機器學習理論與實作 II

講師：杜岳華

Support vector machine

S . V . M .

一種二類分類模型，
其基本模型定義為特徵空間上的間隔最大的線性分類器，
其學習策略便是間隔最大化，
最終可轉化為一個凸二次規劃問題的求解。

From Preceptron

感知

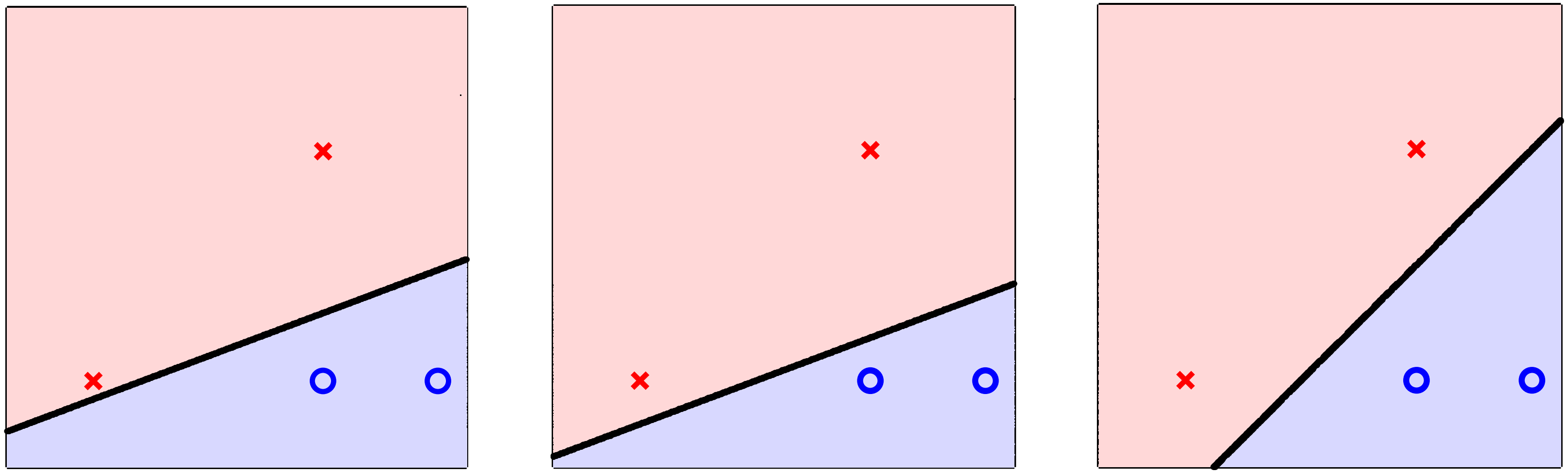
在 Preceptron 模型當中，PLA 只能幫你決定切的 **好或壞**，但是無法告訴你有沒有 **更好** 的分法。

$$f(x) = sign(\sum_i w_i x_i) = sign(\mathbf{w}^T \mathbf{x})$$

一顆神經網路

分割

A better way to separate?

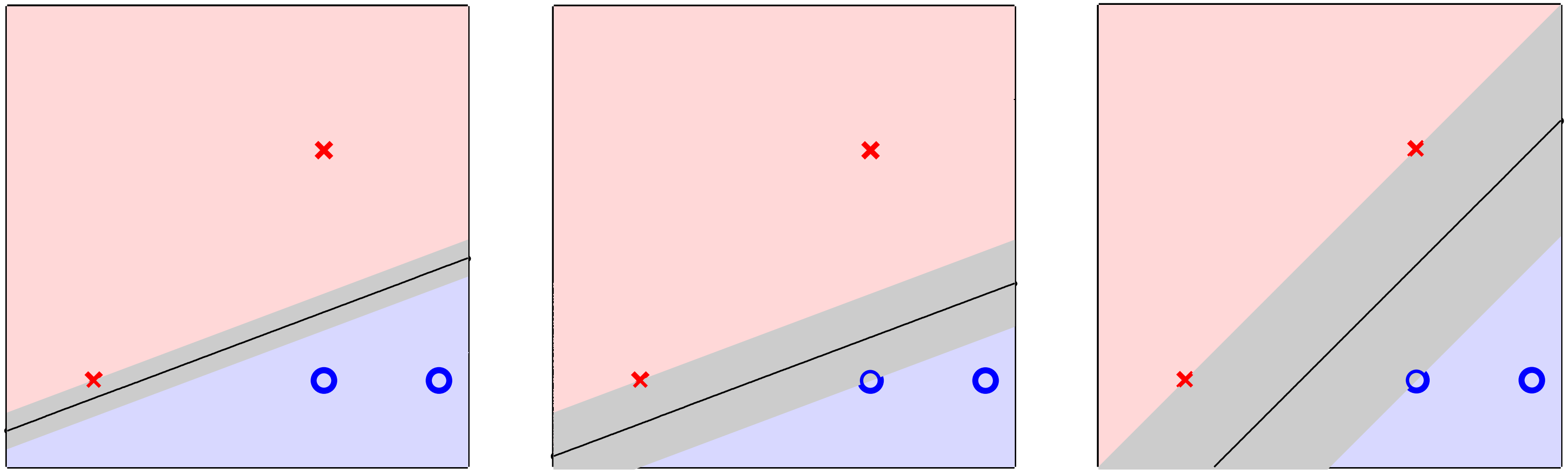


picture from coursera, 《機器學習技法》 - 林軒田

Large-margin hyperplane

線離點愈遠是愈好的分法！

容錯率高



灰色區域：
透過 Support Vector 的協助
來學習出最胖的超平面。

picture from coursera, 《機器學習技法》 - 林軒田

分類器

To large-margin classifier

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\max} \quad & \text{margin}(\mathbf{w}, b) \\ \text{subject to} \quad & \text{classify } (\mathbf{x_n}, y_n) \text{ correctly} \\ & \text{margin}(\mathbf{w}, b) = \min_{i=1...n} \text{distance}(\mathbf{x_n}, \mathbf{w}, b) \end{aligned}$$

將想法化為數學公式

To large-margin classifier

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\max} \quad & \text{margin}(\mathbf{w}, b) \\ \text{subject to} \quad & \text{classify } (\mathbf{x_n}, y_n) \text{ correctly} \\ & \text{margin}(\mathbf{w}, b) = \min_{i=1\ldots n} \text{distance}(\mathbf{x_i}, \mathbf{w}, b) \end{aligned}$$

最短距離？

$$\text{distance}(\mathbf{x_i}, \mathbf{w}, b) = \frac{|ax_i + by_i + c|}{\sqrt{a^2 + b^2}}$$

$$\text{distance}(\mathbf{x_i}, \mathbf{w}, b) = \frac{|\mathbf{w}^T \mathbf{x_i} + b|}{\|\mathbf{w}\|}$$

