ML HW3 R10922H16 蔡家豪

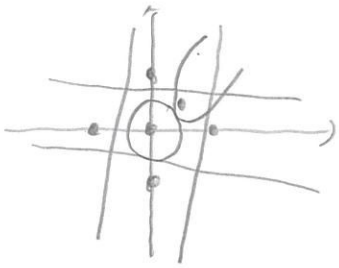
1. total number of K-class ovo $\binom{K}{2} = \frac{K!}{(K-2)!2!} = \frac{K(K-1)}{2}$

every classfier will have $2 \times \frac{N}{K}$ samples, cpu time $2 \times \frac{N}{K} \times a$

$\Rightarrow$ total cpu time $= (K \times (K-1)/2) \times (2 \times \frac{N}{K} \times a)$

$= a(K-1) \times N$ (b)

2.



the input can be shattered like figure.

[e]

3.

$x_1=(0,0)$, $y_1=-1$ $x_2=(4,0)$, $y_2=+1$ $x_3=(-4,0)$, $y_3=+1$ $x_4=(0,2)$, $y_4=-1$, $x_5=(0,-2)$, $y_5=-1$

$w_1=(-1,0,0,0.5,0,-0.5)$ $w_2=(-1,0,0,-0.5,0,0.5)$ $w_3=(-2,0,0,1,0,1)$ $w_4=(-1,0,0,0.2,0,0.1)$

$\Rightarrow h_1(x)=\text{sign}(-1+0.5x_1^2-0.5x_2^2)$ $h_2(x)=\text{sign}(-1-0.5x_1^2+0.5x_2^2)$ $h_3(x)=(2+x_1^2+x_2^2)$ $h_4(x)=(-1+0.2x_1^2+0.1x_2^2)$

$h_1(x_1)=-1=y_1$ $h_2(x_1)=-1=y_1$ $h_3(x_1)=-1=y_1$ $h_4(x_1)=-1=y_1$

$h_1(x_2)=+1=y_2$ $h_2(x_2)=-1\neq y_2$ $h_3(x_2)=+1=y_2$ $h_4(x_2)=+1=y_2$

$h_1(x_3)=+1=y_3$ $h_3(x_3)=+1=y_3$ $h_4(x_3)=+1=y_3$

$h_1(x_4)=-1=y_4$ $h_3(x_4)=+1\neq y_4$ $h_4(x_4)=-1=y_4$

w_1 & w_4 can separate

$\Rightarrow$ [c] 2

4.

$$\phi(X) = \begin{bmatrix} \Gamma x_1 & \cdots & \Gamma x_n \\ \vdots & & \end{bmatrix}^{T}$$

$$\Rightarrow \begin{bmatrix} \Gamma x_1 \\ \vdots \\ \Gamma x_n \end{bmatrix} \Rightarrow X\Gamma^T$$

$$\nabla E_{in}(\tilde{w}) = \frac{\partial\left(\frac{1}{N}\left(\tilde{w}^T \phi(x)^T \phi(x)\tilde{w} - 2\tilde{w}^T \phi(x)^T y + y^T y\right)\right)}{\partial w}$$

$$\Rightarrow \nabla E_{in}(\tilde{w}) = 0 = \frac{2}{N}\left(\left((X\Gamma^T)^T (X\Gamma^T)\right)\tilde{w} - (X\Gamma^T)^T y\right)$$

$$\Rightarrow \tilde{w} = \left((X\Gamma^T)^T (X\Gamma^T)\right)^{-1} (X\Gamma^T)^T y = (\Gamma X^T X \Gamma^T)^{-1} \Gamma X^T y$$

$$= (\Gamma^T)^{-1} X^{-1} (X^T)^{-1} \Gamma^{-1} \Gamma X^T y$$

$$= (\Gamma^T)^{-1} X^{-1} (X^T)^{-1} X^T y \qquad W_{lin} = (X^T X)^{-1} X^T y$$

$$\nabla E_{in}(W_{lin}) = 0 = \frac{2}{N}(X^T X W_{lin} - X^T y)$$

$$\Rightarrow W_{lin} = \Gamma^T \tilde{w} \quad \#$$

$$E_{in}(W_{lin}) = \frac{1}{N}\| X\Gamma^T \tilde{w} - y \|^2 = E_{in}(\tilde{w})$$

