ML HW5

B06303126 Lo Yun Chien

December 2020

Question 1

Correct answer: (d)
After transformation, we yield three inequality,

$$\begin{aligned} 2w_1 - 4w_2 - b &\geq 1 \\ b &\geq 1 \\ -2w_1 - 4w_2 - b &\geq 1 \end{aligned}$$

We want to minimize $\mathbf{w}^T\mathbf{w}$, we should let $2w_1 + 4w_2$ as small as possible, then we want $b = 1$. For the last two inequality, after plotting it on 2D plane, the smallest circle centered at 0 intersecting with the area would be $(0, \frac{1}{2})$. Therefore $w_1 = 0$.

Question 2

Correct answer: (c)
By definition, margin is $\frac{1}{\|\mathbf{w}\|}$, with $\mathbf{w} = (0, \frac{1}{2})$, we should have margin is 4.

Question 3

Correct answer: (e)
For this kind of data, only points in (x_M, x_{M+1}) can separate the data. The margin is $\min\{x^* - x_M, x_{M+1} - x^*\}$, then to max this value, the biggest margin is $\frac{1}{2}(x_{M+1} - x_M)$

Question 4

Correct answer: (a)
For $\mathbf{y} = (1, 1)$ or $\mathbf{y} = (-1, -1)$, 1D perceptron can always produce these dichotomy by setting cutting point at -0.5. So the expected dichotomies should be

$$E[Dichotomy] = 2 + 2P(|x_1 - x_2| \geq 2\rho)$$

To calculate $P(|x_1 - x_2| \geq 2\rho)$, note that the graph of $|x_1 - x_2| \geq 2\rho$ intersect with $[0, 1] \times [0, 1]$ on 2D plane yield two triangles, each with area $\frac{1}{2}(1 - 2\rho)^2$. Since it's uniform distribution, hence $P(|x_1 - x_2| \geq 2\rho) = (1 - 2\rho)^2$.

Question 5

Correct answer:(c)
Using Lagrange method, define a new function as

$$\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \alpha_n(\rho^+ - y_n(\mathbf{w}^T x_n + b))[y_n = 1] + \sum_{i=1}^{N} \alpha_n(\rho^- - y_n(\mathbf{w}^T x_n + b))[y_n = -1]$$

with $\alpha_n \geq 0$ for all n. Solve the dual problem with these equation, take partial derivative about $\mathbf{w}, b$. we can find $\sum_{i=1}^{N} \alpha_n y_n = b$, and $\sum_{i=1}^{N} \alpha_n y_n x_n = \mathbf{w}$ when maxing it. Therefore the remaining term is

$$-\frac{1}{2}\|\sum_{i=1}^{N} \alpha_n y_n x_n\| + \sum_{i=1}^{N} \alpha_n \rho^+[y_n = 1] + \sum_{i=1}^{N} \alpha_n \rho^-[y_n = -1]$$

When taking opposite, the result is (c)

Question 6

Correct answer:(e)
By question 5, we are solving

$$\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n\alpha_m y_n y_m x_n^T x_m - \sum_{n=1}^{N} \rho_+[y_n = 1]\alpha_n - \sum_{n=1}^{N} \rho_-[y_n = -1]\alpha_n$$

By using the constraint $\sum_{n=1}^{N} y_n\alpha_n = 0$, we have $\sum_{n=1}^{N}[y_n = 1]\alpha_n = \sum_{n=1}^{N}[y_n = -1]\alpha_n = \frac{1}{2}\sum_{n=1}^{N} \alpha_n$. Using this fact to rewrite the function,

$$\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n\alpha_m y_n y_m x_n^T x_m - \frac{1}{2}\sum_{n=1}^{N} \alpha_n(\rho_+ + \rho_-)$$

Define $\alpha_n = \alpha_n^{'} \frac{\rho_+ + \rho_-}{2}$ and put it into the function

$$\frac{1}{2}(\frac{\rho_+ + \rho_-}{2})^2 \sum_{n=1}^{N}\sum_{m=1}^{N} \alpha_n^{'}\alpha_m^{'} y_n y_m x_n^T x_m - (\frac{\rho_+ + \rho_-}{2})^2 \sum_{n=1}^{N} \alpha_n^{'}$$

Since all the constraints won't change by simply adding a factor, this new problem is the original hard margin SVM, Which solution is α^*. Therefore, by $\alpha_n = \alpha_n^{'} \frac{\rho_+ + \rho_-}{2}$, the uneven SVM solution is (e)

Question 7

Correct answer: (d)
While setting $K = 0$, $\log_2 0$ is undefined.

