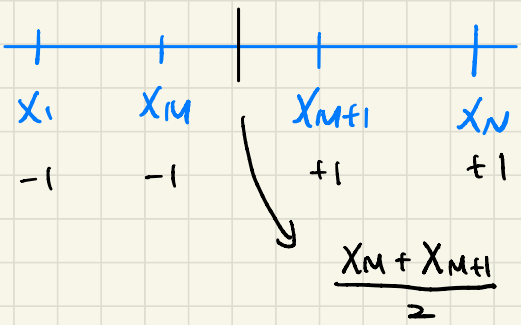
Q1

Plot samples in a line.



The hypothesis set is a positive-rays.

If we choose threshold to be the average of X_M and X_{M+1}, the hypothesis will have the largest margin with the form of $\text{sign}\left(X_n - \frac{X_M+X_{M+1}}{2}\right)$ and can separate all the samples from X_1 to X_N.

Therefore, in the form of $\text{sign}\left(X_n - \frac{X_M+X_{M+1}}{2}\right)$, w will be 1, b will be $-\frac{X_M+X_{M+1}}{2}$.

#

Q2

(3) Correct.

If take z as another kind of X, and according to the derivation in lecture slide, we are doing the following optimization.

$$\begin{aligned} \max_{b,\mathbf{w}} \quad & \frac{1}{\|\mathbf{w}\|} \\ \text{subject to} \quad & \text{every } y_n(\mathbf{w}^T\mathbf{x}_n + b) > 0 \\ & \min_{n=1,\dots,N} \; y_n(\mathbf{w}^T\mathbf{x}_n + b) = 1 \end{aligned}$$

Hence, the margin is equal to $\frac{1}{\|w\|} \times \underbrace{\min_{n=1,\dots,N} y_n(w^T x_n + b)}_{=1}$

$$= \frac{1}{\|w\|} \quad \#$$

(1) (2) False

Because (3) is correct, and there are just root and reciprocal / linear relationship between (1) / (2) and (3), so (1) / (2) are not equal to the length of margin.

(4) Correct

Following from the complementary slackness from the lecture slide,

$$\alpha_n \left(1 - y_n \left(w^T z_n + b\right)\right) = 0$$

$$\Rightarrow \sum_{n=1}^{N} \alpha_n \left(1 - y_n \left(w^T z_n + b\right)\right) = 0$$

$$\Rightarrow \sum_{n=1}^{N} \alpha_n - \underbrace{\sum_{n=1}^{N} \alpha_n y_n w^T z_n}_{= w} - \underbrace{\sum_{n=1}^{N} \alpha_n y_n b}_{= 0} = 0$$

$$\Rightarrow \sum_{n=1}^{N} \alpha_n - w^T w = 0$$

$$\Rightarrow \sum_{n=1}^{N} \alpha_n = \|w\|^2$$

$$\Rightarrow \left(\sum_{n=1}^{N} \alpha_n\right)^{-\frac{1}{2}} = \|w\|^{-1} \text{ (length of margin)}$$

#

(5) False

Because (4) is correct, and there is reciprocal and square relationship between (4) and (5).

Hence, (5) is not equal to length of margin.

#

(6) Correct.

Because (3), (4) is correct, and $w = \sum_{n=1}^{N} \alpha_n y_n z_n$.

$$\left(2 \sum_{n=1}^{N} \alpha_n - \left\| \sum_{n=1}^{N} \alpha_n y_n z_n \right\|^2 \right)^{-1/2}$$

$$= \left(2\|w\|^2 - \|w\|^2 \right)^{-1/2}$$

$$= \left(\|w\|^2 \right)^{-1/2}$$

$$= \|w\|^{-1} \quad (\text{length of margin})$$

#

Q3

Set the Lagarangian equation and let w be (w_1, w_2).

