal{X} \to \mathcal{Y}$, s.t. $\hat{y} = f(X)$ is a good predictor of y for a new (X, y).

Examples:

- Binary Classification: $y \in \{0, 1\}$;
- Multiclass Classification: $y \in \{1, \ldots, K\}$.

Loss function Loss function is used to evaluate your prediction: $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

Example: 0-1 Loss

$$l(y, \hat{y}) = \mathbb{I}_{(y \neq \hat{y})}$$

where y is the true class label and $\hat{y}$ is the prediction.

When we minimize the loss function, we obtain the **optimal classifier**.

Suppose $p(X, y)$ is the joint distribution of (X, y). The Bayes risk of f is

$$R(f) = E[l(y, f(X))]$$

which is called **Expected Classification Error**. e.g., we can use 0-1 loss inside, which means $R(f) = E[l(y, f(X))] = E[\mathbb{I}_{(y \neq f(X))}] = P(y \neq f(X))$. So the **Goal** is to find f s.t. $R(f)$ is minimized.

7.1.2 Classification with 0-1 Loss

Binary Classification Consider the case when $p(X, y)$ is known. We use Binary classifier and 0-1 loss. The **optimal classifier** is $\arg\max_{k=0,1} P(y = k|X)$.

$$R(f) = E[l(y, f(X))] = E_X\left[E(l(y, f(X))|X)\right]$$
$$= E_X[l(1, f(X))P(y = 1|X) + l(0, f(X))P(y = 0|X)]$$
$$(\text{let } \eta(X) = P(y = 1|X),)$$
$$= E_X[\mathbb{I}_{(f(X)\neq 1)}\eta(X) + \mathbb{I}_{(f(X)\neq 0)}(1 - \eta(X))]$$
$$= E_X[\mathbb{I}_{(f(X)\neq 1)}(2\eta(X) - 1) + (1 - \eta(X))]$$

To find f^* that minimizes $R(f)$,

$$f^* = \mathbb{I}_{(\eta(X)\geq\frac{1}{2})} = \mathbb{I}_{(P(y=1|X)\geq\frac{1}{2})}$$
$$= \arg\max_{k=0,1} P(y = k|X)$$

is the optimal Bayes Classifier.

This motivates us to

1. obtain good estimators of $P(y = k|X)$;
2. set $\hat{f}(X) = \mathbb{I}_{(\hat{\eta}(X)\geq\frac{1}{2})}$.

Theorem: For any $\hat{\eta} = \hat{P}(y = 1|X)$, if we use it to construct the optimal classifier $\hat{f} = \mathbb{I}_{(\hat{\eta}(X)\geq\frac{1}{2})}$, we have

$$0 \leq R(\hat{f}) - R(f^*) \leq 2E_X|\eta(X) - \hat{\eta}(X)|$$

Proof. Recall that for any classifier f,

$$R(f) = E(l(y, f(X)) = E_X[\mathbb{I}_{(f(X)\neq 1)}\eta(X) + \mathbb{I}_{(f(X)\neq 0)}(1 - \eta(X))]$$

Let $f^* = \mathbb{I}_{(\eta(X)\geq\frac{1}{2})}$, consider another classifier $\hat{f}$,

$$R(\hat{f}) - R(f^*) = E_X[\mathbb{I}_{(\hat{f}(X)\neq 1)}\eta(X) + \mathbb{I}_{(\hat{f}(X)\neq 0)}(1-\eta(X)) - (\mathbb{I}_{(f^*(X)\neq 1)}\eta(X) + \mathbb{I}_{(f^*(X)\neq 0)}(1-\eta(X)))]$$

- If $\hat{f} = f^*$, $R(\hat{f}) - R(f^*) = 0$;
- If $f^*(X) = 1$, $\hat{f}(X) = 0$, then $\eta(X) \geq \frac{1}{2}$ and $R(\hat{f}) - R(f^*) = \eta(X) - (1 - \eta(X)) = 2|\eta(X) - \frac{1}{2}|$;
- If $f^*(X) = 0$, $\hat{f}(X) = 1$, then $\eta(X) < \frac{1}{2}$ and $R(\hat{f}) - R(f^*) = (1 - \eta(X)) - \eta(X) = 2|\eta(X) - \frac{1}{2}|$.

In summary,

$$R(\hat{f}) - R(f^*) = 2E_X(\mathbb{I}_{(\hat{f}\neq f^*)}|\eta(X) - \frac{1}{2}|)$$

when $f \neq f^*$, then $(\eta(X) - \frac{1}{2})(\hat{\eta}(X) - \frac{1}{2}) \leq 0$. This implies

$$|\eta(X) - \hat{\eta}(X)| = |\eta(X) - \frac{1}{2}| + |\hat{\eta}(X) - \frac{1}{2}| \geq |\eta(X) - \frac{1}{2}|$$

Therefore,

$$R(\hat{f}) - R(f^*) = 2E_X(\mathbb{I}_{(\hat{f}\neq f^*)}|\eta(X) - \frac{1}{2}|) \leq 2E_X(|\eta(X) - \hat{\eta}(X)|)$$

□

General cases When it comes to be K labels instead of two, the optimal Bayes classifier is $f^* = \arg\max_k P(y = k|X)$.

Proof.

$$\max_f 1 - R(f) = E(\mathbb{I}_{(y=f(X))}) = E_X[E(\mathbb{I}_{(y=f(X))}|X)]$$

$$= E_X[\sum_{y=1}^{K} P(y|X)\mathbb{I}_{(y=f(X))}]$$

$$\leq E_X[\sum_{y=1}^{K} \max_k P(y = k|X)\mathbb{I}_{(y=f(X))}]$$

$$= E_X[\max_k P(y = k|X)]$$

and " = " holds $\iff$ $f(X) = \arg\max_k P(y = k|X)$, which is the **optimal Bayes classifier**. □

Estimation of $P(y|X)$

- **Method 1:** Model-based approach: model the joint distribution $P(y, X)$ and then obtain $P(y|X)$.
  e.g., Linear discriminant Analysis (LDA); naive Bayes Classifier.
- **Method 2:** conditionally model $P(y|X)$.
  e.g., logistic regression.

References

Chapter 4, *ESL*.

7.2 Linear discriminant Analysis (LDA)

7.2.1 Model

Model Setting $y \sim Bernoulli(\pi)$, $\pi = P(y=1)$, $X \in \mathbb{R}^p$. $X|y=k \sim N_p(\mu_k, \Sigma)$, $k = 0, 1$. Here the variance Σ is the same for $k = 0, 1$.

$$\eta(x) = P(y=1|X=x) = \frac{P(y=1)P(X=x|y=1)}{p(x)}$$

$$\text{where } p(x) = P(y=1)P(X=x|y=1) + P(y=0)P(X=x|y=0)$$

$$= \frac{1}{1 + \frac{P(y=0)}{P(y=1)} \cdot \frac{P(X=x|y=0)}{P(X=x|y=1)}}$$

$$\text{where } p(y=0) = 1-\pi,\ p(y=1) = \pi, X|y=0 \sim N(\mu_0, \Sigma),\ X|y=1 \sim N(\mu_1, \Sigma)$$

$$= \frac{1}{1 + \exp\{\log(\frac{1-\pi}{\pi}) - \frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) + \frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1)\}}$$

$$= \frac{1}{1 + \exp\{-B^Tx - \gamma\}}$$

$$\text{where } \begin{cases} B = \Sigma^{-1}(\mu_1 - \mu_0) \\ \gamma = -\frac{1}{2}(\mu_1-\mu_0)^T\Sigma^{-1}(\mu_1+\mu_0) + \log(\frac{\pi}{1-\pi}) \end{cases}$$

Optimal Bayes Classifier Therefore, the optimal Bayes Classifier f^* for LDA is:

$$f^*(x) = \mathbb{I}_{(\eta(x) \geq \frac{1}{2})} = \mathbb{I}_{(B^Tx + \gamma \geq 0)}$$

7.2.2 Parameter Estimation

We need to estimate $\pi, \mu_0, \mu_1, \Sigma$ (MLE).

$$L(\pi, \mu_0, \mu_1, \Sigma) = \prod_{i=1}^n p(y_i)p(x_i|y_i)$$

$$= \prod_{i=1}^n \pi^{y_i}(1-\pi)^{1-y_i}(2\pi)^{-\frac{p}{2}}|\Sigma|^{-\frac{1}{2}} \exp\{-\frac{1}{2}(x_i - \mu_{y_i})^T\Sigma^{-1}(x_i - \mu_{y_i})\}$$

$$= \prod_{i=1}^n \prod_{k=0}^1 P(y_i=k)^{\mathbb{I}_{(y_i=k)}} p(x_i|y_i=k)^{\mathbb{I}_{(y_i=k)}}$$

So the MLEs are

$$\begin{cases} \hat{\pi} = \frac{1}{n}\sum_{i=1}^n \mathbb{I}_{(y_i=1)} \\ \hat{\mu_k} = \frac{\sum_{i=1}^n \mathbb{I}_{(y_i=k)}x_i}{\sum_{i=1}^n \mathbb{I}_{(y_i=k)}} \\ \hat{\Sigma} = \frac{1}{n}\sum_{k=0}^1 \sum_{i:y_i=k}(x_i - \hat{\mu_k})(x_i - \hat{\mu_k})^T \end{cases}$$

7.2.3 Quadratic Discriminant Analysis (QDA)

We extend LDA to non-linear settings.

Model Setting $y \sim Bernoulli(\pi)$, $\pi = P(y=1)$, $X \in \mathbb{R}^p$. $X|y=k \sim N_p(\mu_k, \Sigma_k)$, $k=0,1$. Here the variances Σ_k are different for $k=0,1$ ($\Sigma_0 \neq \Sigma_1$).

We set

$$\log \frac{P(y=1|X)}{P(y=0|X)} = \delta_1(X) - \delta_0(X)$$

where $\delta_k(X) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(X-\mu_k)^T\Sigma_k^{-1}(X-\mu_k) + \log\pi_k$.

Optimal Classifier The **optimal classifier** is $f^*(X) = \arg\max_{k=0,1}\delta_k(X)$.