Question 8

Correct answer: (c)

$$\begin{aligned}\|\phi(x) - \phi(x')\| &= \phi(x)^T\phi(x) - 2\phi(x')^T\phi(x) + \phi(x')^T\phi(x') \\ &= K(x,x) - 2K(x',x) + K(x',x') \\ &= 2 - 2e^{-\gamma\|x-x'\|}\end{aligned}$$

We want to maximize $\|\phi(x) - \phi(x')\|$, so takes $\|x' - x\|$ to infinity, then the answer is 2.

Question 9

Correct answer: (d)
We have

$$h_{1,0}(x) = sign(\sum_{i=1}^{N} y_n e^{-\gamma\|x_n - x\|})$$

Since we want $E_{in} = 0$, consider an example (x_0, y_0) with $y_0 = 1$. We need $h_{1,0}(x_0) = 1$ i.e. $\sum_{i=1}^{N} y_n e^{-\gamma\|x_n - x_0\|} > 0$. Worst case is that all the other $y_n = -1$, adding that when $n = 0$, the term is 1, we yield

$$\sum_{n\neq 0} -e^{-\gamma\|x_n - x_0\|} > -1$$

With all $\|x_n - x_0\| \geq \epsilon^2$, $\sum_{n\neq 0} -e^{-\gamma\|x_n - x_0\|} \geq \sum_{n\neq 0} -e^{-\gamma\epsilon^2}$, $(N-1) - e^{-\gamma\epsilon^2} \geq -1$, $(N-1)e^{-\gamma\epsilon^2} \leq 1$. Solve this inequality, we have

$$\gamma \geq \frac{\ln(N-1)}{\epsilon^2}$$

Question 10

Correct answer: (c)
$w_t = \sum_{n=1}^{N} \alpha_{t,n}\phi(x_n)$, using this equation, we have

$$w_{t+1} = \sum_{n=1}^{N} \alpha_{t,n}\phi(x_n) + y_{n(t)}\phi(x_n)$$

As we can see, $\alpha_{t+1} = \alpha_t$. except for index n will add y_n.

Question 11

Correct answer: (a)
$w_t^T\phi(x) = \sum_{n=1}^{N} \alpha_{t,n}\phi(x_n)^T\phi(x)$, by using $w_t = \sum_{n=1}^{N} \alpha_{t,n}\phi(x_n)$. Therefore, kernel trick imply $\sum_{n=1}^{N} \alpha_{t,n}\phi(x_n)^T\phi(x) = \sum_{n=1}^{N} \alpha_{t,n}K(x_n, x)$

Question 12

Correct answer: (b)
Using KKT condition, we yield for $\alpha_n > 0, b = y_n - y_n\xi_n - w^T z_n$. represent it as $\xi_n = \frac{b - y_n w^T z_n}{-y_n}$, since $\xi_n \geq 0$, consider $y_n = 1$, we yield $b \leq 1 - w^T z_n$, for $y_n = -1$, we yield $b \geq -1 - w^T z_s$. To maximize b, we choose b satisfying all n constraint and is the largest, which is the lowest upper bound when $y_n = 1$.

Question 13

Correct answer: (e)
Using Lagrange method, we want to minimize the below value

$$L(w, b, \xi) = \frac{1}{2} w^T w + C \sum_{n=1}^{N} \xi_n^2 + \sum_{n=1}^{N} \alpha_n (1 - \xi_n - y_n (w^T z_n + b))$$

Where $\frac{\partial L}{\partial \xi_i} = 2C\xi_i - \alpha_i = 0$, substitute ξ for α

$$\frac{1}{2} w^T w + C \sum_{n=1}^{N} \frac{\alpha_n^2}{(2C)^2} + \sum_{n=1}^{N} \alpha_n (1 - \frac{\alpha_n}{2C} - y_n (w^T z_n + b))$$

Using the same method calculate $\frac{\partial L}{\partial w} = 0$ and $\frac{\partial L}{\partial b} = 0$ should have $\sum_{n=1}^{N} \alpha_n y_n = 0$ and $w = \sum \alpha_n y_n z_n$ separately, finally we have

$$-\frac{1}{2} w^T w - \sum_{n=1}^{N} \frac{\alpha_n^2}{4C} + \sum_{n=1}^{N} \alpha_n$$

Notice that $\sum_{n=1}^{N} \frac{\alpha_n^2}{4C}$ only affect diagonal entry, and the coefficient is $\frac{1}{2C}$.

