THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 1**

Bayesian Methods

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Bayes Decision

1. Let the conditional densities for a two-category one-dimensional problem be given by the Cauchy distribution described as follows:

$$p(x|\omega_i) = \frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_i}{b}\right)^2}, \quad i = 1, 2. \tag{1}$$

1.1. By explicit integration, check that the distributions are indeed normalized.

解: 直接积分可得, $\forall\ i = 1, 2$, 有

$$\begin{aligned} \int_{-\infty}^{\infty} p(x|\omega_i)\mathrm{d}x &= \int_{-\infty}^{\infty} \frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_i}{b}\right)^2}\mathrm{d}x \\ &= \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{1}{1 + u^2}\mathrm{d}u, \quad u = \frac{x - a_i}{b} \\ &= \frac{1}{\pi}\arctan(u)\Big|_{-\infty}^{\infty} \\ &= \frac{1}{\pi}\left(\frac{\pi}{2} + \frac{\pi}{2}\right) \\ &= 1 \end{aligned} \tag{2}$$

故此分布函数是归一化的.

1.2. Assuming $P(\omega_1) = P(\omega_2)$, show that $P(\omega_1|x) = P(\omega_2|x)$ if $x = (a_1 + a_2)/2$, i.e., the minimum error decision boundary is a point midway between the peaks of the two distributions, regardless of b.

解: 由 Bayes 公式可知

$$P(\omega_1|x) = \frac{p(x|\omega_1)P(\omega_1)}{p(x)} \tag{3}$$

$$P(\omega_2|x) = \frac{p(x|\omega_2)P(\omega_2)}{p(x)} \tag{4}$$

令 $P(\omega_1|x) = P(\omega_2|x)$ 可得

$$\frac{p(x|\omega_1)P(\omega_1)}{p(x)} = \frac{p(x|\omega_2)P(\omega_2)}{p(x)} \tag{5}$$

又 $P(\omega_1) = P(\omega_2)$, 则

$$p(x|\omega_1) = p(x|\omega_2) \tag{6}$$

即

$$\frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_1}{b}\right)^2} = \frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_2}{b}\right)^2} \tag{7}$$

整理得

$$|x - a_1| = |x - a_2| \tag{8}$$

易知 $x = (a_1 + a_2)/2$ 满足上式.

1.3. Assuming $P(\omega_1) = P(\omega_2)$, plot $P(\omega_1|x)$ for the case $a_1 = 3$, $a_2 = 5$ and $b = 1$.

解: 由 Bayes 公式可得

$$\begin{aligned} P(\omega_1|x) &= \frac{p(x|\omega_1)P(\omega_1)}{p(x)} \\ &= \frac{p(x|\omega_1)P(\omega_1)}{p(x|\omega_1)P(\omega_1) + p(x|\omega_2)P(\omega_2)} \\ &= \frac{p(x|\omega_1)}{p(x|\omega_1) + p(x|\omega_2)} \\ &= \frac{\frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_1}{b}\right)^2}}{\frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_1}{b}\right)^2} + \frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_2}{b}\right)^2}} \\ &= \frac{1 + \left(\frac{x - a_2}{b}\right)^2}{1 + \left(\frac{x - a_1}{b}\right)^2 + 1 + \left(\frac{x - a_2}{b}\right)^2} \end{aligned} \tag{9}$$

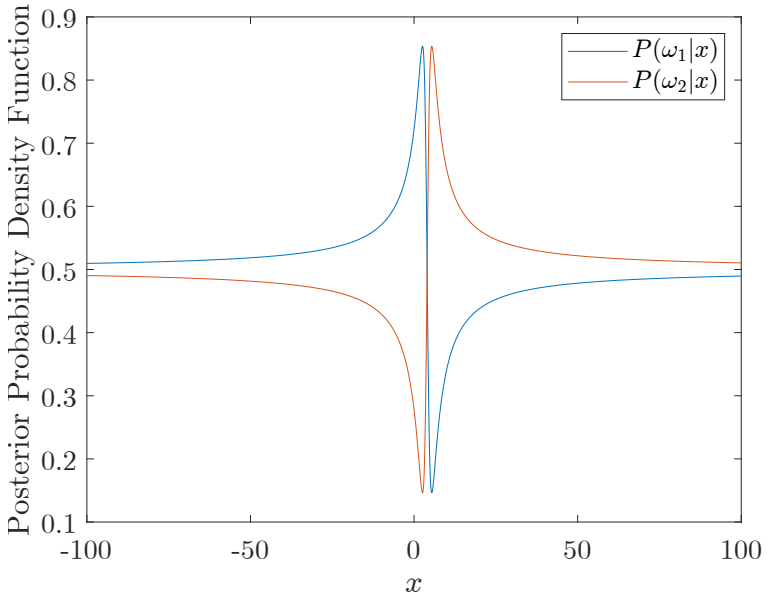


图 1: 后验概率

易知

$$P(\omega_2|x) = 1 - P(\omega_1|x) \tag{10}$$

代入已知数值可绘图如图 1 所示.

1.4. Assuming $P(\omega_1) = P(\omega_2)$, how do $P(\omega_1|x)$ and $P(\omega_2|x)$ behave as $x \to -\infty$? $x \to +\infty$? Explain.

解: 由图 1 可知, 当 $x \to \pm\infty$ 时, $P(\omega_1|x) = P(\omega_2|x) = \dfrac{1}{2}$, 证明如下.

对式 (9) 取极限可得

$$\lim_{x\to\pm\infty} P(\omega_1|x) = \lim_{x\to\pm\infty} \frac{1 + \left(\dfrac{x-a_2}{b}\right)^2}{1 + \left(\dfrac{x-a_1}{b}\right)^2 + 1 + \left(\dfrac{x-a_2}{b}\right)^2} = \frac{1}{2} \tag{11}$$

则有

$$\lim_{x\to\pm\infty} P(\omega_2|x) = \frac{1}{2} \tag{12}$$

2. Consider Neyman-Pearson criteria for two Cauchy distributions in one dimension described in Problem 1. Assume a zero-one error loss, and for simplicity $a_2 > a_1$, the same "width" b, and equal priors.

2.1. Suppose the maximum acceptable error rate for classifying a pattern that is actually in ω_1 as if it were in ω_2 is E_1. Determine the decision boundary in terms of the variables given.

解: 本题要求考虑 Neyman-Pearson 准则, 则边界并不一定是单边边界, 以下为大部分错误解法.

设边界点为 t, 由 $a_2 > a_1$ 可知

$$R_1 = (-\infty, t), \quad R_2 = (t, \infty) \tag{13}$$

又先验概率相等, 则

$$P(\omega_1) = p(\omega_2) = \frac{1}{2} \tag{14}$$

故第一类错误率

$$\begin{aligned} E_1 &= \int_t^\infty p(x|\omega_1)P(\omega_1)\mathrm{d}x \\ &= \frac{1}{2}\int_t^\infty \frac{1}{\pi b} \cdot \frac{1}{1+\left(\dfrac{x-a_1}{b}\right)^2}\mathrm{d}x \\ &= \frac{1}{2\pi}\int_{(t-a_1)/b}^\infty \frac{1}{1+u^2}\mathrm{d}u, \quad u = \frac{x-a_1}{b} \\ &= \frac{1}{2\pi}\arctan(u)\Big|_{(t-a_1)/b}^\infty \\ &= \frac{1}{2\pi}\left[\frac{\pi}{2} - \arctan\left(\frac{t-a_1}{b}\right)\right] \end{aligned} \tag{15}$$

即

$$\arctan\left(\frac{t-a_1}{b}\right) = \frac{\pi}{2} - 2\pi E_1 \tag{16}$$

两边取正切可得

$$\frac{t-a_1}{b} = \tan\left(\frac{\pi}{2} - 2\pi E_1\right) = \cot(2\pi E_1) \tag{17}$$

故

$$t = a_1 + b\cot(2\pi E_1) = a_1 + \frac{b}{\tan(2\pi E_1)} \tag{18}$$

正确解法应为, 考虑两类错误

$$\begin{aligned} P_1(e) &= \int_{\mathcal{R}_2} p(x|\omega_1)\mathrm{d}x = E_1 \\ P_2(e) &= \int_{\mathcal{R}_1} p(x|\omega_2)\mathrm{d}x \end{aligned} \tag{19}$$

最小化 $P_2(e)$ 等价于

$$\min \gamma = P_2(e) + \lambda(P_1(e) - E_1) \tag{20}$$

对 λ 和 $\mathcal{R}_2$ 边界求导等于 0, 有

$$\lambda = \frac{p(x|\omega_2)}{p(x|\omega_1)} = \frac{b^2 + (x-a_1)^2}{b^2 + (x-a_2)^2} \tag{21}$$

$$\int_{\mathcal{R}_2} p(x|\omega_1)\mathrm{d}x = E_1 \tag{22}$$

则决策边界为

$$t_{1,2} = \frac{(a_1 - \lambda a_2) \pm \sqrt{\lambda(a_1 - a_2)^2 - b^2(\lambda - 1)^2}}{1 - \lambda} \tag{23}$$

2.2. For this boundary, what is the error rate for classifying ω_2 as ω_1?

解: 第二类错误率为

$$\begin{aligned} E_2 &= \int_{-\infty}^{\top} p(x|\omega_2) P(\omega_2) \mathrm{d}x \\ &= \frac{1}{2} \int_{-\infty}^{\top} \frac{1}{\pi b} \cdot \frac{1}{1 + \left(\frac{x - a_2}{b}\right)^2} \mathrm{d}x \\ &= \frac{1}{2\pi} \int_{-\infty}^{(t-a_2)/b} \frac{1}{1 + u^2} \mathrm{d}u \\ &= \frac{1}{2\pi} \left[\arctan\left(\frac{t - a_2}{b}\right) + \frac{\pi}{2} \right] \end{aligned} \tag{24}$$

2.3. What is the overall error rate under zero-one loss?

解: 总错误率为

$$E = E_1 + E_2 = E_1 + \frac{1}{2\pi} \left[\arctan\left(\frac{t - a_2}{b}\right) + \frac{\pi}{2} \right] \tag{25}$$

2.4. Apply your results to the specific case $b = 1$ and $a_1 = -1, a_2 = 1$ and $E_1 = 0.1$.

解: 代入数值得边界点为

$$t = a_1 + \frac{b}{\tan(2\pi E_1)} = -1 + \frac{1}{\tan(2\pi \times 0.1)} \approx 0.376 \tag{26}$$

则总错误率为

$$\begin{aligned} E &= E_1 + \frac{1}{2\pi} \left[\arctan\left(\frac{t - a_2}{b}\right) + \frac{\pi}{2} \right] \\ &= 0.1 + \frac{1}{2\pi} \left[\arctan\left(\frac{0.376 - 1}{1}\right) + \frac{\pi}{2} \right] \\ &\approx 0.261 \end{aligned} \tag{27}$$

2.5. Compare your result to the Bayes error rate (i.e. without the Neyman-Pearson conditions).

解: 对于 Bayes 决策, 分界点为

$$t = \frac{a_1 + a_2}{2} = \frac{-1 + 1}{2} = 0 \tag{28}$$

故 Bayes 错误率为

$$\begin{aligned} E_B &= \int_{-\infty}^{\top} p(x|\omega_2)P(\omega_2)\mathrm{d}x + \int_{t}^{\infty} p(x|\omega_1)P(\omega_1)\mathrm{d}x \\ &= \frac{1}{2\pi}\left[\frac{\pi}{2} - \arctan\left(\frac{t-a_1}{b}\right)\right] + \frac{1}{2\pi}\left[\arctan\left(\frac{t-a_2}{b}\right) + \frac{\pi}{2}\right] \\ &= \frac{1}{2} - \frac{1}{2\pi}\arctan\left(\frac{0+1}{1}\right) + \frac{1}{2\pi}\arctan\left(\frac{0-1}{1}\right) \\ &= \frac{1}{4} \end{aligned} \tag{29}$$

由 $E_B < E$ 可知 Bayes 决策错误率小于 Neyman-Pearson 决策错误率.

3. In many pattern classification problems one has the option either to assign the pattern to one of c classes, or to *reject* it as being unrecognizable. If the cost for rejects is not too high, rejection may be a desirable action. Let

$$\lambda(\alpha_i|\omega_j) = \begin{cases} 0 & i = j, \quad i, j = 1, 2, \cdots, c \\ \lambda_r & i = c+1 \\ \lambda_s & \text{otherwise,} \end{cases} \tag{30}$$

where λ_r is the loss incurred for choosing the $(c+1)$th action, rejection, and λ_s is the loss incurred for making a substitution error. Show that the minimum risk is obtained if we decide ω_i if $P(\omega_i|\boldsymbol{x}) \geqslant P(\omega_j|\boldsymbol{x})$ for all j and if $P(\omega_i|\boldsymbol{x}) \geqslant 1 - \lambda_r/\lambda_s$, and reject otherwise. What happens if $\lambda_r = 0$? What happens if $\lambda_r > \lambda_s$?

解: 令

$$m = \underset{j=1,2,\cdots,c}{\operatorname{argmax}} P(\omega_j|x) \tag{31}$$

即 ω_m 表示后验概率最大的类别, 分类讨论.

当选择 ω_m 时, 风险为

$$R_m = \lambda_s \sum_{j \neq m} P(\omega_j|x) = \lambda_s[1 - P(\omega_m|x)] \tag{32}$$

当选择其他类别 ω_k, $k \neq m$ 时, 风险为

$$R_k = \lambda_s \sum_{j \neq k} P(\omega_j|x) = \lambda_s[1 - P(\omega_k|x)] \geqslant \lambda_s[1 - P(\omega_m|x)] \tag{33}$$

当选择拒绝时, 风险为

$$R_r = \lambda_r \tag{34}$$

所以最小风险为

$$\begin{aligned} R &= \min\{R_m, R_k, R_r\} \\ &= \min\{\lambda_s[1 - P(\omega_m|x)], \lambda_r\} \end{aligned} \tag{35}$$

当 $\lambda_s[1-P(\omega_m|x)] \leqslant \lambda_r$, 即

$$P(\omega_m|x) \geqslant 1-\frac{\lambda_r}{\lambda_s} \tag{36}$$

时, 选择 ω_m 风险最小, 否则选择拒绝风险最小.

若 $\lambda_r = 0$, 则选择拒绝风险最小, 会拒绝所有样本.

若 $\lambda_r > \lambda_s$, 则选择最大后验概率类别 ω_m 风险最小, 不会拒绝任何样本.

4. Consider a two-category classification problem in two dimensions. Given the following conditions, calculate the Bayes decision boundary respectively and plot them in the coordinate space (with computer or by hand). You should indicate the mean point of each category.

4.1. We set $p(\boldsymbol{x}|\omega_1) \sim N(\boldsymbol{0}, \boldsymbol{I})$, $p(\boldsymbol{x}|\omega_2) \sim N\left((1,1)^\top, \boldsymbol{I}\right)$ and $P(\omega_1) = P(\omega_2) = 0.5$.

解: 忽略类别无关项, 判别函数为

$$g_i(\boldsymbol{x}) = -\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i) \tag{37}$$

决策面满足

$$g_i(\boldsymbol{x}) = g_j(\boldsymbol{x}) \tag{38}$$

故

$$-\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i) = -\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^\top(\boldsymbol{x}-\boldsymbol{\mu}_j) \tag{39}$$

化简得

$$-2\boldsymbol{x}^\top\boldsymbol{\mu}_i + \boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i = -2\boldsymbol{x}^\top\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^\top\boldsymbol{\mu}_j \tag{40}$$

即

$$(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)^\top\boldsymbol{x} - \frac{1}{2}(\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^\top\boldsymbol{\mu}_j) = 0 \tag{41}$$

代入数据得

$$x_1 + x_2 - 1 = 0 \tag{42}$$

决策面如图 2 所示.

4.2. We set $p(\boldsymbol{x}|\omega_1) \sim N(\boldsymbol{0}, \boldsymbol{I})$, $p(\boldsymbol{x}|\omega_2) \sim N\left((1,1)^\top, \boldsymbol{I}\right)$ and $P(\omega_1) = 0.2, P(\omega_2) = 0.8$.

解: 忽略类别无关项, 判别函数为

$$g_i(\boldsymbol{x}) = -\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i) + \ln P(\omega_i) \tag{43}$$

决策面满足

$$g_i(\boldsymbol{x}) = g_j(\boldsymbol{x}) \tag{44}$$

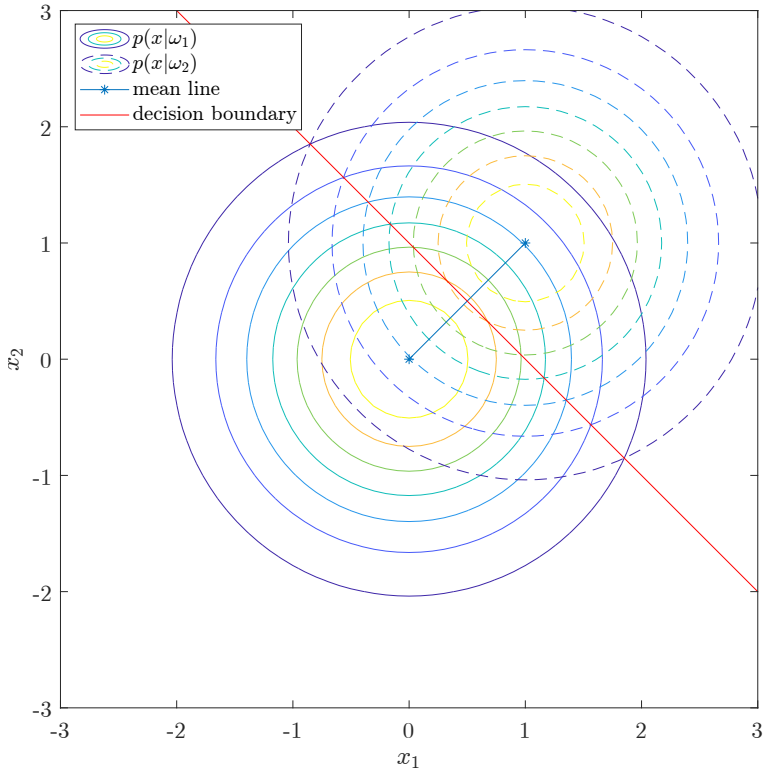

$p(x|\omega_1)$
$p(x|\omega_2)$
mean line
decision boundary
3
2
1
0
-1
-2
-3
x_2
-3
-2
-1
0
1
2
3
x_1

图 2: 决策面

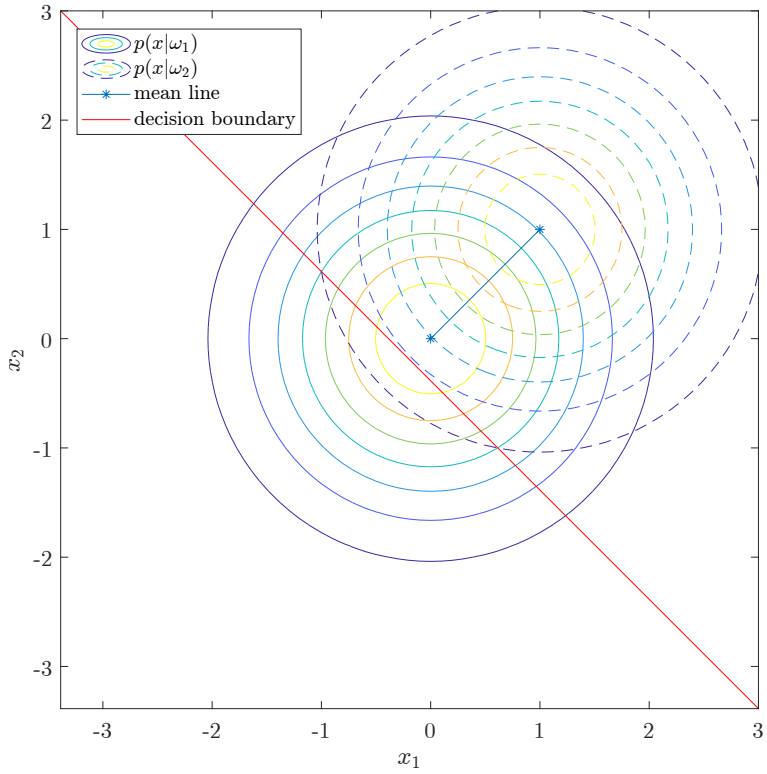


图 3: 决策面

故

$$-\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top(\boldsymbol{x}-\boldsymbol{\mu}_i)+\ln P(\omega_i)=-\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^\top(\boldsymbol{x}-\boldsymbol{\mu}_j)+\ln P(\omega_j) \tag{45}$$

化简得

$$-2\boldsymbol{x}^\top\boldsymbol{\mu}_i+\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i-2\sigma^2\ln P(\omega_i)=-2\boldsymbol{x}^\top\boldsymbol{\mu}_j+\boldsymbol{\mu}_j^\top\boldsymbol{\mu}_j-2\sigma^2\ln P(\omega_j) \tag{46}$$

即

$$(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)^\top\boldsymbol{x}-\frac{1}{2}(\boldsymbol{\mu}_i^\top\boldsymbol{\mu}_i-\boldsymbol{\mu}_j^\top\boldsymbol{\mu}_j)+\sigma^2\ln\frac{P(\omega_i)}{P(\omega_j)}=0 \tag{47}$$

代入数据得

$$x_1+x_2-1+\ln 4=0 \tag{48}$$

决策面如图 3 所示.

4.3. We set $p(\boldsymbol{x}|\omega_1)\sim N\left(\boldsymbol{0},\begin{bmatrix}1&1\\1&2\end{bmatrix}\right)$, $p(\boldsymbol{x}|\omega_2)\sim N\left((1,1)^\top,\begin{bmatrix}1&1\\1&2\end{bmatrix}\right)$ and $P(\omega_1)=P(\omega_2)=0.5$.

解: 忽略类别无关项, 判别函数为

$$g_i(\boldsymbol{x})=-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i) \tag{49}$$

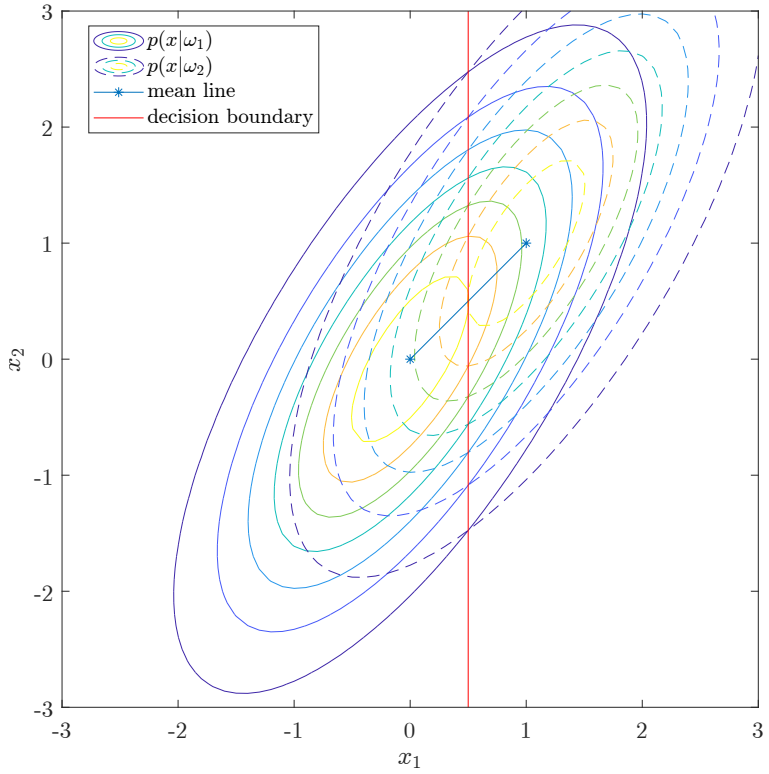


图 4: 决策面

决策面满足

$$g_i(\boldsymbol{x}) = g_j(\boldsymbol{x}) \tag{50}$$

故

$$-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^\top\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j) \tag{51}$$

化简得

$$-2\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \boldsymbol{\mu}_i^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i = -2\boldsymbol{x}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j \tag{52}$$

即

$$(\boldsymbol{\mu}_i-\boldsymbol{\mu}_j)^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}(\boldsymbol{\mu}_i^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_j) = 0 \tag{53}$$

代入数据得

$$x_1 - \frac{1}{2} = 0 \tag{54}$$

决策面如图 4 所示.

4.4. We set $p(\boldsymbol{x}|\omega_1) \sim N\left(\mathbf{0}, \begin{bmatrix}1 & 0\\ 0 & 2\end{bmatrix}\right)$, $p(\boldsymbol{x}|\omega_2) \sim N\left((1,1)^\top, \begin{bmatrix}2 & 0\\ 0 & 1\end{bmatrix}\right)$ and $P(\omega_1) = P(\omega_2) = 0.5$.

解: 忽略类别无关项, 判别函数为

$$g_i(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| \tag{55}$$

决策面满足

$$g_i(\boldsymbol{x}) = g_j(\boldsymbol{x}) \tag{56}$$

故

$$-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_i) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_j)^\top \boldsymbol{\Sigma}_j^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_j) - \frac{1}{2}\ln|\boldsymbol{\Sigma}_j| \tag{57}$$

化简得

$$\boldsymbol{x}^\top \boldsymbol{\Sigma}_i^{-1}\boldsymbol{x} - 2\boldsymbol{x}^\top \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \boldsymbol{\mu}_i^\top \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \ln|\boldsymbol{\Sigma}_i| = \boldsymbol{x}^\top \boldsymbol{\Sigma}_j^{-1}\boldsymbol{x} - 2\boldsymbol{x}^\top \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^\top \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j + \ln|\boldsymbol{\Sigma}_j| \tag{58}$$

即

$$\boldsymbol{x}^\top(\boldsymbol{\Sigma}_j^{-1} - \boldsymbol{\Sigma}_i^{-1})\boldsymbol{x} + 2(\boldsymbol{\mu}_i\boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\mu}_j\boldsymbol{\Sigma}_j^{-1})^\top \boldsymbol{x} - (\boldsymbol{\mu}_i^\top \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i - \boldsymbol{\mu}_j^\top \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j) - (\ln|\boldsymbol{\Sigma}_i| - \ln|\boldsymbol{\Sigma}_j|) = 0 \tag{59}$$

代入数据得

$$\frac{1}{2}x_1^2 - \frac{1}{2}x_2^2 + x_1 + 2x_2 - \frac{3}{2} = 0 \tag{60}$$

化简得

$$x_1^2 - x_2^2 + 2x_1 + 4x_2 - 3 = 0 \tag{61}$$

即

$$(x_1+1)^2 - (x_2-2)^2 = 0 \tag{62}$$

即

$$|x_1+1| = |x_2-2| \tag{63}$$

决策面如图 5 所示.

Naive Bayes

5. Consider a simple learning problem of determining whether Alice and Bob from CA will go to hiking or not, $Y : Hike \in \{T, F\}$ given the weather conditions $X_1 : Sunny \in \{T, F\}$ and $X_2 : Windy \in \{T, F\}$ by a Naive Bayes classifier. Using training data, we estimated the parameters $P(Hike) = 0.5$, $P(Sunny|Hike) = 0.9$, $P(Windy|\neg Hike) = 0.8$, $P(Windy|Hike) = 0.3$ and $P(Sunny|\neg Hike) = 0.4$. Assume that the true distributions of X_1, X_2, and Y satisfy the Naive Bayes assumption of conditional independence with the above parameters.

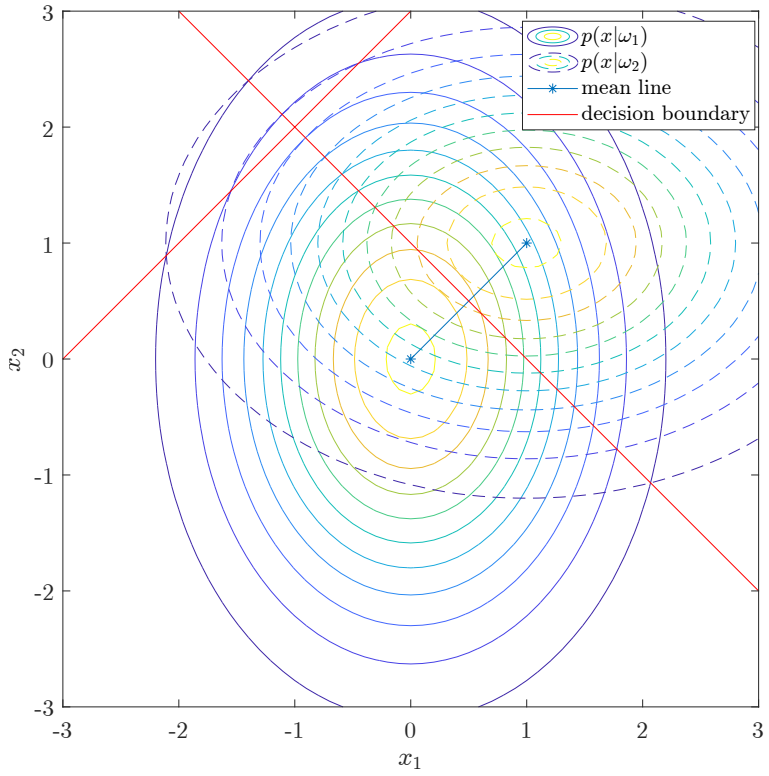


图 5: 决策面

5.1. Assume Sunny and Windy are truly independent given Hike. Write down the Naive Bayes decision rule for this problem using both attributes Sunny and Windy.

解: 已知 $P(Hike) = 0.5$, 则 $P(\neg Hike) = P(Hike) = 0.5$.

由 Bayes 公式结合条件独立的假设可得

$$\begin{aligned} P(Hike|Sunny, Windy) &= \frac{0.9 \times 0.3}{0.9 \times 0.3 + 0.4 \times 0.8} = 0.4576 \\ P(Hike|Sunny, \neg Windy) &= \frac{0.9 \times 0.7}{0.9 \times 0.7 + 0.4 \times 0.2} = 0.8873 \\ P(Hike|\neg Sunny, Windy) &= \frac{0.1 \times 0.3}{0.1 \times 0.3 + 0.6 \times 0.8} = 0.0588 \\ P(Hike|\neg Sunny, \neg Windy) &= \frac{0.1 \times 0.7}{0.1 \times 0.7 + 0.6 \times 0.2} = 0.3684 \end{aligned} \tag{64}$$

所以

$$\begin{aligned} P(Hike|Sunny, Windy) &< P(\neg Hike|Sunny, Windy) \\ P(Hike|Sunny, \neg Windy) &> P(\neg Hike|Sunny, \neg Windy) \\ P(Hike|\neg Sunny, Windy) &< P(\neg Hike|\neg Sunny, Windy) \\ P(Hike|\neg Sunny, \neg Windy) &< P(\neg Hike|\neg Sunny, \neg Windy) \end{aligned} \tag{65}$$

因此 Bayes 决策规则为

$$\begin{aligned}(Sunny, Windy) &\to \neg Hike \\ (Sunny, \neg Windy) &\to Hike \\ (\neg Sunny, Windy) &\to \neg Hike \\ (\neg Sunny, \neg Windy) &\to \neg Hike\end{aligned} \tag{66}$$

5.2. Given the decision rule above, what is the expected error rate of the Naive Bayes classifier? (The expected error rate is the probability that each class generates an observation where the decision rule is incorrect.)

解: 由 Bayes 决策规则可知, 错误率为

$$\begin{aligned}E &= P(Hike)[1 - P(Sunny, \neg Windy|Hike)] + P(\neg Hike)P(Sunny, \neg Windy|\neg Hike) \\ &= 0.5 \times (1 - 0.9 \times 0.7) + 0.5 \times 0.4 \times 0.2 \\ &= 0.225\end{aligned} \tag{67}$$

5.3. What is the joint probability that Alice and Bob go to hiking and the weather is sunny and windy, that is $P(Sunny, Windy, Hike)$?

解: 由条件乘法公式与条件独立性可知, 联合概率密度为

$$\begin{aligned}P(Sunny, Windy, Hike) &= P(Hike)P(Sunny, Windy|Hike) \\ &= P(Hike)P(Sunny|Hike)P(Windy|Hike) \\ &= 0.5 \times 0.9 \times 0.3 \\ &= 0.135\end{aligned} \tag{68}$$

5.4. Next, suppose that we gather more information about weather conditions and introduce a new feature denoting whether the weather is X_3: Rainy or not. Assume that each day the weather in CA can be either Rainy or Sunny. That is, it can not be both Sunny and Rainy (similarly, it can not be $\neg Sunny$ and $\neg Rainy$). In the above new case, are any of the Naive Bayes assumptions violated? Why (not)? What is the joint probability that Alice and Bob go to hiking and the weather is sunny, windy and not rainy, that is $P(Sunny, Windy, \neg Rainy, Hike)$?

解: 确有 Naive Bayes 的假设不成立, 事实上

$$P(Sunny, Windy, Rainy|Hike) = 0 \tag{69}$$

而

$$\begin{aligned}&P(Sunny|Hike)P(Windy|Hike)P(Rainy|Hike) \\ =\ &P(Sunny|Hike)P(Windy|Hike)P(\neg Sunny|Hike) \\ =\ &0.9 \times 0.3 \times 0.1 \\ =\ &0.027\end{aligned} \tag{70}$$

所以

$$P(Sunny, Windy, Rainy|Hike) \neq P(Sunny|Hike)P(Windy|Hike)P(Rainy|Hike) \tag{71}$$

故 Naive Bayes 的条件独立性假设不成立.

由题干假设以及条件乘法公式可知, 联合概率密度为

$$\begin{aligned} P(Sunny, Windy, \neg Rainy, Hike) &= P(Sunny, Windy, Hike) \\ &= P(Hike)P(Sunny, Windy|Hike) \\ &= P(Hike)P(Sunny|Hike)P(Windy|Hike) \\ &= 0.5 \times 0.9 \times 0.3 \\ &= 0.135 \end{aligned} \tag{72}$$

5.5. What is the expected error rate when the Naive Bayes classier uses all three attributes? Will the performance of Naive Bayes be improved by observing the new attribute Rainy? Explain why.

解: 考虑三种因素的 Bayes 决策规则为

$$\begin{aligned} (Sunny, Windy, \neg Rainy) &\rightarrow \neg Hike \\ (Sunny, \neg Windy, \neg Rainy) &\rightarrow Hike \\ (\neg Sunny, Windy, Rainy) &\rightarrow \neg Hike \\ (\neg Sunny, \neg Windy, Rainy) &\rightarrow \neg Hike \end{aligned} \tag{73}$$

错误率为

$$\begin{aligned} \tilde{E} &= P(Hike)[1 - P(Sunny, \neg Windy, \neg Rainy|Hike)] + P(\neg Hike)P(Sunny, \neg Windy, \neg Rainy|\neg Hike) \\ &= P(Hike)[1 - P(Sunny, \neg Windy|Hike)] + P(\neg Hike)P(Sunny, \neg Windy|\neg Hike) \\ &= 0.5 \times (1 - 0.9 \times 0.7) + 0.5 \times 0.4 \times 0.2 \\ &= 0.225 \end{aligned} \tag{74}$$

由 $\tilde{E} = E$ 可知, 增加新要素 Rainy 并不能改进 Naive Bayes 的性能, 因为 Rainy 信息可以直接从 Sunny 信息获得, 本质上并未真正增加新的信息.

ROC Analysis

6. The purpose of this assignment is to learn the theory and practical applications of ROC analysis. Therefore you should first read a paper about ROC curves [1] that is available in this homework zip (especially pay attention to section 1-5 and 7-8). In the following task you will work on a given theoretical example of two

表 1: 分类器 1

Class	Score	FP Rate	TP Rate
1	0.6	0	1/4
0	0.51	1/4	1/4
1	0.5	1/4	2/4
1	0.4	1/4	3/4
0	0.33	2/4	3/4
0	0.3	3/4	3/4
1	0.22	3/4	1
0	0.2	1	1

表 2: 分类器 2

Class	Score	FP Rate	TP Rate
0	0.8	1/4	0
1	0.68	1/4	1/4
0	0.53	2/4	1/4
1	0.4	2/4	2/4
1	0.22	2/4	3/4
0	0.11	3/4	3/4
0	0.1	1	3/4
1	0.04	1	1

classifier (C_1 and C_2). We have a set of samples that we wish to classify in one of two classes and a ground truth class of each sample (denoted as 0 and 1). For each sample a classifier gives a score based on which we can determine to which class should be the sample belong to (score closer to 0 means class 0, score closer to 1 means class 1). Below are the results for 8 samples, their ground truth values (S_{id}) and the score values for both classifiers (S_{C_1} and S_{C_2}).

$$\begin{aligned} S_{\text{id}} &= [1 \quad 0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0] \\ S_{C_1} &= [0.5 \quad 0.3 \quad 0.6 \quad 0.22 \quad 0.4 \quad 0.51 \quad 0.2 \quad 0.33] \\ S_{C_2} &= [0.04 \quad 0.1 \quad 0.68 \quad 0.22 \quad 0.4 \quad 0.11 \quad 0.8 \quad 0.53] \end{aligned} \tag{75}$$

For the example above calculate (by hand) and draw the ROC curves for classifier C_1 as well as classifier C_2. Also calculate the area under the curve (AUC) for both classifier. For the classifier C_1 select a decision threshold (working points) $\theta_1 = 0.33$. For the classifier C_2 select a decision threshold $\theta_2 = 0.1$. Based on theory from [1] decide which classifier is better in the selected working points and motivate your decision. Which working point is optimal for each classifier?

解: 分别对两个分类器的分数进行排序, 计算相应的真阳性率和假阳性率, 分类器 1 的计算结果如表 1 所示, 分类器 2 的计算结果如表 2 所示.

根据计算结果, 绘制两个分类器的 ROC 曲线分别如图 6 和图 7 所示.

由图 6 和图 7 可得两个分类器的 AUC 分别为

$$\begin{aligned} \text{AUC}_1 &= 1 - 5 \times \frac{1}{4} \times \frac{1}{4} = \frac{11}{16} \\ \text{AUC}_2 &= 7 \times \frac{1}{4} \times \frac{1}{4} = \frac{7}{16} \end{aligned} \tag{76}$$

由 $\theta_1 = 0.33$ 和 $\theta_2 = 0.1$ 可知分类器 1 和分类器 2 的工作点分别为 (2/4, 3/4) 和 (1, 3/4), 两个工作点的真阳

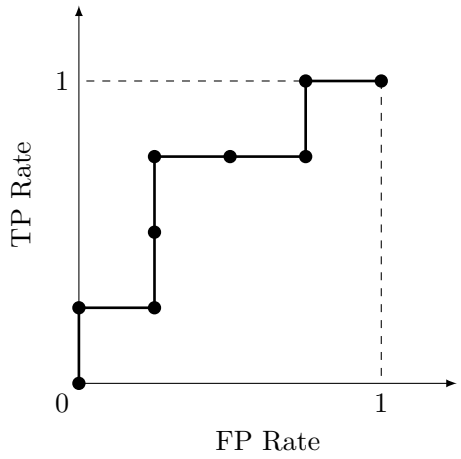


图 6: 分类器 1 ROC 曲线

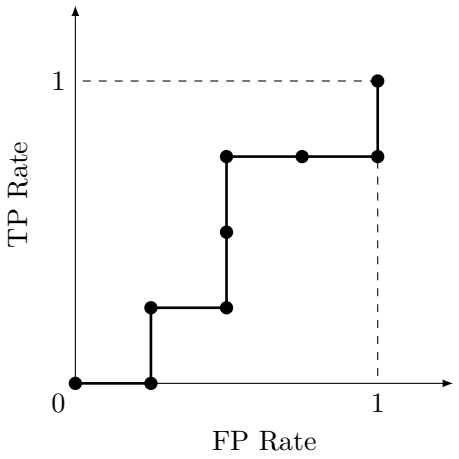


图 7: 分类器 2 ROC 曲线

性率相等, 但是分类器 1 的假阳性率更小, 故在选定工作点处分类器 1 更好.

越靠近 ROC 曲线左上角的工作点越好, 故对分类器 1 来说, 工作点 (1/4, 3/4) 最好, 相应的阈值为 $\theta_1 = 0.4$; 对分类器 2 来说, 工作点 (2/4, 3/4) 最好, 相应的阈值为 $\theta_2 = 0.22$.

参考文献

[1] T. Fawcett. An introduction to ROC analysis. Pattern Recognition Letters, 27(8):861-874, 2006.

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 2**

Density Estimation

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Maximum likelihood and Bayesian parameter estimation

1. Let x have an exponential density

$$p(x|\theta) = \begin{cases} \theta e^{-\theta x}, & x \geqslant 0 \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

1.1 Plot $p(x|\theta)$ versus x for $\theta = 1$. Plot $p(x|\theta)$ versus θ, $(0 \leqslant \theta \leqslant 5)$ for $x = 2$.

解: 分别如图 1 和图 2 所示.

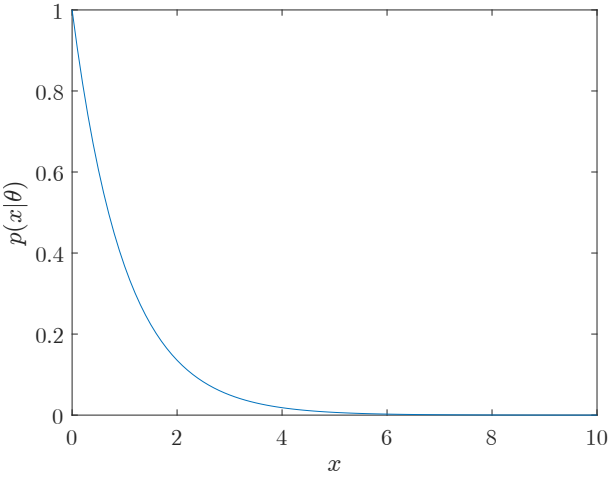


图 1: $p(x|\theta)$ 关于 x 曲线

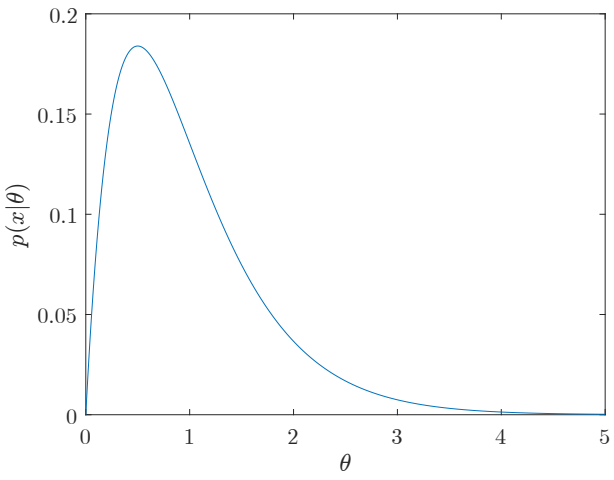


图 2: $p(x|\theta)$ 关于 θ 曲线

1.2 Suppose that n samples $x_1, \cdots, x_n$ are drawn independently according to $p(x|\theta)$. Calculate the maximum likelihood estimate for θ.

解: θ 的最大似然估计应该是下面方程的解

$$\nabla_\theta H(\theta) = \sum_{k=1}^{n} \nabla_\theta \ln p(x_k|\theta) = 0 \tag{2}$$

由指数分布可知

$$\ln p(x_k|\theta) = \ln\theta - \theta x_k, \quad \forall\, k = 1, 2, \cdots, n \tag{3}$$

其梯度为

$$\nabla_\theta \ln p(x_k|\theta) = \frac{1}{\theta} - x_k, \quad \forall\, k = 1, 2, \cdots, n \tag{4}$$

因此 θ 的最大似然估计满足

$$\sum_{k=1}^{n} \left(\frac{1}{\hat{\theta}} - x_k \right) = 0 \tag{5}$$

故 θ 的最大似然估计为

$$\hat{\theta} = \frac{1}{\frac{1}{n}\sum_{k=1}^{n} x_k} \tag{6}$$

2. The purpose of this problem is to derive the Bayesian classifier for the d-dimensional multivariate Bernoulli case. Let x be a d-dimensional binary (0 or 1) vector with a multivariate Bernoulli distribution.

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{d} \theta_i^{x_i} \left(1 - \theta_i\right)^{1 - x_i}, \tag{7}$$

where $\boldsymbol{\theta} = (\theta_1, \cdots, \theta_d)^T$ is an unknown parameter vector, θ_i being the probablility that $x_i = 1$. Let $\mathcal{D}$ be a set of n samples $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$ independently drawn according to $p(\boldsymbol{x}|\boldsymbol{\theta})$. Denote $P\left(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n|\boldsymbol{\theta}\right)$ as $P(\mathcal{D}|\boldsymbol{\theta})$.

2.1 Calculate the maximum likelihood estimate for $\boldsymbol{\theta}$.

解: $\boldsymbol{\theta}$ 的最大似然估计应该是下面方程的解

$$\nabla_{\boldsymbol{\theta}} H(\boldsymbol{\theta}) = \sum_{k=1}^{n} \nabla_{\boldsymbol{\theta}} \ln p(\boldsymbol{x}_k|\boldsymbol{\theta}) = \mathbf{0} \tag{8}$$

由分布函数可知

$$\ln p(\boldsymbol{x}_k|\boldsymbol{\theta}) = \sum_{i=1}^{d} \left[x_{k,i} \ln\theta_i + (1 - x_{k,i}) \ln(1 - \theta_i) \right], \quad \forall\, k = 1, 2, \cdots, n \tag{9}$$

其对 θ_i 的梯度为

$$\nabla_{\theta_i} \ln p(\boldsymbol{x}_k|\boldsymbol{\theta}) = \frac{x_{k,i}}{\theta_i} - \frac{1 - x_{k,i}}{1 - \theta_i}, \quad \forall\, k = 1, 2, \cdots, n,\ \forall\, i = 1, 2, \cdots, d \tag{10}$$

因此

$$\sum_{k=1}^{n}\left(\frac{x_{k,i}}{\theta_i}-\frac{1-x_{k,i}}{1-\theta_i}\right)=0,\quad \forall\, i=1,2,\cdots,d \tag{11}$$

故 θ_i 的最大似然估计满足

$$(1-\hat{\theta}_i)\sum_{k=1}^{n}x_{k,i}=\hat{\theta}_i\sum_{k=1}^{n}(1-x_{k,i})=\hat{\theta}_i\left(n-\sum_{k=1}^{n}x_{k,i}\right),\quad \forall\, i=1,2,\cdots,d \tag{12}$$

化简得

$$\hat{\theta}_i=\frac{1}{n}\sum_{k=1}^{n}x_{k,i},\quad \forall\, i=1,2,\cdots,d \tag{13}$$

写成向量形式为

$$\hat{\boldsymbol{\theta}}=\frac{1}{n}\sum_{k=1}^{n}\boldsymbol{x}_k \tag{14}$$

2.2 Assuming a uniform a priori distribution for $\boldsymbol{\theta}$, $0\leqslant\theta_i\leqslant 1$, and using the identity

$$\int_0^1\theta^m(1-\theta)^n\mathrm{d}\theta=\frac{m!n!}{(m+n+1)!}, \tag{15}$$

calculate the probablility $p(\boldsymbol{\theta}|\mathcal{D})$.

解: 令 $\boldsymbol{s}$ 表示 n 个样本的和, 即

$$\boldsymbol{s}=\sum_{k=1}^{n}\boldsymbol{x}_k \tag{16}$$

则 $P(\mathcal{D}|\boldsymbol{\theta})$ 可以计算为

$$\begin{aligned}P(\mathcal{D}|\boldsymbol{\theta})&=\prod_{k=1}^{n}p(\boldsymbol{x}_k|\boldsymbol{\theta})\\&=\prod_{k=1}^{n}\prod_{i=1}^{d}\theta_i^{x_{k,i}}\left(1-\theta_i\right)^{1-x_{k,i}}\\&=\prod_{i=1}^{d}\theta_i^{s_i}\left(1-\theta_i\right)^{n-s_i}\end{aligned} \tag{17}$$

所以 $p(\mathcal{D})$ 为

$$\begin{aligned}p(\mathcal{D})&=\int_{[0,1]^d}P(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})\mathrm{d}\boldsymbol{\theta}\\&=\int_0^1\int_0^1\cdots\int_0^1\prod_{i=1}^{d}\theta_i^{s_i}\left(1-\theta_i\right)^{n-s_i}\mathrm{d}\theta_1\mathrm{d}\theta_2\cdots\mathrm{d}\theta_d\\&=\prod_{i=1}^{d}\frac{s_i!(n-s_i)!}{(n+1)!}\end{aligned} \tag{18}$$

所以由 Bayes 公式可得概率 $p(\boldsymbol{\theta}|\mathcal{D})$ 为

$$\begin{aligned} p(\boldsymbol{\theta}|\mathcal{D}) &= \frac{p(\boldsymbol{\theta},\mathcal{D})}{p(\mathcal{D})} \\ &= \frac{P(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \\ &= \prod_{i=1}^{d} \frac{(n+1)!}{s_i!(n-s_i)!}\theta_i^{s_i}\left(1-\theta_i\right)^{n-s_i} \end{aligned} \tag{19}$$

2.3 Integrate the product $p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})$ over $\boldsymbol{\theta}$ to obtain the desired probability $p(\boldsymbol{x}|\mathcal{D})$.

解: 注意到 $x_i \in \{0,1\}$ 则有

$$\begin{aligned} p(\boldsymbol{x}|\mathcal{D}) &= \int_{[0,1]^d} p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\mathrm{d}\boldsymbol{\theta} \\ &= \int_{[0,1]^d} \prod_{i=1}^{d} \theta_i^{x_i}(1-\theta_i)^{1-x_i} \prod_{i=1}^{d} \frac{(n+1)!}{s_i!(n-s_i)!}\theta_i^{s_i}(1-\theta_i)^{n-s_i}\mathrm{d}\boldsymbol{\theta} \\ &= \int_0^1 \int_0^1 \cdots \int_0^1 \prod_{i=1}^{d} \frac{(n+1)!}{s_i!(n-s_i)!}\theta_i^{s_i+x_i}(1-\theta_i)^{n+1-s_i-x_i}\mathrm{d}\theta_1\mathrm{d}\theta_2\cdots\mathrm{d}\theta_d \\ &= \prod_{i=1}^{d} \frac{(n+1)!}{s_i!(n-s_i)!}\frac{(s_i+x_i)!(n+1-s_i-x_i)!}{(n+2)!} \\ &= \prod_{i=1}^{d} \frac{1}{n+2}\frac{(s_i+x_i)!}{s_i!}\frac{(n+1-s_i-x_i)!}{(n-s_i)!} \\ &= \prod_{i=1}^{d} \frac{1}{n+2}(s_i+x_i)^{x_i}(n+1-s_i-x_i)^{1-x_i}, \quad x_i \in \{0,1\} \\ &= \prod_{i=1}^{d} \frac{1}{n+2}(s_i+1)^{x_i}(n+1-s_i)^{1-x_i}, \quad x_i \in \{0,1\} \\ &= \prod_{i=1}^{d} \left(\frac{s_i+1}{n+2}\right)^{x_i}\left(1-\frac{s_i+1}{n+2}\right)^{1-x_i} \end{aligned} \tag{20}$$

2.4 If we think of obtaining $p(\boldsymbol{x}|\mathcal{D})$ by substituting an estimate $\hat{\boldsymbol{\theta}}$ for $\boldsymbol{\theta}$ in $p(\boldsymbol{x}|\boldsymbol{\theta})$, what is the effective Bayesian estimate for $\boldsymbol{\theta}$?

解: 已知

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{d} \theta_i^{x_i}\left(1-\theta_i\right)^{1-x_i} \tag{21}$$

且由上题有

$$p(\boldsymbol{x}|\mathcal{D}) = \prod_{i=1}^{d} \left(\frac{s_i+1}{n+2}\right)^{x_i}\left(1-\frac{s_i+1}{n+2}\right)^{1-x_i} \tag{22}$$

根据以上两式, 不难看出参数 $\boldsymbol{\theta}$ 的 Bayes 估计为

$$\hat{\boldsymbol{\theta}}_B = \frac{\boldsymbol{s}+1}{n+2} = \frac{1}{n+2}\left(\sum_{k=1}^{n} x_k + 1\right) \tag{23}$$

2.5 When do maximum likelihood estimation and Bayesian estimation methods differ. (Describe in your own words, not limited to the above examples.)

解: 当训练样本数无穷多的时候, 最大似然估计与 Bayes 估计的结果是一样的, 否则, 他们的结果是不同的.

3. Prove the invariance property of maximum likelihood estimators, i.e., that if $\hat{\theta}$ is the maximum likelihood estimate of θ, then for any differentiable function $\tau(\cdot)$, the maximum likelihood estimate of $\tau(\theta)$ is $\tau(\hat{\theta})$.

证明: 对 $\tau(\theta)$, 文献 [1] 基于似然函数 $L(\theta|x)$ 定义导出似然函数 (induced likelihood function) L^* 为

$$L^*(\eta|x) = \sup_{\{\theta:\tau(\theta)=\eta\}} L(\theta|x) \tag{24}$$

令 $\hat{\eta}$ 表示使得导出似然函数取到最大值的变量, 即

$$\hat{\eta} = \operatorname*{argmax}_{\eta} L^*(\eta|x) \tag{25}$$

所以由导出似然函数的定义和最大似然估计的定义可得

$$\begin{aligned} L^*(\hat{\eta}|x) &= \sup_{\eta} \sup_{\{\theta:\tau(\theta)=\eta\}} L(\theta|x) \\ &= \sup_{\theta} L(\theta|x) \\ &= L(\hat{\theta}|x) \end{aligned} \tag{26}$$

又由 $\hat{\theta}$ 是 θ 的最大似然估计可得

$$\begin{aligned} L(\hat{\theta}|x) &= \sup_{\{\theta:\tau(\theta)=\tau(\hat{\theta})\}} L(\theta|x) \\ &= L^*[\tau(\hat{\theta})|x] \end{aligned} \tag{27}$$

则

$$L^*(\hat{\eta}|x) = L^*[\tau(\hat{\theta})|x] \tag{28}$$

因此 $\tau(\hat{\theta})$ 是 $\tau(\theta)$ 的最大似然估计.

Nonparametric density estimation

4. Consider a normal $p(x) \sim N(\mu, \sigma^2)$ and Parzen-window function $\varphi(x) \sim N(0,1)$. Show that the Parzen-window estimate

$$p_n(x) = \frac{1}{nh_n} \sum_{i=1}^{n} \varphi\left(\frac{x - x_i}{h_n}\right) \tag{29}$$

has the following properties for small h_n:

4.1 $\bar{p}_n(x) \sim N(\mu, \sigma^2 + h_n^2)$.

解: 令

$$\theta^2 \triangleq \frac{1}{\frac{1}{h_n^2} + \frac{1}{\sigma^2}} = \frac{h_n^2 \sigma^2}{h_n^2 + \sigma^2} \tag{30}$$

且

$$\alpha \triangleq \theta^2 \left(\frac{x}{h_n^2} + \frac{\mu}{\sigma^2}\right) \tag{31}$$

则期望值为

$$\begin{aligned}
\bar{p}_n(x) &= \mathbb{E}[p_n(x)] \\
&= \frac{1}{nh_n} \sum_{i=1}^{n} \mathbb{E}\left[\varphi\left(\frac{x - x_i}{h_n}\right)\right] \\
&= \frac{1}{h_n} \int_{-\infty}^{\infty} \varphi\left(\frac{x - v}{h_n}\right) p(v) \mathrm{d}v \\
&= \frac{1}{h_n} \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left[-\frac{1}{2}\left(\frac{x - v}{h_n}\right)^2\right] \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{1}{2}\left(\frac{v - \mu}{\sigma}\right)^2\right] \mathrm{d}v \\
&= \frac{1}{2\pi h_n \sigma} \exp\left[-\frac{1}{2}\left(\frac{x^2}{h_n^2} + \frac{\mu^2}{\sigma^2}\right)\right] \int_{-\infty}^{\infty} \exp\left\{-\frac{1}{2}\left[v^2\left(\frac{1}{h_n^2} + \frac{1}{\sigma^2}\right) - 2v\left(\frac{x}{h_n^2} + \frac{\mu}{\sigma^2}\right)\right]\right\} \mathrm{d}v \\
&= \frac{1}{2\pi h_n \sigma} \exp\left[-\frac{1}{2}\left(\frac{x^2}{h_n^2} + \frac{\mu^2}{\sigma^2}\right) + \frac{1}{2}\frac{\alpha^2}{\theta^2}\right] \int_{-\infty}^{\infty} \exp\left[-\frac{1}{2}\left(\frac{v - \alpha}{\theta}\right)^2\right] \mathrm{d}v \\
&= \frac{\sqrt{2\pi}\theta}{2\pi h_n \sigma} \exp\left[-\frac{1}{2}\left(\frac{x^2}{h_n^2} + \frac{\mu^2}{\sigma^2} - \frac{\alpha^2}{\theta^2}\right)\right] \\
&= \frac{1}{\sqrt{2\pi} h_n \sigma} \frac{h_n \sigma}{\sqrt{h_n^2 + \sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{x^2\sigma^2 + h_n^2\mu^2}{h_n^2\sigma^2} - \frac{h_n^2\sigma^2}{h_n^2 + \sigma^2}\frac{(x\sigma^2 + h_n^2\mu)^2}{(h_n^2\sigma^2)^2}\right)\right] \\
&= \frac{1}{\sqrt{2\pi}\sqrt{h_n^2 + \sigma^2}} \exp\left[-\frac{1}{2}\frac{(x^2\sigma^2 + h_n^2\mu^2)(h_n^2 + \sigma^2) - (x\sigma^2 + h_n^2\mu)^2}{h_n^2\sigma^2(h_n^2 + \sigma^2)}\right] \\
&= \frac{1}{\sqrt{2\pi}\sqrt{h_n^2 + \sigma^2}} \exp\left[-\frac{1}{2}\frac{x^2\sigma^2 h_n^2 + \mu^2 h_n^2 \sigma^2 - 2x\mu h_n^2 \sigma^2}{h_n^2\sigma^2(h_n^2 + \sigma^2)}\right] \\
&= \frac{1}{\sqrt{2\pi}\sqrt{h_n^2 + \sigma^2}} \exp\left[-\frac{1}{2}\left(\frac{x - \mu}{\sqrt{h_n^2 + \sigma^2}}\right)^2\right]
\end{aligned} \tag{32}$$

因此

$$\bar{p}_n(x) \sim N(\mu, \sigma^2 + h_n^2) \tag{33}$$

4.2 $\mathrm{Var}[p_n(x)] \simeq \dfrac{1}{2nh_n\sqrt{\pi}}p(x)$.