7.2.4 Multiclass LDA & QDA

The model is

$$\begin{cases} y \sim \text{Categorical Distribution}, P(y=k) = \pi_k \\ X|y=k \sim N_p(\mu_k, \Sigma_k) \end{cases}$$

Multiclass LDA $\Sigma_1 = \Sigma_2 = \cdots = \Sigma$

$$\log \frac{P(y=k|X)}{P(y=l|X)} = \delta_k(X) - \delta_l(X)$$

where $\delta_k(X) = X^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + \log\pi_k$.
The optimal classifier is

$$f^*(X) = \arg\max_k \delta_k(X)$$

Proof.

$$f^*(X) = \arg\max_k P(y=k|X)$$

$$= \arg\max_k \frac{P(y=k)P(X|y=k)}{\sum_{j=1}^K P(y=j)P(X|y=j)}$$

$$= \arg\max_k P(y=k)P(X|y=k)$$

$$= \arg\max_k \log P(y=k) + \log P(X|y=k)$$

$$= \arg\max_k \log\pi_k - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(X-\mu_k)^T\Sigma^{-1}(X-\mu_k)$$

$$= \arg\max_k \log\pi_k - \frac{1}{2}X^T\Sigma X + \mu_k^T\Sigma^{-1}X - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k$$

$$= \arg\max_k \log \pi_k + \mu_k^T \Sigma^{-1} X - \frac{1}{2}\mu_k^T \Sigma^{-1} \mu_k$$
$$= \arg\max_k \delta_k(X)$$

□

The classifier is linear in Σ^{-1} because $\delta_k(X)$ is linear in Σ^{-1}. In practice, we need to estimate the parameters and use $\hat{\delta}(X)$ instead.

Multiclass QDA $\Sigma_1 \neq \ldots \neq \Sigma_k$

$$\log \frac{P(y=k|X)}{P(y=l|X)} = \delta_k'(X) - \delta_l'(X)$$

where $\delta_k'(X) = -\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(X-\mu_k)^T\Sigma_k^{-1}(X-\mu_k) + \log \pi_k$.
The optimal classifier is

$$f^*(X) = \arg\max_k \delta_k'(X)$$

7.3 Applications of Classification

7.3.1 Generative Model

Model

$$\begin{cases} y \sim \text{Categorical}(\pi_1, \ldots, \pi_K) \\ X|y \sim P(X|y, \theta) \end{cases}$$

Optimal Classifier

$$f^*(X) = \arg\max_k P(y = k|X)$$

where

$$P(y = k|X) = \frac{\pi_k P(X|y = k, \theta)}{\sum_{k=1}^K \pi_k P(X|y = k, \theta)}$$

7.3.2 Naive Bayes Classifier

Model

$$y \sim \text{Categorical}(\pi),\ P(y = k) = \pi_k > 0$$

$$X \sim \text{Categorical Dist (each } X_j \text{ has } L \text{ categories)}$$

"naive": $X_{p\times 1}|y = \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix}|y$ is conditionally independent, i.e., $X_i \perp X_j|y\ (i \neq j)$.

Therefore,

$$X_j|y = k \sim \text{Categorical}(P_{ljk}),\ l = 1, \ldots, L;\ j = 1, \ldots, p;\ k = 1, \ldots, K$$

Derive $P(y|X)$ Next we derive the form of $P(y|X)$:

$$P(y = k|X) = \frac{\pi_k P(X|y = k)}{\sum_{k=1}^K \pi_k P(X|y = k)}$$

where $P(X|y = k) = \prod_{j=1}^p P(X_j|y = k) = \prod_{j=1}^p \prod_{l=1}^L P_{ljk}^{\mathbb{I}(X_j=l)}$.

So

$$\begin{aligned} P(y = k|X) &= \frac{\exp\{\log \pi_k + \sum_{j=1}^p \sum_{l=1}^L \log P_{ljk}\mathbb{I}_{(X_j=l)}\}}{\sum_{k=1}^K \exp\{\log \pi_k + \sum_{j=1}^p \sum_{l=1}^L \log P_{ljk}\mathbb{I}_{(X_j=l)}\}} \\ &= \frac{\exp\{B_k^T Z\}}{\sum_{k=1}^K \exp\{B_k^T Z\}} \end{aligned}$$

where $B_k = (\log \pi_k, \log P_{ljk})$ and $Z = (1, \mathbb{I}_{(X_j=l)})$. And the **optimal classifier** is the k with the largest $P(y = k|X)$, i.e., $f^* = \arg\max_k P(y = k|X)$.
This is similar to K-class logistic regression problem.

7.3.3 Logistic Regression

Model This is a binary classification method for $K = 2$ case. There is no assumption on X and we use binary response $y \in \{0, 1\}$. Logistic Regression models $P(y|X)$ through log-odds

$$\log \frac{P(y=1|X)}{P(y=0|X)} = \log \frac{P(y=1|X)}{1 - P(y=1|X)} = \theta^T X = \theta_0 + \theta_1 X_1 + \cdots + \theta_p X_p$$

where $\theta = (\theta_0, \ldots, \theta_p)^T$ and $X = (1, X_1, \ldots, X_p)^T$.

It is equivalent to

$$y|X \sim Bernoulli(u(X))$$

where $u(X) = P(y = 1|X) = \frac{e^{\theta^T X}}{1+e^{\theta^T X}}$.

Parameter Estimation Suppose we have data $(X_i, y_i),\ i = 1, \ldots, n$. The likelihood function is

$$L(\theta) = \prod_{i=1}^{n} P(y_i|X_i) = \prod_{i=1}^{n} u(X_i)^{y_i}(1 - u(X_i))^{1-y_i}$$

$$l(\theta) = \log L(\theta) = \sum_{i=1}^{n} [y_i \log u(X_i) + (1 - y_i) \log(1 - u(X_i)]$$

$$= \sum_{i=1}^{n} [y_i \theta^T X_i - \log(1 + \exp(\theta^T X_i))]$$

where $u(X_i) = \frac{e^{\theta^T X_i}}{1+e^{\theta^T X_i}}$. Take the derivative:

$$\frac{dl}{d\theta} = \sum_{i=1}^{n} [y_i X_i - \frac{\exp(\theta^T X_i)}{1 + \exp(\theta^T X_i)} \cdot X_i]$$

$$= \sum_{i=1}^{n} (y_i - u(X_i)) X_i$$

which is called **score function**.

There is no closed form solution to the score function. So we use some numerical methods, such as gradient descent method, Newton method, SGD method, etc.

Iterative Reweighted Least Square Algorithm (IRLS)
It is a type of Newton - Raphson Algorithm. The Hessian matrix is

$$H = \frac{\partial^2 l(\theta)}{\partial \theta \partial \theta^T} = -\sum_{i=1}^{n} \frac{\partial u(X_i)}{\partial \theta} X_i^T$$

$$= -\sum_{i=1}^{n} u(X_i)(1-u(X_i))X_i X_i^T = -X^T W X$$

where

$$\begin{cases} X_{(n\times(p+1))} = \begin{pmatrix} X_1^T \\ \vdots \\ X_n^T \end{pmatrix} = \begin{pmatrix} X_{11} & \cdots & X_{1p} \\ \vdots & & \vdots \\ X_{n1} & \cdots & X_{np} \end{pmatrix} \\ W = \begin{pmatrix} u(X_1)(1-u(X_1)) & & \\ & \ddots & \\ & & u(X_n)(1-u(X_n)) \end{pmatrix} \end{cases}$$

At time t, we update parameter θ:

$$\begin{aligned} \theta^{(t+1)} &= \theta^{(t)} - \left(H^{(t)}\right)^{-1} \nabla_\theta \ell(\theta)\big|_{\theta=\theta^{(t)}} \\ &= \theta^{(t)} + \left(X^T W^{(t)} X\right)^{-1} X^T \left(y - u^{(t)}\right) \\ y &= \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}, \quad u^{(t)} = \begin{pmatrix} u^{(t)}(X_1) \\ \vdots \\ u^{(t)}(X_n) \end{pmatrix} \\ \text{multiply by } H^{-1}H \quad &= \quad (H^{-1})^{(t)} \left[H^{(t)}\theta^{(t)} - X^T(y-u^{(t)})\right] \\ &= (H^{(t)})^{-1}\left[-X^T W^{(t)} X\theta^{(t)} - X^T(y-u^{(t)})\right] \\ &= -(H^{(t)})^{-1} X^T W^{(t)} \left[X\theta^{(t)} + (W^{(t)})^{-1}(y-u^{(t)})\right] \\ &= (X^T W^{(t)} X)^{-1} X^T W^{(t)} Z^{(t)} \end{aligned}$$

where $Z^{(t)} = X\theta^{(t)} + (W^{(t)})^{-1}(y-u^{(t)})$ and it's weighted least square (WLS). So $Z_i^{(t)} = X_i^T\theta^{(t)} + \epsilon_i^{(t)}$ where $\epsilon_i^{(t)} \sim N\left(0, \frac{1}{W(X_i)(1+W(X_i))}\right)$.

Penalized Logistic Regression Adding regularization to the objective function of Logistic regression still yields a linear classifier.

LASSO:

$$\frac{1}{N}\sum_{i=1}^{n}[y_i\theta^T X_i - \log(1+\exp(\theta^T X_i))] + \lambda\|\theta\|_1$$

Ridge:

$$\frac{1}{N}\sum_{i=1}^{n}[y_i\theta^T X_i - \log(1+\exp(\theta^T X_i))] + \lambda\|\theta\|_2^2$$

7.3.4 Kernel Logistic Regression

Model Here we do not use $\theta^T X$, instead we use non-linear function $f(X)$.
We specify a kernel function $K: \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$, $X \in \mathbb{R}^p$. The function f satisfies

$$K \to \mathcal{H}_K$$
$$f \in \mathcal{H}_K$$

The loss function is "-log likelihood + penalty function for f".

$$\hat{f} = \arg\min_{f \in \mathcal{H}_K} \sum_{i=1}^{n} [-y_i f(X_i) + \log(1 + e^{f(X_i)})] + \frac{\lambda}{2} \|f\|^2_{\mathcal{H}_K}$$

Parameter Estimation By representer theorem, we know the solution $\hat{f}(x) = \sum_{i=1}^{n} \hat{\alpha}_i K(X_i, x)$ and $\hat{\alpha} = (\hat{\alpha}_1, \ldots, \hat{\alpha}_n)^T_{n \times 1}$.
We know $\|f\|^2_{\mathcal{H}_K} = \alpha^T K_n \alpha$, the questions becomes

$$\hat{\alpha} = \arg\min_{\alpha \in \mathbb{R}^n} \frac{1}{n} \sum_{i=1}^{n} [-y_i \alpha^T K_i + \log(1 + e^{\alpha^T K_i})] + \frac{\lambda}{2} \alpha^T K_n \alpha$$

Use IRLS to solve this question. At t-step,

$$\begin{cases} \alpha^{(t+1)} = (K_n^T W K_n + \lambda K_n)^{-1} K_n^T W^{(t)} Z^{(t)} \\ Z^{(t+1)} = \hat{f}^{(t)} + (W^{(t)})^{-1}(y - u^{(t)}) \end{cases}$$

and $\hat{f}^{(t)} = K_n \alpha^{(t)}$.