Ans [b] #

5.

$$m_H(g) = d(2N-2) + 2 \leq 2Nd$$

$$\Rightarrow 2^N \leq 2Nd$$

$$\Rightarrow 2^{N-1} \leq Nd$$

take $\log_2$

$$\Rightarrow N-1 \leq \log_2 N + \log_2 d \leq \frac{N}{2} + \log_2 d \quad \because \log_2 d \leq \frac{d}{2}$$

$\times 2$

$$\Rightarrow 2N - 2 \leq N + 2\log_2 d$$

$$\Rightarrow N \leq 2\log_2 d + 2 = 2(\log_2 d + 1)$$

[b]

6.

The c statement is wrong.

if we have $g(x) = w^T \phi(x_n) = w^T z_n = y_n$

$\Rightarrow$ $g(z_{x_n}) = y_n$ ✓

TC) ✓

7.

$x_1 = 2, y_1 = 1$

$x_2 = 3, y_2 = 0$

$x_3 = -2, y_3 = 2$

$h(x) = w_0 + w_1 x$

$E_{in} = \frac{1}{N} \sum_{i=1}^{N} \| h(x_i) - y_i \|^2$

$$E_{in}(w) = \frac{1}{3}\left[(w_0 + 2w_1 - 1)^2 + (w_0 + 3w_1)^2 + (w_0 - 2w_1 - 2)^2 \right]$$

$$= \frac{1}{3}(3w_0^2 + 17w_1^2 + 6w_0w_1 - 6w_0 + 4w_1 + 5)$$

$$E_{aug}(w) = \frac{1}{3}(3w_0^2 + 17w_1^2 + 6w_0w_1 - 6w_0 + 4w_1 + 5) + (|w_0| + |w_1|)$$

$$\frac{\partial E_{aug}(w)}{\partial w_1} = \frac{1}{3}(6w_0 + 6w_1 - 6) + \text{sign}(w_0) = 0$$

$$\frac{\partial E_{aug}(w)}{\partial w_2} = \frac{1}{3}(34w_1 + 6w_0 + 4) + \text{sign}(w_1) = 0$$

$\Rightarrow$
$$\begin{cases} 6w_0 + 6w_1 - 6 + 3\,\text{sign}(w_0) = 0 \cdots (1) \Rightarrow 6w_0 = -6w_1 + 6 - 3\,\text{sign}(w_0) \\ 34w_1 + 6w_0 + 4 + 3\,\text{sign}(w_1) = 0 \cdots (2) \end{cases}$$

$\Rightarrow$ $34w_1 - 6w_1 + 6 - 3\,\text{sign}(w_0) + 3\,\text{sign}(w_1) + 4 = 0$

$\Rightarrow$ $28w_1 = 3\,\text{sign}(w_0) - 3\,\text{sign}(w_1) - 10$

if $w_1 > 0, w_0 < 0 \Rightarrow 28w_1 = -16$ ✗ contradiction

$w_1 > 0, w_0 > 0 \Rightarrow 28w_1 = -10$ ✗

$w_1 < 0, w_0 > 0 \Rightarrow 28w_1 = -4$ $w_1 = \frac{-1}{7}$ 代入 (1) 得 $w_0 = \frac{9}{14}$

$w_1 < 0, w_0 < 0 \Rightarrow 28w_1 = -10$, $w_1 = \frac{-5}{14}$ ✗

$$E_{aug}(w) = \frac{1}{3}\left(3 \times \left(\frac{9}{14}\right)^2 + 17 \times \left(\frac{-1}{7}\right)^2 + 6\left(\frac{-1}{7}\right)\left(\frac{9}{14}\right) - 6\left(\frac{9}{14}\right) + 4\left(\frac{-1}{7}\right) + 5\right) + \left(\frac{1}{7} + \frac{9}{14}\right)$$

