CSIE 5432/5433 — Machine Learning Foundations/Techniques

Name: 李吉昌
Student Number: r08922a27

Homework 6
Due Date: January 15 2020, 13:00

Neural Networks

1. Answer: [b]

根據 Lecture 212 slides 第 15 頁公式可得 $\delta_j^{(l)} = \sum_k (\delta_k^{(l+1)})(w_{jk}^{(l+1)})(\tanh'(s_j^{(l)}))$, 欲求取所有 $\delta_j^{(l)}$ 時, 下一層與 $w_{jk}^{(l+1)}$ 的乘積都需要被計算過一次, 因此所有 $\delta_j^{(l)}$ 所需計算量為 $w_{jk}^{(l+1)}$ 的數量。當計算所有 $\delta_j^{(1)}$ 時, 需經過 $5 \times 6 = 30$ 個 operation; 當計算所有 $\delta_j^{(2)}$ 時, 需經過 6×1 個 operation, 共計 $30 + 6 = 36$ 個 operation。

2. Answer: [d]

令 $n^{(l)} = d^{(l)} + 1$, $\sum_{l=1}^{L-1} n^{(l)} = 50$, $n^{(0)} = 19 + 1 = 20$, 證明需要分為兩階段, 第一階段證明選擇 $L \geq 4$ 的架構所得參數量必定不是最大值, 第二階段為比較在 $L = 3$ 和 $L = 2$ 的最大參數量:

Step 1 :
需要分成 $n^{(1)} \geq n^{(2)}$ 和 $n^{(1)} < n^{(2)}$ 兩種情況討論:

(1) 若 $n^{(1)} \geq n^{(2)}$:
存在一 $L-1$ 的新架構將所有 $n^{(2)}$ 拔掉並接到 $n^{(3)}$, 則新架構與舊架構的參數量變化為:
$[n^{(1)} \times (n^{(2)} + n^{(3)} - 1) + (n^{(2)} + n^{(3)}) \times (n^{(4)} - 1)]$
$\quad - [n^{(1)} \times (n^{(2)} - 1) + n^{(2)} \times (n^{(3)} - 1) + n^{(3)} \times (n^{(4)} - 1)]$
$= n^{(3)} \times (n^{(1)} - n^{(2)}) + n^{(2)} \times n^{(4)} > 0$ $[\because n^{(1)} \geq n^{(2)}]$

(2) 若 $n^{(1)} < n^{(2)}$:
存在一 $L-1$ 的新架構將所有 $n^{(1)}$ 拔掉並接到 $n^{(3)}$, 則新架構與舊架構的參數量變化為:
$[n^{(0)} \times (n^{(2)} - 1) + n^{(2)} \times (n^{(1)} + n^{(3)} - 1) + (n^{(1)} + n^{(3)}) \times (n^{(4)} - 1)]$
$\quad - [n^{(0)} \times (n^{(1)} - 1) + n^{(1)} \times (n^{(2)} - 1) + n^{(2)} \times (n^{(3)} - 1) + n^{(3)} \times (n^{(4)} - 1)]$
$= n^{(0)} \times (n^{(2)} - n^{(1)}) + n^{(1)} \times n^{(4)} > 0$ $[\because n^{(1)} < n^{(2)}]$

由 (1)(2) 得, 只要 $L \geq 4$, 必定可以找到一層數更少的新架構參數量大於原本架構, 因此參數量極大值必存在於 $L \leq 3$。

Step 2:
(1) 若 $L = 3$:
$n^{(2)} = 50 - n^{(1)}$, 參數量可以寫作: $N_w(n^{(1)}) = 20 \times (n^{(1)} - 1) + n^{(1)} \times (50 - n^{(1)} - 1) + 3 \times (50 - n^{(1)})$
整理後得 $N_w(n^{(1)}) = -(n^{(1)})^2 + 66n^{(1)} + 130$, 當 $\frac{\partial N_w(n^{(1)})}{\partial n^{(1)}} = 0$ 有極大值, $-2n^{(1)} + 66 = 0$, 則 $n^{(1)} = 33$ 時, $N_w(n^{(1)})$ 有極大值 1219。