Question 14

Correct answer: (e)
Notice that the last maximization problem is solved under constraint $2C\xi_i - \alpha_i = 0$, then the answer is (e).

In [1]:
```python
from svmutil import *
import pandas as pd
import numpy as np
from math import pow
import timeit
import operator
```

In [2]:
```python
# LibSVM function:

#svm_type = model.get_svm_type()
#nr_class = model.get_nr_class()
#svr_probability = model.get_svr_probability()
#class_labels = model.get_labels()
#sv_indices = model.get_sv_indices()
#nr_sv = model.get_nr_sv()
#is_prob_model = model.is_probability_model()
#support_vector_coefficients = model.get_sv_coef()
#support_vectors = model.get_SV()
```

In [3]:
```python
# Read Data

y, x = svm_read_problem("satimage.scale")
test_y, test_x = svm_read_problem("satimage.scale.t")
```

In [4]:
```python
def problem15():
    trainY = [1 if target == 3 else -1 for target in y]
    trainX = x
    model = svm_train(trainY, trainX, "-s 0 -t 0 -c 10")
    SV = pd.DataFrame(model.get_SV())
    SV = SV.fillna(0)
    alphaY = pd.DataFrame(model.get_sv_coef())
    wNorm = np.linalg.norm(SV.T.dot(alphaY))
    print("Problem 15: |w| is ", wNorm)
    return

problem15()
```

```
Problem 15: |w| is  8.457084298367056
```

In [5]:
```python
def problem16():
    print("Problem 16: \n")
    for i in range(1, 6):
        trainY = [1 if target == i else -1 for target in y]
        trainX = x
        model = svm_train(trainY, trainX, "-s 0 -t 1 -d 2 -c 10")
        print(f"This model depicts {i} versus not {i}:")
        result = svm_predict(trainY, trainX, model)
        print()
    return

problem16()
```

```
Problem 16:

This model depicts 1 versus not 1:
Accuracy = 98.3766% (4363/4435) (classification)

This model depicts 2 versus not 2:
Accuracy = 99.301% (4404/4435) (classification)

This model depicts 3 versus not 3:
Accuracy = 95.8061% (4249/4435) (classification)

This model depicts 4 versus not 4:
Accuracy = 90.6426% (4020/4435) (classification)

This model depicts 5 versus not 5:
Accuracy = 95.6483% (4242/4435) (classification)
```

```python
In [6]: def problem17():
    print("Problem 17: \n")
    MostNrSV = 0
    for i in range(1, 6):
        trainY = [1 if target == i else -1 for target in y]
        trainX = x
        model = svm_train(trainY, trainX, "-s 0 -t 1 -d 2 -c 10")
        print(f"This model depicts {i} versus not {i}:")
        result = svm_predict(trainY, trainX, model)
        print(f"And the nr_sv is: {model.get_nr_sv()} \n")
        if MostNrSV < model.get_nr_sv():
            MostNrSV = model.get_nr_sv()

    print(f"The biggest number of SV in these models is {MostNrSV}")
    return

problem17()
```

```
Problem 17:

This model depicts 1 versus not 1:
Accuracy = 98.3766% (4363/4435) (classification)
And the nr_sv is: 520

This model depicts 2 versus not 2:
Accuracy = 99.301% (4404/4435) (classification)
And the nr_sv is: 165

This model depicts 3 versus not 3:
Accuracy = 95.8061% (4249/4435) (classification)
And the nr_sv is: 750

This model depicts 4 versus not 4:
Accuracy = 90.6426% (4020/4435) (classification)
And the nr_sv is: 859

This model depicts 5 versus not 5:
Accuracy = 95.6483% (4242/4435) (classification)
And the nr_sv is: 690

The biggest number of SV in these models is 859
```

