# Homework #5

RELEASE DATE (NON-PROGRAMMING): 12/14/2021

RELEASE DATE (PROGRAMMING): 12/15/2021

RED CORRECTION (WITH SOME FORMATTING FIX) DATE: 12/15/2021 10:30

DUE DATE: 01/06/2022, BEFORE 13:00 on Gradescope

QUESTIONS ARE WELCOMED ON THE NTU COOL FORUM.

*You will use Gradescope to upload your choices and your scanned/printed solutions. For problems marked with (*), please follow the guidelines on the course website and upload your source code to Gradescope as well. Any programming language/platform is allowed.*

*Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.*

*Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.*

*Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.*

*You should write your solutions in English or Chinese with the common math notations introduced in class or in the problems. We do not accept solutions written in any other languages.*

This homework set comes with 16 problems and a total of 400 points. For each problem, there is one correct choice. If you choose the correct answer, you get 20 points; if you choose an incorrect answer, you get 0 points. For four of the secretly-selected problems, the TAs will grade your detailed solution in terms of the written explanations and/or code based on how logical/clear your solution is. Each of the four problems graded by the TAs counts as additional 20 points (in addition to the correct/incorrect choices you made). In general, each homework (except homework 0) is of a total of 400 points.

## Hard-Margin SVM

**1.** Consider $N$ "linearly separable" 1D examples $\{(x_n, y_n)\}_{n=1}^{N}$. That is, $x_n \in \mathbb{R}$. Without loss of generality, assume that $x_1 \leq x_2 \leq \ldots x_M < x_{M+1} \leq x_{M+2} \ldots \leq x_N$, $y_n = -1$ for $n = 1, 2, \ldots, M$, and $y_n = +1$ for $n = M+1, M+2, \ldots, N$. Which of the following represents a large-margin separating hyperplane? Choose the correct answer; explain your answer.

a

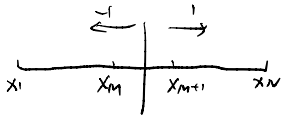
**[a]** $w = 1,\ b = -\frac{x_{M+1}+x_M}{2}$

**[b]** $w = -1,\ b = \frac{x_{M+1}+x_M}{2}$

**[c]** $w = 1,\ b = -\frac{x_{M+1}-x_M}{2}$

**[d]** $w = -1,\ b = \frac{x_{M+1}-x_M}{2}$

**[e]** none of the other choices




$x_M - \frac{x_{M+1}+x_M}{2}$

$= \frac{x_M - x_{M+1}}{2} < 0$

$w\frac{x_M+x_{M+1}}{2} + b = 0$

$\therefore \begin{cases} w=1 \\ b = -\frac{x_{M+1}+x_M}{2} \end{cases}$

$d = \frac{w^T x + b}{\|w\|} = \frac{y(w^T x + b)}{\|w\|}$

(*Hint: The hard-margin SVM gets a specially-scaled version of the solution above.*)

**2.** Following the notations in the lecture for the hard-margin SVM in the $\mathcal{Z}$-space. At the optimal $(b, \mathbf{w})$ and $\boldsymbol{\alpha}$, how many of the following values are equal to the length of margin (the distance between the closest example and the decision boundary)? Choose the correct answer; explain why the values equal the margin in your chosen answer.

C

(1) $\|\mathbf{w}\|^{-1/2}$ ✗

(2) $2\|\mathbf{w}\|^{-1}$ ✓

(3) $\|\mathbf{w}\|^{-1}$ ✗

(4) $(\sum_{n=1}^{N} \alpha_n)^{-1/2}$ ✗

(5) $\sum_{n=1}^{N}[\alpha_n = 1]$ ✗ $[\alpha_n = -1]$ left out

(6) $(2\sum_{n=1}^{N} \alpha_n - \|\sum_{n=1}^{N} \alpha_n y_n \mathbf{z}_n\|^2)^{-1/2}$ ✓

