Machine Learning Homework 6

資工碩一 r09922055 陳柏妤

1. **(b)**

$$\delta_j^{(l)} = \sum_{k=1}^{d^{(l+1)}} (\delta_k^{(l+1)})(w_{jk}^{(l+1)})(tanh'(s_j^{(l)}))$$

· 當 $l = 2$ 時，$j \in \{1, 2,..., d^{(2)}\}$，此時 $k \in \{1, 2,..., d^{(3)}\}$

$\Rightarrow$ 要更新完第 2 層的 δ 需要 $d^{(2)} \cdot d^{(3)} = 6$ 次的 $(\delta_k^{(l+1)})(w_{jk}^{(l+1)})$

· 當 $l = 1$ 時，$j \in \{1, 2,..., d^{(1)}\}$，此時 $k \in \{1, 2,..., d^{(2)}\}$

$\Rightarrow$ 要更新完第 1 層的 δ 需要 $d^{(1)} \cdot d^{(2)} = 30$ 次的 $(\delta_k^{(l+1)})(w_{jk}^{(l+1)})$

$6 + 30 = 36$

2. **(d)**

用 code 計算

```
limit = 50
possible_ds = []
def permute(d, ds, layer):
    if limit==d:
        possible_ds.append(ds)
        return
    elif limit<d:
        return
    else:
        ds_PlusOneNode = ds.copy()
        ds_PlusOneNode[-1] += 1
        d += 1
        permute(d, ds_PlusOneNode, layer)
        d -= 1

        ds_PlusOneLayer = ds.copy()
        d += 4
        ds_PlusOneLayer.append(4)
        permute(d, ds_PlusOneLayer, layer+1)

def compute_num_node():
    permute(2, [2], 0)
    max_num_node = 0
    for i in range(len(possible_ds)):
        num_node = 0
        for j in range(len(possible_ds[i])-1):
            num_node += (possible_ds[i][j] * (possible_ds[i][j+1]-1))
        num_node += (20*(possible_ds[i][0]-1))
        num_node += (possible_ds[i][-1]*3)
        if num_node>max_num_node:
            max_num_node = num_node
    return(max_num_node)
```

```
print("max_num_node: ", compute_num_node())
```

```
max_num_node:  1219
```

3. **(d)**

・整理 error function

$$err(x,y) = -\sum_{k=1}^{K} v_k ln(q_k) = -\sum_{k=1}^{K} v_k ln(\frac{exp(s_k^{(L)})}{\sum_{i=1}^{K} exp(s_i^{(L)})})$$

$$= \sum_{k=1}^{K} v_k(ln(\sum_{i=1}^{K} exp(s_i^{(L)})) - ln(exp(s_k^{(L)})))$$

$$= ln(\sum_{i=1}^{K} exp(s_i^{(L)})) - \sum_{k=1}^{K} v_k(ln(exp(s_k^{(L)})))$$ (前項與 k 無關，能移到 $\sum_{k=1}^{K}$ 左邊)

・error function 微分

$$\frac{\partial[ln(\sum_{i=1}^{K} exp(s_i^{(L)})) - \sum_{k=1}^{K} v_k(ln(exp(s_k^{(L)})))]}{\partial s_k^{(L)}}$$

$$= [\frac{1}{\sum_{i=1}^{K} exp(s_i^{(L)})} \cdot exp(s_k^{(L)})] - [v_k \cdot \frac{1}{exp(s_k^{(L)})} \cdot exp(s_k^{(L)})]$$

$$= q_k - v_k$$

4. **(a)**

・forward

因為每個 $w^{(l)}$ 的初始值為 $0 \Rightarrow$ forward 時算出來的所有 $s_i^{(l)}$ 和 $x_i^{(l)}$ 都是 0 ($l \in \{1, 2\}$)

・backward

by Lecture 212 P.14，$\delta_1^{(L)} = -2(y_n - s_1^{(L)}) = -2(1-0) = -2$

by Lecture 212 P.15，

$$\delta_j^{(l)} = \sum_{k=1}^{d^{(l+1)}} (\delta_k^{(l+1)})(w_{jk}^{(l+1)})(tanh'(s_j^{(l)}))$$

$$\Rightarrow \delta_j^{(1)} = \sum_{k=1}^{d^{(2)}} (\delta_k^{(2)})(w_{jk}^{(2)})(tanh'(s_j^{(1)})) = \sum_{k=1}^{d^{(2)}} (-2) \cdot 0 \cdot tanh'(0) = 0$$