$\doteq$ 1.32 close to 1.3 [e] ✓

8. $x_1 = 2, y_1 = 9$ $\quad h(x) = w_0 + w_1 x$

$x_2 = -2, y_2 = -1$ $\quad h(x_1) = w_0 + 2w_1$

$h(x_2) = w_0 - 2w_1$

$$\min E_{aug}(w) = \frac{1}{2}\left[(w_0 + 2w_1 - 9)^2 + (w_0 - 2w_1 + 1)^2\right] + \frac{\lambda}{2}(w_0^2 + w_1^2)$$

$$= \frac{1}{2}(2w_0^2 + 8w_1^2 - 16w_0 - 40w_1 + 82) + \frac{\lambda}{2}(w_0^2 + w_1^2)$$

$$\frac{\partial E_{aug}(w)}{\partial w_0} = \frac{1}{2}(4w_0 - 16) + \lambda(w_0) = 0.$$

$$\frac{\partial E_{aug}(w)}{\partial w_1} = \frac{1}{2}(16w_1 - 40) + \lambda(w_1) = 0$$

$$\Rightarrow \begin{cases} 4w_0 - 16 + \lambda(w_0) = 0 \\ 16w_1 - 40 + \lambda(w_1) = 0 \end{cases} \Rightarrow \begin{cases} (4+2\lambda)w_0 = 16 \\ (16+2\lambda)w_1 = 40 \end{cases}$$

$(2+\lambda)w_0 = 8$

$(8+\lambda)w_1 = 20$

$$w = [w_0, w_1]^T = \left[\frac{8}{2+\lambda}, \frac{20}{8+\lambda}\right]^T = w_{reg}$$

$$4 = \left[\frac{8}{2+\lambda}, \frac{20}{8+\lambda}\right] \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} \Rightarrow 4 = \frac{8}{2+\lambda} + \frac{20}{8+\lambda}$$

$$\Rightarrow (\lambda+8)(\lambda+2) = 2(8+\lambda) + 5(2+\lambda) \Rightarrow \lambda^2 + 10\lambda + 16 = 7\lambda + 26$$

$$\Rightarrow \lambda^2 + 3\lambda - 10 = 0 \Rightarrow \lambda = -5 \text{ (不合)} \text{ or } 2$$

[b]

9. $E_{aug}(w) = E_{in}(w) + \frac{\lambda}{N} w^T w$

$$\nabla E_{aug}(w) = \frac{\partial E_{aug}}{\partial w} = \nabla E_{in}(w) + \frac{2\lambda}{N} w$$

$$\Rightarrow w_{t+1} \leftarrow w_t - \eta\left(\nabla E_{in}(w_t) + \frac{2\lambda}{N} w_t\right)$$

$$\Rightarrow w_{t+1} \leftarrow \left(1 - \eta\frac{2\lambda}{N}\right) w_t - \eta(\nabla E_{in}(w_t))$$

$\Rightarrow \lambda = \left(1 - \eta\frac{2\lambda}{N}\right)$

[b]

10.

$$\min \frac{1}{N+K}\left(\sum_{n=1}^{N}(y_n - w^T x_n)^2 + \sum_{k=1}^{K}(\tilde{y}_k - w^T \tilde{x}_k)^2\right)$$

$$\frac{\partial(\quad)}{\partial w} = 0 = \sum_{n=1}^{N}(y_n - w^T x_n) + \sum_{k=1}^{K}(\tilde{y}_k - w^T \tilde{x}_k)$$

The order is not important

$$\Rightarrow \sum_{n=1}^{N}(w^T x_n - y_n) = \sum_{n=1}^{N}(w^T \tilde{x}_k - \tilde{y}_k)$$

$$\Rightarrow \sum_{k=1}^{K}(\tilde{y}_k - w^T \tilde{x}_k) = \frac{\lambda}{N}\|w\|^2$$

$$= \frac{\lambda}{N}(w_1^2 + w_2^2 + \cdots + w_K^2) = \sum_{k=1}^{K}(w^T \tilde{x}_k)^2$$

$$\Rightarrow \underline{y = 0}$$

$$\Rightarrow \frac{1}{N}\|\tilde{x}w - \tilde{y}\|^2 = \frac{\lambda}{N}\|w\|^2$$

$$\Rightarrow \frac{1}{\lambda}(\tilde{x}w)^2 = \|w\|^2$$

$$\Rightarrow \underline{\tilde{x} = \sqrt{\lambda}\, I,\ y = 0} \qquad \underline{[b]}$$