To large-margin classifier

$$\begin{aligned} \arg\max_{\mathbf{w},b} \quad & \text{margin}(\mathbf{w},b) \\ \text{subject to} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{x_i}+b) > 0 \\ & \text{margin}(\mathbf{w},b) = \min_{i=1\ldots n} \frac{|\mathbf{w}^T\mathbf{x_i}+b|}{\|\mathbf{w}\|} \end{aligned}$$

$$\begin{aligned} \arg\max_{\mathbf{w},b} \quad & \text{margin}(\mathbf{w},b) \\ \text{subject to} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{x_i}+b) > 0 \\ & \text{margin}(\mathbf{w},b) = \min_{i=1\ldots n} \frac{y_i(\mathbf{w}^T\mathbf{x_i}+b)}{\|\mathbf{w}\|} \end{aligned}$$

縮放技巧

Scaling trick

$$\text{margin}(\mathbf{w}, b) = \min_{i=1...n} \frac{y_i(\mathbf{w}^T\mathbf{x_i} + b)}{\|\mathbf{w}\|}$$

$\mathbf{w}^T\mathbf{x_i} + b = 0$ is the same as $c\mathbf{w}^T\mathbf{x_i} + cb = 0$.

$$\text{let } \min_{i=1...n} y_i(\mathbf{w}^T\mathbf{x_i} + b) = 1$$

$$\text{margin}(\mathbf{w}, b) = \frac{1}{\|\mathbf{w}\|}$$

SVM

$$\begin{aligned} \arg\max_{\mathbf{w},b} \quad & \frac{1}{\|\mathbf{w}\|} \\ \text{subject to} \quad & \min_{i=1\ldots n} y_i(\mathbf{w}^T\mathbf{x_i}+b)=1 \end{aligned}$$

relax: $\min_{i=1\ldots n} y_i(\mathbf{w}^T\mathbf{x_i}+b)=1 \Rightarrow y_i(\mathbf{w}^T\mathbf{x_i}+b)\geq 1$

$max \Rightarrow min, remove \sqrt{}, add \frac{1}{2}$

SVM雛形

$$\begin{aligned} \arg\min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{x_i}+b)\geq 1 \end{aligned}$$

How to solve the problem?

二次規劃

Quadratic programming

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\min} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 \end{aligned}$$

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\min} \quad & \frac{1}{2}\mathbf{u}^T Q\mathbf{u} + \mathbf{p}^T\mathbf{u} \\ \text{subject to} \quad & \forall i, \mathbf{a}_i^T\mathbf{u} \geq c_i \end{aligned}$$

Quadratic programming

$$\mathbf{u} = \begin{bmatrix} b \\ \mathbf{w} \end{bmatrix}, Q = \begin{bmatrix} 0 & \mathbf{0}_d^T \\ \mathbf{0}_d & I_d \end{bmatrix}, \mathbf{p} = \mathbf{0}_{d+1}, \mathbf{a}_i^T = y_i \begin{bmatrix} 1 & \mathbf{x}_i^T \end{bmatrix}, c_i = 1$$

$$\begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} = QP(Q, \mathbf{p}, A, \mathbf{c})$$

$$\begin{bmatrix} b \\ \mathbf{w} \end{bmatrix} \in \mathbb{R}^{d+1}$$

放入參數即可

非線性變換

Nonlinear transformation

非線性的轉換其實可以依我們的需求轉換到非常高維，甚至可能到無限多維

Nonlinear transformation

Want non-linear transform?

$$\mathbf{z_i} = \phi(\mathbf{x_i})$$

φ非線性變換

Nonlinear transformation

$$\begin{aligned} \arg\min_{\mathbf{w},b} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq 1 \end{aligned}$$

限制式

Solving $d+1$ variables and n constraints! 很大的矩陣

What if large d? or infinite?

Want SVM without depending on $d+1$!

雙

Dual support vector machine

Dual SVM
為了可以做到無限多維特徵轉換，
我們需要將 SVM 轉為另外一個問題，
在數學上已證明這兩個問題其實是一樣的，
所以又稱為是 SVM 的對偶問題。

< Brook 補充 >

對偶理論

每一個線性規劃問題都會有一個「對偶問題」與其對應，
而原來的問題則稱之為「原始問題」。

有關於原始 - 對偶間的關係，一個最基本的性質，那
就是原始與對偶問題兩者**有相同的最佳目標函數值**，
此特性稱之為「對偶理論」

弱對偶定理

**原始、對偶均是可行解條件下，最小化問題的目標值恆大於
等於最大化問題之目標值**

強對偶定理

**原始問題與對偶問題中任一個存在最佳解，則另一個也必存
在最佳解，且兩目標值相等。**

<Brook 補充>

對偶問題

第 I 類型	第 II 類型
目標函數 max　Z	目標函數 min　W
第 i 個功能限制式 $\leq$ $=$ $\geq$	第 i 個變數 ≥ 0 $\in R$ ≤ 0
第 j 個變數 ≥ 0 $\in R$ ≤ 0	第 j 個功能限制式 $\geq$ $=$ $\leq$

原始問題_變數＝對偶問題_限制式
原始問題_限制式＝對偶問題_變數

$$\max \quad Z = c_1x_1 + c_2x_2 + \cdots\cdots + c_nx_n$$

$$\text{s.t.} \begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots\cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots\cdots + a_{2n}x_n \leq b_2 \quad (\text{模式 } 3.1) \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots\cdots + a_{mn}x_n \leq b_m \end{cases}$$

$$x_1, x_2, \cdots, x_n \geq 0$$

(注意： $b_i \in R$，即不需非負限制，現　在不是要化成正規型　)

