HTML

Homework 4

B07502166 魏子翔

1. The optimal solution to the problem can be calculated by the differential of the original problem, which is
$\frac{1}{N}\sum_{n=1}^{N} 2x_n(w \cdot x_n - y_n) + \frac{2\lambda}{N} w = 0 \;\Rightarrow\; \left(\sum_{n=1}^{N} x_n^2 + \lambda\right) w = \sum_{n=1}^{N} x_n y_n$
$$w = \frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2 + \lambda} \;,\; C = \left(\frac{\sum_{n=1}^{N} x_n y_n}{\sum_{n=1}^{N} x_n^2 + \lambda}\right)^2$$

2. $\tilde{w}^T \Gamma^{-1} = w^T \;\Rightarrow\; \tilde{w}^T = w^T \Gamma \;,\; \tilde{w} = \Gamma w$
$\Rightarrow\; \Omega(w) = \tilde{w}^T \tilde{w} = w^T \Gamma^2 w$

3. For $y = +1$, $\text{err}_{smooth}(w, x, +1) = (1 - \frac{\alpha}{2}) \ln(1 + \exp(-w^T x)) + \frac{\alpha}{2} \ln(1 + \exp(w^T x))$
$= (1-\alpha)\ln(1 + \exp(-w^T x)) + \frac{\alpha}{2}\ln(1 + \exp(-w^T x)) + \frac{\alpha}{2}\ln(1 + \exp(w^T x))$
$\Rightarrow\; \min_w \frac{1}{N}\sum_{n=1}^{N} \text{err}_{smooth}(w, x_n, +1)$
$= \min_w \frac{1}{N}\sum_{n=1}^{N} \text{err}(w, x_n, +1) + \frac{\alpha}{1-\alpha}\frac{1}{N}\sum_{n=1}^{N} \frac{1}{2}(\ln(1 + \exp(-w^T x_n)) + \ln(1 + \exp(w^T x_n)))$
$= \min_w \frac{1}{N}\sum_{n=1}^{N} \text{err}(w, x_n, +1) + \frac{\lambda}{N}\sum_{n=1}^{N} \frac{1}{2}(\ln(\frac{1}{h(w^T x_n)}) + \ln(\frac{1}{1-h(w^T x_n)}))$
$= \min_w \frac{1}{N}\sum_{n=1}^{N} \text{err}(w, x_n, +1) + \frac{\lambda}{N}\sum_{n=1}^{N} D_{KL}(P_u || P_h)$
Note that the constants won't change the result of minimum.
Similarly, $\min_w \frac{1}{N}\sum_{n=1}^{N} \text{err}_{smooth}(w, x_n, -1) = \min_w \frac{1}{N}\sum_{n=1}^{N} \text{err}(w, x_n, -1) + \frac{\lambda}{N}\sum_{n=1}^{N} D_{KL}(P_u || P_h)$
So the answer is $D_{KL}(P_u || P_h)$

4. Since the average of two points makes the smallest squared error, so
$$E_{loocv} = \frac{1}{3}\left(\left(1 - \frac{y_1 + y_2}{2}\right)^2 + \left(y_1 - \frac{1 + y_2}{2}\right)^2 + \left(y_2 - \frac{y_1 + 1}{2}\right)^2\right) \le \frac{1}{3}$$
$\Rightarrow\; y_1^2 + y_2^2 + 1 - y_1 - y_2 - y_1 y_2 \le \frac{2}{3}$
If we regard y_1 and y_2 as a 2-dimensional coordinate, then the original problem can be considered as the probability of point (x, y) in the ellipse $x^2 + y^2 + 1 - x - y - xy = \frac{2}{3}$, for $x, y \in [0, 2]$, which is equal to the area of the ellipse divided by 2×2.
The ellipse is rotated by 45^o, to calculate its long and short axis, we can let $x = \cos\frac{\pi}{4}u - \sin\frac{\pi}{4}v,\; y = \sin\frac{\pi}{4}u + \cos\frac{\pi}{4}v$, then the original equation will be
$$\frac{u^2}{2} + \frac{3v^2}{2} - \sqrt{2}u + 1 = \frac{2}{3} \;\Rightarrow\; \frac{(u-\sqrt{2})^2}{2} + \frac{3v^2}{2} = \frac{2}{3} \;\Rightarrow\; \frac{(u-\sqrt{2})^2}{\left(\frac{2}{\sqrt{3}}\right)^2} + \frac{3v^2}{\left(\frac{2}{3}\right)^2} = 1$$
So the area of the ellipse is $\pi \times \frac{2}{\sqrt{3}} \times \frac{2}{3}$, which gives the answer $\frac{\pi}{3\sqrt{3}}$

