Hard-Margin SVM

1. ANS [a]

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 $x_1 \le x_2 \le \ldots x_M < x_{M+1} \le x_{M+2} \cdots \le x_N$, $y_n = -1$ 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] $w = 1,\ b = -\frac{x_{M+1}+x_M}{2}$

[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

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

hyperplane : $w^T x + b = 0$

1D : $wx + b = 0$

由於要求 large margin 所以切割點在中點

中點 = $\frac{x_{M+1}+x_M}{2}$

將中點代入 $wx+b$ 時會等於 0

① $w=-1$

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

代 x_M 入 $wx+b$ 應為 <0

即 $-x_M + \frac{x_{M+1}+x_M}{2}$ = 大 − 小 > 0

即不符合

② $w=1$

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

代 x_M 入 $wx+b$ 應為 <0

即 $x_M - \frac{x_{M+1}+x_M}{2}$ = 小 − 大 < 0

即符合

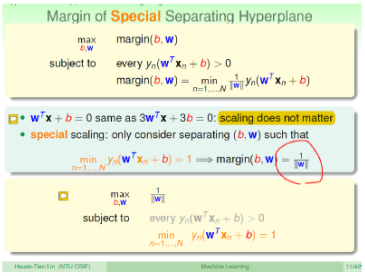
故選 [a] , $w=1$, $b = -\frac{x_{M+1}+x_M}{2}$

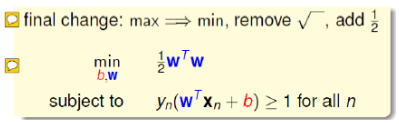
2. ANS [c]

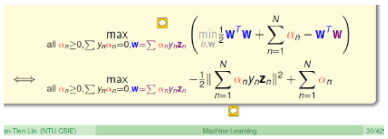
Following the notations in the lecture for the hard-margin SVM in the $\mathcal{Z}$-space. At the optimal $(b, \mathbf{w})$ and $\boldsymbol{\alpha}$, how many of the following values are equal to the length of margin (the distance between the closest example and the decision boundary)? Choose the correct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

(1) $\|\mathbf{w}\|^{-1/2}$
(2) $2\|\mathbf{w}\|^{-1}$
(3) $\|\mathbf{w}\|^{-1}$
(4) $(\sum_{n=1}^{N} \alpha_n)^{-1/2}$
(5) $\sum_{n=1}^{N} [\alpha_n = 1]$
(6) $(2\sum_{n=1}^{N} \alpha_n - \|\sum_{n=1}^{N} \alpha_n y_n \mathbf{z}_n\|^2)^{-1/2}$

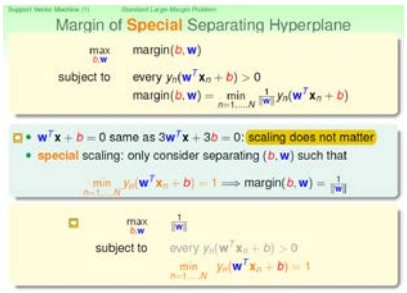
[a] 0
[b] 1
[c] 2
[d] 3
[e] 4

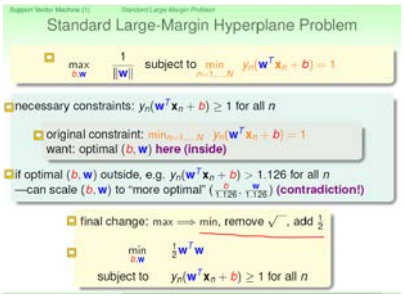




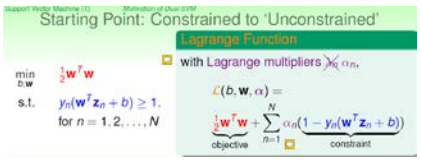


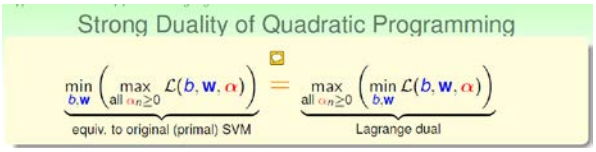
根據上課講義的推倒可以知道在最佳的(b,W)跟α下，margin length = 1/||W||

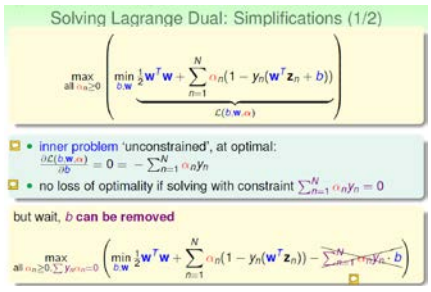


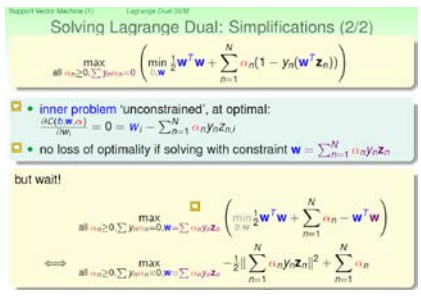


且繼續推倒過程中會將根號拿掉並承上 1/2，變成 1/2$\mathbf{w}^T\mathbf{w}$。









經過 Lagrange Dual 的推倒會發現 1/2$\mathbf{w}^T\mathbf{w}$相當於

$$\max_{\text{all } \alpha_n \geq 0, \sum y_n\alpha_n = 0, \mathbf{w} = \sum \alpha_n y_n \mathbf{z}_n} -\frac{1}{2}\|\sum_{n=1}^{N} \alpha_n y_n \mathbf{z}_n\|^2 + \sum_{n=1}^{N} \alpha_n$$

故 margin length = 1/||W||

$$= 2\sqrt{-\frac{1}{2}||\sum_{n=1}^{N}\alpha_n y_n \boldsymbol{z}_n||^2 + \sum_{n=1}^{N}\alpha_n}$$

$$= 2(-\frac{1}{2}||\sum_{n=1}^{N}\alpha_n y_n \boldsymbol{z}_n||^2 + \sum_{n=1}^{N}\alpha_n)^{-\frac{1}{2}}$$

$$= (||\sum_{n=1}^{N}\alpha_n y_n \boldsymbol{z}_n||^2 + 2\sum_{n=1}^{N}\alpha_n)^{-\frac{1}{2}}$$

選 c。

3. ANS [c]

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

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

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.)