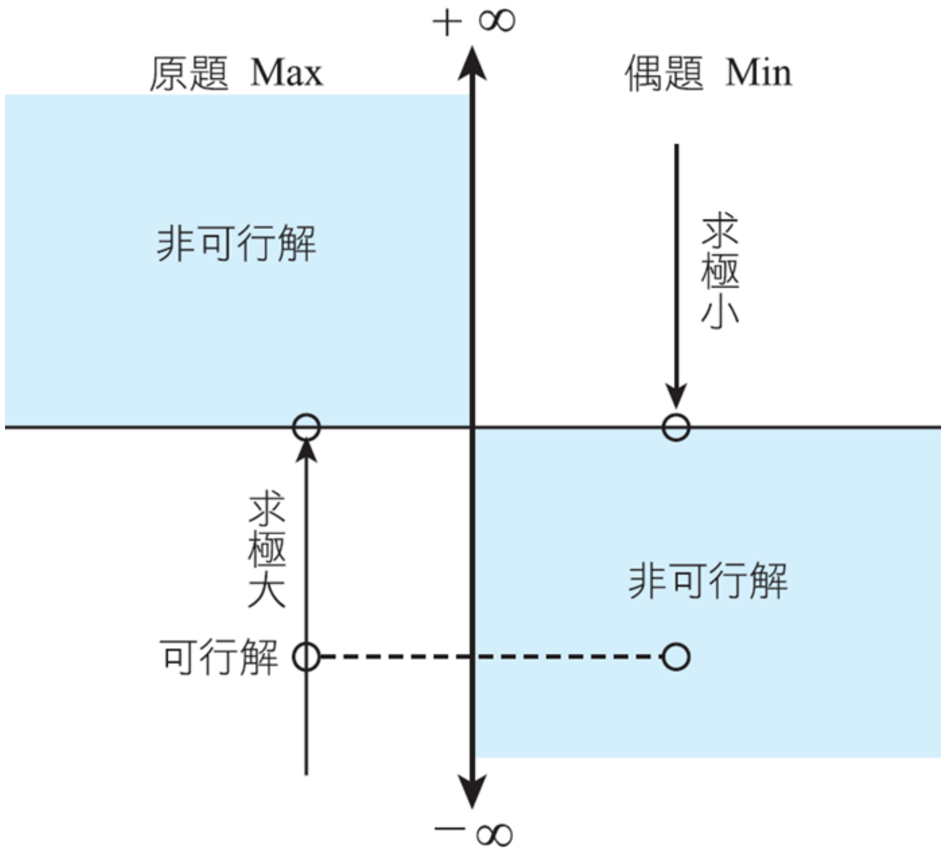
$$\Updownarrow$$

$$\min \quad W = b_1y_1 + b_2y_2 + \cdots\cdots + b_my_m$$

$$\text{s.t.} \begin{cases} a_{11}y_1 + a_{21}y_2 + \cdots\cdots + a_{m1}y_m \geq c_1 \\ a_{12}y_1 + a_{22}y_2 + \cdots\cdots + a_{m2}y_m \geq c_2 \quad (\text{模式}3.2) \\ \vdots \\ a_{1n}y_1 + a_{2n}y_2 + \cdots\cdots + a_{mn}y_m \geq c_n \end{cases}$$

$$y_1 \geq 0, y_2 \geq 0, \cdots, y_m \geq 0$$

< Brook 補充 >



舉例：

原始問題：評估每一產品的獲利情況，而使總利潤極大化

對偶問題：評估生產產品所付出的成本，而使總成本極小化

這樣做的優點在於：

一者對偶問題往往更容易求解；

二者可以自然的引入核函式，進而推廣到非線性分類問題。

Dual problem

用d + 1變量和n個約束解決QP問題

primal SVM problem: solve QP problem with $d+1$ variables and n constraints.

轉換對偶問題 用n個變量和n + 1個約束解決QP問題

dual SVM problem: solve QP problem with n variables and $n+1$ constraints

QP : Quadratic Programming 二次規劃

拉格朗日乘數

是一種尋找多元函數在其變數受到一個或多個條件的約束時的極值的方法。

Lagrange multiplier

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\min} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq 1 \end{aligned}$$

Lagrange function:

$$\mathcal{L}(b, \mathbf{w}, \lambda) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \lambda_i (1 - y_i(\mathbf{w}^T\mathbf{z_i} + b))$$

$$\underset{\mathbf{w},b}{\arg\min} \left(\underset{\forall \lambda_i \geq 0}{\arg\max}\, \mathcal{L}(b, \mathbf{w}, \lambda) \right)$$

原始

Primal SVM problem

$$\arg\min_{\mathbf{w},b}\Big(\arg\max_{\forall\lambda_i\geq 0}\mathcal{L}(b,\mathbf{w},\lambda)\Big)$$

找極值

拉格朗日的雙重問題

Lagrange dual problem

$$\underset{\mathbf{w},b}{\arg\min}\left(\underset{\forall\lambda_i\geq 0}{\arg\max}\,\mathcal{L}(b,\mathbf{w},\lambda)\right)\geq\underset{\mathbf{w},b}{\arg\min}\,\mathcal{L}(b,\mathbf{w},\lambda')$$

$$max \geq any$$

$$\underset{\mathbf{w},b}{\arg\min}\left(\underset{\forall\lambda_i\geq 0}{\arg\max}\,\mathcal{L}(b,\mathbf{w},\lambda)\right)\geq\underset{\forall\lambda'_i\geq 0}{\arg\max}\left(\underset{\mathbf{w},b}{\arg\min}\,\mathcal{L}(b,\mathbf{w},\lambda')\right)$$

the max of any

Lagrange dual problem

$$\arg\max_{\forall \lambda'_i \geq 0} \left(\arg\min_{\mathbf{w}, b} \mathcal{L}(b, \mathbf{w}, \lambda') \right)$$

$\geq$: weak duality 弱對偶性

$=$: strong duality 強對偶性

$$\underbrace{\min_{b,\mathbf{w}} \left(\max_{\text{all } \alpha_n \geq 0} \mathcal{L}(b, \mathbf{w}, \alpha) \right)}_{\text{equiv. to original (primal) SVM}} \geq \underbrace{\max_{\text{all } \alpha_n \geq 0} \left(\min_{b,\mathbf{w}} \mathcal{L}(b, \mathbf{w}, \alpha) \right)}_{\text{Lagrange dual}}$$

簡單化

Simplification

$$\underset{\forall \lambda_i' \geq 0}{\arg\max} \left(\underset{\mathbf{w}, b}{\arg\min} \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^{n} \lambda_i' (1 - y_i (\mathbf{w}^T \mathbf{z_i} + b)) \right)$$

$$\frac{\partial \mathcal{L}(b, \mathbf{w}, \lambda')}{\partial b} = 0$$

$$\frac{\partial \mathcal{L}(b, \mathbf{w}, \lambda')}{\partial b} = -\sum_{i=1}^{n} \lambda_i' y_i = 0$$

$$\underset{\forall \lambda_i' \geq 0, \sum \lambda_i' y_i = 0}{\arg\max} \left(\underset{\mathbf{w}}{\arg\min} \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^{n} \lambda_i' (1 - y_i (\mathbf{w}^T \mathbf{z_i})) - b \cdot \sum_{i=1}^{n} \lambda_i' y_i \right)$$

Simplification

$$\underset{\forall \lambda_i' \geq 0, \sum \lambda_i' y_i = 0}{\arg\max} \left(\underset{\mathbf{w}}{\arg\min} \frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^{n} \lambda_i' (1 - y_i (\mathbf{w}^T \mathbf{z_i})) \right)$$

$$\frac{\partial \mathcal{L}(b, \mathbf{w}, \lambda')}{\partial \mathbf{w}} = 0$$

$$\frac{\partial \mathcal{L}(b, \mathbf{w}, \lambda')}{\partial w_j} = w_j - \sum_{i=1}^{n} \lambda_i' y_i z_{ij} = 0$$

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i' y_i \mathbf{z_i}$$

$$\underset{\forall \lambda_i' \geq 0, \sum \lambda_i' y_i = 0, \mathbf{w} = \sum \lambda_i' y_i \mathbf{z_i}}{\arg\max} \left(\frac{1}{2} \mathbf{w}^T \mathbf{w} + \sum_{i=1}^{n} \lambda_i' - \mathbf{w}^T \mathbf{w} \right)$$