$(j \in \{1, 2, ..., d^{(1)}\})$

・Gradient descent

$w_{01}^{(1)} \leftarrow w_{01}^{(1)} - \eta \cdot x_0^{(0)} \cdot \delta_1^{(1)} = w_{01}^{(1)} - 1 \cdot 1 \cdot 0 = w_{01}^{(1)}$ (更新後 $w_{01}^{(1)}$ 沒有變化)

* 因為 backward 時每個 $\delta_j^{(1)}$ 都是 $0 \Rightarrow$ 更新後第一層的 $w^{(1)}$ 皆不變

* 因為 forward 時每個 $x_i^{(1)}$ 都是零

$\Rightarrow$ 更新後第 2 層的 $w_{i1}^{(2)} \leftarrow w_{i1}^{(2)} - \eta \cdot x_i^{(1)} \cdot \delta_1^{(2)} = w_{i1}^{(2)} - 1 \cdot 0 \cdot -2 = w_{i1}^{(2)}$ (不變)

在一輪的更新後 w 皆不變，重複三輪後 $w_{01}^{(1)} = 0$

5. **(e)**

$V = [2 \quad 2 \quad \dots \quad 2]_{1\times N}$，$W = [w_1 \quad w_2 \quad \cdots \quad w_m \quad \cdots \quad w_M]_{1\times M}$

$square\ error = \sum_m (\sum_{(x_n, r_{nm}) \in D_m} (r_{nm} - w_m^T v_n)^2)$

by Lecture 215 P.10, step 2.1 update w_m by m^{th} movie linear regression on $\{(v_n, r_{nm})\}$

$square\ error\ of\ m^{th}\ movie = \sum_{(x_n, r_{nm}) \in D_m} (r_{nm} - w_m^T v_n)^2 = \sum_{(x_n, r_{nm}) \in D_m} (r_{nm} - w_m^T 2)^2$

by linear regression, $w_m = (VV^T)^{-1} V r_{nm}$ 時 $error$ 會最小

$$w_m = ([2 \quad 2 \quad \dots \quad 2]_{1\times N} \begin{bmatrix} 2 \\ 2 \\ \vdots \\ 2 \end{bmatrix}_{N\times 1})^{-1} [2 \quad 2 \quad \dots \quad 2]_{1\times N} \begin{bmatrix} r_{1m} \\ r_{2m} \\ \vdots \\ r_{nm} \end{bmatrix}_{N\times 1}$$

$$= \frac{1}{4N}(2r_{1m} + 2r_{2m} + \dots + 2r_{nm}) = \frac{1}{2} \frac{r_{1m} + r_{2m} + \dots + r_{nm}}{N}$$

⋆實際情況下有些 r_{nm} 會沒有值，因此只會考慮有評分資料的點

$w_m =$ （half the average rating of the m^{th} movie）

6. **(b)**

$err = (r_{nm} - w_m^T v_n - a_m - b_n)^2$

$\nabla a_m = -2(r_{nm} - w_m^T v_n - a_m - b_n)$

$a_m \leftarrow a_m + \frac{\eta}{2}(-\nabla a_m) = a_m + \frac{\eta}{2}(2(r_{nm} - w_m^T v_n - a_m - b_n))$

$= a_m + \eta(r_{nm} - w_m^T v_n - a_m - b_n) = (1-\eta)a_m + (r_{nm} - w_m^T v_n - b_n)$

7. **(d)**

G 預測錯 $\Rightarrow g_1, g_2, g_3$ 有兩個預測錯一個預測對，或是三個全部預測錯

・三個全部預測錯的 $E_{out}(G)$ 最大值：

$max\ E_{out}(all\ wrong) = min(E_{out}(g_1), E_{out}(g_2), E_{out}(g_3))$

(a) 0.04 (b) 0.04 (c) 0.04 (d) 0.08 (e) 0.04

・有兩個預測錯、一個預測對的 $E_{out}(G)$ 最大值：

(a) 0.16，當 g_2 預測錯的點也是 g_3 預測錯的點時

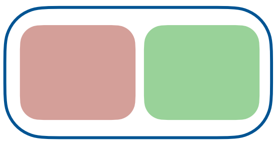
(b) 0.04+0.08 = 0.12 (c) 0.06+0.04 = 0.1 (d) 0.16+0.08 = 0.24 (e) 0.04+0.06 = 0.1