11. The virtual examples are "uniformly distribution" between $[-r, r]$.

a simple noise product example

$$(x_1 + \epsilon)(x_2 + \epsilon) = x_1 x_2 + x_1\epsilon + x_2\epsilon + \epsilon^2$$

We know the expectation $E(k\epsilon) = 0$ (k is constant, because uniform distribution is symmetric.

The product of $x_n^T x_n$ $\quad [x_1\ \tilde{x}_1]\,[x_1\ \tilde{x}_1]^T \begin{bmatrix} +\epsilon^2 & \cdots \\ \cdots & +\epsilon^2 \end{bmatrix}$

will be $= x^T x + N \times E(\epsilon^2) \cdot I_{d+1}$

$$= x^T x + N \times \frac{1}{2r}\int_{-r}^{r} \epsilon^2 d\epsilon \cdot I_{d+1}$$

$$= x^T x + N \times \frac{1}{2r} \times \frac{2r^3}{3}$$

$$= x^T x + \frac{N}{3} r^2 I_{d+1} \qquad \underline{[C]}$$

12.

The optimal $y = \frac{(\sum_{n=1}^{N} y_n) + K}{N+2K}$

$\Rightarrow (\sum_{n=1}^{N} y_n) = y(N+2K) - K$ ... (1)

$$\min_{y \in R} \frac{1}{N} \sum_{n=1}^{N} (y - y_n)^2 + \frac{\lambda}{N} \Omega(y)$$

$$\frac{\partial(\ )}{\partial y} := \frac{1}{N} 2 \sum_{n=1}^{N} (y - y_n) + \frac{\lambda}{N} (\Omega(y))' = 0$$

$$\Rightarrow \frac{2}{N} (Ny - \sum_{n=1}^{N} y_n) + \frac{\lambda}{N} (\Omega(y))' = 0$$

Replace $\sum_{n=1}^{N} y_n$ by (1)

$$\Rightarrow \frac{2}{N} (Ny - y(N+2K) + K) + \frac{\lambda}{N} (\Omega(y))' = 0$$

$$\Rightarrow \quad = \lambda (\Omega(y))'$$

$$\Rightarrow \frac{2K}{\lambda} (2y - 1) = (\Omega(y))'$$

$$\Rightarrow \int (\Omega(y))' dy = \int \frac{2K}{\lambda} (2y - 1) dy = \frac{2K}{\lambda} (y^2 - y) + C$$