解: 由 4.1 可知, 当 h_n 充分小时, 有

$$\bar{p}_n(x) \sim N(\mu, \sigma^2) \tag{34}$$

即有

$$\frac{1}{h_n}\mathbb{E}\left[\varphi\left(\frac{x-v}{h_n}\right)\right] \simeq p(x) \tag{35}$$

则当 h_n 充分小时, 利用上述结论以及 $x_1, x_2, \cdots, x_n$ 的独立性可得方差为

$$\begin{aligned}
\mathrm{Var}[p_n(x)] &= \mathrm{Var}\left[\frac{1}{nh_n}\sum_{i=1}^{n}\varphi\left(\frac{x-x_i}{h_n}\right)\right] \\
&= \frac{1}{n^2h_n^2}\sum_{i=1}^{n}\mathrm{Var}\left[\varphi\left(\frac{x-x_i}{h_n}\right)\right] \\
&= \frac{1}{nh_n^2}\mathrm{Var}\left[\varphi\left(\frac{x-v}{h_n}\right)\right] \\
&= \frac{1}{nh_n^2}\left\{\mathbb{E}\left[\varphi^2\left(\frac{x-v}{h_n}\right)\right] - \mathbb{E}^2\left[\varphi\left(\frac{x-v}{h_n}\right)\right]\right\} \\
&= \frac{1}{nh_n^2}\left\{\frac{1}{\sqrt{2\pi}}\mathbb{E}\left[\varphi\left(\frac{x-v}{h_n/\sqrt{2}}\right)\right] - \mathbb{E}^2\left[\varphi\left(\frac{x-v}{h_n}\right)\right]\right\} \\
&= \frac{1}{2nh_n\sqrt{\pi}}\frac{1}{h_n/\sqrt{2}}\mathbb{E}\left[\varphi\left(\frac{x-v}{h_n/\sqrt{2}}\right)\right] - \frac{1}{n}\left\{\frac{1}{h_n}\mathbb{E}\left[\varphi\left(\frac{x-v}{h_n}\right)\right]\right\}^2 \\
&\simeq \frac{p(x)}{2nh_n\sqrt{\pi}}\left[1 - 2\sqrt{\pi}h_np(x)\right] \\
&\simeq \frac{p(x)}{2nh_n\sqrt{\pi}}
\end{aligned} \tag{36}$$

4.3 $p(x) - \bar{p}_n(x) \simeq \dfrac{1}{2}\left(\dfrac{h_n}{\sigma}\right)^2\left[1 - \left(\dfrac{x-\mu}{\sigma}\right)^2\right]p(x)$.

解: 偏差为

$$\begin{aligned}
p(x) - \bar{p}_n(x) &= p(x)\left[1 - \frac{\bar{p}_n(x)}{p(x)}\right] \\
&= p(x)\left\{1 - \frac{\sigma}{\sqrt{\sigma^2 + h_n^2}}\exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2 + h_n^2} + \frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right]\right\} \\
&= p(x)\left\{1 - \frac{1}{\sqrt{1 + (h_n/\sigma)^2}}\exp\left[\frac{1}{2}\frac{h_n^2}{\sigma^2}\frac{(x-\mu)^2}{\sigma^2 + h_n^2}\right]\right\}
\end{aligned} \tag{37}$$

当 h_n 充分小时, 有

$$\frac{1}{\sqrt{1+(h_n/\sigma)^2}} \simeq 1 - \frac{1}{2}\frac{h_n^2}{\sigma^2} \tag{38}$$

和

$$\exp\left[\frac{1}{2}\frac{h_n^2}{\sigma^2}\frac{(x-\mu)^2}{\sigma^2+h_n^2}\right] \simeq 1 + \frac{1}{2}\frac{h_n^2}{\sigma^2}\frac{(x-\mu)^2}{\sigma^2+h_n^2} \tag{39}$$

所以

$$\begin{aligned} p(x) - \bar{p}_n(x) &\simeq p(x)\left[1 - \left(1 - \frac{1}{2}\frac{h_n^2}{\sigma^2}\right)\left(1 + \frac{1}{2}\frac{h_n^2}{\sigma^2}\frac{(x-\mu)^2}{\sigma^2+h_n^2}\right)\right] \\ &\simeq p(x)\left[1 - 1 + \frac{1}{2}\frac{h_n^2}{\sigma^2} - \frac{1}{2}\frac{h_n^2}{\sigma^2}\frac{(x-\mu)^2}{\sigma^2+h_n^2}\right] \\ &= \frac{1}{2}\frac{h_n^2}{\sigma^2}p(x)\left[1 - \frac{(x-\mu)^2}{\sigma^2+h_n^2}\right] \\ &\simeq \frac{1}{2}\left(\frac{h_n}{\sigma}\right)^2\left[1 - \left(\frac{x-\mu}{\sigma}\right)^2\right]p(x) \end{aligned} \tag{40}$$

5. Consider classifiers based on samples from the distributions

$$p\left(x|\omega_1\right) = \begin{cases} 3/2, & \text{for } 0 \leqslant x \leqslant 2/3 \\ 0, & \text{otherwise} \end{cases} \tag{41}$$

and

$$p\left(x|\omega_2\right) = \begin{cases} 3/2, & \text{for } 1/3 \leqslant x \leqslant 1 \\ 0, & \text{otherwise.} \end{cases} \tag{42}$$

5.1 What is the Bayes decision rule and the Bayes classification error?

解: 假定 $P(\omega_1) = P(\omega_2)$, 则边界点

$$t \in \left[\frac{1}{3},\ \frac{2}{3}\right] \tag{43}$$

取定 t, 则 Bayes 决策为

$$\begin{aligned} x \in (0,t) &\to x \in \omega_1 \\ x \in (t,1) &\to x \in \omega_2 \end{aligned} \tag{44}$$

Bayes 决策错误率为

$$\begin{aligned} P_B(e) &= P(\omega_1)\int_t^1 p(x|\omega_1)\mathrm{d}x + P(\omega_2)\int_0^t p(x|\omega_2)\mathrm{d}x \\ &= \frac{1}{2}\int_t^{2/3}\frac{3}{2}\mathrm{d}x + \frac{1}{2}\int_{1/3}^t\frac{3}{2}\mathrm{d}x \\ &= \frac{1}{2}\times\frac{3}{2}\times\frac{1}{3} \\ &= \frac{1}{4} \end{aligned} \tag{45}$$

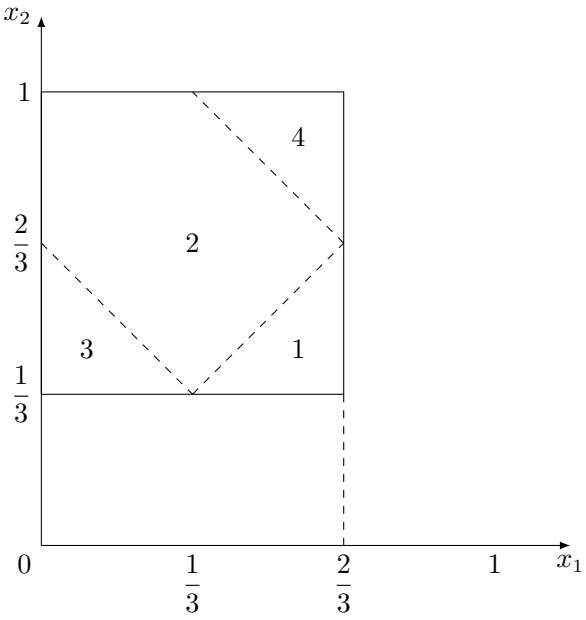


图 3: 从两个类别中取出的采样点构成的区域

5.2 Suppose we randomly select a single point from ω_1 and a single point from ω_2, and create a nearest-neighbor classifier. Suppose too we select a test point from one of the categories (ω_1 for definiteness). Integrate to find the expected error rate $P_1(e)$.

解: 设从 ω_1 中取出的点为 x_1, 从 ω_2 中取出的点为 x_2, 则最近邻法的分类边界点为

$$t = \frac{x_1 + x_2}{2} \tag{46}$$

x_1 与 x_2 的取值范围构成的区域如图 3 所示, 根据 x_1 与 x_2 之间的大小关系和边界点 t 所处的不同区间可以分成4个子区域, 下面分类讨论.

在区域1中, $x_1 \geqslant x_2$, 且 $t \in [1/3, 2/3]$, 所以该区域的错误率可以计算为

$$P_1(e_1) = \frac{1}{8}\left(\frac{1}{2}\int_0^t \frac{3}{2}\mathrm{d}x + \frac{1}{2}\int_t^1 \frac{3}{2}\mathrm{d}x\right) = \frac{3}{32} \tag{47}$$

在区域2中, $x_1 \leqslant x_2$, 且 $t \in [1/3, 2/3]$, 所以该区域的错误率可以计算为

$$P_1(e_2) = \frac{5}{8}\left(\frac{1}{2}\int_t^{2/3} \frac{3}{2}\mathrm{d}x + \frac{1}{2}\int_{1/3}^t \frac{3}{2}\mathrm{d}x\right) = \frac{5}{32} \tag{48}$$

在区域3中, $x_1 \leqslant x_2$, 且 $t \in [0, 1/3]$, 所以该区域的错误率可以计算为

$$P_1(e_3) = \frac{9}{4}\int_0^{1/3}\int_{1/3}^{2/3-x_1}\left(\frac{1}{2}\int_t^{2/3}\frac{3}{2}\mathrm{d}x\right)\mathrm{d}x_2\,\mathrm{d}x_1 = \frac{7}{192} \tag{49}$$

在区域4中, $x_1 \leqslant x_2$, 且 $t \in [2/3, 1]$, 所以该区域的错误率可以计算为

$$P_1(e_4) = \frac{9}{4}\int_{1/3}^{2/3}\int_{4/3-x_1}^{1}\left(\frac{1}{2}\int_{1/3}^{t}\frac{3}{2}\mathrm{d}x\right)\mathrm{d}x_2\,\mathrm{d}x_1 = \frac{7}{192} \tag{50}$$

故总错误率为

$$P_1(e) = \sum_{k=1}^{4} P_1(e_k) = \frac{3}{32} + \frac{5}{32} + \frac{7}{192} + \frac{7}{192} = \frac{31}{96} \approx 0.3329 \tag{51}$$

5.3 With the limit of the training sample number $n \to \infty$, compare the expected error $P_n(e)$ with the Bayes error.

解: 当 $n \to \infty$ 时, 只有位于区间 [1/3, 2/3] 之内的测试点会被错分, 且错分概率为 0.5, 则错误率为

$$\begin{aligned} P_\infty(e) &\triangleq \lim_{n\to\infty} P_n(e) \\ &= \frac{1}{2}P(\omega_1)P\left(\frac{1}{3} \leqslant x \leqslant \frac{2}{3}\middle|\omega_1\right) + \frac{1}{2}P(\omega_2)P\left(\frac{1}{3} \leqslant x \leqslant \frac{2}{3}\middle|\omega_2\right) \\ &= \frac{1}{2}\times\frac{1}{2}\times\frac{1}{2} + \frac{1}{2}\times\frac{1}{2}\times\frac{1}{2} \\ &= \frac{1}{4} \end{aligned} \tag{52}$$

因此 $P_\infty(e) = P_B(e)$, 即误差率相等.

5.4 What are the factors that affect the classification error rate?

解: 在本题中, 影响误差率的因素主要是两类条件概率密度的重合区间长度, 重合区间越长, 则错误率越大.

Programming: Parzen Window

6. Assume $p(x) \sim 0.2\,\mathcal{N}(-1, 1) + 0.8\,\mathcal{N}(1, 1)$. Draw n samples from $p(x)$, for example,

$$n = 5, 10, 50, 100, \cdots, 1000, \cdots, 10000. \tag{53}$$

Use Parzen-window method to estimate $p_n(x) \approx p(x)$. (Hint: use randn() function in matlab to draw samples.)

(a) Try window-function

$$K(x) = \begin{cases} \dfrac{1}{a}, & -\dfrac{a}{2} \leqslant x \leqslant \dfrac{a}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{54}$$

Estimate $p(x)$ with different window width a. Please draw $p_n(x)$ under different n and a and $p(x)$ to show the estimation effect.

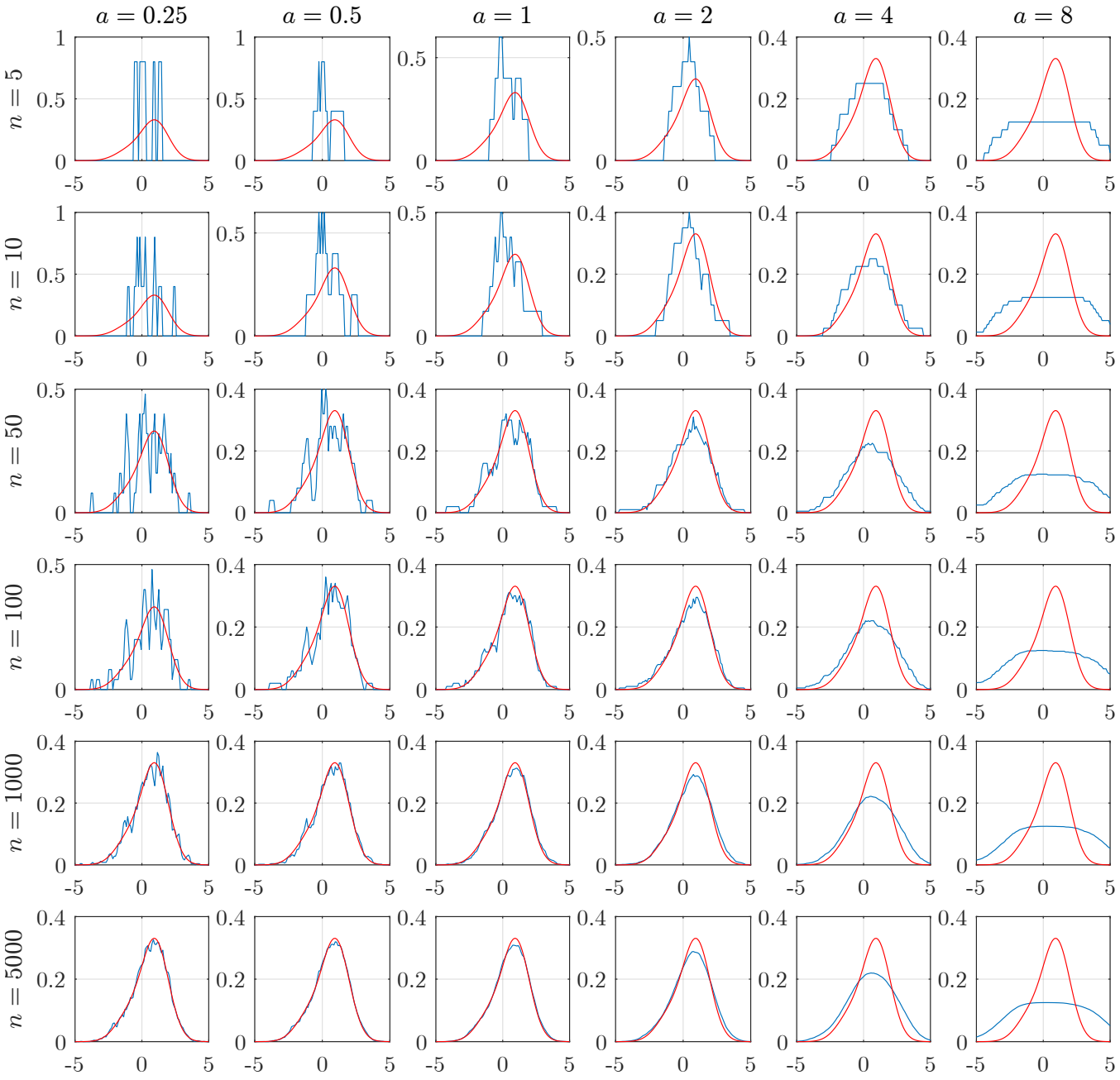


图 4: 方窗估计

解: 取 $n = 5, 10, 50, 100, 1000, 5000$, $a = 0.25, 0.5, 1, 2, 4, 8$, 使用方窗函数进行估计, 结果如图 4 所示, 图中红色曲线表示 $p(x)$. 由图可知, 固定 a 时, 随着 n 的增大, 估计效果逐渐变好; 固定 n 时, 随着 a 的增大, 估计效果先变好后变差, 可知存在适当的 a 使得估计效果达到最好.

(b) Derive how to compute

$$\epsilon(p_n) = \int \left[p_n(x) - p(x)\right]^2 \mathrm{d}x \tag{55}$$

numerically.

解: 取积分区间为 [-5, 5], 将区间 N 等分, 记区间长度为 $\Delta\xi$, ξ_j 为第 j 个区间内一点, 当 N 足够大时, 有

$$\begin{aligned}
\epsilon(p_n) &= \int \left[p_n(x) - p(x)\right]^2 \mathrm{d}x \\
&\simeq \sum_{j=1}^{N} \left[p_n(\xi_j) - p(\xi_j)\right]^2 \Delta\xi \\
&= \sum_{j=1}^{N} \left\{ \frac{1}{n} \sum_{i=1}^{n} K(\xi_j - x_i) - \frac{0.2}{\sqrt{2\pi}} \exp\left[-\frac{(\xi_j + 1)^2}{2}\right] - \frac{0.8}{\sqrt{2\pi}} \exp\left[-\frac{(\xi_j - 1)^2}{2}\right] \right\}^2 \Delta\xi
\end{aligned} \tag{56}$$

(c) Demonstrate the expectation and variance of $\epsilon(p_n)$ w.r.t different n and a.

解: 为了数值计算 $\epsilon(p_n)$ 的期望和方差, 对每一组给定的 (n, a), 进行多次重复试验, 而后对多次试验得到的 $\epsilon(p_n)$ 数组计算均值和方差, 结果分别如表 1 和表 2 所示.

	$a = 0.25$	$a = 0.5$	$a = 1$	$a = 2$	$a = 4$	$a = 8$
$n = 5$	0.0754	0.0344	0.0153	0.0051	0.0038	0.0086
$n = 10$	0.0377	0.0200	0.0088	0.0035	0.0025	0.0084
$n = 50$	0.0079	0.0038	0.0017	0.0009	0.0025	0.0083
$n = 100$	0.0040	0.0016	0.0009	0.0004	0.0019	0.0083
$n = 1000$	0.0003	0.0001	0.0001	0.0002	0.0020	0.0083
$n = 5000$	0.0001	0.0000	0.0000	0.0002	0.0020	0.0083

表 1: 方窗估计均方误差期望

(d) With n given, how to choose optimal a from above the empirical experiences?

解: 由以上经验, 对于给定的 n, 可以通过 $\epsilon(p_n)$ 的均值和方差来选取适当的 a 值. 固定 n 时, 随着 a 的增大, 估计效果先变好后变差, 所以 a 在适当的取值下才能最好地逼近真实概率密度函数, 因此可以将 $\epsilon(p_n)$ 的均值与 a 的函数关系绘制出来, 寻找其最小值所在区间, 并在该区间内选择 $\epsilon(p_n)$ 最小方差对应的 a.

(e) Substitute $K(x)$ in (a) with Gaussian window. Repeat (a)-(d).

解: 取 $n = 5, 10, 50, 100, 1000, 5000$, $a = 0.25, 0.5, 1, 2, 4, 8$, 使用高斯核函数

$$K(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{x^2}{2\sigma^2}\right] \tag{57}$$

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	0	5.2849e-05	5.0611e-05	1.1692e-05	4.3810e-06	1.6058e-07
$n=10$	8.6365e-05	5.2020e-05	2.4856e-05	1.0281e-05	5.3322e-07	9.2295e-08
$n=50$	5.5701e-06	3.1296e-06	9.1607e-07	5.1668e-07	3.5719e-07	2.0359e-08
$n=100$	1.4699e-06	4.4399e-07	1.3575e-07	4.8224e-08	7.9064e-08	5.5102e-09
$n=1000$	6.1293e-09	3.7063e-09	2.0755e-09	5.9962e-09	4.8550e-09	6.2758e-10
$n=5000$	0	4.8334e-41	4.8334e-41	6.9602e-39	0	3.1676e-36

表 2: 方窗估计均方误差方差

进行估计, 其中取 $\sigma = a/\sqrt{n}$, 结果如图 5 所示, 图中红色曲线表示 $p(x)$. 对每一组给定的 (n, a), 进行多次重复试验, 而后对多次试验得到的 $\epsilon(p_n)$ 数组计算均值和方差, 结果分别如表 3 和表 4 所示.

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	0.0440	0.0186	0.0088	0.0055	0.0041	0.0095
$n=10$	0.0330	0.0168	0.0062	0.0022	0.0025	0.0065
$n=50$	0.0147	0.0077	0.0031	0.0017	0.0007	0.0015
$n=100$	0.0110	0.0052	0.0024	0.0011	0.0005	0.0007
$n=1000$	0.0035	0.0019	0.0009	0.0004	0.0002	0.0001
$n=5000$	0.0011	0.0007	0.0003	0.0002	0.0001	0.0000

表 3: 高斯估计均方误差期望

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	8.8794e-05	3.5598e-05	3.2554e-05	5.0078e-05	2.1877e-06	1.0487e-07
$n=10$	6.2303e-05	3.7369e-05	6.7287e-06	2.0091e-06	1.9361e-06	1.2545e-07
$n=50$	7.7749e-06	9.8888e-06	1.4582e-06	8.6869e-07	1.8728e-07	1.7556e-07
$n=100$	9.5059e-06	2.0828e-06	5.2320e-07	3.1134e-07	9.8478e-08	1.1429e-07
$n=1000$	5.2851e-07	9.1313e-08	5.3496e-08	1.0761e-08	5.6957e-09	1.6131e-09
$n=5000$	6.6818e-38	7.0530e-38	1.6240e-38	7.4242e-39	5.7711e-39	5.6068e-39

表 4: 高斯估计均方误差方差

(f) Try different window functions and parameters as many as you can. Which window function/parameter is the best one? Demonstrate it numerically.

解: 取 $n = 5, 10, 50, 100, 1000, 5000$, $a = 0.25, 0.5, 1, 2, 4, 8$, 使用三角形核函数

$$K(x) = \begin{cases} \frac{1}{a}\left(1 - \left|\frac{x}{a}\right|\right), & -a \leqslant x \leqslant a \\ 0, & \text{otherwise.} \end{cases} \tag{58}$$

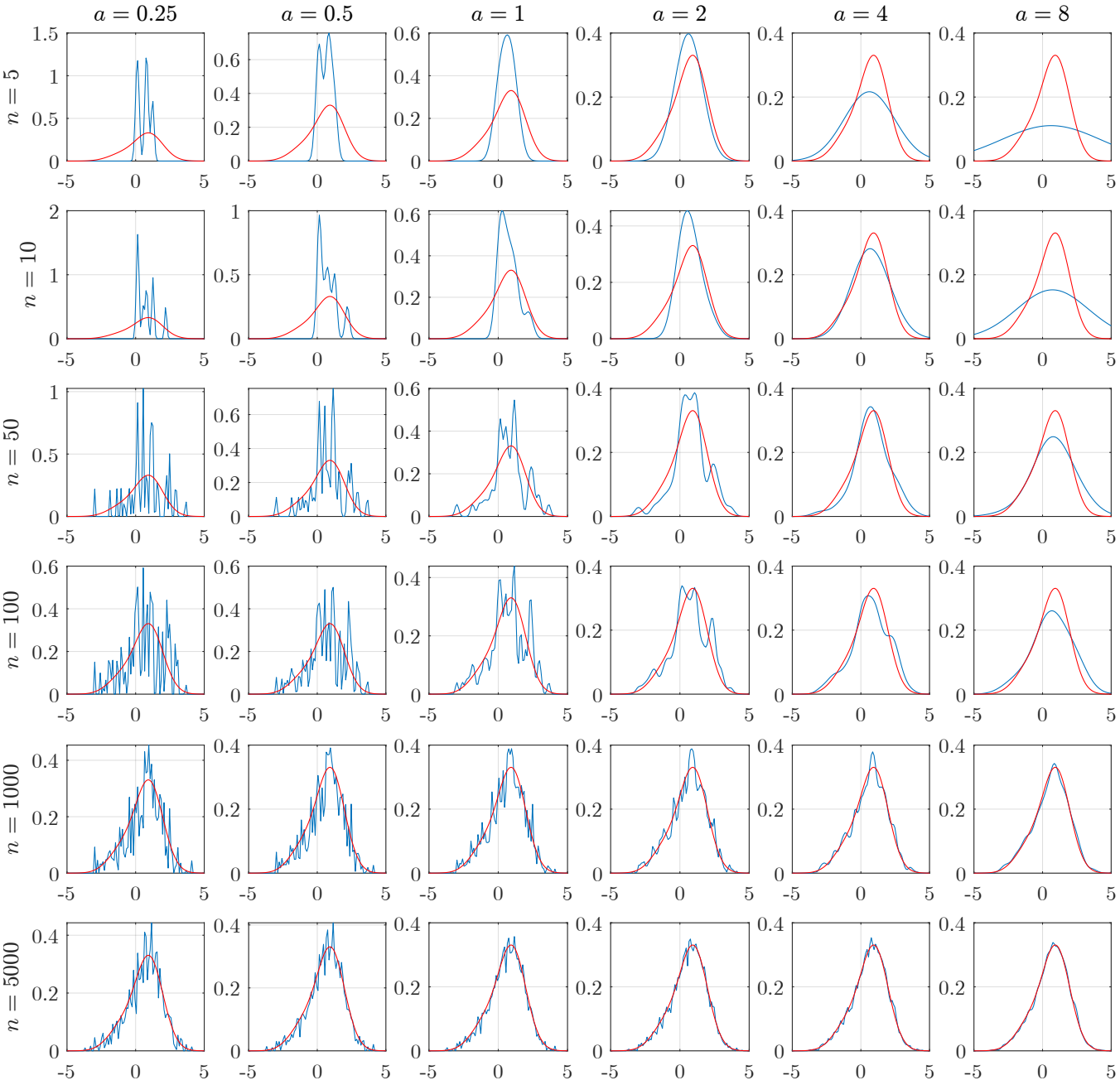


图 5: 高斯估计

进行估计, 结果如图 6 所示, 图中红色曲线表示 $p(x)$. 对每一组给定的 (n, a), 进行多次重复试验, 而后对多次试验得到的 $\epsilon(p_n)$ 数组计算均值和方差, 结果分别如表 5 和表 6 所示.

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	0.0456	0.0231	0.0085	0.0038	0.0045	0.0092
$n=10$	0.0250	0.0113	0.0049	0.0021	0.0038	0.0094
$n=50$	0.0050	0.0024	0.0009	0.0011	0.0037	0.0092
$n=100$	0.0020	0.0011	0.0005	0.0008	0.0038	0.0092
$n=1000$	0.0002	0.0001	0.0001	0.0006	0.0036	0.0092
$n=5000$	0.0000	0.0000	0.0001	0.0006	0.0036	0.0092

表 5: 三角估计均方误差期望

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	7.1407e-05	1.3686e-04	2.7454e-05	9.2369e-06	1.8634e-06	1.6729e-07
$n=10$	4.9860e-05	2.2354e-05	7.8718e-06	4.1982e-06	6.1847e-07	1.9638e-07
$n=50$	3.1465e-06	1.1115e-06	2.8525e-07	3.0190e-07	1.1315e-07	1.1064e-08
$n=100$	3.4641e-07	1.4459e-07	1.6658e-07	7.9556e-08	7.8658e-08	1.0195e-08
$n=1000$	3.8268e-09	2.1874e-09	1.6549e-09	4.0677e-09	6.4841e-09	8.1923e-10
$n=5000$	1.3256e-39	8.1878e-40	3.6360e-38	1.1693e-37	1.0988e-36	7.7607e-36

表 6: 三角估计均方误差方差

取 $n=5,10,50,100,1000,5000$, $a=0.25,0.5,1,2,4,8$, 使用余弦核函数

$$K(x)=\begin{cases}\dfrac{\pi}{4a}\cos\left(\dfrac{\pi x}{2a}\right), & -a\leqslant x\leqslant a\\ 0, & \text{otherwise.}\end{cases}\tag{59}$$

进行估计, 结果如图 7 所示, 图中红色曲线表示 $p(x)$. 对每一组给定的 (n, a), 进行多次重复试验, 而后对多次试验得到的 $\epsilon(p_n)$ 数组计算均值和方差, 结果分别如表 7 和表 8 所示.

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	0.0433	0.0195	0.0068	0.0045	0.0049	0.0106
$n=10$	0.0227	0.0104	0.0032	0.0022	0.0044	0.0106
$n=50$	0.0047	0.0021	0.0006	0.0009	0.0044	0.0106
$n=100$	0.0024	0.0009	0.0005	0.0009	0.0045	0.0106
$n=1000$	0.0002	0.0001	0.0001	0.0008	0.0044	0.0106
$n=5000$	0.0000	0.0000	0.0001	0.0008	0.0044	0.0106

表 7: 余弦估计均方误差期望

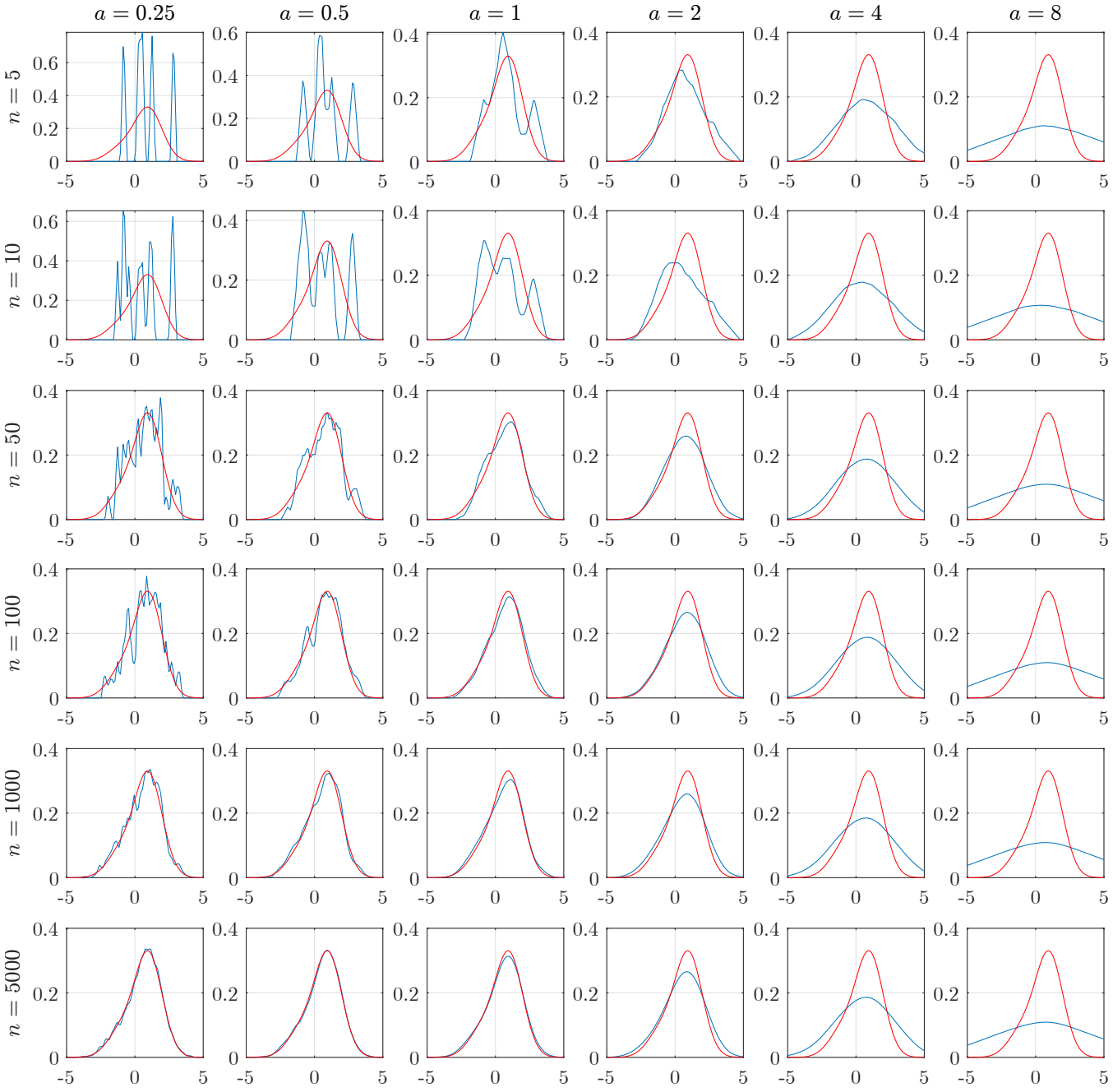


图 6: 三角形估计

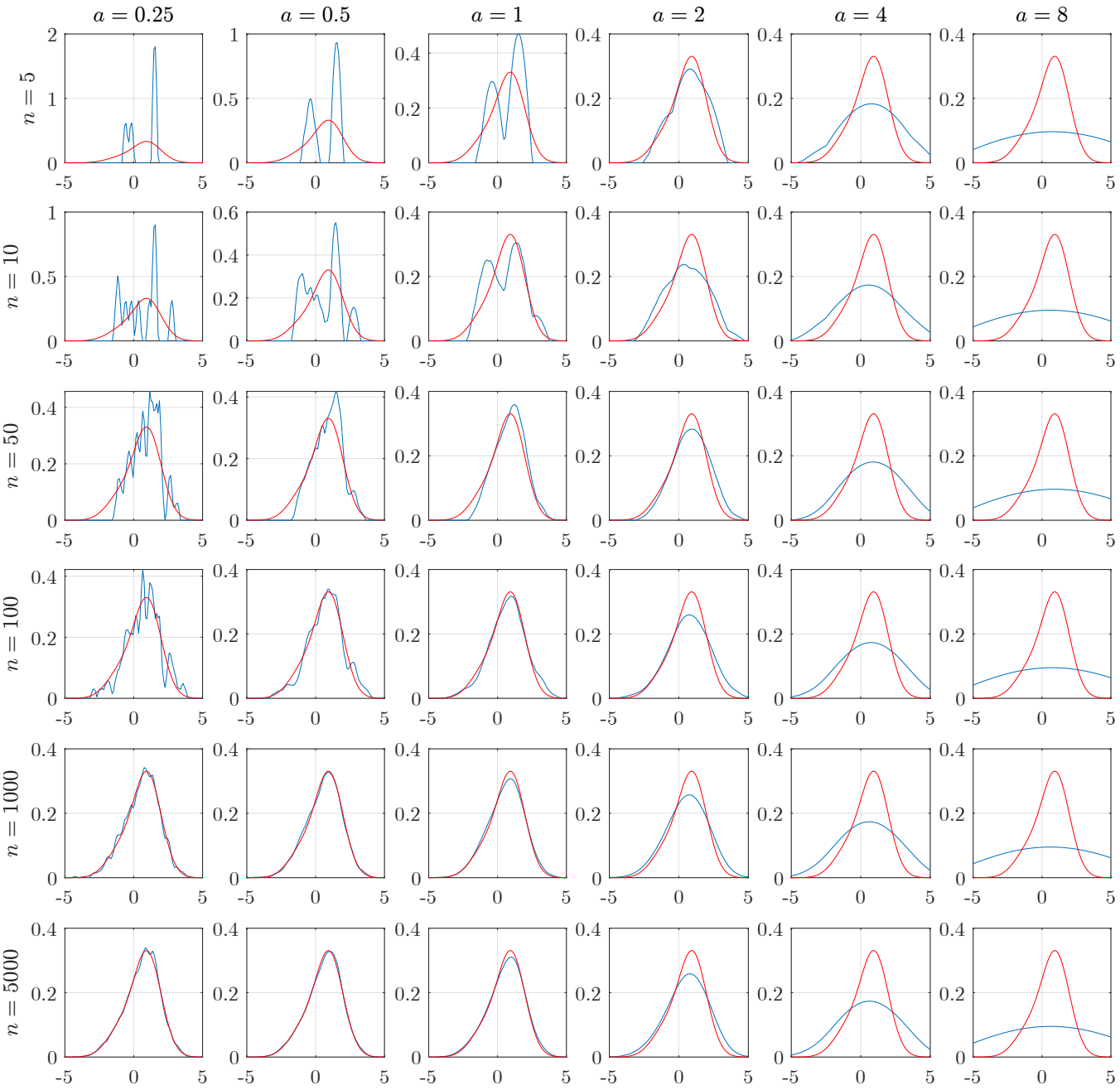


图 7: 余弦估计

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	2.2951e-04	9.5301e-05	1.5815e-05	1.9215e-05	1.2055e-06	4.0842e-08
$n=10$	6.0942e-05	2.9110e-05	2.7757e-06	3.3023e-06	2.0050e-07	1.4286e-08
$n=50$	3.2806e-06	1.3003e-06	1.8299e-07	2.7372e-07	4.0559e-08	3.9168e-09
$n=100$	5.7175e-07	1.3395e-07	1.0991e-07	1.0813e-07	4.0889e-08	1.5858e-09
$n=1000$	3.9053e-09	1.2351e-09	1.7076e-09	5.7179e-09	3.2295e-09	1.2545e-10
$n=5000$	1.4506e-39	1.4286e-39	1.4239e-38	3.6626e-37	1.0295e-36	9.5029e-36

表 8: 余弦估计均方误差方差

取 $n=5,10,50,100,1000,5000$, $a=0.25,0.5,1,2,4,8$, 使用指数核函数

$$K(x)=\frac{1}{2a}\exp\left(-\left|\frac{x}{a}\right|\right) \tag{60}$$

进行估计, 结果如图 8 所示, 图中红色曲线表示 $p(x)$. 对每一组给定的 (n,a), 进行多次重复试验, 而后对多次试验得到的 $\epsilon(p_n)$ 数组计算均值和方差, 结果分别如表 9 和表 10 所示.

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	0.0159	0.0043	0.0029	0.0049	0.0097	0.0139
$n=10$	0.0069	0.0034	0.0026	0.0050	0.0097	0.0139
$n=50$	0.0015	0.0009	0.0017	0.0049	0.0097	0.0140
$n=100$	0.0008	0.0006	0.0016	0.0049	0.0096	0.0140
$n=1000$	0.0001	0.0003	0.0016	0.0050	0.0096	0.0140
$n=5000$	0.0000	0.0003	0.0016	0.0049	0.0096	0.0140

表 9: 指数估计均方误差期望

	$a=0.25$	$a=0.5$	$a=1$	$a=2$	$a=4$	$a=8$
$n=5$	4.4102e-05	4.6225e-06	3.6103e-06	7.1588e-07	6.2318e-07	5.3395e-08
$n=10$	4.9972e-06	5.7290e-06	1.8833e-06	8.6840e-07	3.0770e-07	2.0078e-08
$n=50$	8.6323e-07	1.6730e-07	3.5128e-07	1.9786e-07	4.7670e-08	5.5717e-09
$n=100$	9.7696e-08	1.1821e-07	1.8168e-07	8.9720e-08	1.5764e-08	1.6416e-09
$n=1000$	2.5749e-09	5.8766e-09	1.3514e-08	4.2008e-09	1.4476e-09	2.4628e-10
$n=5000$	7.1807e-39	1.7308e-37	6.2858e-37	4.9494e-36	1.1087e-36	3.6428e-36

表 10: 指数估计均方误差方差

最优参数与窗函数: 通过整体比较方窗, 高斯窗, 三角形窗, 余弦窗, 指数窗等5种类型窗函数的估计效果, 均方误差的期望和方差大小, 可以看出当窗宽 $a=0.5$ 且样本数为 $n=5000$ 时, 三角形窗的估计效果最好, 其均方误差均值 6.4746×10^{-6} 和方差 8.1878×10^{-40} 都是所有估计结果中最小的.

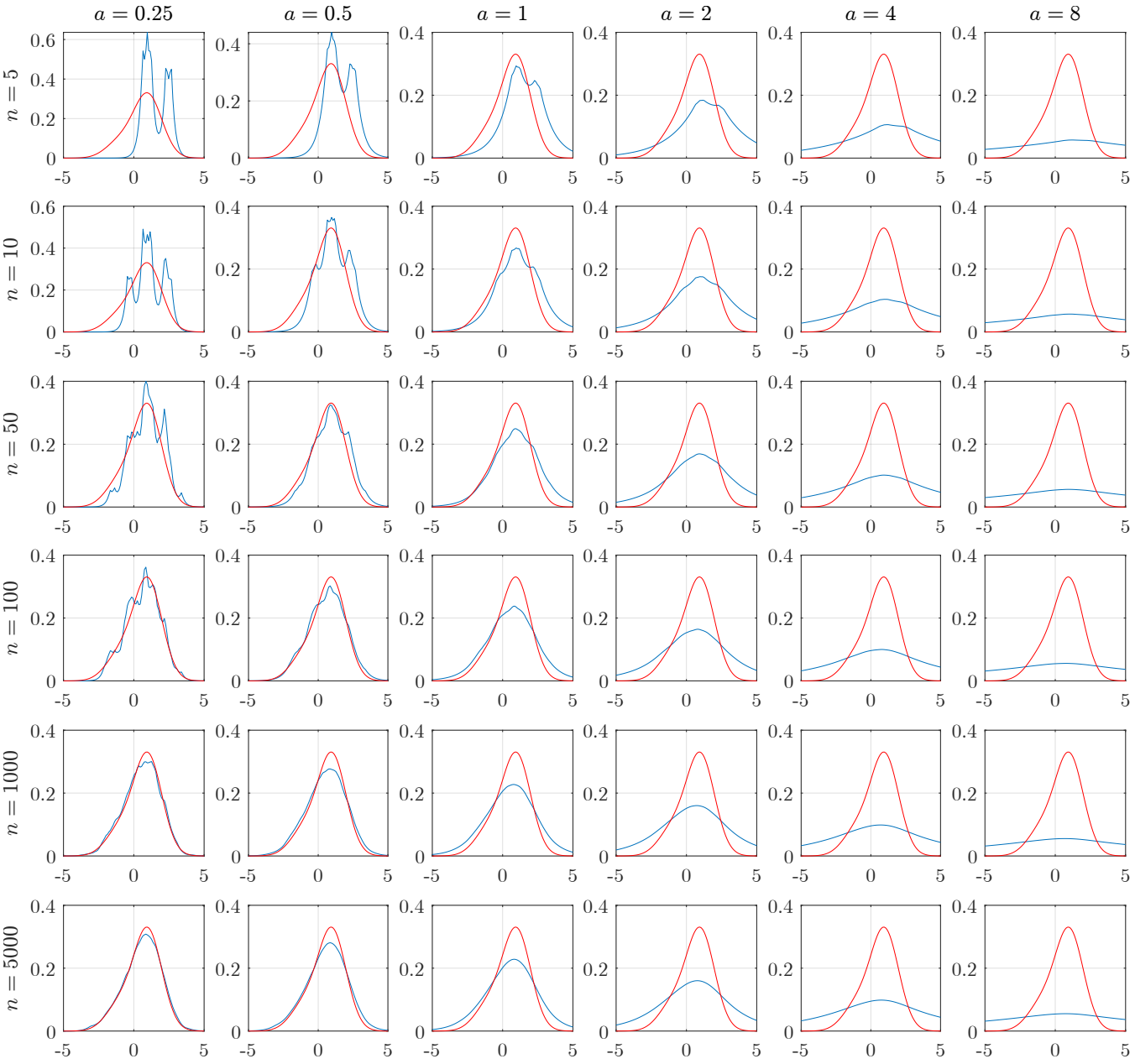


图 8: 指数估计

参考文献

[1] Casella, George, and Roger L. Berger. Statistical inference. Cengage Learning, 2021.

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 3**

GMM and EM

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

EM and GD

1. In this problem you will see connections between the EM algorithm and gradient descent. Consider a GMM with known mixture weight π_k and spherical covariances (but the radius of spheres might be different). Its log likelihood is given by

$$l\left(\left\{\mu_k, \sigma_k^2\right\}_{k=1}^K\right) = \sum_{i=1}^n \log\left(\sum_{k=1}^K \pi_k N(x_i|\mu_k, \sigma_k^2 I)\right). \tag{1}$$

A maximization algorithm based on gradient descent should be something like:

- Initialize μ_k and σ_k^2, $k \in \{1, 2, \cdots, K\}$. Set the iteration counter $t \leftarrow 1$.
- Repeat the following until convergence:
  - For $k = 1, 2, \cdots, K$,
$$\mu_k^{(t+1)} \leftarrow \mu_k^{(t)} + \eta_k^{(t)} \nabla_{\mu_k} l\left(\left\{\mu_k^{(t)}, (\sigma_k^2)^{(t)}\right\}_{k=1}^K\right) \tag{2}$$
  - For $k = 1, 2, \cdots, K$,
$$(\sigma_k^2)^{(t+1)} \leftarrow (\sigma_k^2)^{(t)} + s_k^{(t)} \nabla_{\sigma_k^2} l\left(\left\{\mu_k^{(t+1)}, (\sigma_k^2)^{(t)}\right\}_{k=1}^K\right) \tag{3}$$
  - Increase the iteration counter $t \leftarrow t + 1$.

Please prove that with properly chosen step size $\eta_k^{(t)}$ and $s_k^{(t)}$, the above gradient descent algorithm is essentially equivalent to the following *modified* EM algorithm:

- Initialize μ_k and σ_k^2, $k \in \{1, 2, \cdots, K\}$. Set the iteration counter $t \leftarrow 1$.
- Repeat the following until convergence:

– E-step:

$$\tilde{z}_{ik}^{(t+0.5)} \leftarrow P\left(x_i \in \text{cluster}_k \middle| \left\{\mu_j^{(t)}, (\sigma_j^2)^{(t)}\right\}_{j=1}^{K}, x_i\right) \tag{4}$$

– M-step:

$$\left\{\mu_k^{(t+1)}\right\}_{k=1}^{K} \leftarrow \underset{\{\mu_k\}_{k=1}^{K}}{\text{argmax}} \sum_{i=1}^{n}\sum_{k=1}^{K} \tilde{z}_{ik}^{(t+0.5)} \left[\log N\left(x_i \middle| \mu_k, (\sigma_k^2)^{(t)} I\right) + \log \pi_k\right] \tag{5}$$

– E-step:

$$\tilde{z}_{ik}^{(t+1)} \leftarrow P\left(x_i \in \text{cluster}_k \middle| \left\{\mu_j^{(t+1)}, (\sigma_j^2)^{(t)}\right\}_{j=1}^{K}, x_i\right) \tag{6}$$

– M-step:

$$\left\{(\sigma_k^2)^{(t+1)}\right\}_{k=1}^{K} \leftarrow \underset{\{\sigma_k^2\}_{k=1}^{K}}{\text{argmax}} \sum_{i=1}^{n}\sum_{k=1}^{K} \tilde{z}_{ik}^{(t+1)} \left[\log N\left(x_i \middle| \mu_k^{(t+1)}, \sigma_k^2 I\right) + \log \pi_k\right] \tag{7}$$

– Increase the iteration counter $t \leftarrow t+1$.

The main modification is inserting an extra E-step between the M-step for μ_k's and the M-step for σ_k^2's.

Hint: Find the exact algebraic form of step size $\eta_k^{(t)}$ and $s_k^{(t)}$ from M-step.

解: 设样本点 x_i 的维数为 d, 则正态分布概率密度为

$$N(x_i|\mu_k, \sigma_k^2 I) = \frac{1}{(2\pi\sigma_k^2)^{d/2}} \exp\left[-\frac{1}{2\sigma_k^2}(x_i-\mu_k)^\top (x_i-\mu_k)\right] \tag{8}$$

其对 μ_k 的梯度为

$$\nabla_{\mu_k} N(x_i|\mu_k, \sigma_k^2 I) = N(x_i|\mu_k, \sigma_k^2 I)\frac{x_i-\mu_k}{\sigma_k^2} \tag{9}$$

对 σ_k^2 的梯度为

$$\nabla_{\sigma_k^2} N(x_i|\mu_k, \sigma_k^2 I) = N(x_i|\mu_k, \sigma_k^2 I)\left[-\frac{d}{2\sigma_k^2} + \frac{1}{2(\sigma_k^2)^2}(x_i-\mu_k)^\top (x_i-\mu_k)\right] \tag{10}$$

所以, 似然函数对 μ_k 的梯度为

$$\begin{aligned} \nabla_{\mu_k} l\left(\left\{\mu_k, \sigma_k^2\right\}_{k=1}^{K}\right) &= \nabla_{\mu_k} \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \sigma_k^2 I)\right) \\ &= \sum_{i=1}^{n} \frac{\pi_k N(x_i|\mu_k, \sigma_k^2 I)}{\sum_{k=1}^{K} \pi_k N(x_i|\mu_k, \sigma_k^2 I)} \frac{x_i-\mu_k}{\sigma_k^2} \end{aligned} \tag{11}$$

似然函数对 σ_k^2 的梯度为

$$\begin{aligned}\nabla_{\sigma_k^2} l\left(\left\{\mu_k, \sigma_k^2\right\}_{k=1}^K\right) &= \nabla_{\sigma_k^2} \sum_{i=1}^n \log\left(\sum_{k=1}^K \pi_k N(x_i|\mu_k, \sigma_k^2 I)\right) \\ &= \sum_{i=1}^n \frac{\pi_k N(x_i|\mu_k, \sigma_k^2 I)}{\sum_{k=1}^K \pi_k N(x_i|\mu_k, \sigma_k^2 I)} \left[-\frac{d}{2\sigma_k^2} + \frac{1}{2(\sigma_k^2)^2}(x_i-\mu_k)^\top (x_i-\mu_k)\right]\end{aligned} \tag{12}$$

对改进 EM 算法, 第一个 E-step: 由 Bayes 公式可得

$$\begin{aligned}\tilde{z}_{ik}^{(t+0.5)} &= P\left(x_i \in \text{cluster}_k \middle| \left\{\mu_j^{(t)}, (\sigma_j^2)^{(t)}\right\}_{j=1}^K, x_i\right) \\ &= \frac{\pi_k N\left(x_i \middle| \mu_k^{(t)}, (\sigma_k^2)^{(t)} I\right)}{\sum_{k=1}^K \pi_k N\left(x_i \middle| \mu_k^{(t)}, (\sigma_k^2)^{(t)} I\right)}\end{aligned} \tag{13}$$

第一个 M-step: 令

$$Q^{(t+0.5)} \triangleq \sum_{i=1}^n \sum_{k=1}^K \tilde{z}_{ik}^{(t+0.5)} \left[\log N\left(x_i \middle| \mu_k, (\sigma_k^2)^{(t)} I\right) + \log \pi_k\right] \tag{14}$$

则其对 μ_k 的梯度为

$$\begin{aligned}\frac{\partial Q^{(t+0.5)}}{\partial \mu_k} &= \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \frac{N\left(x_i \middle| \mu_k, (\sigma_k^2)^{(t)} I\right)}{N\left(x_i \middle| \mu_k, (\sigma_k^2)^{(t)} I\right)} \frac{x_i - \mu_k}{(\sigma_k^2)^{(t)}} \\ &= \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \frac{x_i - \mu_k}{(\sigma_k^2)^{(t)}} \\ &= \frac{1}{(\sigma_k^2)^{(t)}} \left(\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} x_i - \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \mu_k\right)\end{aligned} \tag{15}$$

令此梯度为 0, 则有

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} x_i}{\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)}} \tag{16}$$

对梯度下降算法, $\mu_k^{(t+1)}$ 为

$$\begin{aligned}\mu_k^{(t+1)} &= \mu_k^{(t)} + \eta_k^{(t)} \nabla_{\mu_k} l\left(\left\{\mu_k^{(t)}, (\sigma_k^2)^{(t)}\right\}_{k=1}^K\right) \\ &= \mu_k^{(t)} + \eta_k^{(t)} \sum_{i=1}^n \frac{\pi_k N\left(x_i \middle| \mu_k^{(t)}, (\sigma_k^2)^{(t)} I\right)}{\sum_{k=1}^K \pi_k N\left(x_i \middle| \mu_k^{(t)}, (\sigma_k^2)^{(t)} I\right)} \frac{x_i - \mu_k^{(t)}}{(\sigma_k^2)^{(t)}} \\ &= \mu_k^{(t)} + \eta_k^{(t)} \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \frac{x_i - \mu_k^{(t)}}{(\sigma_k^2)^{(t)}}\end{aligned} \tag{17}$$

令梯度下降算法得到的 $\mu_k^{(t+1)}$ 与 EM 算法得到的结果相等, 则有

$$\frac{\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} x_i}{\sum_{j=1}^n \tilde{z}_{jk}^{(t+0.5)}} = \mu_k^{(t)} + \eta_k^{(t)} \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \frac{x_i - \mu_k^{(t)}}{(\sigma_k^2)^{(t)}} \tag{18}$$

通分有

$$\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} x_i = \sum_{j=1}^n \tilde{z}_{jk}^{(t+0.5)} \mu_k^{(t)} + \sum_{j=1}^n \tilde{z}_{jk}^{(t+0.5)} \eta_k^{(t)} \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \frac{x_i - \mu_k^{(t)}}{(\sigma_k^2)^{(t)}} \tag{19}$$

移项可得

$$\sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \left(x_i - \mu_k^{(t)}\right) = \frac{\eta_k^{(t)} \sum_{j=1}^n \tilde{z}_{jk}^{(t+0.5)}}{(\sigma_k^2)^{(t)}} \sum_{i=1}^n \tilde{z}_{ik}^{(t+0.5)} \left(x_i - \mu_k^{(t)}\right) \tag{20}$$

即

$$\frac{\eta_k^{(t)} \sum_{j=1}^n \tilde{z}_{jk}^{(t+0.5)}}{(\sigma_k^2)^{(t)}} = 1 \tag{21}$$

所以

$$\eta_k^{(t)} = \frac{(\sigma_k^2)^{(t)}}{\sum_{j=1}^n \tilde{z}_{jk}^{(t+0.5)}} \tag{22}$$

对改进 EM 算法, 第二个 E-step: 由 Bayes 公式可得

$$\begin{aligned}\tilde{z}_{ik}^{(t+1)} &= P\left(x_i \in \text{cluster}_k \middle| \left\{\mu_j^{(t+1)}, (\sigma_j^2)^{(t)}\right\}_{j=1}^K, x_i\right) \\ &= \frac{\pi_k N\left(x_i \middle| \mu_k^{(t+1)}, (\sigma_k^2)^{(t)} I\right)}{\sum_{k=1}^K \pi_k N\left(x_i \middle| \mu_k^{(t+1)}, (\sigma_k^2)^{(t)} I\right)}\end{aligned} \tag{23}$$

第二个 M-step: 令

$$Q^{(t+1)} \triangleq \sum_{i=1}^{n}\sum_{k=1}^{K} \tilde{z}_{ik}^{(t+1)} \left[\log N\left(x_i \middle| \mu_k^{(t+1)}, \sigma_k^2 I\right) + \log \pi_k\right] \tag{24}$$

则其对 σ_k^2 的梯度为

$$\begin{aligned}
\frac{\partial Q^{(t+1)}}{\partial \sigma_k^2} &= \sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \frac{N\left(x_i \middle| \mu_k^{(t+1)}, \sigma_k^2 I\right)}{N\left(x_i \middle| \mu_k^{(t+1)}, \sigma_k^2 I\right)} \left[-\frac{d}{2\sigma_k^2} + \frac{1}{2(\sigma_k^2)^2}\left(x_i - \mu_k^{(t+1)}\right)^\top \left(x_i - \mu_k^{(t+1)}\right)\right] \\
&= \sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left[-\frac{d}{2\sigma_k^2} + \frac{1}{2(\sigma_k^2)^2}\left(x_i - \mu_k^{(t+1)}\right)^\top \left(x_i - \mu_k^{(t+1)}\right)\right] \\
&= \frac{1}{2(\sigma_k^2)^2}\left[\sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left\|x_i - \mu_k^{(t+1)}\right\|^2 - \sigma_k^2 d \sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)}\right]
\end{aligned} \tag{25}$$

令此梯度为 0, 则有

$$(\sigma_k^2)^{(t+1)} = \frac{\sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left\|x_i - \mu_k^{(t+1)}\right\|^2}{d\sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)}} \tag{26}$$

对梯度下降算法, $(\sigma_k^2)^{(t+1)}$ 为

$$\begin{aligned}
(\sigma_k^2)^{(t+1)} &= (\sigma_k^2)^{(t)} + s_k^{(t)} \nabla_{\sigma_k^2} l\left(\left\{\mu_k^{(t+1)}, (\sigma_k^2)^{(t)}\right\}_{k=1}^{K}\right) \\
&= (\sigma_k^2)^{(t)} + s_k^{(t)} \sum_{i=1}^{n} \frac{\pi_k N\left(x_i \middle| \mu_k^{(t+1)}, (\sigma_k^2)^{(t)} I\right)}{\sum_{k=1}^{K} \pi_k N\left(x_i \middle| \mu_k^{(t+1)}, (\sigma_k^2)^{(t)} I\right)} \left[-\frac{d}{2(\sigma_k^2)^{(t)}} + \frac{\left\|x_i - \mu_k^{(t+1)}\right\|^2}{2\left[(\sigma_k^2)^{(t)}\right]^2}\right] \\
&= (\sigma_k^2)^{(t)} + s_k^{(t)} \sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left[-\frac{d}{2(\sigma_k^2)^{(t)}} + \frac{1}{2\left[(\sigma_k^2)^{(t)}\right]^2}\left\|x_i - \mu_k^{(t+1)}\right\|^2\right]
\end{aligned} \tag{27}$$

令梯度下降算法得到的 $(\sigma_k^2)^{(t+1)}$ 与 EM 算法得到的结果相等, 则有

$$\frac{\sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left\|x_i - \mu_k^{(t+1)}\right\|^2}{d\sum_{j=1}^{n} \tilde{z}_{jk}^{(t+1)}} = (\sigma_k^2)^{(t)} + s_k^{(t)} \sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left[-\frac{d}{2(\sigma_k^2)^{(t)}} + \frac{1}{2\left[(\sigma_k^2)^{(t)}\right]^2}\left\|x_i - \mu_k^{(t+1)}\right\|^2\right] \tag{28}$$

通分并移项可得

$$\sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left(\left\|x_i - \mu_k^{(t+1)}\right\|^2 - d(\sigma_k^2)^{(t)}\right) = \frac{s_k^{(t)} d \sum_{j=1}^{n} \tilde{z}_{jk}^{(t+1)}}{2\left[(\sigma_k^2)^{(t)}\right]^2} \sum_{i=1}^{n} \tilde{z}_{ik}^{(t+1)} \left(\left\|x_i - \mu_k^{(t+1)}\right\|^2 - d(\sigma_k^2)^{(t)}\right) \tag{29}$$

即

$$\frac{s_k^{(t)} d \sum_{j=1}^{n} \tilde{z}_{jk}^{(t+1)}}{2\left[(\sigma_k^2)^{(t)}\right]^2} = 1 \tag{30}$$

所以

$$s_k^{(t)} = \frac{2\left[(\sigma_k^2)^{(t)}\right]^2}{d \sum_{j=1}^{n} \tilde{z}_{jk}^{(t+1)}} \tag{31}$$

综上所述, 若取步长分别为

$$\eta_k^{(t)} = \frac{(\sigma_k^2)^{(t)}}{\sum_{j=1}^{n} \tilde{z}_{jk}^{(t+0.5)}}, \quad s_k^{(t)} = \frac{2\left[(\sigma_k^2)^{(t)}\right]^2}{d \sum_{j=1}^{n} \tilde{z}_{jk}^{(t+1)}} \tag{32}$$

则梯度下降算法与改进 EM 算法本质上是等价的.