Dual SVM problem

$$\underset{\forall \lambda_i' \geq 0, \sum \lambda_i' y_i = 0, \mathbf{w} = \sum \lambda_i' y_i \mathbf{z_i}}{\arg\max} \; -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{n} \lambda_i'$$

$$\underset{\forall \lambda_i' \geq 0, \sum \lambda_i' y_i = 0}{\arg\max} \; -\frac{1}{2}\left\| \sum_{i=1}^{n} \lambda_i' y_i \mathbf{z_i} \right\|^2 + \sum_{i=1}^{n} \lambda_i'$$

$$\begin{aligned} \underset{\lambda'}{\arg\min} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \lambda_i' \lambda_j' y_i y_j \mathbf{z_i}^T \mathbf{z_j} - \sum_{i=1}^{n} \lambda_i' \\ \text{subject to} \quad & \forall \lambda_i' \geq 0 \\ & \sum \lambda_i' y_i = 0 \end{aligned}$$

雙重問題的二次規劃

Quadratic programming for dual problem

$$\begin{aligned} \underset{\boldsymbol{\lambda}'}{\arg\min} \quad & \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\lambda'_i\lambda'_j y_i y_j \mathbf{z_i}^T\mathbf{z_j} - \sum_{i=1}^{n}\lambda'_i \\ \text{subject to} \quad & \forall\lambda'_i \geq 0 \\ & \sum \lambda'_i y_i = 0 \end{aligned}$$

$$\begin{aligned} \underset{\boldsymbol{\lambda}'}{\arg\min} \quad & \frac{1}{2}\boldsymbol{\lambda}'^T Q\boldsymbol{\lambda}' + \mathbf{p}^T\boldsymbol{\lambda}' \\ \text{subject to} \quad & \forall i, \mathbf{a}_i^T\boldsymbol{\lambda}' \geq c_i \end{aligned}$$

$$Q = [q_{ij}] = [y_i y_j \mathbf{z_i}^T\mathbf{z_j}],$$

$$\mathbf{p} = -\mathbf{1}_n, \mathbf{a}_i^T = \ldots, c_i = 0$$

Quadratic programming for dual problem

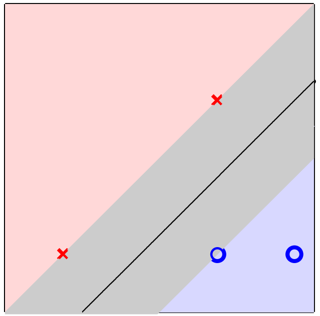
$$Q = [q_{ij}] = [y_i y_j \mathbf{z_i}^T \mathbf{z_j}],$$
$$\mathbf{p} = -\mathbf{1}_n, \mathbf{a}_i^T = \ldots, c_i = 0$$
$$\boldsymbol{\lambda}' = QP(Q, \mathbf{p}, A, \mathbf{c})$$

我們需要所有的點嗎？

Do we need all data points?

我們只需要支持向量！(取最接近線的點)

We only need support vectors!



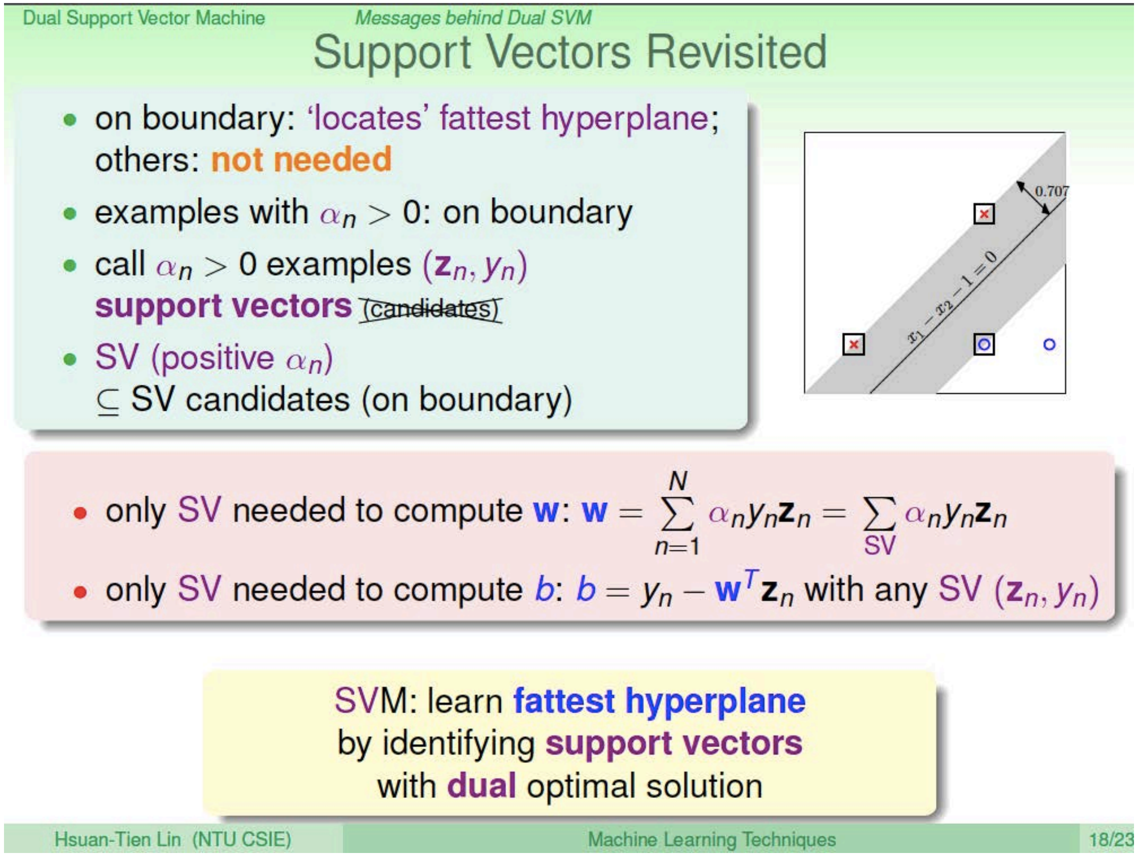
picture from coursera, 《機器學習技法》 - 林軒田

支持向量

Support vector

$$\mathbf{w} = \sum_{i=1}^{n} \lambda_i' y_i \mathbf{z_i} = \sum_{SV} \lambda_i' y_i \mathbf{z_i}$$

$$b = y_i - \mathbf{w}^T \mathbf{z_i} \text{ with SV}(\mathbf{z_i}, y_i)$$



由於 w = sigma(anynzn)，所以其實也只有 an 大於 0 的點會影響到 w 的計算，b 也是只有 an 大於 0 時才有辦法計算，所以 an 大於 0 的資料點其實就是 Support Vector。

核心

Kernel SVM

Kernel Function

假設考慮一個非線性轉換，將X空間轉換到Z空間，那如果我需要計算轉換過的兩個新Features相乘Zn(Xn)×Zm(Xm)，我有辦法不需要先做特徵轉換再相乘，而是直接使用原有的Features Xn和Xm求出Zn(Xn)×Zm(Xm)的最後結果？這種情形數學可以表示成K(Xn,Xm)=Zn(Xn)×Zm(Xm)，這個函式就叫Kernel Function。