7.3.5 Generalized Additive Model (GAM)

Model Binary response $y \in \{0, 1\}$. Assume

$$f(X) = \alpha + f_1(X_1) + \cdots + f_p(X_p)$$

where $f_i(X_i)$ is unknown and smooth.
The Loss function is

$$-\text{log likelihood} + \sum_{j=1}^{p} \lambda_j \int [f_j''(t_j)]^2 dt_j$$

where we add smoothing penalty.

Parameter Estimation Here we use **Local Scoring Algorithm**:

1. Initiate starting value $\hat{f}_j = 0,\ j = 1, \ldots, p.\ \hat{L} = \log(\frac{\bar{y}}{1-\bar{y}})$;
2. Define $\hat{f}(X_i) = \hat{\alpha} + \sum_{j=1}^{p} \hat{f}_j$,
   (a) Construct $Z_i = \hat{f}(X_i) + \frac{y_i - \hat{u}_i}{\hat{u}_i(1-\hat{u}_i)}, \quad \hat{u}_i = \frac{e^{\hat{f}(X_i)}}{1+e^{\hat{f}(X_i)}}$;

(b) Set $\hat{w}_i = \hat{u}_i(1-\hat{u}_i)$;

(c) Fit an additive model to Z_i's with weights w_i using backfitting algorithm;

(d) Repeat until convergence.

The estimated function is given by

$$\hat{f}_1, \ldots, \hat{f}_p = \arg\min \sum_{i=1}^{n} \hat{w}_i (Z_i - \hat{\alpha} - \sum_{j=1}^{p} f_j(X_{ij}))^2 + \sum_{j=1}^{p} \lambda_j \int [f_j''(t_j)]^2 dt_j$$

where the first term, $\sum_{i=1}^{n} \hat{w}_i (Z_i - \hat{\alpha} - \sum_{j=1}^{p} f_j(X_{ij}))^2$ is weighted RSS; the second term, $\sum_{j=1}^{p} \lambda_j \int [f_j''(t_j)]^2 dt_j$, is the smoothing penalty.

References

Chapter 9 Section 1, *ESL*.

7.3.6 Projection Pursuit Regression (PPR)

This method involves nonparametric regression and dimension reduction.

Model We model $f(X)$ nonparametrically on Z_i's where $Z_1, \ldots, Z_m$ are new features that is linear combination of $X_1, \ldots, X_p$ ($Z_m = W_m^T X$).

$$f(X) = \sum_{m=1}^{M} g_m(w_m^T X), \ w_m \in \mathbb{R}^p \text{ unknown and } g_m \text{ is unknown too.}$$

Then we perform regression

$$E(y|X) = f(X)$$

And the loss function is given by

$$\sum_{i=1}^{n} (y_i - \sum_{m=1}^{M} g_m(w_m^T X_i)]^2) + \text{penalty } (g)$$

Parameter Estimation Use iterative algorithm.
For each $m = 1, \ldots, M$,

- Given w_m, use smoothing spline to get g_m;
- Given g_m, use Gradient based algorithms to estimate w_m.

Note

- We select M using cross-validation. When $M = 1$, it is the single index model;
- It is closely related to neural network where model where $f(X)$ is often specified to be multilayer.

References

Chapter 11 Section 2, *ESL*.

Chapter 8

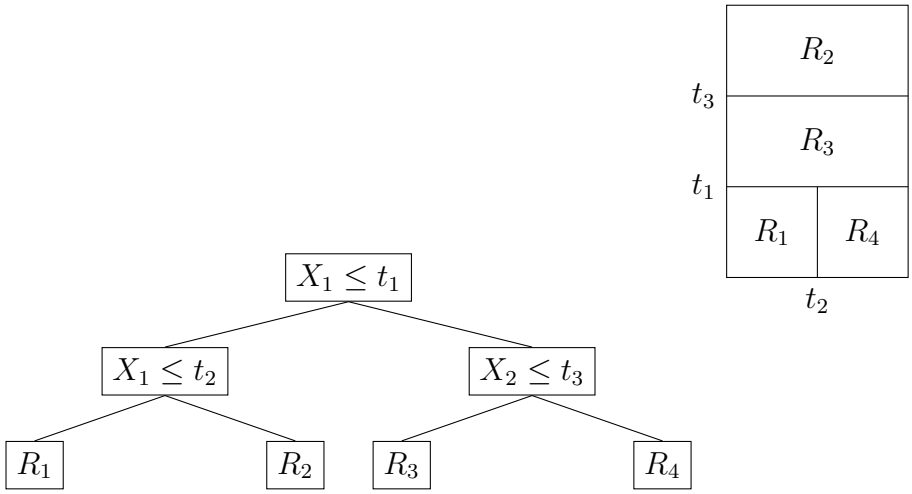
Classification Trees

8.1 Classification And Regression Tree (CART)

8.1.1 Idea

Partition the sample space (feature space) into hypercubes (regions) and assign classes to each hypercube (region).

Example 8.1.1
Suppose $X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \in [0,1]^2$. Construct a classification tree:



Notes

In this section, we only talk about classification trees. Regression trees will not be included.

8.1.2 CART

Suppose we have data $(y_i, X_i)_{i=1}^n$ where $X_i \in \mathbb{R}^p, \quad y_i \in \{1, \dots, K\}$. Suppose we have a partition of X space into M categories $R_1, \dots, R_M$.

Let $\hat{p}_{mk} = \frac{1}{n_m}\sum_{X_i \in R_m} \mathbb{I}_{(y_i=k)}$, and n_m is the number of observations in the m-th region R_m: $n_m = \sum_{i=1}^{n} \mathbb{I}_{(X_i \in R_m)}$. So this is the estimate of the proportion $p_{mk} = P(y_i = k | X_i \in R_m)$.
For a new observation X^* in R_m, we classify it to

$$K(m) = \arg\max_{1 \le k \le K} \hat{p}_{mk}$$

Notes

CART can be viewed as modeling:

Binary:

$$P(y = 1|X) = \sum_{m=1}^{M} p_m \mathbb{I}_{(X \in R_m)}$$

where $p_m = P(y = 1 | X \in R_m)$. The formula above is equivalent to

$$\log \frac{P(y=1|X)}{P(y=0|X)} = f(X) = \sum_{m=1}^{M} \log \frac{p_m}{1-p_m} \mathbb{I}_{(X \in R_m)}$$

Multiclass $(k = 1, \ldots, K)$:

$$P(y = k|X) = \sum_{m=1}^{M} p_{mk} \mathbb{I}_{(X \in R_m)}$$

Our **GOAL** is to estimate p_{mk}, R_m and M then.

8.1.3 Estimation

How to do partition of n data points?
Use node impurity to help us to partition the data points. Our goal is to make the impurity decrease most in each step.

Node impurity Let $Q_m(T)$ to be a loss function that measures node impurity. m means m-th region and T is the tree. There are three common impurity measures as follows:

1. **Misclassification error**:

$$Q_m(T) = \frac{1}{n_m} \sum_{X_i \in R_m} \mathbb{I}_{(y_i \neq k(m))} = 1 - \hat{p}_{mk}(m) = 1 - \max_k \hat{p}_{mk}$$

where $k(m) = \arg\max_k \hat{p}_{mk}$ and $\hat{p}_{mk}(m) = \max_k \hat{p}_{mk}$;

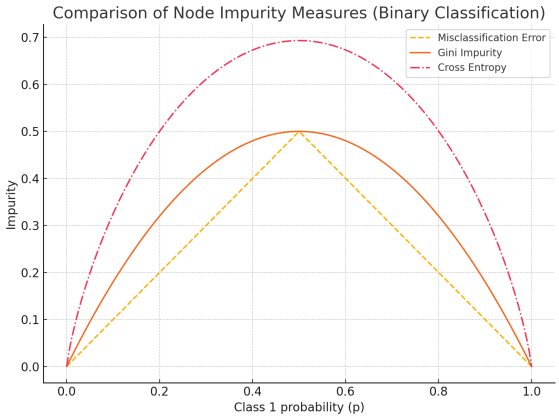


Figure 8.1: comparison of three impurity measures

2. **Gini index**:

$$Q_m(T) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2 = \sum_{k \neq k'} \hat{p}_{mk}\hat{p}_{mk'}$$

3. **Cross entropy (or deviance)**

$$Q_m(T) = -\sum_{k=1}^{K} \hat{p}_{mk} \log(\hat{p}_{mk})$$

This is the negative log likelihood.

From Figure 8.1, we can observe that the **misclassification error** is the least sensitive to class imbalance (p close to 0 or 1); **cross entropy** is very sensitive to small or large probabilities; **Gini index** is smooth and used by default in CART.

Procedure First we start with all data points.
Consider a splitting variable $X_j,\ j \in \{1, \ldots, p\}$ and a split point $s \in \mathbb{R}$. We have region 1 and region 2

$$R_1 = \{X_j \leq s\}, \quad R_2 = \{X_j > s\}$$

Then estimate $j,\ s$ by

$$\hat{j},\ \hat{s} = \arg\min_{j,s}(n_1 Q_1 + n_2 Q_2)$$

where $Q_1,\ Q_2$ are the loss functions for region 1 and 2; $n_1,\ n_2$ are the numbers of observations in region 1 and 2. We go through all j and s and choose the pair (j, s) with the smallest $(n_1 Q_1 + n_2 Q_2)$.
Then we do splitting and repeat the procedure inside the two subregions.

Algorithm First we start with a large tree T (we could not split it anymore). Prune the tree by minimizing

$$C_\alpha(T) = \sum_{m=1}^{m(T)} n_m Q_m(T) + \alpha m(T)$$

where α is the tuning parameter which controls the number of regions. $\sum_{m=1}^{m(T)} n_m Q_m(T)$ is a kind of the total prediction error of the tree; $\alpha m(T)$ is the penalty of complexity which prevents overfitting. The more branches the tree has, the more complicated it is.

8.1.4 Discussion

Is CART a good method?

Advantages: Interpretable;
Disadvantages:

- It has large variance;
- An error in the top split is going to affect the accuracy;
- Not robust.

So we use Bagging more often, which is more stable and will be introduced in the next section.

8.2 Bootstrap Aggregating (Bagging)

8.2.1 Idea

- Generate perturbed version of the data. Usually it is done by bootstrapping;
- Train a CART on each perturbed data;
- Aggregating results by majority voting.