EM for MAP Estimation

2. The EM algorithm that we talked about in class was for solving a maximum likelihood estimation problem in which we wished to maximize

$$\prod_{i=1}^{m} p(x^{(i)}|\theta) = \prod_{i=1}^{m} \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta) \tag{33}$$

where $x^{(i)}$ were visible variables, $z^{(i)}$ were hidden variables and m was the number of samples. Suppose we are working in a Bayesian framework, and wanted to find the MAP estimate of the parameters θ by maximizing

$$\left(\prod_{i=1}^{m} p(x^{(i)}|\theta)\right) p(\theta) = \left(\prod_{i=1}^{m} \sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta)\right) p(\theta) \tag{34}$$

Here, $p(\theta)$ is our prior on the parameters. Please generalize the EM algorithm to work for MAP estimation. You may assume that $\log p(x, z|\theta)$ and $\log p(\theta)$ are both concave in θ, so that the M-step is tractable if it requires only maximizing a linear combination of these quantities. (This roughly corresponds to assuming that MAP estimation is tractable when x, z is fully observed, just like in the frequentist case where we considered examples in which maximum likelihood estimation was easy if x, z was fully observed.)

Make sure your M-step is tractable, and also prove that $\left(\prod_{i=1}^{m} p(x^{(i)}|\theta)\right) p(\theta)$ (viewed as a function of θ) monotonically increases with each iteration of your algorithm.

解: 令 $X=\left\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\right\}$, 对式 (34) 取自然对数可得

$$\begin{aligned}\tilde{H}(\theta) &= \log\left[p(X|\theta)p(\theta)\right] \\ &= \log\left[\left(\prod_{i=1}^{m} p(x^{(i)}|\theta)\right)p(\theta)\right] \\ &= \log\left[\left(\prod_{i=1}^{m}\sum_{z^{(i)}} p(x^{(i)}, z^{(i)}|\theta)\right)p(\theta)\right] \\ &= \sum_{i=1}^{m}\log\sum_{z^{(i)}} p\left(x^{(i)}, z^{(i)}|\theta\right) + \log p(\theta)\end{aligned} \tag{35}$$

引入隐变量分布 $q(\cdot)$, 由 Jensen 不等式可得

$$\tilde{H}(\theta) \geqslant \sum_{i=1}^{m}\sum_{z^{(i)}} q(z^{(i)})\log p\left(x^{(i)}, z^{(i)}|\theta\right) - \sum_{i=1}^{m}\sum_{z^{(i)}} q(z^{(i)})\log q(z^{(i)}) + \log p(\theta) \triangleq \tilde{F}(q,\theta) \tag{36}$$

直接优化 $\tilde{H}(\theta)$ 可能是很困难的, 所以我们转而优化 $\tilde{H}(\theta)$ 的下界函数 $\tilde{F}(q,\theta)$. 优化函数 $\tilde{F}(q,\theta)$ 可能也是比较困难的, 因此我们采用一种简单的迭代算法对 $\tilde{F}(q,\theta)$ 寻优, 首先对变量 q, θ 初始化, 然后固定变量 $\theta_{[k]}$ 寻找能够最大化函数 $\tilde{F}(q,\theta_{[k]})$ 的参数 $q_{[k+1]}$, 再固定参数 $q_{[k+1]}$ 寻找能够最大化函数 $\tilde{F}(q_{[k+1]},\theta)$ 的参数 $\theta_{[k+1]}$, 即反复执行下面的两个步骤:

$$q_{[k+1]} \leftarrow \underset{q}{\operatorname{argmax}}\,\tilde{F}(q,\theta_{[k]}) \tag{37}$$

$$\theta_{[k+1]} \leftarrow \underset{\theta}{\operatorname{argmax}}\,\tilde{F}(q_{[k+1]},\theta) \tag{38}$$

当 $q_{[k+1]}(z^{(i)}) = p(z^{(i)}|x^{(i)},\theta_{[k]})$ 时, 式 (37) 取到最大值, 因为此时有

$$\begin{aligned}\tilde{F}(q_{[k+1]},\theta_{[k]}) &= \sum_{i=1}^{m}\sum_{z^{(i)}} q_{[k+1]}(z^{(i)})\log\frac{p\left(x^{(i)}, z^{(i)}|\theta_{[k]}\right)}{q_{[k+1]}(z^{(i)})} + \log p(\theta_{[k]}) \\ &= \sum_{i=1}^{m}\sum_{z^{(i)}} p(z^{(i)}|x^{(i)},\theta_{[k]})\log\frac{p(z^{(i)}|x^{(i)},\theta_{[k]})p(x^{(i)}|\theta_{[k]})}{p(z^{(i)}|x^{(i)},\theta_{[k]})} + \log p(\theta_{[k]}) \\ &= \sum_{i=1}^{m}\sum_{z^{(i)}} p(z^{(i)}|x^{(i)},\theta_{[k]})\log p(x^{(i)}|\theta_{[k]}) + \log p(\theta_{[k]}) \\ &= \sum_{i=1}^{m}\log p(x^{(i)}|\theta_{[k]}) + \log p(\theta_{[k]}) \\ &= \log\left[\left(\prod_{i=1}^{m} p(x^{(i)}|\theta_{[k]})\right)p(\theta_{[k]})\right] \\ &= \tilde{H}(\theta_{[k]})\end{aligned} \tag{39}$$

当 $q_{[k+1]}(z^{(i)}) = p(z^{(i)}|x^{(i)},\theta_{[k]})$ 时,

$$\tilde{F}(q_{[k+1]},\theta) = \sum_{i=1}^{m}\sum_{z^{(i)}} q_{[k+1]}(z^{(i)})\log p\left(x^{(i)}, z^{(i)}|\theta\right) - \sum_{i=1}^{m}\sum_{z^{(i)}} q_{[k+1]}(z^{(i)})\log q_{[k+1]}(z^{(i)}) + \log p(\theta) \tag{40}$$

由于第二项不包含需要优化的变量 θ, 则可定义

$$\tilde{Q}(\theta_{[k]}, \theta) = \sum_{i=1}^{m} \sum_{z^{(i)}} p(z^{(i)}|x^{(i)}, \theta_{[k]}) \log p\left(x^{(i)}, z^{(i)}|\theta\right) + \log p(\theta) \tag{41}$$

假设 $\log p(x, z|\theta)$ 和 $\log p(\theta)$ 都是 θ 的凹函数, 则最大化函数 $\tilde{Q}$ 是一个无约束的凸优化问题, 即 M-step 是容易处理的 (tractable).

综上, 广义 EM 算法为:

- 初始化变量 q, θ
- E 步骤, 计算函数

$$\tilde{Q}(\theta_{[k]}, \theta) = \sum_{i=1}^{m} \sum_{z^{(i)}} p(z^{(i)}|x^{(i)}, \theta_{[k]}) \log p\left(x^{(i)}, z^{(i)}|\theta\right) + \log p(\theta) \tag{42}$$

- M 步骤,

$$\theta_{[k+1]} \leftarrow \operatorname*{argmax}_{\theta} \tilde{Q}(\theta_{[k]}, \theta) \tag{43}$$

- 如果算法收敛则停止, 否则回到 E 步骤.

下面证明在广义 EM 算法下, $\left(\prod_{i=1}^{m} p(x^{(i)}|\theta)\right) p(\theta)$ 作为 θ 的函数的单调性. 由式 (39) 并结合广义 EM 算法不断迭代求最大值可知

$$\tilde{H}(\theta_{[k]}) = \tilde{F}(q_{[k+1]}, \theta_{[k]}) \leqslant \tilde{F}(q_{[k+1]}, \theta_{[k+1]}) \leqslant \tilde{F}(q_{[k+2]}, \theta_{[k+1]}) = \tilde{H}(\theta_{[k+1]}), \quad \forall\ k \geqslant 1 \tag{44}$$

所以, 函数

$$\tilde{H}(\theta) = \log \left[\left(\prod_{i=1}^{m} p(x^{(i)}|\theta) \right) p(\theta) \right]$$

随着算法的迭代是单调递增的, 又自然对数函数是一一映射, 则 $\left(\prod_{i=1}^{m} p(x^{(i)}|\theta)\right) p(\theta)$ 作为 θ 的函数也是随着算法的迭代单调递增的.

Programming 1 (EM and GMM)

3. Consider the case that the hidden variable $y \in \{1, 2, \cdots, m\}$ is discrete while the visible variable $x \in R^d$ is continuous. In other words, we consider mixture models of the form

$$p(x) = \sum_{j=1}^{m} p(x|y = j)p(y = j) \tag{45}$$

We assume throughout that x is conditionally Gaussian in the sense that $x \sim \mathcal{N}(\mu_j, \Sigma_j)$ when $y = j$. We have provided you with an example EM code for mixture of Gaussians (with visualization) in MATLAB. The command to run is:

```
[param,history,ll] = em_mix(data,m,eps);
```

where the input points are given as rows of `data`, `m` is the number of components in the estimated mixture, and `eps` determines the stopping criteria of EM: the algorithm stops when the relative change in log-likelihood falls below `eps`. In the output, `param` is a cell array with `m` elements. Each element is a structure with the following fields:

`mean` - the resulting mean of the Gaussian component,

`cov` - the resulting covariance matrix of the component,

`p` - the resulting estimate of the mixing parameter.

The value of `param` is updated after every iteration of EM; the output argument `history` contains copies of these subsequent values of `param` and allows to analyze our experiments. Finally, `ll` is the vector where the t^{th} element is the value of the log-likelihood of the `data` after t iterations (i.e. the last element is the final log-likelihood of the fitted mixture of Gaussians).

Hint: For the following two questions you are encouraged to google "BIC (Bayesian Information Criterion)" to help you with the model selection process. Of course other criteria are welcomed as long as you give convincing reasons.

Hint: For this assignment, you are allowed to implement EM algorithm manually in python, and you can use scipy.io.loadmat to load the data.

3.1. Run the EM algorithm based on `data` provided by `emdata.mat` with `m = 2, 3, 4, 5` components. Select the appropriate model (number of components) and give reasons for your choice. Note that you may have to rerun the algorithm a few times (and select the model with the highest log-likelihood) for each choice of `m` as EM can sometimes get stuck in a local minimum. Is the model selection result sensible based on what you would expect visually? Why or why not?

解: 对每个 m 进行 10 次测试, 选择对数似然概率最大的那次测试结果作为该组的结果, 如图 1 所示.

选择 BIC 作为模型参数选择依据,

$$\text{BIC} = k \ln n - 2 \ln \hat{l} \tag{46}$$

其中, n 为数据点个数, $\hat{l}$ 为对数似然概率, k 为模型估计的变量个数, 其中分配概率为 1, 均值为 2, 协方差对称矩阵为 $4 - 1 = 3$, 又因为分配概率之和固定 $(= 1)$ 则最后需 -1,

$$k = (1 + d + d^2 - 1)m - 1 = 6m - 1 \tag{47}$$

其中, d 为数据点维数, 此处为 $d = 2$.

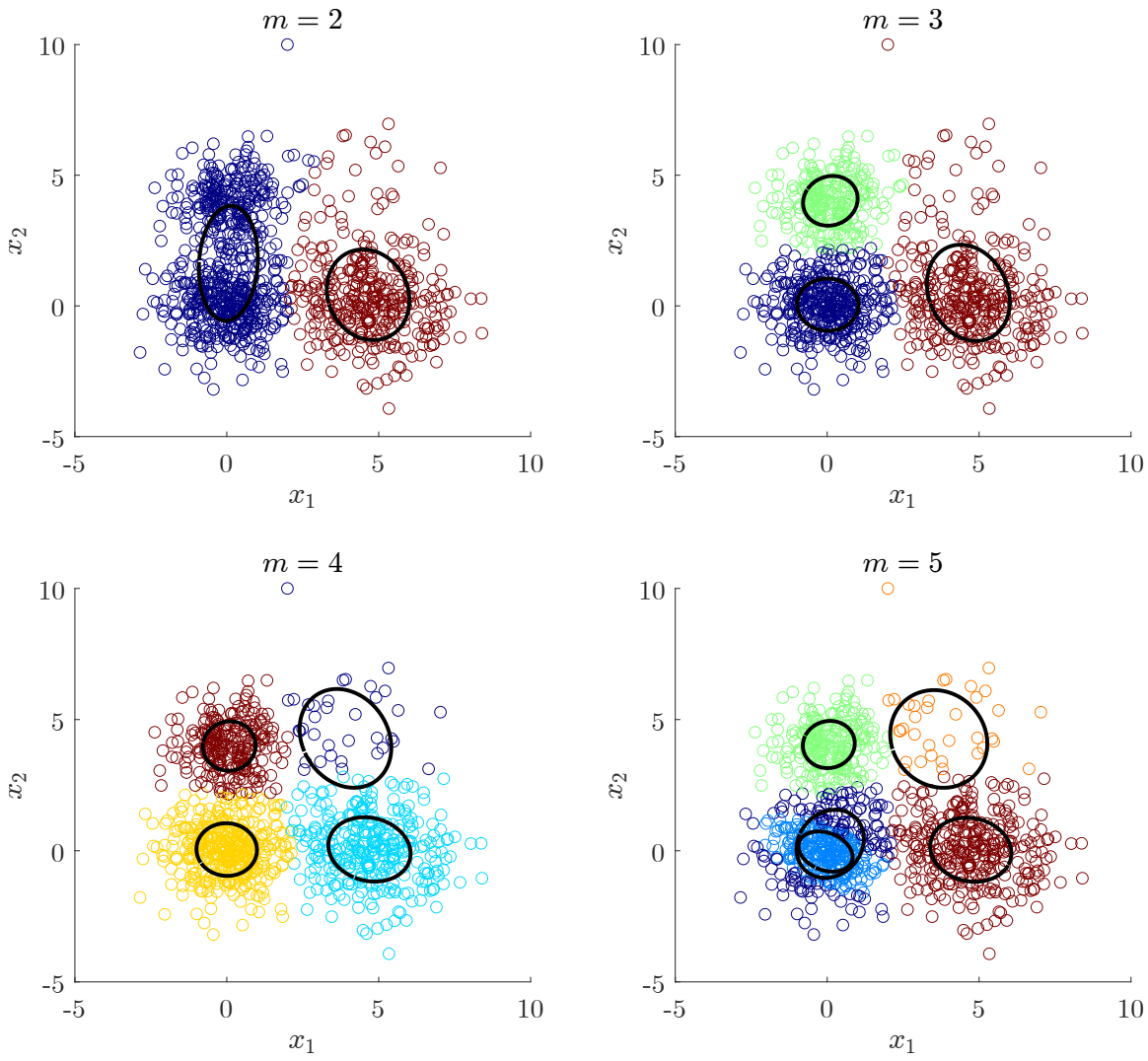


图 1: 使用 EM 算法对 GMM 模型进行估计

BIC 计算结果如表 1 所示, 可知当 $m=4$ 时, BIC 最小, 即 $m=4$ 是最恰当的模型参数.

表 1: 不同参数 m 取值对应的 BIC 数值

	$m=2$	$m=3$	$m=4$	$m=5$
BIC	8542.29	8371.61	8330.65	8372.01

此模型选择结果与视觉预估不符, 在初看数据时, 可以发现有 3 个明显的数据聚集中心, 因此猜测 $m=3$ 是最恰当的模型参数. 但是程序运行结果表明 $m=4$ 是最恰当的模型参数, 这是因为右上角那部分点虽然不够集中, 但是比较不容易被另外三个组分共同作用而产生, 相反若右上角存在一个单独的组分, 并且赋予其很小的比例系数, 则会使得模型拟合效果更好.

3.2. Modify the M-step of the EM code so that the covariance matrices of the Gaussian components are constrained to be equal. Give detailed derivation. Rerun the code and then select an appropriate model. Would we select a different number of components in this case?

解: EM 算法的 M-step 为

$$\Theta^{(i)} = \underset{\Theta}{\operatorname{argmax}}\, Q(\Theta, \Theta^{(i-1)}) \tag{48}$$

设 GMM 模型为

$$p(\boldsymbol{x}|\Theta) = \sum_{i=1}^{M} \alpha_i p_i(\boldsymbol{x}|\theta_i) \tag{49}$$

则有

$$\begin{aligned} Q(\Theta, \Theta^g) &= \sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l p_l(x_i|\theta_l)) p(l|x_i, \Theta^g) \\ &= \sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l) p(l|x_i, \Theta^g) + \sum_{l=1}^{M}\sum_{i=1}^{N} \log(p_l(x_i|\theta_l)) p(l|x_i, \Theta^g) \end{aligned} \tag{50}$$

其中第一项不含 θ_l, 第二项不含 α_l, 则可在优化 $Q(\Theta, \Theta^g)$ 时分别进行优化.

由于 α_l 满足

$$\sum_{l=1}^{M} \alpha_l = 1 \tag{51}$$

则引入 Lagrange 乘子 λ 得到 Lagrange 函数

$$\mathcal{L}(\alpha_l, \lambda) = \sum_{l=1}^{M}\sum_{i=1}^{N} \log(\alpha_l) p(l|x_i, \Theta^g) + \lambda \left(\sum_{l=1}^{M} \alpha_l - 1 \right) \tag{52}$$

Lagrange 函数对 α_l 的偏导为

$$\frac{\partial \mathcal{L}(\alpha_l, \lambda)}{\partial \alpha_l} = \sum_{i=1}^{N} \frac{1}{\alpha_l} p(l|x_i, \Theta^g) + \lambda \tag{53}$$

令此偏导为 0 并对 l 求和, 有

$$\sum_{l=1}^{M}\sum_{i=1}^{N} p(l|x_i,\Theta^g) + \sum_{l=1}^{M}\alpha_l\lambda = 0 \tag{54}$$

所以 $\lambda = -N$, 则

$$\alpha_l = \frac{1}{N}\sum_{i=1}^{N} p(l|x_i,\Theta^g) \tag{55}$$

定义

$$\begin{aligned} B &\triangleq \sum_{l=1}^{M}\sum_{i=1}^{N}\log(p_l(x_i|\theta_l))p(l|x_i,\Theta^g) \\ &= \sum_{l=1}^{M}\sum_{i=1}^{N}\left[-\frac{\log|\Sigma|}{2} - \frac{(x_i-\mu_l)^\top\Sigma^{-1}(x_i-\mu_l)}{2}\right]p(l|x_i,\Theta^g) \\ &= \sum_{l=1}^{M}\left[\frac{1}{2}\log|\Sigma^{-1}|\sum_{i=1}^{N}p(l|x_i,\Theta^g) - \frac{1}{2}\sum_{i=1}^{N}p(l|x_i,\Theta^g)\mathrm{tr}\left(\Sigma^{-1}N_{l,i}\right)\right] \end{aligned} \tag{56}$$

其中 $N_{l,i} = (x_i-\mu_l)(x_i-\mu_l)^\top$.

B 对 μ_l 的偏导为

$$\frac{\partial B}{\partial \mu_l} = \sum_{i=1}^{N}\Sigma^{-1}(x_i-\mu_l)p(l|x_i,\Theta^g) \tag{57}$$

令此偏导为 0, 可得

$$\mu_l = \frac{\sum_{i=1}^{N}x_i p(l|x_i,\Theta^g)}{\sum_{i=1}^{N}p(l|x_i,\Theta^g)} \tag{58}$$

B 对 Σ^{-1} 的偏导为

$$\begin{aligned} \frac{\partial B}{\partial \Sigma^{-1}} &= \sum_{l=1}^{M}\left[\frac{1}{2}\sum_{i=1}^{N}p(l|x_i,\Theta^g)(2\Sigma - \mathrm{diag}(\Sigma)) - \frac{1}{2}\sum_{i=1}^{N}p(l|x_i,\Theta^g)(2N_{l,i} - \mathrm{diag}(N_{l,i}))\right] \\ &= \frac{1}{2}\sum_{l=1}^{M}\sum_{i=1}^{N}p(l|x_i,\Theta^g)(2M_{l,i} - \mathrm{diag}(M_{l,i})) \\ &= 2S - \mathrm{diag}(S) \end{aligned} \tag{59}$$

其中

$$M_{l,i} = \Sigma - N_{l,i}, \quad S = \frac{1}{2}\sum_{l=1}^{M}\sum_{i=1}^{N}p(l|x_i,\Theta^g)M_{l,i} \tag{60}$$

由 $2S - \mathrm{diag}(S) = 0$, 可知 $S = 0$, 即

$$\sum_{l=1}^{M}\sum_{i=1}^{N}p(l|x_i,\Theta^g)(\Sigma - N_{l,i}) = 0 \tag{61}$$

所以

$$\begin{aligned}\Sigma &= \frac{\sum_{l=1}^{M}\sum_{i=1}^{N} p(l|x_i,\Theta^g)N_{l,i}}{\sum_{l=1}^{M}\sum_{i=1}^{N} p(l|x_i,\Theta^g)} \\ &= \frac{\sum_{l=1}^{M}\sum_{i=1}^{N} p(l|x_i,\Theta^g)(x_i-\mu_l)(x_i-\mu_l)^\top}{\sum_{l=1}^{M}\sum_{i=1}^{N} p(l|x_i,\Theta^g)}\end{aligned} \tag{62}$$

则修改的 M-step 为

$$\begin{aligned}\alpha_l^{\text{new}} &= \frac{1}{N}\sum_{i=1}^{N} p(l|x_i,\Theta^g) \\ \mu_l^{\text{new}} &= \frac{\sum_{i=1}^{N} x_i p(l|x_i,\Theta^g)}{\sum_{i=1}^{N} p(l|x_i,\Theta^g)} \\ \Sigma^{\text{new}} &= \frac{\sum_{l=1}^{M}\sum_{i=1}^{N} p(l|x_i,\Theta^g)(x_i-\mu_l^{\text{new}})(x_i-\mu_l^{\text{new}})^\top}{\sum_{l=1}^{M}\sum_{i=1}^{N} p(l|x_i,\Theta^g)}\end{aligned} \tag{63}$$

当每个组分的协方差矩阵都相同时, 测试过程中经常会出现 EM 算法收敛到局部最优解的情况, 为了尽可能避免局部最优解的影响, 对每个 m 进行 200 次测试, 选择对数似然概率最大的那次测试结果作为该组的结果, 如图 2 所示.

BIC 计算结果如表 2 所示, 可知当 $m=4$ 时, BIC 最小, 即 $m=4$ 仍为最恰当的模型参数.

表 2: 不同参数 m 取值对应的 BIC 数值

	$m=2$	$m=3$	$m=4$	$m=5$
BIC	8597.40	8463.17	8397.21	8415.93

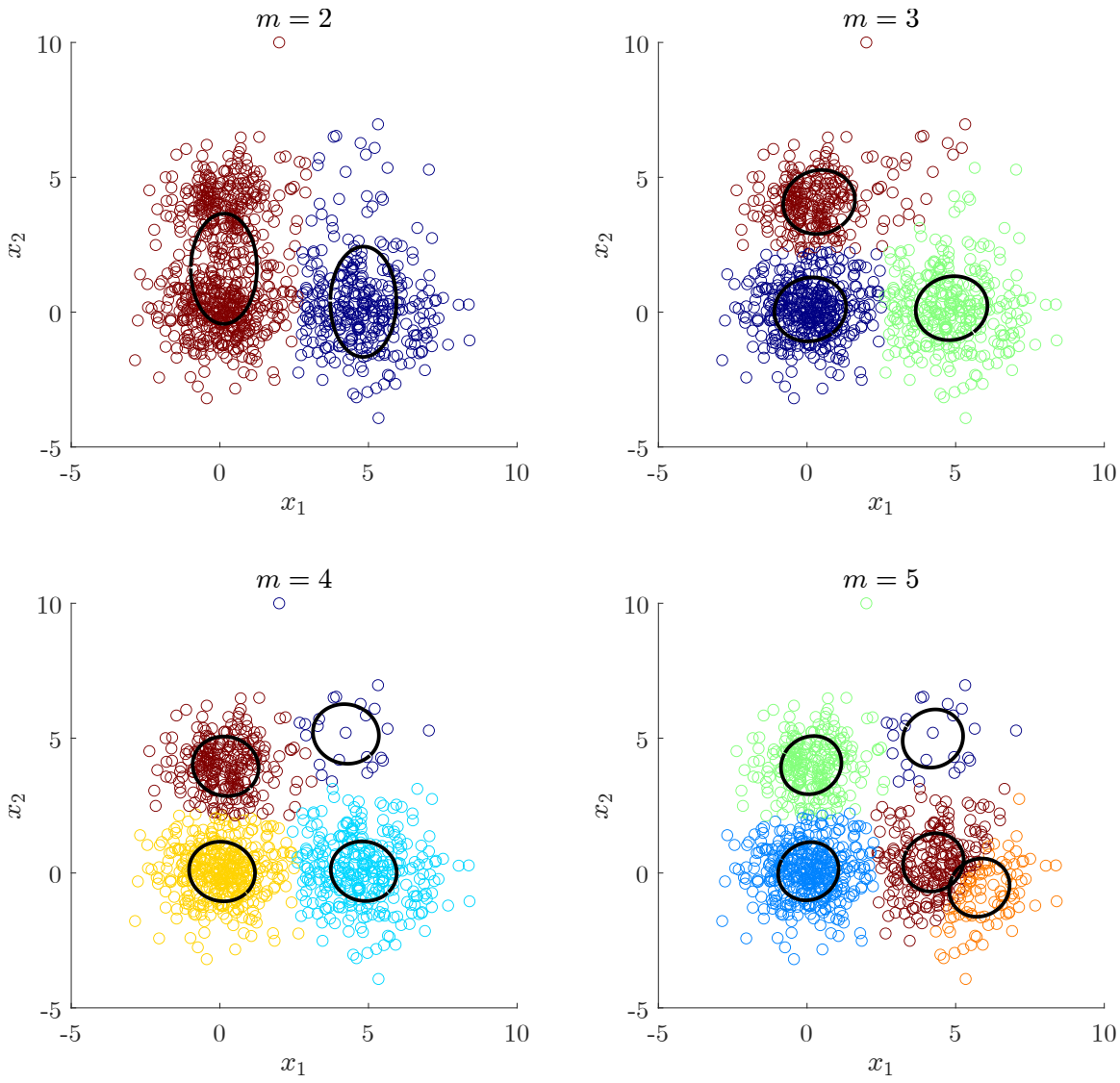


图 2: 使用修改的 EM 算法对 GMM 模型进行估计

Programming 2 (Missing Data)

4. Suppose we know that the ten data points in category ω_1 in Tab. 3 come from a three-dimensional Gaussian. Suppose, however, that we do not have access to the x_3 components for the even-numbered data points.

表 3: 样本数据点

	ω_1		
point	x_1	x_2	x_3
1	0.42	-0.087	0.58
2	-0.2	-3.3	-3.4
3	1.3	-0.32	1.7
4	0.39	0.71	0.23
5	-1.6	-5.3	-0.15
6	-0.029	0.89	-4.7
7	-0.23	1.9	2.2
8	0.27	-0.3	-0.87
9	-1.9	0.76	-2.1
10	0.87	-1.0	-2.6

4.1. Write an EM program to estimate the mean and covariance of the distribution. Start your estimate with $\boldsymbol{\mu}^0 = \mathbf{0}$ and $\boldsymbol{\Sigma}^0 = \boldsymbol{I}$, the three-dimensional identity matrix.

解: 记 $O = \{1, 3, 5, 7, 9\}$, $E = \{2, 4, 6, 8, 10\}$. 偶数点的 x_3 数据丢失, 则令 x_{i3}, $i \in E$ 为隐变量, 并令 $q(\cdot)$ 表示其概率分布, 则对数似然函数为

$$\begin{aligned} H(\theta) &= \sum_{i\in O} \ln p(x_{i1}, x_{i2}, x_{i3}|\theta) + \sum_{i\in E} \ln p(x_{i1}, x_{i2}|\theta) \\ &= \sum_{i\in O} \ln p(x_{i1}, x_{i2}, x_{i3}|\theta) + \sum_{i\in E} \ln \int_{-\infty}^{\infty} p(x_{i1}, x_{i2}, x_{i3}|\theta) \mathrm{d}x_{i3} \\ &= \sum_{i\in O} \ln p(x_{i1}, x_{i2}, x_{i3}|\theta) + \sum_{i\in E} \ln \int_{-\infty}^{\infty} q(x_{i3}) \frac{p(x_{i1}, x_{i2}, x_{i3}|\theta)}{q(x_{i3})} \mathrm{d}x_{i3} \\ &\geqslant \sum_{i\in O} \ln p(x_{i1}, x_{i2}, x_{i3}|\theta) + \sum_{i\in E} \int_{-\infty}^{\infty} q(x_{i3}) \ln \frac{p(x_{i1}, x_{i2}, x_{i3}|\theta)}{q(x_{i3})} \mathrm{d}x_{i3} \end{aligned} \tag{64}$$

由此可得 Q 函数为

$$Q(\theta^{(k)}, \theta) = \sum_{i\in O} \ln p(x_{i1}, x_{i2}, x_{i3}|\theta) + \sum_{i\in E} \int_{-\infty}^{\infty} p(x_{i3}|x_{i1}, x_{i2}, \theta^{(k)}) \ln p(x_{i1}, x_{i2}, x_{i3}|\theta) \mathrm{d}x_{i3} \tag{65}$$

令 $x_i = (x_{i1}, x_{i2}, x_{i3})^\top$, $\mu = (\mu_1, \mu_2, \mu_3)^\top$, 则

$$\ln p(x_{i1}, x_{i2}, x_{i3}|\theta) = \ln p(x_i|\theta) = -\frac{3}{2}\ln(2\pi) + \frac{1}{2}\ln|\Sigma^{-1}| - \frac{1}{2}(x_i - \mu)^\top \Sigma^{-1}(x_i - \mu) \tag{66}$$

Q 对 μ 的偏导数为

$$\begin{aligned}\frac{\partial Q(\theta^{(k)},\theta)}{\partial \mu} &= \sum_{i\in O}\Sigma^{-1}(x_i-\mu)+\sum_{i\in E}\int_{-\infty}^{\infty}p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})\Sigma^{-1}(x_i-\mu)\mathrm{d}x_{i3}\\ &= \sum_{i\in O}\Sigma^{-1}\begin{pmatrix}x_{i1}-\mu_1\\ x_{i2}-\mu_2\\ x_{i3}-\mu_3\end{pmatrix}+\sum_{i\in E}\Sigma^{-1}\begin{pmatrix}x_{i1}-\mu_1\\ x_{i2}-\mu_2\\ \displaystyle\int_{-\infty}^{\infty}x_{i3}p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})\mathrm{d}x_{i3}-\mu_3\end{pmatrix}\\ &= \Sigma^{-1}\begin{pmatrix}\displaystyle\sum_{i=1}^{10}(x_{i1}-\mu_1)\\ \displaystyle\sum_{i=1}^{10}(x_{i2}-\mu_2)\\ \displaystyle\sum_{i\in O}x_{i3}+\sum_{i\in E}\mathbb{E}(x_{i3}|x_{i1},x_{i2},\theta^{(k)})-\sum_{i=1}^{10}\mu_3\end{pmatrix}\end{aligned}\tag{67}$$

令此偏导数为 0, 有

$$\begin{aligned}\mu_1 &= \frac{1}{10}\sum_{i=1}^{10}x_{i1}\\ \mu_2 &= \frac{1}{10}\sum_{i=1}^{10}x_{i2}\\ \mu_3 &= \frac{1}{10}\sum_{i\in O}x_{i3}+\frac{1}{10}\sum_{i\in E}\mathbb{E}(x_{i3}|x_{i1},x_{i2},\theta^{(k)})\end{aligned}\tag{68}$$

由高斯分布的性质可知, 高斯分布的条件分布仍为高斯分布, 记

$$\Sigma^{(k)}=\begin{bmatrix}\Sigma_{12,12}^{(k)} & \Sigma_{12,3}^{(k)}\\ \Sigma_{3,12}^{(k)} & \Sigma_{3,3}^{(k)}\end{bmatrix}\tag{69}$$

则由相关公式 [1–3] 可知

$$\mathbb{E}(x_{i3}|x_{i1},x_{i2},\theta^{(k)})=\mu_3^{(k)}+\Sigma_{3,12}^{(k)}\left[\Sigma_{12,12}^{(k)}\right]^{-1}\begin{pmatrix}x_{i1}-\mu_1^{(k)}\\ x_{i2}-\mu_2^{(k)}\end{pmatrix}\tag{70}$$

所以

$$\mu_3=\frac{1}{10}\sum_{i\in O}x_{i3}+\frac{1}{10}\sum_{i\in E}\left[\mu_3^{(k)}+\Sigma_{3,12}^{(k)}\left[\Sigma_{12,12}^{(k)}\right]^{-1}\begin{pmatrix}x_{i1}-\mu_1^{(k)}\\ x_{i2}-\mu_2^{(k)}\end{pmatrix}\right]\tag{71}$$

令

$$N_i^{(k)}=(x_i-\mu^{(k+1)})(x_i-\mu^{(k+1)})^{\top},\quad M_i=\Sigma-N_i^{(k)}\tag{72}$$

则 Q 对 Σ^{-1} 的偏导数为

$$\frac{\partial Q(\theta^{(k)},\theta)}{\partial \Sigma^{-1}} = \sum_{i\in O}\frac{1}{2}\big[2M_i - \mathrm{diag}(M_i)\big] + \sum_{i\in E}\int_{-\infty}^{\infty} p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})\frac{1}{2}\big[2M_i - \mathrm{diag}(M_i)\big]\mathrm{d}x_{i3} \tag{73}$$

令此偏导数为 0, 有

$$\sum_{i\in O} M_i + \sum_{i\in E}\int_{-\infty}^{\infty} p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})M_i\mathrm{d}x_{i3} = 0 \tag{74}$$

即

$$\sum_{i\in O}(\Sigma - N_i^{(k)}) + \sum_{i\in E}\int_{-\infty}^{\infty} p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})(\Sigma - N_i^{(k)})\mathrm{d}x_{i3} = 0 \tag{75}$$

由概率密度函数积分为 1, 可得

$$\sum_{i=1}^{10}\Sigma - \sum_{i\in O} N_i^{(k)} - \sum_{i\in E}\int_{-\infty}^{\infty} p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})N_i^{(k)}\mathrm{d}x_{i3} = 0 \tag{76}$$

所以

$$\begin{aligned}\Sigma &= \frac{1}{10}\sum_{i\in O} N_i^{(k)} + \frac{1}{10}\sum_{i\in E}\int_{-\infty}^{\infty} p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})N_i^{(k)}\mathrm{d}x_{i3}\\ &\triangleq \frac{1}{10}\sum_{i\in O} N_i^{(k)} + \frac{1}{10}\sum_{i\in E} W_i^{(k)}\end{aligned} \tag{77}$$

其中

$$\begin{aligned}W_i^{(k)} &= \int_{-\infty}^{\infty} p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})(x_i-\mu^{(k+1)})(x_i-\mu^{(k+1)})^\top\mathrm{d}x_{i3}\\ &= \int_{-\infty}^{\infty} p(x_{i3}|x_{i1},x_{i2},\theta^{(k)})\begin{bmatrix}(x_{i1}-\mu_1)^2 & (x_{i1}-\mu_1)(x_{i2}-\mu_2) & (x_{i1}-\mu_1)(x_{i3}-\mu_3)\\ (x_{i2}-\mu_2)(x_{i1}-\mu_1) & (x_{i2}-\mu_2)^2 & (x_{i2}-\mu_2)(x_{i3}-\mu_3)\\ (x_{i3}-\mu_3)(x_{i1}-\mu_1) & (x_{i3}-\mu_3)(x_{i2}-\mu_2) & (x_{i3}-\mu_3)^2\end{bmatrix}\mathrm{d}x_{i3}\end{aligned} \tag{78}$$

记

$$E_i^{(k)} \triangleq \mathbb{E}(x_{i3}|x_{i1},x_{i2},\theta^{(k)}) = \mu_3^{(k)} + \Sigma_{3,12}^{(k)}\left[\Sigma_{12,12}^{(k)}\right]^{-1}\begin{pmatrix}x_{i1}-\mu_1^{(k)}\\ x_{i2}-\mu_2^{(k)}\end{pmatrix} \tag{79}$$

以及

$$D_i^{(k)} \triangleq \mathrm{Var}(x_{i3}|x_{i1},x_{i2},\theta^{(k)}) = \Sigma_{3,3}^{(k)} - \Sigma_{3,12}^{(k)}\left[\Sigma_{12,12}^{(k)}\right]^{-1}\Sigma_{12,3}^{(k)} \tag{80}$$

则由期望和方差的定义得到

$$W_i^{(k)} = \begin{bmatrix}(x_{i1}-\mu_1)^2 & (x_{i1}-\mu_1)(x_{i2}-\mu_2) & (x_{i1}-\mu_1)\left(E_i^{(k)}-\mu_3\right)\\ (x_{i2}-\mu_2)(x_{i1}-\mu_1) & (x_{i2}-\mu_2)^2 & (x_{i2}-\mu_2)\left(E_i^{(k)}-\mu_3\right)\\ \left(E_i^{(k)}-\mu_3\right)(x_{i1}-\mu_1) & \left(E_i^{(k)}-\mu_3\right)(x_{i2}-\mu_2) & D_i^{(k)} + \left(E_i^{(k)}\right)^2 - 2\mu_3 E_i^{(k)} + \mu_3^2\end{bmatrix} \tag{81}$$

综上所述, 有

$$\begin{aligned}
\mu_1^{k+1} &= \frac{1}{10}\sum_{i=1}^{10} x_{i1} \\
\mu_2^{k+1} &= \frac{1}{10}\sum_{i=1}^{10} x_{i2} \\
\mu_3^{k+1} &= \frac{1}{10}\sum_{i\in O} x_{i3} + \frac{1}{10}\sum_{i\in E} E_i^{(k)} \\
\Sigma^{k+1} &= \frac{1}{10}\sum_{i\in O} N_i^{(k)} + \frac{1}{10}\sum_{i\in E} W_i^{(k)}
\end{aligned} \tag{82}$$

编写程序, 得到估计的均值向量和协方差矩阵分别为

$$\hat{\mu} = \begin{pmatrix} -0.0709 \\ -0.6047 \\ 0.7728 \end{pmatrix}, \quad \hat{\Sigma} = \begin{bmatrix} 0.9062 & 0.5678 & 0.8814 \\ 0.5678 & 4.2007 & 0.4621 \\ 0.8814 & 0.4621 & 1.7828 \end{bmatrix} \tag{83}$$

4.2. Compare your final estimation with the case when we remove all even-numbered data points (2, 4, 6, 8, 10).

解: 均值向量和协方差矩阵分别为

$$\hat{\mu} = \begin{pmatrix} -0.4020 \\ -0.6094 \\ 0.4460 \end{pmatrix}, \quad \hat{\Sigma} = \begin{bmatrix} 1.4563 & 0.9843 & 1.4148 \\ 0.9843 & 6.1061 & 0.8287 \\ 1.4148 & 0.8287 & 2.3009 \end{bmatrix} \tag{84}$$

全部偶数数据点的缺失导致均值和协方差矩阵的估计有了很大的变化, 除了 x_2 的均值估计基本保持不变外, 其余数值均发生了较大的变化, 与只缺失 x_3 的偶数数据点相比近乎是两个不同的模型了, 可以看出数据点的个数对模型参数的估计有较大的影响.

4.3. Compare your final estimation with the case when there are no missing data, namely we have access to all x_3.

解: 均值向量和协方差矩阵分别为

$$\hat{\mu} = \begin{pmatrix} -0.0709 \\ -0.6047 \\ -0.9110 \end{pmatrix}, \quad \hat{\Sigma} = \begin{bmatrix} 0.9062 & 0.5678 & 0.3941 \\ 0.5678 & 4.2007 & 0.7337 \\ 0.3941 & 0.7337 & 4.5419 \end{bmatrix} \tag{85}$$

由于只缺失第三维的部分数据, 所以前两维的均值和协方差估计都是准确的, 而第三维有明显的误差, 且误差较大. 这主要是因为缺失的五个数据, 恰好都是 x_3 比较小的数据, 导致估计有了较大的误差.

参考文献

[1] Flying pig. Deriving the conditional distributions of a multivariate normal distribution, 2012. https://stats.stackexchange.com/questions/30588

[2] Ruye Wang. Marginal and conditional distributions of multivariate normal distribution, 2006. http://fourier.eng.hmc.edu/e161/lectures/gaussianprocess/node7.html

[3] Steffen Lauritzen. The Multivariate Gaussian Distribution, 2009. http://www.stats.ox.ac.uk/ steffen/teaching/bs2HT9/gauss.pdf

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 4**

Overfitting, Error Rate Estimation and Linear Classifiers

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Overfitting

1. Consider N i.i.d. observations $\boldsymbol{X} = \{x_1, x_2, \ldots, x_N\}$ with corresponding target values $\boldsymbol{T} = \{t_1, t_2, \ldots, t_N\}$. We want to fit these observations into a model

$$t = y(x, \boldsymbol{w}) + \epsilon \tag{1}$$

where $\boldsymbol{w}$ is the model parameter and ϵ is the error term.

1.1 To find $\boldsymbol{w}$, we can minimize the sum of square error

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \left[y(x_n, \boldsymbol{w}) - t_n \right]^2 \tag{2}$$

Now suppose we believe that the distribution of error term ϵ is Gaussian

$$p(\epsilon|\beta) = \mathcal{N}(\epsilon|0, \beta^{-1}) \tag{3}$$

where $\beta = \dfrac{1}{\sigma^2}$ is the inverse of variance. Using the property of Gaussian distribution, we have

$$p(t|x, \boldsymbol{w}, \beta) = \mathcal{N}(t|y(x, \boldsymbol{w}), \beta^{-1}) \tag{4}$$

Under this assumption, the likelihood function is given by

$$p(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n, \boldsymbol{w}), \beta^{-1}) \tag{5}$$

Show that the problem of finding the maximum likelihood solution for $\boldsymbol{w}$ is equivalent to the problem of minimizing the sum of square error (2).

解: 对数似然函数为

$$\begin{aligned} H(\boldsymbol{\omega}) &= \ln p(\boldsymbol{T}|\boldsymbol{X}, \boldsymbol{w}, \beta) \\ &= \ln \left[\prod_{n=1}^{N} \mathcal{N}(t_n|y(x_n, \boldsymbol{w}), \beta^{-1}) \right] \\ &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|y(x_n, \boldsymbol{w}), \beta^{-1}) \\ &= \sum_{n=1}^{N} \ln \left[\frac{1}{\sqrt{2\pi\beta^{-1}}} \exp\left(-\frac{[t_n - y(x_n, \boldsymbol{w})]^2}{2\beta^{-1}} \right) \right] \\ &= \sum_{n=1}^{N} \left(\ln \frac{1}{\sqrt{2\pi\beta^{-1}}} - \frac{\beta}{2}[t_n - y(x_n, \boldsymbol{w})]^2 \right) \\ &= -N \ln \sqrt{2\pi\beta^{-1}} - \frac{\beta}{2} \sum_{n=1}^{N} [t_n - y(x_n, \boldsymbol{w})]^2 \end{aligned} \tag{6}$$

所以, $\boldsymbol{w}$ 的最大似然估计为

$$\hat{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmax}} \left\{ -N \ln \sqrt{2\pi\beta^{-1}} - \frac{\beta}{2} \sum_{n=1}^{N} [t_n - y(x_n, \boldsymbol{w})]^2 \right\} \tag{7}$$

忽略第一项常数项, 上述问题等价于

$$\hat{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmax}} \left\{ -\frac{\beta}{2} \sum_{n=1}^{N} [t_n - y(x_n, \boldsymbol{w})]^2 \right\} \tag{8}$$

又 $\beta > 0$, 则有

$$\hat{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{n=1}^{N} [t_n - y(x_n, \boldsymbol{w})]^2 \right\} \tag{9}$$

对最小二乘法, 有

$$\tilde{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{n=1}^{N} [y(x_n, \boldsymbol{w}) - t_n]^2 \right\} \tag{10}$$

比较式 (9) 与式 (10) 可知 $\tilde{\boldsymbol{w}} = \hat{\boldsymbol{w}}$, 即最大似然估计与最小二乘的结果等价.

1.2 In order to avoid overfitting, we often add a weight decay term to (2)

$$\tilde{E}(\boldsymbol{w}) = \frac{1}{2} \sum_{n=1}^{N} \left[y(x_n, \boldsymbol{w}) - t_n \right]^2 + \frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 \tag{11}$$

On the other hand, we believe that $\boldsymbol{w}$ has a prior distribution of

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1}\boldsymbol{I})$$

Using Bayes theorem, the posterior distribution for $\boldsymbol{w}$ is proportional to the product of the prior distribution and the likelihood function

$$p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{T},\alpha,\beta) \propto p(\boldsymbol{T}|\boldsymbol{X},\boldsymbol{w},\beta)p(\boldsymbol{w}|\alpha) \tag{12}$$

Show that the problem of finding Maximum A Posterior (MAP) solution for $\boldsymbol{w}$ (i.e., maximizing (12)) is equivalent to the problem of minimizing (11).

解: 令

$$\begin{aligned}
\tilde{H}(\boldsymbol{\omega}) &\triangleq \ln\left[p(\boldsymbol{T}|\boldsymbol{X},\boldsymbol{w},\beta)p(\boldsymbol{w}|\alpha)\right] \\
&= \ln\left[\prod_{n=1}^{N}\mathcal{N}(t_n|y(x_n,\boldsymbol{w}),\beta^{-1})p(\boldsymbol{w}|\alpha)\right] \\
&= \sum_{n=1}^{N}\ln\mathcal{N}(t_n|y(x_n,\boldsymbol{w}),\beta^{-1}) + \ln p(\boldsymbol{w}|\alpha) \\
&= \sum_{n=1}^{N}\left(\ln\frac{1}{\sqrt{2\pi\beta^{-1}}} - \frac{[t_n - y(x_n,\boldsymbol{w})]^2}{2\beta^{-1}}\right) + \ln\frac{1}{\sqrt{2\pi\alpha^{-1}}} - \frac{\|\boldsymbol{w}\|_2^2}{2\alpha^{-1}} \\
&= -N\ln\sqrt{2\pi\beta^{-1}} - \ln\sqrt{2\pi\alpha^{-1}} - \frac{\beta}{2}\sum_{n=1}^{N}[t_n - y(x_n,\boldsymbol{w})]^2 - \frac{\alpha}{2}\|\boldsymbol{w}\|_2^2
\end{aligned} \tag{13}$$

所以, $\boldsymbol{w}$ 的最大后验估计 (MAP) 为

$$\hat{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmax}}\left\{-N\ln\sqrt{2\pi\beta^{-1}} - \ln\sqrt{2\pi\alpha^{-1}} - \frac{\beta}{2}\sum_{n=1}^{N}[t_n - y(x_n,\boldsymbol{w})]^2 - \frac{\alpha}{2}\|\boldsymbol{w}\|_2^2\right\} \tag{14}$$

忽略常数项, 上述问题等价于

$$\hat{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmax}}\left\{-\frac{\beta}{2}\sum_{n=1}^{N}[t_n - y(x_n,\boldsymbol{w})]^2 - \frac{\alpha}{2}\|\boldsymbol{w}\|_2^2\right\} \tag{15}$$

又 $\beta > 0$, 则有

$$\hat{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmin}}\left\{\frac{1}{2}\sum_{n=1}^{N}[t_n - y(x_n,\boldsymbol{w})]^2 + \frac{\alpha}{2\beta}\|\boldsymbol{w}\|_2^2\right\} \tag{16}$$

对加入正则项的最小二乘法, 有

$$\tilde{\boldsymbol{w}} \in \underset{\boldsymbol{w}}{\operatorname{argmin}}\left\{\frac{1}{2}\sum_{n=1}^{N}[y(x_n,\boldsymbol{w}) - t_n]^2 + \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2\right\} \tag{17}$$

比较式 (16) 与式 (17) 可知, 当 $\lambda = \dfrac{\alpha}{\beta}$ 时, 有 $\tilde{\boldsymbol{w}} = \hat{\boldsymbol{w}}$, 即最大后验估计与正则化最小二乘的结果等价.

Programming for Error Rate Estimation

2. Consider a two dimensional classification problems: $p(\omega_1) = p(\omega_2) = 0.5$, $p(x|\omega_1) \sim N(\mu_1, \Sigma_1)$, $p(x|\omega_2) \sim N(\mu_2, \Sigma_2)$, where

$$\mu_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix},\ \mu_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},\quad \Sigma_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},\ \Sigma_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{18}$$

2.1 Derive the misclassification rate of the Bayesian classifier theoretically.

解: Bayes 决策边界满足

$$p(x|\omega_1)p(\omega_1) = p(x|\omega_2)p(\omega_2) \tag{19}$$

即

$$\frac{1}{2}\frac{1}{2\pi}\exp\left[-\frac{(x_1+1)^2+x_2^2}{2}\right] = \frac{1}{2}\frac{1}{2\pi}\exp\left[-\frac{(x_1-1)^2+x_2^2}{2}\right] \tag{20}$$

化简得

$$x_1 = 0 \tag{21}$$

所以, 错误率为

$$\begin{aligned} P(e) &= \int_{-\infty}^{\infty}\int_{-\infty}^{0} p(x|\omega_2)p(\omega_2)\mathrm{d}x_1\mathrm{d}x_2 + \int_{-\infty}^{\infty}\int_{0}^{\infty} p(x|\omega_1)p(\omega_1)\mathrm{d}x_1\mathrm{d}x_2 \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{0} \frac{1}{2}\frac{1}{2\pi}\exp\left[-\frac{(x_1-1)^2+x_2^2}{2}\right]\mathrm{d}x_1\mathrm{d}x_2 + \int_{-\infty}^{\infty}\int_{0}^{\infty} \frac{1}{2}\frac{1}{2\pi}\exp\left[-\frac{(x_1+1)^2+x_2^2}{2}\right]\mathrm{d}x_1\mathrm{d}x_2 \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{0} \frac{1}{2\pi}\exp\left[-\frac{(x_1-1)^2+x_2^2}{2}\right]\mathrm{d}x_1\mathrm{d}x_2 \\ &= \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}}\exp\left[-\frac{(x_1-1)^2}{2}\right]\mathrm{d}x_1 \\ &= \Phi(-1) \\ &= 0.1587 \end{aligned} \tag{22}$$

2.2 Choose a proper n and draw n samples from $p(x|\omega_1)$ and $p(x|\omega_2)$ with labels respectively. Estimate $p_n(x|\omega_1)$ and $p_n(x|\omega_2)$ by Parzen window method, with Gaussian window function and unit hypercube window function. Design Bayesian classifier with your estimated $p_n(x|\omega_1)$ and $p_n(x|\omega_2)$. Compare their misclassification rate with the theoretical optimal Bayesian classifier in theory.

解: 取 $n = 1000$, 使用二维高斯核函数

$$K(x) = \frac{1}{2\pi a^2}\exp\left(-\frac{\|x\|_2^2}{2a^2}\right) \tag{23}$$

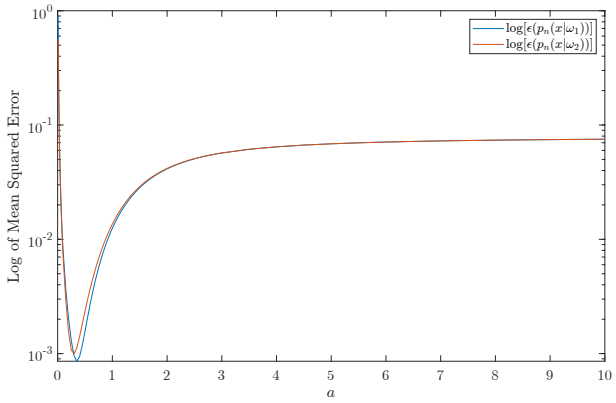


图 1: 高斯估计均方误差与窗宽变化关系

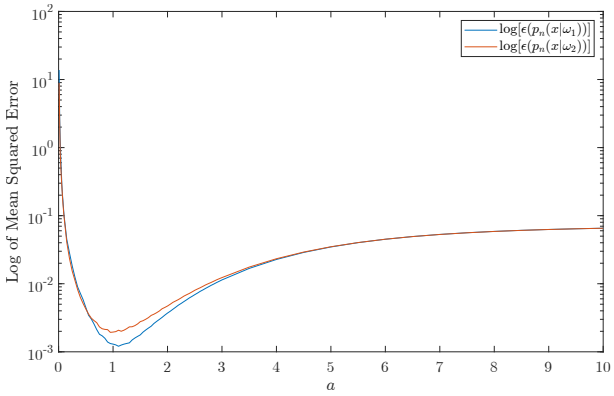


图 2: 方窗估计均方误差与窗宽变化关系

进行估计, 取窗宽 $a = [0.01, 10]$, 做出高斯估计均方误差与窗宽的变化曲线如图 1 所示, 由图可以得出 $n = 1000$ 时高斯估计的最优窗宽为 $a = 0.3$. 在最优窗宽的情况下做 10 次估计, 得到平均均方误差为

$$\epsilon(p_n(x|\omega_1)) = 0.0011, \quad \epsilon(p_n(x|\omega_2)) = 0.0011 \tag{24}$$

使用二维方窗核函数

$$K(x) = \begin{cases} \dfrac{1}{a^2}, & \|x\|_\infty \leqslant \dfrac{a}{2} \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

进行估计, 同样取 $a = [0.01, 10]$, 做出方窗估计均方误差与窗宽的变化曲线如图 2 所示, 由图可以得出 $n = 1000$ 时方窗估计的最优窗宽为 $a = 1$. 在最优窗宽的情况下做 10 次估计, 得到平均均方误差为

$$\epsilon(p_n(x|\omega_1)) = 0.0012, \quad \epsilon(p_n(x|\omega_2)) = 0.0010 \tag{26}$$

由于 $p(\omega_1) = p(\omega_2) = 0.5$, 则 Bayes 分类器为

$$\begin{aligned} p_n(x|\omega_1) > p_n(x|\omega_2) &\to x \in \omega_1 \\ p_n(x|\omega_2) > p_n(x|\omega_1) &\to x \in \omega_2 \end{aligned} \tag{27}$$

在最优窗宽的情况下做 10 次估计, 高斯估计和方窗估计的错误率分别如表 1 和表 2 所示, 平均错误率为

$$\bar{P}_{\text{Guassian}}(e) = 0.1569, \quad \bar{P}_{\text{hypercube}}(e) = 0.1587 \tag{28}$$

理论上, 当样本数量 $n \to \infty$ 时, Parzen 窗法得到的概率密度函数估计是没有偏差的, 则错误率也与 Bayes 理论错误率相同. 但是实际估计时样本数量有限, 估计的概率密度函数与理论值之间存在偏差, 则 Parzen 窗法错误率也与 Bayes 理论错误率存在偏差. 通过实验数据比较两种 Parzen 窗法和 Bayes 的理论错误率, 可以看出方窗的平均错误率与 Bayes 理论错误率几乎相同, 高斯窗的平均错误率比 Bayes 理论错误率略小一些.