Kernel Function可以簡化和優化因為「特徵轉換」所帶來的複雜計算。

計算效率

Computational efficiency

$$Q = [q_{ij}] = [y_i y_j \mathbf{z_i}^T \mathbf{z_j}]$$
$$\mathbf{z_i}^T \mathbf{z_j} = \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$$

require faster than $\mathcal{O}(d)$

內在

Inner product for ϕ_2

2nd order polynomial transform:

$$\begin{gathered}\phi_2(\mathbf{x}) = (1, x_1, x_2, \ldots, x_d, \\ x_1^2, x_1 x_2, \ldots, x_1 x_d, \\ x_2 x_1, x_2^2, \ldots, x_2 x_d, \\ \ldots, x_d^2)\end{gathered}$$

$$\phi_2(\mathbf{x})^T \phi_2(\mathbf{x}') = 1 + \sum_{i=1}^{d} x_i x_i' + \sum_{i=1}^{d} \sum_{j=1}^{d} x_i x_j x_i' x_j'$$

Inner product for ϕ_2

$$\phi_2(\mathbf{x})^T \phi_2(\mathbf{x}') = 1 + \sum_{i=1}^{d} x_i x_i' + \sum_{i=1}^{d} x_i x_i' \sum_{j=1}^{d} x_j x_j'$$

$$= 1 + \mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}')(\mathbf{x}^T \mathbf{x}')$$

$\mathcal{O}(d)$

核技巧

Kernel trick

kernel function: $K_\phi(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$

$$K_{\phi_2}(\mathbf{x}, \mathbf{x}') = 1 + \mathbf{x}^T \mathbf{x}' + (\mathbf{x}^T \mathbf{x}')^2$$

$$q_{ij} = y_i y_j \mathbf{z_i}^T \mathbf{z_j} = y_i y_j K_{\phi_2}(\mathbf{x}, \mathbf{x}')$$

一般

General poly-2 kernel

$$K_{\phi_2}(\mathbf{x}, \mathbf{x}') = 1 + \mathbf{x}^T\mathbf{x}' + (\mathbf{x}^T\mathbf{x}')^2$$

$$K_2(\mathbf{x}, \mathbf{x}') = 1 + 2\mathbf{x}^T\mathbf{x}' + (\mathbf{x}^T\mathbf{x}')^2 = (1 + \mathbf{x}^T\mathbf{x}')^2$$

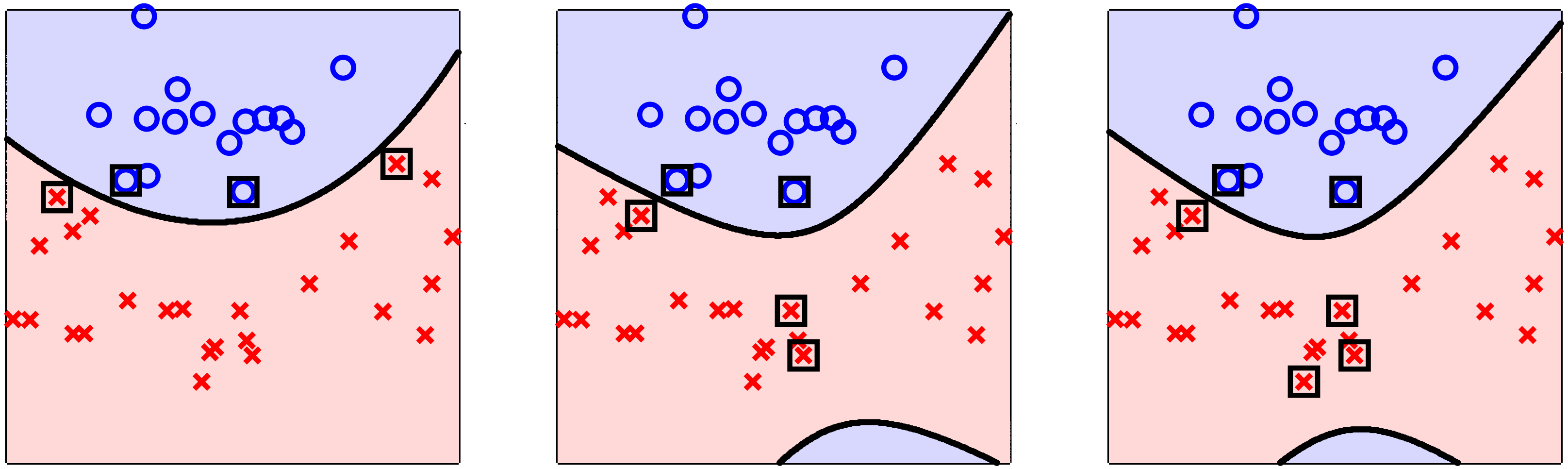
$$K_2(\mathbf{x}, \mathbf{x}') = 1 + 2\gamma\mathbf{x}^T\mathbf{x}' + \gamma^2(\mathbf{x}^T\mathbf{x}')^2 = (1 + \gamma\mathbf{x}^T\mathbf{x}')^2$$

equivalent power, different inner product.

$K_2(\mathbf{x}, \mathbf{x}')$ is commonly used.

實現

Poly-2 kernel in practice



$(1 + 0.001\mathbf{x}^T\mathbf{x})^2$ $1 + \mathbf{x}^T\mathbf{x}^0 + (\mathbf{x}^T\mathbf{x})^2$ $(1 + 1000\mathbf{x}^T\mathbf{x})^2$

picture from coursera, 《機器學習技法》 - 林軒田

調整 gamma 參數得出不一樣的分類曲線

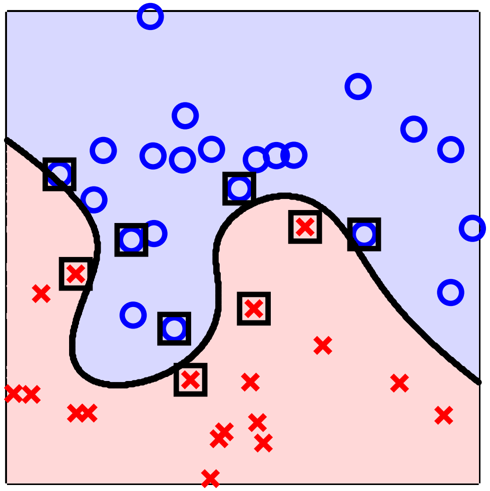
一般化多項式核

General polynomial kernel

$$
\begin{aligned}
K_2(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^2, \gamma > 0, \zeta \geq 0 \\
K_3(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^3, \gamma > 0, \zeta \geq 0 \\
&\vdots \\
K_Q(\mathbf{x}, \mathbf{x}') &= (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q, \gamma > 0, \zeta \geq 0
\end{aligned}
$$