這四個最大值發生在下圖的情況下：

藍框：g_3 預測錯的點

紅色塊：g_1 和 g_3 預測錯，但 g_2 預測對的點

綠色塊：g_2 和 g_3 預測錯，但 g_1 預測對的點



$max\ E_{out}(G) = min[E_{out}(g_1), E_{out}(g_3)] + min[E_{out}(g_2), E_{out}(g_3)]$

5 個選項中唯一 $E_{out}(G)$ 有可能超過 0.2 的只有 (d)

8. **(c)**

$E_{out}(G)$ = (3 個預測錯的機率) + (4 個預測錯的機率) + （5 個預測錯的機率）

$$= C_3^5 \cdot (0.4)^3 \cdot (0.6)^2 + C_4^5 \cdot (0.4)^4 \cdot (0.6)^1 + C_5^5 \cdot (0.4)^5$$

$$= 0.2304 + 0.0768 + 0.01024 = 0.31744$$

9. **(b)**

$$probability\ of\ not\ sampled = \lim_{N\to\infty} (1 - \frac{1}{N})^{\frac{N}{2}}$$

by Lecture 210 P.8, $\lim_{N\to\infty} (1 - \frac{1}{N})^N = \lim_{N\to\infty} \frac{1}{(1 + \frac{1}{N-1})^N} \approx \frac{1}{e}$

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

$$(\frac{1}{e})^{\frac{1}{2}} \approx 0.607$$

10. **(e)**

$$K_{ds}(x, x') = (\phi_{ds}(x))^T(\phi_{ds}(x')) = \sum_{i=1}^{d} \sum_{s} \sum_{\theta} s \cdot sign(x_i - \theta) \cdot s \cdot sign(x_i' - \theta)$$

$$= \sum_{i=1}^{d} \sum_{s} \sum_{\theta} s^2 \cdot sign(x_i - \theta) \cdot sign(x_i' - \theta)$$

$$= \sum_{i=1}^{d} \sum_{\theta} \underbrace{1 \cdot sign(x_i - \theta) \cdot sign(x_i' - \theta)}_{when\ s = -1} + \underbrace{1 \cdot sign(x_i - \theta) \cdot sign(x_i' - \theta)}_{when\ s = +1}$$

$$= \sum_{i=1}^{d} \sum_{\theta} 2 \cdot sign(x_i - \theta) \cdot sign(x_i' - \theta) = \sum_{i=1}^{d} 2 \cdot [\frac{2R - 2L}{2} - 2 \cdot \frac{|x_i - x_i'|}{2}]$$

＊解釋：

如果不管 θ 是多少 $sign(x_i - \theta) \cdot sign(x_i' - \theta)$ 都是 +1，則 $\sum_{\theta} sign(x_i - \theta) \cdot sign(x_i' - \theta)$

的值會是 $\frac{2R - 2L}{2}$ （θ 是奇數所以要除 2）。然而事實上當 θ 在 x_i 和 x_i' 之間的時候，

$sign(x_i - \theta)$和 $sign(x_i' - \theta)$ 相乘會是 -1，所以要再把這個區間的值扣掉（前面要乘 2 是因為從 +1 變成 -1 會少 2）

$$= \sum_{i=1}^{d} 2 \cdot [(R - L) - |x_i - x_i'|] = \sum_{i=1}^{d} (2 \cdot (R - L)) - \sum_{i=1}^{d} (2 \cdot |x_i - x_i'|)$$

$$= 2d(R - L) - 2 \cdot \|x - x'\|_1$$

11. **(a)**

by Lecture 208 P.17, $initial\ u^{(1)} = [\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N}]$

$$\epsilon_1 = \frac{\sum_{n=1}^{N} u_n^{(1)}[y_n \neq g_1(x_n)]}{\sum_{n=1}^{N} u_n^{(1)}} = 0.05 \Rightarrow \blacklozenge_1 = \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} = \sqrt{\frac{0.95}{0.05}}$$

$$weight\ of\ incorrect\ positive\ example\ u_+^{(2)} \leftarrow u_+^{(1)} \cdot \blacklozenge_1 = \frac{1}{N} \cdot \sqrt{\frac{0.95}{0.05}}$$

$$weight\ of\ correct\ negative\ example\ u_-^{(2)} \leftarrow u_-^{(1)} / \blacklozenge_1 = \frac{1}{N} \cdot \sqrt{\frac{0.05}{0.95}}$$

$$\frac{u_+^{(2)}}{u_-^{(2)}} = \frac{\sqrt{\frac{0.95}{0.05}}}{\sqrt{\frac{0.05}{0.95}}} = \frac{0.95}{0.05} = 19$$