(2) 若 $L = 2$:
$n^{(1)} = 50$, 其參數量為 $20 \times (n^{(1)} - 1) + n^{(1)} \times 3 = 1130$

因為$1219 > 1130$, 由 (1)(2) 得, 在 $L = 3$ 且 $n^{(1)} = 33$ 時, 參數量為最大值 1219。

3. Answer: [d]

$$\frac{\partial}{\partial s_k^{(L)}}\{\text{err}(\mathbf{x},y)\} = \frac{\partial}{\partial s_k^{(L)}}\{-\sum_{i=1}^{K} v_k \ln q_k\} = \frac{\partial}{\partial s_k^{(L)}}\{-\sum_{i=1}^{K} v_i \ln(\frac{\exp(s_i^{(L)})}{\sum_{j=1}^{K}\exp(s_j^{(L)})})\}$$
$$= \frac{\partial}{\partial s_k^{(L)}}\{\sum_{i=1}^{K}(v_i \ln(\sum_{j=1}^{K}\exp(s_j^{(L)})) - v_i s_i^{(L)})\} = \frac{\frac{\partial}{\partial s_k^{(L)}}\{\sum_{j=1}^{K}\exp(s_j^{(L)})\}}{\sum_{j=1}^{K}\exp(s_j^{(L)})} - v_k$$
$$= \frac{\exp(s_k^{(L)})}{\sum_{j=1}^{K}\exp(s_j^{(L)})} - v_k = q_k - v_k$$

4. Answer: [a]

根據 Lecture 212 slides 第 15/16 頁公式可得 $\begin{cases} \delta_j^{(l)} = \sum_k(\delta_k^{(l+1)})(w_{jk}^{(l+1)})(\tanh'(s_j^{(l)})) \\ w_{ij}^{(l)} \leftarrow w_{ij}^{(l)} - \eta x_i^{(l-1)}\delta_j^{(l)} \end{cases}$

$x_0^{(0)}$ 為 bias 項恆為 $+1$, $w_{01}^{(1)}$ 的疊代更新式可寫作 $w_{01}^{(1)} \leftarrow w_{01}^{(1)} - \eta\delta_1^{(1)}$。
由題意給定所有參數初始化為 0, 在 $l \geq 1$ 的情況, 除了 bias 的 $x_0^{(l)}$ 項直接指定為 1, 其他所有 $x_j^{(l)} = 0$, 最後一層的 bias 項的權重 $w_{01}^{(L)}$ 對應到的 $\delta_1^{(L)}$ 可不為 0, 但 bias 項不被前一層的 neural 連接, 無法影響前一層的 $\delta_j^{(l)}$; 最後一層非 bias 的輸入, 因初始化參數設 0, 因此 $x_j^{(L-1)} = 0$, 在疊代後, 除了 bias 項的所有參數為 $w_{ij}^{(L)} = 0 - 0 \cdot \eta\delta_j^{(L)} = 0$。

由上述可推至所有不是最後一層的 $\delta_j^{(l)} = \sum_k 0 \cdot (\delta_k^{(l+1)})(\tanh'(s_j^{(l)})) = 0$, 使所有不是最後一層的 $w_{ij}^{(l)} = 0 - \eta x_i^{(l-1)} \cdot 0 = 0$, 互相影響下, 不是最後一層的 $\delta_j^{(l)} = 0$ 恆成立, 代入 $w_{01}^{(1)}$ 疊代式可得 $w_{01}^{(1)} = 0 - \eta \cdot 0 = 0$, 故 $w_{01}^{(1)} = 0$ 恆成立。

Matrix Factorization

5. Answer: [e]

令 v_n 為 $\mathbf{V}_{1\times N}$ 的第 n 個 element, w_m 為 $\mathbf{W}_{1\times M}$ 的第 m 個 element, 當 alternating least squares 更新 w_m 時, 目標函數 $L(w_m)$ 可以寫成 $\sum_{n=1}^{N}(v_n \cdot w_m - r_{nm})^2$, 根據題意, 因為 v_n 這時候為固定參數, 可代入 $v_n = 2$, 則 $L(w_m) = \sum_{n=1}^{N}(2 \cdot w_m - r_{nm})^2$。