多項式內核實踐

Polynomial kernel in practice



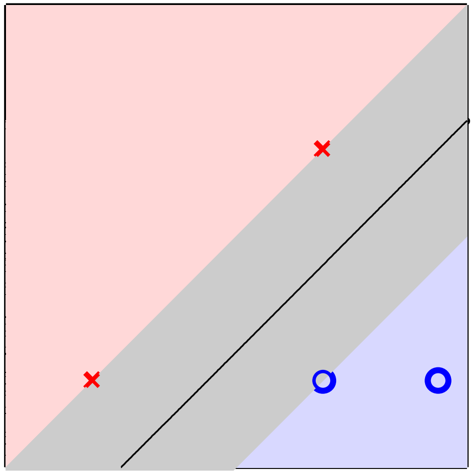
由 Poly-2 Kernel，我們可以再做更多變化，
常數項用 zeta 當參數、特徵空間轉換用 Q 當參數，
加上原本的 gamma 參數，
Polynomial SVM 可以很自由地調整 Kernel 參數來得到更好的分類效果。

picture from coursera, 《機器學習技法》 - 林軒田

線性核

Linear kernel

$$K_1(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^T \mathbf{x}')$$
$$K_Q(\mathbf{x}, \mathbf{x}') = (\zeta + \gamma \mathbf{x}^T \mathbf{x}')^Q, \gamma > 0, \zeta \geq 0$$



優點是模型較為簡單，也因此比較安全，不容易 overfit；可以算出確切的 W 及 Support Vectors，解釋性較好。

缺點就是，限制會較多，如果資料點非線性可分就沒用。

picture from coursera, 《機器學習技法》 - 林軒田

無限多維內核

Infinite dimensional kernel

Infinite dimension ϕ?

YES, if K(x, x') efficiently computable
(高效的計算能力)

$$K(x, x') = exp(-(x - x')^2)$$

$$= exp(-x^2 + 2xx' - x'^2)$$

$$= exp(-x^2)exp(-x'^2)exp(2xx')$$

$$= exp(-x^2)exp(-x'^2)(\sum_{i=0}^{\infty} \frac{(2xx')^i}{i}) \text{ (Taylor expansion)}$$

Infinite dimensional kernel

$$= \sum_{i=0}^{\infty} (exp(-x^2)exp(-x'^2)\frac{2^i}{i}x^i x'^i))$$

$$= \sum_{i=0}^{\infty} (exp(-x^2)exp(-x'^2)\frac{\sqrt{2^i}}{i}\frac{\sqrt{2^i}}{i}x^i x'^i))$$

$$= \sum_{i=0}^{\infty} (\frac{\sqrt{2^i}}{i}x^i exp(-x^2)) \sum_{i=0}^{\infty} (\frac{\sqrt{2^i}}{i}x'^i exp(-x'^2)))$$

$$= \phi(x)^T \phi(x')$$ 非線性轉換

$$\phi(x) = exp(-x^2) \cdot (1, \frac{\sqrt{2}}{1}x, \frac{\sqrt{2^2}}{2}x^2, \ldots,)$$ 無限維度的轉換

指數函數 exp(-(x-x')^2) 就是一個對 X 的無限多維轉換，
由此我們可以推導出無限多維轉換 Kernel，也稱為 Gaussian kernel。

高斯核

Gaussian kernel

$$K(\mathbf{x}, \mathbf{x'}) = exp(-\gamma\|\mathbf{x} - \mathbf{x'}\|^2), \gamma > 0$$

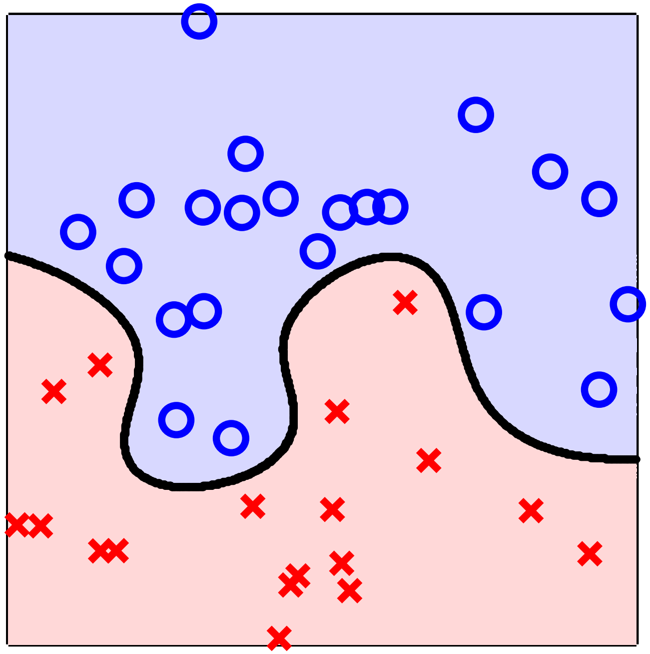
$$g_{SVM}(\mathbf{x}) = sign(\sum_{SV} \lambda_i y_i K(\mathbf{x_i}, \mathbf{x}) + b)$$

$$= sign(\sum_{SV} \lambda_i y_i \|\mathbf{x} - \mathbf{x_i}\|^2 + b)$$

also called radial basis function kernel.

推導出 Gaussian Kernel 之後，
使用 Gaussian Kernel 的 SVM 就是 Gaussian SVM。
Gaussian SVM 演算法會與 Polynomial SVM 一樣，
只是 Kernel 不一樣，由於是無限多維轉換，
我們也不用再去煩惱要用幾次的轉換。

Gaussian kernel



picture from coursera, 《機器學習技法》 - 林軒田

高斯

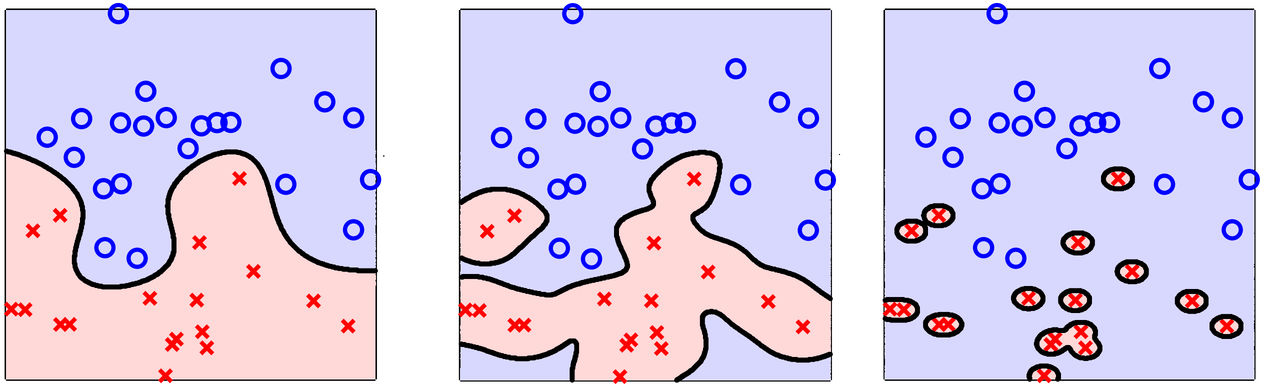
Hard-margin (Gaussian) kernel SVM

Large margin hyperplane + kernel transform

$$\gamma = 1, \gamma = 10, \gamma = 100$$

強度過強時

Overfitting!