表 1: 高斯窗估计错误率

No.	$P_1(e)$	$P_2(e)$	$P(e)$
1	0.1389	0.1746	0.1568
2	0.1641	0.1493	0.1567
3	0.1672	0.1493	0.1582
4	0.1392	0.1754	0.1573
5	0.1623	0.1510	0.1566
6	0.1611	0.1516	0.1563
7	0.1518	0.1624	0.1571
8	0.1568	0.1568	0.1568
9	0.1634	0.1495	0.1564
10	0.1446	0.1694	0.1570

表 2: 方窗估计错误率

No.	$P_1(e)$	$P_2(e)$	$P(e)$
1	0.1671	0.1507	0.1589
2	0.1495	0.1671	0.1583
3	0.1642	0.1510	0.1576
4	0.1609	0.1549	0.1579
5	0.1748	0.1426	0.1587
6	0.1434	0.1733	0.1584
7	0.1549	0.1621	0.1585
8	0.1304	0.1898	0.1601
9	0.1632	0.1538	0.1585
10	0.1699	0.1497	0.1598

2.3 From above experiments, what's your suggestion for choosing optimal window function and parameters with given n?

解: 建议选择窗函数和窗宽参数时, 根据估计的概率密度函数的均方误差和相应的错误率来选择, 综合选择均方误差较小且错误率也较小的窗函数和窗宽参数. 根据上述实验结果可知, 本实验中若选择高斯函数, 且最优窗宽选择 $a = 0.3$, 则此时估计的均方误差和错误率相对于其他测试参数而言都是最小的.

2.4 Sample $2n$ points from the Gaussian mixture distribution $p(x)$ without labels. Use EM to estimate μ_1, μ_2, Σ_1, Σ_2 so that we estimate $p_{2n}(x|\omega_1)$ and $p_{2n}(x|\omega_2)$. Which method is more accurate in estimating $p(x|\omega_1)$ and $p(x|\omega_2)$, EM or Parzen window? Prove your statement by experiments.

解: 使用 EM 算法估计 10 次, 其中似然概率最大的一组估计结果为

$$\hat{\mu}_1 = \begin{bmatrix} -1.0212 \\ -0.0259 \end{bmatrix},\ \hat{\mu}_2 = \begin{bmatrix} 0.9645 \\ -0.0234 \end{bmatrix},\quad \hat{\Sigma}_1 = \begin{bmatrix} 1.0156 & -0.0165 \\ -0.0165 & 1.0035 \end{bmatrix},\ \hat{\Sigma}_2 = \begin{bmatrix} 1.0156 & -0.0165 \\ -0.0165 & 1.0035 \end{bmatrix} \tag{29}$$

且估计的均方误差为

$$\epsilon(p_{2n}(x|\omega_1)) = 0.0003, \quad \epsilon(p_{2n}(x|\omega_2)) = 0.0009 \tag{30}$$

通过均方误差的比较来看, EM 算法最好的估计结果比两种 Parzen 窗的估计效果都更加精确.

2.5 Design Bayesian classifier with the estimated $p_{2n}(x|\omega_1)$ and $p_{2n}(x|\omega_2)$ by EM. Analyze its performance, i.e., the expectation and variance of misclassification rate and compare them with that of optimal Bayesian classifier.

解: 由于 $p(\omega_1) = p(\omega_2) = 0.5$, 则 Bayes 分类器为

$$\begin{aligned} p_{2n}(x|\omega_1) > p_{2n}(x|\omega_2) &\to x \in \omega_1 \\ p_{2n}(x|\omega_2) > p_{2n}(x|\omega_1) &\to x \in \omega_2 \end{aligned} \tag{31}$$

使用 EM 算法估计 10 次, 错误率如表 3 所示.

表 3: EM 算法估计错误率

No.	$P_1(e)$	$P_2(e)$	$P(e)$
1	0.2229	0.3611	0.2920
2	0.1777	0.1732	0.1754
3	0.1819	0.1695	0.1757
4	0.1546	0.1570	0.1558
5	0.1611	0.1504	0.1557
6	0.2014	0.2941	0.2478
7	0.1749	0.1480	0.1615
8	0.1582	0.1528	0.1555
9	0.1782	0.1728	0.1755
10	0.1552	0.1563	0.1558

错误率的期望和方差分别为

$$\mathbb{E}[P(e)] = 0.1851, \quad \mathrm{Var}[P(e)] = 0.0022 \tag{32}$$

由实验结果可知, EM 算法的平均错误率比 Bayes 分类的理论错误率高, 方差较小但还是有一些, 主要是因为 EM 算法偶尔会收敛到局部最优解.

2.6 Conclude your results. Which method is your favorite to estimate parameters and which classifier is your favorite classifier? Why?

解: 这个实验内容包含了之前所学的统计决策方法和概率密度函数的估计两部分内容. 首先, 在概率分布模型以及参数都已知时, 可以直接使用 Bayes 分类, 得到最小错误率的分类器. 若已知概率分布模型, 但是不知道具体参数值, 则可以选择 EM 算法等参数方法进行参数估计, 再进行 Bayes 分类; 当概率分布模型也未知时, 可以选用 Parzen 窗等非参数方法进行估计, 之后进行 Bayes 分类.

相对而言, 我更倾向于选择 EM 算法等参数化方法来估计概率模型的参数并使用 Bayes 分类器, 这是因为 Parzen 窗等非参数方法存在维数爆炸的问题, 同时运算量很大, 还要选择合适的窗函数以及最优窗宽大小; 而参数方法相对比较简单, 且运算量更小.

Programming for Single Sample Correction Algorithm

3. The training process of the Single Sample Correction Algorithm (Algorithm 1) can be regarded as a searching process for a solution in feasible solution region, whereas no strict restrictions are demanded for the capacity of this solution. The solution only needs to satisfy $a^\top y_n > 0$, where a is the weight vector of the perceptron, and y_n is the normalized augmented sample vector. However, the margin perceptron (Algorithm 2) requires the finally converged hyperplane possesses a margin ($> \gamma$), where γ is a predefined positive scalar. It means that the final solution of perceptron need to satisfy $a^\top y_n > \gamma$.

Thus, there are two types of "mistakes" during the training of perceptron, namely (1) the prediction mistake and (2) margin mistake (i.e., its prediction is correct, but its margin is not large enough).

Algorithm 1 Fixed-Increment Single Sample Correction Algorithm

initialize a, $k \leftarrow 0$
repeat
  $k \leftarrow (k+1) \bmod n$
  if y_k is misclassified by a **then**
    $a \leftarrow a + y_k$
  end if
until all patterns are properly classified
return a

Algorithm 2 Single Sample Correction Algorithm With Margin

initialize a, $k \leftarrow 0$, $\gamma > 0$
repeat
  $k \leftarrow (k+1) \bmod n$
  if $a^\top y_k \leqslant \gamma$ **then**
    $a \leftarrow a + y_k$
  end if
until all patterns are properly classified with a large enough margin γ
return a

3.1 Please generate 200 data points in the 2D plane, among which 100 data points are labeled as 1 and the remaining 100 are labeled as -1. Make sure that these 200 data points are linearly separable. Plot these 200 data points in a 2D plane.

解: 设定超平面保证两类数据点线性可分, 生成的 200 个数据点保存在 `percepData.mat` 文件, 如图 3 所示.

3.2 Implement the classical perceptron algorithm and run it on the above generated data points. Plot the classification boundary and these data points in one figure.

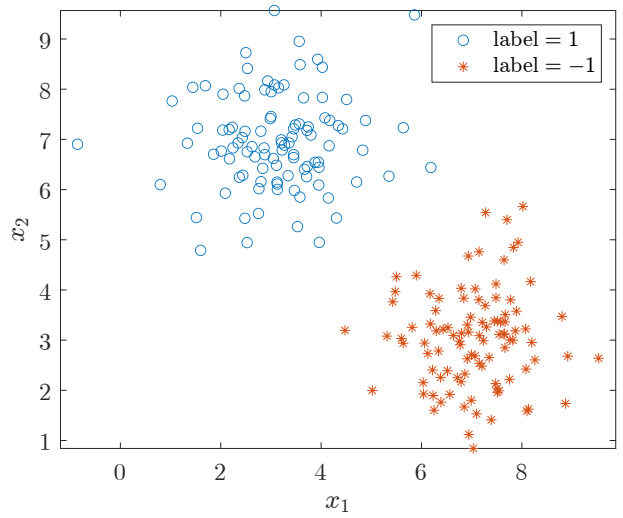


图 3: 200 个数据点, 分为两类

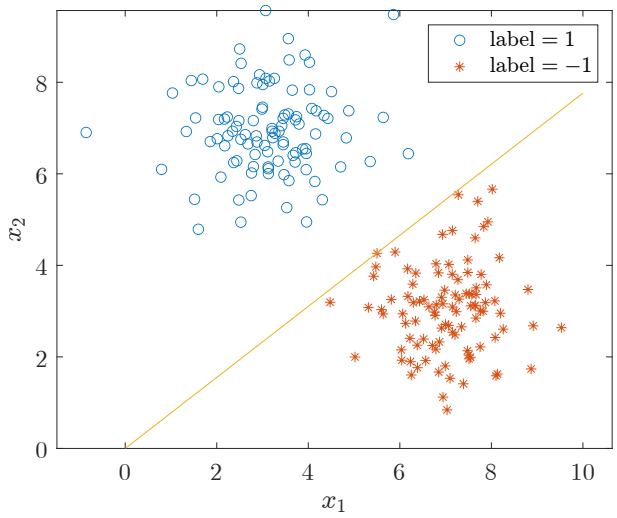


图 4: 经典感知器分类结果

解: 经典感知器的分类边界如图 4 所示.

3.3 Implement the margin perceptron algorithm and run it on the above generated data points. Plot the classification boundary and these data points in one figure. Analyze the impacts of γ on algorithm convergence and the classification boundary.

解: 取 $\gamma = 10$, margin 感知器的分类边界如图 5 所示.

γ 大小对算法迭代次数的影响如图 6 所示, 可知随着 γ 的增大, 算法迭代次数不断震荡波动但是大体上逐渐增加, 因此总得来说增加 margin 对算法收敛速度是不利的.

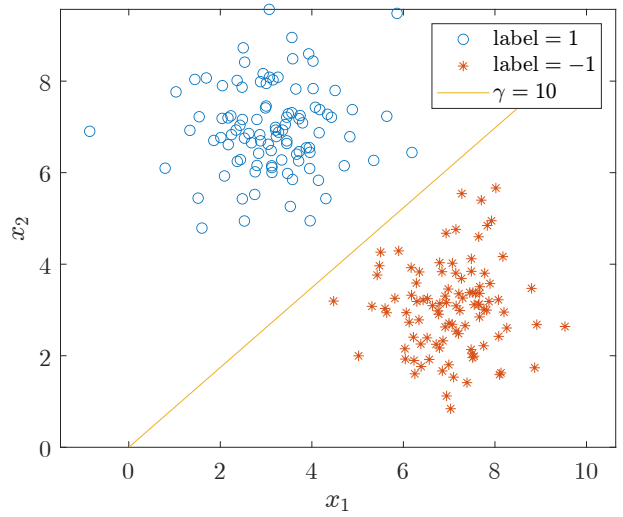


图 5: 感知器分类结果, margin 为 10

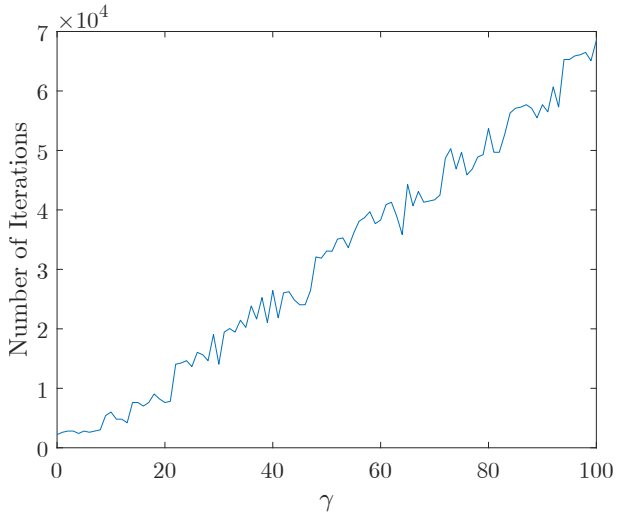


图 6: margin 大小对算法迭代次数的影响

取 $\gamma = 0, 10, \ldots, 80$, 分别绘制分类边界如图 7 所示, 可以看出分类边界的位置随着 γ 的增大而改变, 分类边界到与其最近的数据点的距离随着 γ 的增大而有所增大, 逐渐趋于两类之间居中的位置.

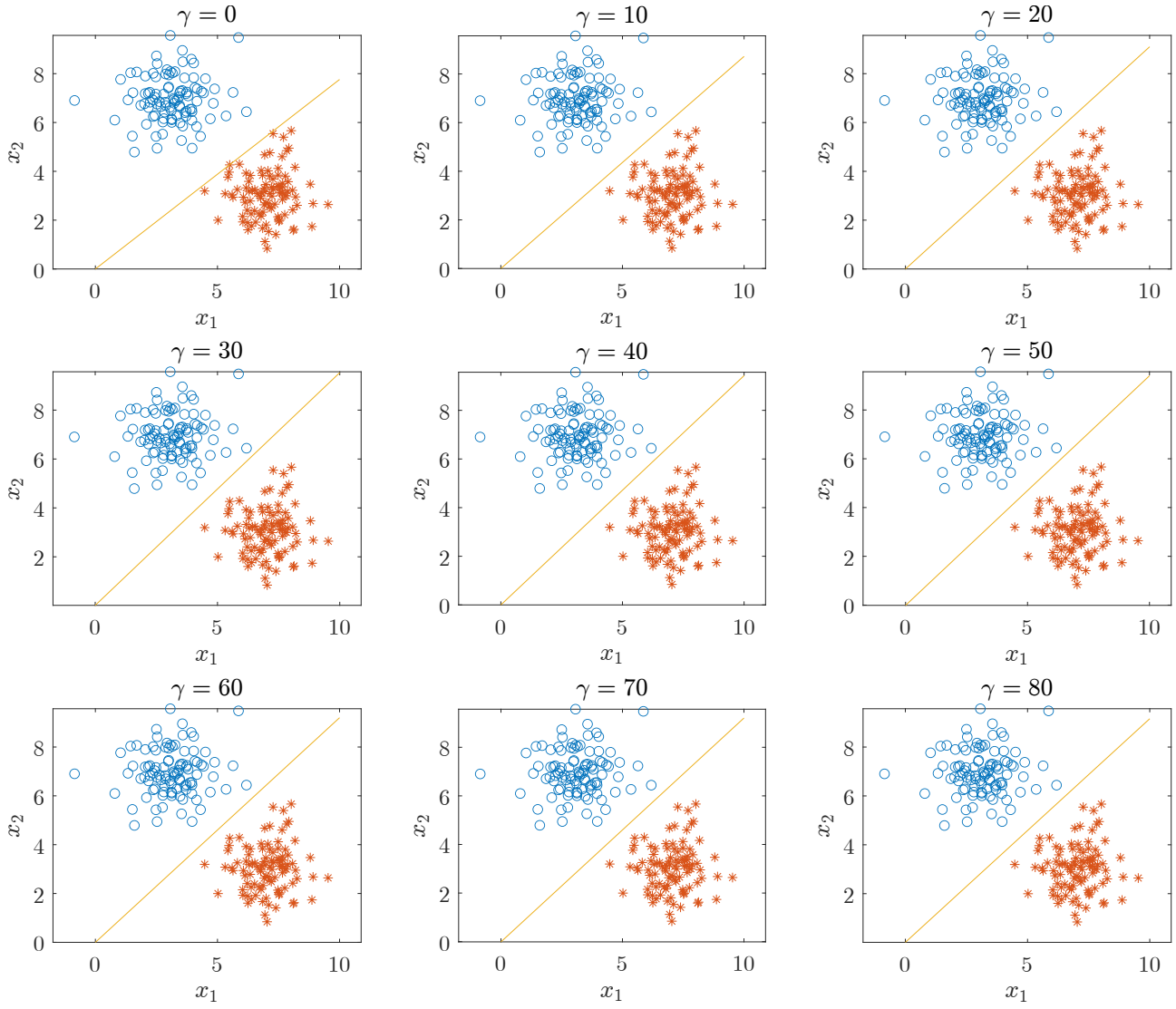


图 7: 感知器分类结果, margin 从 0 变化到 80

Proof of Single Sample Correction Algorithm

4. Suppose we have N points x_i in $\mathbb{R}^p$ with class labels $\omega_i \in \{-1, 1\}$. Prove that the perceptron algorithm converges to a separating hyperplane in finite steps:

- Denoting a hyperplane by $f(x) = a_1^\top x + a_0 = 0$, or in more compact notation $a^\top y = 0$, where $y = (x, 1)$ and $a = (a_1, a_0)$. Let $z_i = \dfrac{y}{\|y\|_2}$. Show that separability implies there exists a a_{sep} such that $\omega_i a_{\text{sep}}^\top z_i \geqslant 1$, $\forall\ i = 1, 2, \ldots, N$.
- Given a current a_{old}, the Algorithm 1 identifies a point z_i that is misclassified, and produces the update $a_{\text{new}} = a_{\text{old}} + \omega_i z_i$. Show that $\|a_{\text{new}} - a_{\text{sep}}\|_2^2 \leqslant \|a_{\text{old}} - a_{\text{sep}}\|_2^2 - 1$, and hence the algorithm converges to a separating hyperplane in no more than $\|a_{\text{start}} - a_{\text{sep}}\|_2^2$ steps.

证明: 假设数据是线性可分的, 则 $\exists\ \bar{a} \in \mathbb{R}^{p+1}$ 使得

$$\omega_i \bar{a}^\top y_i > 0, \quad \forall\ i = 1, 2, \ldots, N \tag{33}$$

其中 $y_i = (x_i, 1)$, 则 $\|y_i\|_2 > 0, \forall\ i = 1, 2, \ldots, N$, 所以

$$\omega_i \bar{a}^\top \frac{y_i}{\|y_i\|_2} > 0, \quad \forall\ i = 1, 2, \ldots, N \tag{34}$$

令 $z_i = \dfrac{y_i}{\|y_i\|_2}$, 则有

$$\omega_i \bar{a}^\top z_i > 0, \quad \forall\ i = 1, 2, \ldots, N \tag{35}$$

所以

$$\omega_i \bar{a}^\top z_i \geqslant \min_i \{\omega_i \bar{a}^\top z_i\} > 0, \quad \forall\ i = 1, 2, \ldots, N \tag{36}$$

因此

$$\frac{\omega_i \bar{a}^\top z_i}{\min_i \{\omega_i \bar{a}^\top z_i\}} \geqslant 1, \quad \forall\ i = 1, 2, \ldots, N \tag{37}$$

取

$$a_{\text{sep}} = \frac{\bar{a}}{\min_i \{\omega_i \bar{a}^\top z_i\}}$$

则有

$$\omega_i a_{\text{sep}}^\top z_i \geqslant 1, \quad \forall\ i = 1, 2, \ldots, N \tag{38}$$

假设点 z_i 被 a_{old} 错分, 即有

$$\omega_i a_{\text{old}}^\top z_i < 0 \tag{39}$$

则算法下一步更新

$$a_{\text{new}} = a_{\text{old}} + \omega_i z_i \tag{40}$$

结合以上三式可得

$$\begin{aligned} \|a_{\text{new}} - a_{\text{sep}}\|_2^2 &= \|a_{\text{old}} - a_{\text{sep}} + \omega_i z_i\|_2^2 \\ &= \|a_{\text{old}} - a_{\text{sep}}\|_2^2 + \|\omega_i z_i\|_2^2 + 2\omega_i (a_{\text{old}} - a_{\text{sep}})^\top z_i \\ &= \|a_{\text{old}} - a_{\text{sep}}\|_2^2 + 1 + 2\omega_i a_{\text{old}}^\top z_i - 2\omega_i a_{\text{sep}}^\top z_i \\ &\leqslant \|a_{\text{old}} - a_{\text{sep}}\|_2^2 + 1 + 0 - 2 \\ &= \|a_{\text{old}} - a_{\text{sep}}\|_2^2 - 1 \end{aligned} \tag{41}$$

假设算法从 a_{start} 开始, 经 n 步更新得到 $a^{(n)}$, 则由上式可知

$$0 \leqslant \|a^{(n)} - a_{\text{sep}}\|_2^2 \leqslant \|a^{(n-1)} - a_{\text{sep}}\|_2^2 - 1 \leqslant \ldots \leqslant \|a_{\text{start}} - a_{\text{sep}}\|_2^2 - n \tag{42}$$

因此有

$$n \leqslant \|a_{\text{start}} - a_{\text{sep}}\|_2^2 \tag{43}$$

即算法更新次数不会超过 $\|a_{\text{start}} - a_{\text{sep}}\|_2^2$, 所以算法不超过 $\|a_{\text{start}} - a_{\text{sep}}\|_2^2$ 步就会收敛到一个分离超平面. □

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 5**

SVM

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Problem 1

Given an unlabeled set of examples $\left\{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\right\}$ the one-class SVM algorithm tries to find a direction ω that maximally separates the data from the origin. (This turns out to be useful for anomaly detection.)

More precisely, it solves the (primal) optimization problem:

$$\begin{aligned} \min_{\omega} \ & \frac{1}{2}\omega^{\top}\omega \\ \text{s.t.} \ & \omega^{\top}x^{(i)} \geqslant 1, \ i = 1, 2, \ldots, n \end{aligned} \tag{1}$$

A new test example x is labeled 1 if $\omega^{\top}x \geqslant 1$, and 0 otherwise.

1.1 The primal optimization problem for the one-class SVM was given above. Write down the corresponding dual optimization problem. Simplify your answer as much as possible. In particular, ω should not appear in your answer.

解: Lagrange 函数为

$$L(\omega, \alpha) = \frac{1}{2}\omega^{\top}\omega - \sum_{i=1}^{n}\alpha_i(\omega^{\top}x^{(i)} - 1) \tag{2}$$

其对 ω 的偏导数为

$$\frac{\partial L(\omega, \alpha)}{\partial \omega} = \omega - \sum_{i=1}^{n}\alpha_i x^{(i)} \tag{3}$$

令此偏导数为 0, 可得

$$\omega = \sum_{i=1}^{n}\alpha_i x^{(i)} \tag{4}$$

代入 Lagrange 函数, 可得对偶函数为

$$Q(\alpha) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j[x^{(i)}]^\top x^{(j)} + \sum_{i=1}^{n}\alpha_i \tag{5}$$

所以, 对偶问题为

$$\begin{aligned} \max_{\alpha}\ & Q(\alpha) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j[x^{(i)}]^\top x^{(j)} + \sum_{i=1}^{n}\alpha_i \\ \text{s.t.}\quad & \alpha_i \geqslant 0,\ i = 1, 2, \ldots, n \end{aligned} \tag{6}$$

1.2 Can the one-class SVM be kernelized (both in training and testing)? Justify your answer.

解: 由于样本点只以内积形式存在, 则可以使用核函数进行训练和测试.

记核函数为 $K(\cdot,\cdot)$, 则在训练时需要解如下问题

$$\begin{aligned} \max_{\alpha}\ & Q(\alpha) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j K(x^{(i)}, x^{(j)}) + \sum_{i=1}^{n}\alpha_i \\ \text{s.t.}\quad & \alpha_i \geqslant 0,\ i = 1, 2, \ldots, n \end{aligned} \tag{7}$$

求得上式最优解 α^* 后, 对新样本 x 测试时计算

$$f(x) = \sum_{i=1}^{n}\alpha_i^* K(x^{(i)}, x) \tag{8}$$

若 $f(x) \geqslant 1$ 则将 x 标记为 1, 否则标记为 0.

Problem 2

Consider a dataset with 2 points in 1-D: $x_1 = 0$, $x_2 = \sqrt{2}$ with labels $y_1 = -1$, $y_2 = 1$. Consider mapping each point to 3-D using the feature vector $\Phi = [1, \sqrt{2}x, x^2]^\top$. (This is equivalent to using a second order polynomial kernel.) The max margin classifier has the form:

$$\begin{aligned} \min_{w}\ & \|w\|^2 \\ \text{s.t.}\quad & y_1(w^\top\Phi(x_1) + b) \geqslant 1 \\ & y_2(w^\top\Phi(x_2) + b) \geqslant 1 \end{aligned} \tag{9}$$

2.1. Write down a vector that is parallel to the optimal vector w.

解: 与最优解 w 平行的一个向量可以为

$$\bar{w} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \tag{10}$$

2.2. What is the value of the margin that is achieved by this w? Hint 1: recall that the margin is the distance from each support vector to the decision boundary. Hint 2: think about the geometry of 2 points in space, with a line separating one from the other.

解: 根据 Hint 1 对 margin 的定义可知

$$\text{margin} = \sqrt{2} \tag{11}$$

2.3. Solve for w, using the fact the margin is equal to $1/\|w\|$.

解: 由 margin 值可知

$$\|w\| = \frac{1}{\sqrt{2}} \tag{12}$$

则

$$w = \frac{\|w\|}{\|\bar{w}\|}\bar{w} = \begin{bmatrix} 0 \\ \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} \tag{13}$$

2.4. Write down the form of the discriminant function $f(x) = w^\top \Phi(x) + b$ as an explicit function of x.

解: 由上题可知

$$f(x) = w^\top \Phi(x) + b = \frac{\sqrt{2}}{2}x + \frac{1}{2}x^2 + b \tag{14}$$

又 $y_1 f(x_1) = y_2 f(x_2) = 1$ 可知

$$b = -1 \tag{15}$$

则判别函数为

$$f(x) = \frac{1}{2}x^2 + \frac{\sqrt{2}}{2}x - 1 \tag{16}$$

Problem 3

The exclusive-OR is the simplest problem that cannot be solved using a linear discriminant operating directly on the features. The points $k = 1, 3$ at $\boldsymbol{x} = (1, 1)^\top$ and $\boldsymbol{x} = (-1, -1)^\top$ are in category ω_1 (red in Figure 1), while $k = 2, 4$ at $\boldsymbol{x} = (1, -1)^\top$ and $\boldsymbol{x} = (-1, 1)^\top$ are in ω_2 (black in Figure 1). Following the approach of SVM, we use kernel function to map the features to a higher dimension space where they can be linearly separated.

3.1 Consider the polynomial kernel of degree 2:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x_i}^\top \boldsymbol{x_j} + 1)^2, \tag{17}$$

where $\boldsymbol{x}_i = (x_{i1}, x_{i2})^\top$ and $\boldsymbol{x}_j = (x_{j1}, x_{j2})^\top$.

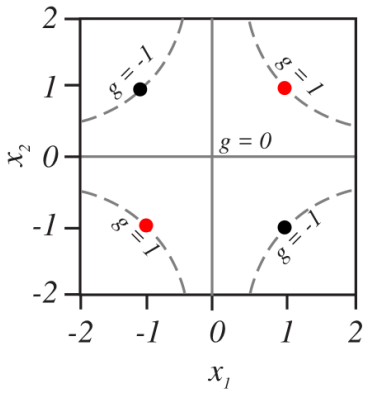


图 1: The XOR problem in the original (x_1, x_2) feature space.

Show that it corresponds to mapping

$$\Phi(x_1, x_2) = \left[1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, x_1^2, x_2^2\right]. \tag{18}$$

解: 对核函数有

$$\begin{aligned} K(\boldsymbol{x}_i, \boldsymbol{x}_j) &= (\boldsymbol{x_i}^\top \boldsymbol{x_j} + 1)^2 \\ &= (x_{i1}x_{j1} + x_{i2}x_{j2} + 1)^2 \\ &= 1 + x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 2x_{i1}x_{j1}x_{i2}x_{j2} \end{aligned} \tag{19}$$

又映射的内积为

$$\Phi(x_{i1}, x_{i2}) \cdot \Phi(x_{j1}, x_{j2}) = 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i1}^2x_{j1}^2 + x_{i2}^2x_{j2}^2 \tag{20}$$

则

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \Phi(x_{i1}, x_{i2}) \cdot \Phi(x_{j1}, x_{j2}) \tag{21}$$

因此核函数对应的映射为 $\Phi(\cdot)$.

3.2 Derive the dual problem in the 6-dimensional space with Lagrange multipliers α_i, $i = 1, 2, 3, 4$ as the only variants.

解: 记 $y_1 = 1$, $y_2 = -1$, $y_3 = 1$, $y_4 = -1$,

$$\boldsymbol{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \quad \boldsymbol{x}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \boldsymbol{x}_3 = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \boldsymbol{x}_4 = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \tag{22}$$

则原问题为

$$\begin{aligned} &\min_{\boldsymbol{\omega}} \ \frac{1}{2}\|\boldsymbol{\omega}\|_2^2 \\ &\text{s.t.} \quad y_i[\boldsymbol{\omega}^\top \Phi(\boldsymbol{x}_i) - b] - 1 \geqslant 0, \ i = 1, 2, 3, 4 \end{aligned} \tag{23}$$

引入对偶变量 $\boldsymbol{\alpha}$, 则 Lagrange 函数为

$$L(\boldsymbol{\omega}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\omega}\|_2^2 - \sum_{i=1}^{4} \alpha_i [y_i \boldsymbol{\omega}^\top \Phi(\boldsymbol{x}_i) - y_i b - 1] \tag{24}$$

其对 $\boldsymbol{\omega}$ 的偏导数为

$$\frac{\partial L(\boldsymbol{\omega}, b, \boldsymbol{\alpha})}{\partial \boldsymbol{\omega}} = \boldsymbol{\omega} - \sum_{i=1}^{4} \alpha_i y_i \Phi(\boldsymbol{x}_i) \tag{25}$$

令此偏导数为 0, 可得

$$\boldsymbol{\omega} = \sum_{i=1}^{4} \alpha_i y_i \Phi(\boldsymbol{x}_i) \tag{26}$$

Lagrange 函数对 b 的偏导数为

$$\frac{\partial L(\boldsymbol{\omega}, b, \boldsymbol{\alpha})}{\partial b} = \sum_{i=1}^{4} \alpha_i y_i \tag{27}$$

令此偏导数为 0, 可得

$$\sum_{i=1}^{4} \alpha_i y_i = 0 \tag{28}$$

将 $\boldsymbol{\omega}$ 代入 Lagrange 函数, 可得对偶函数为

$$Q(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) + \sum_{i=1}^{4} \alpha_i \tag{29}$$

所以, 对偶问题为

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \ & Q(\boldsymbol{\alpha}) = -\frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) + \sum_{i=1}^{4} \alpha_i \\ \text{s.t.} \quad & \sum_{i=1}^{4} \alpha_i y_i = 0 \\ & \alpha_i \geqslant 0, \ i = 1, 2, 3, 4 \end{aligned} \tag{30}$$

3.3 Solve the dual problem analytically (without programming). What are the support vectors? Hint: use the symmetry of the problem.

解: 对偶函数为

$$\begin{aligned} Q(\boldsymbol{\alpha}) &= -\frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j \Phi(\boldsymbol{x}_i) \cdot \Phi(\boldsymbol{x}_j) + \sum_{i=1}^{4} \alpha_i \\ &= \sum_{i=1}^{4} \alpha_i - \frac{1}{2} \sum_{i=1}^{4} \sum_{j=1}^{4} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{31}$$

其中 $\forall\ i, j = 1, 2, 3, 4$, 核函数

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} 9, & \text{if } i = j \\ 1, & \text{otherwise} \end{cases} \tag{32}$$

则对偶函数对 α_i 的偏导为

$$\begin{aligned} \frac{\partial Q(\boldsymbol{\alpha})}{\partial \alpha_i} &= 1 - \alpha_i y_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}_i) - \sum_{j \neq i} \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= 1 - 9\alpha_i - y_i \sum_{j \neq i} \alpha_j y_j, \quad \forall\ i = 1, 2, 3, 4 \end{aligned} \tag{33}$$

又由对偶问题约束可知

$$\sum_{j \neq i} \alpha_j y_j = \sum_{j=1}^{4} \alpha_j y_j - \alpha_i y_i = -\alpha_i y_i, \quad \forall\ i = 1, 2, 3, 4 \tag{34}$$

代入偏导数表达式可得

$$\frac{\partial Q(\boldsymbol{\alpha})}{\partial \alpha_i} = 1 - 9\alpha_i + \alpha_i y_i y_i = 1 - 8\alpha_i, \quad \forall\ i = 1, 2, 3, 4 \tag{35}$$

令这些偏导数为 0, 有

$$\alpha_i = \frac{1}{8}, \quad \forall\ i = 1, 2, 3, 4 \tag{36}$$

则由互补松弛 (complementary slackness) 性可知, $\boldsymbol{x}_i$, $i = 1, 2, 3, 4$ 都是支持向量.

3.4 Derive the final discriminant function $g(\boldsymbol{x})$ and the decision hyperplane.

解: 由上题可知

$$\begin{aligned} \boldsymbol{\omega} &= \sum_{i=1}^{4} \alpha_i y_i \Phi(\boldsymbol{x}_i) \\ &= \frac{1}{8} \left[\begin{pmatrix} 1 \\ \sqrt{2} \\ \sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ \sqrt{2} \\ -\sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} + \begin{pmatrix} 1 \\ -\sqrt{2} \\ -\sqrt{2} \\ \sqrt{2} \\ 1 \\ 1 \end{pmatrix} - \begin{pmatrix} 1 \\ -\sqrt{2} \\ \sqrt{2} \\ -\sqrt{2} \\ 1 \\ 1 \end{pmatrix} \right] \\ &= \frac{1}{8} \begin{pmatrix} 0 \\ 0 \\ 0 \\ 4\sqrt{2} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{\sqrt{2}}{2} \\ 0 \\ 0 \end{pmatrix} \end{aligned} \tag{37}$$

所以

$$\begin{aligned} g(\boldsymbol{x}) &= \boldsymbol{\omega}^\top \Phi(\boldsymbol{x}) - b \\ &= x_1 x_2 - b \end{aligned} \tag{38}$$

又 $\boldsymbol{x}_i$, $i = 1, 2, 3, 4$ 都是支持向量, 则有

$$y_i g(\boldsymbol{x}_i) - 1 = 0, \quad \forall\, i = 1, 2, 3, 4 \tag{39}$$

从而解得

$$b = 0 \tag{40}$$

因此, 判别函数为

$$g(\boldsymbol{x}) = x_1 x_2 \tag{41}$$

决策超平面为

$$g(\boldsymbol{x}) = x_1 x_2 = 0 \tag{42}$$

3.5 Plot the data points on the subspace of $(\sqrt{2}x_1, \sqrt{2}x_1x_2)$. Demonstrate the decision hyperplane and the margin in your plot.

解: 映射到子空间 $(\sqrt{2}x_1, \sqrt{2}x_1x_2)$ 的四个数据点, 决策超平面以及间隔如图 2 所示.

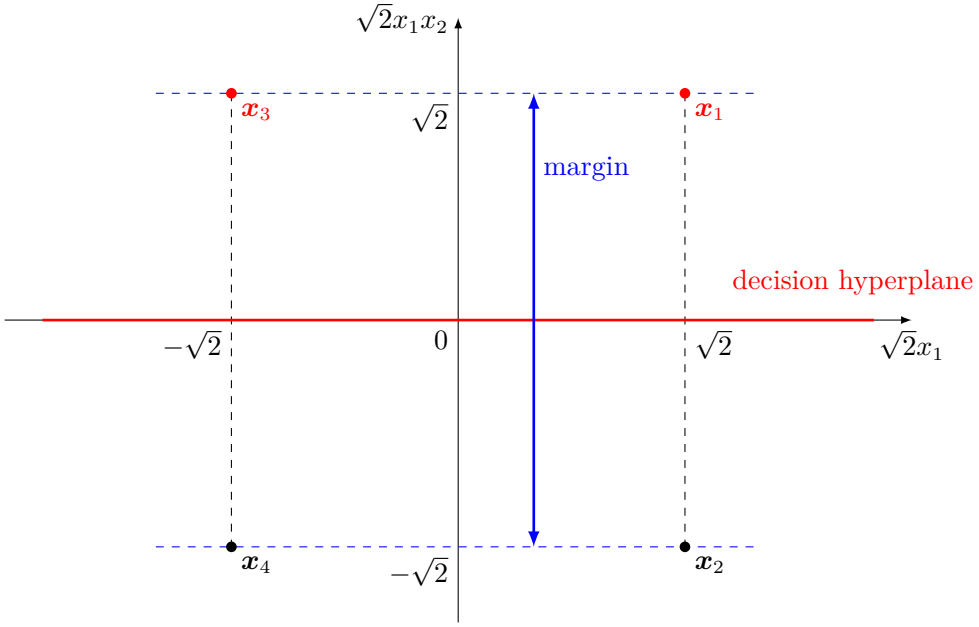


图 2: XOR 问题映射到子空间 $(\sqrt{2}x_1, \sqrt{2}x_1x_2)$, 决策超平面以及间隔

Problem 4

Write a program to complete the following task adopting the SVM algorithm (you could use some toolkits or source code). Train a SVM classifier with data from ω_1 and ω_2 in the following table. Preprocess each training pattern to form a new vector having components $1, x_1, x_2, x_1^2, x_1x_2$ and x_2^2.

	ω_1		ω_2	
sample	x_1	x_2	x_1	x_2
1	-3.0	-2.9	-2.0	-8.4
2	0.5	8.7	-8.9	0.2
3	2.9	2.1	-4.2	-7.7
4	-0.1	5.2	-8.5	-3.2
5	-4.0	2.2	-6.7	-4.0
6	-1.3	3.7	-0.5	-9.2
7	-3.4	6.2	-5.3	-6.7
8	-4.1	3.4	-8.7	-6.4
9	-5.1	1.6	-7.1	-9.7
10	1.9	5.1	-8.0	-6.3

Hint: You needn't program the SVM algorithm by yourself, you can just use some toolkits or source code such as libsvm for MATLAB or scikit-learn for python. You should declare the toolkit you used in your project.

4.1 Train you classifier with just the first point in the ω_1 and ω_2 and find the separating hyperplane and the margin.

解: 使用 MATLAB 内置 `fitcsvm` 函数, 对两个类别的第一个点训练 SVM 得到分类边界如图 3 中蓝色实线所示, 分类超平面的表达式为

$$g(\boldsymbol{x}) = w_1 + w_2x_1 + w_3x_2 + w_4x_1^2 + w_5x_1x_2 + w_6x_2^2 + b = 0 \tag{43}$$

其中 w_i, $i = 1, 2, \ldots, 6$ 的数值见表 1 中第一行, 且间隔 (margin) 为

$$\text{margin} = 63.1228 \tag{44}$$

4.2 Repeat 4.1 with the first two points in each category (four points total), the first three points and so on, until the transformed patterns cannot be linearly separated in the transformed space.

解: 依次使用前 2 至前 10 个点训练 SVM 得到分类边界如图 4 所示, 分类超平面的参数以及间隔如表 1 所示. 由图可知依次生成的训练集数据在转换后的空间上均是线性可分的, 随着训练点数的增多, 间隔逐渐减小, 但是如果新增的点不是支持向量, 则对分类面没有影响.

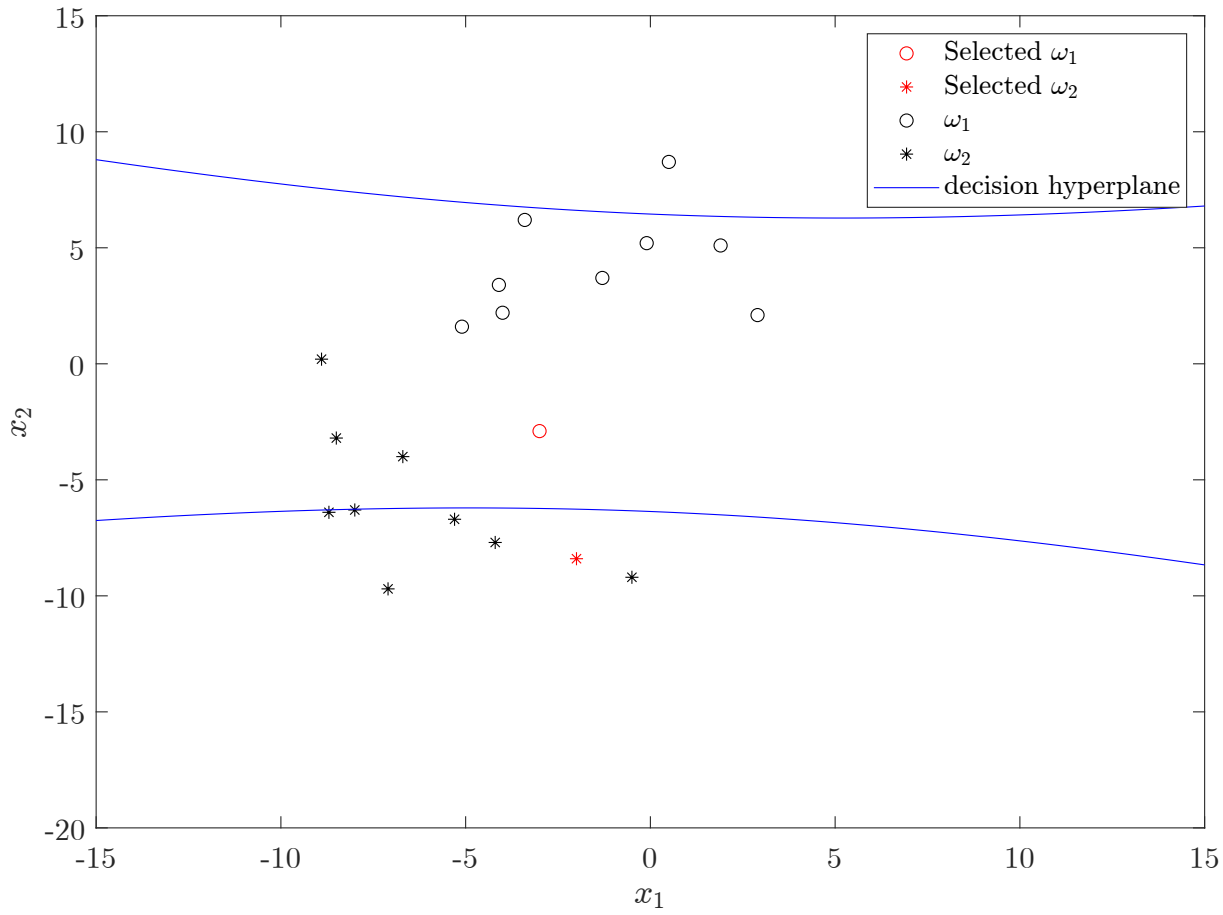


图 3: 使用两类第一个点训练 SVM 得到分类边界

表 1: 分别使用前 1 至 10 个点训练 SVM 得到分类超平面参数和间隔

# of Points	w_1	w_2	w_3	w_4	w_5	w_6	b	margin
1	0	-0.0005	0.0028	0.0025	-0.0041	-0.0312	1.2816	63.1228
2	0	0.0121	0.0739	-0.0413	-0.0534	-0.0222	2.2739	19.3674
3	0	0.0121	0.0739	-0.0413	-0.0534	-0.0222	2.2739	19.3674
4	0	0.0121	0.0739	-0.0413	-0.0534	-0.0222	2.2739	19.3674
5	0	0.0121	0.0739	-0.0413	-0.0534	-0.0222	2.2740	19.3686
6	0	0.0111	0.0989	-0.0379	-0.0210	-0.0239	2.0453	17.9957
7	0	0.0111	0.0989	-0.0379	-0.0210	-0.0239	2.0453	17.9957
8	0	0.0111	0.0989	-0.0379	-0.0210	-0.0239	2.0453	17.9957
9	0	0.0111	0.0989	-0.0379	-0.0210	-0.0239	2.0451	17.9976
10	0	0.0111	0.0989	-0.0379	-0.0210	-0.0239	2.0451	17.9976

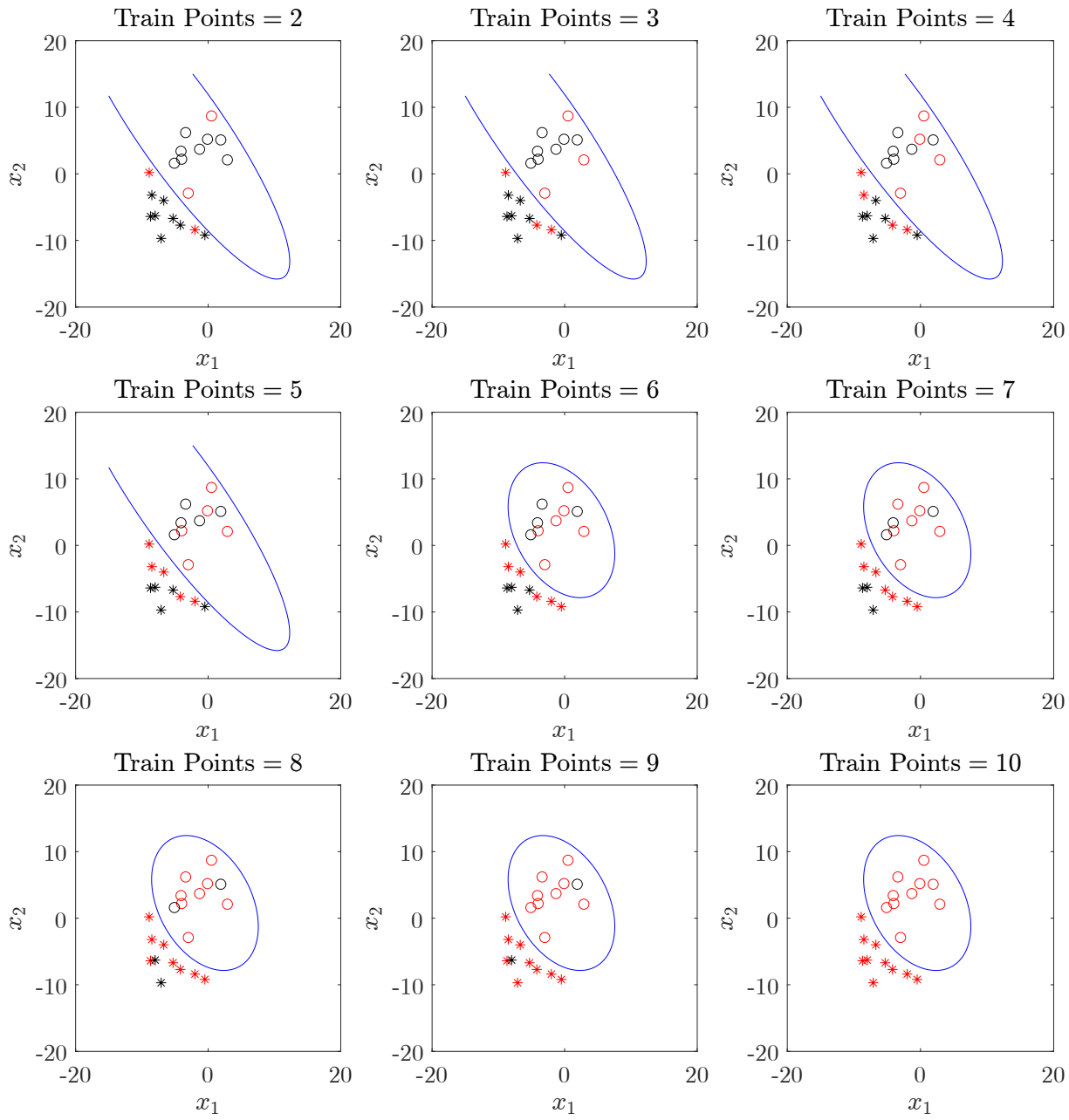


图 4: 依次使用前 2 至前 10 个点训练 SVM 得到分类边界

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 6**

k-NN and Metric

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Property of Euclidean Distance

1. When the metric space is a finite-dimensional Euclidean space, please prove that the Voronoi cells induced by the single-nearest neighbor algorithm must always be convex. Does this property hold when the metric becomes Manhattan distance?

In mathematics, a Voronoi diagram is a partition of a plane into regions close to each of a given set of objects. In the simplest case, these objects are just finitely many points in the plane (called seeds, sites, or generators). For each seed there is a corresponding region consisting of all points of the plane closer to that seed than to any other. These regions are called Voronoi cells, as shown in Figure 1. Similarly, Voronoi cells of a discrete set in higher-order Euclidean space are known as generalized polyhedra.

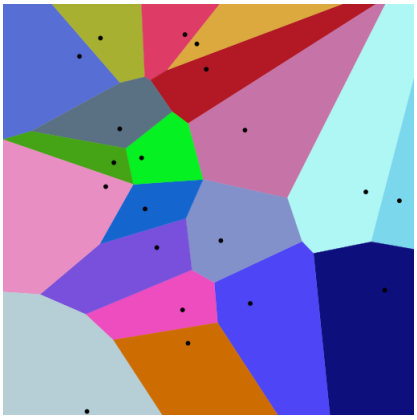


图 1: Voronoi cells with Euclidean distance [1].

Hint: Convex means for any two points x_1 and x_2 in a cell, all points on the segment linking x_1 and x_2 must also lie in the cell.

证明: 令 $\{x_1, x_2, \ldots, x_n\}$ 为 n 个样本点, V_i, $i = 1, 2, \ldots, n$ 表示 x_i 所在的由 ℓ_2 范数 (Euclidean Distance) 最近邻法生成的 Voronoi 区域. 设 ξ_1, $\xi_2 \in V_i$, 即有

$$\begin{aligned} \|\xi_1 - x_i\| &\leqslant \|\xi_1 - x_j\|, \ \forall\, j \neq i \\ \|\xi_2 - x_i\| &\leqslant \|\xi_2 - x_j\|, \ \forall\, j \neq i \end{aligned} \tag{1}$$

对 $\xi = \lambda\xi_1 + (1-\lambda)\xi_2$, $0 \leqslant \lambda \leqslant 1$, $\forall\, j \neq i$, 由上式及余弦定理

$$\langle a, b\rangle = a^\top b = \frac{\|a\|^2 + \|b\|^2 - \|a-b\|^2}{2} \tag{2}$$

可得

$$\begin{aligned} \|\xi - x_i\|^2 &= \|\lambda\xi_1 + (1-\lambda)\xi_2 - x_i\|^2 \\ &= \|\lambda(\xi_1 - x_i) + (1-\lambda)(\xi_2 - x_i)\|^2 \\ &= \lambda^2\|\xi_1 - x_i\|^2 + 2\lambda(1-\lambda)(\xi_1 - x_i)^\top(\xi_2 - x_i) + (1-\lambda)^2\|(\xi_2 - x_i)\|^2 \\ &= \lambda^2\|\xi_1 - x_i\|^2 + \lambda(1-\lambda)(\|\xi_1 - x_i\|^2 + \|\xi_2 - x_i\|^2 - \|\xi_1 - \xi_2\|^2) + (1-\lambda)^2\|(\xi_2 - x_i)\|^2 \\ &= \lambda\|\xi_1 - x_i\|^2 - \lambda(1-\lambda)\|\xi_1 - \xi_2\|^2 + (1-\lambda)\|(\xi_2 - x_i)\|^2 \\ &\leqslant \lambda\|\xi_1 - x_j\|^2 - \lambda(1-\lambda)\|\xi_1 - \xi_2\|^2 + (1-\lambda)\|\xi_2 - x_j\|^2 \\ &= \|\xi - x_j\|^2 \end{aligned} \tag{3}$$

即

$$\|\xi - x_i\| \leqslant \|\xi - x_j\|, \ \forall\, j \neq i \tag{4}$$

因此, $\xi \in V_i$, 即由 ℓ_2 范数最近邻法生成的 Voronoi 区域都是凸的.

由 ℓ_1 范数 (Manhattan Distance) 最近邻法生成的 Voronoi 区域不是凸的, 如图 2 所示.

图 2: Voronoi cells with Manhattan distance [2].

Properties of Metric

2. Please prove that the Minkowski metric indeed possesses the three properties required of all metrics.

Hint: A metric $D(\cdot,\cdot)$ must have three properties: for all vectors a, b and c,

(1) Identity of indiscernibles: $D(a,b)=0$ if and only if $a=b$.

(2) Symmetry: $D(a,b)=D(b,a)$.

(3) Triangle inequality: $D(a,b)+D(b,c)\geqslant D(a,c)$.

证明: s ($s\geqslant 1$) 阶 Minkowski 距离为

$$D(x,y)=\left(\sum_{j=1}^{d}|x_j-y_j|^s\right)^{1/s} \tag{5}$$

首先证明 identity of indiscernibles, $\forall\ x,y$, 当 $x=y$ 时, 有

$$D(x,y)=D(x,x)=\left(\sum_{j=1}^{d}|x_j-x_j|^s\right)^{1/s}=0 \tag{6}$$

当 $D(x,y)=0$ 时, 有

$$\sum_{j=1}^{d}|x_j-y_j|^s=0 \tag{7}$$

由于每一项绝对值均非负, 则

$$x_j=y_j,\quad \forall\ j=1,2,\ldots,d \tag{8}$$

即 $x=y$.

其次证明对称性, $\forall\ x,y$, 有

$$D(x,y)=\left(\sum_{j=1}^{d}|x_j-y_j|^s\right)^{1/s}=\left(\sum_{j=1}^{d}|y_j-x_j|^s\right)^{1/s}=D(y,x) \tag{9}$$

下面证明三角不等式, $\forall\ x,y,z$, 由绝对值性质有

$$\begin{aligned} D(x,y)&=\left(\sum_{j=1}^{d}|x_j-y_j|^s\right)^{1/s}\\ &=\left(\sum_{j=1}^{d}|x_j-z_j+z_j-y_j|^s\right)^{1/s}\\ &\leqslant\left(\sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)^s\right)^{1/s} \end{aligned} \tag{10}$$

其中, 由 Hölder 不等式可得

$$\begin{aligned}\sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)^s &= \sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)\cdot(|x_j-z_j|+|z_j-y_j|)^{s-1}\\ &= \sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)\cdot(|x_j-z_j|+|z_j-y_j|)^{s/t},\quad \frac{1}{s}+\frac{1}{t}=1\\ &= \sum_{j=1}^{d}|x_j-z_j|\cdot(|x_j-z_j|+|z_j-y_j|)^{s/t}+\sum_{j=1}^{d}|z_j-y_j|\cdot(|x_j-z_j|+|z_j-y_j|)^{s/t}\\ &\leqslant \left[\left(\sum_{j=1}^{d}|x_j-z_j|^s\right)^{1/s}+\left(\sum_{j=1}^{d}|z_j-y_j|^s\right)^{1/s}\right]\left(\sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)^s\right)^{1/t}\end{aligned} \tag{11}$$

整理即得

$$\left(\sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)^s\right)^{1-1/t}\leqslant\left(\sum_{j=1}^{d}|x_j-z_j|^s\right)^{1/s}+\left(\sum_{j=1}^{d}|z_j-y_j|^s\right)^{1/s} \tag{12}$$

又 $1-\frac{1}{t}=\frac{1}{s}$, 则

$$\left(\sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)^s\right)^{1/s}\leqslant\left(\sum_{j=1}^{d}|x_j-z_j|^s\right)^{1/s}+\left(\sum_{j=1}^{d}|z_j-y_j|^s\right)^{1/s} \tag{13}$$

此即 Minkowski 不等式, 所以

$$\begin{aligned}D(x,y) &\leqslant \left(\sum_{j=1}^{d}(|x_j-z_j|+|z_j-y_j|)^s\right)^{1/s}\\ &\leqslant \left(\sum_{j=1}^{d}|x_j-z_j|^s\right)^{1/s}+\left(\sum_{j=1}^{d}|z_j-y_j|^s\right)^{1/s}\\ &= D(x,z)+D(z,y)\end{aligned} \tag{14}$$

因此, 三角不等式成立. □

本题用到的 Hölder 不等式可由 Jensen 不等式证明如下 [3]. 简便起见, 对向量 $\boldsymbol{a}=(a_1,a_2,\ldots,a_n)\in\mathbb{R}^n$ 和 $p\geqslant 1$, 定义 ℓ_p 范数为

$$\|\boldsymbol{a}\|_p=\left(\sum_{k=1}^{n}|a_k|^p\right)^{1/p}, \tag{15}$$

则 Hölder 不等式可表述为对 $p\geqslant 1$ 和 $\forall\ \boldsymbol{a},\boldsymbol{b}\in\mathbb{R}_+^n$, 有

$$\sum_{k=1}^{n}a_kb_k\leqslant\|\boldsymbol{a}\|_p\,\|\boldsymbol{b}\|_q,\quad \frac{1}{p}+\frac{1}{q}=1. \tag{16}$$

证明: 对凸函数 $f(x)=x^p$ 应用 Jensen 不等式有

$$\left(\sum_{k=1}^{n}\lambda_k x_k\right)^p \leqslant \sum_{k=1}^{n}\lambda_k x_k^p, \quad \sum_{k=1}^{n}\lambda_k=1,\ \lambda_k\geqslant 0,\ k=1,2,\ldots,n \tag{17}$$

取 $a_k=\lambda_k^{1/p}x_k$, $c_k=\lambda_k^{1/q}$, 则上式变为

$$\sum_{k=1}^{n}a_k c_k \leqslant \left(\sum_{k=1}^{n}a_k^p\right)^{1/p}, \quad \sum_{k=1}^{n}c_k^q=1,\ c_k\geqslant 0,\ k=1,2,\ldots,n \tag{18}$$

取

$$c_k=\frac{b_k}{\|\boldsymbol{b}\|_q} \tag{19}$$

满足

$$\sum_{k=1}^{n}c_k^q=1,\ c_k\geqslant 0,\ k=1,2,\ldots,n \tag{20}$$

则有

$$\sum_{k=1}^{n}a_k\frac{b_k}{\|\boldsymbol{b}\|_q}\leqslant\left(\sum_{k=1}^{n}a_k^p\right)^{1/p}=\|\boldsymbol{a}\|_p \tag{21}$$

即

$$\sum_{k=1}^{n}a_k b_k\leqslant\|\boldsymbol{a}\|_p\ \|\boldsymbol{b}\|_q \tag{22}$$

因此, Hölder 不等式成立. □

k-NN Classifier

3. Let $\mathcal{D}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,\ldots,\boldsymbol{x}_n\}$ be a set of n independent labelled samples and let $\mathcal{D}_k(\boldsymbol{x})=\{\boldsymbol{x}'_1,\boldsymbol{x}'_2,\ldots,\boldsymbol{x}'_k\}$ be the k nearest neighbors of $\boldsymbol{x}$. Recall that the k-nearest-neighbor rule for classifying $\boldsymbol{x}$ is to give $\boldsymbol{x}$ the label most frequently represented in $\mathcal{D}_k(\boldsymbol{x})$. Consider a two-category problem with $P(\omega_1)=P(\omega_2)=1/2$. Assume further that the conditional densities $p(x|\omega_i)$ are uniform within unit hyperspheres, and the two categories center on two points ten units apart. Figure 3 shows a diagram of this situation.