$$L(b, w, \alpha)$$

$$= \frac{1}{2}(w_1^2 + w_2^2) + \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n \left(1 - y_n (w_1 x_{n1} + w_2 x_{n2} + b)\right)$$

$$+ \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n \left(\underbrace{\rho_-}_{=4} - y_n (w_1 x_{n1} + w_2 x_{n2} + b)\right)$$

, which x_{n1} means the 1st term of x_n, and so on.

$$\boxed{\frac{\partial L}{\partial w_1} = 0} \quad - ①$$

$$w_1 - \alpha_4 \overset{1}{y_4} \overset{1}{x_{41}} - \alpha_1 \overset{-1}{y_1} \overset{0}{x_{11}} - \alpha_2 y_2 \overset{0}{x_{21}} - \alpha_3 y_3 \overset{0}{x_{31}}$$

$$= w_1 - \alpha_4$$

$$= 0$$

$$\boxed{\frac{\partial L}{\partial w_2} = 0} \quad - ②$$

$$w_2 - \alpha_4 \overset{0}{y_4} \overset{-1}{x_{42}} - \alpha_1 \overset{1}{y_1} x_{12} - \alpha_2 \overset{0}{y_2} x_{22} - \alpha_3 \overset{-1}{y_3} \overset{-1}{x_{32}}$$

$$= w_2 + \alpha_1 - \alpha_3$$

$$= 0$$

$$\boxed{\frac{\partial L}{\partial b} = 0} \quad - (3)$$

$$-\alpha_4 y_4^{1} - \alpha_1 y_1^{-1} - \alpha_2 y_2^{-1} - \alpha_3 y_3^{-1}$$

$$= -\alpha_4 + \alpha_1 + \alpha_2 + \alpha_3$$

$$= 0$$

$$\boxed{\frac{\partial L}{\partial \alpha_1} = 0} \quad - (4)$$

$$4 - y_1 w_1 x_{11}^{0} - y_1 w_2 x_{12}^{1} - y_1 b^{-1}$$

$$= 4 + w_2 + b$$

$$= 0$$

$$\boxed{\frac{\partial L}{\partial \alpha_2} = 0} \quad - (5)$$

$$4 - y_2 w_1 x_{21}^{0} - y_2 w_2 x_{12}^{0} - y_2 b^{-1}$$

$$= 4 + b$$

$$= 0$$

$$\boxed{\frac{\partial L}{\partial \alpha_3} = 0} \quad - ⑥$$

$$4 - y_3 w_1 x_{31} - y_3 w_2 x_{32} - y_3 b$$

(x_{31}: 0, $y_3 w_1$: −1, x_{32}: −1, $y_3 b$: −1)

$$= 4 - w_2 + b$$

$$= 0$$

$$\boxed{\frac{\partial L}{\partial \alpha_4} = 0} \quad - ⑦$$

$$1 - y_4 w_1 x_{41} - y_4 w_2 x_{42} - y_4 b$$

(y_4: 1, x_{41}: 1, x_{42}: 0, $y_4 b$: 1)

$$= 1 - w_1 - b$$

$$= 0$$

Through ⑤, $b = -4$

Through ⑥, $w_2 = 0$

Through ⑦, $w_1 = 5$

$\Rightarrow w = (5, 0)$

$b = -4$

#

Q4

Apply duality, solve lagarange dual problem.

$$\min_{b,w}\left(\max_{\text{all } \alpha_n \geq 0} L(b,w,\alpha)\right) = \max_{\text{all } \alpha_n \geq 0}\left(\min_{b,w} L(b,w,\alpha)\right)$$

$$L(b,w,\alpha)$$

$$= \frac{1}{2} w^T w$$

$$+ \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n (1 - y_n (w^T z_n + b)) \quad (4)$$

$$+ \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n (\rho - y_n (w^T z_n + b)) \quad (1,2,3)$$

$$\frac{\partial L}{\partial b} = 0 = \sum_{n=1}^{N} [\![y_n = +1]\!] - \alpha_n y_n + \sum_{n=1}^{N} [\![y_n = -1]\!] - \alpha_n y_n$$