當 $\frac{\partial L(w_m)}{\partial w_m} = 0$ 時有極值:
$\frac{\partial L(w_m)}{\partial w_m} = \frac{\partial}{\partial w_m}\{\sum_{n=1}^{N}(2 \cdot w_m - r_{nm})^2\} = \sum_{n=1}^{N} 2(2 \cdot w_m - r_{nm})$, 令 $\frac{\partial L(w_m)}{\partial w_m} = 0$, $2N \cdot w_m = \sum_{n=1}^{N} r_{nm}$, 綜上所述, 得 $w_m = \frac{1}{2}(\frac{\sum_{n=1}^{N} r_{nm}}{N})$。

6. Answer: [b]

令 $\nabla_{a_m} = \frac{\partial}{\partial a_m}\{(r_{nm} - \mathbf{w}_m^T\mathbf{v}_n - a_m - b_n)^2\} = -2(r_{nm} - \mathbf{w}_m^T\mathbf{v}_n - a_m - b_n)$。

將 $\nabla_{a_m} = -2(r_{nm} - \mathbf{w}_m^T\mathbf{v}_n - a_m - b_n)$ 代入更新疊代式 $a_m \leftarrow a_m - \frac{\eta}{2}\nabla_{a_m}$ 可寫作:

$a_m \leftarrow a_m - \frac{\eta}{2} \cdot (-2(r_{nm} - \mathbf{w}_m^T\mathbf{v}_n - a_m - b_n))$, 整理移項可得 $a_m \leftarrow (1-\eta)a_m + \eta \cdot (r_{nm} - \mathbf{w}_m^T\mathbf{v}_n - b_n)$

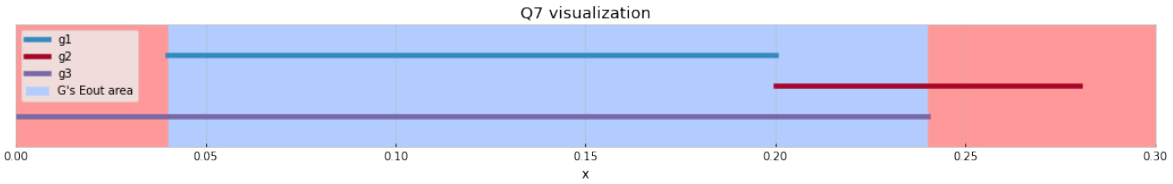
Aggregation

7. Answer: [d]

令 input 為 $\mathbf{x}_n$, label 為 $y_n \in \{-1, +1\}$, $G(\mathbf{x}_n) = \underset{k=-1,+1}{\text{argmax}} \sum_{t=1}^{3} [\![g_t(\mathbf{x}_n) = k]\!]$, 若 $G(\mathbf{x}_n) \neq y_n$ 則至少 g_1, g_2, g_3 至少有兩個 hypothesis 發生 $g_t(\mathbf{x}_n) \neq y_n$ 的情況。
由上述可知, 發生 $E_{out}(G) = 0.2$ 需滿足「至少兩個以上 hypothesis 發生錯誤的交集事件的機率為 0.2 」的條件, 令三個 hypothesis 的 E_{out} 大小關係為 $E_{out}(g_i) \geq E_{out}(g_j) \geq E_{out}(g_k)$, $e_i = [\![g_i(\mathbf{x}_n) \neq y_n]\!], e_j = [\![g_j(\mathbf{x}_n) \neq y_n]\!], e_k = [\![g_k(\mathbf{x}_n) \neq y_n]\!]$, 由題意給定的情況, 必要條件有二:

(1) 「$E_{out}(g_i) \geq 0.2$」, 因為 hypothesis G 發生錯誤的事件必為任意兩個以上 g_t 發生錯誤的交集事件, 因此 $E_{out}(G) \leq E_{out}(g_i)$ 恆成立, 若 $E_{out}(g_i) < 0.2$, 則 $E_{out}(G) < 0.2$ 亦恆成立。