$$\Rightarrow \Omega(y) = \frac{2K}{\lambda} (y - 0.5)^2 + C$$

(b) ★

```python
import numpy as np
from dataclasses import dataclass, field
from typing import Optional

@dataclass
class Regression:
    algo:Optional[str] = None
    n_init:Optional[int] = None
    learning_rate:Optional[float] = None
    Q: Optional[int] = None

    def fit(self, x, y):
        self.x = np.insert(x, 0, 1, axis=1)
        self.y = y
        self.n = x.shape[0]

        self.Ein_linear_sgd = []
        self.Ein_logistic_sgd = []
        self.Ein_logistic_sgd_wlin = []

        if self.algo == 'linear regression':
            self.Ein = self.run_regression(self.x, self.y)[0]
        elif self.algo == 'linear sgd':
            for i in range(self.n_init):
                result = self.run_sgd_linear(self.x, self.y)[0]
                self.Ein_linear_sgd += [result]
        elif self.algo == 'logistic sgd':
            for i in range(self.n_init):
                result = self.run_sgd_logistic(self.x, self.y)[0]
                self.Ein_logistic_sgd += [result]
        elif self.algo == 'logistic sgd wlin':
            for i in range(self.n_init):
                result = self.run_sgd_logistic_wlin(self.x, self.y)[0]
                self.Ein_logistic_sgd_wlin += [result]

        return self

    def w800_ein_eout(self, x, y, x_test, y_test):
        self.x = np.insert(x, 0, 1, axis=1)
        self.y = y
        self.n = x.shape[0]

        self.x_test = np.insert(x_test, 0, 1, axis=1)
        self.y_test = y_test

        self.ans17 = []

        for i in range(self.n_init):
            w800 = self.run_sgd_logistic_wlin(self.x, self.y)[1]
            self.ans17 += [np.abs(self.Error01(w800, self.x, self.y) - self.Error01(w800, self.x_test,
self.y_test))]

        return self

    def wlin_ein_eout(self, x, y, x_test, y_test):
        self.x = np.insert(x, 0, 1, axis=1)
        self.y = y
        self.n = x.shape[0]

        self.x_test = np.insert(x_test, 0, 1, axis=1)
        self.y_test = y_test

        self.ans18 = 0

        wlin = self.run_regression(self.x, self.y)[1].flatten()
        self.ans18 = np.abs(self.Error01(wlin, self.x, self.y) - self.Error01(wlin, self.x_test,
self.y_test))

        return self

    def Q_ein_eout(self, x, y, x_test, y_test):
        self.x = np.insert(x, 0, 1, axis=1)
        self.y = y
        self.n = x.shape[0]

        self.x_test = np.insert(x_test, 0, 1, axis=1)
        self.y_test = y_test

        self.ans_last2 = []

        self.transform_x = self.Q_transform(self.Q, self.x)
        self.transform_x_test = self.Q_transform(self.Q, self.x_test)

        wlin = self.run_regression(self.transform_x, self.y)[1].flatten()
        self.ans_last2 = np.abs(self.Error01(wlin, self.transform_x, self.y) - self.Error01(wlin,
self.transform_x_test, self.y_test))

        return self

    def Q_transform(self, Q, x):
        original_x = x[:, 1:]
        new_x = x.copy()
        for i in range(2, Q+1):
            new_x = np.hstack((new_x, original_x**i))
        # print(new_x.shape)
        return new_x

    def run_regression(self, x, y):
        w = self.pseudo_inverse(x, y)
        Ein = self.Ein_sqrt(w, x, y)

        return Ein, w


    def run_sgd_linear(self, x, y):
        eta = self.learning_rate
        # w0 init
        w_t = np.zeros(x.shape[1])

        for i in range(800):
            sample_idx = np.random.randint(self.n, size=1)
            x_random = x[sample_idx].flatten()
            y_random = y[sample_idx].flatten()

            w_t += eta * 2 * (y_random-np.dot(w_t, x_random)) * x_random

        Ein = self.Ein_sqrt(w_t, x, y)
        return Ein, w_t

    def run_sgd_logistic(self, x, y):
        eta = self.learning_rate
        w_t = np.zeros(x.shape[1])

        for i in range(800):
            sample_idx = np.random.randint(self.n, size=1)
            x_random = x[sample_idx].flatten()
            y_random = y[sample_idx].flatten()

            w_t += eta * self.sigmoid(-y_random*np.dot(w_t, x_random))*(y_random * x_random)

        return self.Ein_ce(w_t, x, y), w_t

    def run_sgd_logistic_wlin(self, x, y):
        eta = self.learning_rate
        w_t = self.pseudo_inverse(x, y).flatten()

        for i in range(800):
            sample_idx = np.random.randint(self.n, size=1)
            x_random = x[sample_idx].flatten()
            y_random = y[sample_idx].flatten()

            w_t += eta * self.sigmoid(-y_random*np.dot(w_t, x_random))*(y_random * x_random)

        return self.Ein_ce(w_t, x, y), w_t

    def pseudo_inverse(self, x, y):
        return np.dot(np.linalg.pinv(x), y)

    def Ein_sqrt(self, w, x, y):
        return np.mean([(np.dot(w.flatten(), x[i]) - y[i])**2 for i in range(len(y))])

    def sigmoid(self, s):
        return 1/(1+np.exp(-s))

    # def Ein_ce(self, w, x, y):
    #     return np.mean(np.log(1+np.exp(-y*np.dot(x,w))))

    def Ein_ce(self, w, x, y):
        return np.mean(np.array([np.log(1+np.exp(-y[i].flatten()*np.dot(x[i].flatten(),w))) for i in
range(len(y))]))

    def Error01(self, w, x, y):
        error = 0.0
        for i in range(len(y)):
            sample_x = x[i].flatten()
            sample_y = y[i].flatten()
            if self.sign(np.dot(w, sample_x)) != sample_y:
                error += 1
        return error/len(y)

    def sign(self, value):
        if value >= 0:
            return 1
        else:
            return -1

}
```