[a] 0

[b] 1

[c] 2

[d] 3

[e] 4

**3.** Sometimes we hope to achieve a smaller margin for the positive examples, and a bigger margin for the negative ones. For instance, if we have very few negative examples on hand, we may hope to give them a larger margin to better protect them from noise. Consider an *uneven-margin* SVM that solves

$$\begin{aligned} \min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & \Big(\mathbf{w}^T\mathbf{x}_n + b\Big) \geq 1 \text{ for } y_n > 0, \\ & -\Big(\mathbf{w}^T\mathbf{x}_n + b\Big) \geq \rho_- \text{ for } y_n < 0. \end{aligned}$$

Consider the following examples

$$\begin{aligned} \mathbf{x}_1 &= (0, 1) & y_1 &= -1 \\ \mathbf{x}_2 &= (0, 0) & y_2 &= -1 \\ \mathbf{x}_3 &= (0, -1) & y_3 &= -1 \\ \mathbf{x}_4 &= (1, 0) & y_4 &= +1 \end{aligned}$$

Take $\rho_- = 4$. What is the optimal $\mathbf{w}$ and $b$? Choose the correct answer; explain your answer. (Note: You can calculate your answer with a QP solver if you want, but you need to "explain" the solution that was found. We suggest you to visualize what happens.)

C

[a] the optimal $\mathbf{w} = (1, 0), b = -1$

[b] the optimal $\mathbf{w} = (2, 0), b = -1$

[c] the optimal $\mathbf{w} = (5, 0), b = -4$

[d] the optimal $\mathbf{w} = (\frac{1}{5}, 0), b = -4$

[e] none of the other choices

**4.** The dual problem of the uneven-margin SVM defined in Problem 3 can be written as

$$\min_{\boldsymbol{\alpha}} \quad \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m \mathbf{x}_n^T\mathbf{x}_m + \square$$
$$\text{subject to} \quad \sum_{n=1}^{N} y_n\alpha_n = 0$$
$$\alpha_n \geq 0 \text{ for } n = 1, 2, \ldots, N.$$

c

What is $\square$? Choose the correct answer; explain your answer.

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

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

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

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

**[e]** none of the other choices

**5.** Let $\boldsymbol{\alpha}^*$ be an optimal solution of the original hard-margin SVM (i.e. even margin). Which of the following is an optimal solution of the uneven-margin SVM for a given $\rho_-$? Choose the correct answer; explain your answer.

d

**[a]** $\boldsymbol{\alpha}^*$

**[b]** $\sqrt{\rho_-}\boldsymbol{\alpha}^*$

**[c]** $\frac{2}{1+\rho_-}\boldsymbol{\alpha}^*$

**[d]** $\frac{1+\rho_-}{2}\boldsymbol{\alpha}^*$

**[e]** none of the other choices

## Kernels

**6.** Kernels are able to embed high-dimensional feature spaces. Consider the homogeneous polynomial kernel with degree $Q$,

$$K(\mathbf{x}, \mathbf{x}') = \left(\mathbf{x}^T\mathbf{x}'\right)^Q,$$

where each $\mathbf{x}$ and $\mathbf{x}'$ is in $\mathbb{R}^d$, *without* padding $x_0 = 1$. Now, decompose $K(\mathbf{x}, \mathbf{x}')$ as some $\boldsymbol{\Phi}(\mathbf{x})^T\boldsymbol{\Phi}(\mathbf{x}')$, where $\boldsymbol{\Phi}(\mathbf{x})$ includes *unique* terms calculated from $\mathbf{x}$. That is, $x_3x_5$ would be considered the same term as $x_5x_3$ (*Note: this is different from the* $\boldsymbol{\Phi}(\mathbf{x})$ *that we considered in class*). What is dimension of $\boldsymbol{\Phi}(\mathbf{x})$? Choose the correct answer; explain your answer.