[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

去看 b、w，代入每個點看有沒有符合 2 條件

① $(w^Tx_n + b) \geq 1$ for $y_n > 0$

② $-(w^Tx_n + b) \geq \rho_- = 4$ for $y_n < 0$

(A) $w = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ $b = -1$

$x_1 = (0,1)$, $y_1 = -1$ 代入②

$-([1,0]\begin{bmatrix} 0 \\ 1 \end{bmatrix} - 1) = 1$ ，不符合

(B) $w = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$ $b = -1$

$x_1 = (0,1)$, $y_1 = -1$ 代入②

$-([2,0]\begin{bmatrix} 0 \\ 1 \end{bmatrix} - 1) = 1$ ，不符合

(c) $w=\begin{bmatrix}5\\0\end{bmatrix}$ $b=-4$

$x_1=(0,1)$, $y_1=-1$ 代入②

$-(\begin{bmatrix}5,0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}-4)=4$, 符合

$x_2=(0,0)$, $y_2=-1$ 代入②

$-(\begin{bmatrix}5,0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}-4)=4$, 符合

$x_3=(0,-1)$, $y_3=-1$ 代入②

$-(\begin{bmatrix}5,0\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}-4)=4$, 符合

$x_4=(1,0)$, $y_4=1$ 代入①

$\begin{bmatrix}5,0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}-4=5-4=1\geq 1$, 符合

選 C ✻

(D) $w=\begin{bmatrix}\frac{1}{5}\\0\end{bmatrix}$ $b=-4$

$x_1=(0,1)$, $y_1=-1$ 代入②

$-(\begin{bmatrix}\frac{1}{5},0\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}-4)=4$, 符合

$x_2=(0,0)$, $y_2=-1$ 代入②

$-(\begin{bmatrix}\frac{1}{5},0\end{bmatrix}\begin{bmatrix}0\\0\end{bmatrix}-4)=4$, 符合

$x_3=(0,-1)$, $y_3=-1$ 代入②

$-(\begin{bmatrix}\frac{1}{5},0\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}-4)=4$, 符合

$x_4=(1,0)$, $y_4=1$ 代入①

$\begin{bmatrix}\frac{1}{5},0\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix}-4=\frac{1}{5}-4=-3\frac{4}{5}$, 不符合

4. ANS [c]

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

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

What is □? 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

$$\min_{w,b}\ \frac{1}{2}w^Tw$$

$$\text{s.t.}\quad (w^Tx_n + b) \geq 1 \text{ for } y_n > 0$$

$$-(w^Tx_n + b) \geq \rho_- \text{ for } y_n < 0$$

$$\mathcal{L}(b,w,\alpha) = \frac{1}{2}w^Tw + \sum_{n=1,\,y_n>0}^{\sim}\alpha_n(1 - y_n(w^Tx_n+b)) + \sum_{n=1,\,y_n<0}^{\sim}\alpha_n(\rho_- - y_n(w^Tx_n+b))$$

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

① 對 b 微分

$$\frac{\partial\mathcal{L}(b,w,\alpha)}{\partial b} = 0 = -\sum_{n=1,\,y_n>0}^{\sim}\alpha_n y_n - \sum_{n=1,\,y_n<0}^{\sim}\alpha_n y_n$$

得 $\sum_{n=1,\,y_n>0}^{\sim}\alpha_n y_n + \sum_{n=1,\,y_n<0}^{\sim}\alpha_n y_n = 0$

$$\max_{\text{all}\alpha_n\geq 0,\ \sum_{n=1,\,y_n>0}^{\sim}\alpha_n y_n + \sum_{n=1,\,y_n<0}^{\sim}\alpha_n y_n = 0}\left(\min_{b,w}\left(\frac{1}{2}w^Tw + \sum_{n=1,\,y_n>0}^{\sim}\alpha_n(1 - y_n(w^Tx_n)) + \sum_{n=1,\,y_n<0}^{\sim}\alpha_n(\rho_- - y_n(w^Tx_n))\right)\right)$$

② 對 w 微分

$$\frac{\partial\mathcal{L}(b,w,\alpha)}{\partial w} = 0 = w_i - \sum_{n=1,\,y_n>0}^{\sim}\alpha_n y_n x_{n,i} - \sum_{n=1,\,y_n<0}^{\sim}\alpha_n y_n x_{n,i}$$

得 $w = \sum_{n=1,\,y_n>0}^{\sim}\alpha_n y_n x_n + \sum_{n=1,\,y_n<0}^{\sim}\alpha_n y_n x_n$

$$\max \left(\frac{1}{2} w^T w + \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n - \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n y_n (w^T x_n) + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n \rho_- - \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n y_n (w^T x_n) \right)$$

① $\alpha_n \geq 0$

② $\sum_{n=1, y_n>0}^{\sim} \alpha_n y_n + \sum_{n=1, y_n<0}^{\sim} \alpha_n y_n = 0$

③ $w = \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n y_n x_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n y_n x_n$

$$\Rightarrow \max_{①②③} \left(\frac{1}{2} w^T w + \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n \rho_- - w^T \underbrace{\left(\sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n y_n x_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n y_n x_n \right)}_{w} \right)$$

$$\Rightarrow \max_{①②③} \left(\frac{1}{2} w^T w + \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n \rho_- - w^T w \right)$$

$$\Rightarrow \max_{①②③} \left(-\frac{1}{2} w^T w + \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n \rho_- \right)$$

$$\Rightarrow \max_{①②③} -\frac{1}{2} \left\| \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n y_n x_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n y_n x_n \right\|^2 + \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n \rho_-$$

$$\min_{①②③} \frac{1}{2} \left\| \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n y_n x_n + \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n y_n x_n \right\|^2 - \sum_{\substack{n=1 \\ y_n>0}}^{\sim} \alpha_n - \sum_{\substack{n=1 \\ y_n<0}}^{\sim} \alpha_n \rho_- \quad \#$$

選 C

5. ANS [d]

Let α^* 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 ρ_-? Choose the correct answer; explain your answer.