3.1. Show that if k is odd, the average probability of error is given by

$$P_n(e)=\frac{1}{2^n}\sum_{j=0}^{(k-1)/2}\binom{n}{j}.$$

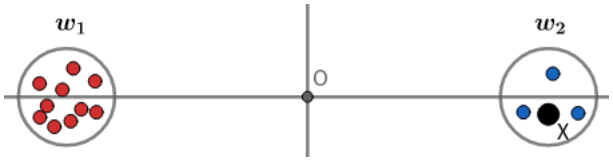


图 3: A diagram of assumed situation. When $k \geqslant 7$, X is misclassified as there are only 3 samples in ω_2.

解: 令 $P(x \in \omega_i, \omega_j)$, $i, j = 1, 2$ 表示 x 属于 ω_j 类, 但是 k-NN 算法将其判断为 ω_i 类. 当 ω_j 类的样本数量小于 $(k-1)/2$ 时, 就会出现分类错误的情况, 即此时 $i \neq j$.

因此, 错误率为

$$\begin{aligned} P_n(e) &= P(x \in \omega_1, \omega_2) + P(x \in \omega_2, \omega_1) \\ &= P(x \in \omega_1 | \omega_2) P(\omega_2) + P(x \in \omega_2 | \omega_1) P(\omega_1) \\ &= \frac{1}{2} \sum_{j=0}^{(k-1)/2} \binom{n}{j} \left(\frac{1}{2}\right)^j \left(\frac{1}{2}\right)^{n-j} + \frac{1}{2} \sum_{j=0}^{(k-1)/2} \binom{n}{j} \left(\frac{1}{2}\right)^j \left(\frac{1}{2}\right)^{n-j} \\ &= \frac{1}{2^n} \sum_{j=0}^{(k-1)/2} \binom{n}{j} \end{aligned} \tag{23}$$

3.2. Show that for this case the single-nearest neighbor rule has a lower error rate than the k-nearest-neighbor error rate for $k > 1$.

解: 当 $k = 1$ 时, 由上题可知最近邻法错误率为

$$\tilde{P}_n(e) = \frac{1}{2^n} \tag{24}$$

当 $k > 1$ 时

$$\begin{aligned} P_n(e) &= \frac{1}{2^n} \sum_{j=0}^{(k-1)/2} \binom{n}{j} \\ &= \frac{1}{2^n} \sum_{j=1}^{(k-1)/2} \binom{n}{j} + \frac{1}{2^n} \\ &> \frac{1}{2^n} = \tilde{P}_n(e) \end{aligned} \tag{25}$$

即最近邻法错误率 $\tilde{P}_n(e) < P_n(e)$.

3.3. If k is odd and is allowed to increase with n but is restricted by $k < a\sqrt{n}$, where a is a positive constant, show that $P_n(e) \to 0$ as $n \to \infty$.

解: 当 $n \to \infty$ 时, 有

$$\begin{aligned}
\lim_{n\to\infty} P_n(e) &= \lim_{n\to\infty} \frac{1}{2^n} \sum_{j=0}^{(k-1)/2} \binom{n}{j} \\
&\leqslant \lim_{n\to\infty} \frac{1}{2^n} \sum_{j=0}^{\lfloor (a\sqrt{n}-1)/2 \rfloor} \binom{n}{j} \\
&= \lim_{n\to\infty} \frac{1}{2^n} \sum_{j=0}^{\lfloor (a\sqrt{n}-1)/2 \rfloor} n^j, \quad \binom{n}{j} = \frac{n!}{j!(n-j)!} \sim n^j \\
&= \lim_{n\to\infty} \frac{n^{(a\sqrt{n}-1)/2}}{2^n} \\
&= \lim_{n\to\infty} \frac{n^{a\sqrt{n}/2}}{2^n} \\
&= \lim_{n\to\infty} \frac{\exp(\frac{a}{2}\sqrt{n}\log n)}{\exp(n\log 2)} \\
&= \lim_{n\to\infty} \exp\left(\frac{a}{2}\sqrt{n}\log n - n\log 2\right) \\
&= \lim_{n\to\infty} \exp\left[-n\log 2 \cdot \left(1 - \frac{a\log n}{2\sqrt{n}\log 2}\right)\right] \\
&= \lim_{n\to\infty} \exp\left[-n\log 2 \cdot \left(1 - \frac{an^{-1}}{n^{-1/2}\log 2}\right)\right] \\
&= \lim_{n\to\infty} \exp(-n\log 2) \\
&= 0
\end{aligned} \tag{26}$$

又 $P_n(e) \geqslant 0$, 因此

$$\lim_{n\to\infty} P_n(e) = 0 \tag{27}$$

Programming: k-NN Classifier on MNIST

4. Please implement k-NN classifier and run on MNIST [4]. You need to follow the official train/test split of MNIST. Compare the performance with the following settings:

- Using 100, 300, 1000, 3000, 10000 training samples.
- Using different values of k.
- Using at least three different distance metrics.

In this assignment, you are *NOT* allowed to use any existing libraries or code snippets that provides k-NN algorithm.

解: 首先分析训练样本数量对算法性能的影响, 分别使用 $n = 100,\ 300,\ 1000,\ 3000,\ 10000$ 个训练样本作为训练集, 比较最近邻分类器 ($k = 1$) 的性能变化, 计算采用 Minkowski 距离 $p = 2$, 即欧氏距离. k-NN 算法的准确率和运行时间分别如图 4 和图 5 所示.

由图可知, 算法的正确率随着样本数量增加而增加, 在样本数量增加到一定值后, 正确率的增长速度变慢; 算法的运行时间随着训练样本数量的增加而近乎线性增加.

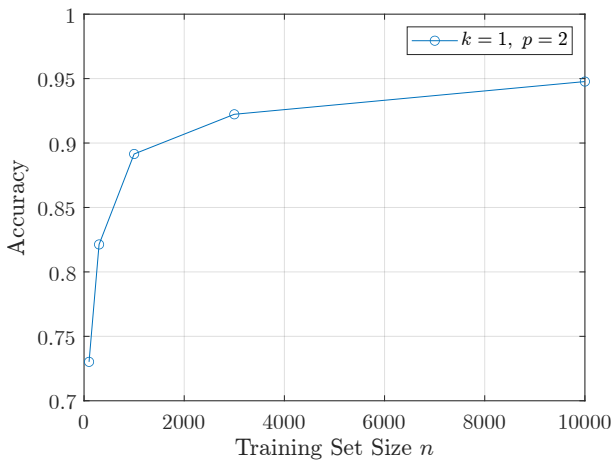


图 4: 正确率与训练样本数变化关系

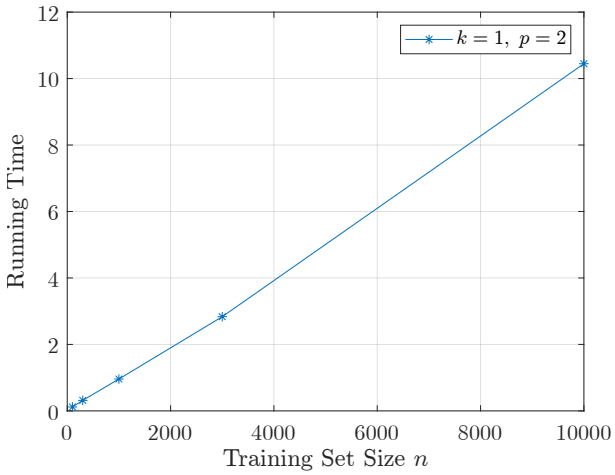


图 5: 运行时间与训练样本数变化关系

其次, 分析 k-NN 算法的参数 k 对算法性能的影响, 训练样本数取 $n = 3000$, Minkowski 距离参数取 $p = 2$, 即欧氏距离, 取 $k = [1, 100]$, 得到 k-NN 算法的准确率和运行时间分别如图 6 和图 7 所示.

由图可知, 算法的正确率随着 k 的增加先增加后减小, $k = 3$ 时正确率最高, 取过大的 k 反而会使得算法的正确率下降; 算法的运行时间基本稳定在 2.85 s, 与 k 的取值基本无关. 因为 k-NN 的时间复杂度主要体现在计算待预测样本点与训练集所有样本点之间的距离, 选择不同的 k 值对实际算法运行时间无明显影响.

最后分析不同的距离对算法性能的影响, 训练样本数取 $n = 3000$, 比较最近邻分类器 ($k = 1$) 的性能变化, Minkowski 距离参数取 $p = 1,\ 2,\ 4,\ \infty$, 得到 k-NN 算法的准确率和运行时间分别如图 8 和图 9 所示.

由图可知, 算法的正确率随着 p 的增大而有所增加，但 Chebyshev 距离 ($p = \infty$) 的正确率最低; 算法的运行时间与 ℓ_p 范数的计算复杂度有关, ℓ_1, ℓ_2 与 ℓ_∞ 范数的复杂度基本相同, 而 ℓ_4 范数的复杂度远远高于其他三种范数的复杂度.

Literature Reading

Please read a paper about metric learning [5].

Hint: You do not have to submit anything for this reading section.

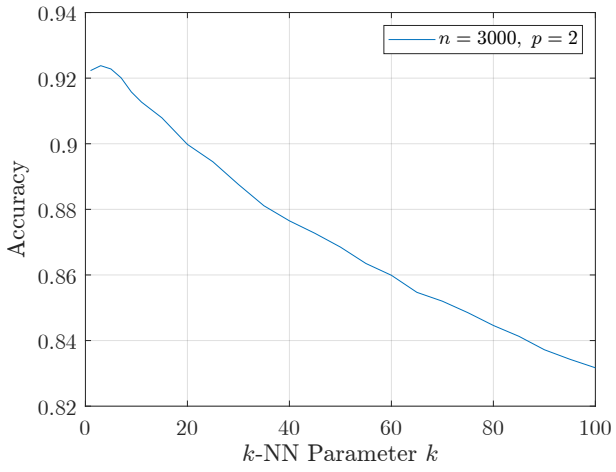


图 6: 正确率与 k-NN 参数 k 变化关系

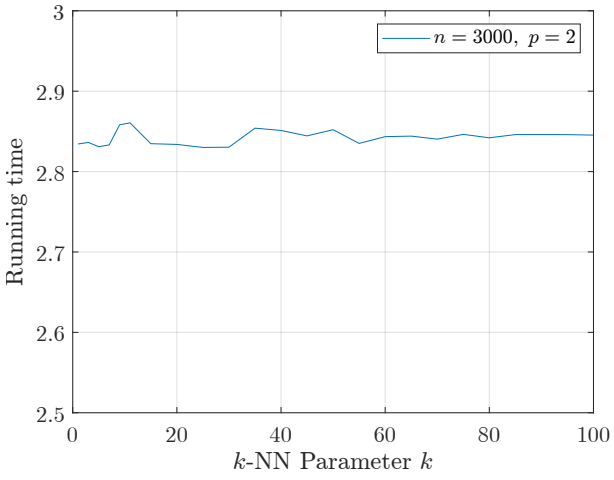


图 7: 运行时间与 k-NN 参数 k 变化关系

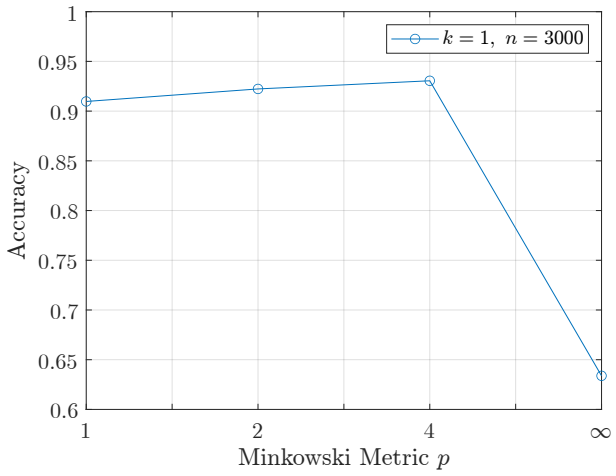


图 8: 正确率与 Minkowski 距离参数 p 变化关系

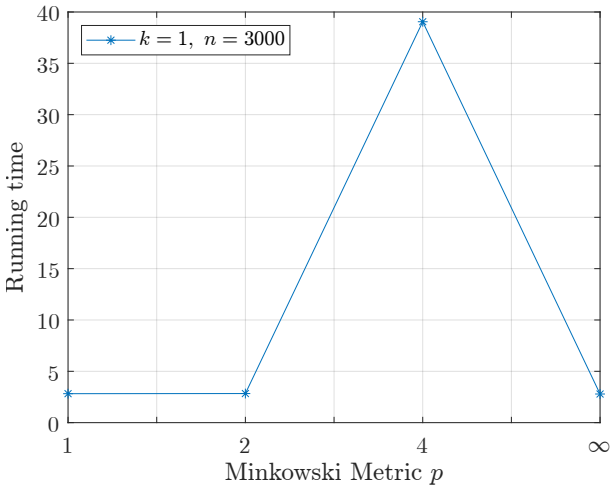


图 9: 运行时间与 Minkowski 距离参数 p 变化关系

参考文献

[1] File: Euclidean Voronoi diagram.svg. (2020, October 10). Wikimedia Commons, the free media repository. Retrieved 09:39, March 29, 2021 from https://commons.wikimedia.org/wiki/File:Euclidean_Voronoi_diagram.svg

[2] File: Manhattan Voronoi Diagram.svg. (2020, September 17). Wikimedia Commons, the free media repository. Retrieved 08:26, June 19, 2021 from https://commons.wikimedia.org/wiki/File:Manhattan_Voronoi_Diagram.svg

[3] J.M. Steele, The Cauchy-Schwarz Master Class, An Introduction to the Art of Mathematical Inequalities, MAA Problem Book Series, Cambridge University Press, 2008.

[4] LeCun Y. The MNIST database of handwritten digits[J]. http://yann.lecun.com/exdb/mnist/, 1998.

[5] Xing E P, Ng A Y, Jordan M I, et al. Distance metric learning with application to clustering with side-information[C] // NIPS. 2002, 15(505-512): 12.

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 7**

Feature Extraction and Selection

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Fisher Criterion

1. It's interesting to see that under some circumstances, the Fisher criterion can be obtained as a special case of the least squares. Consider the binary classification problem, let's unify the expression at the very beginning for convenience of the following steps, and you are required to obey the notations given below.

Suppose we have N_1 points of class $\mathcal{C}_1$ and N_2 of class $\mathcal{C}_2$, then the mean vectors of the two classes are given by

$$\boldsymbol{m}_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}\boldsymbol{x}_n, \quad \boldsymbol{m}_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}\boldsymbol{x}_n. \tag{1}$$

In the lecture notes, we have defined *between-class* covariance matrix and *within-class* covariance matrix

$$S_B = (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^\top, \tag{2}$$

$$S_W = \sum_{n\in\mathcal{C}_1}(\boldsymbol{x}_n - \boldsymbol{m}_1)(\boldsymbol{x}_n - \boldsymbol{m}_1)^\top + \sum_{n\in\mathcal{C}_2}(\boldsymbol{x}_n - \boldsymbol{m}_2)(\boldsymbol{x}_n - \boldsymbol{m}_2)^\top. \tag{3}$$

Now, let's turn to the least square problem. We take the targets for $\mathcal{C}_1$ to be N/N_1 and $\mathcal{C}_2$ to be $-N/N_2$ where $N = N_1 + N_2$ (This may be a little confusing, but you will see the reasons of doing so in a short time). Then the sum-of-square error function can be written as

$$E = \frac{1}{2}\sum_{n=1}^{N}(\boldsymbol{w}^\top\boldsymbol{x}_n + w_0 - t_n)^2, \tag{4}$$

where, $(\boldsymbol{x}_n, t_n)$ are the points we have. Target t_n equals to N/N_1 or $-N/N_2$ according to its class. Our goal is to estimate $\boldsymbol{w}$ and w_0.

1.1. Show that the optimal w_0 is $w_0 = -\boldsymbol{w}^\top\boldsymbol{m}$, where

$$\boldsymbol{m} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n. \tag{5}$$

解: 最小二乘表达式对 w_0 求导得

$$\frac{\partial E}{\partial w_0} = \sum_{n=1}^{N}(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0 - t_n) \tag{6}$$

令此偏导数为 0, 有

$$\begin{aligned} w_0 &= -\frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{w}^\top \boldsymbol{x}_n - t_n) \\ &= -\frac{1}{N}\sum_{n=1}^{N}\boldsymbol{w}^\top \boldsymbol{x}_n + \frac{1}{N}\sum_{n\in\mathcal{C}_1} t_n + \frac{1}{N}\sum_{n\in\mathcal{C}_2} t_n \\ &= -\boldsymbol{w}^\top \boldsymbol{m} + \frac{1}{N}\sum_{n\in\mathcal{C}_1}\frac{N}{N_1} - \frac{1}{N}\sum_{n\in\mathcal{C}_2}\frac{N}{N_2} \\ &= -\boldsymbol{w}^\top \boldsymbol{m} + 1 - 1 \\ &= -\boldsymbol{w}^\top \boldsymbol{m} \end{aligned} \tag{7}$$

1.2. Derive the equation that the optimal $\boldsymbol{w}$ should obey

$$\left(S_W + \frac{N_1 N_2}{N} S_B\right)\boldsymbol{w} = N(\boldsymbol{m}_1 - \boldsymbol{m}_2). \tag{8}$$

解: 由 $w_0 = -\boldsymbol{w}^\top \boldsymbol{m}$ 以及 $N\boldsymbol{m} = N_1\boldsymbol{m}_1 + N_2\boldsymbol{m}_2$ 可知最小二乘表达式对 $\boldsymbol{w}$ 求导为

$$\begin{aligned} \frac{\partial E}{\partial \boldsymbol{w}} &= \sum_{n=1}^{N}\boldsymbol{x}_n(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0 - t_n) \\ &= \sum_{n=1}^{N}\boldsymbol{x}_n(\boldsymbol{w}^\top \boldsymbol{x}_n - \boldsymbol{w}^\top \boldsymbol{m}) - \sum_{n\in\mathcal{C}_1} t_n\boldsymbol{x}_n - \sum_{n\in\mathcal{C}_2} t_n\boldsymbol{x}_n \\ &= \sum_{n=1}^{N}\boldsymbol{x}_n(\boldsymbol{x}_n^\top \boldsymbol{w} - \boldsymbol{m}^\top \boldsymbol{w}) - \sum_{n\in\mathcal{C}_1}\frac{N}{N_1}\boldsymbol{x}_n + \sum_{n\in\mathcal{C}_2}\frac{N}{N_2}\boldsymbol{x}_n \\ &= \sum_{n=1}^{N}(\boldsymbol{x}_n\boldsymbol{x}_n^\top - \boldsymbol{x}_n\boldsymbol{m}^\top)\boldsymbol{w} - N(\boldsymbol{m}_1 - \boldsymbol{m}_2) \\ &= \left(\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top - \sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{m}^\top\right)\boldsymbol{w} - N(\boldsymbol{m}_1 - \boldsymbol{m}_2) \\ &= \left(\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top - N\boldsymbol{m}\boldsymbol{m}^\top\right)\boldsymbol{w} - N(\boldsymbol{m}_1 - \boldsymbol{m}_2) \\ &= \left(\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top - \frac{N\boldsymbol{m}(N\boldsymbol{m})^\top}{N}\right)\boldsymbol{w} - N(\boldsymbol{m}_1 - \boldsymbol{m}_2) \\ &= \left(\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top - \frac{(N_1\boldsymbol{m}_1 + N_2\boldsymbol{m}_2)(N_1\boldsymbol{m}_1 + N_2\boldsymbol{m}_2)^\top}{N}\right)\boldsymbol{w} - N(\boldsymbol{m}_1 - \boldsymbol{m}_2) \end{aligned} \tag{9}$$

又由 S_W 定义可知

$$\begin{aligned}
S_W &= \sum_{n\in\mathcal{C}_1}(\boldsymbol{x}_n-\boldsymbol{m}_1)(\boldsymbol{x}_n-\boldsymbol{m}_1)^\top + \sum_{n\in\mathcal{C}_2}(\boldsymbol{x}_n-\boldsymbol{m}_2)(\boldsymbol{x}_n-\boldsymbol{m}_2)^\top \\
&= \sum_{n\in\mathcal{C}_1}(\boldsymbol{x}_n\boldsymbol{x}_n^\top-\boldsymbol{x}_n\boldsymbol{m}_1^\top-\boldsymbol{m}_1\boldsymbol{x}_n^\top+\boldsymbol{m}_1\boldsymbol{m}_1^\top) + \sum_{n\in\mathcal{C}_2}(\boldsymbol{x}_n\boldsymbol{x}_n^\top-\boldsymbol{x}_n\boldsymbol{m}_2^\top-\boldsymbol{m}_2\boldsymbol{x}_n^\top+\boldsymbol{m}_2\boldsymbol{m}_2^\top) \\
&= \sum_{n\in\mathcal{C}_1}\boldsymbol{x}_n\boldsymbol{x}_n^\top - \sum_{n\in\mathcal{C}_1}\boldsymbol{x}_n\boldsymbol{m}_1^\top - \sum_{n\in\mathcal{C}_1}\boldsymbol{m}_1\boldsymbol{x}_n^\top + N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top + \sum_{n\in\mathcal{C}_2}\boldsymbol{x}_n\boldsymbol{x}_n^\top - \sum_{n\in\mathcal{C}_2}\boldsymbol{x}_n\boldsymbol{m}_2^\top - \sum_{n\in\mathcal{C}_2}\boldsymbol{m}_2\boldsymbol{x}_n^\top + N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top \\
&= \sum_{n\in\mathcal{C}_1}\boldsymbol{x}_n\boldsymbol{x}_n^\top - N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top - N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top + N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top + \sum_{n\in\mathcal{C}_2}\boldsymbol{x}_n\boldsymbol{x}_n^\top - N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top - N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top + N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top \\
&= \sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top - N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top - N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top
\end{aligned} \tag{10}$$

即

$$\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top = S_W + N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top + N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top \tag{11}$$

代入偏导数表达式可得

$$\begin{aligned}
\frac{\partial E}{\partial \boldsymbol{w}} &= \left(\sum_{n=1}^{N}\boldsymbol{x}_n\boldsymbol{x}_n^\top - \frac{(N_1\boldsymbol{m}_1+N_2\boldsymbol{m}_2)(N_1\boldsymbol{m}_1+N_2\boldsymbol{m}_2)^\top}{N}\right)\boldsymbol{w} - N(\boldsymbol{m}_1-\boldsymbol{m}_2) \\
&= \left(S_W + N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top + N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top - \frac{(N_1\boldsymbol{m}_1+N_2\boldsymbol{m}_2)(N_1\boldsymbol{m}_1+N_2\boldsymbol{m}_2)^\top}{N}\right)\boldsymbol{w} - N(\boldsymbol{m}_1-\boldsymbol{m}_2) \\
&= \left(S_W + \frac{NN_1\boldsymbol{m}_1\boldsymbol{m}_1^\top + NN_2\boldsymbol{m}_2\boldsymbol{m}_2^\top - (N_1\boldsymbol{m}_1+N_2\boldsymbol{m}_2)(N_1\boldsymbol{m}_1+N_2\boldsymbol{m}_2)^\top}{N}\right)\boldsymbol{w} - N(\boldsymbol{m}_1-\boldsymbol{m}_2) \\
&= \left(S_W + \frac{N_2N_1\boldsymbol{m}_1\boldsymbol{m}_1^\top + N_1N_2\boldsymbol{m}_2\boldsymbol{m}_2^\top - N_1N_2\boldsymbol{m}_1\boldsymbol{m}_2^\top - N_2N_1\boldsymbol{m}_2\boldsymbol{m}_1^\top}{N}\right)\boldsymbol{w} - N(\boldsymbol{m}_1-\boldsymbol{m}_2) \\
&= \left(S_W + \frac{N_1N_2}{N}(\boldsymbol{m}_1\boldsymbol{m}_1^\top - \boldsymbol{m}_1\boldsymbol{m}_2^\top - \boldsymbol{m}_2\boldsymbol{m}_1^\top + \boldsymbol{m}_2\boldsymbol{m}_2^\top)\right)\boldsymbol{w} - N(\boldsymbol{m}_1-\boldsymbol{m}_2) \\
&= \left(S_W + \frac{N_1N_2}{N}(\boldsymbol{m}_1-\boldsymbol{m}_2)(\boldsymbol{m}_1-\boldsymbol{m}_2)^\top\right)\boldsymbol{w} - N(\boldsymbol{m}_1-\boldsymbol{m}_2) \\
&= \left(S_W + \frac{N_1N_2}{N}S_B\right)\boldsymbol{w} - N(\boldsymbol{m}_1-\boldsymbol{m}_2)
\end{aligned} \tag{12}$$

令此偏导数为 0, 则有

$$\left(S_W + \frac{N_1N_2}{N}S_B\right)\boldsymbol{w} = N(\boldsymbol{m}_1-\boldsymbol{m}_2) \tag{13}$$

1.3. Show that $\boldsymbol{w}$ satisfies: $\boldsymbol{w}\propto S_W^{-1}(\boldsymbol{m}_2-\boldsymbol{m}_1)$ from equation (8), which means we've got the same form as that of Fisher criterion.

解: 记

$$R \triangleq (\boldsymbol{m}_2-\boldsymbol{m}_1)^\top\boldsymbol{w} \tag{14}$$

由式 (8) 可知

$$S_W \boldsymbol{w} + \frac{N_1 N_2}{N} S_B \boldsymbol{w} = -N(\boldsymbol{m}_2 - \boldsymbol{m}_1) \tag{15}$$

移项并代入 S_B 定义可得

$$\begin{aligned} S_W \boldsymbol{w} &= -N(\boldsymbol{m}_2 - \boldsymbol{m}_1) - \frac{N_1 N_2}{N} S_B \boldsymbol{w} \\ &= -N(\boldsymbol{m}_2 - \boldsymbol{m}_1) - \frac{N_1 N_2}{N} (\boldsymbol{m}_2 - \boldsymbol{m}_1)(\boldsymbol{m}_2 - \boldsymbol{m}_1)^\top \boldsymbol{w} \\ &= -N(\boldsymbol{m}_2 - \boldsymbol{m}_1) - \frac{N_1 N_2}{N} (\boldsymbol{m}_2 - \boldsymbol{m}_1) R \\ &= \left(-N - \frac{R N_1 N_2}{N} \right) (\boldsymbol{m}_2 - \boldsymbol{m}_1) \end{aligned} \tag{16}$$

假设矩阵 S_W 可逆, 则有

$$\begin{aligned} \boldsymbol{w} &= \left(-N - \frac{R N_1 N_2}{N} \right) S_W^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1) \\ &\propto S_W^{-1} (\boldsymbol{m}_2 - \boldsymbol{m}_1) \end{aligned} \tag{17}$$

Linear Discriminant Analysis (LDA)

2. Consider the generalization of Fisher discriminant to $K > 2$ classes, and assume that the dimensionality of the input space is greater than the number K of classes. Next, we introduce linear features $y_k = \boldsymbol{w}_k^\top \boldsymbol{x}$. The weight vectors $\{\boldsymbol{w}_k\}$ can be considered to be the columns of a matrix $\boldsymbol{W}$, so that:

$$\boldsymbol{y} = \boldsymbol{W}^\top \boldsymbol{x}, \tag{18}$$

where, $\boldsymbol{x} \in \mathbb{R}^D$ and $\boldsymbol{y} \in \mathbb{R}^{D'}$. By this means, we have projected the D-dimensional $\boldsymbol{x}$-space onto the D'-dimensional $\boldsymbol{y}$-space, in which we can better separate the data.

The generalization of the *within-class* covariance matrix to the case of K classes is:

$$S_W = \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^\top, \tag{19}$$

where

$$\boldsymbol{m}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \boldsymbol{x}_n. \tag{20}$$

The total covariance matrix is:

$$S_T = \sum_{n=1}^{N} (\boldsymbol{x}_n - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m})^\top, \tag{21}$$

where

$$\boldsymbol{m} = \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{m}_k. \tag{22}$$

2.1. Decompose the total covariance matrix S_T into *within-class* covariance matrix S_W and *between-class* covariance matrix S_B, and show that S_B has the form:

$$S_B = \sum_{k=1}^{K} N_k(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^\top. \tag{23}$$

解: 由矩阵 S_T 定义可得

$$\begin{aligned}
S_T &= \sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m})^\top \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\boldsymbol{x}_n - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m})^\top \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\boldsymbol{x}_n - \boldsymbol{m}_k + \boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m}_k + \boldsymbol{m}_k - \boldsymbol{m})^\top \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}\Big[(\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^\top + (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{m}_k - \boldsymbol{m})^\top + (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m}_k)^\top \\
&\qquad\qquad + (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^\top\Big] \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^\top + \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}\Big[(\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{m}_k - \boldsymbol{m})^\top + (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m}_k)^\top\Big] \\
&\quad + \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^\top \\
&= S_W + \sum_{k=1}^{K}\left\{\sum_{n\in\mathcal{C}_k}\Big[(\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{m}_k - \boldsymbol{m})^\top + (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{x}_n - \boldsymbol{m}_k)^\top\Big]\right\} + S_B \\
&= S_W + \sum_{k=1}^{K}\Big[(N_k\boldsymbol{m}_k - N_k\boldsymbol{m}_k)(\boldsymbol{m}_k - \boldsymbol{m})^\top + (\boldsymbol{m}_k - \boldsymbol{m})(N_k\boldsymbol{m}_k - N_k\boldsymbol{m}_k)^\top\Big] + S_B \\
&= S_W + \sum_{k=1}^{K}\Big[\mathbf{0}(\boldsymbol{m}_k - \boldsymbol{m})^\top + (\boldsymbol{m}_k - \boldsymbol{m})\mathbf{0}^\top\Big] + S_B \\
&= S_W + S_B
\end{aligned} \tag{24}$$

其中类间协方差矩阵为

$$\begin{aligned}
S_B &= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^\top \\
&= \sum_{k=1}^{K} N_k(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^\top
\end{aligned} \tag{25}$$

2.2. Write down the *within-class* covariance matrix s_W and *between-class* covariance matrix s_B of the projected D'-dimensional $\boldsymbol{y}$-space.

解: 在 $\boldsymbol{y}$ 空间中, $\boldsymbol{y}_n = \boldsymbol{W}^\top \boldsymbol{x}_n$, $\forall\ n = 1, 2, \ldots, N$, 且

$$\tilde{\boldsymbol{m}} = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{y}_n = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{W}^\top \boldsymbol{x}_n = \boldsymbol{W}^\top \boldsymbol{m} \tag{26}$$

$$\tilde{\boldsymbol{m}}_k = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k} \boldsymbol{y}_n = \frac{1}{N_k}\sum_{n\in\mathcal{C}_k} \boldsymbol{W}^\top \boldsymbol{x}_n = \boldsymbol{W}^\top \boldsymbol{m}_k, \quad \forall\ k = 1, 2, \ldots, K \tag{27}$$

故类内协方差矩阵为

$$\begin{aligned} s_W &= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k} (\boldsymbol{y}_n - \tilde{\boldsymbol{m}}_k)(\boldsymbol{y}_n - \tilde{\boldsymbol{m}}_k)^\top \\ &= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k} (\boldsymbol{W}^\top \boldsymbol{x}_n - \boldsymbol{W}^\top \boldsymbol{m}_k)(\boldsymbol{W}^\top \boldsymbol{x}_n - \boldsymbol{W}^\top \boldsymbol{m}_k)^\top \\ &= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k} \boldsymbol{W}^\top (\boldsymbol{x}_n - \boldsymbol{m}_k)(\boldsymbol{x}_n - \boldsymbol{m}_k)^\top \boldsymbol{W} \\ &= \boldsymbol{W}^\top S_W \boldsymbol{W} \end{aligned} \tag{28}$$

类间协方差矩阵为

$$\begin{aligned} s_B &= \sum_{k=1}^{K} N_k (\tilde{\boldsymbol{m}}_k - \tilde{\boldsymbol{m}})(\tilde{\boldsymbol{m}}_k - \tilde{\boldsymbol{m}})^\top \\ &= \sum_{k=1}^{K} N_k (\boldsymbol{W}^\top \boldsymbol{m}_k - \boldsymbol{W}^\top \boldsymbol{m})(\boldsymbol{W}^\top \boldsymbol{m}_k - \boldsymbol{W}^\top \boldsymbol{m})^\top \\ &= \sum_{k=1}^{K} N_k \boldsymbol{W}^\top (\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^\top \boldsymbol{W} \\ &= \boldsymbol{W}^\top S_B \boldsymbol{W} \end{aligned} \tag{29}$$

2.3. Many possible choices of criterion can be implemented to decide the weight matrix $\boldsymbol{W}$, more than 5 examples are shown in lecture slide. Here, we are using another criterion

$$J(\boldsymbol{W}) = \frac{\prod_{\text{diag}} s_B}{\prod_{\text{diag}} s_W}, \tag{30}$$

where, $\prod_{\text{diag}} \boldsymbol{A}$ means multiplication of the diagonal elements of matrix $\boldsymbol{A}$. Represent $J(\boldsymbol{W})$ explicitly with $\boldsymbol{W}$, S_W and S_B.

解: 由上题可得

$$J(\boldsymbol{W}) = \frac{\prod_{\text{diag}} s_B}{\prod_{\text{diag}} s_W} = \frac{\prod_{\text{diag}} \boldsymbol{W}^\top S_B \boldsymbol{W}}{\prod_{\text{diag}} \boldsymbol{W}^\top S_W \boldsymbol{W}} = \frac{\prod_{k=1}^{D'} \boldsymbol{w}_k^\top S_B \boldsymbol{w}_k}{\prod_{k=1}^{D'} \boldsymbol{w}_k^\top S_W \boldsymbol{w}_k} = \prod_{k=1}^{D'} \frac{\boldsymbol{w}_k^\top S_B \boldsymbol{w}_k}{\boldsymbol{w}_k^\top S_W \boldsymbol{w}_k} \tag{31}$$

2.4. As is stated above, we now want to project the original data space onto a space with D' dimensions, while at the same time trying to maximize $J(\boldsymbol{W})$ represented by equation (30). Write down the equations that columns of weight matrix $\boldsymbol{W}$ should obey (which means the selected projection directions).

解: 记

$$J_k(\boldsymbol{w}_k) \triangleq \frac{\boldsymbol{w}_k^\top S_B \boldsymbol{w}_k}{\boldsymbol{w}_k^\top S_W \boldsymbol{w}_k}, \quad \forall\ k = 1, 2, \ldots, D' \tag{32}$$

由上题可得

$$J(\boldsymbol{W}) = \prod_{k=1}^{D'} \frac{\boldsymbol{w}_k^\top S_B \boldsymbol{w}_k}{\boldsymbol{w}_k^\top S_W \boldsymbol{w}_k} = \prod_{k=1}^{D'} J_k(\boldsymbol{w}_k) \tag{33}$$

则

$$\max_{\boldsymbol{W}} J(\boldsymbol{W}) \Leftrightarrow \max_{\boldsymbol{w}_k} J_k(\boldsymbol{w}_k) = \frac{\boldsymbol{w}_k^\top S_B \boldsymbol{w}_k}{\boldsymbol{w}_k^\top S_W \boldsymbol{w}_k}, \quad k = 1, 2, \ldots, D' \tag{34}$$

由于 $J_k(\boldsymbol{w}_k)$ 为广义 Rayleigh 商, 则其优化问题可以转换为

$$\begin{aligned} &\max_{\boldsymbol{w}_k} \ \boldsymbol{w}_k^\top S_B \boldsymbol{w}_k \\ &\text{s.t.} \quad \boldsymbol{w}_k^\top S_W \boldsymbol{w}_k = c \neq 0 \end{aligned} \tag{35}$$

引入 Lagrange 乘子 λ_k 则有 Lagrange 函数

$$L(\boldsymbol{w}_k, \lambda_k) = \boldsymbol{w}_k^\top S_B \boldsymbol{w}_k - \lambda_k (\boldsymbol{w}_k^\top S_W \boldsymbol{w}_k - c) \tag{36}$$

其对 $\boldsymbol{w}_k$ 的偏导数为

$$\frac{\partial L(\boldsymbol{w}_k, \lambda_k)}{\partial \boldsymbol{w}_k} = 2 S_B \boldsymbol{w}_k - 2 \lambda_k S_W \boldsymbol{w}_k \tag{37}$$

令此偏导数为 0, 可知 $\boldsymbol{w}_k$ 需满足

$$S_B \boldsymbol{w}_k - \lambda_k S_W \boldsymbol{w}_k = 0 \tag{38}$$

若矩阵 S_W 可逆, 则有

$$S_W^{-1} S_B \boldsymbol{w}_k = \lambda_k \boldsymbol{w}_k \tag{39}$$

即此时 $\boldsymbol{w}_k$ 是矩阵 $S_W^{-1} S_B$ 的特征向量.

2.5. As is stated in the problem, we have K classes in all, and we are trying to find linear features (or projection directions) by maximizing $J(\boldsymbol{W})$. How many such features at most are we able to find? Explain your reason.

解: 最多可取 $K-1$ 个特征, 证明如下. 由上题可知 $\boldsymbol{w}_k$ 是矩阵 $S_W^{-1} S_B$ 的特征向量, 当 $\boldsymbol{w}_k$ 取线性无关的特征向量时, 最多取得的特征向量数为 $\operatorname{rank}(S_W^{-1} S_B)$. 易知 $\operatorname{rank}(S_W^{-1} S_B) \leqslant \operatorname{rank}(S_B)$, 则问题转化为求矩阵 S_B 的秩的最大值.

由矩阵 S_B 定义

$$S_B = \sum_{k=1}^{K} N_k(\boldsymbol{m}_k - \boldsymbol{m})(\boldsymbol{m}_k - \boldsymbol{m})^\top \tag{40}$$

可知

$$\operatorname{rank}(S_B) = \operatorname{rank}(\boldsymbol{m}_1 - \boldsymbol{m}, \boldsymbol{m}_2 - \boldsymbol{m}, \ldots, \boldsymbol{m}_K - \boldsymbol{m}) \tag{41}$$

又

$$\begin{aligned}
\sum_{k=1}^{K} N_k(\boldsymbol{m}_k - \boldsymbol{m}) &= \sum_{k=1}^{K} N_k \left(\frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \boldsymbol{x}_n - \frac{1}{N} \sum_{n=1}^{N} \boldsymbol{x}_n \right) \\
&= \sum_{k=1}^{K} \sum_{n \in \mathcal{C}_k} \boldsymbol{x}_n - \sum_{k=1}^{K} \sum_{n=1}^{N} \frac{N_k}{N} \boldsymbol{x}_n \\
&= \sum_{n=1}^{N} \boldsymbol{x}_n - \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{N_k}{N} \boldsymbol{x}_n \\
&= \sum_{n=1}^{N} \boldsymbol{x}_n - \sum_{n=1}^{N} \boldsymbol{x}_n \\
&= \mathbf{0}
\end{aligned} \tag{42}$$

则 $\{\boldsymbol{m}_1 - \boldsymbol{m}, \boldsymbol{m}_2 - \boldsymbol{m}, \ldots, \boldsymbol{m}_K - \boldsymbol{m}\}$ 线性相关, 所以

$$\operatorname{rank}(\boldsymbol{m}_1 - \boldsymbol{m}, \boldsymbol{m}_2 - \boldsymbol{m}, \ldots, \boldsymbol{m}_K - \boldsymbol{m}) \leqslant K - 1 \tag{43}$$

因此 $\operatorname{rank}(S_W^{-1} S_B) \leqslant \operatorname{rank}(S_B) \leqslant K - 1$, 即最多可选取 $K - 1$ 个特征.

Feature selection and error rate

An intuitive understanding between features and error rate.

3. Let's review the definition of binary-class Bayesian error rate at first. In classification problems, our goal is always to make as few misclassifications as possible. We need a rule that assigns each $\boldsymbol{x}$ to one of the available classes. Such a rule will divide the input space into regions $\mathcal{R}_k$ called *decision regions*, one for each class, such that all points in $\mathcal{R}_k$ are assigned to class $\mathcal{C}_k$. Take binary classification as an example: A mistake occurs when an input vector belonging to class $\mathcal{C}_1$ is assigned to class $\mathcal{C}_2$ or vice versa. The error rate is then given by

$$\begin{aligned}
p(\text{mistake}) &= p(\boldsymbol{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\boldsymbol{x} \in \mathcal{R}_2, \mathcal{C}_1) \\
&= \int_{\mathcal{R}_1} p(\boldsymbol{x}, \mathcal{C}_2) \mathrm{d}\boldsymbol{x} + \int_{\mathcal{R}_2} p(\boldsymbol{x}, \mathcal{C}_1) \mathrm{d}\boldsymbol{x}.
\end{aligned} \tag{44}$$

To minimize the above error rate, we have to make use of posterior distribution: If $p(\mathcal{C}_1|\boldsymbol{x}) > p(\mathcal{C}_2|\boldsymbol{x})$, then we assign that $\boldsymbol{x}$ to class $\mathcal{C}_1$, and vice versa. Thus leading to the Bayesian error rate.

3.1. Suppose we consider a K-class problem, derive the corresponding error rate as that of equation (44).

解: 令 $\mathcal{R}$ 表示输入空间 (input space), 即

$$\mathcal{R} \triangleq \bigcup_{k=1}^{K} \mathcal{R}_k \tag{45}$$

则错误率为

$$P(\text{mistake}) = \sum_{k=1}^{K} p(\boldsymbol{x} \in \mathcal{R} \setminus \mathcal{R}_k, \mathcal{C}_k) = \sum_{k=1}^{K} \int_{\mathcal{R} \setminus \mathcal{R}_k} p(\boldsymbol{x}, \mathcal{C}_k) \mathrm{d}\boldsymbol{x} \tag{46}$$

3.2. Let x_i, $i = 1, 2, 3$ be independent binary-valued features, and $P(x_i = 1|w_1) = \alpha_i$, $P(x_i = 1|w_2) = \beta_i$, $P(w_1) = P(w_2)$. Assume that $\beta_1 - \alpha_1 > \beta_2 - \alpha_2 > \beta_3 - \alpha_3$ and $\alpha_i < \beta_i$, $\forall\, i = 1, 2, 3$. Prove that the Bayesian error rate with only one feature will satisfy $e(x_1) < e(x_2) < e(x_3)$. Give your explanation of this phenomenon based on the three features.

解: 由题意可得 $\forall\, i = 1, 2, 3$, 有

$$\begin{aligned} P(x_i = 1|w_1)P(w_1) = \alpha_i P(w_1) < P(x_i = 1|w_2)P(w_2) = \beta_i P(w_2) \\ P(x_i = 0|w_1)P(w_1) = (1 - \alpha_i)P(w_1) > P(x_i = 0|w_2)P(w_2) = (1 - \beta_i)P(w_2) \end{aligned} \tag{47}$$

则 $\forall\, i = 1, 2, 3$, Bayes 决策规则为

$$\begin{aligned} x_i = 1 \to x_i \in w_2 \\ x_i = 0 \to x_i \in w_1 \end{aligned} \tag{48}$$

所以错误率为

$$\begin{aligned} e(x_i) &= P(x_i = 1, w_1) + P(x_i = 0, w_2) \\ &= P(x_i = 1|w_1)P(w_1) + P(x_i = 0|w_2)P(w_2) \\ &= \frac{1}{2}\alpha_i + \frac{1}{2}(1 - \beta_i) \\ &= \frac{1}{2} - \frac{1}{2}(\beta_i - \alpha_i), \quad \forall\, i = 1, 2, 3 \end{aligned} \tag{49}$$

由 $\beta_1 - \alpha_1 > \beta_2 - \alpha_2 > \beta_3 - \alpha_3$ 可知

$$e(x_1) < e(x_2) < e(x_3) \tag{50}$$

对每个特征, w_1 类的均值为

$$\mu_{i1} = 0 \cdot P(x_i = 0|w_1) + 1 \cdot P(x_i = 1|w_1) = \alpha_i, \quad \forall\, i = 1, 2, 3 \tag{51}$$

w_2 类的均值为

$$\mu_{i2} = 0 \cdot P(x_i = 0|w_2) + 1 \cdot P(x_i = 1|w_2) = \beta_i, \quad \forall\, i = 1, 2, 3 \tag{52}$$

两类的均值差为

$$d_i \triangleq \mu_{i2} - \mu_{i1} = \beta_i - \alpha_i, \quad \forall\, i = 1, 2, 3 \tag{53}$$

则由 $\beta_1 - \alpha_1 > \beta_2 - \alpha_2 > \beta_3 - \alpha_3$ 可知两类均值差

$$d_1 > d_2 > d_3 \tag{54}$$

即特征 x_1 的两类均值差最大, 特征 x_3 的两类均值差最小, 而均值差越大则在该特征上两类分离得越开, 从而使用该特征进行分类的错误率越小, 因此有 $e(x_1) < e(x_2) < e(x_3)$.

3.3. With the following parameters:

$$\alpha_1 = 0.1, \quad \alpha_2 = 0.05, \quad \alpha_3 = 0.01, \quad \beta_1 = 0.9, \quad \beta_2 = 0.8, \quad \beta_3 = 0.7, \tag{55}$$

calculate $e(x_1)$, $e(x_2)$, $e(x_3)$; $e(x_1, x_2)$, $e(x_1, x_3)$, $e(x_2, x_3)$. Compare the values of different error rate and present your explanation from the view of feature selection.

解: 由上题可得

$$e(x_1) = \frac{1}{2} - \frac{1}{2}(\beta_1 - \alpha_1) = \frac{1}{2} - \frac{1}{2} \times (0.9 - 0.1) = 0.1 \tag{56}$$

$$e(x_2) = \frac{1}{2} - \frac{1}{2}(\beta_2 - \alpha_2) = \frac{1}{2} - \frac{1}{2} \times (0.8 - 0.05) = 0.125 \tag{57}$$

$$e(x_3) = \frac{1}{2} - \frac{1}{2}(\beta_3 - \alpha_3) = \frac{1}{2} - \frac{1}{2} \times (0.7 - 0.01) = 0.155 \tag{58}$$

若使用两个特征 (x_1, x_2) 进行分类, 注意到特征相互独立可得

$$\begin{aligned}
P(x_1 = 1, x_2 = 1|w_1)P(w_1) = \alpha_1\alpha_2 P(w_1) &< P(x_1 = 1, x_2 = 1|w_2)P(w_2) = \beta_1\beta_2 P(w_2) \\
P(x_1 = 1, x_2 = 0|w_1)P(w_1) = \alpha_1(1 - \alpha_2)P(w_1) &< P(x_1 = 1, x_2 = 0|w_2)P(w_2) = \beta_1(1 - \beta_2)P(w_2) \\
P(x_1 = 0, x_2 = 1|w_1)P(w_1) = (1 - \alpha_1)\alpha_2 P(w_1) &< P(x_1 = 0, x_2 = 1|w_2)P(w_2) = (1 - \beta_1)\beta_2 P(w_2) \\
P(x_1 = 0, x_2 = 0|w_1)P(w_1) = (1 - \alpha_1)(1 - \alpha_2)P(w_1) &> P(x_1 = 0, x_2 = 0|w_2)P(w_2) = (1 - \beta_1)(1 - \beta_2)P(w_2)
\end{aligned} \tag{59}$$

因此 Bayes 决策规则为

$$\begin{aligned}
x_1 = 1,\ x_2 = 1 &\to (x_1, x_2) \in w_2 \\
x_1 = 1,\ x_2 = 0 &\to (x_1, x_2) \in w_2 \\
x_1 = 0,\ x_2 = 1 &\to (x_1, x_2) \in w_2 \\
x_1 = 0,\ x_2 = 0 &\to (x_1, x_2) \in w_1
\end{aligned} \tag{60}$$

同理可得, 使用特征 (x_1, x_2) 进行分类的 Bayes 决策规则为

$$\begin{aligned} x_1 = 1,\ x_3 = 1 \to (x_1, x_3) \in w_2 \\ x_1 = 1,\ x_3 = 0 \to (x_1, x_3) \in w_2 \\ x_1 = 0,\ x_3 = 1 \to (x_1, x_3) \in w_2 \\ x_1 = 0,\ x_3 = 0 \to (x_1, x_3) \in w_1 \end{aligned} \tag{61}$$

以及使用特征 (x_2, x_3) 进行分类的 Bayes 决策规则为

$$\begin{aligned} x_2 = 1,\ x_3 = 1 \to (x_2, x_3) \in w_2 \\ x_2 = 1,\ x_3 = 0 \to (x_2, x_3) \in w_2 \\ x_2 = 0,\ x_3 = 1 \to (x_2, x_3) \in w_2 \\ x_2 = 0,\ x_3 = 0 \to (x_2, x_3) \in w_1 \end{aligned} \tag{62}$$

所以错误率为

$$\begin{aligned} e(x_1, x_2) &= P(x_1 = 1, x_2 = 1, w_1) + P(x_1 = 1, x_2 = 0, w_1) + P(x_1 = 0, x_2 = 1, w_1) + P(x_1 = 0, x_2 = 0, w_2) \\ &= [1 - P(x_1 = 0, x_2 = 0|w_1)]P(w_1) + P(x_1 = 0, x_2 = 0|w_2)P(w_2) \\ &= [1 - P(x_1 = 0|w_1)P(x_2 = 0|w_1)]P(w_1) + P(x_1 = 0|w_2)P(x_2 = 0|w_2)P(w_2) \\ &= \frac{1}{2}[1 - (1 - \alpha_1)(1 - \alpha_2)] + \frac{1}{2}(1 - \beta_1)(1 - \beta_2) \\ &= \frac{1}{2} - \frac{1}{2}(\beta_1 - \alpha_1) - \frac{1}{2}(\beta_2 - \alpha_2) + \frac{1}{2}(\beta_1\beta_2 - \alpha_1\alpha_2) \\ &= 0.0825 \end{aligned} \tag{63}$$

同理可得

$$e(x_1, x_3) = \frac{1}{2} - \frac{1}{2}(\beta_1 - \alpha_1) - \frac{1}{2}(\beta_3 - \alpha_3) + \frac{1}{2}(\beta_1\beta_3 - \alpha_1\alpha_3) = 0.0695 \tag{64}$$

$$e(x_2, x_3) = \frac{1}{2} - \frac{1}{2}(\beta_2 - \alpha_2) - \frac{1}{2}(\beta_3 - \alpha_3) + \frac{1}{2}(\beta_2\beta_3 - \alpha_2\alpha_3) = 0.05975 \tag{65}$$

因此不同错误率的大小关系为

$$e(x_3) > e(x_2) > e(x_1) > e(x_1, x_2) > e(x_1, x_3) > e(x_2, x_3) \tag{66}$$

从特征选择的角度来看, 根据错误率的大小关系可知若使用单个特征进行分类, 则 x_1 最好, x_2 次之, x_3 最差. 使用两个特征进行分类的错误率均要小于使用单个特征的错误率, 可知增加特征可以使错误率减小, 提高分类准确率. 但是, 取最好的两个特征 x_1, x_2 进行组合的效果却并不是最好的, 反而是错误率最高的, 说明特征之间的相互作用关系对分类错误率有着很重要的影响.

Programming: Relief

4. Please read at least one of the papers about relief [1, 2]. Then implement a relief-based feature selection method and analyze the result on the dataset in the file `watermelon_3.csv` [3]. Finally, design a classifier on the selected feature space.

解: 数据集 `watermelon_3.csv` 如表 1 所示, 其中有 8 个特征, 共 17 个样本.

表 1: 西瓜数据表格

编号	色泽	根蒂	敲声	纹理	脐部	触感	密度	含糖率	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	0.697	0.46	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	0.774	0.376	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	0.634	0.264	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	0.608	0.318	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	0.556	0.215	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	0.403	0.237	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	0.481	0.149	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	0.437	0.211	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	0.666	0.091	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	0.243	0.267	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	0.245	0.057	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	0.343	0.099	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	0.639	0.161	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	0.657	0.198	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	0.36	0.37	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	0.593	0.042	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	0.719	0.103	否

应用 Relief 算法得到 8 个特征的权重为

$$\boldsymbol{w} = [-0.1275,\ 0.0294,\ -0.0294,\ 0.3235,\ 0.1275,\ -0.0588,\ 0.0001,\ 0.1495] \tag{67}$$

按照权重大小对 8 个特征进行排序为

$$w_4 > w_8 > w_5 > w_2 > w_7 > w_3 > w_6 > w_1 \tag{68}$$

取排序第一的第四个特征纹理 x_4 作为特征空间进行分类器设计, 令 $x_4 = 1, 2, 3$ 分别表示纹理清晰, 稍糊和模糊, 假设好瓜与否的先验概率相等 $P(\omega_1) = P(\omega_2)$, 其中 ω_1 表示好瓜, ω_2 表示坏瓜, 则可设计 Bayes 分类器.

由数据集可计算得

$$\begin{aligned} P(x_4=1|\omega_1)&=\frac{7}{8}>P(x_4=1|\omega_2)=\frac{2}{9}\\ P(x_4=2|\omega_1)&=\frac{1}{8}<P(x_4=2|\omega_2)=\frac{4}{9}\\ P(x_4=3|\omega_1)&=\frac{0}{8}<P(x_4=3|\omega_2)=\frac{3}{9} \end{aligned} \tag{69}$$

则 Bayes 分类器规则为

$$\begin{aligned} x_4=1&\rightarrow x_4\in\omega_1\\ x_4=2&\rightarrow x_4\in\omega_2\\ x_4=3&\rightarrow x_4\in\omega_2 \end{aligned} \tag{70}$$

若按照权重排序取前两个特征纹理 x_4 和含糖率 x_8 作为特征空间进行 SVM 分类器设计, 则可得到分类边界如图 1 所示, 可以看出分类边界是一竖直直线, 只有特征 x_4 起到了实际的分类作用, 与上述 Bayes 分类器等价.

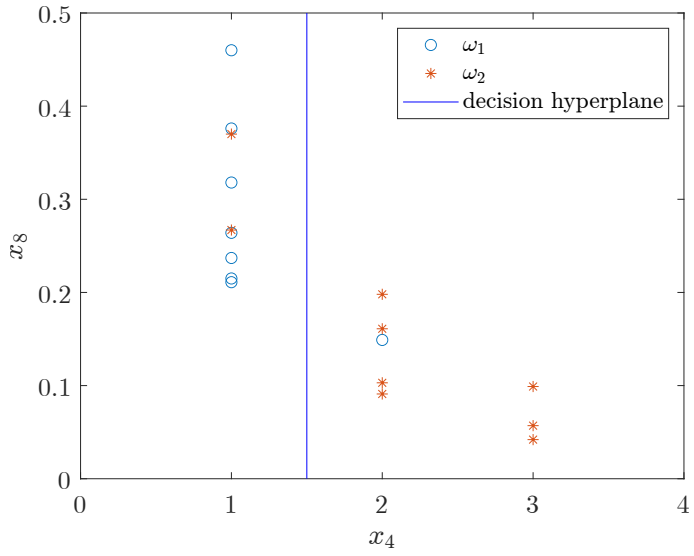


图 1: SVM 分类器

参考文献

[1] Robnik-Šikonja M, Kononenko I. Theoretical and empirical analysis of ReliefF and RReliefF[J]. Machine learning, 2003, 53(1): 23-69.

[2] Urbanowicz R J, Meeker M, La Cava W, et al. Relief-based feature selection: Introduction and review[J]. Journal of biomedical informatics, 2018, 85: 189-203.

[3] Zhihua Zhou. Machine Learning[J]. 2016. ISBN 978-730-24-2327-8.

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 8**

Deep Learning 1

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Backward Propagation Algorithm

1. We have a k-class classification problem. The input $\boldsymbol{x}$ is a d-dimensional vector, and the corresponding label $\boldsymbol{y}$ is a one-hot k-dimensional vector with one element being 1 and others being 0. We define the following neural network $\hat{\boldsymbol{y}} = f(\boldsymbol{x}; \boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2)$:

$$\boldsymbol{h}_1 = \boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1, \quad \boldsymbol{h}_1 \in \mathbb{R}^s \tag{1}$$

$$\boldsymbol{a}_1 = \mathrm{ReLU}(\boldsymbol{h}_1), \quad \boldsymbol{a}_1 \in \mathbb{R}^s \tag{2}$$

$$\boldsymbol{h}_2 = \mathrm{Concat}(\boldsymbol{a}_1, \boldsymbol{x}), \quad \boldsymbol{h}_2 \in \mathbb{R}^{s+d} \tag{3}$$

$$\boldsymbol{m} \sim \mathrm{Bernoulli}(p), \quad \boldsymbol{m} \in \mathbb{R}^{s+d} \tag{4}$$

$$\boldsymbol{a}_2 = \boldsymbol{h}_2 \odot \boldsymbol{m}, \quad \boldsymbol{a}_2 \in \mathbb{R}^{s+d} \tag{5}$$

$$\boldsymbol{h}_3 = \boldsymbol{W}_2^\top \boldsymbol{a}_2 + \boldsymbol{b}_2, \quad \boldsymbol{h}_3 \in \mathbb{R}^k \tag{6}$$

$$\hat{\boldsymbol{y}} = \mathrm{Softmax}(\boldsymbol{h}_3), \quad \hat{\boldsymbol{y}} \in \mathbb{R}^k \tag{7}$$

where $\mathrm{ReLU}(\cdot)$ is element-wise ReLU activation function, $\mathrm{Concat}(\boldsymbol{a}, \boldsymbol{b})$ means concatenating vector $\boldsymbol{b}$ after vector $\boldsymbol{a}$ to make a vector with larger dimensionality, $\boldsymbol{m}$ is current dropout mask, $\odot$ means element-wise multiplication(Specifically, equation(5) randomly zeroes some of the elements of the input tensor $\boldsymbol{h}_2$ with probability p using $\boldsymbol{m}$ sampled from a Bernoulli distribution. For example, if $\boldsymbol{h}_2 = [a, b, c, d]^\top$, and $\boldsymbol{m} = [e, f, g, h]^\top$, then $\boldsymbol{a}_2 = [ae, bf, cg, dh]^\top$ according to equation(5)), and $\mathrm{Softmax}(\cdot)$ is softmax activation function. The dimensionalities of parameters and internal variables could be inferred from the neural network definition.