In [7]:
```python
def problem18():
    print("problem 18: \n")
    BestACC, BestC = -1, 0
    trainY, trainX = [1 if target == 6 else -1 for target in y], x
    testY, testX = [1 if target == 6 else -1 for target in test_y], test_x

    for i in range(-2, 3, 1):
        cost = pow(10, i)
        print(f"This model consider cost is {cost}:")
        model = svm_train(trainY, trainX, f'-s 0 -t 2 -g 10 -c {cost}')
        result = svm_predict(testY, testX, model)
        print()
        if BestACC < result[1][0]:
            BestC = cost
            BestACC = result[1][0]

    print(f"Among these five models, the best ACC is {BestACC}, best C is {BestC}")
    return

problem18()
```

```
problem 18:

This model consider cost is 0.01:
Accuracy = 76.5% (1530/2000) (classification)

This model consider cost is 0.1:
Accuracy = 83.65% (1673/2000) (classification)

This model consider cost is 1.0:
Accuracy = 89.35% (1787/2000) (classification)

This model consider cost is 10.0:
Accuracy = 90.3% (1806/2000) (classification)

This model consider cost is 100.0:
Accuracy = 90.3% (1806/2000) (classification)

Among these five models, the best ACC is 90.3, best C is 10.0
```

In [8]:

```python
def problem19():
    print("Problem 19: \n")
    BestACC, BestGamma = -1, 0
    trainY, trainX = [1 if target == 6 else -1 for target in y], x
    testY, testX = [1 if target == 6 else -1 for target in test_y], test_x

    for i in range(-1, 4, 1):
        gamma = pow(10, i)
        print(f"This model using gamma equal {gamma}: ")
        model = svm_train(trainY, trainX, f'-s 0 -t 2 -g {gamma} -c 0.1')
        result = svm_predict(testY, testX, model)
        print()
        if BestACC < result[1][0]:
            BestGamma = gamma
            BestACC = result[1][0]

    print(f"Among these five models, the best ACC is {BestACC}, best gamma is {BestGamma}")
    return

problem19()
```

```
Problem 19:

This model using gamma equal 0.1:
Accuracy = 90.15% (1803/2000) (classification)

This model using gamma equal 1.0:
Accuracy = 93% (1860/2000) (classification)

This model using gamma equal 10.0:
Accuracy = 83.65% (1673/2000) (classification)

This model using gamma equal 100.0:
Accuracy = 76.5% (1530/2000) (classification)

This model using gamma equal 1000.0:
Accuracy = 76.5% (1530/2000) (classification)

Among these five models, the best ACC is 93.0, best gamma is 1.0
```

In [12]:
```python
def problem20():
    start = timeit.default_timer()
    gammaDict = {0.1:0, 1:0, 10:0, 100:0, 1000:0}

    X = pd.DataFrame(x)
    X = X.fillna(0)
    col = X.columns.tolist()
    col.sort()
    X = X[col]
    Y = [1 if target == 6 else -1 for target in y]
    Y = pd.Series(Y)

    for _ in range(1000):
        best_gamma = 0
        best_ACC = -1
        indexes = np.random.choice(X.index, size = 200)
        ValX, ValY = X.loc[indexes], Y.loc[indexes]
        TrainX, TrainY = X.drop(indexes, axis = 0), Y.drop(indexes, axis = 0)
        for gamma in [0.1, 1, 10, 100, 1000]:
            #print(f"This model uses gamma equal {gamma}: ")
            model = svm_train(TrainY.to_numpy(), TrainX.to_numpy(), f'-s 0 -t 2 -g {gamma} -c 0.1')
            result = svm_predict(ValY.to_numpy(), ValX.to_numpy(), model)
            if result[1][0] > best_ACC:
                best_ACC = result[1][0]
                best_gamma = gamma
        gammaDict[best_gamma] += 1
        #print(f"The best gamma is {best_gamma}")

    stop = timeit.default_timer()
    print('Time: ', stop - start)
    return gammaDict

answer20 = problem20()
```

In [13]:
```
print(f"The most frequent gamma is {max(answer20.items(), key=operator.itemgetter(1))[0]}")
```

```
The most frequent gamma is 1
```