Because trees are sensitive to small changes in data, they easily overfit. Bagging reduces model variance by averaging predictions from multiple trees trained on slightly different data, thus improving generalization.

8.2.2 Procedure

1. Create B bootstrapped datasets: $D_1, \ldots, D_B$, where B is large and D_b includes $(X_i^{(b)}, y^{(b)})_{i=1}^n$;
2. Train a CART $T_b(X)$ using D_b for $b = 1, \ldots, B$;
3. For a new observation, classify it to the majority class predicted by B trees.

8.2.3 Discussion

Advantages

- Reduces the variance of CART, but does not affect the bias;
- It is an average of B random variables; $T_1, \ldots, T_B$ are identically distributed (not independent) with positive correlation:

$$var(\frac{T_1 + \cdots + T_B}{B}) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2 \to \rho\sigma^2 \quad \text{if } B \to \infty$$

where ρ represents the correlation between trees. Higher correlation means less variance reduction; lower correlation means more variance reduction.

Notes

Compute variable (feature) importance: the total amount that the loss function decreased by splitting over a predictor averaged over B trees.

- Imagine you have built B different trees (models) in your bagging method.
- For each predictor (feature), you look at every time a tree split on that predictor and measure how much that split reduced the model's loss function.
- You add up all these reductions for the predictor across all B trees.
- Finally, you average this total by dividing by the number of trees (B).

8.2.4 Random Forest

Procedure

1. Create B bootstrapped datasets: $D_1, \ldots, D_B$, where B is large and D_b includes $(X_i^{(b)}, y^{(b)})_{i=1}^n$;
2. Train a CART $T_b(X)$ using D_b for $b = 1, \ldots, B$ as follows:
   at each node, consider a random subset of $(d < p)$ **variables for the next split (usually** $d = \sqrt{p}$ **or** $d < \sqrt{p}$**) instead of all** p **variables;**
3. For a new observation, classify it to the majority class predicted by B trees.

This can increase the diversity of trees due to feature randomness. Therefore, it can further reduce variance and the risk of overfitting.

Advantages We have $T_1, \ldots, T_B$ are identically distributed (not independent) with positive correlation:

$$var(\frac{T_1 + \cdots + T_B}{B})_{RF} = \rho_{RF}\sigma^2 + \frac{1 - \rho_{RF}}{B}\sigma^2 \to \rho_{RF}\sigma^2 \quad \text{if } B \to \infty$$

where ρ_{RF} represents the correlation between trees. It is smaller than ρ in Bagging because of the random subsets. Therefore, the variance of Random Forest is smaller than Bagging.

References

Chapter 15, *ESL*.

8.3 Boosting

Famous boosting methods: AdaBoost, Gradient Boosting (XGBoost, GBDT, LightGBM, CatBoost).

Main Idea Boosting makes predictions by sequentially combining a sequence of weak classifiers (e.g., trees).
Most boosting algorithms consist of iteratively learning weak classifiers w.r.t. a distribution and adding them to a final strong classifier.
When they are added, they are weighted in a way that is related to the weak learners' accuracy. After a weak learner is added, the data weights are readjusted, known as "re-weighting".

Note:

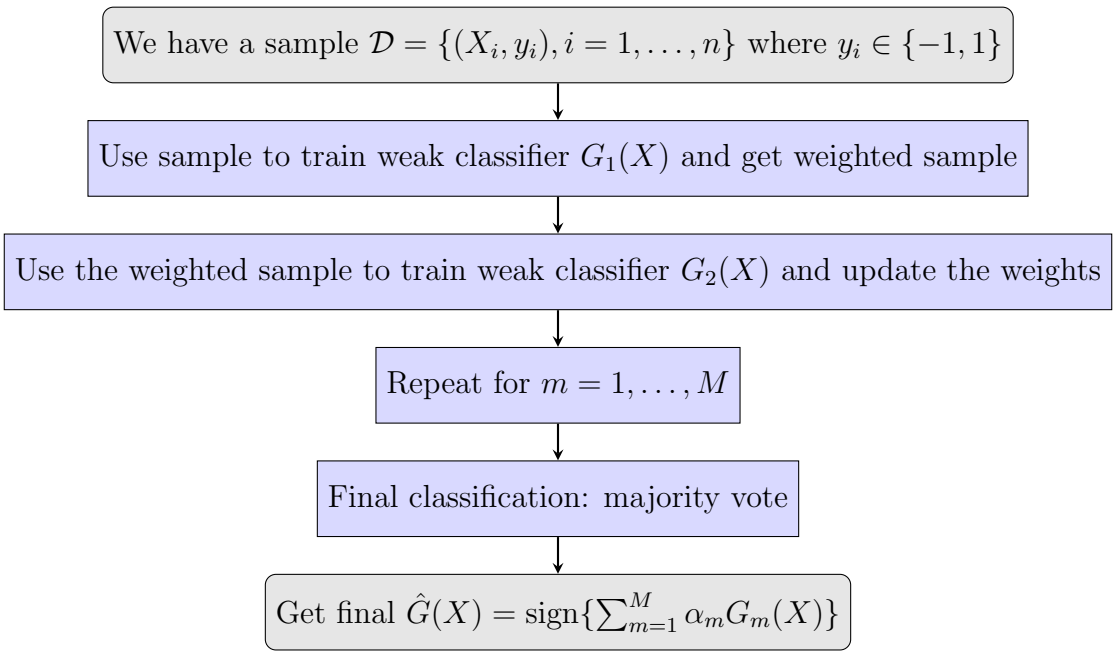
- Weak classifiers must be better than random guessing;
- Misclassified input data gain a higher weight and examples that are classified correctly lose weight. Thus, future weak learners focus more on the examples that previous weak learners misclassified.

References

Chapter 10, *ESL*.

8.3.1 Adaptive Boosting (AdaBoost)

Procedure



where α_m's and G_m's are the weights and trees calculated from the data (α_m is not the weight of sample w_i).

Notes

- Each tree is grown using information from previous trees;
- Does not use bootstrapping. Instead, we fit trees on modified data of the original data;
- At m-th step, the observations that are misclassified get higher weights.

Algorithm

1. Initialize $w_i = \frac{1}{n}, \quad i = 1, \ldots, n$;
2. For $m = 1, \ldots, M$,
   fit $G_m(X)$ to training data (X_i, y_i) with weights $w_i, \quad i = 1, \ldots, n$;

$$err_m = \frac{\sum_{i=1}^n w_i \mathbb{I}_{(y_i \neq G_m(X_i))}}{\sum_{i=1}^n w_i}$$

$$\alpha_m = \log \frac{1 - err_m}{err_m}$$

3. Update weights $w_i \leftarrow w_i \exp\{\alpha_m \mathbb{I}_{(y_i \neq G_m(X_i))}\}$ and scale the weights;
4. Output: $\hat{G}(X) = sign\left(\sum_{m=1}^M \alpha_m G_m(X)\right)$.

Interpretation of α_m

The weight α_m tells the AdaBoost algorithm how much faith it should place in the m-th weak classifier.

- Large α_m (positive) means the classifier is better than random (significantly less than 50% error) and thus deserves a stronger vote in the final ensemble;
- Small (positive) α_m means that while the classifier does some good, it is close to random and only worth a modest vote;
- Negative α_m means the classifier is actually doing worse than random guessing on this weighted dataset. Ideally, this should never happen because our weak learner should at least be better than random guessing.

8.3.2 Boosting and Additive models

Boosting can be seen as fitting on an additive model of the form

$$f(X) = \sum_{m=1}^M B_m b(X, \gamma_m)$$

where $B_m,\ m = 1, \ldots, M$ are the coefficients for the model and $b(X, \gamma_m)$'s are the functions of X with a set of parameters γ_m.
The **goal** is to solve this optimization problem:

$$\min_{\{B_m, \gamma_m\}_{m=1}^M} \sum_{i=1}^N L(y_i, \sum_{m=1}^M B_m b_m\left(X_i, \gamma_m\right))$$

Algorithm: Forward Stagewise Additive Modeling

1. Initialize $f_0(X) = 0$;
2. For $m = 1$ to M,
   Compute
   $$B_m, \gamma_m = \arg\min_{B,\gamma} \sum_{i=1}^n L(y_i, f_{m-1}(X_i) + Bb(X_i, \gamma));$$
   Set $f_m(X) = f_{m-1}(X) + B_m b(X, \gamma_m)$;
3. Return $\hat{f}(X) = f_M(X)$.

Equivalence of AdaBoost and Exponential loss Consider exponential loss function

$$L(y, f(X)) = \exp\{-yf(X)\}$$

The optimal function is given by

$$f^*(X) = \arg\min_f E_{y|X}[\exp(-yf(X))] = \frac{1}{2}\log\frac{P(y=1|X)}{P(y=-1|X)}$$

So $P(y=1|X) = \frac{e^{f(X)}}{e^{f(X)}+e^{-f(X)}} = \frac{1}{1+e^{-2f(X)}}$.

We can show that AdaBoost is equivalent to the forward stagewise additive modeling algorithm with exponential loss.

For AdaBoost, the basis functions are the individual classifiers (trees) $G_m(X) \in \{-1, 1\}$. Consider forward stagewise additive modeling algorithm, we solve

$$(B_m, G_m) = \arg\min_{B,G} \sum_{i=1}^n \exp\{-y_i[f_{m-1}(X_i) + BG(X_i)]\}$$

This can be expressed as

$$(B_m, G_m) = \arg\min_{B,G} \sum_{i=1}^n w_i^{(m)} \exp\{-y_i BG(X_i)\} \qquad (\star)$$

where $w_i^{(m)} = \exp(-yf_{m-1}(X_i))$ are weights applied to each observation and changeable at each iteration.

The solution to $(\star)$ is:
For any $B > 0$,

$$G_m = \arg\min \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}_{(y_i \neq G(X_i))} \qquad (\star\star)$$

which is the classification tree that minimizes error rate for predicting y_i.