訓練資料正確分群，
丟新的測試資料進來，
可能就會出錯。

picture from coursera, 《機器學習技法》 - 林軒田

Gaussian SVM 是無限多維的轉換，因此可以預期它有很強的 power 可以做好分類，
同時又保證 mergin 最大可以避免 overfitting。
但上圖中實驗調整 Gaussian Kernel 的 gamma 參數，其實還是有可能會產生 overfitting，
所以 Gaussian SVM 也不是萬能的，還是要謹慎驗證計算出來的結果。

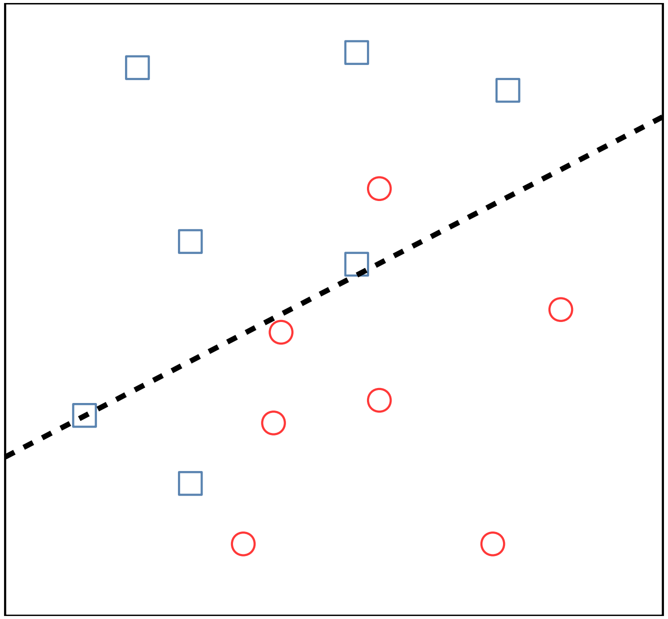
Soft-margin SVM

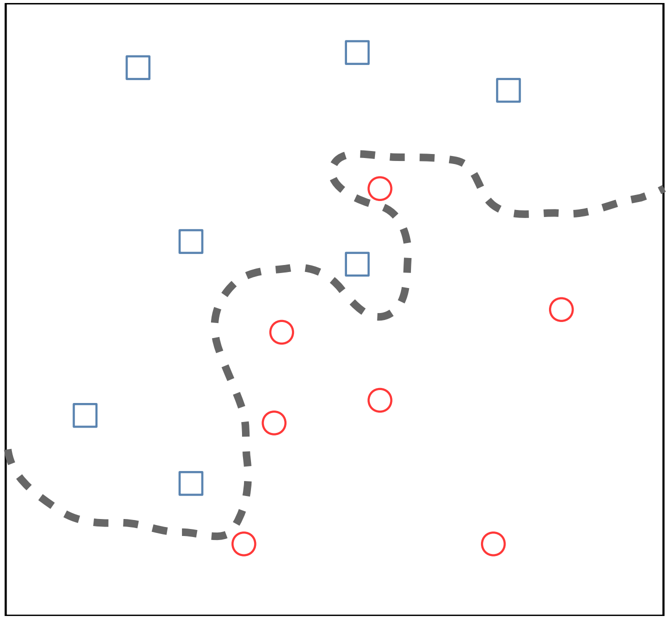
Why soft-margin?

缺點

Disadvantage of hard-margin SVM

由於 Hard Margin SVM 堅持分好資料，所以在高維 Polynomial 及 Gaussian SVM 的學習模型可能會有 Overfitting 的現象，即使 SVM 的 Fat Margin 性質可以避掉一些，但 Overfitting 還是有可能發生，如下圖。





Noise tolerance

Hard-margin SVM:

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\min} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} \\ \text{subject to} \quad & \forall i, y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq 1 \end{aligned}$$

Noise tolerance:

$$\begin{aligned} \underset{\mathbf{w},b}{\arg\min} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}[y_i \neq sign(\mathbf{w}^T\mathbf{z_i} + b)] \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq 1 \text{ for correct} \\ & y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq -\infty \text{ for incorrect} \end{aligned}$$

放棄一些小錯

在這邊我們將 SVM 要最佳化的式子加上了容錯項，在做錯的點上面允許 yi(W^TZi + b) >= 負無限大，也就是說錯了也沒關係，如此最佳化時就能允許錯誤了。其中 C 可以調整 large margin 跟容錯項，C 越大代表容錯越小，C 越小代表容錯越大。

犯錯值的量

Margin violation

$$\underset{\mathbf{w},b}{\arg\min} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}[y_i \neq sign(\mathbf{w}^T\mathbf{z_i} + b)]$$

$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq 1 - \infty[y_i \neq sign(\mathbf{w}^T\mathbf{z_i} + b)]$$

let margin violation

$$\xi_i = [y_i \neq sign(\mathbf{w}^T\mathbf{z_i} + b)]$$

$$\underset{\mathbf{w},b,\xi}{\arg\min} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