a

**[a]** $\binom{Q+d-1}{Q}$

**[b]** $\binom{Q+d-1}{d}$

**[c]** $\binom{Q+d}{Q}$

**[d]** $\binom{Q+d}{d}$

**[e]** none of the other choices

**7.** For any feature transform $\mathbf{\Phi}$ from $\mathcal{X}$ to $\mathcal{Z}$, the squared distance between two examples $\mathbf{x}$ and $\mathbf{x}'$ is $\|\mathbf{\Phi}(\mathbf{x}) - \mathbf{\Phi}(\mathbf{x}')\|^2$ in the $\mathcal{Z}$-space. The distance can be computed with the kernel trick. Consider the degree-2 quadratic kernel $K_2(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T\mathbf{x}')^2$. What is the tightest upper bound for the squared distance between two *unit vectors* $\mathbf{x}$ and $\mathbf{x}'$ in the $\mathcal{Z}$-space? Choose the correct answer; explain your answer.

d

$K_2(x, x') = (1 + x^T x')^2$

$\|\phi(x) - \phi(x')\|^2$ in Z-space

upper bound

$\|\phi(x) - \phi'(x)\|^2 \le (2 \cdot \sqrt{2})^2 = 8$

[a] 0
[b] 1
[c] 2
[d] 8
[e] 1126

## Kernel Perceptron Learning Algorithm

**8.** In this problem, we are going to apply the kernel trick to the perceptron learning algorithm. If we run the perceptron learning algorithm on the transformed examples $\{(\boldsymbol{\phi}(\mathbf{x}_n), y_n)\}_{n=1}^N$, the algorithm updates $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$ when the current $\mathbf{w}_t$ makes a mistake on $\left(\boldsymbol{\phi}(\mathbf{x}_{n(t)}), y_{n(t)}\right)$:

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + y_{n(t)}\boldsymbol{\phi}(\mathbf{x}_{n(t)})$$

C

Because every update is based on one (transformed) example, if we take $\mathbf{w}_0 = \mathbf{0}$, we can represent every $\mathbf{w}_t$ as a linear combination of $\{\boldsymbol{\phi}(\mathbf{x}_n)\}_{n=1}^N$. We can then maintain the linear combination coefficients instead of the whole $\mathbf{w}$. Assume that we maintain an $N$-dimensional vector $\boldsymbol{\alpha}_t$ in the $t$-th iteration such that

$$\mathbf{w}_t = \sum_{n=1}^{N} \boldsymbol{\alpha}_t[n]\boldsymbol{\phi}(\mathbf{x}_n)$$

for $t = 0, 1, 2, \ldots$, where $\boldsymbol{\alpha}_t[n]$ indicates the $n$-th component of $\boldsymbol{\alpha}_t$. Set $\boldsymbol{\alpha}_0 = \mathbf{0}$ ($N$ zeros) to match $\mathbf{w}_0 = \mathbf{0}$ ($\tilde{d} + 1$ zeros). What should $\boldsymbol{\alpha}_{t+1}[n(t)]$ be when the current $\mathbf{w}_t$ (represented by $\boldsymbol{\alpha}_t$) makes a mistake on $\left(\boldsymbol{\phi}(\mathbf{x}_{n(t)}), y_{n(t)}\right)$? Choose the correct answer; explain your answer.

$\alpha_t[n(t)] + y_{n(t)}$ for updating PLA

$w_{t+1} \leftarrow w_t + y_{n(t)} \phi(x_{n(t)})$

$w_t = \sum_{n=1}^{N} \alpha_t[n] \phi(x_n)$

$\alpha_0 = 0$ $w_t = \sum_{n=1}^{N} \alpha_t[n] \phi(x_n)$

[a] $\boldsymbol{\alpha}_t[n(t)] + 1$
[b] $\boldsymbol{\alpha}_t[n(t)] - 1$
[c] $\boldsymbol{\alpha}_t[n(t)] + y_{n(t)}$
[d] $\boldsymbol{\alpha}_t[n(t)] - y_{n(t)}$
[e] none of the other choices

## Soft-Margin SVM

a

**9.** Suppose we want to emphasize that some training examples are more important than others. Formally, consider a data set $\mathcal{D} = \{(\mathbf{x}_n, y_n, u_n)\}_{n=1}^N$, where $u_n$ is a non-negative weight that indicates the importance of the $n$-th example. The soft-margin SVM with this *weighted* classification problem solves the following constrained optimization problem:

$u_n$ be the soft factor

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{n=1}^{N} u_n \cdot \xi_n \\ \text{subject to} \quad & y_n\Big(\mathbf{w}^T\boldsymbol{\Phi}(\mathbf{x}_n) + b\Big) \geq 1 - \xi_n \\ & \xi_n \geq 0, \quad n = 1, \ldots, N. \end{aligned}$$

We can then derive the dual version of the *weighted* soft-margin SVM problem that involves only $\boldsymbol{\alpha}$, $\mathbf{y}$, X, and $\mathbf{u}$:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \Diamond \\ \text{subject to} \quad & \sum_{n=1}^{N} y_n\alpha_n = 0 \\ & 0 \leq \alpha_n \leq u_n \text{ for } n = 1, 2, \cdots, N \end{aligned}$$

Which of the following is the correct form of $\Diamond$? Choose the correct answer; explain your answer.

only change on the condition similar to hard-margin SVM

[a] $\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m \boldsymbol{\Phi}(\mathbf{x}_n)^T\boldsymbol{\Phi}(\mathbf{x}_m) - \sum_{n=1}^{N}\alpha_n$

[b] $\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m u_n u_m \boldsymbol{\Phi}(\mathbf{x}_n)^T\boldsymbol{\Phi}(\mathbf{x}_m) - \sum_{n=1}^{N}\alpha_n$

[c] $\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m u_n u_m \boldsymbol{\Phi}(\mathbf{x}_n)^T\boldsymbol{\Phi}(\mathbf{x}_m) - \sum_{n=1}^{N} u_n\alpha_n$

[d] $\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m \boldsymbol{\Phi}(\mathbf{x}_n)^T\boldsymbol{\Phi}(\mathbf{x}_m) - \sum_{n=1}^{N} u_n\alpha_n$

[e] none of the other choices

**10.** As discussed in class, the primal optimization problem for the soft-margin SVM is equivalent to the following unconstrained optimization problem.

$$\min_{b,\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{n=1}^{N}\max(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b), 0)$$

Let $s_n = \mathbf{w}^T\mathbf{x}_n + b$ and $\rho_n = y_n \cdot s_n$. The error function $E_{\text{hinge}}(\rho) = \max(1 - \rho, 0)$ is widely known as the hinge error. The hinge error is convex but is not differentiable everywhere. Therefore, it can be technically complicated to run gradient descent on the error. A possible workaround is to approximate the hinge error with a "smooth hinge error." A good candidate of "smooth hinge error" is the following function

e

$$E_{\text{smooth}}(\rho) = \begin{cases} 0 & \rho \geq 1 \\ (1-\rho)^2 & 0 < \rho < 1 \\ 0.5 - \rho & \rho \leq 0 \end{cases}$$

Since $E_{\text{smooth}}$ is differentiable everywhere, we can then apply gradient descent and related algorithms. $E_{\text{smooth}}$ has a constant slope of $-1$ for all $\rho \leq 0$ and is 0 for all $\rho \geq 1$, just like $E_{\text{hinge}}$. Now, within $(0, 1)$, what is the uniformly-averaged squared difference between $E_{\text{smooth}}$ and $E_{\text{hinge}}$? Choose the correct answer; explain your answer.

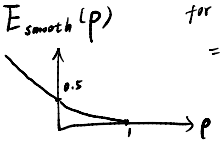
[a] $\frac{1}{15}$

[b] $\frac{1}{24}$

[c] $\frac{1}{30}$

[d] $\infty$

[e] none of the other choices



## Experiments with Soft-Margin SVM

For Problems 11 to 16, we are going to experiment with a real-world data set. Download the processed satimage data sets from LIBSVM Tools.

Training: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale`

Testing: `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale.t`

We will consider binary classification problems of the form "one of the classes" (as the positive class) versus "the other classes" (as the negative class).