[a] α^*

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

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

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

[e] none of the other choices

令 α^*, w^* 為 hard margin SVM 的 optimal solution

求 uneven margin SVM 的 optimal solution for uneven margin SVM

假設為 α , w

hard margin SVM $\Rightarrow$ $w^{*T}x_n + b^* = y_n$

uneven SVM

$w^Tx_n + b = 1$, if $y_n = +1$

$w^Tx_n + b = \rho_-$, if $y_n = -1$

$\Rightarrow w^Tx_n + b = y_n' \times (+1) + (1-y_n') \times (-1)\rho_-$

假設 $y_n' = \frac{1+y_n}{2}$

$\Rightarrow w^Tx_n + b = y_n' - (1-y_n')\rho_-$

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

$= \frac{1+y_n}{2} - \frac{1-y_n}{2}\rho_-$

將 $y_n = w^{*T}x_n + b^*$ 代入

$\Rightarrow w^Tx_n + b = \frac{1 + w^{*T}x_n + b^*}{2} - \frac{1 - w^{*T}x_n - b^*}{2}\rho_-$

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

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

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

由於 $w = \sum_{n=1}^{N} \alpha_n y_n z_n$, w 與 α 為線性關係

$\Rightarrow \alpha = \left(\frac{1}{2} + \frac{1}{2}\rho_-\right)\alpha^*$

$= \frac{1}{2}(1+\rho_-)\alpha^*$ 選 [d]

Kernels

6. ANS [a]

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 $\mathbf{\Phi}(\mathbf{x})^T \mathbf{\Phi}(\mathbf{x}')$, where $\mathbf{\Phi}(\mathbf{x})$ includes *unique* terms calculated from $\mathbf{x}$. That is, $x_3 x_5$ would be considered the same term as $x_5 x_3$ (*Note: this is different from the* $\mathbf{\Phi}(\mathbf{x})$ *that we considered in class*). What is dimension of $\mathbf{\Phi}(\mathbf{x})$? Choose the correct answer; explain your answer.

[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

$x, x' \text{ in } \mathbb{R}^d$

特殊情況 $d=2$, $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $x' = \begin{bmatrix} x_1' \\ x_2' \end{bmatrix}$

$K(x, x') = (x^T x')^Q$

$Q = 2$

$K(x, x') = (x^T x)^2 = (x_1 x_1' + x_2 x_2')^2$

$= x_1^2 x_1'^2 + 2 x_1 x_1' x_2 x_2'^2 + x_2^2 x_2'^2$

$= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)(x_1'^2, \sqrt{2} x_1' x_2', x_2'^2)$

$= \Phi(x)^T \Phi(x')$ dim = 3

$C_Q^{Q+d-1} = C_2^{2+2-1} = C_2^3 = 3$

$Q = 3$

$K(x, x') = (x^T x)^3 = (x_1 x_1' + x_2 x_2')^3$

$= x_1^3 x_1'^3 + 3 x_1^2 x_1'^2 x_2 x_2' + 3 x_1 x_1' x_2^2 x_2'^2 + x_2^3 x_2'^3$

dim = 4

$= (x_1^3, \sqrt{3} x_1^2 x_2, \sqrt{3} x_1 x_2^2, x_2^3)$
$(x_1'^3, \sqrt{3} x_1'^2 x_2', \sqrt{3} x_1' x_2'^2, x_2'^3)$

$= \Phi(x)^T \Phi(x')$

$C_Q^{Q+d-1} = C_3^4 = 4$

$Q=4$

$K(X,X^2)=(X^TY)^4=(X_1X_1'+X_2X_2')^4$

$=X_1^4X_1'^4+4X_1^3X_1'^3X_2X_2'+6X_1^2X_1'^2X_2^2X_2'^2$

$+4X_1X_1'X_2^3X_2'^3+X_2^4X_2'^4$

$=(X_1^4, 2X_1^3X_2, \sqrt{6}X_1^2X_2^2, 2X_1X_2^3, X_2^4)$

$(X_1'^4, 2X_1'^3X_2', \sqrt{6}X_1'^2X_2'^2, 2X_1'X_2'^3, X_2'^4)$

$=\Phi(X)^T\Phi(X')$

$C_Q^{Q+d-1}=C_4^5=5$ dim $=5$

ANS. [a] ※

7. ANS [d]

For any feature transform Φ from $\mathcal{X}$ to $\mathcal{Z}$, the squared distance between two examples $\mathbf{x}$ and $\mathbf{x}'$ is $\|\Phi(\mathbf{x}) - \Phi(\mathbf{x}')\|^2$ 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.

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

Norm of a Point: We can compute the norm of a point $\phi(\mathbf{x})$ in feature space as follows

$$\|\phi(\mathbf{x})\|^2 = \phi(\mathbf{x})^T\phi(\mathbf{x}) = K(\mathbf{x}, \mathbf{x})$$

which implies that $\|\phi(\mathbf{x})\| = \sqrt{K(\mathbf{x}, \mathbf{x})}$.