5. $Var[E_{val}] = Var[\frac{1}{K}\sum_{i=1}^{K} err] = \frac{1}{K^2}Var[\sum_{i=1}^{K} err] = \frac{1}{K^2}\left(E[\left(\sum_{i=1}^{K} err\right)^2] - E[\sum_{i=1}^{K} err]^2\right)$

   $E[\left(\sum_{i=1}^{K} err\right)^2] = E[\sum_{i=1}^{K}\sum_{j=1}^{K} err_i err_j] = E[\sum_{i=1}^{K} err_i^2] + E[\sum_{i=1}^{K}\sum_{j=1,i\neq j}^{K} err_i err_j]$

   $= KE[err^2] + \sum_{i=1}^{K}\sum_{j=1,i\neq j}^{K} E[err_i]E[err_j] = KE[err^2] + (K^2-K)E[err]^2$

   $Var[E_{val}] = \frac{1}{K^2}\left(KE[err^2] + (K^2-K)E[err]^2 - K^2E[err]^2\right) = \frac{1}{K^2}\left(KE[err^2] - KE[err]^2\right)$

   $= \frac{1}{K}Var[err]$

6. If we leave a positive example out, then the algorithm will return negative, so the prediction on that example will be wrong. Similarly, if we take a negative example, then the algorithm gives positive, so the prediction is also wrong. $E_{loocv}(A_{majority}) = \frac{1}{2N}2N = 1$

7. The error only occur when leaves the smallest positive or the largest negative one out, cause the model may give a too positive or too negative value. So the upper bound of loo error is $\frac{2}{N}$.

8. The margin is equal to the min distance between the points and w, so the largest margin is $\frac{1}{2}(x_{M+1} - x_M)$, when w is in the middle of x_{M+1} and x_M.

9. From the examples, we can derive four conditions as follow.

$$4w_2 + b \geq 1 \tag{1}$$
$$2w_1 + b \leq -1126 \tag{2}$$
$$-w_1 + b \geq 1 \tag{3}$$
$$b \geq 1 \tag{4}$$

   Then from (1)-(4), we have $w_2 \geq 0$, from (2)+(-(4)), we have $w_1 \leq \frac{-1127}{2}$, so $w_1 = \frac{-1127}{2},\ w_2 = 0,\ b = 1$ is the optimal solution.

10. For any $(x_i, y_i = +1)$ pair, to maximize the probability of error, assume y_j of all the other examples are -1, then

    $\hat{h}(x_i) = \text{sign}\left(y_i + \sum_{j=1,j\neq i}^{N} y_j \exp\left(-\gamma||x_j - x_i||^2\right)\right) = \text{sign}\left(y_i - (N-1)\exp\left(-\gamma\epsilon^2\right)\right)$

    $\Rightarrow\ 1 - (N-1)\exp\left(-\gamma\epsilon^2\right) \geq 0,\ \exp(-\gamma\epsilon^2) \leq \frac{1}{N-1},\ \gamma \geq \frac{\ln(N-1)}{\epsilon^2}$

    Similarly, for those $(x_i, y_i = -1)$ pairs, $\hat{h}(x_i) = \text{sign}\left(y_i + (N-1)\exp\left(-\gamma\epsilon^2\right)\right)$

    $\Rightarrow\ -1 + (N-1)\exp\left(-\gamma\epsilon^2\right) \leq 0,\ \exp(-\gamma\epsilon^2) \leq \frac{1}{N-1},\ \gamma \geq \frac{\ln(N-1)}{\epsilon^2}$

    So if $\gamma \geq \frac{\ln(N-1)}{\epsilon^2}$, the hypothesis on all examples should be all correct, $E_{in}(\hat{h}) = 0$

11. $||\phi(x) - \phi(x')||^2 = K(\phi(x),\phi(x)) + K(\phi(x'),\phi(x')) - 2K(\phi(x),\phi(x')) = 2 - 2K(\phi(x),\phi(x')) \leq 2$