12. **(d)**

$$U_{T+1} = \sum_{n=1}^{N} u_n^{(T+1)} = \sum_{n=1}^{N} (u_n^{(T)} \cdot \blacklozenge_T [y_n \neq g_T(x_n)]) + \sum_{n=1}^{N} (u_n^{(T)} \cdot \frac{1}{\blacklozenge_T} [y_n = g_T(x_n)])$$

帶入 $\epsilon_T = \frac{\sum_{n=1}^{N} u_n^{(T)} \cdot [y_n \neq g_T(x_n)]}{\sum_{n=1}^{N} u_n^{(T)}}$; $(1-\epsilon_T) = \frac{\sum_{n=1}^{N} u_n^{(T)} \cdot [y_n = g_T(x_n)]}{\sum_{n=1}^{N} u_n^{(T)}}$

$$= \blacklozenge_T \cdot \epsilon_T \cdot \sum_{n=1}^{N} u_n^{(T)} + \frac{1}{\blacklozenge_T} \cdot (1-\epsilon_T) \cdot \sum_{n=1}^{N} u_n^{(T)}$$

帶入 $\blacklozenge_T = \sqrt{\frac{1-\epsilon_T}{\epsilon_T}}$

$$= \sqrt{\epsilon_T(1-\epsilon_T)} \cdot \sum_{n=1}^{N} u_n^{(T)} + \sqrt{\epsilon_T(1-\epsilon_T)} \cdot \sum_{n=1}^{N} u_n^{(T)}$$

$$= 2\sqrt{\epsilon_T(1-\epsilon_T)} \cdot \sum_{n=1}^{N} u_n^{(T)} = 2\sqrt{\epsilon_T(1-\epsilon_T)} \cdot U_T = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \cdot U_1$$

又 $U_1 = \sum_{n=1}^{N} u_n^{(1)} = \sum_{n=1}^{N} \frac{1}{N} = 1$

$$E_{in}(G_T) \leq U_{T+1} = \prod_{t=1}^{T} 2\sqrt{\epsilon_t(1-\epsilon_t)} \leq \prod_{t=1}^{T} 2 \cdot \frac{1}{2} exp(-2(\frac{1}{2}-\epsilon)^2) \quad \text{(by hint)}$$

$$= \prod_{t=1}^{T} exp(-2(\frac{1}{2}-\epsilon)^2) = exp(\sum_{t=1}^{T} -2(\frac{1}{2}-\epsilon)^2) \quad \text{(相乘就是指數相加)}$$

$$= exp(-2T(\frac{1}{2}-\epsilon)^2)$$

13. **(d)**

(a) $1-\mu_+^2-\mu_-^2=1-\mu_+^2-(1-\mu_+)^2=-2\mu_+^2+2\mu_+$

$$\frac{\partial(-2\mu_+^2+2\mu_+)}{\partial\mu_+}=-4\mu_++2$$

上式最大值落在 $\mu_+=0.5$ 時，此時最大值為 0.5

$$normalized\ impurity\ function=\frac{1-\mu_+^2-\mu_-^2}{0.5}$$

(b) $\mu_+(1-(\mu_+-\mu_-))^2+\mu_-(-1-(\mu_+-\mu_-))^2=-4(\mu_+^2-\mu_+)$

$$\frac{\partial-4(\mu_+^2-\mu_+)}{\partial\mu_+}=-8\mu_++4$$

上式最大值落在 $\mu_+=0.5$ 時，此時最大值為 1

$$normalized\ impurity\ function=\mu_+(1-(\mu_+-\mu_-))^2+\mu_-(-1-(\mu_+-\mu_-))^2$$