Distance between Points: The distance between two points $\phi(\mathbf{x}_i)$ and $\phi(\mathbf{x}_j)$ can be computed as follows

$$\begin{aligned}\|\phi(\mathbf{x}_i) - \phi(\mathbf{x}_j)\|^2 &= \|\phi(\mathbf{x}_i)\|^2 + \|\phi(\mathbf{x}_j)\|^2 - 2\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \\ &= K(\mathbf{x}_i, \mathbf{x}_i) + K(\mathbf{x}_j, \mathbf{x}_j) - 2K(\mathbf{x}_i, \mathbf{x}_j)\end{aligned} \tag{5.13}$$

- $\|\Phi(x)\|^2 = \Phi(x)^T\Phi(x) = K(x,x)$

  $\|\Phi(x)\| = \sqrt{K(x,x)}$

$\|\Phi(x) - \Phi(x')\|^2 = \|\Phi(x)\|^2 + \|\Phi(x')\|^2 - 2\Phi(x)^T\Phi(x')$

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

tightest upper bound 代表 $-2K(x,x')$ 要小

也就是 $K(x,x')$ 要小

$K(x,x') = (1 + x^Tx')^2$ 為正規.

且 x, x' 為單位向量

所以 $K(x,x) + K(x',x') = (1+1)^2 + (1+1)^2 = 8$

假設一最小情況：$x = \begin{bmatrix}0\\1\end{bmatrix}$ $x' = \begin{bmatrix}0\\-1\end{bmatrix}$

$\|\Phi(x) - \Phi(x')\|^2 = \left(1 + \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\right)^2 + \left(1 + \begin{bmatrix}0 & -1\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right)^2$

$- 2\left(1 + \begin{bmatrix}0 & 1\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right)^2$

$= 8 - 2 \times 0 = 8$

$=$ 8 為 tightest upper bound

選 [d] ✱

Kernel Perceptron Learning Algorithm

8. ANS [c]

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 $\{(\phi(\mathbf{x}_n), y_n)\}_{n=1}^N$, the algorithm updates $\mathbf{w}_t$ to $\mathbf{w}_{t+1}$ when the current $\mathbf{w}_t$ makes a mistake on $\left(\phi(\mathbf{x}_{n(t)}), y_{n(t)}\right)$:

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

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 $\{\phi(\mathbf{x}_n)\}_{n=1}^N$. 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

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

for $t = 0, 1, 2, \ldots$, where $\alpha_t[n]$ 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 $\left(\phi(\mathbf{x}_{n(t)}), y_{n(t)}\right)$? Choose the correct answer; explain your answer.

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

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

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

$\Rightarrow w_{t+1} \leftarrow \underbrace{\sum_{n=1}^{N} \alpha_{t,n}\phi(x_n)}_{①} + \underbrace{y_{n(t)}\phi(x_{n(t)})}_{②}$

① 為 α_t 向量與所有 $\phi(x)$ 相乘後加總
每一個 $\phi(x_n)$ 都有一個與之對應的純量 $\alpha_{t,n}$

② 是針對錯的那一筆 $\phi(x_{n(t)})$ 乘上 $y_{n(t)}$ 倍

①+②，對於 $\alpha_{t+1}[n(t)]$ 來說
就是錯的那筆 $\phi(x_{n(t)})$ 與 $\alpha_t[n(t)]$
更新後的結果

①+②可知 $\alpha_{t+1}[n(t)] \leftarrow \alpha_t[n(t)] + y_{n(t)}$
原本的

選 [c] ✱

Soft-Margin SVM

9. ANS [a]

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:

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

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.

[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

$$\min_{w,b,\xi} \frac{1}{2}w^Tw + \sum_{n=1}^{N} u_n \cdot \xi_n$$

$$s.t.\ \ y_n(w^T\Phi(x_n)+b) \geq 1-\xi_n$$

$$\xi_n \geq 0\ ,\ n=1,\cdots,N$$

$$\mathcal{L}(b,w,\xi,\alpha,\beta) = \frac{1}{2}w^Tw + \sum_{n=1}^{N} u_n\cdot\xi_n + \sum_{n=1}^{N}\alpha_n\cdot(1-\xi_n-y_n(w^T\Phi(x_n)+b)) + \sum_{n=1}^{N}\beta_n\cdot(-\xi_n)$$

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

① 對 ξ_n 微分

$$\frac{d\mathcal{L}}{d\xi_n} = 0 = \sum_{n=1}^{N}(u_n-\alpha_n-\beta_n) \quad \text{得 } \beta_n = \sum_{n=1}^{N}(u_n-\alpha_n)$$

with constraint: $0\leq\alpha_n\leq u_n$

$$\hookrightarrow \max_{0\leq\alpha_n\leq u_n,\ \beta_n=\sum_{n=1}^{N}(u_n-\alpha_n)}\left(\min_{b,w,\xi}\left(\frac{1}{2}w^Tw + \sum_{n=1}^{N}\alpha_n(1-y_n(w^T\Phi(x_n))+b) + \sum_{n=1}^{N}(u_n-\alpha_n-\beta_n)\cdot\xi_n\right)\right)$$

$\because \beta_n = \sum_{n=1}^{N}(u_n-\alpha_n)$ 去掉

② 對 b 微分

$$\frac{\partial \mathcal{L}}{\partial b} = 0 = -\sum_{n=1}^{\tilde{N}} \alpha_n y_n$$

$$\Rightarrow \max_{\substack{0 \le \alpha_n \le u_n \\ \beta_n = \sum_{n=1}^{\tilde{N}} (u_n - \alpha_n) \\ \sum_{n=1}^{\tilde{N}} \alpha_n y_n = 0}} \left(\min_{b,w} \left(\frac{1}{2} w^T w + \sum_{n=1}^{\tilde{N}} \alpha_n \cdot (1 - y_n (w^T \bar{\phi}(x_n)) - \sum_{n=1}^{\tilde{N}} \alpha_n y_n \cdot b \right) \right)$$