$$= \frac{1}{2} w^T w$$

$$+ \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n (1 - y_n w^T z_n) \quad (4)$$

$$+ \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n (\rho - y_n w^T z_n) \quad (1,2,3)$$

$$\frac{\partial L}{\partial w_i} = w_i - \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n y_n z_{n,i} - \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n y_n z_{n,i}$$

$$w = \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n y_n z_n + \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n y_n z_n$$

$$= \frac{1}{2} w^T w + \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n - \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n y_n w^T z_n + \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n \rho - \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n y_n w^T z_n$$

$$\hookrightarrow = -w^T w$$

$$= -\frac{1}{2} w^T w + \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n + \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n \rho$$

Let constraints be

$$\sum_{n=1}^{N} [\![y_n = +1]\!] - \alpha_n y_n + \sum_{n=1}^{N} [\![y_n = -1]\!] - \alpha_n y_n = 0,$$

all $\alpha_n \geq 0$,

$$w = \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n y_n z_n + \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n y_n z_n = \sum_{n=1}^{N} \alpha_n y_n z_n$$

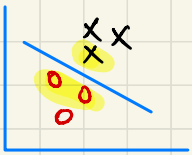
$$\max_{\text{constraints}} \left(\min_{b,w} -\frac{1}{2} w^T w + \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n + \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n \rho_- \right)$$

$$\Leftrightarrow \max_{\text{constraints}} -\frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n y_n z_n \right\|^2 + \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n + \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n \rho_-$$

$$\Leftrightarrow \min_{\text{constraints}} \frac{1}{2} \left\| \sum_{n=1}^{N} \alpha_n y_n z_n \right\|^2 - \sum_{n=1}^{N} [\![y_n = +1]\!] \alpha_n - \sum_{n=1}^{N} [\![y_n = -1]\!] \alpha_n \rho_-$$

the ans #

Q5



Let w', b' be the weight & bias in the uneven margin SVM.

Consider the support vectors (the same as the even margin SVM), the following equation will hold.

$$w'^T x_n + b' = \boxed{\frac{1+y_n}{2} \times 1} + \boxed{\frac{1-y_n}{2} \times (-\rho_-)}$$

↘ positive case ↘ negative case

Replace y_n with $w^T x_n + b$, which the w and b is the weight and bias in even margin SVM.

$$w'^T x_n + b' = \frac{1 + w^T x_n + b}{2} + \frac{1 - w^T x_n - b}{2}(-\rho_-)$$

$$= \left(\frac{1+\rho_-}{2}\right) w^T x_n + \left(\frac{1+\rho_-}{2}\right) b + \frac{1-\rho_-}{2}$$

From the derivation above, we see that

$$w' = \frac{1+\rho_-}{2} w$$

$$b' = \left(\frac{1+\rho_-}{2}\right) b + \frac{1-\rho_-}{2}$$

And if we focus on the w part, we can find the following relationship, (convert the notation α_n to be α_n^*)

$$\sum_{sv} \alpha'_n y_n x_n = \frac{1+\rho_-}{2} \sum_{sv} \alpha_n^* y_n x_n$$

To this end, we can roughly say that α' should be equal to $\frac{(1+\rho_-)}{2} \alpha^*$.

#

Q6

If we take $Q=2$ for example, we can decompose this homogeneous polynomial kernel into $\Phi(x)^T \Phi(x')$ which $\Phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$.