We use the following cross-entropy loss:

$$L = -\boldsymbol{y}^\top \log \hat{\boldsymbol{y}}. \tag{8}$$

1.1. Please use backward propagation algorithm to find the following derivatives:

$$\frac{\partial L}{\partial \hat{\boldsymbol{y}}}, \quad \frac{\partial L}{\partial \boldsymbol{h}_3}, \quad \frac{\partial L}{\partial \boldsymbol{W}_2}, \quad \frac{\partial L}{\partial \boldsymbol{b}_2}, \quad \frac{\partial L}{\partial \boldsymbol{a}_2}, \quad \frac{\partial L}{\partial \boldsymbol{h}_2}, \quad \frac{\partial L}{\partial \boldsymbol{a}_1}, \quad \frac{\partial L}{\partial \boldsymbol{h}_1}, \quad \frac{\partial L}{\partial \boldsymbol{W}_1}, \quad \frac{\partial L}{\partial \boldsymbol{b}_1}, \quad \frac{\partial L}{\partial \boldsymbol{x}}. \tag{9}$$

Please express your results in matrices and vectors instead of sum of individual elements.

解: 原题目中 $\boldsymbol{w}_1, \boldsymbol{w}_2$ 实为矩阵, 故改用 $\boldsymbol{W}_1, \boldsymbol{W}_2$ 符号代替之. 为避免符号下标混淆, 本题中一律以右上角小括号内指标表示向量, 矩阵和张量的元素, 如 $\boldsymbol{a}^{(1)}$ 表示向量 $\boldsymbol{a}$ 的第 1 个元素, $\boldsymbol{A}^{(2,3)}$ 表示矩阵 $\boldsymbol{A}$ 的第 2 行第 3 列的元素, $\mathsf{A}^{(1,2,3)}$ 表示张量 A 的第 1 行第 2 列第 3 竖的元素.

交叉熵损失函数 L 对 $\hat{\boldsymbol{y}}$ 的偏导数为

$$\frac{\partial L}{\partial \hat{\boldsymbol{y}}} = \frac{\partial\big(-\boldsymbol{y}^\top \log \hat{\boldsymbol{y}}\big)}{\partial \hat{\boldsymbol{y}}} = -\frac{\partial \log \hat{\boldsymbol{y}}}{\partial \hat{\boldsymbol{y}}}\boldsymbol{y} = -\operatorname{diag}(\mathbf{1} \oslash \hat{\boldsymbol{y}})\,\boldsymbol{y} = -\boldsymbol{y} \oslash \hat{\boldsymbol{y}} \tag{10}$$

其中, 符号 $\oslash$ 表示逐项除法 (element-wise division).

函数 $\hat{\boldsymbol{y}}$ 对 $\boldsymbol{h}_3$ 的偏导数为

$$\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h}_3} = \frac{\partial}{\partial \boldsymbol{h}_3}\left(\frac{\exp(\boldsymbol{h}_3)}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \tag{11}$$

其中, $\forall\, i, j = 1, 2, \ldots, k$, 当 $i \neq j$ 时, 有

$$\begin{aligned}
\frac{\partial \hat{\boldsymbol{y}}^{(i)}}{\partial \boldsymbol{h}_3^{(j)}} &= \frac{\partial}{\partial \boldsymbol{h}_3^{(j)}}\left(\frac{\exp(\boldsymbol{h}_3^{(i)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \\
&= \frac{0 - \exp(\boldsymbol{h}_3^{(i)})\exp(\boldsymbol{h}_3^{(j)})}{[\mathbf{1}^\top \exp(\boldsymbol{h}_3)]^2} \\
&= -\frac{\exp(\boldsymbol{h}_3^{(i)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)} \\
&= -\hat{\boldsymbol{y}}^{(i)}\hat{\boldsymbol{y}}^{(j)}
\end{aligned} \tag{12}$$

当 $i = j$ 时, 有

$$\begin{aligned}
\frac{\partial \hat{\boldsymbol{y}}^{(j)}}{\partial \boldsymbol{h}_3^{(j)}} &= \frac{\partial}{\partial \boldsymbol{h}_3^{(j)}}\left(\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \\
&= \frac{\exp(\boldsymbol{h}_3^{(j)})\mathbf{1}^\top \exp(\boldsymbol{h}_3) - \exp(\boldsymbol{h}_3^{(j)})\exp(\boldsymbol{h}_3^{(j)})}{[\mathbf{1}^\top \exp(\boldsymbol{h}_3)]^2} \\
&= -\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)} + \frac{\exp(\boldsymbol{h}_3^{(j)})}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)} \\
&= -\hat{\boldsymbol{y}}^{(j)}\hat{\boldsymbol{y}}^{(j)} + \hat{\boldsymbol{y}}^{(j)}
\end{aligned} \tag{13}$$

综上, $\hat{\boldsymbol{y}}$ 对 $\boldsymbol{h}_3$ 的偏导数为

$$\begin{aligned}
\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h}_3} &= \frac{\partial}{\partial \boldsymbol{h}_3}\left(\frac{\exp(\boldsymbol{h}_3)}{\mathbf{1}^\top \exp(\boldsymbol{h}_3)}\right) \\
&= -\hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^\top + \operatorname{diag}(\hat{\boldsymbol{y}})
\end{aligned} \tag{14}$$

注意到 $\boldsymbol{y}$ 是 one-hot 向量, 则交叉熵损失函数 L 对 $\boldsymbol{h}_3$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{h}_3} &= \frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{h}_3}\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\ &= [-\hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^\top + \operatorname{diag}(\hat{\boldsymbol{y}})](-\boldsymbol{y} \oslash \hat{\boldsymbol{y}}) \\ &= \hat{\boldsymbol{y}}\hat{\boldsymbol{y}}^\top(\boldsymbol{y} \oslash \hat{\boldsymbol{y}}) - \operatorname{diag}(\hat{\boldsymbol{y}})(\boldsymbol{y} \oslash \hat{\boldsymbol{y}}) \\ &= \hat{\boldsymbol{y}}\sum_{i=1}^{k}\boldsymbol{y}^{(i)} - \boldsymbol{y} \\ &= \hat{\boldsymbol{y}} - \boldsymbol{y}\end{aligned} \tag{15}$$

函数 $\boldsymbol{h}_3$ 对 $\boldsymbol{W}_2$ 的偏导数为 3 阶张量 $\mathsf{W} \in \mathbb{R}^{(s+d)\times k\times k}$, 其元素为

$$\mathsf{W}^{(m,n,i)} = \frac{\partial \boldsymbol{h}_3^{(i)}}{\partial \boldsymbol{W}_2^{(m,n)}} = \frac{\partial\left[(\boldsymbol{W}_2^\top)^{(i,:)}\boldsymbol{a}_2 + \boldsymbol{b}_2^{(i)}\right]}{\partial \boldsymbol{W}_2^{(m,n)}} = \delta_{in}\boldsymbol{a}_2^{(m)}, \quad \forall\ i,n = 1,2,\ldots,k,\ m = 1,2,\ldots,s+d \tag{16}$$

其中, δ_{in} 为 Kronecker delta 符号.

故交叉熵损失函数 L 对 $\boldsymbol{W}_2^{(m,n)}$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}_2^{(m,n)}} &= \sum_{i=1}^{k}\frac{\partial \boldsymbol{h}_3^{(i)}}{\partial \boldsymbol{W}_2^{(m,n)}}\frac{\partial L}{\partial \boldsymbol{h}_3^{(i)}} \\ &= \sum_{i=1}^{k}\delta_{in}\boldsymbol{a}_2^{(m)}\left[\hat{\boldsymbol{y}}^{(i)} - \boldsymbol{y}^{(i)}\right] \\ &= \boldsymbol{a}_2^{(m)}\left[\hat{\boldsymbol{y}}^{(n)} - \boldsymbol{y}^{(n)}\right]\end{aligned} \tag{17}$$

所以, 交叉熵损失函数 L 对 $\boldsymbol{W}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_2} = \boldsymbol{a}_2(\hat{\boldsymbol{y}} - \boldsymbol{y})^\top \tag{18}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_2} = \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{b}_2}\frac{\partial L}{\partial \boldsymbol{h}_3} = \frac{\partial(\boldsymbol{W}_2^\top\boldsymbol{a}_2 + \boldsymbol{b}_2)}{\partial \boldsymbol{b}_2}\frac{\partial L}{\partial \boldsymbol{h}_3} = \boldsymbol{I}(\hat{\boldsymbol{y}} - \boldsymbol{y}) = \hat{\boldsymbol{y}} - \boldsymbol{y} \tag{19}$$

交叉熵损失函数 L 对 $\boldsymbol{a}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{a}_2} = \frac{\partial \boldsymbol{h}_3}{\partial \boldsymbol{a}_2}\frac{\partial L}{\partial \boldsymbol{h}_3} = \boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{20}$$

交叉熵损失函数 L 对 $\boldsymbol{h}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{h}_2} = \frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}_2}\frac{\partial L}{\partial \boldsymbol{a}_2} = \frac{\partial(\boldsymbol{h}_2 \odot \boldsymbol{m})}{\partial \boldsymbol{h}_2}\frac{\partial L}{\partial \boldsymbol{a}_2} = \operatorname{diag}(\boldsymbol{m})\boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) = \boldsymbol{m} \odot [\boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y})] \tag{21}$$

交叉熵损失函数 L 对 $\boldsymbol{a}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{a}_1} = \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{a}_1}\frac{\partial L}{\partial \boldsymbol{h}_2} = \frac{\partial \begin{bmatrix}\boldsymbol{a}_1 \\ \boldsymbol{x}\end{bmatrix}}{\partial \boldsymbol{a}_1}\frac{\partial L}{\partial \boldsymbol{h}_2} = [\boldsymbol{I}_{s\times s} \quad \boldsymbol{0}_{s\times d}]\,\mathrm{diag}(\boldsymbol{m})\boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{22}$$

交叉熵损失函数 L 对 $\boldsymbol{h}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{h}_1} = \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\frac{\partial L}{\partial \boldsymbol{a}_1} = \frac{\partial\ \mathrm{ReLU}(\boldsymbol{h}_1)}{\partial \boldsymbol{h}_1}\frac{\partial L}{\partial \boldsymbol{a}_1} = \mathrm{diag}\left(\frac{\boldsymbol{1} + \mathrm{sgn}(\boldsymbol{h}_1)}{2}\right)\frac{\partial L}{\partial \boldsymbol{a}_1} \tag{23}$$

函数 $\boldsymbol{h}_1$ 对 $\boldsymbol{W}_1$ 的偏导数为

$$\frac{\partial \boldsymbol{h}_1^{(i)}}{\partial \boldsymbol{W}_1^{(m,n)}} = \frac{\partial\left[(\boldsymbol{W}_1^\top)^{(i,:)}\boldsymbol{x} + \boldsymbol{b}_1^{(i)}\right]}{\partial \boldsymbol{W}_1^{(m,n)}} = \delta_{in}\boldsymbol{x}^{(m)}, \quad \forall\ i, n = 1, 2, \ldots, s,\ m = 1, 2, \ldots, d \tag{24}$$

故交叉熵损失函数 L 对 $\boldsymbol{W}_1^{(m,n)}$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{W}_1^{(m,n)}} &= \sum_{i=1}^{s}\frac{\partial \boldsymbol{h}_1^{(i)}}{\partial \boldsymbol{W}_1^{(m,n)}}\frac{\partial L}{\partial \boldsymbol{h}_1^{(i)}} \\ &= \sum_{i=1}^{s}\frac{\partial L}{\partial \boldsymbol{h}_1^{(i)}}\delta_{in}\boldsymbol{x}^{(m)} \\ &= \boldsymbol{x}^{(m)}\frac{\partial L}{\partial \boldsymbol{h}_1^{(n)}}\end{aligned} \tag{25}$$

所以, 交叉熵损失函数 L 对 $\boldsymbol{W}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_1} = \boldsymbol{x}\left(\frac{\partial L}{\partial \boldsymbol{h}_1}\right)^\top \tag{26}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_1} = \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{b}_1}\frac{\partial L}{\partial \boldsymbol{h}_1} = \frac{\partial(\boldsymbol{W}_1^\top\boldsymbol{x} + \boldsymbol{b}_1)}{\partial \boldsymbol{b}_1}\frac{\partial L}{\partial \boldsymbol{h}_1} = \boldsymbol{I}\frac{\partial L}{\partial \boldsymbol{h}_1} = \frac{\partial L}{\partial \boldsymbol{h}_1} \tag{27}$$

交叉熵损失函数 L 对 $\boldsymbol{x}$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{x}} &= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_1} + \frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_2} \\ &= \frac{\partial(\boldsymbol{W}_1^\top\boldsymbol{x} + \boldsymbol{b}_1)}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_1} + \frac{\partial \begin{bmatrix}\boldsymbol{a}_1 \\ \boldsymbol{x}\end{bmatrix}}{\partial \boldsymbol{x}}\frac{\partial L}{\partial \boldsymbol{h}_2} \\ &= \boldsymbol{W}_1\frac{\partial L}{\partial \boldsymbol{h}_1} + [\boldsymbol{0}_{d\times s} \quad \boldsymbol{I}_{d\times d}]\frac{\partial L}{\partial \boldsymbol{h}_2}\end{aligned} \tag{28}$$

1.2. Derive the back-propagation updates for convolutional neural networks: In this situation, the input $\boldsymbol{x}$ is an image with size $C_{\text{in}} \times H \times W$, where the three dimensions represent channel, height and width respectively. We define the following simple convolutional neural network:

$$\boldsymbol{u}_1 = \text{Conv2d}(C_{\text{in}}, C_{\text{out}}, k)(\boldsymbol{x}) \tag{29}$$

$$\boldsymbol{h}_1 = \text{MaxPool2d}(N)(\boldsymbol{u}_1) \tag{30}$$

$$\boldsymbol{a}_1 = \text{ReLU}(\boldsymbol{h}_1) \tag{31}$$

$$\boldsymbol{u}_2 = \text{Flatten}(\boldsymbol{a}_1) \tag{32}$$

$$\boldsymbol{h}_2 = \boldsymbol{W}_2^\top \boldsymbol{u}_2 + \boldsymbol{b}_2 \tag{33}$$

$$\hat{\boldsymbol{y}} = \text{Softmax}(\boldsymbol{h}_2) \tag{34}$$

where Conv2d (There are learnable parameters $\boldsymbol{W}_1, \boldsymbol{b}_1$ in this function. See this link for detailed definitions of the function arguments and this link for visualization) and MaxPool2d are convolutional function and pooling function defined in a PyTorch style, Flatten means reshaping the input vector into a one-dimensional vector, and the final loss function is the same with Eq. (8). What is the derivatives for the network parameters (i.e. convolutional and linear weights and biases)?

Hint: Read the following reference material: Notes on Convolutional Neural Networks.

解: 首先确定各个变量的维数, 输入图片 $\boldsymbol{x} \in \mathbb{R}^{C_{\text{in}} \times H \times W}$, Conv2d 函数输出 $\boldsymbol{u}_1 \in \mathbb{R}^{C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}}}$, 其中

$$H_{\text{out}} = H - k + 1, \quad W_{\text{out}} = W - k + 1 \tag{35}$$

MaxPool2d 函数输出 $\boldsymbol{h}_1 \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, 其中

$$H_{\text{mp}} = \left\lfloor \frac{H_{\text{out}}}{N} \right\rfloor, \quad W_{\text{mp}} = \left\lfloor \frac{W_{\text{out}}}{N} \right\rfloor \tag{36}$$

ReLU 函数输出 $\boldsymbol{a}_1 \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, Flatten 函数输出 $\boldsymbol{u}_2 \in \mathbb{R}^d$, 其中

$$d = C_{\text{out}} + H_{\text{mp}} + W_{\text{mp}} \tag{37}$$

设 $\boldsymbol{W}_2 \in \mathbb{R}^{d \times s}$, 则 $\boldsymbol{b}_2 \in \mathbb{R}^s$, $\boldsymbol{h}_2 \in \mathbb{R}^s$, $\hat{\boldsymbol{y}} \in \mathbb{R}^s$.

由 1.1 题可知, 交叉熵损失函数 L 对 $\boldsymbol{W}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_2} = \boldsymbol{a}_2(\hat{\boldsymbol{y}} - \boldsymbol{y})^\top \tag{38}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_2} = \hat{\boldsymbol{y}} - \boldsymbol{y} \tag{39}$$

交叉熵损失函数 L 对 $\boldsymbol{u}_2$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{u}_2} = \boldsymbol{W}_2(\hat{\boldsymbol{y}} - \boldsymbol{y}) \tag{40}$$

函数 $\boldsymbol{u}_2$ 对 $\boldsymbol{a}_1$ 的偏导数为 4 阶张量 $\mathsf{A} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}} \times d}$, 其元素为

$$\mathsf{A}^{(i,j,k,m)} = \frac{\partial \boldsymbol{u}_2^{(m)}}{\partial \boldsymbol{a}_1^{(i,j,k)}} = \frac{\partial [\text{Flatten}(\boldsymbol{a}_1)]^{(m)}}{\partial \boldsymbol{a}_1^{(i,j,k)}} = \delta_{mn(i,j,k)} \tag{41}$$

其中 $n(\cdot,\cdot,\cdot)$ 是三元函数,

$$n(i,j,k) \triangleq (i-1)H_{\text{mp}}W_{\text{mp}} + (j-1)W_{\text{mp}} + k \tag{42}$$

交叉熵损失函数 L 对 $\boldsymbol{a}_1$ 的偏导数为

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{a}_1^{(i,j,k)}} &= \sum_{m=1}^{d} \frac{\partial \boldsymbol{u}_2^{(m)}}{\partial \boldsymbol{a}_1^{(i,j,k)}} \frac{\partial L}{\partial \boldsymbol{u}_2^{(m)}} \\ &= \sum_{m=1}^{d} \delta_{mn(i,j,k)} \boldsymbol{W}_2^{(m,:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \\ &= \boldsymbol{W}_2^{(n(i,j,k),:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \end{aligned} \tag{43}$$

函数 $\boldsymbol{a}_1$ 对 $\boldsymbol{h}_1$ 的偏导数为 6 阶张量 $\mathsf{H} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}} \times C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, 其元素为

$$\mathsf{H}^{(r,s,t,i,j,k)} = \frac{\partial \boldsymbol{a}_1^{(i,j,k)}}{\partial \boldsymbol{h}_1^{(r,s,t)}} = \frac{\partial \ \text{ReLU}(\boldsymbol{h}_1^{(i,j,k)})}{\partial \boldsymbol{h}_1^{(r,s,t)}} = \delta_{ir}\delta_{js}\delta_{kt} \frac{1 + \text{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \tag{44}$$

交叉熵损失函数 L 对 $\boldsymbol{h}_1$ 的偏导数为

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{h}_1^{(r,s,t)}} &= \sum_{i=1}^{C_{\text{out}}} \sum_{j=1}^{H_{\text{mp}}} \sum_{k=1}^{W_{\text{mp}}} \frac{\partial \boldsymbol{a}_1^{(i,j,k)}}{\partial \boldsymbol{h}_1^{(r,s,t)}} \frac{\partial L}{\partial \boldsymbol{a}_1^{(i,j,k)}} \\ &= \sum_{i=1}^{C_{\text{out}}} \sum_{j=1}^{H_{\text{mp}}} \sum_{k=1}^{W_{\text{mp}}} \delta_{ir}\delta_{js}\delta_{kt} \frac{1 + \text{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \boldsymbol{W}_2^{(n(i,j,k),:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \\ &= \frac{1 + \text{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \boldsymbol{W}_2^{(n(r,s,t),:)} (\hat{\boldsymbol{y}} - \boldsymbol{y}) \end{aligned} \tag{45}$$

函数 $\boldsymbol{h}_1$ 对 $\boldsymbol{u}_1$ 的偏导数为 6 阶张量 $\mathsf{U} \in \mathbb{R}^{C_{\text{out}} \times H_{\text{out}} \times W_{\text{out}} \times C_{\text{out}} \times H_{\text{mp}} \times W_{\text{mp}}}$, 其元素为

$$\begin{aligned} \mathsf{U}^{(p,q,l,r,s,t)} &= \frac{\partial \boldsymbol{h}_1^{(r,s,t)}}{\partial \boldsymbol{u}_1^{(p,q,l)}} \\ &= \frac{\partial \ \text{MaxPool2d}(\boldsymbol{h}_1^{(r,s,t)})}{\partial \boldsymbol{u}_1^{(p,q,l)}} \\ &= \begin{cases} \delta_{\boldsymbol{h}_1^{(r,s,t)}) \boldsymbol{u}_1^{(p,q,l)}}, & \text{if } \ p-1 < \dfrac{r}{N} \leqslant p, \ q-1 < \dfrac{s}{N} \leqslant q, \ l-1 < \dfrac{t}{N} \leqslant l \\ 0, & \text{o.w.} \end{cases} \end{aligned} \tag{46}$$

交叉熵损失函数 L 对 $\boldsymbol{u}_1$ 的偏导数为

$$\begin{aligned}\frac{\partial L}{\partial \boldsymbol{u}_1^{(p,q,l)}} &= \sum_{r=1}^{C_{\text{out}}}\sum_{s=1}^{H_{\text{mp}}}\sum_{t=1}^{W_{\text{mp}}} \frac{\partial \boldsymbol{h}_1^{(r,s,t)}}{\partial \boldsymbol{u}_1^{(p,q,l)}} \frac{\partial L}{\partial \boldsymbol{h}_1^{(r,s,t)}} \\ &= \sum_{r=1}^{C_{\text{out}}}\sum_{s=1}^{H_{\text{mp}}}\sum_{t=1}^{W_{\text{mp}}} \frac{\partial \boldsymbol{h}_1^{(r,s,t)}}{\partial \boldsymbol{u}_1^{(p,q,l)}} \frac{1+\operatorname{sgn}(\boldsymbol{h}_1^{(r,s,t)})}{2} \boldsymbol{W}_2^{(n(r,s,t),:)}(\hat{\boldsymbol{y}}-\boldsymbol{y})\end{aligned} \tag{47}$$

由函数 Conv2d 定义可知

$$\boldsymbol{u}_1^{(j,:,:)} = \boldsymbol{b}_1^{(j,:,:)} + \sum_{k=1}^{C_{\text{in}}} \boldsymbol{W}_1^{(j,k,:,:)} \star \boldsymbol{x}^{(k,:,:)} \tag{48}$$

其中, $\star$ 为二维互相关 (cross-correlation) 运算符号,

$$\boldsymbol{b}_1 \in \mathbb{R}^{C_{\text{out}}\times H_{\text{out}}\times W_{\text{out}}}, \quad \boldsymbol{W}_1 \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times k\times k} \tag{49}$$

交叉熵损失函数 L 对 $\boldsymbol{b}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{b}_1} = \frac{\partial L}{\partial \boldsymbol{u}_1} \tag{50}$$

函数 $\boldsymbol{u}_1$ 对 $\boldsymbol{W}_1$ 的偏导数为 7 阶张量 $\mathsf{W} \in \mathbb{R}^{C_{\text{out}}\times C_{\text{in}}\times k\times k\times C_{\text{out}}\times H_{\text{out}}\times W_{\text{out}}}$, 难以具体表出, 根据参考文献 [2] 可知可以使用 MATLAB 内置函数直接计算交叉熵损失函数 L 对 $\boldsymbol{W}_1$ 的偏导数.

交叉熵损失函数 L 对 $\boldsymbol{W}_1$ 的偏导数为

$$\frac{\partial L}{\partial \boldsymbol{W}_1^{(m,n,:,:)}} = \text{rot180}\left\{\text{conv2}\left[\boldsymbol{x}^{(n,:,:)},\ \text{rot180}\left(\frac{\partial L}{\partial \boldsymbol{u}_1^{(m,:,:)}}\right),\ \text{'valid'}\right]\right\} \tag{51}$$

其中, rot180 和 conv2 均为 MATLAB 中的函数.

Vanishing and Exploding Gradients

2. In this section we will investigate the vanishing/exploding gradient problem and see why ReLU and ResNet can mitigate this problem. The vanishing/exploding gradient problem often occurs during training very deep neural networks and could make the training process failed.

Suppose the input $\boldsymbol{x}$ and output $\hat{\boldsymbol{y}}$ are both d-dimensional vectors, and the gradient of loss w.r.t. $\hat{\boldsymbol{y}}$: $\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \in \mathbb{R}^d$ is known. Suppose the deep feed-forward neural network has l layers (i.e., $l > 100$), and all internal nodes are also d-dimensional. All weights are initialized with zero-mean Gaussian distribution and all biases are initialized to zero. We now analyze the gradients in the first optimization iteration.

Effect of ReLU. The neural network is defined by:

$$\begin{aligned} \boldsymbol{a}_1 &= \text{Sigmoid}(\boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1) \\ \boldsymbol{a}_2 &= \text{Sigmoid}(\boldsymbol{W}_2^\top \boldsymbol{a}_1 + \boldsymbol{b}_2) \\ &\vdots \\ \hat{\boldsymbol{y}} = \boldsymbol{a}_l &= \text{Sigmoid}(\boldsymbol{W}_l^\top \boldsymbol{a}_{l-1} + \boldsymbol{b}_l) \end{aligned} \tag{52}$$

2.1. Find $\frac{\partial L}{\partial \boldsymbol{W}_1}$.

解: 对 $z \in \mathbb{R}$, Sigmoid 函数定义为

$$\text{Sigmoid}(z) = \frac{1}{1+\exp(-z)} \tag{53}$$

其对 z 的偏导数为

$$\begin{aligned} \frac{\partial\ \text{Sigmoid}(z)}{\partial z} &= \frac{-[-\exp(-z)]}{[1+\exp(z)]^2} \\ &= \frac{1}{1+\exp(-z)} \frac{\exp(-z)}{1+\exp(-z)} \\ &= \text{Sigmoid}(z)[1-\text{Sigmoid}(z)] \end{aligned} \tag{54}$$

对 $\boldsymbol{z} \in \mathbb{R}^d$, Sigmoid 函数可用逐项除法 $\oslash$ 符号表示为

$$\text{Sigmoid}(\boldsymbol{z}) = \boldsymbol{1} \oslash [\boldsymbol{1} + \exp(-\boldsymbol{z})] \tag{55}$$

则由式 (54) 可知 $\text{Sigmoid}(\boldsymbol{z})$ 对 $\boldsymbol{z}$ 的偏导数为

$$\frac{\partial\ \text{Sigmoid}(\boldsymbol{z})}{\partial \boldsymbol{z}} = \text{diag}\left\{\text{Sigmoid}(\boldsymbol{z}) \odot \left[\boldsymbol{1} - \text{Sigmoid}(\boldsymbol{z})\right]\right\} \tag{56}$$

记 $\boldsymbol{a}_0 \triangleq \boldsymbol{x}$, 令

$$\boldsymbol{h}_i \triangleq \boldsymbol{W}_i^\top \boldsymbol{a}_{i-1} + \boldsymbol{b}_i, \quad \forall\ i = 1, 2, \ldots, l \tag{57}$$

则

$$\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}} = \boldsymbol{W}_i, \quad \forall\ i = 1, 2, \ldots, l \tag{58}$$

又由式 (56) 可知

$$\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i} = \frac{\partial\ \text{Sigmoid}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i} = \text{diag}[\boldsymbol{a}_i \odot (\boldsymbol{1} - \boldsymbol{a}_i)], \quad \forall\ i = 1, 2, \ldots, l \tag{59}$$

则由链式法则及式 (18) 可得

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}_1} &= \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{a}_1}\cdots\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_{l-1}}\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \left(\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\right)\left(\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{a}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}_2}\right)\cdots\left(\frac{\partial \boldsymbol{h}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{h}_{l-1}}\right)\left(\frac{\partial \boldsymbol{h}_l}{\partial \boldsymbol{a}_{l-1}}\frac{\partial \boldsymbol{a}_l}{\partial \boldsymbol{h}_l}\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\left(\prod_{i=2}^{l}\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}}\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i}\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\operatorname{diag}\left[\boldsymbol{a}_1\odot(\boldsymbol{1}-\boldsymbol{a}_1)\right]\left(\prod_{i=2}^{l}\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\boldsymbol{1}-\boldsymbol{a}_i)\right]\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\left(\prod_{i=2}^{l}\boldsymbol{W}_i\right)\left(\prod_{j=1}^{l}\operatorname{diag}\left[\boldsymbol{a}_j\odot(\boldsymbol{1}-\boldsymbol{a}_j)\right]\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \boldsymbol{x}\left[\left(\prod_{i=2}^{l}\boldsymbol{W}_i\right)\left(\prod_{j=1}^{l}\operatorname{diag}\left[\boldsymbol{a}_j\odot(\boldsymbol{1}-\boldsymbol{a}_j)\right]\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}}\right]^{\top}
\end{aligned} \tag{60}$$

2.2. Suppose $z\in\mathbb{R}$ is a scalar, find the maximum of sigmoid function's derivative: $g(z)=\text{Sigmoid}'(z)$.

解: 由式 (54) 可知

$$g(z)=\text{Sigmoid}'(z)=\text{Sigmoid}(z)[1-\text{Sigmoid}(z)] \tag{61}$$

根据基本不等式可得

$$g(z)=\text{Sigmoid}(z)[1-\text{Sigmoid}(z)]\leqslant\frac{\left[\text{Sigmoid}(z)+1-\text{Sigmoid}(z)\right]^2}{4}=\frac{1}{4} \tag{62}$$

等号成立当且仅当

$$\text{Sigmoid}(z)=1-\text{Sigmoid}(z) \tag{63}$$

即

$$\text{Sigmoid}(z)=\frac{1}{2} \tag{64}$$

即

$$z=0 \tag{65}$$

故当 $z=0$ 时, Sigmoid 函数的梯度 $g(z)$ 取到最大值

$$\max_{z\in\mathbb{R}}g(z)=g(0)=\frac{1}{4} \tag{66}$$

2.3. Since many sigmoid functions' derivatives are multiplied together in $\frac{\partial L}{\partial \boldsymbol{W}_1}$, the values in $\frac{\partial L}{\partial \boldsymbol{W}_1}$ will tend to vanish to zero. Is there any possibility that we can initialize weight matrices carefully to recover the values into a good range for float32 (e.g., -0.01 to 1000)? (*Hint: Since the sigmoid function saturates in both sides,*

in order to keep the gradient scale "reasonable", one needs to make sure the values before sigmoid function lie in non-saturation region.)

解: 由上题可知 Sigmoid(z) 函数的最大梯度位于 $z=0$ 处, 且 Sigmoid(0) = 1/2, 即要求

$$\begin{aligned} \boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1 &= \boldsymbol{0} \\ \boldsymbol{W}_i^\top \frac{\boldsymbol{1}}{2} + \boldsymbol{b}_i &= \boldsymbol{0}, \quad \forall\, i = 2, 3, \ldots, l \end{aligned} \tag{67}$$

初始化时偏置 $\boldsymbol{b}_i = \boldsymbol{0}$, $\forall\, i = 1, 2, \ldots, l$, 则初始化权重矩阵需满足

$$\begin{aligned} \boldsymbol{W}_1^\top \boldsymbol{x} &= \boldsymbol{0} \\ \boldsymbol{W}_i^\top \boldsymbol{1} &= \boldsymbol{0}, \quad \forall\, i = 2, 3, \ldots, l \end{aligned} \tag{68}$$

此时获得的 Sigmoid 函数梯度连乘是最大的, 但由于

$$\max_{z \in \mathbb{R}} g(z) = \frac{1}{4} \tag{69}$$

又 $l > 100$, $\forall\, z_j \in \mathbb{R},\ j = 1, 2, \ldots, l$ 有

$$\prod_{j=1}^{l} g(z_j) < \prod_{j=1}^{l} \max_{z_j \in \mathbb{R}} g(z_j) = 2^{-2l} < 2^{-200} \ll 2^{-126} \tag{70}$$

故当层数很多时, 在梯度都取最大值这种最好的情况下也几乎不可能把梯度值恢复到对 float32 精度友好的数值范围 (e.g., -0.01 to 1000) 内, 甚至会远远超出 float32 精度的表示范围 ($2^{-126} \sim 2^{127}$).

2.4. If our computer supports efficient arbitrary precision floating point calculation, do we need to care about vanishing gradient problem anymore?

解: 需要考虑, 因为即便计算精确, 梯度 $\frac{\partial L}{\partial \boldsymbol{W}_1}$ 过小也会使得权重矩阵的更新过程

$$\boldsymbol{W}_1 \leftarrow \boldsymbol{W}_1 - \eta \frac{\partial L}{\partial \boldsymbol{W}_1} \tag{71}$$

极为缓慢, 最终导致整个神经网络长时间无法收敛.

2.5. Can we alleviate the vanishing problem by replacing the activation function from sigmoid to ReLU? Why?

解: 可以缓解一部分. 此时

$$\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i} = \frac{\partial\ \mathrm{ReLU}(\boldsymbol{h}_i)}{\partial \boldsymbol{h}_i} = \mathrm{diag}\left(\frac{\boldsymbol{1} + \mathrm{sgn}(\boldsymbol{a}_j)}{2}\right), \quad \forall\, i = 1, 2, \ldots, l \tag{72}$$

则

$$\begin{aligned} \frac{\partial L}{\partial \boldsymbol{W}_1} &= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1} \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1} \left(\prod_{i=2}^{l} \frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}} \frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i}\right) \frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\ &= \boldsymbol{x} \left[\left(\prod_{i=2}^{l} \boldsymbol{W}_i\right) \left(\prod_{j=1}^{l} \mathrm{diag}\left(\frac{\boldsymbol{1} + \mathrm{sgn}(\boldsymbol{a}_j)}{2}\right)\right) \frac{\partial L}{\partial \hat{\boldsymbol{y}}}\right]^\top \end{aligned} \tag{73}$$

所以, ReLU 函数梯度连乘的部分

$$\prod_{j=1}^{l} \operatorname{diag}\left(\frac{\mathbf{1}+\operatorname{sgn}(\boldsymbol{a}_j)}{2}\right)$$

不会消失, 但若 $\det(\boldsymbol{W}_i)$ 过小, 则权重矩阵连乘的部分

$$\prod_{i=2}^{l} \boldsymbol{W}_i$$

会消失, 这也会导致梯度的消失. 因此, 把 Sigmoid 函数换成 ReLU 函数可以缓解一部分梯度消失的问题, 但不能完全避免这个问题.

Effect of ResNet. The neural network now has skip connections:

$$\begin{aligned}
\boldsymbol{a}_1 &= \operatorname{Sigmoid}(\boldsymbol{W}_1^\top \boldsymbol{x} + \boldsymbol{b}_1) + \boldsymbol{x} \\
\boldsymbol{a}_2 &= \operatorname{Sigmoid}(\boldsymbol{W}_2^\top \boldsymbol{a}_1 + \boldsymbol{b}_2) + \boldsymbol{a}_1 \\
&\vdots \\
\hat{\boldsymbol{y}} = \boldsymbol{a}_l &= \operatorname{Sigmoid}(\boldsymbol{W}_l^\top \boldsymbol{a}_{l-1} + \boldsymbol{b}_l) + \boldsymbol{a}_{l-1}
\end{aligned} \tag{74}$$

2.7. Find $\frac{\partial L}{\partial \boldsymbol{W}_1}$ and explain why the gradients can not easily vanish.

解: 与 2.1 题类似, 由链式法则及式 (18) 可得

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{W}_1} &= \frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{a}_1}\cdots\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \hat{\boldsymbol{y}}}{\partial \boldsymbol{a}_{l-1}}\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \left(\frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\right)\left(\frac{\partial \boldsymbol{h}_2}{\partial \boldsymbol{a}_1}\frac{\partial \boldsymbol{a}_2}{\partial \boldsymbol{h}_2}+\boldsymbol{I}\right)\cdots\left(\frac{\partial \boldsymbol{h}_{l-1}}{\partial \boldsymbol{a}_{l-2}}\frac{\partial \boldsymbol{a}_{l-1}}{\partial \boldsymbol{h}_{l-1}}+\boldsymbol{I}\right)\left(\frac{\partial \boldsymbol{h}_l}{\partial \boldsymbol{a}_{l-1}}\frac{\partial \boldsymbol{a}_l}{\partial \boldsymbol{h}_l}+\boldsymbol{I}\right)\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\frac{\partial \boldsymbol{a}_1}{\partial \boldsymbol{h}_1}\left[\prod_{i=2}^{l}\left(\frac{\partial \boldsymbol{h}_i}{\partial \boldsymbol{a}_{i-1}}\frac{\partial \boldsymbol{a}_i}{\partial \boldsymbol{h}_i}+\boldsymbol{I}\right)\right]\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \frac{\partial \boldsymbol{h}_1}{\partial \boldsymbol{W}_1}\operatorname{diag}\left[\boldsymbol{a}_1\odot(\mathbf{1}-\boldsymbol{a}_1)\right]\left[\prod_{i=2}^{l}\left(\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\mathbf{1}-\boldsymbol{a}_i)\right]+\boldsymbol{I}\right)\right]\frac{\partial L}{\partial \hat{\boldsymbol{y}}} \\
&= \boldsymbol{x}\left\{\operatorname{diag}\left[\boldsymbol{a}_1\odot(\mathbf{1}-\boldsymbol{a}_1)\right]\left[\prod_{i=2}^{l}\left(\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\mathbf{1}-\boldsymbol{a}_i)\right]+\boldsymbol{I}\right)\right]\frac{\partial L}{\partial \hat{\boldsymbol{y}}}\right\}^\top
\end{aligned} \tag{75}$$

所以, 连乘部分

$$\prod_{i=2}^{l}\left(\boldsymbol{W}_i\operatorname{diag}\left[\boldsymbol{a}_i\odot(\mathbf{1}-\boldsymbol{a}_i)\right]+\boldsymbol{I}\right)$$

中的每一项都加上了一个单位矩阵, 则此部分轻易不会消失, 因此 ResNet 可以有效防止梯度消失的问题.

Programming: Image Classification with CIFAR-10

Please install PyTorch and run the CIFAR-10 tutorial. If you want to use TensorFlow or other deep learning frameworks, please find corresponding CIFAR-10 tutorial for that framework and run it.

If you successfully run the tutorial, write "Success" in the report and otherwise please write "Failed". For this assignment report, only this single word is required, no details/accuracies/codes are needed. The assignment intends to get you familiar with the deep learning library. If you have already trained some deep classifiers using any deep learning library before, you can safely ignore the tutorial and directly write "Success" in the report.

解: Success.

参考文献

[1] Dimitri P. Bertsekas. Nonlinear Programming: 3rd Edition[M]. Athena Scientific, 2016. ISBN: 978-1-886529-05-2, http://www.athenasc.com/nonlinbook.html.

[2] Bouvrie, Jake. Notes on Convolutional Neural Networks. CogPrints.org, 2006. http://cogprints.org/5869/1/cnn_tutorial.pdf.

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 9**

Deep Learning 2

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Recurrent Neural Network (RNN)

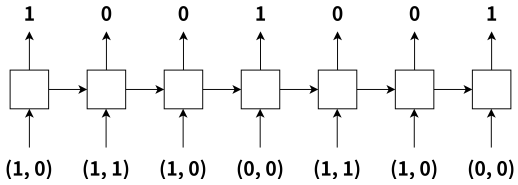
1. In this problem, you will implement a recurrent neural network which implements binary addition. The inputs are given as binary sequences, starting with the least significant binary digit. (It is easier to start from the least significant bit, just like how you did addition in grade school.) The sequences will be padded with at least one zero on the end. For instance, the problem

$$110111 + 10010 = 1001001 \tag{1}$$

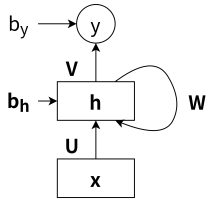
would be represented as:

- **Input 1**: 1, 1, 1, 0, 1, 1, 0
- **Input 2**: 0, 1, 0, 0, 1, 0, 0
- **Correct output**: 1, 0, 0, 1, 0, 0, 1

There are two input units corresponding to the two inputs, and one output unit. Therefore, the pattern of inputs and outputs for this example would be:



Design the weights and biases for an RNN which has two input units, three hidden units, and one output unit, which implements binary addition. All the units use the hard threshold activation function. In particular,



specify the values of weight matrices $\boldsymbol{U}, \boldsymbol{V}$, and $\boldsymbol{W}$, bias vector $\boldsymbol{b_h}$, and scalar bias b_y. The details of the architecture and the computation are as follows:

$$\begin{aligned} \boldsymbol{h}^{(t)} &= \text{hard}(\boldsymbol{U}\boldsymbol{x}^{(t)} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{b_h}) \\ y^{(t)} &= \text{hard}(\boldsymbol{V}\boldsymbol{h}^{(t)} + b_y) \end{aligned} \tag{2}$$

$$\text{hard}(x) = \begin{cases} 0, & \text{if } x < 0 \\ 1, & \text{if } x \geqslant 0 \end{cases} \tag{3}$$

If $\boldsymbol{x}$ is a vector, hard($\boldsymbol{x}$) represents element-wise operation. And we initialize $\boldsymbol{h}^{(0)}$ as $\mathbf{0}$.

Hint: One simple implementation of the first layer is that you just add up the values of each unit, including the carry. Activate the first one of your hidden units if the sum is at least 1, the second one if it is at least 2, and the third one if it is 3. Obviously, the answer is not unique, and you are encouraged to find more interesting implementation, but one is enough for this homework.

解: 根据提示, 可设计 RNN 如下.

$$\boldsymbol{U} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \boldsymbol{V} = \begin{bmatrix} 1 & -1 & 1 \end{bmatrix}, \quad \boldsymbol{W} = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \boldsymbol{b_h} = \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}, \quad b_y = -\frac{1}{2} \tag{4}$$

初始化 $\boldsymbol{h}^{(0)} = \mathbf{0}$, 对于示例题目可计算如下. 输入为

$$\boldsymbol{x} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{5}$$

正确输出为

$$\boldsymbol{y} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

第 1 步计算:

$$\boldsymbol{h}^{(1)} = \text{hard}\left(\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 0 \end{bmatrix} + \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 \\ -2 \\ -3 \end{bmatrix}\right) = \text{hard}\left(\begin{bmatrix} 0 \\ -1 \\ -2 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{7}$$

$$y^{(1)} = \text{hard}\left(\begin{bmatrix}1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} - \frac{1}{2}\right) = \text{hard}\left(\frac{1}{2}\right) = 1 \tag{8}$$

第 2 步计算:

$$\boldsymbol{h}^{(2)} = \text{hard}\left(\begin{bmatrix}1 & 1\\1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}-1\\-2\\-3\end{bmatrix}\right) = \text{hard}\left(\begin{bmatrix}3/2\\1/2\\-1/2\end{bmatrix}\right) = \begin{bmatrix}1\\1\\0\end{bmatrix} \tag{9}$$

$$y^{(2)} = \text{hard}\left(\begin{bmatrix}1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} - \frac{1}{2}\right) = \text{hard}\left(-\frac{1}{2}\right) = 0 \tag{10}$$

第 3 步计算:

$$\boldsymbol{h}^{(3)} = \text{hard}\left(\begin{bmatrix}1 & 1\\1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}-1\\-2\\-3\end{bmatrix}\right) = \text{hard}\left(\begin{bmatrix}1\\0\\-1\end{bmatrix}\right) = \begin{bmatrix}1\\1\\0\end{bmatrix} \tag{11}$$

$$y^{(3)} = \text{hard}\left(\begin{bmatrix}1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} - \frac{1}{2}\right) = \text{hard}\left(-\frac{1}{2}\right) = 0 \tag{12}$$

第 4 步计算:

$$\boldsymbol{h}^{(4)} = \text{hard}\left(\begin{bmatrix}1 & 1\\1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}-1\\-2\\-3\end{bmatrix}\right) = \text{hard}\left(\begin{bmatrix}0\\-1\\-2\end{bmatrix}\right) = \begin{bmatrix}1\\0\\0\end{bmatrix} \tag{13}$$

$$y^{(4)} = \text{hard}\left(\begin{bmatrix}1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} - \frac{1}{2}\right) = \text{hard}\left(\frac{1}{2}\right) = 1 \tag{14}$$

第 5 步计算:

$$\boldsymbol{h}^{(5)} = \text{hard}\left(\begin{bmatrix}1 & 1\\1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}-1\\-2\\-3\end{bmatrix}\right) = \text{hard}\left(\begin{bmatrix}3/2\\1/2\\-1/2\end{bmatrix}\right) = \begin{bmatrix}1\\1\\0\end{bmatrix} \tag{15}$$

$$y^{(5)} = \text{hard}\left(\begin{bmatrix}1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} - \frac{1}{2}\right) = \text{hard}\left(-\frac{1}{2}\right) = 0 \tag{16}$$

第 6 步计算:

$$\boldsymbol{h}^{(6)} = \text{hard}\left(\begin{bmatrix}1 & 1\\1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}-1\\-2\\-3\end{bmatrix}\right) = \text{hard}\left(\begin{bmatrix}1\\0\\-1\end{bmatrix}\right) = \begin{bmatrix}1\\1\\0\end{bmatrix} \tag{17}$$

$$y^{(6)} = \text{hard}\left(\begin{bmatrix}1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} - \frac{1}{2}\right) = \text{hard}\left(-\frac{1}{2}\right) = 0 \tag{18}$$

第 7 步计算:

$$\boldsymbol{h}^{(7)} = \text{hard}\left(\begin{bmatrix}1 & 1\\1 & 1\\1 & 1\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix} + \frac{1}{2}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}\begin{bmatrix}1\\1\\0\end{bmatrix} + \begin{bmatrix}-1\\-2\\-3\end{bmatrix}\right) = \text{hard}\left(\begin{bmatrix}0\\-1\\-2\end{bmatrix}\right) = \begin{bmatrix}1\\0\\0\end{bmatrix} \tag{19}$$

$$y^{(7)} = \text{hard}\left(\begin{bmatrix}1 & -1 & 1\end{bmatrix}\begin{bmatrix}1\\0\\0\end{bmatrix} - \frac{1}{2}\right) = \text{hard}\left(\frac{1}{2}\right) = 1 \tag{20}$$

由于

$$\boldsymbol{y} = \begin{bmatrix}1 & 0 & 0 & 1 & 0 & 0 & 1\end{bmatrix} = \begin{bmatrix}y^{(1)} & y^{(2)} & y^{(3)} & y^{(4)} & y^{(5)} & y^{(6)} & y^{(7)}\end{bmatrix} \tag{21}$$

则上述设计的 RNN 可以实现二进制加法.

Long-Term Short-Term Memory (LSTM)

2. Here, you'll derive the Backprop Through Time (BPTT) equations for the univariate version of the Long-Term Short-Term Memory (LSTM) architecture.

Note: This question is an important context for understanding LSTMs, but it is just ordinary BPTT, so you have enough knowledge to do parts (a), (b). As for parts (c), you may find it helpful to read more materials about LSTM.

For reference, here are the computations it performs for inputs $x^{(t)}$, $t = 1, 2, \ldots, T$:

$$\begin{aligned} i^{(t)} &= \sigma\left(w_{ix}x^{(t)} + w_{ih}h^{(t-1)}\right) \\ f^{(t)} &= \sigma\left(w_{fx}x^{(t)} + w_{fh}h^{(t-1)}\right) \\ o^{(t)} &= \sigma\left(w_{ox}x^{(t)} + w_{oh}h^{(t-1)}\right) \\ g^{(t)} &= \tanh\left(w_{gx}x^{(t)} + w_{gh}h^{(t-1)}\right) \\ c^{(t)} &= f^{(t)}c^{(t-1)} + i^{(t)}g^{(t)} \\ h^{(t)} &= o^{(t)}\tanh\left(c^{(t)}\right) \end{aligned} \tag{22}$$

And the loss function $\mathcal{L}$ with label y is:

$$\mathcal{L} = \frac{1}{2}\left(y - h^{(T)}\right)^2 \tag{23}$$

A slightly more convenient notation:

- σ is the activation function, we use Sigmoid function here.
- Use $\bar{y}$ to denote the derivative $\partial\mathcal{L}/\partial y$, sometimes called the error signal, where $\mathcal{L}$ is the loss function, y can be any intermediate variable. This emphasizes that the error signals are just values our program is computing rather than a mathematical operation.
- As an example, we compute the loss:

  $$\begin{aligned} z &= wx + b \\ y &= \sigma(z) \\ \mathcal{L} &= \frac{1}{2}(y - y_{\text{pred}})^2 \end{aligned} \tag{24}$$

  Then we could compute the derivatives:

  $$\bar{y} = y - y_{\text{pred}}, \quad \bar{z} = \bar{y}\sigma'(z), \quad \bar{w} = \bar{z}x, \quad \bar{b} = \bar{z} \tag{25}$$

- You should use this notation in the following questions.

Suppose that $T = 2$, and the initial values of $h^{(0)}$ and $c^{(0)}$ are known.

(a) Derive the Backprop Through Time equations for the activations and the gates for $t = 1, 2$.

$$\overline{h^{(t)}}, \quad \overline{c^{(t)}}, \quad \overline{o^{(t)}}, \quad \overline{g^{(t)}}, \quad \overline{i^{(t)}}, \quad \overline{f^{(t)}}. \tag{26}$$

解: 由于本题是单变量 (univariate) 版本 LSTM, 即全部变量为实数, 故将 $\odot$ 省略.

由 $T = 2$ 可知损失函数 $\mathcal{L}$ 为

$$\mathcal{L} = \frac{1}{2}\left(y - h^{(2)}\right)^2 \tag{27}$$

所以, 损失函数 $\mathcal{L}$ 对 $h^{(2)}$ 的偏导数为

$$\overline{h^{(2)}} = \frac{\partial \mathcal{L}}{\partial h^{(2)}} = h^{(2)} - y \tag{28}$$

损失函数 $\mathcal{L}$ 对 $c^{(2)}$ 的偏导数为

$$\overline{c^{(2)}} = \frac{\partial \mathcal{L}}{\partial c^{(2)}} = \frac{\partial \mathcal{L}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial c^{(2)}} = \overline{h^{(2)}} o^{(2)} \left[1 - \tanh^2\left(c^{(2)}\right)\right] \tag{29}$$

损失函数 $\mathcal{L}$ 对 $o^{(2)}$ 的偏导数为

$$\overline{o^{(2)}} = \frac{\partial \mathcal{L}}{\partial o^{(2)}} = \frac{\partial \mathcal{L}}{\partial h^{(2)}} \frac{\partial h^{(2)}}{\partial o^{(2)}} = \overline{h^{(2)}} \tanh\left(c^{(2)}\right) \tag{30}$$

损失函数 $\mathcal{L}$ 对 $g^{(2)}$ 的偏导数为

$$\overline{g^{(2)}} = \frac{\partial \mathcal{L}}{\partial g^{(2)}} = \frac{\partial \mathcal{L}}{\partial c^{(2)}} \frac{\partial c^{(2)}}{\partial g^{(2)}} = \overline{c^{(2)}} i^{(2)} \tag{31}$$

损失函数 $\mathcal{L}$ 对 $i^{(2)}$ 的偏导数为

$$\overline{i^{(2)}} = \frac{\partial \mathcal{L}}{\partial i^{(2)}} = \frac{\partial \mathcal{L}}{\partial c^{(2)}} \frac{\partial c^{(2)}}{\partial i^{(2)}} = \overline{c^{(2)}} g^{(2)} \tag{32}$$

损失函数 $\mathcal{L}$ 对 $f^{(2)}$ 的偏导数为

$$\overline{f^{(2)}} = \frac{\partial \mathcal{L}}{\partial f^{(2)}} = \frac{\partial \mathcal{L}}{\partial c^{(2)}} \frac{\partial c^{(2)}}{\partial f^{(2)}} = \overline{c^{(2)}} c^{(1)} \tag{33}$$

损失函数 $\mathcal{L}$ 对 $h^{(1)}$ 的偏导数为

$$\begin{aligned}
\overline{h^{(1)}} &= \frac{\partial \mathcal{L}}{\partial h^{(1)}} \\
&= \frac{\partial \mathcal{L}}{\partial i^{(2)}} \frac{\partial i^{(2)}}{\partial h^{(1)}} + \frac{\partial \mathcal{L}}{\partial f^{(2)}} \frac{\partial f^{(2)}}{\partial h^{(1)}} + \frac{\partial \mathcal{L}}{\partial o^{(2)}} \frac{\partial o^{(2)}}{\partial h^{(1)}} + \frac{\partial \mathcal{L}}{\partial g^{(2)}} \frac{\partial g^{(2)}}{\partial h^{(1)}} \\
&= \overline{i^{(2)}} i^{(2)}(1 - i^{(2)}) w_{ih} + \overline{f^{(2)}} f^{(2)}(1 - f^{(2)}) w_{fh} + \overline{o^{(2)}} o^{(2)}(1 - o^{(2)}) w_{oh} + \overline{g^{(2)}}[1 - (g^{(2)})^2] w_{gh}
\end{aligned} \tag{34}$$

损失函数 $\mathcal{L}$ 对 $c^{(1)}$ 的偏导数为

$$\begin{aligned}
\overline{c^{(1)}} &= \frac{\partial \mathcal{L}}{\partial c^{(1)}} \\
&= \frac{\partial \mathcal{L}}{\partial c^{(2)}} \frac{\partial c^{(2)}}{\partial c^{(1)}} + \frac{\partial \mathcal{L}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial c^{(1)}} \\
&= \overline{c^{(2)}} f^{(2)} + \overline{h^{(1)}} o^{(1)} \left[1 - \tanh^2\left(c^{(1)}\right)\right]
\end{aligned} \tag{35}$$

损失函数 $\mathcal{L}$ 对 $o^{(1)}$ 的偏导数为

$$\overline{o^{(1)}} = \frac{\partial \mathcal{L}}{\partial o^{(1)}} = \frac{\partial \mathcal{L}}{\partial h^{(1)}} \frac{\partial h^{(1)}}{\partial o^{(1)}} = \overline{h^{(1)}} \tanh\left(c^{(1)}\right) \tag{36}$$

损失函数 $\mathcal{L}$ 对 $g^{(1)}$ 的偏导数为

$$\overline{g^{(1)}} = \frac{\partial \mathcal{L}}{\partial g^{(1)}} = \frac{\partial \mathcal{L}}{\partial c^{(1)}} \frac{\partial c^{(1)}}{\partial g^{(1)}} = \overline{c^{(1)}} i^{(1)} \tag{37}$$

损失函数 $\mathcal{L}$ 对 $i^{(1)}$ 的偏导数为

$$\overline{i^{(1)}} = \frac{\partial \mathcal{L}}{\partial i^{(1)}} = \frac{\partial \mathcal{L}}{\partial c^{(1)}} \frac{\partial c^{(1)}}{\partial i^{(1)}} = \overline{c^{(1)}} g^{(1)} \tag{38}$$

损失函数 $\mathcal{L}$ 对 $f^{(1)}$ 的偏导数为

$$\overline{f^{(1)}} = \frac{\partial \mathcal{L}}{\partial f^{(1)}} = \frac{\partial \mathcal{L}}{\partial c^{(1)}} \frac{\partial c^{(1)}}{\partial f^{(1)}} = \overline{c^{(1)}} c^{(0)} \tag{39}$$

(b) Derive the Backprop Through Time equation for the weight w_{ix}. (The other weight matrices are basically the same, so we won't make you write those out.)

解: 损失函数 $\mathcal{L}$ 对 w_{ix} 的偏导数为

$$\begin{aligned} \overline{w_{ix}} &= \frac{\partial \mathcal{L}}{\partial w_{ix}} \\ &= \frac{\partial \mathcal{L}}{\partial i^{(2)}} \frac{\partial i^{(2)}}{\partial w_{ix}} + \frac{\partial \mathcal{L}}{\partial i^{(1)}} \frac{\partial i^{(1)}}{\partial w_{ix}} \\ &= \overline{i^{(2)}} i^{(2)} (1 - i^{(2)}) x^{(2)} + \overline{i^{(1)}} i^{(1)} (1 - i^{(1)}) x^{(1)} \end{aligned} \tag{40}$$

(c) (*Optional*) Based on your answers above, can you explain why the gradient doesn't explode if the values of the forget gates ($f^{(t)}$) are very close to 1 and the values of the input and output gates ($i^{(t)}$ and $o^{(t)}$) are very close to 0? (Your answer may involve both $\overline{h^{(t)}}$ and $\overline{c^{(t)}}$.)