去掉

③ 對 w_i 微分

$$\frac{\partial \mathcal{L}}{\partial w_i} = 0 = w_i - \sum_{n=1}^{\tilde{N}} \alpha_n y_n \bar{\phi}(x_n, i)$$

$$\Rightarrow \max_{\substack{0 \le \alpha_n \le u_n \\ \beta_n = \sum_{n=1}^{\tilde{N}} (u_n - \alpha_n) \\ \sum_{n=1}^{\tilde{N}} \alpha_n y_n = 0 \\ w = \sum_{n=1}^{\tilde{N}} \alpha_n y_n \bar{\phi}(x_n)}} \left(\frac{1}{2} w^T w + \sum_{n=1}^{\tilde{N}} \alpha_n - w^T w \right)$$

$$\Rightarrow \max_{\substack{0 \le \alpha_n \le u_n \\ \beta_n = \sum_{n=1}^{\tilde{N}} (u_n - \alpha_n) \\ \sum_{n=1}^{\tilde{N}} \alpha_n y_n = 0 \\ w = \sum_{n=1}^{\tilde{N}} \alpha_n y_n \bar{\phi}(x_n)}} \left(-\frac{1}{2} \left\| \sum_{n=1}^{\tilde{N}} \alpha_n y_n \bar{\phi}(x(n)) \right\|^2 + \sum_{n=1}^{\tilde{N}} \alpha_n \right)$$

$$\Rightarrow \min_{\alpha} \left(\frac{1}{2} \sum_{n=1}^{\tilde{N}} \sum_{m=1}^{\tilde{N}} \alpha_n \alpha_m y_n y_m \bar{\phi}(x_n)^T \bar{\phi}(x_m) - \sum_{n=1}^{\tilde{N}} \alpha_n \right)$$

變 [a] ※

10. ANS [e]

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

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

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

[a] $\frac{1}{15}$
[b] $\frac{1}{24}$
[c] $\frac{1}{30}$
[d] ∞
[e] none of the other choices

$$\min_{b,w} \frac{1}{2}w^Tw + C\sum_{n=1}^{N}\max(1 - \underbrace{y_n\underbrace{(w^Tx_n+b)}_{s_n}}_{\rho_n}, 0)$$

within $(0,1) \Rightarrow 0<\rho<1$

$E_{hinge}(\rho) = \max(1-\rho, 0) \Rightarrow E_{hinge}(\rho) = 1-\rho$

$\rho<1$, $1-\rho>0$ 取 max 剩 $1-\rho$

$E_{smooth}(\rho) = \frac{1}{2}(1-\rho)^2$

uniformly-averaged squared difference

$\Rightarrow$ MSE : $\mathbb{E}[(X-\hat{X})^2]$

$$\mathbb{E}\left[\left(E_{hinge}(\rho) - E_{smooth}(\rho)\right)^2\right]$$

$$= \mathbb{E}\left[\left(1-\rho-\frac{1}{2}(1-\rho)^2\right)^2\right] = \mathbb{E}\left[\left(1-\rho-\frac{1}{2}(1-2\rho+\rho^2)\right)^2\right]$$

$$= \mathbb{E}\left[\left(-\frac{1}{2}\rho^2+\frac{1}{2}\right)^2\right] = \mathbb{E}\left[\frac{1}{4}\rho^4-\frac{1}{2}\rho^2+\frac{1}{4}\right]$$

對 ρ 積分

$$\Rightarrow \int_0^1 \left(\frac{1}{4}\rho^4-\frac{1}{2}\rho^2+\frac{1}{4}\right)d\rho = \left(\frac{1}{20}\rho^5-\frac{1}{6}\rho^3+\frac{1}{4}\rho\right)\Big|_0^1$$

$$= \frac{1}{20}-\frac{1}{6}+\frac{1}{4} = \frac{3-10+15}{60}$$

$$= \frac{8}{60} = \frac{2}{15}$$

沒有此答案

選 [e] ※

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:

$$\min_{\alpha} \quad \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}\alpha_n\alpha_m y_n y_m K(\mathbf{x}_n,\mathbf{x}_m) - \sum_{n=1}^{N}\alpha_n$$

$$\text{subjectto} \quad \sum_{n=1}^{N} y_n\alpha_n = 0$$

$$0 \le \alpha_n \le C \; n = 1,\ldots,N.$$

In the following problems, please use the 0/1 error for evaluating E_{in}, E_{val} and E_{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. ANS [a]

(*) 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.

4.62308650196781

[a] 4.5

[b] 5.0

[c] 5.5

[d] 6.0

[e] 6.5

```
There are three ways to call svm_t

>>> model = svm_train(y, x [, 'tra
>>> model = svm_train(prob [, 'tra
>>> model = svm_train(prob, param)
```

colab source：https://colab.research.google.com/drive/1c3Ih-MVUN6Ubw_Xs_de6RCpWmrETKCiE?usp=sharing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install -U libsvm-official
```

```
Requirement already satisfied: libsvm-official in /usr/local/lib/python3.7/dist-packages (3.25.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from libsvm-official) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->libsvm-official) (1.19.5)
```

```
import urllib.request
import numpy as np
import math
from sklearn.preprocessing import PolynomialFeatures
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem
```

```
def getData(url, filename, class_n):
    # download data to .txt by url
    contents = urllib.request.urlopen(url).read()
    text = str(contents,'utf-8')
    # save file
    fileurl = '/content/drive/MyDrive/Colab Notebooks/ML/HW5/' + filename
    with open(fileurl, 'w') as writefile:
        for line in text:
            writefile.write(line)
    # Read data in LIBSVM format
    Y, X = svm_read_problem(fileurl)
    for i in range(len(Y)):
        if Y[i] == class_n:
            Y[i] = 1
        else:
            Y[i] = -1
    return Y, X
```