If observe more, we can find that the dimension of $\Phi(x)$ follows the rule of combination with repetition, that is, we choose Q terms from d types of elements, and leads to $\tilde{d} = \binom{Q+d-1}{Q}$. #

For instance,

$Q=2, d=2$

$\Phi((x_1, x_2)) = (x_1^2, x_1x_2, x_2^2)$, $\tilde{d} = \binom{2+2-1}{2} = 3$

ignore coefficient

$Q=2, d=3$

$\Phi((x_1, x_2, x_3)) = (x_1^2, x_2^2, x_3^2, x_1x_2, x_2x_3, x_1x_3)$,

ignore coefficient

$\tilde{d} = \binom{2+3-1}{2} = 6$

Q7

$$\|\phi(x) - \phi(x')\|^2$$

$$= (\phi(x) - \phi(x'))^T (\phi(x) - \phi(x'))$$

$$= (\phi(x)^T - \phi(x')^T)(\phi(x) - \phi(x'))$$

$$= \phi(x)^T\phi(x) - \phi(x)^T\phi(x') - \phi(x')^T\phi(x) + \phi(x')^T\phi(x')$$

$$K_2(x,x') = (1+x^Tx')^2, \quad (\text{min. max}) = (0, 4)$$

$$K_2(x,x) = (1+x^Tx)^2 = 4$$

$$= K_2(x,x) + K_2(x',x') - 2K_2(x,x')$$

$$= 8 - 2K_2(x,x')$$

$$\Rightarrow 0 \le K_2(x,x') \le 4$$

$$-8 \le -2K_2(x,x') \le 0$$

$$0 \le 8 - 2K_2(x,x') \le \textcircled{8} \#$$

→ tightest upper bound

Q8

$$w_{t+1} = w_t + y_{n(t)}\,\phi(x_{n(t)}) \quad \text{— ①}$$

$$w_t = \sum_{n=1}^{N} \alpha_t[n]\,\phi(x_n) \quad \text{②}, \qquad \alpha_t = (\alpha_t[1], \ldots, \alpha_t[n])$$

$$\alpha_0 = 0 \ (N \text{ zeros}) \longrightarrow w_0 = 0 \ (\tilde{d}+1 \text{ zeros})$$

Explain by illustration

If w_0 makes a mistake on $(\phi(x_{1(0)}), y_{1(0)})$,

$$\boxed{w_1} = w_0 + y_{1(0)}\,\phi(x_{1(0)})$$

Due to ②, $\boxed{w_1} = \sum_{n=1}^{N} \alpha_1[n]\,\phi(x_n)$.

Hence, α_1 should be $(\underbrace{0 + y_{1(0)}}_{1^{st}\text{ term}}, \underbrace{0, \ldots, 0}_{N-1\text{ zeros}})$.

If w_1 keep making new mistake on $(\phi(x_{3(1)}), y_{3(1)})$,

$$w_2 = w_1 + y_{3(1)}\,\phi(x_{3(1)})$$

Follow the same rule above,

α_2 now should be $(\underline{0 + y_{1(0)}}, 0, \underline{0 + y_{3(1)}}, \ldots, 0)$

Through the observation above, we can find that in PLA, when the current w_t makes a mistake on $(\phi(x_{n(t)}), y_{n(t)})$, the $\alpha_{t+1}[n(t)]$ term should be updated by adding the original $\alpha_t[n(t)]$ and current $y_{n(t)}$. And this updating way will make $w_{t+1} = w_t + y_{n(t)}\phi(x_{n(t)})$ equivalent to $w_{t+1} = \sum_{n=1}^{N} \alpha_{t+1}[n]\phi(x_n)$.

Q9

Let $z_n = \phi(x_n)$

Write Lagarange function:

$$L(b, w, \xi, \alpha, \beta)$$

$$= \frac{1}{2} w^T w + \sum_{n=1}^{N} u_n \cdot \xi_n$$

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

$$+ \sum_{n=1}^{N} \beta_n \cdot (-\xi_n)$$

Due to strong duality:

$$\min_{b,w,\xi} \left(\max_{\alpha_n \geq 0, \beta_n \geq 0} L(b, w, \xi, \alpha, \beta) \right)$$

$$\Leftrightarrow \max_{\alpha_n \geq 0, \beta_n \geq 0} \left(\min_{b,w,\xi} L(b, w, \xi, \alpha, \beta) \right)$$