(c) $-\mu_+ln\mu_+-\mu_-ln\mu_-$

上式最大值落在 $\mu_+=0.5$ 時，此時最大值為 $-ln0.5$（by computer）

$$normalized\ impurity\ function=\frac{-\mu_+ln\mu_+-\mu_-ln\mu_-}{-ln0.5}$$

(d) $1-|\mu_+-\mu_-|=1-|2\mu_+-1|$

上式最大值落在 $2\mu_+-1=0\Rightarrow\mu_+=0.5$ 時，此時最大值為 1

$$normalized\ impurity\ function=1-|\mu_+-\mu_-|=1-|2\mu_+-1|$$

・當 $\mu_+<0.5$ 時：

$$normalized\ impurity\ function=1+(2\mu_+-1)=2\mu_+=2min(\mu_+,\mu_-)$$

・當 $\mu_+>0.5$ 時：

$$normalized\ impurity\ function=1-(2\mu_+-1)=2\mu_-=2min(\mu_+,\mu_-)$$

14. **(c)** 0.18 Eout: 0.16600000000000004
15. **(d)** 0.23 Average Eout: 0.24629232039636662
16. **(a)** 0.01 G_Ein: 0.015
17. **(d)** 0.16 G_Eout: 0.171
18. **(b)** 0.07 Eoob: 0.078

19. **(d)**

上課前的那段辨識蘋果的例子很有趣，而且印象深刻、看過就不會忘。也覺得 adaboost 演算法想法上很厲害。

20. **(a)**

數學很多很麻煩 X D

In [2]:
```
import numpy as np
import math
import sys
import os
```

In [3]:
```
def readfile(filename):
    X = []
    Y = []
    for lines in open(filename).readlines():
        temp = lines.strip().split()
        x = []
        for feature in temp[:-1]:
            x.append(float(feature))
        X.append(x)
        Y.append(int(float(temp[-1])))
    return np.asarray(X), np.asarray(Y)
```

In [4]:
```
def Gini_impurity(x, y, feature, theta, num_l, num_r):
    N = np.shape(x)[0]
    lp, ln, rp, rn = 0, 0, 0, 0
    for i in range(N):
        if x[:, feature][i] > theta:
            if y[i]==1:
                rp += 1
            rn = num_r-rp
        else:
            if y[i]==1:
                lp += 1
            ln = num_l-lp
    l_gini = 1-((lp/num_l)**2)-((ln/num_l)**2)
    r_gini = 1-((rp/num_r)**2)-((rn/num_r)**2)
    impurity = num_l*l_gini + num_r*r_gini
    return impurity

def select_theta(x, y, feature):
    v = sorted(set(x[:, feature]))
    values = np.array(v)
    N = np.shape(x)[0]
    min_impurity, best_theta = 10000000, None
    left = 0
    right = len(values)
    for i in range(len(values)-1):
        theta = (values[i]+values[i+1]) / 2
        left += 1
        right -= 1
        impurity = Gini_impurity(x, y, feature, theta, left, right)
        if impurity<min_impurity:
            min_impurity, best_theta = impurity, theta
    return min_impurity, best_theta

def select_feature(x, y):
    min_impurity, best_theta, best_feature = 10000000, None, None
    for i in range(10):
        impurity, theta = select_theta(x, y, i)
        if impurity<min_impurity:
            min_impurity, best_theta, best_feature = impurity, thet
a, i
    return best_theta, best_feature
```

In [5]:
```
class Node:
    def __init__(self, theta, feature):
        self.left = None
        self.right = None
        self.theta = theta
        self.feature = feature
        self.is_leaf = False
        self.predict = None
def binary_tree(x, y):
    y_list = y.tolist()
    p = y_list.count(1)
    n = y_list.count(-1)
    ## terminate => return g(t)
    if (p==0) or (n==0):
        leaf = Node(None,None)
        leaf.is_leaf = True
        if p>0: leaf.predict =  1
        else:   leaf.predict = -1
        return leaf
    elif (x!=x[0]).sum()==0:
        leaf = Node(None,None)
        leaf.is_leaf = True
        leaf.predict = -1
        return leaf
    ## no terminate
    else:
        ## learn branching criteria
        theta, feature = select_feature(x, y)
        ## split D to 2 parts = {X[:, feature]<=theta}{X[:, feature
]>theta}
        x1, y1, x2, y2 = [], [], [], []
        N = np.shape(x)[0]
        split = np.where(x[:, feature] > theta, 1, -1)
        for i in range(N):
            if split[i]==-1:
                x1.append(x[i])
                y1.append(y[i])
            elif split[i]==1:
                x2.append(x[i])
                y2.append(y[i])
        ## build two sub-tree
        if(len(y1)==0)or(len(y2)==0):
            split_list = split.tolist()

        tree = Node(theta, feature)
        tree.left  = binary_tree(np.asarray(x1), np.asarray(y1))
        tree.right = binary_tree(np.asarray(x2), np.asarray(y2))
        ## return tree
        return tree
```