Proof. Fix B,

$$G_m = \arg\min_G \sum_{i=1}^{n} \exp\big(-y_i\big(f_{m-1}(X_i) + BG(X_i)\big)\big)$$

$$= \arg\min_G \sum_{i=1}^{n} w_i^{(m)} \exp\big(-y_i BG(X_i)\big)$$

Recall $y_i \in \{-1, 1\}$, $G(X) \in \{-1, 1\}$,

$$= \arg\min_G \sum_{i=1}^{n} w_i^{(m)} \Big(\exp(-B) \underbrace{\mathbb{I}_{(y_i = G(X_i))}}_{=1-\mathbb{I}_{(y_i \neq G(X_i))}} + \exp(B)\mathbb{I}_{(y_i \neq G(X_i))}\Big)$$

$$= \arg\min_G \sum_{i=1}^{n} w_i^{(m)} \exp(-B) + \sum_{i=1}^{n} w_i^{(m)} \big(e^B - e^{-B}\big)\, \mathbb{I}_{(y_i \neq G(X_i))}$$

$$= \arg\min_G \sum_{i=1}^{n} w_i^{(m)}\, \mathbb{I}_{(y_i \neq G(X_i))}$$

□

Plug in $(\star\star)$ into $(\star)$, we get

$$\hat{B}_m = \frac{1}{2} \log \frac{1 - err_m}{err_m}$$

where $err_m = \frac{\sum_{i=1}^n w_i^{(m)} \mathbb{I}_{(y_i \neq G_m(X_i))}}{\sum_{i=1}^n w_i^{(m)}}$.

Proof.

$$B_m = \arg\min_B \sum_{i=1}^{n} \exp\Big(-y_i\big(G_{m-1}(X_i) + B\, G_m(X_i)\big)\Big)$$

$$= \arg\min_B \sum_{i=1}^{n} w_i^{(m)} \exp\big(-B\, y_i\, G_m(X_i)\big)$$

$$= \arg\min_B \sum_{i=1}^{n} \exp(-B)\, \mathbb{I}_{(y_i = G_m(X_i))} + \exp(B)\, \mathbb{I}_{(y_i \neq G_m(X_i))}$$

$$= \arg\min_B \sum_{i=1}^{n} \exp(-B)\Big(1 - \mathbb{I}_{(y_i \neq G_m(X_i))}\Big) + \exp(B)\, \mathbb{I}_{(y_i \neq G_m(X_i))}$$

$$= \arg\min_B \sum_{i=1}^n w_i^{(m)} \left[e^{-B} + (e^B - e^{-B}) \mathbb{I}_{(y_i \neq G_m(X_i))} \right]$$

$$= \arg\min_B \; e^{-B} + (e^B - e^{-B}) \, \mathrm{err}_m$$

where

$$\mathrm{err}_m = \frac{\sum_{i=1}^n w_i^{(m)} \mathbb{I}_{(y_i \neq G_m(X_i))}}{\sum_{i=1}^n w_i^{(m)}}$$

Calculate the derivative, and we can find the best value for B_m is:

$$B_m = \frac{1}{2} \log \frac{1 - err_m}{err_m}$$

□

Then $f_m(X) = f_{m-1}(X) + B_m(G_m(X))$ and $w_i^{(m)} = w_i^{(m)} \exp\{-B_m y_i G_m(X_i)\}$. Use the fact that

$$-y_i G_m(X_i) = 2\mathbb{I}_{(y_i \neq G_m(X_i))} - 1$$

We update weights by

$$w_i^{(m+1)} = w_i^{(m)} \exp\{\alpha_m \mathbb{I}_{(y_i \neq G_m(X_i))}\}$$

where $\alpha_m = 2B_m$. The steps above are the same as AdaBoost's method.

References

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The annals of statistics*, 28(2), 337-407.

8.3.3 Gradient Boosting

While AdaBoost tries to fit a tree that minimizes the loss, Gradient Boosting tries to fit the gradient, which is arguably easier.

Here we use Steepest Descent algorithm, a.k.a. Gradient Descent Algorithm. Note that we need to find the optimal step size (or learning rate) at each step!

Algorithm: Gradient Tree Boosting

1. We have dataset $\mathcal{D} = \{(X_i, y_i)\}_{i=1}^n$, initialize $G_0(X) = \arg\min_\gamma \sum_{i=1}^n L(y_i, \gamma)$;
2. For $m = 1, \ldots, M$,
   - **(a)** Compute (negative) gradients:

$$g_{im} = -\frac{\partial L(y_i, G_{m-1}(X_i))}{\partial G_{m-1}(X_i)}.$$

   - **(b)** Fit a tree to the targets g_{im} to get terminal regions R_{jm}, $j = 1, \ldots, J_m$.

(c) Compute for each terminal node:

$$b_{jm} \;=\; \arg\min_{b} \sum_{X_i \in R_{jm}} L\Big(y_i,\; G_{m-1}(X_i) \;+\; b\Big).$$

(d) Take the gradient step:

$$G_m(X) \;=\; G_{m-1}(X) \;+\; \sum_{j=1}^{J_m} b_{jm}\, \mathbb{I}_{(X \in R_{jm})}.$$

Return: $\hat{G}(X) \;=\; G_M(X)$.

References

Chapter 10, Section 10, *ESL*.

8.4 Support Vector Machine (SVM)

8.4.1 SVM

References

Chapter 12, *ESL*;
Chapter 7, *PRML*.

We focus on binary classification, usually $\mathcal{Y} = \{-1, 1\}$ and we observe $(X_i, y_i)_{i=1}^n$ from $p(X, y)$.

Assumptions We assume two classes can be separated by at least one hyperplane. A (linear) hyperplane can be written as $f(X) = B_0 + B^T X = 0, \quad X \in \mathbb{R}^p$.

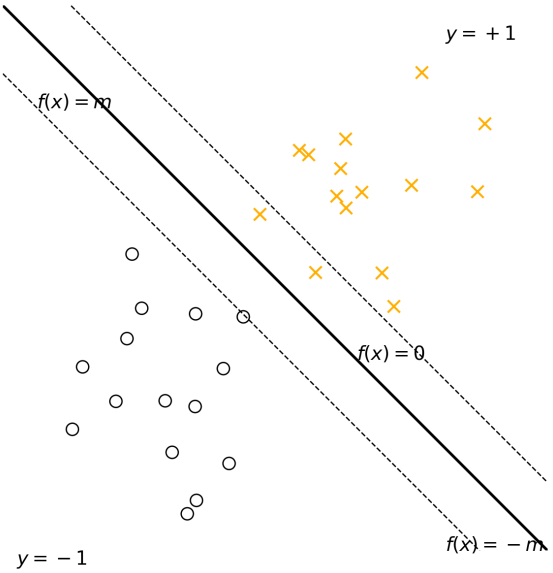


Figure 8.2

Calculation rule:

$$y = \begin{cases} 1 & \text{if } f(X) > 0 \\ -1 & \text{if } f(X) < 0 \end{cases}$$

Notes

For any points X_1, X_2 in $L : f(X) = 0$:

$$\begin{cases} B_0 + B^T X_1 = 0 \\ B_0 + B^T X_2 = 0 \end{cases} \quad \Rightarrow \quad B^T(X_1 - X_2) = 0 \quad \Rightarrow \quad B \perp L$$

Let $B^* = \frac{B}{\|B\|_2}$ be a unit vector that orthogonal to L.
The distance of any point X_i to L is

$$\frac{f(X_i)y_i}{\|B\|_2}$$

Proof. Assume $y_i > 0$, $\forall X_1 \in L$, we have

$$\begin{aligned} (B^*)^T(X_i - X_1) &= \frac{1}{\|B\|_2} B^T(X_i - X_1) \\ &= \frac{1}{\|B\|_2}(B^T X_i + B_0 - 0) \\ &= \frac{1}{\|B\|_2} f(X_i) \end{aligned}$$

□

Optimization SVM tries to find the best separating hyperplane that maximizes the distance from either class to the closest points.
The optimization problem for SVM is:

$$\max_{B_0, B} M$$

$$s.t. \quad \frac{y_i(X_i^T B + B_0)}{\|B\|_2} \geq M, \quad i = 1, \ldots, n$$

where M represents the margin.

If we set $\|B\|_2 = \frac{1}{M}$, then it becomes

$$\min_{B_0, B} \frac{1}{2}\|B\|_2^2$$

$$s.t. \quad y_i(X_i^T B + B_0) \geq 1, \quad i = 1, \ldots, n$$

To solve the problem, we set $\alpha_1, \ldots, \alpha_n$ be the Lagrange multipliers. the Lagrangian function is

$$L_p(B_0, B, \alpha) = \frac{1}{2}\|B\|_2^2 - \sum_{i=1}^{n} \alpha_i [y_i(X_i^T B + B_0) - 1]$$

Take derivatives

$$\begin{cases} \frac{dL_p}{dB_0} = -\sum_{i=1}^n \alpha_i y_i = 0 \\ \frac{dL_p}{dB} = B - \sum_{i=1}^n \alpha_i y_i X_i = 0 \end{cases}$$

Plug them into $L_p(B_0, B, \alpha)$, and we will get the new objective function:

$$L_D(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j X_i^T X_j$$

So the new optimization problem is

$$\max_{\alpha} L_D(\alpha)$$

$$s.t. \begin{cases} \alpha_i \geq 0 \\ \sum \alpha_i y_i = 0 \end{cases}$$

KKT conditions give:

- $y_i(X_i^T B + B_0) \geq 1$;
- $\alpha_i \geq 0$;
- $\alpha_i[y_i(X_i^T B + B_0) - 1] = 0$;
- $\sum \alpha_i y_i = 0, B = \sum_{i=1}^{n} \alpha_i y_i X_i$.

by KKT we have

- If $\alpha_i > 0$, $y_i(X_i^T B + B_0) = 1$, which means (X_i, y_i) are at the boundaries. They are called supported points;
- If $y_i(X_i^T B + B_0) > 1$, then $\alpha_i = 0$.

Interpretation:
the SVM classification rule is derived by a linear combination of observation on the boundary:

$$\hat{B} = \sum_{i \in S} \alpha_i y_i X_i$$

where S is the set of supported points.
Estimation of B_0:

$$\hat{B}_0 = \frac{1}{|S|} \sum_{i \in S} (y_i - \hat{B}^T X_i)$$

because $\forall i \in S, y_i(X_i^T B + B_0) = 1$ and $y_i = \pm 1$.

8.4.2 Kernel SVM

We specify a kernel $K(X_i, X_j) \to \mathcal{H}_K$, $X_i \in \mathbb{R}^p \to \phi(X_i) \in \mathcal{H}_K$.
if $K(X_i, X_j) = X_i^T X_j$, kernel SVM reduces to linear SVM.

Optimization The optimization problem for Kernel SVM is

$$\min_{f \in \mathcal{H}_K} \|f\|_{\mathcal{H}_K}$$

$$s.t. \begin{cases} y_i f(X_i) = y_i(X_i^T B + B_0) \geq 1, \quad i = 1, \ldots, n \\ f \in \mathcal{H}_K \end{cases}$$

the solution can be obtained by solving

$$\hat{\alpha} = \arg\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(X_i, X_j)$$

$$s.t.\ \alpha_i \geq 0,\ \sum_{i=1}^{n} \alpha_i y_i = 0, \quad i = 1, \ldots, n$$

8.4.3 Soft-margin SVM

Consider the case when the data is not linearly separable. We maximize the margin allowing some points to be on the wrong side.