$\frac{\partial L}{\partial \xi_n} = 0$:

$$u_n - \alpha_n - \beta_n = 0$$

We can rewrite L as :

$$L(b, w, \xi, \alpha, \beta)$$

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

$$\boxed{+ \sum_{n=1}^{N} \underbrace{(u_n - \alpha_n - \beta_n)}_{=0} \xi_n}$$

↘ can be removed without loss of optimality

Rewrite dual problems as :

$$\max_{0 \le \alpha_n \le u_n,\ \beta_n = u_n - \alpha_n} \left(\underbrace{\min_{b, w} \ \frac{1}{2} w^T w + \sum_{n=1}^{N} \alpha_n \cdot (1 - y_n(w^T z_n + b))}_{\Rightarrow \text{inner problem same as hard-margin SVM}} \right)$$

Due to the same inner problem as hard-margin SVM, we can derive the standard dual using the same steps, for which the objective function will be $\frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \alpha_n \alpha_m y_n y_m z_n^T z_m - \sum_{n=1}^{N} \alpha_n$

↘ ◇ #

Q10

Within $(0, 1)$, calculate uniformly-averaged squared difference between E_{smooth} and E_{hinge}.

uniformly $\rightarrow$ uniformly sampling

$$\int_0^1 (E_{hinge} - E_{smooth})^2 \underbrace{f(\rho)}_{\rightarrow = 1 \text{ (probability density)}} d\rho$$

$$= \int_0^1 (\underbrace{1-\rho}_{E_{hinge} \text{ in } (0,1)} - \underbrace{\frac{1}{2}(1-\rho)^2}_{\rightarrow E_{smooth} \text{ in } (0,1)})^2 d\rho$$

$$= \int_0^1 (\frac{1}{2} - \frac{1}{2}\rho^2)^2 d\rho$$

$$= \frac{1}{4} \int_0^1 (1 - 2\rho^2 + \rho^4) d\rho$$

$$= \frac{1}{4} (\rho - \frac{2}{3}\rho^3 + \frac{1}{5}\rho^5 \Big|_0^1)$$

$$= \frac{1}{4} \times \frac{8}{15}$$

$$= \boxed{\frac{2}{15}} \#$$

Q11 - code

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 26 07:01:01 2021

@author: Eric
"""

import numpy as np
import pandas as pd
from libsvm.svmutil import *


# %% Q11 - Main
# read data
trainY, trainX = svm_read_problem("satimage.scale.txt", return_scipy=True)

# re-label
trainY[trainY != 5] = -1
trainY[trainY == 5] = 1

# training
m = svm_train(trainY, trainX, "-s 0 -t 0 -c 10")

# retreive support vectors
sv = m.get_SV()
sv = pd.DataFrame(sv)
sv = sv.sort_index(axis=1)

# replace nan values with zeros
sv = sv.fillna(0)

# retreive support vector coefficients
svCoef = m.get_sv_coef()
svCoef = np.array(svCoef)

# calculate weight and its norm
w = svCoef.T @ sv
w = w.values.ravel()
wNorm = np.sum(w**2) ** (0.5)
print("wNorm:", wNorm)  # 4.645
```