The data set contains thousands of examples, and some quadratic programming packages cannot handle this size. We recommend that you consider the LIBSVM package

`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Regardless of the package that you choose to use, please read the manual of the package carefully to make sure that you are indeed solving the soft-margin support vector machine taught in class like the dual formulation below:

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(\mathbf{x}_n,\mathbf{x}_m) - \sum_{n=1}^{N}\alpha_n \\ \text{subjectto} \quad & \sum_{n=1}^{N} y_n\alpha_n = 0 \\ & 0 \le \alpha_n \le C \quad n = 1,\ldots,N. \end{aligned}$$

In the following problems, please use the 0/1 error for evaluating $E_{\text{in}}$, $E_{\text{val}}$ and $E_{\text{out}}$ (through the test set). Some practical remarks include

(i) Please tell your chosen package to **not** automatically scale the data for you, lest you should change the effective kernel and get different results.

(ii) It is your responsibility to check whether your chosen package solves the designated formulation with enough numerical precision. Please read the manual of your chosen package for software parameters whose values affect the outcome—any ML practitioner needs to deal with this kind of added uncertainty.

**11.** (*) Consider the linear soft-margin SVM. That is, either solve the primal formulation of soft-margin SVM with the given $\mathbf{x}_n$, or take the linear kernel $K(\mathbf{x}_n,\mathbf{x}_m) = \mathbf{x}_n^T\mathbf{x}_m$ in the dual formulation. With $C = 10$, and the binary classification problem of "5" versus "not 5", which of the following numbers is closest to $\|\mathbf{w}\|$ after solving the linear soft-margin SVM? Choose the closest answer; provide your command/code.

based on code

a

**[a]** 4.5

**[b]** 5.0

**[c]** 5.5

**[d]** 6.0

**[e]** 6.5

**12.** (*) Consider the polynomial kernel $K(\mathbf{x}_n, \mathbf{x}_m) = (1 + \mathbf{x}_n^T \mathbf{x}_m)^Q$, where $Q$ is the degree of the polynomial. With $C = 10$, $Q = 3$, which of the following soft-margin SVM classifiers reaches the largest $E_{\text{in}}$? Choose the correct answer; provide your command/code.

c based on code

[a] "2" versus "not 2"
[b] "3" versus "not 3"
[c] "4" versus "not 4"
[d] "5" versus "not 5"
[e] "6" versus "not 6"

**13.** (*) Following Problem 12, which of the following numbers is closest to the maximum number of support vectors within those five soft-margin SVM classifiers? Choose the closest answer; provide your command/code.

e

[a] 450
[b] 500
[c] 550
[d] 600
[e] 650

**14.** (*) Consider the Gaussian kernel $K(\mathbf{x}_n, \mathbf{x}_m) = \exp\left(-\gamma ||\mathbf{x}_n - \mathbf{x}_m||^2\right)$. For the binary classification problem of "1" versus "not 1", when fixing $\gamma = 10$, which of the following values of $C$ results in the lowest $E_{\text{out}}$? If there is a tie, please pick the smallest $C$. Choose the correct answer; provide your command/code.

d

[a] 0.01
[b] 0.1
[c] 1
[d] 10
[e] 100

**15.** (*) Following Problem 14, when fixing $C = 0.1$, which of the following values of $\gamma$ results in the lowest $E_{\text{out}}$? If there is a tie, please pick the smallest $\gamma$. Choose the correct answer; provide your command/code.

b

[a] 0.1
[b] 1
[c] 10
[d] 100
[e] 1000

**16.** (*) Following Problem 14 and consider a validation procedure that randomly samples 200 examples from the training set for validation and leaves the other examples for training $g_{\text{SVM}}^-$. Fix $C = 0.1$ and use the validation procedure to choose the best $\gamma$ among $\{0.1, 1, 10, 100, 1000\}$ according to $E_{\text{val}}$. If there is a tie of $E_{\text{val}}$, choose the smallest $\gamma$. Repeat the procedure 1000 times. Which of the following values of $\gamma$ is selected the most number of times? Choose the correct answer; provide your command/code.