In [ ]:
```
import numpy as np
from Regression import Regression
```

In [ ]:
```
"""
load data and preprocess
"""
# read data
with open('hw3_train.dat', 'rb') as f:
    training_data = np.array([np.float64(i.split()) for i in f.readlines()])

# turn x and y into numpy array
# x is the input feature vector space and y is the corresponding label
x = np.array(training_data[:,0:10])
y = np.reshape(np.array(list(map(int, training_data[:,10]))), (len(x), 1))
print(x.shape)
print(y.shape)

N = len(x)

with open('hw3_test.dat', 'rb') as f:
    test_data = np.array([np.float64(i.split()) for i in f.readlines()])

# turn x and y into numpy array
# x is the input feature vector space and y is the corresponding label
x_test = np.array(test_data[:,0:10])
y_test = np.reshape(np.array(list(map(int, test_data[:,10]))), (len(x_test), 1))
print(x_test.shape)
print(y_test.shape)

N = len(x)
```

```
(100, 10)
(100, 1)
(400, 10)
(400, 1)
```

In [ ]:
```
p13_kwargs = {
    'algo': 'linear regression',
}

p13 = Regression(**p13_kwargs).fit(x, y)
p13_ans = p13.Ein
p13_ans
```

Out[ ]:
```
0.7922347761105571
```

In [ ]:
```
p14_kwargs = {
    'algo': 'linear sgd',
    'n_init': 1000,
    'learning_rate': 0.001
}
p14 = Regression(**p14_kwargs).fit(x, y)
p14_ans = np.mean(p14.Ein_linear_sgd)
p14_ans
```

Out[ ]:
```
0.8229484676966267
```

In [ ]:
```
p15_kwargs = {
    'algo': 'logistic sgd',
    'n_init': 1000,
    'learning_rate': 0.001
}
p15 = Regression(**p15_kwargs).fit(x, y)
p15_ans = np.mean(p15.Ein_logistic_sgd)
p15_ans
```

Out[ ]:
```
0.657143936490148
```

In [ ]:
```
p16_kwargs = {
    'algo': 'logistic sgd wlin',
    'n_init': 1000,
    'learning_rate': 0.001,
}
p16 = Regression(**p16_kwargs).fit(x, y)
p16_ans = np.mean(p16.Ein_logistic_sgd_wlin)
p16_ans
```

Out[ ]:
```
0.6052759638463251
```

In [ ]:
```
p17_kwargs = {
    'n_init': 1000,
    'learning_rate': 0.001,
}
p17 = Regression(**p17_kwargs).w800_ein_eout(x, y, x_test, y_test)
p17_ans = np.mean(p17.ans17)
p17_ans
```

Out[ ]:
```
0.030962500000000018
```

In [ ]:
```
p18_kwargs = {
    'learning_rate': 0.001
}
p18 = Regression(**p18_kwargs).wlin_ein_eout(x, y, x_test, y_test)
p18_ans = np.mean(p18.ans18)
p18_ans
```

Out[ ]:
```
0.040000000000000036
```

In [ ]:
```
p19_kwargs = {
    'Q': 2
}
p19 = Regression(**p19_kwargs).Q_ein_eout(x, y, x_test, y_test)
p19_ans = p19.ans_last2
p19_ans
```

Out[ ]:
```
0.08249999999999999
```

In [ ]:
```
p20_kwargs = {
    'Q': 8
}
p20 = Regression(**p20_kwargs).Q_ein_eout(x, y, x_test, y_test)
p20_ans = p20.ans_last2
p20_ans
```

Out[ ]: 0.415