Machine Learning Homework 6

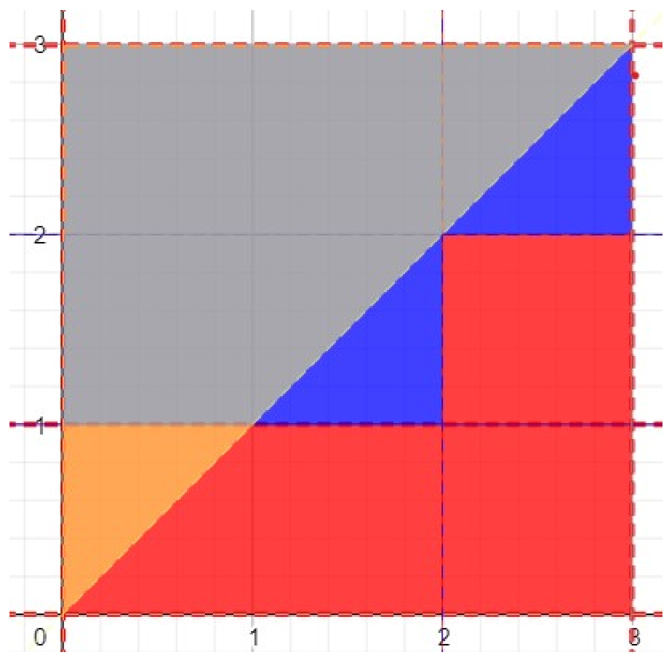
1

設一共有 N 筆資料，所有分類器猜錯的資料數總和為 $\sum_{t=1}^{11} Ne_t$，而 G 分類錯誤代表有多於一半(6個)分類器分類錯誤，因此 G 的最大分類錯誤數為 $\frac{1}{6}\sum_{t=1}^{11} Ne_t$，進一步得到上界 $E_{out}(G) = \frac{1}{6}\sum_{t=1}^{11} e_t$。

答案為 [c].

2

設x軸、y軸依序為 x_1, x_2 ，我們先用觀察得到當 $\alpha_1 = -1, \alpha_2 = 2, \alpha_3 = 2$ 時 $E_{out}(G) = \frac{3}{18}$，也就是下圖的藍色、橘色區域:



若要將左下角的錯誤去除，則必須調整 α_2，則至少右邊 $\frac{3}{18}$ 的區域會被判錯為正。

若要將正中間的錯誤去除，則必須調整 α_3，則至少左邊 $\frac{3}{18}$ 的區域會被判錯為負。

若要將左下角的錯誤去除，則必須調整 α_1，則至少左邊 $\frac{1}{18}$ 的區域會被判錯為負。

答案為 [d].

3

已知 $g_{s,i,\theta}$ 為 $+1$ 或 -1，而所有 $g_{s,i,\theta}(x)g_{s,i,\theta}(x')$ 中 s 相乘必為 1，而 $(x_i - \theta)$ 在兩者皆大於或小於 θ 時為1，對兩者異號的所有 x_i，會有 $2\|\frac{x_i - x'_i}{2}\|$個 -1， K_{ds} 的最大值為 $2 * d * \frac{2R-2L}{2}$，減去所有sign值異號的數量得到 $K_{ds} = 2d(R-L) - 2\|\frac{x_i - x'_i}{2}\|_1$，乘二是因為要減去原先誤加的數，採用norm-1是因為所有feature分開計算後加總。

答案為 [a].

4

設 $u_n^{(1)} = \frac{1}{N}$ 計算 ϵ_1 及 scaling factor s_1

$$\epsilon_1 = \frac{\sum_{n=1}^{N} u_n^{(1)} [\![y_n \neq g_1(x_n)]\!]}{\sum_{n=1}^{N} u_n^{(1)}} = \frac{(0.01N)\frac{1}{N}}{N\frac{1}{N}} = 0.01$$

$$s_1 = \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} = \sqrt{\frac{0.99}{0.01}} = \sqrt{99}$$

得知 $u_n^{(2)}$ 在 $y_n = +1$ 時為 $\frac{1}{\sqrt{99}N}$，在 $y_n = -1$ 時為 $\frac{\sqrt{99}}{N}$。

計算出答案為

$$\frac{\sum_{n:y_n>0} u_n^{(2)}}{\sum_{n:y_n<0} u_n^{(2)}} = \frac{(0.99N)\frac{1}{\sqrt{99}N}}{(0.01N)\frac{\sqrt{99}}{N}} = \frac{0.99}{0.99} = 1$$

答案為 [c].