(2) 「$E_{out}(g_j) + E_{out}(g_k) \geq 0.2$」, 因為 $E_{out}(g_j) + E_{out}(g_k) \geq P(e_j \cup e_k) \geq E_{out}(G)$ 恆成立, 若 $E_{out}(g_j) + E_{out}(g_k) < 0.2$, 則 $E_{out}(G) < 0.2$ 亦恆成立。



綜合上述兩條件, 僅 [d] 選項滿足上述兩必要條件, 由上圖所示, [d] 選項至少存在一組合滿足 $E_{out}(G) = 0.2$。

8. Answer: [c]

發生 $[\![y \neq G(\mathbf{x})]\!]$ 等價於發生 $\sum_{t=1}^{5} [\![y \neq g_t(\mathbf{x})]\!] \geq 3$, 則:

$$E_{out}(G) = P([\![y \neq G(\mathbf{x})]\!]) = P(\sum_{t=1}^{5} [\![y \neq g_t(\mathbf{x})]\!] \geq 3) = P(\bigcup_{i=3,4,5} (\sum_{t=1}^{5} [\![y \neq g_t(\mathbf{x})]\!] = i))$$
$$= P(\sum_{t=1}^{5} [\![y \neq g_t(\mathbf{x})]\!] = 3) + P(\sum_{t=1}^{5} [\![y \neq g_t(\mathbf{x})]\!] = 4) + P(\sum_{t=1}^{5} [\![y \neq g_t(\mathbf{x})]\!] = 5)$$
$$= C_3^5 \cdot 0.4^3 \cdot 0.6^2 + C_4^5 \cdot 0.4^4 \cdot 0.6^1 + C_5^5 \cdot 0.4^5 \cdot 0.6^0 = 0.31744 \approx 0.32$$

9. Answer: [b]

根據 Lecture 210 slides 第 8 頁公式, 一筆資料發生 Out-Of-Bag 的機率 P_{OOB} 可以寫做 $(1 - \frac{1}{N})^{N_s}$, 其中, N 為實際訓練資料的總數, N_s 為每次 bootstrap 抽樣來做訓練的資料數量。
根據題意 $N_s = 0.5N$, 當 $N \to \infty$ 時:

$$P_{\text{OOB}} = \lim_{N\to\infty} (1 - \tfrac{1}{N})^{0.5N} = \lim_{N\to\infty} (\tfrac{1}{(\frac{N}{N-1})^N})^{0.5} = \lim_{N\to\infty} (\tfrac{1}{(1+\frac{1}{N-1})^N})^{0.5} = \sqrt{\lim_{N\to\infty} \tfrac{1}{(1+\frac{1}{N-1})^N}}$$

根據 Lecture 210 slides 第 8 頁, 已知 $\lim_{N\to\infty} \frac{1}{(1+\frac{1}{N-1})^N} \approx \frac{1}{e}$, 則 $P_{\text{OOB}} \approx \frac{1}{\sqrt{e}} \approx 0.607 = 60.7\%$

10. Answer: [e]

$$K_{ds}(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}_{ds}(\mathbf{x})^T \boldsymbol{\phi}_{ds}(\mathbf{x}') = \sum_{i=1}^{d} \sum_{s\in\{-1,+1\}} \sum_{\theta\in\{2L+1, 2L+3, \ldots, 2R-1\}} g_{s,i,\theta}(\mathbf{x}) \cdot g_{s,i,\theta}(\mathbf{x}')$$

令 $\mathbf{x}$ 和 $\mathbf{x}'$ 的第 i 個維度為 x_i 和 x_i', $\sum_{\theta\in\{2L+1,2L+3,...,2R-1\}} g_{s,i,\theta}(\mathbf{x})\cdot g_{s,i,\theta}(\mathbf{x}')$ 需要分成兩種情況討論:

(1) 若 θ 介於 x_i 和 x_i' 之間的時候, $g_{s,i,\theta}(\mathbf{x})$ 和 $g_{s,i,\theta}(\mathbf{x}')$ 為異號, 則 $g_{s,i,\theta}(\mathbf{x})\cdot g_{s,i,\theta}(\mathbf{x}')=-1$,
當 $x_i'\geq x_i$ 時, $\theta\in\{x_i+1,x_i+3,...,x_i'-1\}$, 所有可能的 θ 總個數為 $\frac{(x_i'-1)-(x_i+1)}{2}+1=\frac{1}{2}(x_i'-x_i)$;
當 $x_i'<x_i$ 時, $\theta\in\{x_i'+1,x_i'+3,...,x_i-1\}$, 所有可能的 θ 總個數為 $\frac{(x_i-1)-(x_i'+1)}{2}+1=\frac{1}{2}(x_i-x_i')$,
得 $\sum_{\theta\text{介於}x_i\text{和}x_i'\text{之間}} g_{s,i,\theta}(\mathbf{x})\cdot g_{s,i,\theta}(\mathbf{x}')=\frac{1}{2}|x_i-x_i'|\cdot(-1)=-\frac{1}{2}|x_i-x_i'|$

(2) 若 θ 不在 x_i 和 x_i' 之間的時候, $g_{s,i,\theta}(\mathbf{x})$ 和 $g_{s,i,\theta}(\mathbf{x}')$ 為同號, 則 $g_{s,i,\theta}(\mathbf{x})\cdot g_{s,i,\theta}(\mathbf{x}')=+1$, 所有可能的 θ 總個數為 $\frac{(2R-1)-(2L+1)}{2}-\frac{1}{2}|x_i-x_i'|=(R-L)-\frac{1}{2}|x_i-x_i'|$,
得 $\sum_{\theta\text{不在}x_i\text{和}x_i'\text{之間}} g_{s,i,\theta}(\mathbf{x})\cdot g_{s,i,\theta}(\mathbf{x}')=((R-L)-\frac{1}{2}|x_i-x_i'|)\cdot(+1)=(R-L)-\frac{1}{2}|x_i-x_i'|$

由 (1)(2) 得,
$$\sum_{\theta\in\{2L+1,2L+3,...,2R-1\}} g_{s,i,\theta}(\mathbf{x})\cdot g_{s,i,\theta}(\mathbf{x}')=((R-L)-\tfrac{1}{2}|x_i-x_i'|)+(-\tfrac{1}{2}|x_i-x_i'|)=(R-L)-|x_i-x_i'|$$

綜上所述,
$$K_{ds}(\mathbf{x},\mathbf{x}')=\sum_{i=1}^{d}\sum_{s\in\{-1,+1\}}(R-L)-|x_i-x_i'|=2d\cdot(R-L)-2\cdot\sum_{i=1}^{d}|x_i-x_i'|=2d(R-L)-2\|\mathbf{x}-\mathbf{x}'\|_1$$

Adaptive Boosting

11. Answer: [a]

由題意可知, $g_1=-1$, $\text{err}_{0/1}=1-0.95=0.05$。
令資料總量為 N, 根據 Lecture 208 slides 第 14 頁公式可得 $\epsilon_t=\frac{\sum_{n=1}^{N}u_n^{(t)}\llbracket y_n\neq g_t(\mathbf{x_n})\rrbracket}{\sum_{n=1}^{N}u_n^{(t)}}$, $\blacklozenge_t=\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$
在 $t=1$ 時, $u_n^{(1)}=\frac{1}{N}$, 則 $\epsilon_1=\sum_{n=1}^{N}\frac{\llbracket y_n\neq g_1(\mathbf{x_n})\rrbracket}{N}=\text{err}_{0/1}=0.05$。

因為 $u_-^{(1)}=u_+^{(1)}=\frac{1}{N}$ 且 positive sample 全部答錯/negative sample 全部答對,
得 $\frac{u_+^{(2)}}{u_-^{(2)}}=\frac{u_+^{(1)}\cdot\blacklozenge_1}{u_-^{(1)}/\blacklozenge_1}=\blacklozenge_1^2=\frac{1-\epsilon_1}{\epsilon_1}=\frac{0.95}{0.05}=19$