In this situation, we use slack variables $\epsilon = \begin{pmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{pmatrix}$, and the optimization problem is

$$\min_{\epsilon, B_0, B} \frac{1}{2}\|B\|_2^2$$

$$s.t. \begin{cases} y_i(X_i^T B + B_0) \geq 1 - \epsilon_i, \quad i = 1, \ldots, n \\ \epsilon_i \geq 0 \\ \sum_{i=1}^{n} \epsilon_i \leq C \end{cases}$$

which is equivalent to

$$\min_{\epsilon, B_0, B} \frac{1}{2}\|B\|_2^2 + C \sum_{i=1}^{n} \epsilon_i$$

$$s.t. \begin{cases} y_i(X_i^T B + B_0) \geq 1 - \epsilon_i, \quad i = 1, \ldots, n \\ \epsilon_i \geq 0 \end{cases}$$

where C is a tuning parameter controlling the tradeoff between maximizing the margin and maximizing the training error.

Another form of the optimization problem is using Hinge loss:

$$\min_{B_0, B} \frac{1}{2}\|B\|_2^2 + c \sum_{i=1}^{n} L(y_i, f(X_i))$$

where $L(y_i, f(X_i)) = \max(1 - y_i f(X_i), 0)$ is Hinge loss and $f(X_i) = B_0 + B^T X_i$.

Chapter 9

Special Topic: Neural Network

9.1 Shallow Neural Network

9.1.1 Data

Suppose we have $X = \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix} \in \mathbb{R}^p$ and $y = \begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix} \in \mathbb{R}^K$. If it is used for classification, then $Y = \{1, \ldots, K\}$ and $y_k = \mathbb{I}_{(y=k)}$.

- Regression: $E(y_k|x) = f_k(x)$;
- Classification: $p(y = k|x) = \frac{\exp[f_k(x)]}{\sum_{l=1}^{K} \exp[f_l(x)]}$.

9.1.2 Model

$$f_k(X) = w_{k0}^{(2)} + (w_k^{(2)})^T Z, \quad Z = \begin{pmatrix} Z_1 \\ \vdots \\ Z_M \end{pmatrix}$$

and

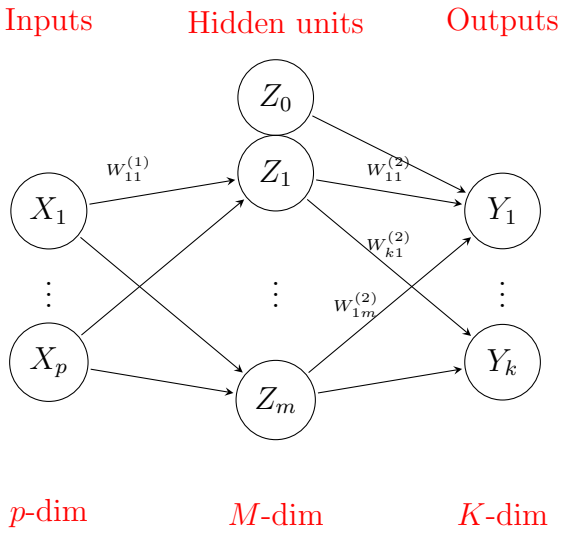
$$Z_m = \sigma(w_{m0}^{(1)} + (w_m^{(1)})^T X)$$

where w are model parameters:

$$(w_k^{(2)})_{M\times 1} = \begin{pmatrix} w_{k1}^{(2)} \\ \vdots \\ w_{kM}^{(2)} \end{pmatrix}, \quad (w_m^{(1)})_{p\times 1} = \begin{pmatrix} w_{m1}^{(1)} \\ \vdots \\ w_{mp}^{(1)} \end{pmatrix}$$

$\sigma(\cdot)$ is an activation function.

- sigmoid: $\sigma(v) = \frac{e^v}{1+e^v} = \frac{1}{1+e^{-v}}$;
- ReLU: rectified linear unit $\max(v, 0)$.



References

Chapter 11, *ESL*;
Chapter 5, *PRML*.

9.2 Deep Neural Network

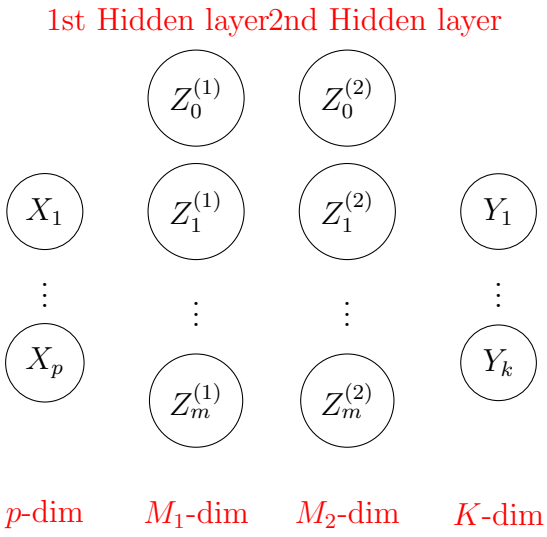
9.2.1 Multiple hidden layers

Suppose $Z_m^{(2)}$, $m = 1, \ldots, M_2$ is the 2nd hidden layer. We have

$$Z_m^{(2)} = \sigma(w_{m0}^{(2)} + (w_m^{(2)})^T Z^{(1)})$$

where $Z^{(1)}$ is the first hidden layer. Then, the output is denoted by

$$f_k(x) = w_{k0}^{(3)} + (w_k^{(3)})_{m_2 \times 1}^T Z_{m_2 \times 1}^{(2)}$$



Why DNN works Since we have a lot of layers, there will be too many parameters to be estimated.
DNN is similar to **hierarchical interaction function class**.

Examples:

1. Single index model: $f(X) = g(h(X)) = g(\theta^T X)$;
2. Additive model: $f(X) = \sum f_l(X_l)$;
3. Projection pursuit regression: $f(X) = \sum g_k(\theta_k^T X)$.

References

Kohler, M., & Krzyżak, A. (2016). Nonparametric regression based on hierarchical interaction models. *IEEE Transactions on Information Theory*, 63(3), 1620-1630.

Function class DNN is

$$f(X) = \mathcal{L}_{D+1} \circ \sigma \circ \mathcal{L}_D \circ \cdots \circ \mathcal{L}_2 \circ \sigma \circ \mathcal{L}_1$$

where $L_j(X) = A_j X + b_j$ and σ is the activation function.

9.2.2 Convergence of DNN

Let $f^* \in \mathcal{F}_{DNN}(p, D, W)$ be any function with the form above, where D and W represents depth and width; $f_0 \in H(p)$.

The approximation bias is given by

$$\Big(\int_{[0,1]^p} |f^*(X) - f_0(X)|^2 dx \Big)^{\frac{1}{2}} \leq (DW)^{-2\gamma^*}$$

where γ^* is a constant in H.

So the estimation:

$$\|\hat{f} - \hat{f}_0\|_2 \leq \sqrt{\frac{1}{n} D^2 W^2 \log(DW) \log n} + (DW)^{-2\gamma^*} + \text{other bias}$$

The first term is the estimation error (variance); the second term is the approximation bias. Here there exists bias-variance tradeoff for DW.

This shows that even the depth of a neural network is 1, as long as W is long enough, we will still attain convergence.

If we pick DW approximately,

$$\|\hat{f} - \hat{f}_0\|_2 \leq (\frac{\log \sigma(n)}{n})^{\frac{\gamma^*}{2\gamma^*+1}}$$

9.2.3 Estimator of the weights

Here we only consider shallow neural network:
The model parameters are

$$\begin{cases} (w_{m0}^{(1)}, w_m^{(1)}) = (w_{m0}^{(1)}, \ldots, w_{mp}^{(1)}), & m = 1, \ldots, M \\ (w_{k0}^{(2)}, w_k^{(2)}) = (w_{k0}^{(2)}, \ldots, w_{kM}^{(2)}), & k = 1, \ldots, K \end{cases}$$

The loss function is

- Regression: $L(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{n} [y_{ik} - f_k(X_i)]^2$;
- Classification: negative log lik $L(\theta) = -\sum_{k=1}^{K} \sum_{i=1}^{n} y_i k \log P(y_i = k|X)$.

We often use **Stochastic Gradient Descent (SGD)** to estimate the parameters:

$$L(\theta) = \sum_{i=1}^{n} L_i(\theta) \quad \text{with} \quad L_i(\theta) = \sum_{k=1}^{K} \big(y_{ik} - f_k(X_i)\big)^2$$

Compute partial derivatives:

$$\begin{aligned}\frac{\partial L_i(\theta)}{\partial w_{km}^{(2)}} &= \frac{\partial L_i(\theta)}{\partial f_k(X_i)}\frac{\partial f_k(X_i)}{\partial w_{km}^{(2)}} \\ &= \left[-2\left(y_{ik} - f_k(X_i)\right)\right] Z_{im}, \quad \text{where } Z_i = \begin{pmatrix} Z_{i1} \\ \vdots \\ Z_{im} \end{pmatrix}.\end{aligned}$$

$$\begin{aligned}\frac{\partial L_i(\theta)}{\partial w_{ml}^{(1)}} &= \sum_{k=1}^{K} \frac{\partial L_i(\theta)}{\partial f_k(X_i)}\frac{\partial f_k(X_i)}{\partial W_{ml}^{(1)}} \\ &= \sum_{k=1}^{K} \frac{\partial L_i(\theta)}{\partial f_k(X_i)}\, w_{km}^{(2)}\, \sigma'\big(w_{m0}^{(1)} + (w_m^{(1)})^T X_i\big)\, X_{il}.\end{aligned}$$

where $\sigma'(\cdot)$ is the derivative of $\sigma(\cdot)$.