a

[a] 0.1
[b] 1
[c] 10
[d] 100
[e] 1000

```
import numpy as np
from sklearn import svm
from libsvm svmutil import *
import scipy
import matplotlib pyplot as plt #導入繪圖包，使得結果更加直觀
#數據預處理
#train
y_train Xtrain=                    'satimage.scale.txt' return_scipy=True
#test
y_test Xtest=                    'satimage.scale.t' return_scipy=True
X_train=Xtrain
X_test=Xtest

#Problem11
#訓練模型
y_train_5 =  y_train==5          "int"
C =  1
W =
for i in C
    c = 10 ** i
    clf = svm SVC kernel="linear"  C=c
    clf fit X_train  y_train_5
    w = clf coef_
    W append np linalg norm w
W

#Problem 12 and Problem 13
```

```
#將標籤修改為-1, 1
y_trian_2 = 2 *  y_train==2            "int"  – 1
##y_trian_2 = 2 * (y_train==3).astype("int") – 1
#實際為y_trian_3
##y_trian_2 = 2 * (y_train==4).astype("int") – 1
#實際為y_trian_4
##y_trian_2 = 2 * (y_train==5).astype("int") – 1
#實際為y_trian_5
##y_trian_2 = 2 * (y_train==6).astype("int") – 1
#實際為y_trian_6
C =  1
Ein =
alpha =
for i in C
    c = 10 ** i
    clf = svm SVC kernel='poly'  degree=3  coef0=1
gamma=1  C=c
    clf fit X_train  y_trian_2
    e = np mean clf predict X_train  != y_trian_2
    #支持向量的索引
    support = clf support_
    #计算系数
    coef = np sum clf dual_coef_ 0  * y_trian_2 support
    alpha append coef
    Ein append e
Ein
#回答Problem 12
support
#觀察support維度來回答Problem 13

#Problem 14
C =  -2  -1  0  1  2
Distance =
```

```
y_train_1 =  y_train==1              "int"
#將標籤修改為-1, 1
y = 2 * y_train_1 - 1
for i in C
    c = 10**i
    clf = svm SVC kernel='rbf'  gamma=10  C=c
    clf fit X_train  y
    X = X_train clf support_
    #距離矩陣
    d1 = np sum X ** 2  axis=1               -1  1
    d2 = np sum X ** 2  axis=1               1  -1
    dist = d1 + d2 - 2 * X       X
    #Kernel矩陣
    K = np       - c * dist
    #計算anyn
    y1 = clf dual_coef_ 0  * y clf support_
    w2 = y1       K        y1
    #計算距離
    distance = 1 / np       w2
    Distance append distance
plt plot  C  Distance
plt title "$\log_{10}C$ VS distance"  #以10為底的橫坐標
plt show
#觀察圖像最低點

#Problem 15
y_test_1 =  y_test == 1

Gamma =  -1 0 1 2 3
Eout =
for i in Gamma
    gamma = 10**i
```

```
    clf = svm SVC kernel='rbf'  gamma=gamma  C=0.1
    clf fit X_train  y_train_1
    e = np mean  clf predict X_test   != y_test_1
    Eout append e
plt plot  Gamma  Eout
plt title "$\log_{10}\gamma$ VS $E_{out}$"
plt show   #圖形更加直觀

#Problem 16
from sklearn model_selection import train_test_split
#合併資料，以便呼叫train_test_split函式
Data = np c_  X_train  y_train
N = 100
#記錄最小的Eval對應gamma的索引次數
Cnt = np zeros  5   #初始化
Gamma =   -1 0 1 2 3
for _ in range  N
    #劃分資料
     train_set  val_set = train_test_split Data
test_size=0.05
     #200筆sample
    #取feature
    X_train = train_set      36
    #取label
    y_train = train_set     36
    X_val = val_set      36
    y_val = val_set     36
     Eval =
    for i in Gamma
        gamma = 10**i
        clf = svm SVC kernel='rbf'  gamma=gamma
C=0.1
        clf fit  X_train  y_train            'int'
```

```
        e = np.mean(clf.predict(X_val) != y_val)
        Eval.append(e)
    #找到最小Eval對應的索引
    index = np.argmin(Eval)
    #索引次數加1
    Cnt[index] += 1
#觀察圖像得到最高頻次gamma
plt.bar(Gamma, Cnt)
plt.show()
```