In [6]:
```
def get_predict_label(node, xn):
    if node.is_leaf:
        return node.predict
    if xn[node.feature] <= node.theta:
        return get_predict_label(node.left, xn)
    elif xn[node.feature] > node.theta:
        return get_predict_label(node.right, xn)
def predict(x, y, root):
    N, correct = np.shape(x)[0], 0
    pre = []
    for i in range(N):
        predict_label = get_predict_label(root, x[i])
        pre.append(predict_label)
        if predict_label==y[i]:
            correct += 1
    return pre, round(correct/N, 3)
```

In [7]:
```
X, Y = readfile("hw6_train.dat")
Xt, Yt = readfile("hw6_test.dat")
```

In [9]:
```
## 14.
root = binary_tree(X, Y)
Train_Pre, Train_Acc = predict(X, Y, root)
Test_Pre, Test_Acc = predict(Xt, Yt, root)
print("14. Eout: ", 1-Test_Acc)
```

```
14. Eout:  0.16600000000000004
```

In [12]:
```
T = sys.argv[1]
idx = np.random.randint(1000, size=500)
Xb = X[idx, :]
Yb = Y[idx]
root = binary_tree(Xb, Yb)
np.save('Choose_points-'+str(T), idx)
np.save('Train_Pre-'+str(T), Train_Pre)
np.save('Train_Acc-'+str(T), Train_Acc)
np.save('Test_Pre-'+str(T), Test_Pre)
np.save('Test_Acc-'+str(T), Test_Acc)
```

In [15]:
```
## 15
test_acc = []
for i in range(4):
    for j in range(400):
        if os.path.isfile(str(i) + "/Test_Acc-" + str(j) + ".npy"):
            test_acc.append(float(np.load(str(i) + "/Test_Acc-" + s
tr(j) + ".npy")))
print('15. Average Eout:', 1-np.mean(test_acc))
```

```
15. Average Eout: 0.24629232039636662
```

In [18]:
```
## 16 17
G_train = np.zeros(1000)
G_test = np.zeros(1000)
for i in range(4):
    for j in range(400):
        if  os.path.isfile(str(i) + "/Train_Pre-" + str(j) + ".npy"
):
            Train_Pre  = np.load(str(i) + "/Train_Pre-" + str(j) +
".npy")
            G_train += Train_Pre
        if  os.path.isfile(str(i) + "/Test_Pre-" + str(j) + ".npy")
:
            Test_Pre  = np.load(str(i) + "/Test_Pre-" + str(j) + ".
npy")
            G_test += Test_Pre
G_Ein, G_Eout = 0, 0
for i in range(1000):
    if np.sign(G_train[i])!=Y[i]:
        G_Ein += 1
    if np.sign(G_test[i])!=Yt[i]:
        G_Eout += 1
print('16. G_Ein:', G_Ein/1000)
print('17. G_Eout:', G_Eout/1000)
```

```
16. G_Ein: 0.015
17. G_Eout: 0.171
```

In [27]:
```
## 18
OOB = np.zeros(1000)
for i in range(4):
    for j in range(400):
        if  os.path.isfile(str(i) + "/Choose_points-" + str(j) + ".
npy"):
            Choose_points = np.load(str(i) + "/Choose_points-" + st
r(j) + ".npy")
        if  os.path.isfile(str(i) + "/Train_Pre-" + str(j) + ".npy"
):
            Train_Pre = np.load(str(i) + "/Train_Pre-" + str(j) + "
.npy")
        for k in range(1000):
            if k in Choose_points:
                continue
            else:
                OOB[k] += Train_Pre[k]
Eoob = 0
for i in range(1000):
    if np.sign(OOB[i])!=Y[i]:
        Eoob += 1
print('18. Eoob:', Eoob/1000)
```

```
18. Eoob: 0.078
```