12. Answer: [d]

根據 Lecture 208 slides 第 14 頁公式可得 $\epsilon_t=\frac{\sum_{n=1}^{N}u_n^{(t)}\llbracket y_n\neq g_t(\mathbf{x_n})\rrbracket}{\sum_{n=1}^{N}u_n^{(t)}}$, $\blacklozenge_t=\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$

$$\begin{aligned}U_{t+1}&=\sum_{n=1}^{N}u_n^{(t+1)}=\sum_{n=1}^{N}u_n^{(t)}\llbracket y_n\neq g_t(\mathbf{x_n})\rrbracket\cdot\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}+\sum_{n=1}^{N}u_n^{(t)}\llbracket y_n=g_t(\mathbf{x_n})\rrbracket\cdot\sqrt{\frac{\epsilon_t}{1-\epsilon_t}}\\&=\sum_{n=1}^{N}u_n^{(t)}\cdot\frac{\sum_{n=1}^{N}u_n^{(t)}\llbracket y_n\neq g_t(\mathbf{x_n})\rrbracket}{\sum_{n=1}^{N}u_n^{(t)}}\cdot\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}+\sum_{n=1}^{N}u_n^{(t)}\cdot\frac{\sum_{n=1}^{N}u_n^{(t)}\llbracket y_n=g_t(\mathbf{x_n})\rrbracket}{\sum_{n=1}^{N}u_n^{(t)}}\cdot\sqrt{\frac{\epsilon_t}{1-\epsilon_t}}\\&=U_t\cdot\epsilon_t\cdot\sqrt{\frac{1-\epsilon_t}{\epsilon_t}}+U_t\cdot(1-\epsilon_t)\cdot\sqrt{\frac{\epsilon_t}{1-\epsilon_t}}=2\sqrt{\epsilon_t(1-\epsilon_t)}\cdot U_t\end{aligned}$$

因為 $0<\epsilon_t\leq\epsilon<\frac{1}{2}$, 則 $\frac{U_{t+1}}{U_t}=2\sqrt{\epsilon_t(1-\epsilon_t)}\leq2\sqrt{\epsilon(1-\epsilon)}\leq2\cdot\frac{1}{2}\exp(-2(\frac{1}{2}-\epsilon)^2)=\exp(-2(\frac{1}{2}-\epsilon)^2)$

$$E_{in}(G_T)\leq U_{t+1}=U_1\cdot\prod_{t=1}^{T}\tfrac{U_{t+1}}{U_t}\leq1\cdot\prod_{t=1}^{T}\exp(-2(\tfrac{1}{2}-\epsilon)^2)=\exp(-2T(\tfrac{1}{2}-\epsilon)^2)$$

13. Answer: [d]

因為 $|\mu_+ - \mu_-| \geq 0$, $1 \geq 1 - |\mu_+ - \mu_-|$, 當 $\mu_+ = \mu_- = 0.5$ 時, $1 - |\mu_+ - \mu_-|$ 有極大值 1, 其對應的正規化後的 impurity function 為 $\frac{1-|\mu_+-\mu_-|}{1} = 1 - |\mu_+ - \mu_-|$。

$$\begin{aligned}
1 - |\mu_+ - \mu_-| &= (\mu_+ + \mu_-) - |\mu_+ - \mu_-| = (\mu_+ + \mu_-) - \max(\mu_+ - \mu_-, \mu_- - \mu_+) \\
&= (\mu_+ + \mu_-) + \min(\mu_+ - \mu_-, \mu_- - \mu_+) \\
&= \min((\mu_+ + \mu_-) + (\mu_+ - \mu_-), (\mu_+ + \mu_-) + (\mu_- - \mu_+)) \\
&= \min(2\mu_+, 2\mu_-) = 2\min(\mu_+, \mu_-)
\end{aligned}$$