```
In [1]: runfile('C:/Users/Eric/Desktop/ml21fall/hw5/Q11.py', wdir='C:/Users/Eric/Desktop/ml21fall/hw5')
wNorm: 4.644563442136298
```

Q11 - result

Q12 - code

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 26 09:09:43 2021

@author: Eric
"""

import numpy as np
import copy
from libsvm.svmutil import *


# %% Q12 - Main
# read data
trainY, trainX = svm_read_problem("satimage.scale.txt", return_scipy=True)

# iterate over each classifier
for target in [2, 3, 4, 5, 6]:
    # re-label
    currentY = copy.deepcopy(trainY)
    currentY[currentY != target] = -1
    currentY[currentY == target] = 1

    # training
    m = svm_train(currentY, trainX, "-s 0 -t 1 -d 3 -g 1 -r 1 -c 10")

    # show acc and Ein information
    print("Target:", target)
    p_label, p_acc, p_val = svm_predict(currentY, trainX, m)
    Ein = 100 - p_acc[0]
    print("Ein: {}%".format(np.round(Ein, 2)))
    print()
```

```
In [1]: runfile('C:/Users/Eric/Desktop/ml21fall/hw5/Q12.py', wdir='C:/Users/Eric/Desktop/ml21fall/hw5')
Target: 2
Accuracy = 100% (4435/4435) (classification)
Ein: 0.0%

Target: 3
Accuracy = 99.7294% (4423/4435) (classification)
Ein: 0.27%

Target: 4
Accuracy = 99.1657% (4398/4435) (classification)
Ein: 0.83%

Target: 5
Accuracy = 100% (4435/4435) (classification)
Ein: 0.0%

Target: 6
Accuracy = 99.7069% (4422/4435) (classification)
Ein: 0.29%
```

Q12 - result

Q13 - code

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 26 09:26:03 2021

@author: Eric
"""

import numpy as np
import copy
from libsvm.svmutil import *


# %% Q13 - Main
# read data
trainY, trainX = svm_read_problem("satimage.scale.txt", return_scipy=True)

# iterate over each classifier
for target in [2, 3, 4, 5, 6]:
    # re-label
    currentY = copy.deepcopy(trainY)
    currentY[currentY != target] = -1
    currentY[currentY == target] = 1

    # training
    m = svm_train(currentY, trainX, "-s 0 -t 1 -d 3 -g 1 -r 1 -c 10")

    # show number of support vectors
    print("Target:", target)
    p_label, p_acc, p_val = svm_predict(currentY, trainX, m)
    svNum = m.get_nr_sv()
    print("svNum:", svNum)
    print()
```

```
In [1]: runfile('C:/Users/Eric/Desktop/ml21fall/hw5/Q13.py', wdir='C:/Users/Eric/Desktop/ml21fall/hw5')
Target: 2
Accuracy = 100% (4435/4435) (classification)
svNum: 93

Target: 3
Accuracy = 99.7294% (4423/4435) (classification)
svNum: 385

Target: 4
Accuracy = 99.1657% (4398/4435) (classification)
svNum: 659

Target: 5
Accuracy = 100% (4435/4435) (classification)
svNum: 281

Target: 6
Accuracy = 99.7069% (4422/4435) (classification)
svNum: 607
```

Q13 - result

Q14 - code

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 26 09:28:39 2021

@author: Eric
"""

import numpy as np
import copy
from libsvm.svmutil import *


# %% Q14 - Main
# read data
trainY, trainX = svm_read_problem("satimage.scale.txt", return_scipy=True)
testY, testX = svm_read_problem("satimage.scale.t.txt", return_scipy=True)

# re-label
trainY[trainY != 1] = -1
trainY[trainY == 1] = 1
testY[testY != 1] = -1
testY[testY == 1] = 1

# iterate over each C parameter
for C in [0.01, 0.1, 1, 10, 100]:
    # training
    m = svm_train(trainY, trainX, "-s 0 -t 2 -g 10 -c {}".format(C))

    # show the current C and corresponding Eout
    print("C:", C)
    p_label, p_acc, p_val = svm_predict(testY, testX, m)
    Eout = 100 - p_acc[0]
    print("Eout: {}%".format(np.round(Eout, 2)))
    print()
```

```
In [1]: runfile('C:/Users/Eric/Desktop/ml21fall/hw5/Q14.py', wdir='C:/Users/Eric/Desktop/ml21fall/hw5')
C: 0.01
Accuracy = 76.95% (1539/2000) (classification)
Eout: 23.05%

C: 0.1
Accuracy = 79.25% (1585/2000) (classification)
Eout: 20.75%

C: 1
Accuracy = 89.05% (1781/2000) (classification)
Eout: 10.95%

C: 10
Accuracy = 89.95% (1799/2000) (classification)
Eout: 10.05%

C: 100
Accuracy = 89.95% (1799/2000) (classification)
Eout: 10.05%
```