```
if __name__ == '__main__':
    #Get Data
    # class_n control class_n vs not class_n
    class_n = 5
    train_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale'
    text_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale.t'
    train_filename = 'satimage_scale.txt'
    text_filename = 'satimage_scale_t.txt'

    train_Y, train_X = getData(train_url, train_filename, class_n)
    text_Y, text_X = getData(text_url, text_filename, class_n)
    # print(len(train_Y))
    # print(type(train_Y))

    #Train data by LIBSVM
    # isKernel=True must be set for precomputed kernel
    prob = svm_problem(train_Y, train_X, isKernel = True)
    # -s 0 C-SVC, -t 0 linear: u'*v, -c C-SVC's C
    param = svm_parameter('-s 0 -t 0 -c 10')
    m = svm_train(prob, param)

    #Get ||W||
    sv_coefs = m.get_sv_coef()
    svs = m.get_SV()

    W = np.zeros(36)
    for i in range(len(svs)):
        for j in range(36):
            if j in svs[i].keys():
                W[j] += svs[i][j] * sv_coefs[i][0]
    WTW = np.dot(W, W)
    W_len = np.sqrt(WTW)
    print(W_len)
```

```
4.62308650196781
```

12. ANS [c]

(*) 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_{in}? Choose the correct answer; provide your command/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"

```
Accuracy = 100% (4435/4435) (classification)
Accuracy = 99.7294% (4423/4435) (classification)
Accuracy = 99.1657% (4398/4435) (classification)
Accuracy = 100% (4435/4435) (classification)
Accuracy = 99.7069% (4422/4435) (classification)
4 vs not 4
```

colab source：https://colab.research.google.com/drive/1-zT3Srr6xjU1P8S0xVomBvXtimLfu778?usp=sharing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install -U libsvm-official
```

```
Requirement already satisfied: libsvm-official in /usr/local/lib/python3.7/dist-packages (3.25.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from libsvm-official) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->libsvm-official) (1.19.5)
```

```
import urllib.request
import numpy as np
import math
from sklearn.preprocessing import PolynomialFeatures
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem
```

```
def getData(url, filename, class_n):
    # download data to .txt by url
    contents = urllib.request.urlopen(url).read()
    text = str(contents,'utf-8')
    # save file
    fileurl = '/content/drive/MyDrive/Colab Notebooks/ML/HW5/' + filename
    with open(fileurl, 'w') as writefile:
        for line in text:
            writefile.write(line)
    # Read data in LIBSVM format
    Y, X = svm_read_problem(fileurl)
    for i in range(len(Y)):
        if Y[i] == class_n:
            Y[i] = 1
        else:
            Y[i] = -1
    return Y, X
```

```python
if __name__ == '__main__':
    #Get Data
    # class_n control class_n vs not class_n
    class_n_list = [2, 3, 4, 5, 6]
    train_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale'
    text_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale.t'
    train_filename = 'satimage_scale.txt'
    text_filename = 'satimage_scale_t.txt'
    Eins = []

    for class_n in class_n_list:
        train_Y, train_X = getData(train_url, train_filename, class_n)
        text_Y, text_X = getData(text_url, text_filename, class_n)

        #Train data by LIBSVM
        # isKernel=True must be set for precomputed kernel
        prob = svm_problem(train_Y, train_X, isKernel = True)
        # -s 0 C-SVC, -t 1 polynomial: (gamma*u'*v + coef0)^degree
        # -c C-SVC's C, -d degree in kernel function (Q=3)
        # -r coef0 : set coef0 in kernel function (default 0)
        # -g gamma : set gamma in kernel function
        param = svm_parameter('-s 0 -t 1 -c 10 -d 3 -r 1 -g 1')
        m = svm_train(prob, param)

        #Get Ein
        p_label, p_acc, p_val = svm_predict(train_Y, train_X, m)
        Eins.append(p_acc[1])

    maxEin = Eins.index(max(Eins)) + 2
    print(str(maxEin) + " vs not " + str(maxEin))
```

```
Accuracy = 100% (4435/4435) (classification)
Accuracy = 99.7294% (4423/4435) (classification)
Accuracy = 99.1657% (4398/4435) (classification)
Accuracy = 100% (4435/4435) (classification)
Accuracy = 99.7069% (4422/4435) (classification)
4 vs not 4
```

13. ANS [e]

(*) 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.

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

```
- Function: int svm_get_nr_sv(const struct svm_model *model)

  This function gives the number of total support vector.
```

```
659
```

colab source：https://colab.research.google.com/drive/1W097I9f0g_5Xr-Sv-IxE0GwdlHwYKIDo?usp=sharing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install -U libsvm-official
```

```
Requirement already satisfied: libsvm-official in /usr/local/lib/python3.7/dist-packages (3.25.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from libsvm-official) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->libsvm-official) (1.19.5)
```

```
import urllib.request
import numpy as np
import math
from sklearn.preprocessing import PolynomialFeatures
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem
```

```
def getData(url, filename, class_n):
    # download data to .txt by url
    contents = urllib.request.urlopen(url).read()
    text = str(contents,'utf-8')
    # save file
    fileurl = '/content/drive/MyDrive/Colab Notebooks/ML/HW5/' + filename
    with open(fileurl, 'w') as writefile:
        for line in text:
            writefile.write(line)
    # Read data in LIBSVM format
    Y, X = svm_read_problem(fileurl)
    for i in range(len(Y)):
        if Y[i] == class_n:
            Y[i] = 1
        else:
            Y[i] = -1
    return Y, X
```