Experiment

程式碼實作細節如下, 可以透過 parser 的 −−tra_path/−−tst_path 設置訓練資料和測試資料的路徑

```
python code.py --tra_path hw6_train.dat --tst_path hw6_test.dat
```

```
import numpy as np
import argparse
import torch
import random
from joblib import Parallel, delayed

'''Define Function'''


def get_data(path):
    X = []
    for x in open(path, 'r'):
        x = x.strip().split(' ')
        X.append([float(v) for v in x])
    X = np.array(X)
    X, Y = np.array(X[:, :-1]), np.array(X[:, -1])
    return X, Y


def get_01_err(Y_pre, Y_tar):
    if len(Y_pre.shape) == 3:
        return (Y_pre != Y_tar).mean(axis=-1).mean()
    return np.mean(Y_pre != Y_tar)


class ThetaNode:
    def __init__(self, idx=None, theta=None, value=None):
        self.child_1st = None
        self.child_2nd = None
        self.idx = idx
        self.theta = theta
        self.value = value
```

```
class DecisionTree:
    def __init__(self, use_cuda=False, data_idx=None):
        self.tree = None
        self.use_cuda = use_cuda
        self.train_idx_list = data_idx

    def check_X_the_same(self, X):
        return (X != X[0, :]).sum() == 0

    def check_Y_the_same(self, Y):
        return (Y != Y[0]).sum() == 0

    def cuda(self):
        self.use_cuda = True

    def decision_stump(self, X, Y):
        X_sort, _ = torch.sort(X, dim=0)
        All_theta_feat = (X_sort[1:] + X_sort[:-1]) / 2

        X_rep = X.unsqueeze(0).repeat(All_theta_feat.shape[0], 1, 1)
        Y_rep = Y.unsqueeze(0).repeat(All_theta_feat.shape[0], 1).unsqueeze(
            0).repeat(All_theta_feat.shape[1], 1, 1)
        All_theta_feat = All_theta_feat.permute(1, 0).unsqueeze(-1)
        X_rep = X_rep.permute(2, 0, 1)

        pos_idx = X_rep > All_theta_feat
        neg_idx = X_rep <= All_theta_feat

        Gini_pos = self.Gini(pos_idx * Y_rep)
        Gini_neg = self.Gini(neg_idx * Y_rep)
        Gini_value = Gini_pos * pos_idx.sum(-1) + Gini_neg * neg_idx.sum(-1)

        min_gini_idx = (Gini_value == Gini_value.min()).nonzero(as_tuple=False)[0]
        idx = min_gini_idx[0].item()
        theta = All_theta_feat[idx, min_gini_idx[1].item()].item()

        return idx, theta

    def branching(self, X, Y):
        if self.check_Y_the_same(Y) or self.check_Y_the_same(X):
            return ThetaNode(value=Y[0].item())

        idx, theta = self.decision_stump(X, Y)
        tree_node = ThetaNode(idx, theta)

        idx_1st = X[:, idx] > theta
        idx_2nd = X[:, idx] <= theta

        tree_node.child_1st = self.branching(X[idx_1st], Y[idx_1st])
        tree_node.child_2nd = self.branching(X[idx_2nd], Y[idx_2nd])
```

```
        return tree_node

    def Gini(self, Y):
        N = (Y != 0).sum(-1)
        N[N == 0] = 1
        return 1 - (((Y == -1).sum(-1) / N)**2 + ((Y == 1).sum(-1) / N)**2)

    def fit(self, X, Y):
        with torch.no_grad():
            X, Y = torch.from_numpy(X), torch.from_numpy(Y)
            if self.use_cuda:
                X, Y = X.cuda(), Y.cuda()
            self.tree = self.branching(X, Y)

    def get_label(self, x, tree=None):
        if tree is None:
            tree = self.tree

        if tree.value is not None:
            return tree.value

        elif x[tree.idx] > tree.theta:
            return self.get_label(x, tree.child_1st)
        else:
            return self.get_label(x, tree.child_2nd)

    def predict(self, X):
        with torch.no_grad():
            return np.array([self.get_label(x, self.tree) for x in X])


class RandomForest:
    def __init__(self, n_estimators=2000, n_jobs=0, use_cuda=False):
        self.use_cuda = use_cuda
        self.n_estimators = n_estimators