Set

$$\delta_{ki} = -2\left(y_{ik} - f_k(X_i)\right)$$

and

$$S_{mi} = \sum_{k=1}^{K} \frac{\partial L_i(\theta)}{\partial f_k(X_i)}\, w_{km}^{(2)}\, \sigma'\big(w_{m0}^{(1)} + (w_m^{(1)})^T X_i\big)$$

Then

$$\begin{cases} \dfrac{\partial L_i(\theta)}{\partial w_{km}^{(2)}} = \delta_{ki}\, Z_{im}, \\ \dfrac{\partial L_i(\theta)}{\partial w_{m\ell}^{(1)}} = S_{mi}\, X_{il}. \end{cases}$$

At $(t+1)$ step,

$$\begin{aligned} w_{km}^{(2),\,t+1} &= w_{km}^{(2),\,t} - e^{(t)}\Big[\frac{\partial L_{i_t}(\theta)}{\partial w_{km}^{(2)}} - 2\lambda\, w_{km}^{(2),\,t}\Big] \\ w_{ml}^{(1),\,t+1} &= w_{ml}^{(1),\,t} - e^{(t)}\Big[\frac{\partial L_{i_t}(\theta)}{\partial w_{ml}^{(1)}} - 2\lambda\, w_{ml}^{(1),\,t}\Big] \end{aligned}$$

where i_{i_t} are chosen randomly; $-2\lambda\, w_{km}^{(2),\,t}$ and $-2\lambda\, w_{ml}^{(1),\,t}$ are the L2 penalty, and e is step size.

Part III

CLUSTERING

Chapter 10

Clustering

10.1 Introduction

GOAL To group a collection of objects $X_i,\ i = 1, \ldots, n$ into clusters.

Assumption We assume each cluster represents a subpopulation within a heterogeneous population.

Different from classification, cluster labels are unknown, which is a typical feature of unsupervised learning. The clusters $Z_i \in \{1, \ldots, K\}$ records the cluster membership of i-th observation, where K is the number of clusters.

Clustering methods

- Data-driven approach: K-means, hierarchical clustering, spectral clustering, DBSCAN;
- Model-based approach: finite mixture model, latent class model.

10.2 Data-driven Clustering

Similarity measure between X_i:

- $D(X_i, X_j) = \|X_i - X_j\|_2$: Euclidean Distance;
- $D(X_i, X_j) = \sum_{l=1}^{q} w_l d_l(X_i, X_j)$;
- ...

Usually we want to minimize the loss function

$$W(Z) = \frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i:Z_i=k}\sum_{j:Z_j=k} D^2(X_i, X_j)$$

where $Z = \begin{pmatrix} Z_1 \\ \vdots \\ Z_n \end{pmatrix}$, $n_K = \sum_{i=1}^{n} \mathbb{I}_{(Z_i=k)}$.

10.2.1 K-means

Measures We use Euclidean Distance $D(X_i, X_j) = \|X_i - X_j\|_2$.
The **within cluster sum of square error (WSS)** is given by

$$\begin{aligned} W(Z) &= \frac{1}{2}\sum_{k=1}^{K}\frac{1}{n_k}\sum_{i:Z_i=k}\sum_{j:Z_j=k}\|X_i - X_j\|_2^2 \\ &= \sum_{k=1}^{K}\sum_{i:Z_i=k}\|X_i - \mu_k\|_2^2 \end{aligned}$$

where $\sum_{i:Z_i=k}\|X_i - \mu_k\|_2^2$ describes the variance measure for observations in the k-th cluster, $\mu_k = \frac{1}{n_k}\sum_{i:Z_i=k} X_i$ is the cluster mean of cluster k.
The **between cluster sum of square error (BSS)** is given by

$$B(Z) = \sum_{k=1}^{K} n_k\|\mu_k - \bar{X}\|_2^2$$

The **total sum of square error (TSS)** is

$$T = \sum_{i=1}^{n}\|X_i - \bar{X}\|_2^2 = W(Z) + B(Z)$$

or

$$TSS = WSS + BSS$$

Optimization Our goal is to minimize within cluster $W(Z)$:

$$\min_Z W(Z) \Leftrightarrow \min_Z \sum_{k=1}^{K} \sum_{i:Z_i=k} \|X_i - \mu_k\|_2^2$$

which is not convex because of Z. Therefore, there are many local optima.
What we can do is running multiple initializations randomly and choosing the solution with the lowest $W(Z)$ value, e.g., $n = 10$ times.

Algorithm

1. Initialize means μ_k, k = 1,. . . , K;
2. Repeat until convergence:
   - For $i = 1, \ldots, n$, $Z_i = \arg\min_{1 \le k \le K} \|X_i - \hat{\mu}_k\|_2^2$;
   - For $k = 1, \ldots, K$, $\hat{\mu}_k = \frac{\sum_{i=1}^n \mathbb{I}_{(Z_i=k)} X_i}{\sum_{i=1}^n \mathbb{I}_{(Z_i=k)}}$.

Selection of number of clusters K There are many methods for selection of K. Here we introduce two main methods.

1. **Elbow method**
   We can plot $W_K(Z) - W_{K+1}(Z)$ against K.

   As K increases, $W_K(Z)$ will decrease and decrease and the rate of decrease gradually slows down. Consequently, $W_K(Z) - W_{K+1}(Z)$will also gradually decrease and converge to 0.

   We choose the first K where the decreasing trend of $W_K(Z) - W_{K+1}(Z)$ approaches 0.

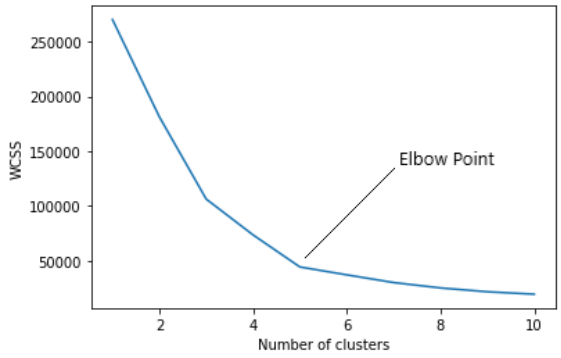


Figure 10.1

2. **Gap statistic** (proposed by Tibshirani in 2001)
   The procedure of selecting K using Gap statistic is as follows:

- For all possible K values, e.g., $K = 1, \ldots, K_{\max}$, run K-means and get all the $W_K(Z)$;
- Generate B reference datasets from the original data;
- For each reference datasets $b = 1, \ldots, B$, cluster it with K clusters and get $W_K(Z)^{b*}$;
- Then we calculate log mean of W_K^{b*}:

$$E(\log(W(Z)^*) = \frac{1}{B}\sum_{b=1}^{B} \log W_K(Z)^{b*}$$

- Calculate the Gap statistic for each K:

$$Gap(K) = E(\log W_K(Z)^*) - \log W_K(Z)$$

- Estimate the standard deviation of $E(\log W_K(Z)^*)$:

$$s'_K = s_K\sqrt{1 + \frac{1}{B}}$$

where

$$s_K = \sqrt{\frac{1}{B}\sum_{b=1}^{B}[\log(W_K^{b*}) - E(\log W_K(Z)^*)]^2}$$

- We choose the optimal K as the smallest value such that

$$Gap(K) \geq Gap(K+1) - s'_{K+1}$$

References

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the royal statistical society: series b (statistical methodology)*, 63(2), 411-423.

Non-linear K-means (Kernel K-means) Kernel function:

$$K(X_i, X_j)$$

$$X_i \xrightarrow{K} \phi(X_i) \in \mathcal{H}_K$$

and we perform K-means on $\phi(X)$.

The optimization for Kernel K-means is

$$Z = \arg\min_Z \sum_{k=1}^{K}\sum_{i:Z_i=k} \|\phi(X_i) - \mu_k\|^2_{\mathcal{H}_K}$$

which is equivalent to

$$Z = \arg\min_Z \sum_{k=1}^{K}\sum_{i=1}^{n} \mathbb{I}_{(Z_i=k)}\big[K(X_i, X_i) - \frac{2}{n_k}\sum_{j=1}^{n}\mathbb{I}_{(Z_j=k)}K(X_i, X_j)$$

$$+\frac{1}{n_k^2}\sum_{j=1}^{n}\sum_{l=1}^{n}\mathbb{I}_{(Z_j=k,Z_l=k)}K(X_j, X_l)\big]$$

10.2.2 Hierarchical Clustering

In this method, we don't need to specify the number of clusters K.

Main idea Build a hierarchical representation of clusters, reflecting different levels of closeness between data points or groups. This is done by progressively merging or splitting clusters to form a tree structure (a dendrogram).

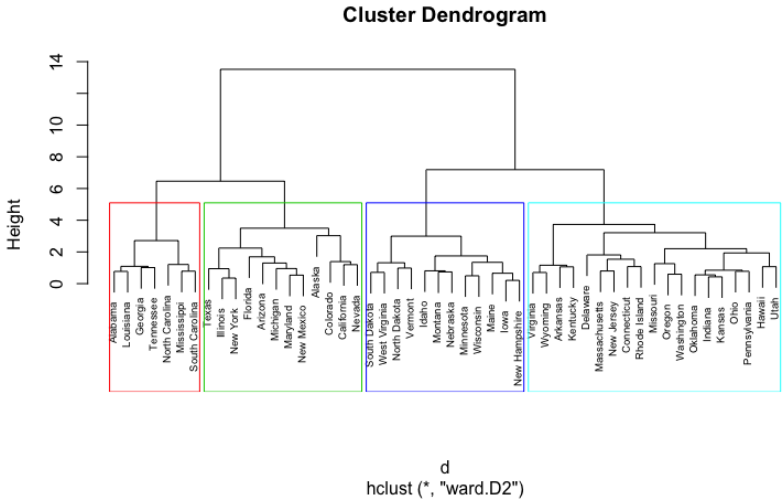


Figure 10.2

Methods There are two methods for hierarchical clustering:

- **Bottom-up method (Agglomerative method)**:
  We start with n clusters for n points, and then merge the two closest clusters at each step until only one cluster containing all data points remains;
- **Top-down method (Divisive method)**:
  Starts with all data points in a single cluster. At each step, it splits a cluster into two or more smaller clusters. This process is repeated until each data point forms its own individual cluster.

We use Bottom-up HC more often because of its lower time complexity.

Bottom up HC The optimization problem for this method is

$$\min_{\mu} \frac{1}{2} \sum_{i=1}^{n} \|X_i - \mu_i\|_2^2 + \lambda \sum_{i<j} \mathbb{I}_{(\mu_i \neq \mu_j)}$$

where the cluster mean $\mu = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_n \end{pmatrix}$.

The problem is non-convex because of $\mathbb{I}_{(\mu_i \neq \mu_j)}$. We can modify the problem to make it convex:

$$\min_{\mu} \frac{1}{2} \sum_{i=1}^{n} \|X_i - \mu_i\|_2^2 + \lambda \sum_{i<j} \|\mu_i - \mu_j\|_q$$

When $q = 1$, it is a fused LASSO penalty; when $q = 2$, it is a group fused LASSO penalty.

References

Hocking, T. D., Joulin, A., Bach, F., & Vert, J. P. (2011, June). Clusterpath an algorithm for clustering using convex fusion penalties. *In 28th international conference on machine learning* (p. 1).