Q14 - result

Q15 - code

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 26 09:40:46 2021

@author: Eric
"""

import numpy as np
import copy
from libsvm.svmutil import *


# %% Q15 - Main
# read data
trainY, trainX = svm_read_problem("satimage.scale.txt", return_scipy=True)
testY, testX = svm_read_problem("satimage.scale.t.txt", return_scipy=True)

# re-label
trainY[trainY != 1] = -1
trainY[trainY == 1] = 1
testY[testY != 1] = -1
testY[testY == 1] = 1

# iterate over each gamma parameter
for gamma in [0.1, 1, 10, 100, 1000]:
    # training
    m = svm_train(trainY, trainX, "-s 0 -t 2 -g {} -c 0.1".format(gamma))

    # show the current gamma and corresponding Eout
    print("Gamma:", gamma)
    p_label, p_acc, p_val = svm_predict(testY, testX, m)
    Eout = 100 - p_acc[0]
    print("Eout: {}%".format(np.round(Eout, 2)))
    print()
```

```
In [1]: runfile('C:/Users/Eric/Desktop/ml21fall/hw5/Q15.py', wdir='C:/Users/Eric/Desktop/ml21fall/hw5')
Gamma: 0.1
Accuracy = 98.75% (1975/2000) (classification)
Eout: 1.25%

Gamma: 1
Accuracy = 98.8% (1976/2000) (classification)
Eout: 1.2%

Gamma: 10
Accuracy = 79.25% (1585/2000) (classification)
Eout: 20.75%

Gamma: 100
Accuracy = 76.95% (1539/2000) (classification)
Eout: 23.05%

Gamma: 1000
Accuracy = 76.95% (1539/2000) (classification)
Eout: 23.05%
```

Q15 - result

Q16 - code

```python
# -*- coding: utf-8 -*-
"""
Created on Sun Dec 26 09:49:32 2021

@author: Eric
"""

import numpy as np
from libsvm.svmutil import *


# %% Q16 - Main
# read data
trainY, trainX = svm_read_problem("satimage.scale.txt")

# re-label
trainX = np.array(trainX)
trainY = np.array(trainY)
trainY[trainY != 1] = -1
trainY[trainY == 1] = 1

# set gamma candidates
gammaSet = [0.1, 1, 10, 100, 1000]

# container for storing best gamma in each experiment
bestGamma = []

np.random.seed(5)  # compare with multi-thread
for expIdx in range(1000):
    print("============ exp: {}/1000 ============".format(expIdx))

    # randomly samples 200 examples for creating Dtrain and Dval
    idxSet = np.random.choice(np.arange(trainX.size), size=200, replace=False)
    DtrainX = list(np.delete(trainX, idxSet, 0))
    DtrainY = list(np.delete(trainY, idxSet))
    DvalX = list(trainX[idxSet])
    DvalY = list(trainY[idxSet])

    # container for storing Eval corresponding to each gamma
    EvalSet = []

    # iterate over each gamma parameter
    for gamma in gammaSet:
        # training
        m = svm_train(DtrainY, DtrainX, "-s 0 -t 2 -g {} -c 0.1".format(gamma))

        # evaluating Eval
        p_label, p_acc, p_val = svm_predict(DvalY, DvalX, m)
        EvalSet.append(100 - p_acc[0])

    # storing the best gamma in the current experiment
    bestGamma.append(gammaSet[np.argmin(EvalSet)])

# analyze the selected number of each gamma
unique, counts = np.unique(bestGamma, return_counts=True)
result = dict(zip(unique, counts))
print("result:", result)
```

```
result: {0.1: 736, 1.0: 264}
```

Q1b - result