        self.n_jobs = n_jobs

    def bootstrap(self, X, Y, sr=0.5):
        random_list = [random.randint(0, X.shape[0]-1)
                       for _ in range(int(sr * len(Y)))]
        return X[random_list], Y[random_list], sorted(list(set(random_list)))

    def get_tree(self, D):
        X, Y, data_idx = D
        tree = DecisionTree(use_cuda=self.use_cuda, data_idx=data_idx)
        tree.fit(X, Y)
        return tree

    def get_label(self, tree, X):
        Y = tree.predict(X)
        return Y
```

```
    def fit(self, X, Y, sr=0.5):
        if self.n_jobs != 0:
            self.tree_list = Parallel(n_jobs=self.n_jobs)(delayed(self.get_tree)(
                self.bootstrap(X, Y, sr)) for _ in range(self.n_estimators))
        else:
            self.tree_list = []
            for _ in range(self.n_estimators):
                self.tree_list.append(self.get_tree(self.bootstrap(X, Y, sr)))

    def predict(self, X, output_all=False):
        Y_pre_list = np.array([self.get_label(tree, X)
                               for tree in self.tree_list])
        if output_all:
            return Y_pre_list
        else:
            Y_pre = np.ones(X.shape[0])
            Y_pre[Y_pre_list.sum(axis=0) <= 0] = -1
            return Y_pre


def main():
    '''Parsing'''
    parser = argparse.ArgumentParser(
        description='Argument Parser for ML HW6.')

    parser.add_argument('--tra_path', default='hw6_train.dat')
    parser.add_argument('--tst_path', default='hw6_test.dat')
    args = parser.parse_args()

    # load data
    tra_X, tra_Y = get_data(args.tra_path)
    tst_X, tst_Y = get_data(args.tst_path)

    '''Answer questions'''
    print('RUNNING Q14...')
    dt = DecisionTree(use_cuda=True)
    dt.fit(tra_X, tra_Y)
    tst_Y_pre = dt.predict(tst_X)
    print('Answer of Q14 : {:.4f}\n'.format(get_01_err(tst_Y_pre, tst_Y)))

    print('RUNNING Q15...')
    rf = RandomForest(n_estimators=2000, n_jobs=8, use_cuda=True)
    rf.fit(tra_X, tra_Y)
    tst_Y_pre = rf.predict(tst_X, output_all=True)
    print('Answer of Q15 : {:.4f}\n'.format(get_01_err(tst_Y_pre, tst_Y)))

    print('RUNNING Q16...')
    tra_Y_pre = rf.predict(tra_X, output_all=False)
    print('Answer of Q16 : {:.4f}\n'.format(get_01_err(tra_Y_pre, tra_Y)))
```

```
    print('RUNNING Q17...')
    tst_Y_pre = rf.predict(tst_X, output_all=False)
    print('Answer of Q17 : {:.4f}\n'.format(get_01_err(tst_Y_pre, tst_Y)))

    print('RUNNING Q18...')
    val_Y_pre = []
    for i in range(tra_X.shape[0]):
        tree_list = []
        for tree in rf.tree_list:
            if i not in tree.train_idx_list:
                tree_list.append(tree)

        if len(tree_list) == 0:
            y_pre = -1
        else:
            y_pre_list = []
            for tree in tree_list:
                y_pre_list.append(tree.get_label(tra_X[i]))
            y_pre = 1 if sum(y_pre_list) > 0 else -1

        val_Y_pre.append(y_pre)
    val_Y_pre = np.array(val_Y_pre)
    print('Answer of Q18 : {:.4f}\n'.format(get_01_err(val_Y_pre, tra_Y)))


if __name__ == "__main__":
    main()
```

14. Ans: [c]	15. Ans: [d]	16. Ans: [a]	17. Ans: [d]	18. Ans: [b]
0.166	0.2304	0.013	0.155	0.069

Learning Comes from Feedback

19. Answer: [d]

就我所知, boosting 在台大的所有課程裡面, 目前只有機器學習技法有教, 撇除結構性很强的 raw feature data 被類神經網路主宰, 在 concrete data 上該經典方法的表現仍然有一定水準, 選這門課其中一個期許就是希望可以把 aggregation 學好, 這也是這門課裡面啓發我最多的一章, 我在 final project 裡面也有投入一些 sampling 的概念來增加 model diversity, 這章於我而言非常實用！

20. Answer: [b]

matrix factorization 這章的内容和其他機器學習課程重疊率極高, 且變化性和差異性不大, 這章的知識即使我不修這門課也可以從其他課程或是文章中學到, 且因為我不是做推薦系統或是檢索的, 這章於我而言生涯相關性是最低的, 個人私心覺得也許這章的時間甚至可以拿去補充一些深度學習理論或是主動學習的内容。