```
if __name__ == '__main__':
    #Get Data
    # class_n control class_n vs not class_n
    class_n_list = [2, 3, 4, 5, 6]
    train_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale'
    text_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale.t'
    train_filename = 'satimage_scale.txt'
    text_filename = 'satimage_scale_t.txt'
    nr_sv = []

    for class_n in class_n_list:
        train_Y, train_X = getData(train_url, train_filename, class_n)
        text_Y, text_X = getData(text_url, text_filename, class_n)

        #Train data by LIBSVM
        # isKernel=True must be set for precomputed kernel
        prob = svm_problem(train_Y, train_X, isKernel = True)
        # -s 0 C-SVC, -t 1 polynomial: (gamma*u'*v + coef0)^degree
        # -c C-SVC's C, -d degree in kernel function (Q=3)
        # -r coef0 : set coef0 in kernel function (default 0)
        # -g gamma : set gamma in kernel function
        param = svm_parameter('-s 0 -t 1 -c 10 -d 3 -r 1 -g 1')
        m = svm_train(prob, param)

        #Get nr_sv
        nr_sv.append(m.get_nr_sv())

    print(max(nr_sv))
```

```
659
```

14. ANS [d]

(*) 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_{out}? If there is a tie, please pick the smallest C. Choose the correct answer; provide your command/code.

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

```
Accuracy = 76.95% (1539/2000) (classification)
Accuracy = 79.25% (1585/2000) (classification)
Accuracy = 89.05% (1781/2000) (classification)
Accuracy = 89.95% (1799/2000) (classification)
Accuracy = 89.95% (1799/2000) (classification)
[0.922, 0.83, 0.438, 0.402, 0.402]
```

colab source：

https://colab.research.google.com/drive/1UYw1IDXTG9_8QOUH7ZZEVUtVsPqVZm7N?usp=sharing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install -U libsvm-official
```

```
Requirement already satisfied: libsvm-official in /usr/local/lib/python3.7/dist-packages (3.25.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from libsvm-official) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->libsvm-official) (1.19.5)
```

```
import urllib.request
import numpy as np
import math
from sklearn.preprocessing import PolynomialFeatures
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem
```

```
def getData(url, filename, class_n):
    # download data to .txt by url
    contents = urllib.request.urlopen(url).read()
    text = str(contents,'utf-8')
    # save file
    fileurl = '/content/drive/MyDrive/Colab Notebooks/ML/HW5/' + filename
    with open(fileurl, 'w') as writefile:
        for line in text:
            writefile.write(line)
    # Read data in LIBSVM format
    Y, X = svm_read_problem(fileurl)
    for i in range(len(Y)):
        if Y[i] == class_n:
            Y[i] = 1
        else:
            Y[i] = -1
    return Y, X
```

```python
if __name__ == '__main__':
    #Get Data
    # class_n control class_n vs not class_n
    class_n = 1
    C = ['0.01', '0.1', '1', '10', '100']
    train_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale'
    text_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale.t'
    train_filename = 'satimage_scale.txt'
    text_filename = 'satimage_scale_t.txt'
    Eouts = []

    for c in C:
        train_Y, train_X = getData(train_url, train_filename, class_n)
        text_Y, text_X = getData(text_url, text_filename, class_n)

        #Train data by LIBSVM
        # isKernel=True must be set for precomputed kernel
        prob = svm_problem(train_Y, train_X, isKernel = True)
        # -s 0 C-SVC, -t 2 radial basis function: exp(-gamma*|u-v|^2)
        # -c C-SVC's C
        # -g gamma : set gamma in kernel function (gamma=10)
        param = svm_parameter('-s 0 -t 2 -c ' + c + ' -g 10')
        m = svm_train(prob, param)

        #Get Eout
        p_label, p_acc, p_val = svm_predict(text_Y, text_X, m)
        Eouts.append(p_acc[1])

    print(Eouts)
```

```
Accuracy = 76.95% (1539/2000) (classification)
Accuracy = 79.25% (1585/2000) (classification)
Accuracy = 89.05% (1781/2000) (classification)
Accuracy = 89.95% (1799/2000) (classification)
Accuracy = 89.95% (1799/2000) (classification)
[0.922, 0.83, 0.438, 0.402, 0.402]
```

最後是對應到 C [0.01, 0.1, 1, 10, 100]

可見 C=10 or 100 時 Eout 最低，選小的 C=10，答案為[d]

15. ANS [b]

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

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

```
Accuracy = 98.75% (1975/2000) (classification)
Accuracy = 98.8% (1976/2000) (classification)
Accuracy = 79.25% (1585/2000) (classification)
Accuracy = 76.95% (1539/2000) (classification)
Accuracy = 76.95% (1539/2000) (classification)
[0.05, 0.048, 0.83, 0.922, 0.922]
```

colab source：

https://colab.research.google.com/drive/1GxpFCkNInLYiYNUXcVsOkhnkTpP6spEn?usp=sharing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install -U libsvm-official
```

```
Requirement already satisfied: libsvm-official in /usr/local/lib/python3.7/dist-packages (3.25.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from libsvm-official) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->libsvm-official) (1.19.5)
```

```
import urllib.request
import numpy as np
import math
from sklearn.preprocessing import PolynomialFeatures
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem
```

```
def getData(url, filename, class_n):
    # download data to .txt by url
    contents = urllib.request.urlopen(url).read()
    text = str(contents,'utf-8')
    # save file
    fileurl = '/content/drive/MyDrive/Colab Notebooks/ML/HW5/' + filename
    with open(fileurl, 'w') as writefile:
        for line in text:
            writefile.write(line)
    # Read data in LIBSVM format
    Y, X = svm_read_problem(fileurl)
    for i in range(len(Y)):
        if Y[i] == class_n:
            Y[i] = 1
        else:
            Y[i] = -1
    return Y, X
```