解: 由 $f^{(t)} \approx 1$, $o^{(t)} \approx 0$, $\forall\, t = 1, 2, \ldots, T$ 可知

$$\overline{c^{(t)}} = \overline{c^{(t+1)}} f^{(t+1)} + \overline{h^{(t)}} o^{(t)} \left[1 - \tanh^2\left(c^{(t)}\right)\right] \approx \overline{c^{(t+1)}}, \quad \forall\, t = 1, 2, \ldots, T-1 \tag{41}$$

所以

$$\overline{c^{(t)}} \approx \overline{c^{(T)}}, \quad \forall\, t = 1, 2, \ldots, T-1 \tag{42}$$

损失函数 $\mathcal{L}$ 对 w_{ix} 的偏导数为

$$\begin{aligned} \overline{w_{ix}} &= \sum_{t=1}^{T} \overline{i^{(t)}} i^{(t)} (1 - i^{(t)}) x^{(t)} \\ &= \sum_{t=1}^{T} \overline{c^{(t)}} g^{(t)} i^{(t)} (1 - i^{(t)}) x^{(t)} \\ &\approx \sum_{t=1}^{T} \overline{c^{(T)}} g^{(t)} i^{(t)} (1 - i^{(t)}) x^{(t)} \\ &= \sum_{t=1}^{T} \overline{h^{(T)}} \left[1 - \tanh^2\left(c^{(T)}\right)\right] g^{(t)} i^{(t)} (1 - i^{(t)}) x^{(t)} \\ &= \sum_{t=1}^{T} (h^{(T)} - y) \left[1 - \tanh^2\left(c^{(T)}\right)\right] g^{(t)} i^{(t)} (1 - i^{(t)}) x^{(t)} \end{aligned} \tag{43}$$

注意到 $i^{(t)} \approx 0, \forall\, t = 1, 2, \ldots, T$, 对 $|\overline{w_{ix}}|$ 估计其上界得

$$\begin{aligned}
|\overline{w_{ix}}| &= |h^{(T)} - y| \left[1 - \tanh^2\left(c^{(T)}\right)\right] \left|\sum_{t=1}^{T} g^{(t)} i^{(t)} (1 - i^{(t)}) x^{(t)}\right| \\
&\leqslant |h^{(T)} - y| \left[1 - \tanh^2\left(c^{(T)}\right)\right] \sum_{t=1}^{T} g^{(t)} i^{(t)} (1 - i^{(t)}) |x^{(t)}| \\
&\leqslant (1 + |y|) \sum_{t=1}^{T} i^{(t)} (1 - i^{(t)}) \max_{1 \leqslant t \leqslant T} |x^{(t)}| \\
&= (1 + |y|) \max_{1 \leqslant t \leqslant T} |x^{(t)}| \sum_{t=1}^{T} i^{(t)} (1 - i^{(t)}) \\
&\approx 0
\end{aligned} \tag{44}$$

因此梯度 $\overline{w_{ix}}$ 并不会爆炸.

Attention

3. Recall that attention can be viewed as an operation on a query $\boldsymbol{q} \in \mathbb{R}^d$, a set of value vectors $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$, $\boldsymbol{v}_i \in \mathbb{R}^d$, and a set of key vectors $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$, $\boldsymbol{k}_i \in \mathbb{R}^d$, specified as follows:

$$\boldsymbol{c} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \tag{45}$$

$$\alpha_i = \frac{\exp(\boldsymbol{k}_i^\top \boldsymbol{q})}{\sum_{j=1}^{n} \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} \tag{46}$$

where α_i are frequently called the "attention weights", and the output $\boldsymbol{c} \in \mathbb{R}^d$ is a correspondingly weighted average over the value vectors.

(a) **Copying via attention:** We'll first show that it's particularly simple for attention to "copy" a value vector to the output $\boldsymbol{c} \in \mathbb{R}^d$. Describe (in one sentence) what properties of the inputs to the attention operation would result in the output $\boldsymbol{c}$ being *approximately* equal to $\boldsymbol{v}_j$ for some $j \in \{1, 2, \ldots, n\}$. Specifically, what must be true about the query $\boldsymbol{q}$, the values $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ and/or the keys $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$?

Hint: When does softmax result approach a one-hot vector?

解: 当 $\boldsymbol{k}_j^\top \boldsymbol{q} \gg \boldsymbol{k}_i^\top \boldsymbol{q}, \forall\, i \neq j$ 时, $\boldsymbol{k}_j$ 对应的注意力权重为

$$\alpha_j = \frac{\exp(\boldsymbol{k}_j^\top \boldsymbol{q})}{\sum_{j=1}^{n} \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} \approx 1 \tag{47}$$

则有

$$\boldsymbol{c} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \approx \boldsymbol{v}_j \tag{48}$$

(b) **An average of two value vectors via attention:** Consider a set of key vectors $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$ where all key vectors are perpendicular, that is $\boldsymbol{k}_i \perp \boldsymbol{k}_j$ for all $i \neq j$. Let $\|\boldsymbol{k}_i\| = 1$ for all i. Let $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ be a set of arbitrary value vectors. Let $\boldsymbol{v}_a, \boldsymbol{v}_b \in \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n\}$ be two of the value vectors. Design a query vector $\boldsymbol{q}$ such that the output $\boldsymbol{c}$ is *approximately* equal to the average of $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$, that is, $\frac{1}{2}(\boldsymbol{v}_a + \boldsymbol{v}_b)$. Note that you can reference the corresponding key vector of $\boldsymbol{v}_a$ and $\boldsymbol{v}_b$ as $\boldsymbol{k}_a$ and $\boldsymbol{k}_b$.

Hint: While the softmax function will never exactly average the two vectors, you can get close by using a large scalar multiple in the expression.

解: 令 $\boldsymbol{q} = M(\boldsymbol{k}_a + \boldsymbol{k}_b)$, 其中 $M \in \mathbb{R}$ 是一个大数, 则 $\boldsymbol{k}_a$ 注意力权重为

$$\alpha_a = \frac{\exp(\boldsymbol{k}_a^\top \boldsymbol{q})}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} = \frac{\exp(M)}{2\exp(M) + n - 2} \approx \frac{1}{2} \tag{49}$$

同理可得, $\boldsymbol{k}_b$ 注意力权重为

$$\alpha_b = \frac{\exp(\boldsymbol{k}_b^\top \boldsymbol{q})}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} = \frac{\exp(M)}{2\exp(M) + n - 2} \approx \frac{1}{2} \tag{50}$$

所以

$$\boldsymbol{c} = \sum_{i=1}^n \alpha_i \boldsymbol{v}_i \approx \frac{1}{2}(\boldsymbol{v}_a + \boldsymbol{v}_b) \tag{51}$$

(c) **Drawbacks of single-headed attention:** In the previous part, we saw how it was possible for a single-headed attention to focus equally on two values. The same concept could easily be extended to any subset of values. In this question we'll see why it's not a practical solution. Consider a set of key vectors $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$ that are now randomly sampled, $\boldsymbol{k}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, where the means $\boldsymbol{\mu}_i$ are known to you, but the covariances $\boldsymbol{\Sigma}_i$ are unknown. Further, assume that the means $\boldsymbol{\mu}_i$ are all perpendicular; $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j = 0$ if $i \neq j$, and unit norm, $\|\boldsymbol{\mu}_i\| = 1$.

i. Assume that the covariance matrices are $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$, for vanishingly small α. Design a query vector $\boldsymbol{q}$ in terms of the $\boldsymbol{\mu}_i$ such that as before, $\boldsymbol{c} \approx \frac{1}{2}(\boldsymbol{v}_a + \boldsymbol{v}_b)$, and provide a brief argument as to why it works.

解: 令 $\boldsymbol{q} = M(\boldsymbol{\mu}_a + \boldsymbol{\mu}_b)$, 其中 $M \in \mathbb{R}$ 是一个大数. 由于协方差矩阵 $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$ 为对角线元素非常小的对角矩阵, 则 $\boldsymbol{k}_i \approx \boldsymbol{\mu}_i$, 又 $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j = 0, \forall\, i \neq j$, 则有 $\boldsymbol{k}_i^\top \boldsymbol{q} \approx 0, \forall\, i \neq a, b$. 则 $\boldsymbol{k}_a$ 注意力权重为

$$\alpha_a = \frac{\exp(\boldsymbol{k}_a^\top \boldsymbol{q})}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} \approx \frac{\exp(M)}{2\exp(M) + n - 2} \approx \frac{1}{2} \tag{52}$$

同理可得, $\boldsymbol{k}_b$ 注意力权重为

$$\alpha_b = \frac{\exp(\boldsymbol{k}_b^\top \boldsymbol{q})}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} \approx \frac{\exp(M)}{2\exp(M) + n - 2} \approx \frac{1}{2} \tag{53}$$

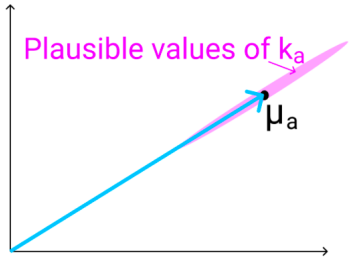


图 1: The vector $\boldsymbol{\mu}_a$ (shown here in 2D as an example), with the range of possible values of $\boldsymbol{k}_a$ shown in red. As mentioned previously, $\boldsymbol{k}_a$ points in roughly the same direction as $\boldsymbol{\mu}_a$, but may have larger or smaller magnitude.

所以

$$\boldsymbol{c} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \approx \frac{1}{2}(\boldsymbol{v}_a + \boldsymbol{v}_b) \tag{54}$$

ii. Though single-headed attention is resistant to small perturbations in the keys, some types of larger perturbations may pose a bigger issue. Specifically, in some cases, one key vector $\boldsymbol{k}_a$ may be larger or smaller in norm than the others, while still pointing in the same direction as $\boldsymbol{\mu}_a$. As an example, let us consider a covariance for item a as $\boldsymbol{\Sigma}_a = \alpha \boldsymbol{I} + \frac{1}{2}\boldsymbol{\mu}_a \boldsymbol{\mu}_a^\top$ for vanishingly small α (as shown in Figure 1). Further, let $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$ for all $i \neq a$. When you sample $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$ multiple times, and use the $\boldsymbol{q}$ vector that you designed in part i., what qualitatively do you expect the vector $\boldsymbol{c}$ will look like for different samples?

解: 定性来说, 不同采样得到的 $\boldsymbol{c}$ 有很大的差别. 由 $\boldsymbol{\Sigma}_a = \alpha \boldsymbol{I} + \frac{1}{2}\boldsymbol{\mu}_a \boldsymbol{\mu}_a^\top$ 及图 1 可知 $\boldsymbol{k}_a$ 的方向近似为 $\boldsymbol{\mu}_a$ 的方向, $\boldsymbol{k}_a$ 的长度大多数情况位于以下区间

$$\|\boldsymbol{k}_a\| \in [1-\beta,\ 1+\beta], \quad \beta \in (0,1) \tag{55}$$

由 i. 可知 $\boldsymbol{q} = M(\boldsymbol{\mu}_a + \boldsymbol{\mu}_b)$, 则 $\boldsymbol{k}_a$ 注意力权重为

$$\begin{aligned} \alpha_a &= \frac{\exp(\boldsymbol{k}_a^\top \boldsymbol{q})}{\sum_{j=1}^{n} \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} \\ &\in \left[\frac{\exp((1-\beta)M)}{\exp((1-\beta)M) + \exp(M) + n - 2},\ \frac{\exp((1+\beta)M)}{\exp((1+\beta)M) + \exp(M) + n - 2} \right] \\ &\approx [0, 1] \end{aligned} \tag{56}$$

$\boldsymbol{k}_b$ 注意力权重为

$$\begin{aligned} \alpha_b &= \frac{\exp(\boldsymbol{k}_b^\top \boldsymbol{q})}{\sum_{j=1}^{n} \exp(\boldsymbol{k}_j^\top \boldsymbol{q})} \\ &\in \left[\frac{\exp(M)}{\exp((1+\beta)M) + \exp(M) + n - 2},\ \frac{\exp(M)}{\exp((1-\beta)M) + \exp(M) + n - 2} \right] \\ &\approx [0, 1] \end{aligned} \tag{57}$$

则由

$$\boldsymbol{c} \approx \alpha_a \boldsymbol{v}_a + \alpha_b \boldsymbol{v}_b \tag{58}$$

可知 $\boldsymbol{c}$ 与 $\boldsymbol{k}_a$ 的长度大小有非常强的关联, 当 $\boldsymbol{k}_a$ 的长度发生变化时, $\boldsymbol{c}$ 也会发生很大的变化.

(d) **Benefits of multi-headed attention:** Now we'll see some power of multi-headed attention. We'll consider a simple version of multi-headed attention which is identical to single-headed self-attention as we've presented it in this homework, except two query vectors ($\boldsymbol{q}_1$ and $\boldsymbol{q}_2$) are defined, which leads to a pair of vectors ($\boldsymbol{c}_1$ and $\boldsymbol{c}_2$), each the output of single-headed attention given its respective query vector. The final output of the multi-headed attention is their average, $\frac{1}{2}(\boldsymbol{c}_1 + \boldsymbol{c}_2)$. As in question (c), consider a set of key vectors $\{\boldsymbol{k}_1, \boldsymbol{k}_2, \ldots, \boldsymbol{k}_n\}$ that are randomly sampled $\boldsymbol{k}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, where the means $\boldsymbol{\mu}_i$ are known to you, but the covariances $\boldsymbol{\Sigma}_i$ are unknown. Also as before, assume that the means $\boldsymbol{\mu}_i$ are mutually orthogonal; $\boldsymbol{\mu}_i^\top \boldsymbol{\mu}_j = 0$ if $i \neq j$, and unit norm, $\|\boldsymbol{\mu}_i\| = 1$.

i. Assume that the covariance matrices are $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$, for vanishingly small α. Design query vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ in terms of the $\boldsymbol{\mu}_i$ such that $\boldsymbol{c}$ is *approximately* equal to $\frac{1}{2}(\boldsymbol{v}_a + \boldsymbol{v}_b)$.

Hint: For the convenience of further analysis, you'd better recall the copy operation in question (a).

解: 令 $\boldsymbol{q}_1 = M\boldsymbol{\mu}_a$, $\boldsymbol{q}_2 = M\boldsymbol{\mu}_b$, 由协方差矩阵 $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$ 可知 $\boldsymbol{k}_i \approx \boldsymbol{\mu}_i$, 则 $\boldsymbol{k}_a$ 相对 $\boldsymbol{q}_1$ 注意力权重为

$$\alpha_a = \frac{\exp(\boldsymbol{k}_a^\top \boldsymbol{q}_1)}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q}_1)} \approx 1 \tag{59}$$

则对 $\boldsymbol{q}_1$ 输出

$$\boldsymbol{c}_1 = \sum_{i=1}^n \alpha_i \boldsymbol{v}_i \approx \boldsymbol{v}_a \tag{60}$$

$\boldsymbol{k}_b$ 相对 $\boldsymbol{q}_2$ 注意力权重为

$$\alpha_b = \frac{\exp(\boldsymbol{k}_b^\top \boldsymbol{q}_2)}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q}_2)} \approx 1 \tag{61}$$

则对 $\boldsymbol{q}_2$ 输出

$$\boldsymbol{c}_2 = \sum_{i=1}^n \alpha_i \boldsymbol{v}_i \approx \boldsymbol{v}_b \tag{62}$$

所以

$$\boldsymbol{c} = \frac{1}{2}(\boldsymbol{c}_1 + \boldsymbol{c}_2) \approx \frac{1}{2}(\boldsymbol{v}_a + \boldsymbol{v}_b) \tag{63}$$

ii. Assume that the covariance matrices are $\boldsymbol{\Sigma}_a = \alpha \boldsymbol{I} + \frac{1}{2}\left(\boldsymbol{\mu}_a \boldsymbol{\mu}_a^\top\right)$ for vanishingly small α, and $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$ for all $i \neq a$. Take the query vectors $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ that you designed in part i. What, qualitatively, do you expect the vector $\boldsymbol{c}$ will look like for different samples? Please briefly explain why. You can ignore cases in which $\boldsymbol{k}_a^\top \boldsymbol{q}_i < 0$.

解: 定性来说, 不同采样得到的 $\boldsymbol{c}$ 几乎相同. 由 $\boldsymbol{\Sigma}_a = \alpha \boldsymbol{I} + \frac{1}{2}\boldsymbol{\mu}_a \boldsymbol{\mu}_a^\top$ 及图 1 可知 $\boldsymbol{k}_a$ 的方向近似为 $\boldsymbol{\mu}_a$ 的方向, $\boldsymbol{k}_a$ 的长度多数情况位于以下区间

$$\|\boldsymbol{k}_a\| \in [1-\beta,\ 1+\beta], \quad \beta \in (0,1) \tag{64}$$

忽略 $\boldsymbol{k}_a^\top \boldsymbol{q}_1 < 0$ 的情况, 由 $\boldsymbol{q}_1 = M\boldsymbol{\mu}_a$ 可知 $\boldsymbol{k}_a$ 相对 $\boldsymbol{q}_1$ 注意力权重为

$$\alpha_a = \frac{\exp(\boldsymbol{k}_a^\top \boldsymbol{q}_1)}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q}_1)} \approx \frac{\exp(\|\boldsymbol{k}_a\| M)}{\exp(\|\boldsymbol{k}_a\| M) + n - 1} \approx 1 \tag{65}$$

$\boldsymbol{k}_b$ 相对 $\boldsymbol{q}_2$ 注意力权重没有变化, 仍然为

$$\alpha_b = \frac{\exp(\boldsymbol{k}_b^\top \boldsymbol{q}_2)}{\sum_{j=1}^n \exp(\boldsymbol{k}_j^\top \boldsymbol{q}_2)} \approx 1 \tag{66}$$

注意到 $\alpha_a \approx 1$ 与 $\boldsymbol{k}_a$ 的长度几乎无关, 则不同的采样仍然都会得到

$$\boldsymbol{c} \approx \frac{1}{2}(\boldsymbol{v}_a + \boldsymbol{v}_b) \tag{67}$$

(e) **Visualization:** Use the computer to sample 2-dimension key vectors multiple times and visualize the distributions of the output $\boldsymbol{c}$ of query $\boldsymbol{q}$ designed in (c) and (d) under different covariance matrices conditions. More specifically, we set $n = d = 2$, $\boldsymbol{v}_1 = \boldsymbol{\mu}_1 = (0,1)^\top$ and $\boldsymbol{v}_2 = \boldsymbol{\mu}_2 = (1,0)^\top$. You may have to carefully choose α to produce the desired phenomenon, we recommend $\alpha = 1 \times 10^{-10}$. Does this small experiment verify your findings in the above questions?

解: 这个小的数值实验可以验证上述题目的分析结果. 对键向量 $\{\boldsymbol{k}_1, \boldsymbol{k}_2\}$ 随机采样 10 次, 得到 (c) 题的可视化结果如图 2 和 3 所示.

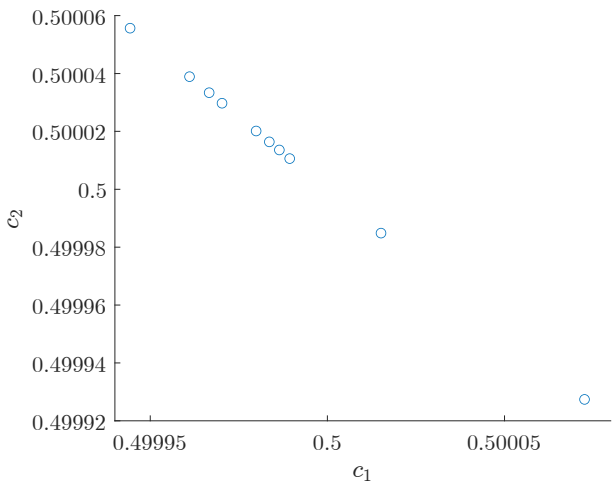


图 2: (c) i. 注意力机制输出结果 $\boldsymbol{c}$

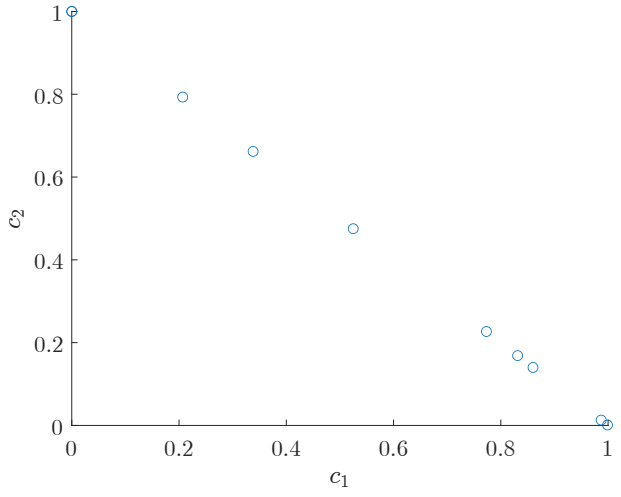


图 3: (c) ii. 注意力机制输出结果 $\boldsymbol{c}$

由图 2 可知, 对单向 (single-headed) 注意力机制而言, 当 $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$ 时, 输出的结果都是 $\boldsymbol{c} \approx \frac{1}{2}(\boldsymbol{v}_1 + \boldsymbol{v}_2)$; 由图 3 可知, 当 $\boldsymbol{\Sigma}_1 = \alpha \boldsymbol{I} + \frac{1}{2}\boldsymbol{\mu}_1 \boldsymbol{\mu}_1^\top$ 时, 输出结果 $\boldsymbol{c}$ 是 $\boldsymbol{v}_1$ 与 $\boldsymbol{v}_2$ 的任意凸组合.

(d) 题的可视化结果如图 4 和 5 所示.

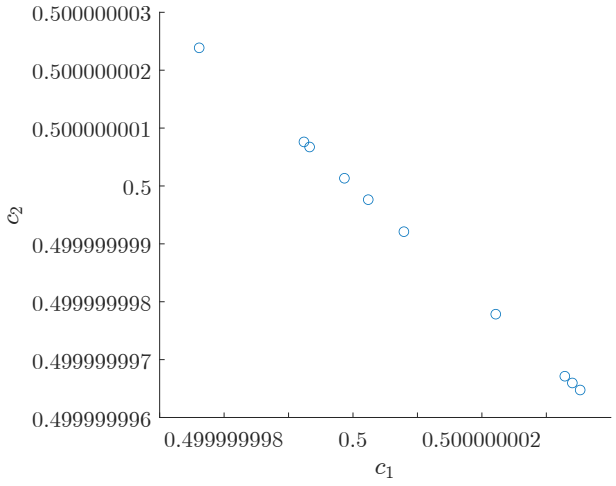


图 4: (d) i. 注意力机制输出结果 $\boldsymbol{c}$

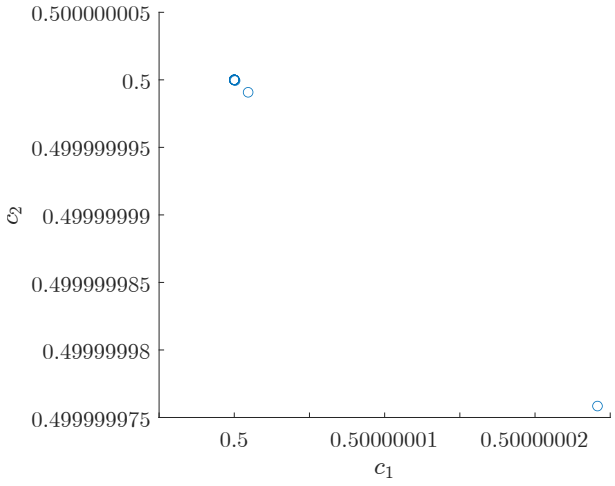


图 5: (d) ii. 注意力机制输出结果 $\boldsymbol{c}$

由图 4 可知, 对多向 (multi-headed) 注意力机制而言, 当 $\boldsymbol{\Sigma}_i = \alpha \boldsymbol{I}$ 时, 输出的结果都是 $\boldsymbol{c} \approx \frac{1}{2}(\boldsymbol{v}_1 + \boldsymbol{v}_2)$; 由图 5 可知, 当 $\boldsymbol{\Sigma}_1 = \alpha \boldsymbol{I} + \frac{1}{2}\boldsymbol{\mu}_1 \boldsymbol{\mu}_1^\top$ 时, 忽略 $\boldsymbol{k}_1^\top \boldsymbol{\mu}_1 < 0$ 的情况, 输出的结果也是 $\boldsymbol{c} \approx \frac{1}{2}(\boldsymbol{v}_1 + \boldsymbol{v}_2)$, 不受协方差矩阵 $\boldsymbol{\Sigma}_1$ 变化的影响, 即若 $\boldsymbol{k}_1$ 只是大小发生变化而方向几乎不变则对输出 $\boldsymbol{c}$ 几乎没有影响.

Programming: Natural Language Processing (NLP)

4. Please install PyTorch, Jupyter Notebook and run the NLP tutorial. If you want to use TensorFlow or other deep learning frameworks, please find corresponding language translation tutorial for that framework and run it. The tutorial is the third example for NLP From Scratch in Pytorch tutorials, where we write our own classes and functions to preprocess the data to do NLP modeling tasks. We hope after you complete this tutorial that you'll proceed to learn how torchtext can handle much of this preprocessing for you in the three tutorials immediately following this one. In this project we will be teaching a neural network to translate from French to English. The code is given as a jupyter notebook file.

All you need to do is to read, run and think. You need not write any code or any report in this programming.

参考文献

[1] Olah, Chris. Understanding LSTM Networks, 2015.
http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 10**

Dimensionality Reduction

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

PCA and Eigenvectors

1. Let $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n$ denote n vectors in $\mathbb{R}^D$, and we know the mean vector

$$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i = \boldsymbol{0} \in \mathbb{R}^D. \tag{1}$$

We project them into a lower dimensional space by performing a linear transformation

$$\boldsymbol{y}_i = \boldsymbol{W}^\top \boldsymbol{x}_i, \tag{2}$$

where $\boldsymbol{y}_i \in \mathbb{R}^d$, $\boldsymbol{W} \in \mathbb{R}^{D\times d}$, and $\boldsymbol{W}^\top\boldsymbol{W} = \boldsymbol{I} \in \mathbb{R}^{d\times d}$.

To simplify notations, we stack $\boldsymbol{x}_i$ column by column to make a data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{D\times n}$, and then perform the same operation on $\boldsymbol{y}_i$ to get $\boldsymbol{Y} \in \mathbb{R}^{d\times n}$. Then we can calculate the covariance matrix $\boldsymbol{\Sigma_X} = \boldsymbol{X}\boldsymbol{X}^\top$, and $\boldsymbol{\Sigma_Y} = \boldsymbol{Y}\boldsymbol{Y}^\top$. Please find the matrix $\boldsymbol{W}$ which maximizes the trace of $\boldsymbol{\Sigma_Y}$. This problem has a closed-form solution and thus numerical solutions will not be accepted.

解: 待求优化问题为

$$\begin{aligned} &\max_{\boldsymbol{W}} \ \mathrm{Tr}(\boldsymbol{\Sigma_Y}) \\ &\text{s.t.} \quad \boldsymbol{W}^\top\boldsymbol{W} = \boldsymbol{I} \end{aligned} \tag{3}$$

由 $\boldsymbol{Y} = \boldsymbol{W}^\top\boldsymbol{X}$ 可得

$$\mathrm{Tr}(\boldsymbol{\Sigma_Y}) = \mathrm{Tr}(\boldsymbol{Y}\boldsymbol{Y}^\top) = \mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{X}\boldsymbol{X}^\top\boldsymbol{W}) = \mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{\Sigma_X}\boldsymbol{W}) \tag{4}$$

所以上述优化问题可写为

$$\begin{aligned} &\max_{\boldsymbol{W}} \ \mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{\Sigma_X}\boldsymbol{W}) \\ &\text{s.t.} \quad \boldsymbol{W}^\top\boldsymbol{W} = \boldsymbol{I} \end{aligned} \tag{5}$$

记 $\boldsymbol{W}=[\boldsymbol{w}_1,\boldsymbol{w}_2,\ldots,\boldsymbol{w}_d]$, $\boldsymbol{w}_i\in\mathbb{R}^D$, 则 $\boldsymbol{W}^\top\boldsymbol{W}=\boldsymbol{I}$ 之约束可写为

$$\boldsymbol{w}_i^\top\boldsymbol{w}_j=\delta_{ij},\quad \forall\, i,j=1,2,\ldots,d \tag{6}$$

其中 δ_{ij} 是 Kronecker delta 函数.

引入 Lagrange 乘子 $\lambda_1,\lambda_2,\ldots,\lambda_d$, 并记 $\boldsymbol{\Lambda}=\mathrm{diag}(\lambda_1,\lambda_2,\ldots,\lambda_d)$, 可得 Lagrange 函数为

$$\begin{aligned} L(\boldsymbol{W},\boldsymbol{\Lambda}) &= \mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{W})-\sum_{i=1}^{d}\lambda_i(\boldsymbol{w}_i^\top\boldsymbol{w}_i-1)\\ &= \mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{W})-\mathrm{Tr}(\boldsymbol{\Lambda}(\boldsymbol{W}^\top\boldsymbol{W}-\boldsymbol{I})) \end{aligned} \tag{7}$$

注意到 $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ 为对称矩阵, 可知 Lagrange 函数对 $\boldsymbol{W}$ 的偏导数为

$$\frac{\partial L(\boldsymbol{W},\boldsymbol{\Lambda})}{\partial\boldsymbol{W}}=2\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{W}-2\boldsymbol{W}\boldsymbol{\Lambda} \tag{8}$$

令此偏导数为 $\boldsymbol{0}$ 可得

$$\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{W}=\boldsymbol{W}\boldsymbol{\Lambda} \tag{9}$$

所以, $\boldsymbol{w}_i$ 为矩阵 $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ 的特征向量, 且 λ_i 为对应的特征值.

令 Lagrange 函数对 $\boldsymbol{\Lambda}$ 的偏导数为 $\boldsymbol{0}$ 可得

$$\boldsymbol{W}^\top\boldsymbol{W}=\boldsymbol{I} \tag{10}$$

此时

$$\mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{\Sigma}_{\boldsymbol{X}}\boldsymbol{W})=\mathrm{Tr}(\boldsymbol{W}^\top\boldsymbol{W}\boldsymbol{\Lambda})=\mathrm{Tr}(\boldsymbol{I}\boldsymbol{\Lambda})=\mathrm{Tr}(\boldsymbol{\Lambda})=\sum_{i=1}^{d}\lambda_i \tag{11}$$

则最大化 $\mathrm{Tr}(\boldsymbol{\Sigma}_{\boldsymbol{Y}})=\mathrm{Tr}(\boldsymbol{\Lambda})$ 要求取 $\lambda_1,\lambda_2,\ldots,\lambda_d$ 为矩阵 $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ 前 d 个最大的特征值. 令 $\tilde{\boldsymbol{w}}_i$ 为 λ_i 对应的特征向量, 由矩阵 $\boldsymbol{\Sigma}_{\boldsymbol{X}}$ 为实对称矩阵可知 $\{\tilde{\boldsymbol{w}}_1,\tilde{\boldsymbol{w}}_2,\ldots,\tilde{\boldsymbol{w}}_d\}$ 互相正交, 取

$$\boldsymbol{w}_i=\frac{\tilde{\boldsymbol{w}}_i}{\|\tilde{\boldsymbol{w}}_i\|},\quad \forall\, i=1,2,\ldots,d \tag{12}$$

则 $\boldsymbol{W}=[\boldsymbol{w}_1,\boldsymbol{w}_2,\ldots,\boldsymbol{w}_d]$ 亦满足 $\boldsymbol{W}^\top\boldsymbol{W}=\boldsymbol{I}$, 所以此 $\boldsymbol{W}$ 可使得 $\mathrm{Tr}(\boldsymbol{\Sigma}_{\boldsymbol{Y}})$ 达到最大.

MDS and Strain

2. In MDS, we have the distance matrix $\boldsymbol{D}\in\mathbb{R}^{n\times n}$ for n data points, where $\boldsymbol{D}_{i,j}=(\boldsymbol{x}_i-\boldsymbol{x}_j)^\top(\boldsymbol{x}_i-\boldsymbol{x}_j)$. We first get the inner product matrix $\boldsymbol{B}$ by

$$\boldsymbol{B}=-\frac{1}{2}\boldsymbol{H}\boldsymbol{D}\boldsymbol{H}, \tag{13}$$

where $\boldsymbol{H}$ is defined as $\boldsymbol{H} \triangleq \boldsymbol{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, $\mathbf{1} = (1, 1, \ldots, 1)^\top \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{I} \in \mathbb{R}^{n\times n}$ is the identity matrix.

Suppose the desired number of dimensions for output is m. In the next step of MDS we should find the m largest eigenvalues values $\lambda_1, \lambda_2, \ldots, \lambda_m$ and corresponding eigenvectors $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m \in \mathbb{R}^n$ of matrix $\boldsymbol{B}$ and the final output of MDS should be $\boldsymbol{X} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m] \cdot \mathrm{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_m})$. Please prove that this procedure is equivalent to find $\boldsymbol{X}$ to minimize the strain, which is defined by

$$\mathrm{Strain}(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n) = \sqrt{\frac{\sum_{i,j}(\boldsymbol{B}_{i,j} - \boldsymbol{x}_i^\top \boldsymbol{x}_j)^2}{\sum_{i,j}\boldsymbol{B}_{i,j}}}. \tag{14}$$

解: 记 $\boldsymbol{U} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m]$, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$, 则 $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, MDS 得到

$$\boldsymbol{X} = [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_m] \cdot \mathrm{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_m}) = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}} \tag{15}$$

关于最小化 strain, 易知

$$\min_{\boldsymbol{X}\in\mathbb{R}^{n\times m}} \sqrt{\frac{\sum_{i,j}(\boldsymbol{B}_{i,j} - \boldsymbol{x}_i^\top \boldsymbol{x}_j)^2}{\sum_{i,j}\boldsymbol{B}_{i,j}}} \Longleftrightarrow \min_{\boldsymbol{X}\in\mathbb{R}^{n\times m}} \sum_{i=1}^{n}\sum_{j=1}^{n}(\boldsymbol{B}_{i,j} - \boldsymbol{x}_i^\top \boldsymbol{x}_j)^2 \tag{16}$$

由于

$$(\boldsymbol{B}_{i,j} - \boldsymbol{x}_i^\top \boldsymbol{x}_j)^2 \geqslant 0, \quad \forall\, i, j = 1, 2, \ldots, n \tag{17}$$

则 strain 最小值为 0, 且等号能够取到当且仅当

$$\boldsymbol{B}_{i,j} - \boldsymbol{x}_i^\top \boldsymbol{x}_j = 0, \quad \forall\, i, j = 1, 2, \ldots, n \tag{18}$$

写成矩阵形式即

$$\boldsymbol{B} = \boldsymbol{X}\boldsymbol{X}^\top \tag{19}$$

又 $\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$, 则有 $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Lambda}^{\frac{1}{2}}$, 与 MDS 得到的 $\boldsymbol{X}$ 相同, 即 MDS 由内积矩阵 $\boldsymbol{B}$ 求取 $\boldsymbol{X}$ 的步骤与最小化 strain 等价.

ISOMAP and LLE

ISOMAP, LLE 对流形的降维

3. 考虑如下的问题并实现 ISOMAP, LLE 等降维方法. 注意: 数据在产生过程中可不必严格保证形状, 大致符合要求即可, 不用在数据的产生上花费过多时间. 可以参考 scikit-learn 的官方文档, 实现类似的效果, 但是不可以直接使用已有的 LLE 和 ISOMAP 函数.

3.1. 在三维空间中产生 "Z" 形状的流形, 使用 ISOMAP 方法降维并作图, 给出数据的三维分布图和最佳参数下的降维效果图.

解: 生成 “Z” 形状的流形数据如图 1 所示, 使用 ISOMAP 方法降维, 选择 20 近邻计算距离, 得到降维后的结果如图 2 所示, ISOMAP 将三维数据降维到二维平面, 并且保持着数据点在二维流形上的位置关系.

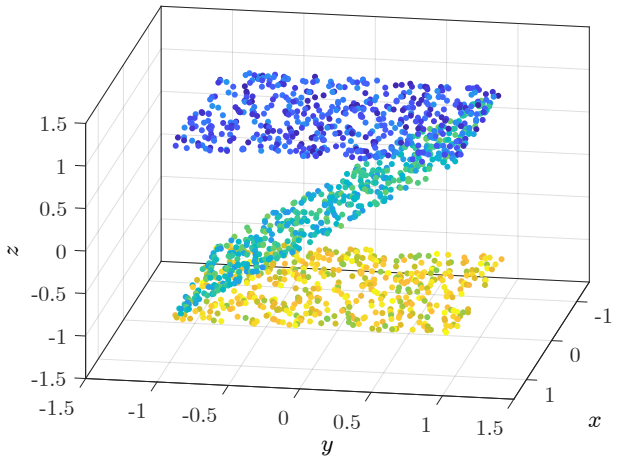


图 1: “Z” 形状流形数据分布图

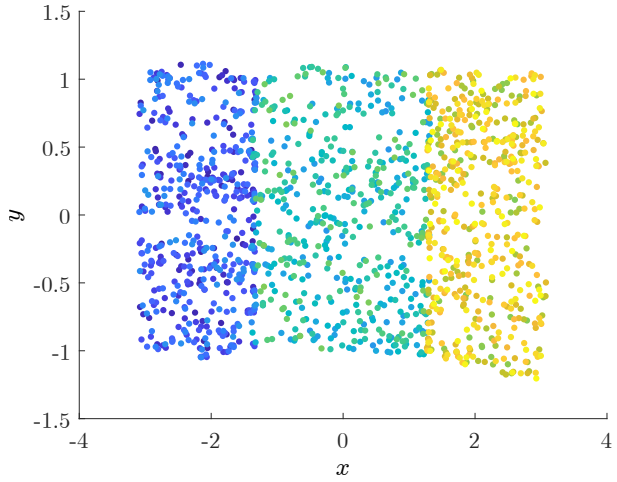


图 2: “Z” 形状流形数据 ISOMAP 降维结果

3.2. 在三维空间中产生 “W” 形状的流形, 使用 LLE 方法降维并作图, 给出数据的三维分布图和最佳参数下的降维效果图.

解: 生成 “W” 形状的流形数据如图 3 所示, 使用 LLE 方法降维, LLE 算法部分代码参考网站 [1], 选择 40 近邻计算距离, 得到降维后的结果如图 4 所示, LLE 将三维数据降维到二维平面, 并且保持了良好的可分性.

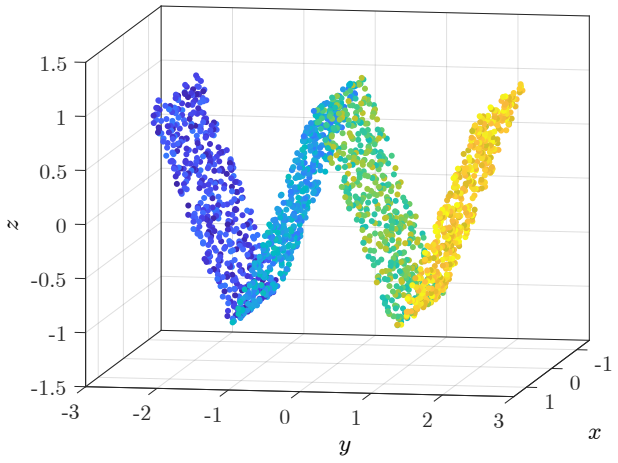


图 3: “W” 形状流形数据分布图

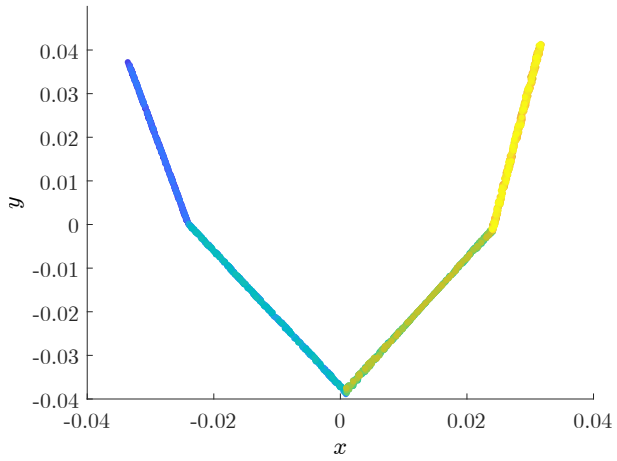


图 4: “W” 形状流形数据 40 近邻 LLE 降维结果

选择不同的近邻参数进行实验, 当近邻参数 k 较小时, 降维结果均为与图 4 类似的四条线段组成的折线, 而当近邻参数 k 约等于一个平面的点数时, 降维结果为四个平行四边形组成的平面图形. “W” 的四个面均由 500 个点构成, 若取 500 近邻, 则 LLE 降维结果如图 5 所示, 此时 LLE 将三维数据降维到二维平面, 并且保持着数据点在二维流形上的位置关系.

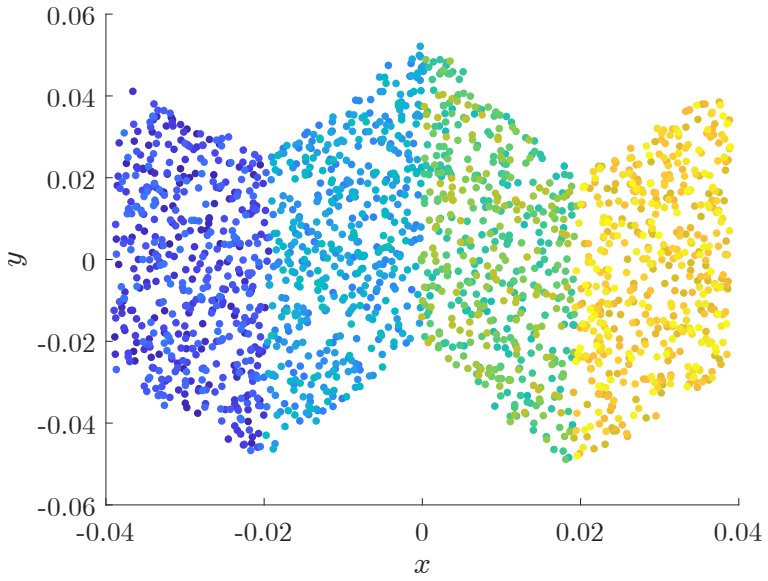


图 5: “W” 形状流形数据 500 近邻 LLE 降维结果

Further Reading

Whitening with PCA and ZCA

“*A whitening transformation or sphering transformation is a linear transformation that transforms a vector of random variables with a known covariance matrix into a set of new variables whose covariance is the identity matrix, meaning that they are uncorrelated and each have variance 1. The transformation is called ‘whitening’ because it changes the input vector into a white noise vector.*” [2]

Suppose we have n d-dimensional data points stored in $\boldsymbol{x} \in \mathbb{R}^{n\times d}$. The covariance matrix is $C(\boldsymbol{X}) = \frac{1}{n}\boldsymbol{X}^T\boldsymbol{X}$ and a whitening transformation is $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$ where $\boldsymbol{W} \in \mathbb{R}^{d\times d}$ is the whitening matrix and $\boldsymbol{Y}$ is the transformed data with $C(\boldsymbol{Y}) = \boldsymbol{I}$. Theoretically, whitening transformation is not unique because a rotated whitening matrix $\boldsymbol{W}_2 = \boldsymbol{R}\boldsymbol{W}_1$ ($\boldsymbol{R}$ is an orthogonal matrix) is also a whitening matrix.

Suppose the eigenvalue decomposition for $C(\boldsymbol{X})$ is given by $C(\boldsymbol{X}) = \boldsymbol{E}\boldsymbol{D}\boldsymbol{E}^\top$ with eigenvectors in columns of $\boldsymbol{E}$ and eigenvalues on the diagonal of $\boldsymbol{D}$. For principal component analysis (PCA), the whitening matrix is calculated by $\boldsymbol{W}_{\text{PCA}} = \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{E}^\top$. For zero-phase component analysis (ZCA), the whitening matrix is $\boldsymbol{W}_{\text{ZCA}} = \boldsymbol{E}\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{E}^\top$. Multiplication by an orthogonal matrix can be seen as rotation and multiplication by a diagonal matrix can be seen as scaling. We can see that ZCA rotates the transformed vectors of PCA back to the original data space with the orthogonal matrix $\boldsymbol{E}$.

In deep learning, we know that batch normalization (BN) is a powerful trick to accelerate and stabilize the training of deep models. BN simply performs standardization for input feature maps. However, it has been

shown that batch whitening (transform the input feature maps with a whitening transformation) further improves BN's optimization efficiency and generalization ability [3]. In batch whitening, the ZCA whitening is much better than the PCA whitening, read the work [3] for further reference of why this happens.

Optional: Construct a toy example, calculate the PCA results and the ZCA results and compare them to illustrate why ZCA is preferred.

Non-classical MDS

The classical derivation of MDS in the class assumed that the distance matrix is calculated by Euclidean distances of paired data points. However, in real applications, this matrix represents a set of dissimilarities which might not be Euclidean distances or not even distances at all. The MDS problem without the Euclidean assumption of distance matrix is called non-classical MDS. This is a generalization of the classical MDS and implemented as default MDS algorithm in Python library scikit-learn. Read the book [4] for solutions in this situation.

In addition, there is a more general form of MDS algorithm called non-metric MDS which aims to preserve the *rank-order* of the distances in the embedding space rather than their *values*. You can also find solutions for non-metric MDS in the same book [4] if you are interested.

参考文献

[1] Sam T. Roweis, lle.m - A simple matlab routine to perform LLE, Locally Linear Embedding (LLE) Code Page, https://cs.nyu.edu/~roweis/lle/code.html.

[2] Wikipedia contributors. "Whitening transformation." Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 15 Dec. 2020. Web. 27 Apr. 2021.

[3] Huang L, Yang D, Lang B, et al. Decorrelated batch normalization [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 791-800.

[4] Boyarski A., Bronstein A. Multidimensional Scaling. In: Ikeuchi K. (eds) Computer Vision. Springer, Cham, 2020.

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 11**

Decision Tree and GNN

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Decision Tree

Consider a data set comprising 400 data points from class C_1 and 400 data points from class C_2. Suppose that a tree model A splits these data points into (300, 100) at the first leaf node and (100, 300) at the second leaf node, where (n, m) denotes that n points are assigned to C_1 and m points are assigned to C_2. Similarly, suppose that a second tree model B splits them into (200, 400) and (200, 0).

Evaluate the misclassification rates for the two trees and show that they are equal. Similarly, evaluate the cross-entropy and Gini index for the two trees and show that they are both lower for tree B than for tree A.

解: 决策树模型 A 的错分率为

$$\begin{aligned} i_A &= P_L i_A(N_L) + P_R i_A(N_R) \\ &= \frac{1}{2} \times \frac{1}{4} + \frac{1}{2} \times \frac{1}{4} \\ &= \frac{1}{4} \end{aligned} \tag{1}$$

决策树模型 B 的错分率为

$$\begin{aligned} i_B &= P_L i_A(N_L) + P_R i_A(N_R) \\ &= \frac{3}{4} \times \frac{1}{3} + \frac{1}{4} \times 0 \\ &= \frac{1}{4} \end{aligned} \tag{2}$$

所以 $i_A = i_B$, 即决策树模型 A 与 B 的错分率相等.

决策树模型 A 的交叉熵为

$$\begin{aligned} \hat{i}_A &= P_L \hat{i}_A(N_L) + P_R \hat{i}_A(N_R) \\ &= \frac{1}{2} \times \left(-\frac{3}{4} \log_2 \frac{3}{4} - \frac{1}{4} \log_2 \frac{1}{4} \right) + \frac{1}{2} \times \left(-\frac{1}{4} \log_2 \frac{1}{4} - \frac{3}{4} \log_2 \frac{3}{4} \right) \\ &= 0.8113 \end{aligned} \tag{3}$$

决策树模型 B 的交叉熵为

$$\begin{aligned}\hat{i}_B &= P_L\hat{i}_B(N_L) + P_R\hat{i}_B(N_R)\\ &= \frac{3}{4}\times\left(-\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3}\right) + \frac{1}{4}\times(-1\log_2 1)\\ &= 0.6887\end{aligned} \tag{4}$$

因此, 交叉熵 $\hat{i}_B < \hat{i}_A$.

决策树模型 A 的 Gini index 为

$$\begin{aligned}\tilde{i}_A &= P_L\tilde{i}_A(N_L) + P_R\tilde{i}_A(N_R)\\ &= \frac{1}{2}\times\left(1-\left(\frac{3}{4}\right)^2-\left(\frac{1}{4}\right)^2\right) + \frac{1}{2}\times\left(1-\left(\frac{1}{4}\right)^2-\left(\frac{3}{4}\right)^2\right)\\ &= \frac{3}{8}\end{aligned} \tag{5}$$

决策树模型 B 的 Gini index 为

$$\begin{aligned}\tilde{i}_B &= P_L\tilde{i}_B(N_L) + P_R\tilde{i}_B(N_R)\\ &= \frac{3}{4}\times\left(1-\left(\frac{1}{3}\right)^2-\left(\frac{2}{3}\right)^2\right) + \frac{1}{4}\times(1-1^2)\\ &= \frac{1}{3}\end{aligned} \tag{6}$$

因此, Gini index $\tilde{i}_B < \tilde{i}_A$.

Graph Neural Network (GNN)

One of the simplest possible propagation rule in GCN (Graph Convolutional Networks) is

$$f(H^i, A) = \sigma(AH^iW^i), \tag{7}$$

where A is a representative description of the graph structure in the form of an adjacency matrix, H^i is the feature matrix for layer i (H^i is an $N \times F^i$ matrix, where N is the number of nodes in the graph, F^i is the feature dimension of one node, and $H^0 = X$ is the input data), W^i is the weight matrix for layer i and σ is a non-linear activation function such as the ReLU function. The weight matrix has dimensions $F^i \times F^{i+1}$; in other words the size of the second dimension of the weight matrix determines the dimension of features at the next layer. If you are familiar with convolutional neural networks, this operation is similar to a filtering operation since these weights are shared across nodes in the graph.

Simplifications: Let's examine the propagation rule at its most simple level. Let

- $i = 1$, s.t. f is a function of the input feature matrix,
- σ be the identity function, and
- choose the weights s.t. $AH^0W^0 = AXW^0 = AX$.

In other words, $f(X, A) = AX$. This propagation rule is perhaps a bit too simple, but we will add in the missing parts later. As a side note, AX is now equivalent to the input layer of a multi-layer perceptron.

We'll use the following graph in Figure 1:

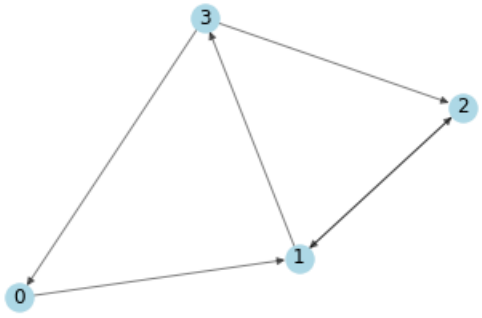


图 1: A simple directed graph

(a) Derive the adjacency matrix representation A and degree matrix D. Note that in this case a node n is a neighbor of node v if there exists an edge from v to n.

解: 邻接矩阵为

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix} \tag{8}$$

出度矩阵为

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix} \tag{9}$$

(b) We generate features for every node based on its index, i.e. $[i, -i]$ for node i. Apply the propagation rule on adjacency matrix A and input features X to derive the output matrix.

解: 由题意可知输入特征为

$$X = \begin{bmatrix} 0 & 0 \\ 1 & -1 \\ 2 & -2 \\ 3 & -3 \end{bmatrix} \tag{10}$$

所以, 输出矩阵为

$$f(X, A) = AX = \begin{bmatrix} 1 & -1 \\ 5 & -5 \\ 1 & -1 \\ 2 & -2 \end{bmatrix} \tag{11}$$

(c) We found that nodes with large degrees will have large values in their feature representation while nodes with small degrees will have small values. This can cause vanishing or exploding gradients. Therefore, the feature representations can be normalized by node degree by transforming the adjacency matrix A by multiplying it with the inverse degree matrix D. Thus our simplified propagation rule looks like this: $f(x, A) = D^{-1}AX$. We also found that the aggregated representation of a node does not include its own features. To address the problem, we add a self-loop to each node, by adding the identity matrix I to the adjacency matrix A before applying the propagation rule, that is, $\hat{A} = A + I$. $\hat{D}$ is the degree matrix of $\hat{A}$, i.e., the degree matrix of A with forced self-loops. Derive the output matrix after normalizing the feature representations and adding self-loops.

解: 加上自回路后, 邻接矩阵为

$$\hat{A} = A + I = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix} \tag{12}$$

其度矩阵为

$$\hat{D} = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix} \tag{13}$$

所以, 输出矩阵为

$$f(X, \hat{A}) = \hat{D}^{-1}\hat{A}X = \begin{bmatrix} \frac{1}{2} & -\frac{1}{2} \\ 2 & -2 \\ \frac{3}{2} & -\frac{3}{2} \\ \frac{5}{3} & -\frac{5}{3} \end{bmatrix} \tag{14}$$

(d) We add back the weights matrix W as follows:

$$W = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

Derive the output matrix.

解: 输出矩阵为

$$f(X, \hat{A}) = \hat{D}^{-1}\hat{A}XW = \begin{bmatrix} 1 & -1 \\ 4 & -4 \\ 3 & -3 \\ \frac{10}{3} & -\frac{10}{3} \end{bmatrix} \tag{15}$$

(e) Use the same W and add the ReLU activation function for σ. Derive the output matrix.

解: 输出矩阵为

$$f(X, \hat{A}) = \text{ReLU}\left(\hat{D}^{-1}\hat{A}XW\right) = \begin{bmatrix} 1 & 0 \\ 4 & 0 \\ 3 & 0 \\ \frac{10}{3} & 0 \end{bmatrix} \tag{16}$$

(f) We can apply a graph convolutional network on a real graph, Zachary's karate club. Zachary's karate club is a commonly used social network where nodes represent members of a karate club and the edges their mutual relations. While Zachary was studying the karate club, a conflict arose between the administrator and the instructor which resulted in the club splitting in two. The figure below shows the graph representation of the network and nodes are labeled according to which part of the club. The administrator and instructor are marked with 'A' and 'I', respectively.

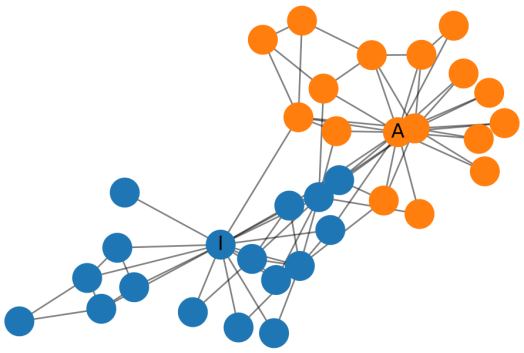


图 2: Zachary's Karate Club

You can use the following python codes to get the dataset:

```
from networkx import karate_club_graph

zkc = karate_club_graph()
```

Design a GNN to separate communities in Zachary's Karate Club. We here use just the identity matrix as input representation, that is, each node is represented as a one-hot encoded variable. Show the final output feature representations for the nodes of Figure 2.

Hint: Please try a GCN with two hidden layers just like (e), and initialize the weights randomly, then extract the feature representations and plot them. You will find even randomly initialized GCNs can separate communities in Zachary's Karate Club. Next, you can try your own GNN for better performance.

解: 首先绘制 Zachary's Karate Club 如图 3 所示, 其中标号 0 的节点对应 'I', 标号 33 的节点对应 'A'.

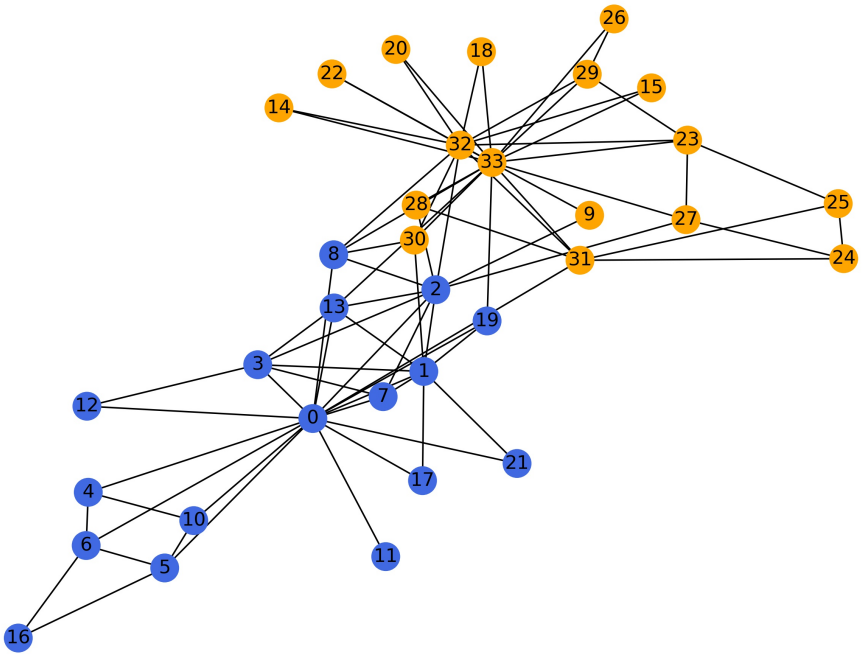


图 3: Zachary's Karate Club

与 (e) 类似, $H^0 = I$, 建立两层 GCN 网络,

$$\begin{aligned} H^1 &= \mathrm{ReLU}(\hat{D}^{-1}\hat{A}H^0W^0) \\ H^2 &= \mathrm{ReLU}(\hat{D}^{-1}\hat{A}H^1W^1) \end{aligned} \tag{17}$$

随机初始化权重矩阵 W^0, W^1 进行计算, 经过数次尝试, 得到 Zachary's Karate Club 的特征表示如图 4 所示, 由图可知即便是随机初始化的权重矩阵也可以将两个类别分开.