5

上課時提過 $E_{in}(G_T)$ 不為 non-increasing，例如用程式跑 $x = [0,1,2,3,4,5], y = [+1,+1,-1,+1,-1,+1]$ 的一維二分類問題，他在 $T = 1,2,3,4$ 的 $E_{in}(G_T)$ 依序為 $[\frac{1}{3}, \frac{1}{3}, \frac{1}{6}, \frac{1}{3}]$，在 $T = 4$ 時出現了上升的現象，$E_{out}(G_T)$ 就更不用說了。

$U^T = \sum_{n=1}^{N} u_n^{(t)}$ *to* $\sum_{n=1}^{N} u_n^{(t+1)} = U^{T+1}$ 可從第六題的證明結果及 $0 \leq \epsilon_t \leq \frac{1}{2}$ 得到 $\frac{U^{T+1}}{U^T} = 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq 1$ 故為 non-decreasing，由 $0 \leq \epsilon_t \leq \frac{1}{2}$ 得到 $\diamond_t = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}} \geq 1$，故第四點為 non-increasing，第五點為 non-decreasing。故兩點符合 non-increasing 的條件。

答案為 [b].

6

由定義

$$\epsilon_t = \frac{\sum_{n=1}^{N} u_n^{(t)} [\![y_n \neq g_t(x_n)]\!]}{\sum_{n=1}^{N} u_n^{(t)}}$$

$$scaling\ \ factor = \sqrt{\frac{1-\epsilon_t}{\epsilon_t}}$$

可計算出 U_{t+1} 為

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

故可得到 $\frac{U_{t+1}}{U_t} = 2\sqrt{\epsilon_t(1-\epsilon_t)}$。

答案為 [b].

7

由 $E_{in}(G_T)$ 公式推得

$$
\begin{aligned}
E_{in}(G_T) &= \frac{1}{N}\sum_{n=1}^{N} 1[\![y_n G_T(x_n) \le 0]\!] \\
&\le \frac{1}{N}\sum_{n=1}^{N} \exp(-y_n G_T(x_n)) \\
&= \frac{1}{N}\sum_{n=1}^{N} \exp(-y_n \sum_{t=1}^{T} {}_t g_t(x_n)) \\
&= \sum_{n=1}^{N} \frac{1}{N} \prod_{t=1}^{T} \exp(y_n \alpha_t g_t(x_n)) \\
&= \sum_{n=1}^{N} u_n^{(1)} \prod_{t=1}^{T} \diamond_t^{-y_n g_t(x_n)} \\
&= \sum_{n=1}^{N} u_n^{T+1} = U^{T+1}
\end{aligned}
$$

再由 U^{T+1} 於第六題的結果加上提示的不等式推得

$$
U^{T+1} = U^1 \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \le \prod_{t=1}^{T} \exp(-2(\frac{1}{2}-\epsilon_t)^2)
$$

而當 $E_{in}(G_T) < \frac{1}{N}$ 時，代表有不到一筆資料分類錯誤，因此所有資料正確 $E_{in}(G_T) = 0$，故結合上式得到

$$
\begin{aligned}
E_{in}(G_T) \le U^{T+1} \le \prod_{t=1}^{T} \exp(-2(\frac{1}{2}-\epsilon_t)^2) &< \frac{1}{N} \\
\exp(-2T(\frac{1}{2}-\epsilon)^2) < \exp(\sum_{t=1}^{T} -2(\frac{1}{2}-\epsilon_t)^2) &< \frac{1}{N} \\
-2T(\frac{1}{2}-\epsilon)^2 &< -\ln N \\
T &> \frac{\ln N}{2(\frac{1}{2}-\epsilon)^2}
\end{aligned}
$$

推得 $T > \frac{\ln N}{2(\frac{1}{2}-\epsilon)^2}$ 時 $E_{in}(G_T) = 0$。
答案為 [b].

8

隨機抽取一個樣本不重複機率為 $\frac{1126}{1126}$，隨機抽取二個樣本不重複機率為 $\frac{1126}{1126} * \frac{1125}{1126}$，故有大於 0.5 機率存在至少兩個樣本相同的最小樣本個數 x 滿足下式

$$1 - \prod_{i=0}^{x-1} \frac{1126 - i}{1126} \geq 0.5$$

透過代入得到

$$i = 39, 1 - \prod_{i=0}^{x-1} \frac{1126 - i}{1126} \approx 0.486093$$
$$i = 40, 1 - \prod_{i=0}^{x-1} \frac{1126 - i}{1126} \approx 0.503893$$

答案為 [d].

9

對 N 筆資料隨機 sample $2N$ 筆資料存在一筆資料不被sample到的機率為 $(\frac{N-1}{N})^{2N}$，故由極限推得

$$\lim_{N \to \infty} (1 - \frac{1}{N})^{2N} = e^{-2} \approx 0.1353$$

答案為 [d].

10

由題目敘述得知所有符合 $x_1 < 0$ 的資料會對應到一個 constant value，此外符合 $x_1 \geq 0 \wedge x_2 \geq 0$ 或 $x_1 \geq 0 \wedge x_2 < 0$ 的資料會在對應到各自的 constant value。若資料集中不存在重複這三種 case 的資料就代表該資料集可被 shattered。只有 [b] 不存在重複的 case。
答案為 [b].