```python
if __name__ == '__main__':
    #Get Data
    # class_n control class_n vs not class_n
    class_n = 1
    gamma = ['0.1', '1', '10', '100', '1000']
    train_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale'
    text_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale.t'
    train_filename = 'satimage_scale.txt'
    text_filename = 'satimage_scale_t.txt'
    Eouts = []

    for g in gamma:
        train_Y, train_X = getData(train_url, train_filename, class_n)
        text_Y, text_X = getData(text_url, text_filename, class_n)

        #Train data by LIBSVM
        # isKernel=True must be set for precomputed kernel
        prob = svm_problem(train_Y, train_X, isKernel = True)
        # -s 0 C-SVC, -t 2 radial basis function: exp(-gamma*|u-v|^2)
        # -c C-SVC's C
        # -g gamma : set gamma in kernel function (gamma=10)
        param = svm_parameter('-s 0 -t 2 -c 0.1 -g ' + g)
        m = svm_train(prob, param)

        #Get Eout
        p_label, p_acc, p_val = svm_predict(text_Y, text_X, m)
        Eouts.append(p_acc[1])

    print(Eouts)
```

```
Accuracy = 98.75% (1975/2000) (classification)
Accuracy = 98.8% (1976/2000) (classification)
Accuracy = 79.25% (1585/2000) (classification)
Accuracy = 76.95% (1539/2000) (classification)
Accuracy = 76.95% (1539/2000) (classification)
[0.05, 0.048, 0.83, 0.922, 0.922]
```

最後是對應到γ [0.1, 1, 10, 100, 1000]

可見γ=1 時 Eout 最低，答案為[b]

16. ANS [a]

(*) 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_{SVM}^{-}. Fix $C = 0.1$ and use the validation procedure to choose the best γ among $\{0.1, 1, 10, 100, 1000\}$ according to E_{val}. If there is a tie of E_{val}, choose the smallest γ. Repeat the procedure 1000 times. Which of the following values of γ is selected the most number of times? Choose the correct answer; provide your command/code.

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

```
[623,  377,  0,  0,  0]
```

colab source：https://colab.research.google.com/drive/1QSxwiec0EAuAnIoTAnt-0PSWCcbIJZru?usp=sharing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
pip install -U libsvm-official
```

```
Requirement already satisfied: libsvm-official in /usr/local/lib/python3.7/dist-packages (3.25.0)
Requirement already satisfied: scipy in /usr/local/lib/python3.7/dist-packages (from libsvm-official) (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in /usr/local/lib/python3.7/dist-packages (from scipy->libsvm-official) (1.19.5)
```

```
import urllib.request
import numpy as np
import math
from sklearn.preprocessing import PolynomialFeatures
from libsvm.svmutil import *
from libsvm.commonutil import svm_read_problem
```

```
def getData(url, filename, class_n):
    # download data to .txt by url
    contents = urllib.request.urlopen(url).read()
    text = str(contents,'utf-8')
    # save file
    fileurl = '/content/drive/MyDrive/Colab Notebooks/ML/HW5/' + filename
    with open(fileurl, 'w') as writefile:
        for line in text:
            writefile.write(line)
    # Read data in LIBSVM format
    Y, X = svm_read_problem(fileurl)
    for i in range(len(Y)):
        if Y[i] == class_n:
            Y[i] = 1
        else:
            Y[i] = -1
    return Y, X
```

```python
if __name__ == '__main__':
    #Get Data
    # class_n control class_n vs not class_n
    class_n = 1
    bestgammas = [0, 0, 0, 0, 0]
    train_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale'
    text_url = 'https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass/satimage.scale.t'
    train_filename = 'satimage_scale.txt'
    text_filename = 'satimage_scale_t.txt'


    for t in tqdm(range(1000)):
        gamma = ['0.1', '1', '10', '100', '1000']
        Evals = []
        train_Y, train_X = getData(train_url, train_filename, class_n)
        text_Y, text_X = getData(text_url, text_filename, class_n)

        for g in gamma:
            train_index = list(range(0,len(train_X)))
            val_index = random.sample(range(len(train_X)),200)
            train_index = set(train_index).difference(set(val_index))
            train_index = list(train_index)

            val_Y = np.array(train_Y)[val_index]
            val_X = np.array(train_X)[val_index]
            train_Y = np.array(train_Y)[train_index]
            train_X = np.array(train_X)[train_index]


            #Train data by LIBSVM
            # isKernel=True must be set for precomputed kernel
            prob = svm_problem(train_Y, train_X, isKernel = True)
            # -s 0 C-SVC, -t 2 radial basis function: exp(-gamma*|u-v|^2)
            # -c C-SVC's C
            # -g gamma : set gamma in kernel function (gamma=10)
            param = svm_parameter('-s 0 -t 2 -c 0.1 -g ' + g)
            m = svm_train(prob, param)

            #Get Eout
            p_label, p_acc, p_val = svm_predict(val_Y, val_X, m)
            Evals.append(p_acc[1])

        print(Evals)
        bestgammas[Evals.index(min(Evals))] += 1
    print(bestgammas)
```

```
100%|■■■■■■■■■■| 999/1000 [2:40:56<00:09,  9.64s/it]Accuracy = 76% (152/200) (classification)
[0.06, 0.04, 0.92, 1.0, 0.96]
Accuracy = 99.5% (199/200) (classification)
Accuracy = 98% (196/200) (classification)
Accuracy = 76.5% (153/200) (classification)
Accuracy = 76% (152/200) (classification)
100%|■■■■■■■■■■| 1000/1000 [2:41:06<00:00,  9.67s/it]Accuracy = 71% (142/200) (classification)
[0.02, 0.08, 0.94, 0.96, 1.16]
[623, 377, 0, 0, 0]
```

最後產生一個紀錄每次 iteration 最好的 gamma 的 list：bestgamma

是對應到γ [0.1, 1, 10, 100, 1000]

可見γ=0.1 時被選作 best gamma (Eval 最低)的次數最多，答案為[a]