10.2.3 Biclustering

This method can perform clustering on both rows and columns of a data matrix simultaneously, whereas common clustering methods typically only cluster rows by finding similar rows across all columns or only cluster columns.

Example Imagine a data table where rows represent objects (e.g., genes) and columns represent features or conditions (e.g., experimental conditions).

- Traditional row clustering finds groups of genes that behave similarly across all experimental conditions;
- Traditional column clustering finds groups of experimental conditions where all genes show similar behavior.

In contrast, biclustering finds a group of genes that exhibit a similar pattern only under a specific set of experimental conditions (for example, these genes are all highly expressed or show an increasing trend under these conditions).

Model Suppose we have data matrix

$$X: \ X_{ij} \sim \mu_{kr} + \epsilon, \quad i = 1, \ldots, n, \ j = 1, \ldots, p, \ k = 1, \ldots, K, \ r = 1, \ldots, R$$

where K, R are the number of row clusters and column clusters.
The optimization problem is

$$\min_{\mu} \frac{1}{2} \sum_{k=1}^{K} \sum_{r=1}^{R} \sum_{i \in C_k} \sum_{j \in D_r} (X_{ij} - \mu_{kr})^2$$

and we can add a sparse penalty $\lambda \sum_k \sum_r |\mu_{kr}|$ if needed.

We can convert this problem into a convex clustering problem:

$$\min_U \frac{1}{2}\|X-U\|_F^2 + \lambda_{\text{row}} \sum_{i<i\prime} w_{ii\prime}\|U_{i\cdot} - U_{i\prime\cdot}\|_2 + \lambda_{\text{col}} \sum_{j<j\prime} h_{jj\prime}\|U_{\cdot j} - U_{\cdot j\prime}\|_2.$$

Note that U is not made of μ_{kr}. Actually it has the same dimension as X. More details in the reference paper below.

References

Chi, E. C., Allen, G. I., & Baraniuk, R. G. (2017). Convex biclustering. *Biometrics*, 73(1), 10-19.

Or we can perform biclustering via sparse SVD:
Decompose $X = \sum_{k=1}^K d_k u_k v_k^T$ where u_k and v_k are sparse. The problem becomes

$$\min_{u_k, v_k} \|X - duv^T\|_F^2 + \lambda_u P_1(u) + \lambda_v P_2(v)$$

where $\lambda_u P_1(u) + \lambda_v P_2(v)$ are sparse penalty on u, v.

References

Lee, M., Shen, H., Huang, J. Z., & Marron, J. S. (2010). Biclustering via sparse singular value decomposition. *Biometrics*, 66(4), 1087-1095.

10.3 Model-based Clustering

10.3.1 Finite Mixture Model

We assume the data are from a mixture of distribution.

Model For each observable variable $X_{p\times 1}$ and latent variable $Z \in \{1, \ldots, K\}$, we have

$$\begin{cases} Z \sim Categorical(\pi_{K\times 1}) \text{ with } \pi = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_K \end{pmatrix}; \\ X|Z = k \sim P(X|\theta_k) \end{cases}$$

where θ_k are unknown parameters.

Examples

1. Continuous type data $X \in \mathbb{R}^p$:
   Example:
   Take $P(X|\theta_k)$ as $N(X|\mu_k, \Sigma_k)$ where π_k are the mixture probabilities and μ_k, Σ_k are unknown parameters, i.e.,

   $$X|Z = k \sim N(X|\mu_k, \Sigma_k)$$

   This gives Gaussian Mixture Model (**GMM**), which we have introduced in Chapter 4, and will discuss more specifically next.

2. Categorical data:
   Here we have $X_{p\times 1} = \begin{pmatrix} X_1 \\ \vdots \\ X_p \end{pmatrix}$ with $X_j \in \{1, \ldots, M\}$. $X_1, \ldots, X_p|Z = k$ are conditionally independent, and

   $$X_j|Z = k \sim Categorical(\theta_{jkm})$$

   where $\theta_{jkm} = P(X_j = m|Z = k)$, and $\sum_{m=1}^{M} \theta_{jkm} = 1$.

3. Mixed type data:
   Some of the X_j are discrete variables, while some are continuous variables.

   Example:

   $$X_i|Z = k \sim N(\mu_k, \Sigma_k), \quad i = 1, \ldots, \frac{n}{2}$$

   $$X_j|Z = k \sim Bernoulli(\theta_{jk}), \quad j = (\frac{n}{2} + 1), \ldots, n$$

Gaussian Mixture Model (GMM) Suppose we have data $X_1, \ldots, X_n \in \mathbb{R}^p$. The model is

$$\begin{cases} Z \sim Categorical(\pi_{K\times 1}) \text{ with } \pi = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_K \end{pmatrix}; \\ X|Z = k \sim N(\mu_k, \Sigma_k) \end{cases}$$

and we want to estimate the parameters π, μ_k, Σ_k. This can be done by EM algorithm:

Procedure for EM algorithm
The likelihood of observed data is

$$L(\pi, \mu, \Sigma \mid X) = \prod_{i=1}^{n} P\big(X_i \mid \pi, \mu, \Sigma\big) = \prod_{i=1}^{n} \sum_{k=1}^{K} \pi_k \, N\big(X_i \mid \mu_k, \Sigma_k\big)$$

So the log likelihood is given by

$$\ell(\pi, \mu, \Sigma \mid X) = \log L(X \mid \pi, \mu, \Sigma) = \sum_{i=1}^{n} \log\Big(\sum_{k=1}^{K} \pi_k \, N(X_i \mid \mu_k, \Sigma_k)\Big)$$

The complete-data log likelihood is given by

$$\ell_c(\pi, \mu, \Sigma \mid X, Z) = \log \prod_{i=1}^{n} \prod_{k=1}^{K} \big[\pi_k \, N(X_i \mid \mu_k, \Sigma_k)\big]^{z_i^k}$$

$$= \sum_{i=1}^{n} \sum_{k=1}^{K} z_i^k \big(\log \pi_k + \log N(X_i \mid \mu_k, \Sigma_k)\big)$$

where $z_i^k = \mathbb{I}_{(Z_i = k)}$.

E-step: we compute $E_{Z|X,\pi^{(t)},\mu^{(t)},\Sigma^{(t)}}(\ell_C(\mu, \pi, \Sigma|X, Z))$.
For $i = 1, \ldots, n$, $k = 1, \ldots, K$,

$$\tau_{ik}^{(t)} \equiv P(Z_i = k|X_i, \pi^{(t)}, \mu^{(t)}, \Sigma^{(t)}) = \frac{P(Z_i = k, X_i|\theta^{(t)})}{P(X_i|\theta^{(t)})} = \frac{\pi_k^{(t)} N\big(X_i|\mu_k^{(t)}, \Sigma_k^{(t)}\big)}{\sum_{l=1}^{K} \pi_l^{(t)} N\big(X_i|\mu_l^{(t)}, \Sigma_l^{(t)}\big)}$$

M-step: we maximize the expectation w.r.t. π, μ, Σ.
For $k = 1, \ldots, K$,

$$\begin{cases} \pi_k^{(t+1)} = \frac{1}{n} \sum_{i=1}^{n} \tau_{ik}^{(t)} \\ \mu_k^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{ik}^{(t)} X_i}{\sum_{i=1}^{n} \tau_{ik}^{(t)}} \\ \Sigma_k^{(t+1)} = \frac{\sum_{i=1}^{n} \tau_{ik}^{(t)} (X_i - \mu_k^{(t+1)})(X_i - \mu_k^{(t+1)})^T}{\sum_{i=1}^{n} \tau_{ik}^{(t)}} \end{cases}$$

Note: We need data to follow Gaussian distribution!

Remark

EM & GMM is a soft version of K-means.
If we fix $\Sigma_k = \sigma^2 I$, then when $\sigma^2 \to 0$, EM algorithm $\to$ K-means:

$$\tau_{ik} = \frac{\pi_k \exp\{-\frac{1}{2\sigma^2}\|X_i - \mu_k\|_2^2\}}{\sum_{l=1}^K \pi_l \exp\{-\frac{1}{2\sigma^2}\|X_i - \mu_l\|_2^2\}}$$

10.3.2 Bayesian Approach

Instead of computing MLE in GMM, we use Bayesian inference which is to infer the latent variable Z by computing the posterior distribution of π, μ, Σ, Z:

Given data $(X_1, \ldots, X_n)$, we compute

$$P(\pi_k|X) = \frac{P(\pi_k, X)}{P(X)} = \frac{P(X|\pi_k)P(\pi_k)}{P(X)}$$

and $P(\mu_k|X), P(\Sigma_k|X)$ similarly. We can use Gibbs sampling.

Example Consider the prior for π:

$$\pi = \begin{pmatrix} \pi_1 \\ \vdots \\ \pi_K \end{pmatrix} \sim Dirichlet(\alpha_1, \ldots, \alpha_K)$$

with PDF $f(\pi) \propto \prod_{k=1}^K \pi_k^{\alpha_k - 1}$.
The prior for μ_k and Σ_k are

- $\mu_k \mid \Sigma_k \sim \mathcal{N}(\mu_{0k}, \Sigma_k/\kappa_0)$;
- $\Sigma_k \sim$ Inverse-Wishart(ν_0, Ψ_0)

For all $k = 1, \ldots, K$, we repeat sampling $Z_i, \pi_k, \mu_k, \Sigma_k$ conditionally on other parameters sequentially. Finally we can estimate the posterior distribution from our samples.

10.3.3 Overlapping Clustering

Overlapping Clustering allows some variables to belong to more than one cluster simultaneously. One method for this purpose is using structured factor model.

Model Consider $X_{p\times 1} = A_{p\times K} Z_{K\times 1} + E_{p\times 1}$.
The loading matrix $A_{p\times K}$ characterizes the cluster membership of X. $A_{jk} \neq 0 \iff X_i$ belongs to Cluster k.

Note that A is only identifiable to signed permutation (matrix permutation or multiplying ± 1). However, if A satisfies

$$\begin{cases} \sum_{a=1}^{K} |A_{ja}| \leq 1 \\ \exists \text{ at least two j} \in \{1, \ldots, p\}, \ s.t. \ \forall b \neq a, \ |A_{ja}| = 1, \ |A_{jb}| = 0. \end{cases}$$

then A is identifiable.

More details can be found in the paper below.

References

Bing, X., Bunea, F., Ning, Y., & Wegkamp, M. (2020). Adaptive estimation in structured factor models with applications to overlapping clustering. *Annals of Statistics*, 48(4).