    $\Rightarrow\ ||\phi(x) - \phi(x')|| \leq \sqrt{2}$

12.~20.
```python
from liblinear.liblinearutil import *
import numpy as np

def read_data(filename):
    X, y = [], []
    with open(filename, "r") as f:
        for line in f:
            line = line[:-1]
            line = list(map(float, line.split()))
            phi = [1]
            for i in range(len(line[:-1])):
                phi.append(line[i])
                now1 = line[i]
                for j in range(i, len(line[:-1])):
                    phi.append(now1*line[j])
                    now2 = now1*line[j]
                    for k in range(j, len(line[:-1])):
                        phi.append(now2*line[k])
                        now3 = now2*line[k]
                        for l in range(k, len(line[:-1])):
                            phi.append(now3*line[l])
            X.append(phi)
            y.append(int(line[-1]))
    return np.array(X), np.array(y)

def P1213(x_train, y_train, x_test, y_test):
    prob = problem(y_train, x_train)
    err_in_list = []
    err_out_list = []
    for lamb in [-6, -3, 0, 3, 6]:
        c = 1/(2*10**lamb)
        param = parameter(f'-s 0 -c {c} -e 0.000001 -q')
        m = train(prob, param)
        _, p_in, _ = predict(y_train, x_train, m)
        _, p_out, _ = predict(y_test, x_test, m)
        err_in_list.append(1-p_in[0]/100)
        err_out_list.append(1-p_out[0]/100)

    print(err_out_list)
    print(err_in_list)

def P1416(x_train, y_train, x_test, y_test):
    lamb_list = [-6, -3, 0, 3, 6]
    choose = [0]*5
    err_out_list = []
```

```python
    err_out16_list = []
    for j in range(256):
        err_val_list = []
        rng = np.random.default_rng(j)
        idx = rng.choice(200, 80, replace=False)
        x_train_test, x_train_train = x_train[idx], np.delete(
            x_train, idx, axis=0)
        y_train_test, y_train_train = y_train[idx], np.delete(
            y_train, idx, axis=0)
        prob = problem(y_train_train, x_train_train)
        prob16 = problem(y_train, x_train)
        for lamb in lamb_list:
            c = 1/(2*10**lamb)
            param = parameter(f'-s 0 -c {c} -e 0.000001 -q')
            m = train(prob, param)
            m16 = train(prob16, param)
            _, p_val, _ = predict(y_train_test, x_train_test, m)
            err_val_list.append(1-p_val[0]/100)
        best_err = 1
        for i in range(len(err_val_list)):
            if err_val_list[i] <= best_err:
                best_err = err_val_list[i]
                best_lamb = i
        choose[best_lamb] += 1
        c = 1/(2*10**lamb_list[best_lamb])
        param = parameter(f'-s 0 -c {c} -e 0.000001 -q')
        m = train(prob, param)
        _, p_out, _ = predict(y_test, x_test, m)
        err_out_list.append(1-p_out[0]/100)

        m16 = train(prob16, param)
        _, p_out16, _ = predict(y_test, x_test, m16)
        err_out16_list.append(1-p_out16[0]/100)
    print(choose)
    print(np.mean(err_out_list))
    print(np.mean(err_out16_list))

def P17(x_train, y_train):
    lamb_list = [-6, -3, 0, 3, 6]
    err_cv_list = []
    for i in range(256):
        np.random.seed(i)
        ori = np.array((range(200)))
        np.random.shuffle(ori)
        err_lamb_list = []
```

```python
        for lamb in lamb_list:
            c = 1/(2*10**lamb)
            param = parameter(f'-s 0 -c {c} -e 0.000001 -q')
            err_list = []
            for j in range(5):
                train_idx = np.concatenate(
                    (ori[:j*40], ori[(j+1)*40:]), axis=0)
                valid_idx = ori[j*40:(j+1)*40]
                prob = problem(y_train[train_idx], x_train[train_idx])
                m = train(prob, param)
                _, p_cv, _ = predict(y_train[valid_idx], x_train[valid_idx], m)
                err_list.append(1-p_cv[0]/100)
            err_lamb_list.append(np.mean(err_list))
        err_cv_list.append(min(err_lamb_list))
    print(np.mean(err_cv_list))

def P1819(x_train, y_train, x_test, y_test):
    lamb_list = [-6, -3, 0, 3, 6]
    prob = problem(y_train, x_train)
    err_out_list = []
    for lamb in lamb_list:
        c = 1/(10**lamb)
        param = parameter(f'-s 6 -c {c} -e 0.000001 -q')
        m = train(prob, param)
        _, p_out, _ = predict(y_test, x_test, m)
        err_out_list.append(1-p_out[0]/100)
    print(err_out_list)
    choose_lamb = np.argmin(err_out_list)
    c = 1/(10**lamb_list[choose_lamb])
    param = parameter(f'-s 6 -c {c} -e 0.000001 -q')
    m = train(prob, param)
    w = m.get_decfun()[0]
    count = 0
    for i in range(len(w)):
        if abs(w[i]) <= 10e-6:
            count += 1
    print(count)

def P20(x_train, y_train):
    prob = problem(y_train, x_train)
    c = 1/(2*10**3)
    param = parameter(f'-s 0 -c {c} -e 0.000001 -q')
    m = train(prob, param)
    w = m.get_decfun()[0]
    count = 0
```

```python
    for i in range(len(w)):
        if abs(w[i]) <= 10e-6:
            count += 1
    print(count)

if __name__ == "__main__":
    x_train, y_train = read_data("./hw4_train.dat")
    x_test, y_test = read_data("./hw4_test.dat")
    P1213(x_train, y_train, x_test, y_test)
    P1416(x_train, y_train, x_test, y_test)
    P17(x_train, y_train)
    P1819(x_train, y_train, x_test, y_test)
    P20(x_train, y_train)
```