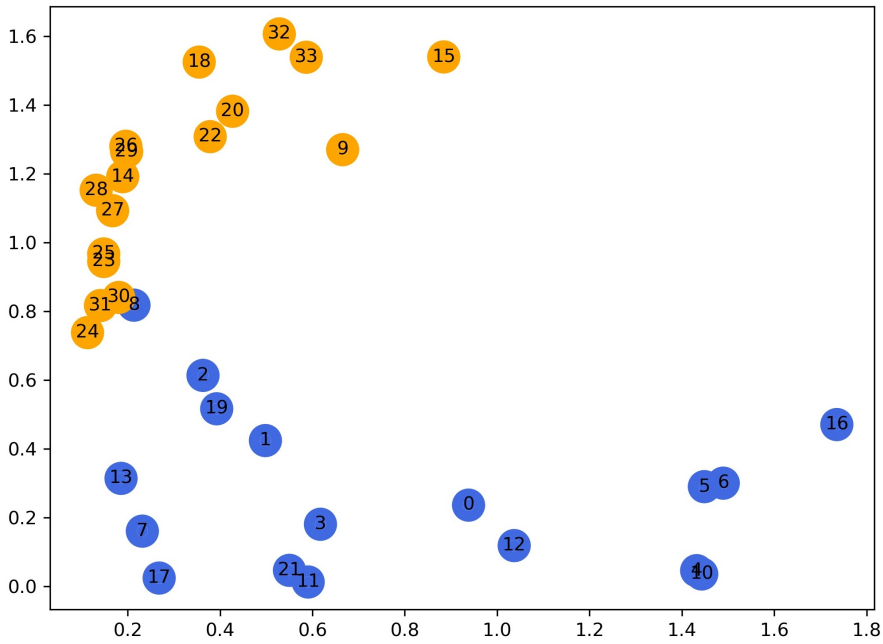


图 4: Feature Representations of Zachary's Karate Club

Programming: Decision Tree

实现决策树算法 C4.5，并且在 *Sogou Corpus* 数据集上测试它的效果（数据集详情见 readme）.

要求：

- 不能调用已有的机器学习包
- 将数据随机分为 3:1:1 的三份，分别为训练集，交叉验证集，测试集。请在训练集上训练，交叉验证集上选择超参数，用选出的最好模型在测试集上给出测试结果。因此，在报告中需要说明算法的超参数有哪些，在不同的超参数设置下训练集和交叉验证集的分类正确率，最好模型的超参数设置，以及最后的测试正确率。
- 请结构化代码，必须包含但不限于如下几个函数（请从代码中分离出来，有明确的这几个函数，函数参数可以有所变化）：
  - **main()**
    要求 main 函数在运行中，逐个测试不同的超参数，然后打印出每个超参数的设置，该设置下的训练、验证正确率（就是上面第二点中提到的要出现在报告中的结果）。
  - **GenerateTree(args)**
    生成树的总代码，*args* 为各种超参数，包括但不限于下面的 *thresh*，或者其他会影响树性能的超参数，自由发挥。

- **SplitNode(samplesUnderThisNode, thresh, ...)**

  对当前节点进行分支，*samplesUnderThisNode* 是当前节点下的样本，*thresh* 是停止分支的阈值，停止分支的条件请在实验报告中说明。

- **SelectFeature(samplesUnderThisNode, ...)**

  对当前节点下的样本，选择待分特征。

- **Impurity(samples)**

  给出样本 *samples* 的不纯度，请在实验报告中说明采用的不纯度度量。

- **Decision(GeneratedTree, XToBePredicted)**

  使用生成的树 *GenerateTree*，对样本 *XToBePredicted* 进行预测。

- **Prune(GeneratedTree, CorssValidationDataset, ...)**

  对生成好的树 *GeneratedTree*（已经过 stopped splitting) 进行剪枝：考虑所有相邻的叶子节点，如果将他们消去可以增加验证集上的正确率，则减去两叶子节点，将他们的共同祖先作为新的叶子节点。或者实现其他的剪枝方法，如有，请在实验报告中说明。

解: 数据集采用均匀分布随机分割为 3:1:1 的三份, 详见函数 `split()`, 分割结果见

`train_split.txt`, `validate_split.txt`, `test_split.txt`.

算法的超参数为分枝不纯度下降的阈值 (threshold), 设置为列表 [0.5, 0.4, 0.3, 0.2, 0.1, 0.09, 0.08, 0.07, 0.06, 0.05, 0.04, 0.03, 0.02, 0.01, 0.005, 0.003, 0.001], 停止分枝的条件为当前节点的数据纯净 (pure), 或者按任何特征进行分割都不能使得不纯度下降大于阈值, 其中不纯度采用熵不纯度 (entropy impurity) 进行度量. 算法采用的剪枝方法为 C4.5 基于规则的算法, 即把生成的决策树转化为等价的规则集合, 剪去任何一个可以使某个规则在验证集上分类正确率上升的前件 (precondition), 直到无法剪除, 最后将规则按分类正确率从高到低排序.

在不同的超参数设置下训练集, 交叉验证集和测试集的分类正确率如表 1 所示, 当阈值 threshold $\leqslant 0.01$ 时, 验证集正确率不再改变, 最佳参数 threshold $= 0.01$ 下最佳模型的测试正确率为 accuracy $= 0.77701$.

训练集正确率, 交叉验证集正确率和测试集正确率与阈值变化关系分别如图 5, 图 6 和图 7 所示, 可见使用 C4.5 的方法, 模型的分类正确率基本随着阈值的减小而逐渐升高, 最后逐渐收敛趋于不变.

表 1: 不同阈值训练集, 交叉验证集和测试集正确率

threshold	train accuracy	validate accuracy	test accuracy
0.5	0.11498	0.10131	0.10942
0.4	0.11498	0.10131	0.10942
0.3	0.11498	0.10131	0.10942
0.2	0.42276	0.41247	0.41828
0.1	0.78746	0.76844	0.73615
0.09	0.80523	0.79221	0.74965
0.08	0.80708	0.79428	0.74931
0.07	0.84251	0.83356	0.77701
0.06	0.84286	0.83425	0.77701
0.05	0.84402	0.83494	0.77909
0.04	0.84634	0.83494	0.77666
0.03	0.84599	0.8346	0.77666
0.02	0.84611	0.83494	0.77666
0.01	0.84541	0.83563	0.77701
0.005	0.84483	0.83563	0.77666
0.003	0.84483	0.83563	0.77666
0.001	0.84483	0.83563	0.77666

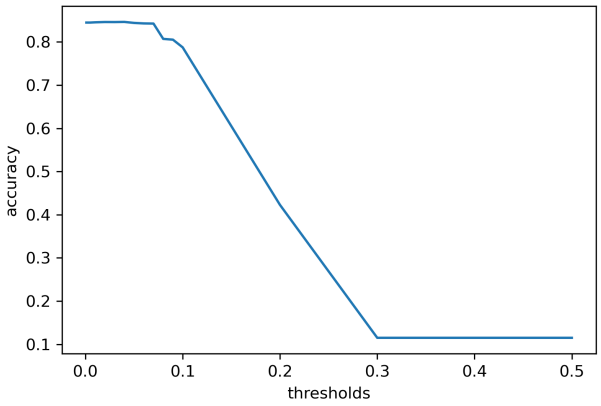


图 5: 训练集正确率与阈值变化关系

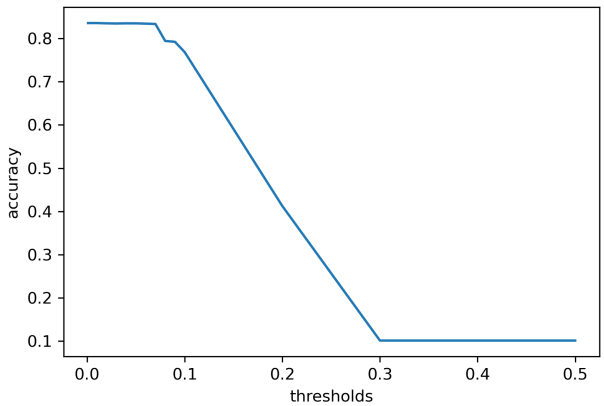


图 6: 交叉验证集正确率与阈值变化关系

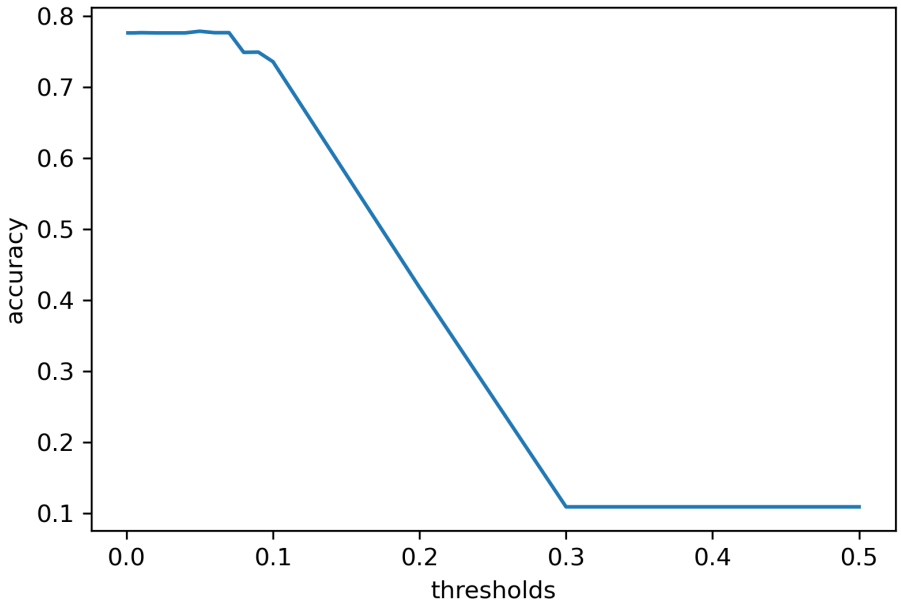


图 7: 测试集正确率与阈值变化关系

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 12**

Ensemble

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

Bagging

In practice, we have only a single data set, and *bagging* is a method to introduce variability between different models within the committee based on one data set. A committee can be viewed as a set of individual models on which we average our predictions.

The very first step is to use *bootstrap* data sets. After we have generated M bootstrap data sets, we then use each to train a separate predictive model y_m where $m = 1, 2, \ldots, M$. Then the prediction is given by

$$y_{\mathrm{COM}} = \frac{1}{M} \sum_{m=1}^{M} y_m(\boldsymbol{x}). \tag{1}$$

Suppose the true regression function that we are trying to predict is given by $h(\boldsymbol{x})$, so that the output of each of the models can be written as the true value plus an error in the form

$$y_m(\boldsymbol{x}) = h(\boldsymbol{x}) + \epsilon_m(\boldsymbol{x}). \tag{2}$$

The average sum-of-square error then takes the form

$$\mathbb{E}_{\boldsymbol{x}}[\{y_m(\boldsymbol{x}) - h(\boldsymbol{x})\}^2] = \mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})], \tag{3}$$

where $\mathbb{E}_{\boldsymbol{x}}$ denotes expectation with respect to the distribution of the input vector $\boldsymbol{x}$.

The average error made by the models acting individually is therefore

$$E_{\mathrm{AV}} = \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})]. \tag{4}$$

Similarly, the expected error from equation (1) is given by

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} y_m(\boldsymbol{x}) - h(\boldsymbol{x})\right\}^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2\right]. \end{aligned} \tag{5}$$

1.1 Assume that errors have zero mean and are uncorrelated

$$\begin{aligned} \mathbb{E}_{\boldsymbol{x}}[\epsilon_m(\boldsymbol{x})] = 0, \\ \mathbb{E}_{\boldsymbol{x}}[\epsilon_m(\boldsymbol{x})\epsilon_l(\boldsymbol{x})] = 0,\ \forall\, m \neq l. \end{aligned} \tag{6}$$

Please prove that

$$E_{\mathrm{COM}} = \frac{1}{M} E_{\mathrm{AV}}. \tag{7}$$

解: 由误差不相关可得

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2\right] \\ &= \frac{1}{M^2}\mathbb{E}_{\boldsymbol{x}}\left[\left\{\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2\right] \\ &= \frac{1}{M^2}\mathbb{E}_{\boldsymbol{x}}\left[\sum_{m=1}^{M}\sum_{l=1}^{M} \epsilon_m(\boldsymbol{x})\epsilon_l(\boldsymbol{x})\right] \\ &= \frac{1}{M^2}\mathbb{E}_{\boldsymbol{x}}\left[\sum_{m=1}^{M} \epsilon_m^2(\boldsymbol{x})\right] \\ &= \frac{1}{M^2}\sum_{m=1}^{M} \mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})] \\ &= \frac{1}{M} E_{\mathrm{AV}} \end{aligned} \tag{8}$$

1.2 In practice, the errors are typically highly correlated. Show that the following inequality holds without assumptions in 1.1.

$$E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}. \tag{9}$$

解: 函数 $f(x) = x^2$ 是凸函数, 则由 Jensen 不等式可得

$$\left\{\frac{1}{M}\sum_{m=1}^{M} \epsilon_m(\boldsymbol{x})\right\}^2 \leqslant \frac{1}{M}\sum_{m=1}^{M} \epsilon_m^2(\boldsymbol{x}) \tag{10}$$

所以,

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\boldsymbol{x})\right\}^2\right] \\ &\leqslant \mathbb{E}_{\boldsymbol{x}}\left[\frac{1}{M}\sum_{m=1}^{M}\epsilon_m^2(\boldsymbol{x})\right] \\ &= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\boldsymbol{x}}[\epsilon_m^2(\boldsymbol{x})] \\ &= E_{\mathrm{AV}} \end{aligned} \tag{11}$$

即 $E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}$.

1.3 In the previous problem, our error function is $f(y(\boldsymbol{x})-h(\boldsymbol{x})) = (y(\boldsymbol{x})-h(\boldsymbol{x}))^2$ (sum-of-square). By making use of *Jensen's inequality*, show that equation (9) holds for any error function $E(y(\boldsymbol{x})-h(\boldsymbol{x}))$ provided it is a convex function of $y(\boldsymbol{x})-h(\boldsymbol{x})$.

解: 函数 $E(\cdot)$ 是凸函数, 则由 Jensen 不等式可得

$$E\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\boldsymbol{x})\right\} \leqslant \frac{1}{M}\sum_{m=1}^{M}E(\epsilon_m(\boldsymbol{x})) \tag{12}$$

所以,

$$\begin{aligned} E_{\mathrm{COM}} &= \mathbb{E}_{\boldsymbol{x}}\left[E\left\{\frac{1}{M}\sum_{m=1}^{M}\epsilon_m(\boldsymbol{x})\right\}\right] \\ &\leqslant \mathbb{E}_{\boldsymbol{x}}\left[\frac{1}{M}\sum_{m=1}^{M}E(\epsilon_m(\boldsymbol{x}))\right] \\ &= \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}_{\boldsymbol{x}}[E(\epsilon_m(\boldsymbol{x}))] \\ &= E_{\mathrm{AV}} \end{aligned} \tag{13}$$

即 $E_{\mathrm{COM}} \leqslant E_{\mathrm{AV}}$.

1.4 Consider the case in which we allow unequal weighting of the individual models

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M}\alpha_m y_m(\boldsymbol{x}). \tag{14}$$

In order to make $y_{\mathrm{COM}}(\boldsymbol{x})$ sensible, we require that for all $y_m(\boldsymbol{x})$ they are bounded at each value of $\boldsymbol{x}$ like

$$y_{\min}(\boldsymbol{x}) \leqslant y_{\mathrm{COM}}(\boldsymbol{x}) \leqslant y_{\max}(\boldsymbol{x}). \tag{15}$$

Show that the necessary and sufficient condition for constraint (15) is

$$\alpha_m \geqslant 0, \quad \sum_{m=1}^{M} \alpha_m = 1. \tag{16}$$

解: 首先证明充分性, 令

$$\alpha_m \geqslant 0, \quad \sum_{m=1}^{M} \alpha_m = 1 \tag{17}$$

则有

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) \geqslant \sum_{m=1}^{M} \alpha_m y_{\min}(\boldsymbol{x}) = y_{\min}(\boldsymbol{x}) \tag{18}$$

且

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) \leqslant \sum_{m=1}^{M} \alpha_m y_{\max}(\boldsymbol{x}) = y_{\max}(\boldsymbol{x}) \tag{19}$$

下面证明必要性, 使用反证法. 假设

$$\sum_{m=1}^{M} \alpha_m \neq 1 \tag{20}$$

取

$$y_m(\boldsymbol{x}) = \tilde{y}(\boldsymbol{x}), \ \forall\, m = 1, 2, \ldots, M \tag{21}$$

则 $y_{\min}(\boldsymbol{x}) = y_{\max}(\boldsymbol{x}) = \tilde{y}(\boldsymbol{x})$, 而

$$y_{\mathrm{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m \tilde{y}(\boldsymbol{x}) \neq \tilde{y}(\boldsymbol{x}) \tag{22}$$

与 $y_{\min}(\boldsymbol{x}) \leqslant y_{\mathrm{COM}}(\boldsymbol{x}) \leqslant y_{\max}(\boldsymbol{x})$ 矛盾, 故

$$\sum_{m=1}^{M} \alpha_m = 1 \tag{23}$$

假设 $\exists\, l \in \{1, 2, \ldots, M\}$ 使得 $\alpha_l < 0$, 则

$$\sum_{m \neq l} \alpha_m = 1 - \alpha_l > 1 \tag{24}$$

取 $y_l(\boldsymbol{x}) = 0$,

$$y_m(\boldsymbol{x}) = \tilde{y}(\boldsymbol{x}), \ \forall\, m \neq l \tag{25}$$

令

$$P \triangleq \{\boldsymbol{x} : \tilde{y}(\boldsymbol{x}) > 0\}, \quad N \triangleq \{\boldsymbol{x} : \tilde{y}(\boldsymbol{x}) < 0\} \tag{26}$$

则

$$y_{\max}(\boldsymbol{x}) = \max(0, \tilde{y}(\boldsymbol{x})) = \begin{cases} \tilde{y}(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in P \\ 0, & \text{otherwise} \end{cases} \tag{27}$$

$$y_{\min}(\boldsymbol{x}) = \min(0, \tilde{y}(\boldsymbol{x})) = \begin{cases} \tilde{y}(\boldsymbol{x}), & \text{if } \boldsymbol{x} \in N \\ 0, & \text{otherwise} \end{cases} \tag{28}$$

对 $\forall\, \boldsymbol{x} \in P$, 由式 (24) 可得

$$y_{\text{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) = \sum_{m \neq l} \alpha_m \tilde{y}(\boldsymbol{x}) > \tilde{y}(\boldsymbol{x}) = y_{\max}(\boldsymbol{x}) \tag{29}$$

与 $y_{\text{COM}}(\boldsymbol{x}) \leqslant y_{\max}(\boldsymbol{x})$ 矛盾.

对 $\forall\, \boldsymbol{x} \in N$, 由式 (24) 可得

$$y_{\text{COM}}(\boldsymbol{x}) = \sum_{m=1}^{M} \alpha_m y_m(\boldsymbol{x}) = \sum_{m \neq l} \alpha_m \tilde{y}(\boldsymbol{x}) < \tilde{y}(\boldsymbol{x}) = y_{\min}(\boldsymbol{x}) \tag{30}$$

与 $y_{\text{COM}}(\boldsymbol{x}) \geqslant y_{\min}(\boldsymbol{x})$ 矛盾.

由以上矛盾可知

$$\alpha_m \geqslant 0,\ \forall\, m = 1, 2, \ldots, M \tag{31}$$

Gradient Boosting

Gradient boosting is a generation of boosting algorithms, using the connection between boosting and optimization. For the boosting part, it builds an additive model in a forward stage-wise fashion. For the optimization part, it allows for the optimization of arbitrary differentiable loss functions by using their gradients.

In any function estimation problem, we wish to find a regression function $f(x) \in \mathcal{F}$ that minimizes the expectation of some loss function, where $f(x)$ is a function that maps from the input space to $\mathbb{R}$, and $\mathcal{F}$ is the hypothesis space of all possible regression functions.

Denote a given loss function as ℓ. The Gradient Boosting algorithm contains M steps. At each step, it tries to build a regression functions $h_m(x)$ and adds it to the ensembled function $f_m(x)$ to minimize ℓ. In the end all functions of M steps add up to form the final regression function $f_M(x)$. The details are described as follows.

1. Initialize $f_0(x) = 0$.
2. For $m = 1$ to M:

(a) Compute the gradient:

$$(\boldsymbol{g}_m)_i = \frac{\partial}{\partial f(x_i)} \ell(y_i, f(x_i)) \Big|_{f(x_i)=f_{m-1}(x_i)}, \tag{32}$$

where $\{y_i, x_i\}_1^n$ are n data samples.

(b) The negative gradient $-\boldsymbol{g}_m$ is said to define the "steepest-descent" direction. Thus, we could use the negative gradient as the working response and fit regression model to $-\boldsymbol{g}_m$:

$$h_m = \underset{h \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} \left((-\boldsymbol{g}_m)_i - h(x_i) \right)^2, \tag{33}$$

each $h_m \in \mathcal{F}$ is chosen in a learning process.

(c) Choose fixed step size $\nu_m = \nu \in (0, 1]$, or take

$$\nu_m = \underset{\nu > 0}{\operatorname{argmin}} \sum_{i=1}^{n} \ell \left(y_i, f_{m-1}(x_i) + \nu h_m(x_i) \right), \tag{34}$$

where ν_m is the size of the step along the direction of the greatest descent.

(d) Update the estimate of $f(x)$ as:

$$f_m(x) = f_{m-1}(x) + \nu_m h_m(x). \tag{35}$$

3. Return f_M.

In this problem we'll derive two special cases of the general gradient boosting framework: L_2-Boosting and Binomial Boost.

2.1 Consider the regression framework, where label space $\mathcal{Y} = \mathbb{R}$. Suppose our loss function is given by

$$\ell(\hat{y}, y) = \frac{1}{2} (\hat{y} - y)^2, \tag{36}$$

and at the beginning of the m'th round of gradient boosting, we have the function $f_{m-1}(x)$. Show that the h_m chosen as the next basis function is given by

$$h_m = \underset{h \in \mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[(y_i - f_{m-1}(x_i)) - h(x_i) \right]^2. \tag{37}$$

In other words, at each stage we find the weak prediction function $h_m \in \mathcal{F}$ that is the best fit to the residuals from the previous stage.

Hint: Once you understand what's going on, this is a pretty easy problem.

解: 由损失函数的定义可得

$$\ell(y_i, f(x_i)) = \frac{1}{2} (y_i - f(x_i))^2 \tag{38}$$

则

$$\begin{aligned}(\boldsymbol{g}_m)_i &= \frac{\partial}{\partial f(x_i)}\ell(y_i, f(x_i))\Big|_{f(x_i)=f_{m-1}(x_i)} \\ &= \frac{\partial}{\partial f_{m-1}(x_i)}\left[\frac{1}{2}(y_i - f_{m-1}(x_i))^2\right] \\ &= -(y_i - f_{m-1}(x_i))\end{aligned} \tag{39}$$

所以,

$$\begin{aligned}h_m &= \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[(-\boldsymbol{g}_m)_i - h(x_i)\right]^2 \\ &= \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[(y_i - f_{m-1}(x_i)) - h(x_i)\right]^2\end{aligned} \tag{40}$$

2.2 Now let's consider the classification framework, where $\mathcal{Y} = \{-1, 1\}$. This time, let's consider the logistic loss

$$\ell(m) = \ln\left(1 + e^{-m}\right), \tag{41}$$

where $m = yf(x)$ is the margin. Similar to what we did in the L_2-Boosting question, write an expression for h_m as an argmin over $\mathcal{F}$.

解: 由损失函数的定义可得

$$\ell(y_i, f(x_i)) = \ln[1 + \exp(-y_i f(x_i))] \tag{42}$$

则

$$\begin{aligned}(\boldsymbol{g}_m)_i &= \frac{\partial}{\partial f(x_i)}\ell(y_i, f(x_i))\Big|_{f(x_i)=f_{m-1}(x_i)} \\ &= \frac{\partial \ln[1 + \exp(-y_i f_{m-1}(x_i))]}{\partial f_{m-1}(x_i)} \\ &= -\frac{y_i \exp(-y_i f_{m-1}(x_i))}{1 + \exp(-y_i f_{m-1}(x_i))}\end{aligned} \tag{43}$$

所以,

$$\begin{aligned}h_m &= \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} [(-\boldsymbol{g}_m)_i - h(x_i)]^2 \\ &= \underset{h\in\mathcal{F}}{\operatorname{argmin}} \sum_{i=1}^{n} \left[\frac{y_i \exp(-y_i f_{m-1}(x_i))}{1 + \exp(-y_i f_{m-1}(x_i))} - h(x_i)\right]^2\end{aligned} \tag{44}$$

2.3 (Optional) What are the similarities and differences between Gradient Boosting and Gradient Descent?

解: 梯度提升算法和梯度下降算法都是在每一轮迭代中, 利用损失函数相对于模型的负梯度方向的信息来对当前模型进行更新, 只不过在梯度下降中, 模型以参数化形式表示, 从而模型的更新是在参数空间进行的. 而在梯度提升中, 模型直接定义在函数空间, 从而模型的更新是在函数空间进行的.

Programming: AdaBoost

The goal of this problem is to give you an overview of the procedure of *AdaBoost*. Here, our "weak learners" are *decision stumps*. Our data consist of $X \in \mathbb{R}^{n\times p}$ matrix with each row a sample and label vector $y \in \{-1,+1\}^n$. A decision stump is defined by:

$$h_{(a,d,j)}(\boldsymbol{x}) = \begin{cases} d, & \text{if } x_j \leqslant a, \\ -d, & \text{otherwise,} \end{cases} \tag{45}$$

where $a \in \mathbb{R}$, $j \in \{1,...,p\}$, $d \in \{-1,+1\}$. Here $\boldsymbol{x} \in \mathbb{R}^p$ is a vector, and x_j is the j-th coordinate.

Directory of the data is `/code/ada_data.mat`. It contains both a training and testing set of data. Each consists of 1000 samples. There are 25 real valued features for each sample, and a corresponding y label.

3.1 Complete the code skeleton `decision_stump.m` (or `decision_stump()` in `adaboost.py` if you use python). This program takes as input: the data along with a set of weights (i.e., $\{(\boldsymbol{x}_i, y_i, w_i)\}_{i=1}^n$, where $w_i \geqslant 0$ and $\sum_{i=1}^n w_i = 1$), and returns the decision stump which minimizes the weighted training error. Note that this requires selecting both the optimal a, d of the stump, and also the optimal coordinate j.

The output should be a pair $(a^\star, d^\star, j^\star)$ with:

$$l(a^\star, d^\star, j^\star) = \min_{a,d,j} l(a,d,j) = \min_{a,d,j} \sum_{i=1}^n w_i 1_{\{h_{a,d,j}(\boldsymbol{x}_i)\neq y_i\}}. \tag{46}$$

Your approach should run in time $O(pn\log n)$ or better. Include details of your algorithm in the report and analyze its running time.

Hint: you may need to use the function `sort` provided by MATLAB or python in your code, we can assume its running time to be $O(m\log m)$ when considering a list of length m.

3.2 Complete other two code skeletons `update_weights.m` and `adaboost_error.m`. Then run the `adaboost.m`, you will carry out AdaBoost using decision stumps as the "weak learners". (Complete the code in `adaboost.py` if you use python).

3.3 Run your AdaBoost loop for 300 iterations on the data set, then plot the training error and testing error with iteration number as the x-axis.

解: 选择决策树桩时首先对每一维特征进行排序, 复杂度为 $O(n\log n)$, 然后将排序后的数据从小到大对加权的标签积分, 积分值最大的地方就是分界面, 求取分界面的复杂度为 $O(n)$, 特征共 p 维, 因此总的时间复杂度为

$$O(pn\log n + pn) \approx O(pn\log n) \tag{47}$$

课堂上经过理论推导得知, 权重调整系数为

$$\alpha_l = \frac{1}{2}\ln\left(\frac{1-\epsilon_l}{\epsilon_l}\right) \tag{48}$$

而示例代码中给出 `alpha[i]=np.log((1-e)/e)`, 故将其修改为 `alpha[i]=1/2 * np.log((1-e)/e)`, 否则算法会出现明显的震荡现象.

在数据集上循环 300 次 AdaBoost 算法, 得到训练错误率和测试错误率与循环次数的变化曲线如图 1 所示, 可以看出随着训练集上的错误率逐渐趋于 0, 测试集错误率逐渐趋于 0.1.

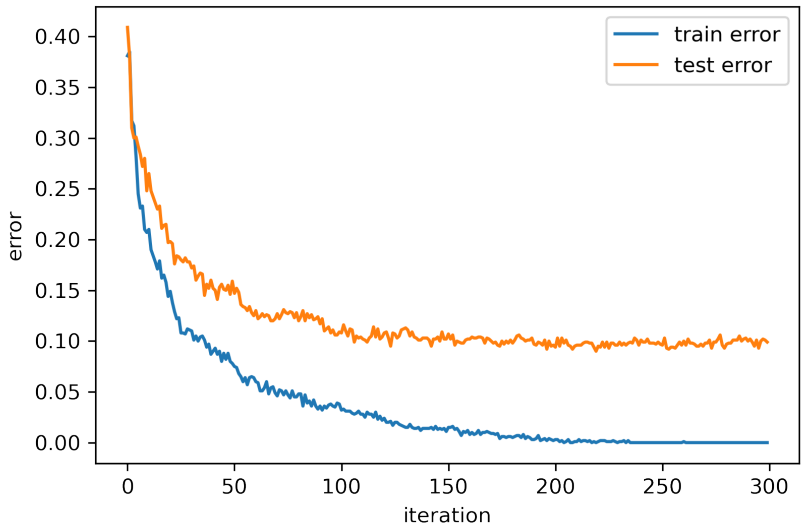


图 1: 训练错误率和测试错误率

THU-70250043-0, Pattern Recognition (Spring 2021) **Homework: 13**

Clustering Methods

Lecturer: Changshui Zhang zcs@mail.tsinghua.edu.cn

Hong Zhao vzhao@tsinghua.edu.cn

Student: Jingxuan Yang yangjx20@mails.tsinghua.edu.cn

K-means

1. Consider a set D of $n = 2k + 1$ samples, k of which coincide at $x = -2$, k at $x = 0$ and one at $x = a > 0$. As you have learned in class, we are always trying to minimize the distortion measure in K-means, given by

$$J_e = \sum_{D_i \neq \varnothing} \sum_{x \in D_i} \|x - m_i\|^2 \tag{1}$$

where m_i is the mean value of the samples of the nonempty subset D_i.

1.1 Show that the two-cluster partitioning that minimizes J_e groups the k samples at $x = 0$ with the one at $x = a$ if $a^2 < 2(k + 1)$.

解: 首先证明 k 个位于 $x = -2$ 的样本属于同一类.

假设 $x_i = -2 \in D_1$, $x_j = -2 \in D_2$, 我们可以断言将 x_i 从 D_1 取出放入 D_2 会使得 J_e 下降, 因为如若不然则将 x_j 从 D_2 取出放入 D_1 会使得 J_e 下降. 因此, 所有 k 个位于 $x = -2$ 的样本属于同一类. 同理可知 k 个位于 $x = 0$ 的样本也属于同一类. 所以, 在进行两类聚类时, 全部可能的结果有三种情况:

(a) $x = -2, x = 0 \in D_1$, $x = a \in D_2$

(b) $x = -2, x = a \in D_1$, $x = 0 \in D_2$

(c) $x = -2 \in D_1$, $x = 0, x = a \in D_2$

对三种情况分别计算 J_e 可得

$$\begin{aligned} J_e(a) &= k\| - 2 - (-1)\|^2 + k\|0 - (-1)\|^2 = 2k \\ J_e(b) &= k\left\|-2 - \frac{-2k + a}{k + 1}\right\|^2 + \left\|a - \frac{-2k + a}{k + 1}\right\|^2 = \frac{k}{k + 1}(a + 2)^2 \\ J_e(c) &= k\left\|0 - \frac{a}{k + 1}\right\|^2 + \left\|a - \frac{a}{k + 1}\right\|^2 = \frac{k}{k + 1}a^2 \end{aligned} \tag{2}$$

由 $a > 0$ 可知 $J_e(b) > J_e(c)$, 若 $a^2 < 2(k+1)$, 则

$$J_e(c) = \frac{k}{k+1}a^2 < 2k = J_e(a) \tag{3}$$

因此, 当 $a^2 < 2(k+1)$ 时, 两类聚类的情况为 (c), 即 k 个 $x = -2$ 的点属于一个类, k 个 $x = 0$ 的点以及 $x = a$ 属于另一个类.

1.2 What is the optimal grouping if $a^2 > 2(k+1)$?

解: 由 1.1 可知当 $a^2 > 2(k+1)$ 时, 有

$$J_e(b) > J_e(c) > J_e(a) \tag{4}$$

因此两类聚类的情况为 (a), 即 k 个 $x = -2$ 的点与 k 个 $x = 0$ 的点属于一个类, $x = a$ 属于另一个类.

Hierarchical Clustering

2. Consider a hierarchical clustering procedure in which clusters are merged to produce the smallest increase in the sum-of-squared error at each step. If the i-th cluster contains n_i samples with the sample mean $\boldsymbol{m}_i$, show that the smallest increase results from merging the pair of clusters for which

$$\frac{n_i n_j}{n_i + n_j} \|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2 \tag{5}$$

is minimum.

解: 合并第 i 类与第 j 类的均方误差增量为

$$\begin{aligned}
\Delta E &= \sum_{\boldsymbol{x}\in D_i, D_j} \left\| \boldsymbol{x} - \frac{n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j}{n_i + n_j} \right\|^2 - \sum_{\boldsymbol{x}\in D_i} \|\boldsymbol{x} - \boldsymbol{m}_i\|^2 - \sum_{\boldsymbol{x}\in D_j} \|\boldsymbol{x} - \boldsymbol{m}_j\|^2 \\
&= \sum_{\boldsymbol{x}\in D_i, D_j} \left[\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top \frac{n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j}{n_i + n_j} + \frac{(n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j)^\top(n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j)}{(n_i + n_j)^2} \right] \\
&\quad - \sum_{\boldsymbol{x}\in D_i} (\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top\boldsymbol{m}_i + \boldsymbol{m}_i^\top\boldsymbol{m}_i) - \sum_{\boldsymbol{x}\in D_j} (\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top\boldsymbol{m}_j + \boldsymbol{m}_j^\top\boldsymbol{m}_j) \\
&= -\frac{(n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j)^\top(n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j)}{n_i + n_j} + n_i\boldsymbol{m}_i^\top\boldsymbol{m}_i + n_j\boldsymbol{m}_j^\top\boldsymbol{m}_j \\
&= -\frac{(n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j)^\top(n_i\boldsymbol{m}_i + n_j\boldsymbol{m}_j)}{n_i + n_j} + \frac{(n_i\boldsymbol{m}_i^\top\boldsymbol{m}_i + n_j\boldsymbol{m}_j^\top\boldsymbol{m}_j)(n_i + n_j)}{n_i + n_j} \\
&= \frac{-2n_in_j\boldsymbol{m}_i^\top\boldsymbol{m}_j + n_in_j\boldsymbol{m}_i^\top\boldsymbol{m}_i + n_in_j\boldsymbol{m}_j^\top\boldsymbol{m}_j}{n_i + n_j} \\
&= \frac{n_in_j}{n_i + n_j}\|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2
\end{aligned} \tag{6}$$

因此, 当

$$\frac{n_i n_j}{n_i + n_j} \|\boldsymbol{m}_i - \boldsymbol{m}_j\|^2 \tag{7}$$

最小时, 合并第 i 类与第 j 类的均方误差增量最小.

Spectral Clustering

3. Given a set of m data points $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_m$, the input to a spectral clustering algorithm typically consists of a matrix A, of pairwise similarities between data points. A is often called the affinity matrix. The choice of how to measure similarity between points is one which often left to the practitioner. A very simple affinity matrix can be constructed as follows:

$$A(i,j) = A(j,i) = \begin{cases} 1, & \text{if } d(\boldsymbol{x}_i, \boldsymbol{x}_j) < \Theta \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

The general idea of spectral clustering is to construct a mapping of the data points to an eigenspace of A with the hope that points are well separated in this eigenspace so that something simple like $k-means$ applied to these new points will perform well.

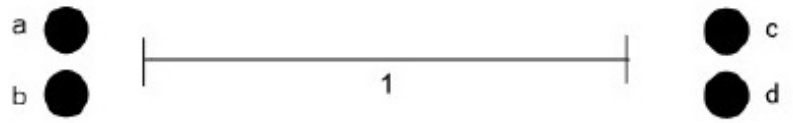


图 1: A simple data set.

As an example, consider forming the affinity matrix for the dataset in Figure 1 using Equation 8 with $\Theta = 1$. Then we get the affinity matrix in Figure 2.

$$A = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 1 & 1 & 0 & 0 \\ b & 1 & 1 & 0 & 0 \\ c & 0 & 0 & 1 & 1 \\ d & 0 & 0 & 1 & 1 \end{array} \qquad \tilde{A} = \begin{array}{c|cccc} & a & c & b & d \\ \hline a & 1 & 0 & 1 & 0 \\ c & 0 & 1 & 0 & 1 \\ b & 1 & 0 & 1 & 0 \\ d & 0 & 1 & 0 & 1 \end{array}$$

(a) (b)

图 2: Affinity matrices of Figure 1 with $\Theta = 1$.

Now for this particular example, the clusters $\{a, b\}$ and $\{c, d\}$ show up as nonzero blocks in the affinity matrix.

This is artificial since we could have constructed the matrix A using an ordering of $\{a, b, c, d\}$. For example, another possible affinity matrix for A could have been as in Figure 2(b).

The key insight here is that the eigenvectors of matrices A and $\tilde{A}$ have the same entries (just permuted). The eigenvectors with nonzero eigenvalues of A are $\boldsymbol{e}_1 = (1, 1, 0, 0)^\top$ and $\boldsymbol{e}_2 = (0, 0, 1, 1)^\top$. And the nonzero eigenvectors of $\tilde{A}$ are: $\boldsymbol{e}_1 = (1, 0, 1, 0)^\top$ and $\boldsymbol{e}_2 = (0, 1, 0, 1)^\top$. Spectral clustering embeds the original data points into a new space by using the coordinates of these eigenvectors. Specifically, it maps the point $\boldsymbol{x}_i$ to the point $\boldsymbol{e}_1(i), \boldsymbol{e}_2(i), \ldots, \boldsymbol{e}_k(i)$ where $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_k$ are the top k eigenvectors of A corresponding to the biggest eigenvalues. We refer to this mapping as the spectral embedding.

Hint: This is different from what you learned in class. With the Graph Laplacian L, we are trying to find its eigenvectors corresponding to the smallest eigenvalues. What you should notice here is that the matrix A is not Graph Laplacian, this leads to different algorithms. You can analyze the underlying meaning of this approach as what we did in the class.

In this problem, we will analyze the operation of one of the variants of spectral clustering methods on a simple data set shown in Figure 3:

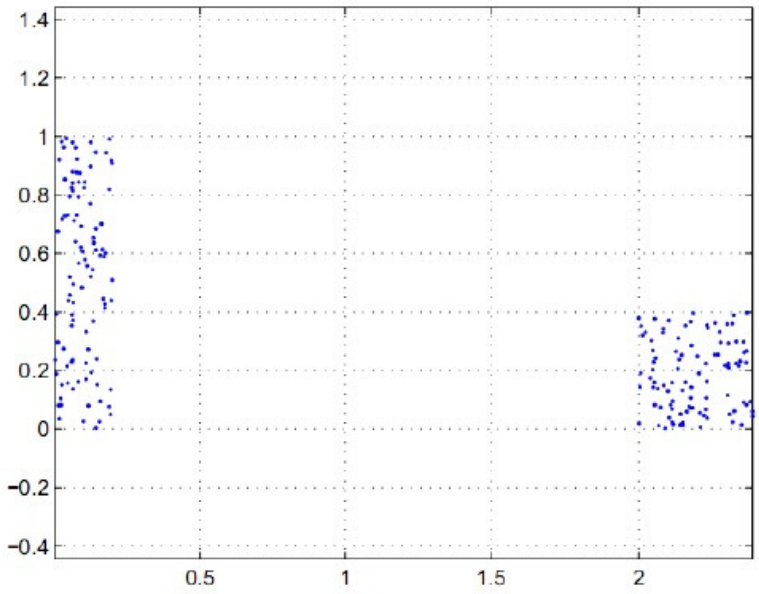


图 3: Dataset with rectangles

3.1 For the data set in the above figure, assume that the first cluster has m_1 points and the second one has m_2 points. If we use Equation 8 to compute the corresponding affinity matrix A, what Θ value would you choose and why?

解: 记图 3 中左侧数据为第 1 堆 (cluster), 右侧数据为第 2 堆. 记第 1 堆数据之间的最大距离为 d_1, 第 1 堆数

据与第 2 堆数据之间的最短距离为 d_2. 为使每堆数据之内的相似性度量为 1 而两堆数据之间的相似性度量为 0, 则需选择

$$\Theta \in (d_1, d_2] \tag{9}$$

根据图中具体数据可知, 选择 $\Theta = 1.5$ 可以满足上述要求.

3.2 The second step is to compute first k dominant eigenvectors of the affinity matrix, where k is the number of clusters we want to have. For the data set in Figure 3 and the affinity matrix defined by Equation 8, is there a value of Θ for which you can analytically compute the first two eigenvalues and eigenvectors? If not, explain why not. If yes, compute and write these eigenvalues and eigenvectors down. What are the other eigenvalues? Explain briefly.

解: 存在, 取 3.1 中 $\Theta = 1.5$ 即可解析计算相似 (affinity) 矩阵 A 的前两个特征值与特征向量.

首先通过调整顺序可写出 A 为

$$A = \begin{bmatrix} \mathbf{1}_{m_1 \times m_1} & \mathbf{0}_{m_1 \times m_2} \\ \mathbf{0}_{m_2 \times m_1} & \mathbf{1}_{m_2 \times m_2} \end{bmatrix} \tag{10}$$

易知其前两个特征值为 m_1 与 m_2, 对应的特征向量分别为

$$\boldsymbol{e}_1 = \begin{bmatrix} \mathbf{1}_{m_1 \times 1} \\ \mathbf{0}_{m_2 \times 1} \end{bmatrix}, \quad \boldsymbol{e}_2 = \begin{bmatrix} \mathbf{0}_{m_1 \times 1} \\ \mathbf{1}_{m_2 \times 1} \end{bmatrix} \tag{11}$$

由于矩阵 A 的秩 $\operatorname{rank}(A) = 2$, 则其他特征值均为 0.

3.3 We can now compute the spectral embedding of the data points using the k top eigenvectors. For the data set in Figure 3, write down your best guess for the coordinates of the $k = 2$ cluster centers using the Θ that you picked in the first part.

解: 根据 $\boldsymbol{e}_1$ 与 $\boldsymbol{e}_2$ 可知, 两堆的聚类中心分别为

$$\boldsymbol{c}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \boldsymbol{c}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{12}$$

Programming: K-means, hierarchical clustering and spectral clustering

Test the clustering algorithms K-means, hierarchical clustering and spectral clustering with different parameters on MNIST dataset or subsets of it when the scale is too large for the algorithm involved.

To compare the effectiveness of different clustering methods, *Normalized mutual information* (NMI) are widely

used as a measurement. NMI is defined as following:

$$\mathrm{NMI} = \frac{\sum_{s=1}^{K}\sum_{t=1}^{K} n_{s,t}\log\left(\frac{nn_{s,t}}{n_s n_t}\right)}{\sqrt{\left(\sum_{s=1}^{K} n_s \log\frac{n_s}{n}\right)\left(\sum_{t=1}^{K} n_t \log\frac{n_t}{n}\right)}}. \tag{13}$$

Where n is the number of data points, n_s and n_t denote the numbers of the data in class s and class t, $n_{s,t}$ denotes the number of data points in both class s and class t. For more details and other measurements, google "evaluation of clustering".

4.1 Give a brief analysis of time complexity of each algorithm mentioned above (of standard implementation). Estimate how many samples each algorithm can manage with a reasonable time cost.

(Optional) Can you verify your estimations with experiments? Can you speed it up further?

解: 记 k 为分类数, n 为点数, T 为算法循环次数, 则 K-means 算法的复杂度为 $O(knT)$. 分级聚类算法计算两类距离复杂度 $O(n^2)$, 总共计算 n 次, 所以分级聚类算法的复杂度为 $O(n^3)$. 谱聚类算法计算相似度矩阵的复杂度为 $O(n^2)$, 计算特征值与特征向量的复杂度为 $O(n^3)$, 所以谱聚类算法的复杂度为 $O(n^3)$.

以 CPU 主频为 3GHz 来估算, 若需要 1s 之内计算完成, 则最多计算样本数大约为:

(a) K-means: 3×10^9

(b) 分级聚类: $\sqrt[3]{3} \times 10^3$

(c) 谱聚类: $\sqrt[3]{3} \times 10^3$

4.2 Consider each data set, and use the true number of classes as the number of clusters.

(1) With K-means, will the initial partition affect the clustering results? How can you solve this problem? And do J_e and NMI match? Show your experiment results.

解: 选择样本数量 $n = 3000$, 分别使用 k-means++ [1], 随机初始化与聚类中心重合初始化三种初始化方法进行 K-means 聚类, 其结果如表 1 所示.

表 1: K-means 聚类结果

methods	$J_e \times 10^9$	NMI
k-means++	7.527918	0.495957
random	7.514934	0.503458
coincide	7.700415	0.473963

由表 1 可知, 三种初始化方法对聚类结果有影响, 但是影响并不是很大. 实际上, K-means 可能会受到初始化影

响而陷入局部极小, 对此我们一般可以采用多次随机初始化, 然后取效果最好的结果. 由表 1 也可看出当样本数量相同时, J_e 越小则 NMI 越大, 聚类效果越好. 但是当样本数量发生变化时, J_e 与 NMI 的变化情况则会有所不同.

为了测试 J_e 与 NMI 关于样本数量的变化关系, 选择样本数量 $n = [100, 200, \ldots, 3000]$, 使用 k-means++ 初始化方法得到 K-means 聚类结果 J_e 如图 4 所示, NMI 如图 5 所示.

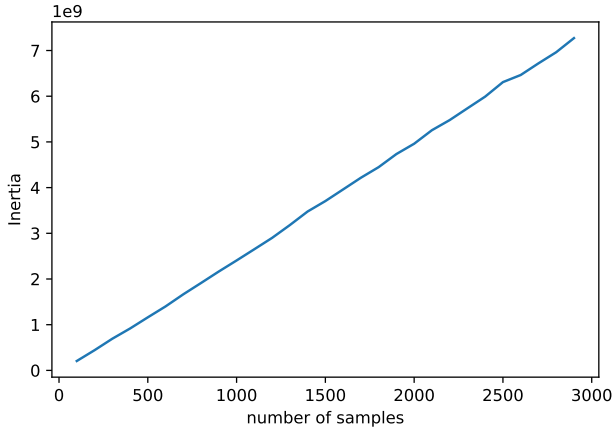


图 4: K-means 算法 k-means++ 初始化 J_e 与样本数量变化关系

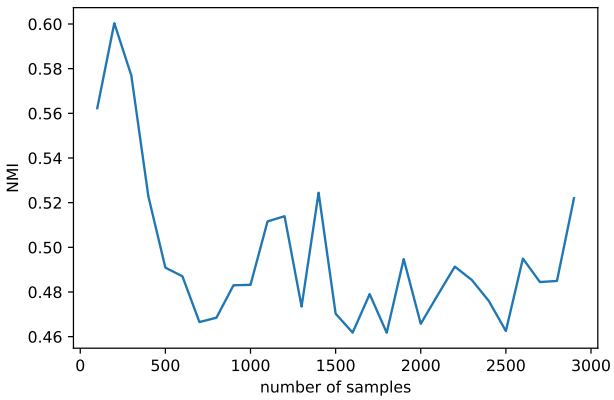


图 5: K-means 算法 k-means++ 初始化 NMI 与样本数量变化关系

使用随机初始化方法得到 K-means 聚类结果 J_e 如图 6 所示, NMI 如图 7 所示.

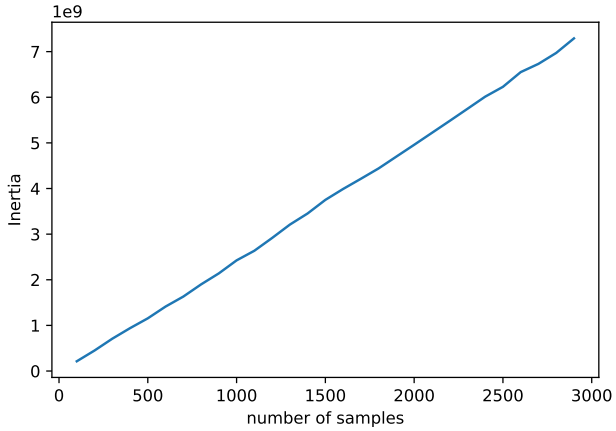


图 6: K-means 算法随机初始化 J_e 与样本数量变化关系

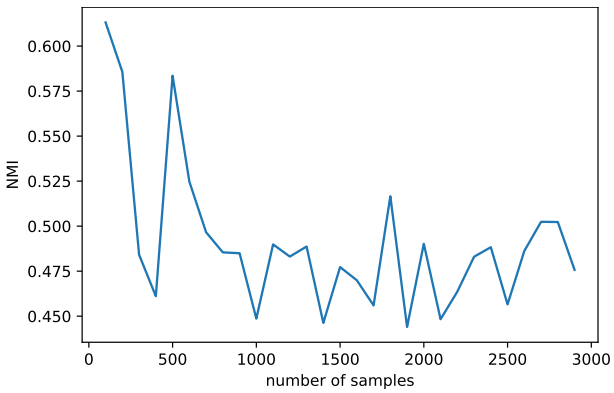


图 7: K-means 算法随机初始化 NMI 与样本数量变化关系

使用聚类中心重合初始化方法得到 K-means 聚类结果 J_e 如图 8 所示, NMI 如图 9 所示.

由此 6 幅对比图可知, 随着样本数量的增长, J_e 单调增加, 但是 NMI 呈现震荡变化, 因此若样本数量发生变化, 则 J_e 与 NMI 的结果不一定匹配 (match).

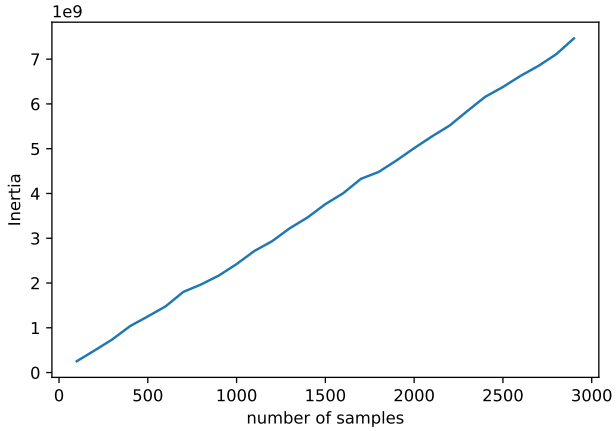


图 8: K-means 算法中心重合初始化 J_e 与样本数量变化关系

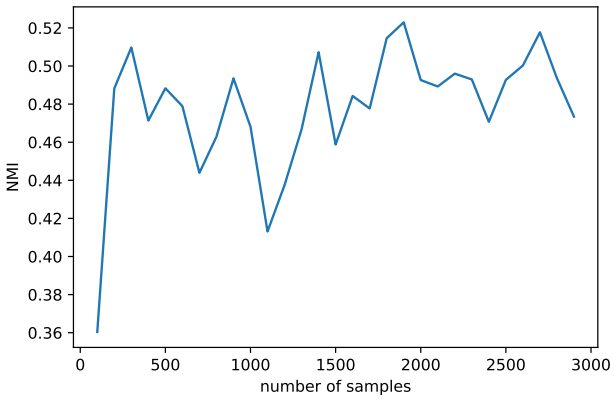


图 9: K-means 算法中心重合初始化 NMI 与样本数量变化关系

(2) When hierarchical clustering is adopted, the choice of linkage method depends on the problem. Give an analysis of linkage method's effects with experiments, and which is better in the sense of NMI?

As introduced in the class, some of the most common metrics of distance between two clusters $\{x_1, \ldots, x_m\}$ and $\{y_1, \ldots, y_p\}$ are:

- *Single linkage:* Distance between clusters is the *minimum* distance between any pair of points from the two clusters, i.e.,
$$\min_{i,j} \|x_i - y_j\| \tag{14}$$
- *Complete linkage:* Distance between clusters is the *maximum* distance between any pair of points from two clusters, i.e.,
$$\max_{i,j} \|x_i - y_j\| \tag{15}$$
- *Average linkage:* Distance between clusters is the *average* distance between all pairs of points from two clusters, i.e.,
$$\frac{1}{mp} \sum_{i=1}^{m} \sum_{j=1}^{p} \|x_i - y_j\| \tag{16}$$

解: 为了测试三种不同距离度量的 NMI 关于样本数量的变化关系, 选择样本数量 $n = [100, 200, \ldots, 3000]$, 使用分级聚类方法得到 NMI 如图 10 所示.

由图 10 可知, 最近距离度量的 NMI 普遍低于另外两种度量的 NMI, 最远距离度量的 NMI 与平均距离度量的 NMI 相互交织, 但是大多时候最远距离度量的 NMI 略优于平均距离度量. 这说明 MNIST 数据的各个类别之间距离较近, 可能出现因为边缘距离近导致类别连接的情况. 同时 MNIST 的数据基本围绕在聚类中心各个方向的方差附近, 所以最远距离度量效果较好.

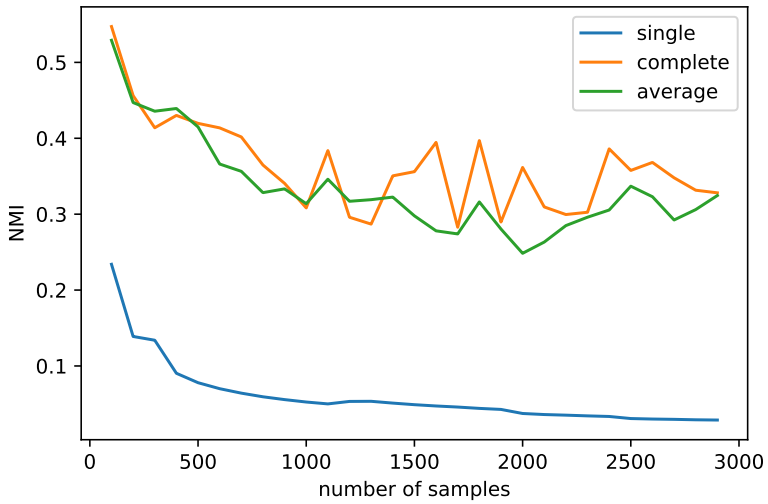


图 10: 分级聚类算法不同距离度量 NMI 与样本数量变化关系

(3) As for spectral clustering, give an experimental analysis of the choice of similarity graph and corresponding parameters. Which one is better?

解: 分别使用高斯核的全连接图和 k 近邻图进行相似矩阵构建, 谱聚类 NMI 结果如表 2 所示. 由表 2 可知最近邻的效果更好, 选择高斯核的全连接时, 很多点是孤立的因而没有构成一个完全图, 所以聚类效果很差. 而最近邻中 $k = 5$ 的效果最好, 若 k 太大, 有可能导致不同类之间的样本连接起来.

表 2: 谱聚类全连通图与 k 近邻图实验结果

methods	NMI
rbf, $\gamma = 1$	0.101404
rbf, $\gamma = 2$	0.076044
rbf, $\gamma = 4$	0.074711
knn, $k = 5$	0.614761
knn, $k = 50$	0.519622
knn, $k = 100$	0.468890

4.3 In practice, we may not know the true number of clusters much. Can you give a strategy to identify the cluster number automatically for each algorithm? Show your results.

解: 可以使用轮廓系数 (Silhouette Coefficient) [2] 来进行聚类数量的选择, 其计算公式为

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{17}$$

其中, $b(i)$ 是点 i 到任何其他堆中所有点的最小平均距离, $a(i)$ 是点 i 到其所在堆中其他点的平均距离.

绘出 K-means 的轮廓系数曲线如图 11 所示, 在轮廓系数曲线中 $k = 2$ 时轮廓系数最大, 按照最大轮廓系数选择

结果为 $k = 2$. 实际上, 我们可以结合一些先验知识, $k = 11$ 的轮廓系数仅次于 $k = 2$, 与实际的类别数 $k = 10$ 更为接近.

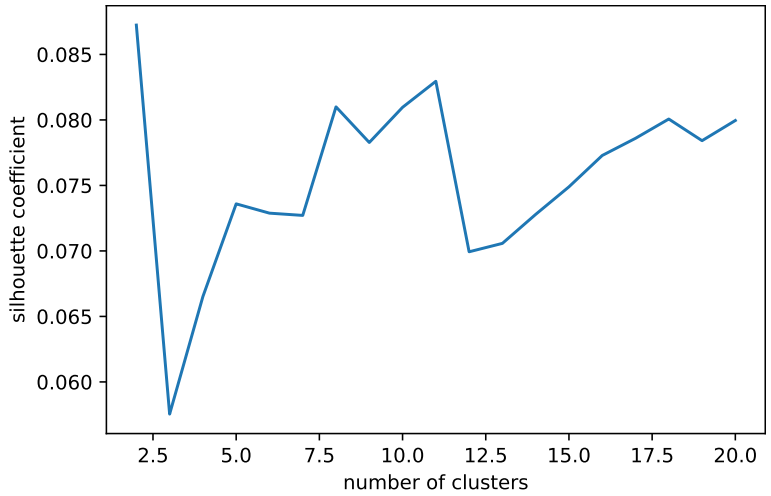


图 11: K-means 算法轮廓系数随聚类类别数变化曲线

4.4 According to the above analysis, which method do you prefer? Why?

解: 以上三种聚类方法各有优劣, 聚类方法的选择要根据样本的分布特性和数量综合考虑. 比如若样本点成团状分布或者样本数很大时, 则用 K-means 算法能取得较好效果, 且速度快; 而多级聚类和谱聚类在样本数很大时由于时间复杂度过大而无法使用. 当样本数量较少时, 可以选择基于最近邻图的谱聚类方法, 其聚类的效果较好, 而且不像分级聚类那样受距离度量选择的影响大.

参考文献

[1] sklearn.cluster.KMeans, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans.

[2] Rousseeuw, Peter J. “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis.” Journal of computational and applied mathematics, 20, 1987: 53-65.