11, 12, 13, 14, 15, 16

```python
import numpy as np
import math
import tqdm

#========================================basic function========================================

def E_01(y, predict):
    predict = np.reshape(predict, y.shape)
    return np.sum(y != predict) / y.shape[0]

def predict(x, G, alpha):
    pred = np.zeros(x.shape[0])

    for t in range(len(G)):
        for i in range(x.shape[0]):
            pred[i] += alpha[t] * G[t].pred(x[i])

    return np.sign(pred)

#========================================AdaBoost-Stump========================================

class Stump:
    s, i, theta = 0, 0, 0

    def pred(self, x):
        return self.s * np.sign(x[self.i] - self.theta)

def DecisionStump(x, y, weight):
    stump = Stump()
    minErr = 1

    for i in range(x.shape[1]):
        x_temp = [x[j][i] for j in range(x.shape[0])]
        xyw = np.hstack((np.reshape(x_temp, y.shape), y, np.reshape(weight, y.shape)))
        xyw_sort = xyw[xyw[:, 0].argsort()]

        negativeSum, positiveSum = [0], [0]
        for j in range(xyw_sort.shape[0]):
            if xyw_sort[j][1] == 1:
                positiveSum.append(positiveSum[-1])
                negativeSum.append(negativeSum[-1] + xyw_sort[j][2])
            elif xyw_sort[j][1] == -1:
                positiveSum.append(positiveSum[-1] + xyw_sort[j][2])
                negativeSum.append(negativeSum[-1])

        for j in range(len(negativeSum) - 1):
            npErr = negativeSum[j] + positiveSum[-1] - positiveSum[j]
            pnErr = positiveSum[j] + negativeSum[-1] - negativeSum[j]

            if min(npErr, pnErr) < minErr:
                minErr = min(npErr, pnErr)
                if npErr < pnErr:
                    stump.s = 1
                else:
                    stump.s = -1

                stump.i = i

                if j == 0:
                    stump.theta = xyw_sort[0][0] - 1
                else:
                    stump.theta = (xyw_sort[j - 1][0] + xyw_sort[j][0]) / 2

    return stump

def epsilonCal(x, y, g, weight):
    errSum = 0

    for i in range(x.shape[0]):
        if g.pred(x[i]) != y[i]:
            errSum += weight[i]

    return errSum / np.sum(weight)
```

```python
def weightUpdate(x, y, g, weight, scal):
    for i in range(weight.shape[0]):
        if g.pred(x[i]) == y[i]:
            weight[i] /= scal
        elif g.pred(x[i]) != y[i]:
            weight[i] *= scal

    return weight

def AdaptiveBoosting(x, y, T = 500):
    weight = np.ones(x.shape[0]) / x.shape[0]
    G = []
    alpha = []

    for t in tqdm.trange(T):
        g = DecisionStump(x, y, weight)
        #print('g', g.s, g.i, g.theta)
        eps = epsilonCal(x, y, g, weight)
        scal = math.sqrt((1 - eps) / eps)
        #print('eps', eps, 'scal', scal)
        weight = weightUpdate(x, y, g, weight, scal)
        #print('weight', weight[:10])

        G.append(g)
        alpha.append(math.log(scal))
        #print('alpha', alpha[-1])

    return G, alpha

#==============================================================================================

def main():
    trainData = np.loadtxt('hw6_train.dat')
    testData = np.loadtxt('hw6_test.dat')
    x_train, y_train = np.hsplit(trainData, [-1])
    x_test, y_test = np.hsplit(testData, [-1])

    p11, p12, p13, p14, p15, p16 = True, True, True, True, True, True

    G, alpha = AdaptiveBoosting(x_train, y_train)
    pred = predict(x_train, G, alpha)
    print(E_01(y_train, pred))

    if p11:
        pred = predict(x_train, G[:1], alpha[:1])
        print('Problem 11:', 'Ein(g_1) =', E_01(y_train, pred))

    if p12:
        maxEin = 0
        for i in range(len(G)):
            pred = predict(x_train, G[i : i+1], alpha[i : i+1])
            if E_01(y_train, pred) > maxEin:
                maxEin = E_01(y_train, pred)
        print('Problem 12:', 'max Ein(g_t) =', maxEin)

    if p13:
        minT = 0
        pred = np.zeros(x_train.shape[0])

        for t in range(len(G)):
            for i in range(x_train.shape[0]):
                pred[i] += alpha[t] * G[t].pred(x_train[i])
                if E_01(y_train, np.sign(pred)) <= 0.05:
                    minT = t + 1
                    break

            if minT != 0:
                break

        print('Problem 13:', 'min Ein(G_t) < 0.05, t =', minT)

    if p14:
        pred = predict(x_test, G[:1], alpha[:1])
        print('Problem 14:', 'Eout(g_1) =', E_01(y_test, pred))

    if p15:
        pred = np.zeros(x_test.shape[0])

        for t in range(len(G)):
            for i in range(x_test.shape[0]):
                pred[i] += G[t].pred(x_test[i])

        print('Problem 15:', 'Eout(G_uniform) =', E_01(y_test, np.sign(pred)))

    if p16:
        pred = predict(x_test, G, alpha)
        print('Problem 16:', 'Eout(G_500) =', E_01(y_test, pred))

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

答案依序為[c], [e], [d], [b], [a], [b].