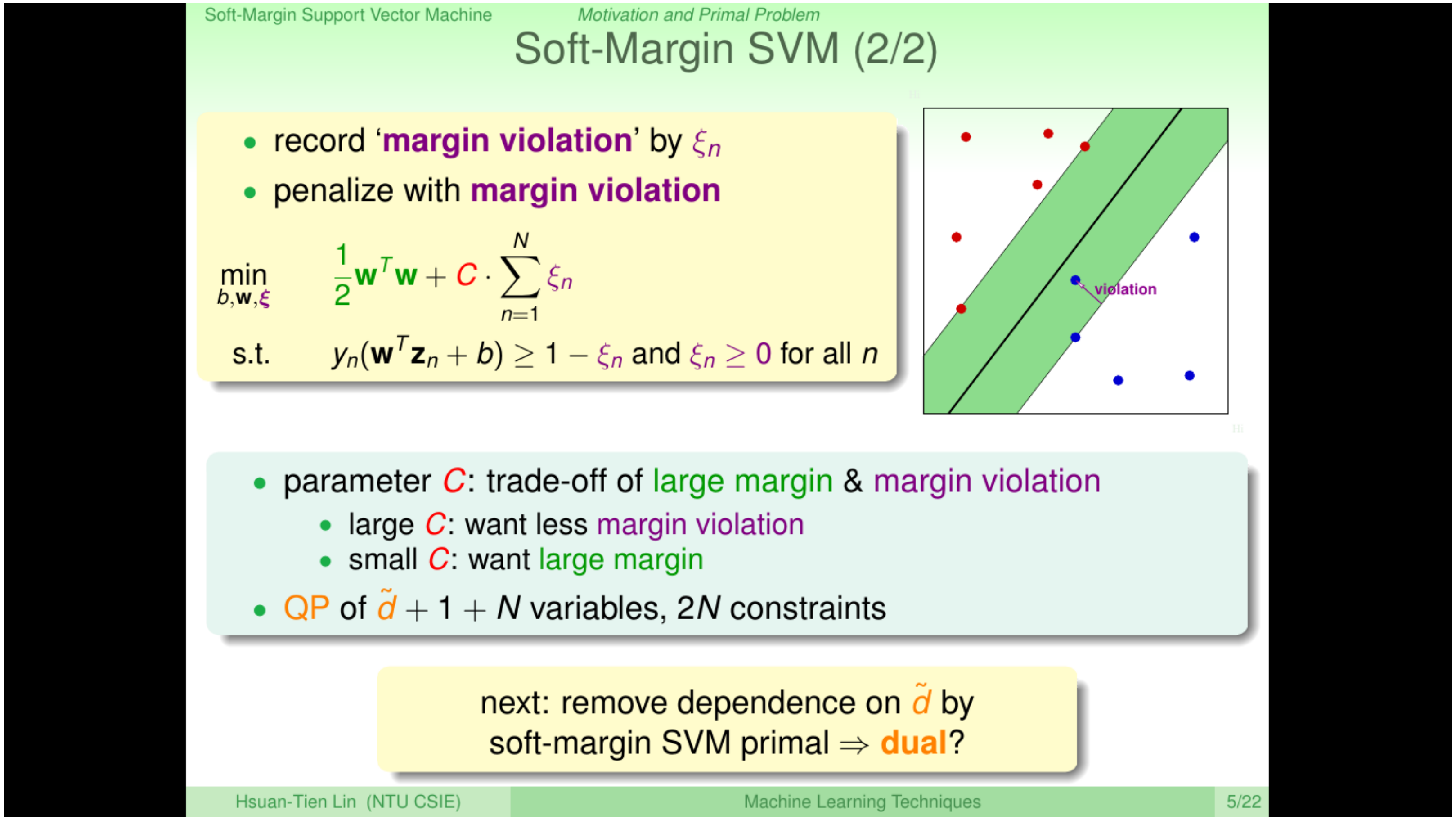
$$\text{subject to} \quad y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq 1 - \xi_i, \xi_i \geq 0$$

整理一下 Soft Margin SVM 的數學式，兩個限制式可以再合起來如上。但這個數學式不再是原來的 QP 問題了，而且目前的數學式是無法記錄犯了小錯或是大錯。

所以我們想辦法講原來的數學式轉換成可以記錄犯了多少錯，將犯了多少錯記錄在 xi 裡，如此就將原來的數學式轉換成線性的形式了。

xi 記錄的錯誤如下圖所示，表示 xi 違反了 margin 多少量，離 margin 越遠的，xi 值會越大，所以在 Soft-Margin SVM 我們除了最佳化 w, b，也要最佳化 xi。

其中 C 可以調整 large margin 跟容錯項，C 越大代表容錯越小，C 越小代表容錯越大，margin 也就越大。



coursera,《機器學習技法》 -林軒田

拉格朗日

Lagrange Dual

仿造 Dual 的方法解 Soft Margin SVM。
由於多了 xi 這 N 個變數，
所以必須多出 N 個 bn Largrange Multiplier。

Primal problem:

$$\begin{aligned} \arg\min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{z_i} + b) \geq 1 - \xi_i, \xi_i \geq 0 \end{aligned}$$

Lagrange function:

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$+\sum_{i=1}^{N}\alpha_i(1-\zeta_i - y_i(\mathbf{w}^T\mathbf{z_i} + b)) + \sum_{i=1}^{N}\beta_i(-\zeta_i)$$

$$\arg\max_{\alpha,\beta}\left(\arg\min_{\mathbf{w},b,\xi}\mathcal{L}(\mathbf{w}, b, \xi, \alpha, \beta)\right)$$

Simplify ξ and β

然後對 xi 偏微分，得到在最佳解時 0 = C - an - bn。
將最佳解時的條件帶回原式，
我們會得到更簡化的式子如下圖所示。

$$\mathcal{L}(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$+\sum_{i=1}^{N}\alpha_i(1-\xi_i-y_i(\mathbf{w}^T\mathbf{z_i}+b)) + \sum_{i=1}^{N}\beta_i(-\xi_i)$$

$$\frac{\partial\mathcal{L}}{\partial\xi_i} = 0 = C - \alpha_i - \beta_i$$

$$\beta_i = C - \alpha_i \geq 0$$

$$0 \leq \alpha_i, \alpha_i \leq C$$

Simplifications

我們繼續對數學式簡化，對 b 進行偏微分、對 w 進行偏微分，都可以得到在最佳解時的限制式。

$$\arg\max_{\alpha}\Big(\arg\min_{\mathbf{w},b}\frac{1}{2}\mathbf{w}^T\mathbf{w}+\sum_{i=1}^{N}\alpha_i(1-y_i(\mathbf{w}^T\mathbf{z_i}+b))\Big)$$

$$\frac{\partial\mathcal{L}}{\partial b}=0\Rightarrow\sum_{i=1}^{N}\alpha_i y_i=0$$

$$\frac{\partial\mathcal{L}}{\partial\mathbf{w}}=0\Rightarrow\mathbf{w}=\sum_{i=1}^{N}\alpha_i y_i\mathbf{z_i}$$

$$\arg\max_{\alpha}\Big(\arg\min_{\mathbf{w},b}\frac{1}{2}\mathbf{w}^T\mathbf{w}+\sum_{i=1}^{N}\alpha_i-\sum_{i=1}^{N}\alpha_i y_i\mathbf{w}^T\mathbf{z_i}-b\sum_{i=1}^{N}\alpha_i y_i\Big)$$

Soft-margin dual SVM

經過上述處理之後，
Soft Margin SVM 的標準對偶數學式如下圖所示，
跟 Hard Margin 不一樣的地方是 ai的上邊界
這是一個 QP 問題，
只要帶入 QP Solver 就可以解出來。

$$\arg\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \mathbf{z_i}^T\mathbf{z_j} - \sum_{i=1}^{N}\alpha_i$$

$$\text{subject to} \quad \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{z_i}$$

$$0 \le \alpha_i \le C$$

Kernel soft-margin SVM

$$
\begin{aligned}
\arg\min_{\alpha} \quad & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(\mathbf{x_i},\mathbf{x_j}) - \sum_{i=1}^{N}\alpha_i \\
\text{subject to} \quad & \sum_{i=1}^{N}\alpha_i y_i = 0 \\
& \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{z_i} \\
& 0 \le \alpha_i \le C
\end{aligned}
$$

最後結果 $SVM(\mathbf{x}) = sign\left(\mathbf{w}^T\mathbf{x} + b\right) = sign\left(\sum_{i\in SV}\alpha_i y_i K(\mathbf{x_i},\mathbf{x}) + b\right)$

SV for support vectors

支持向量？

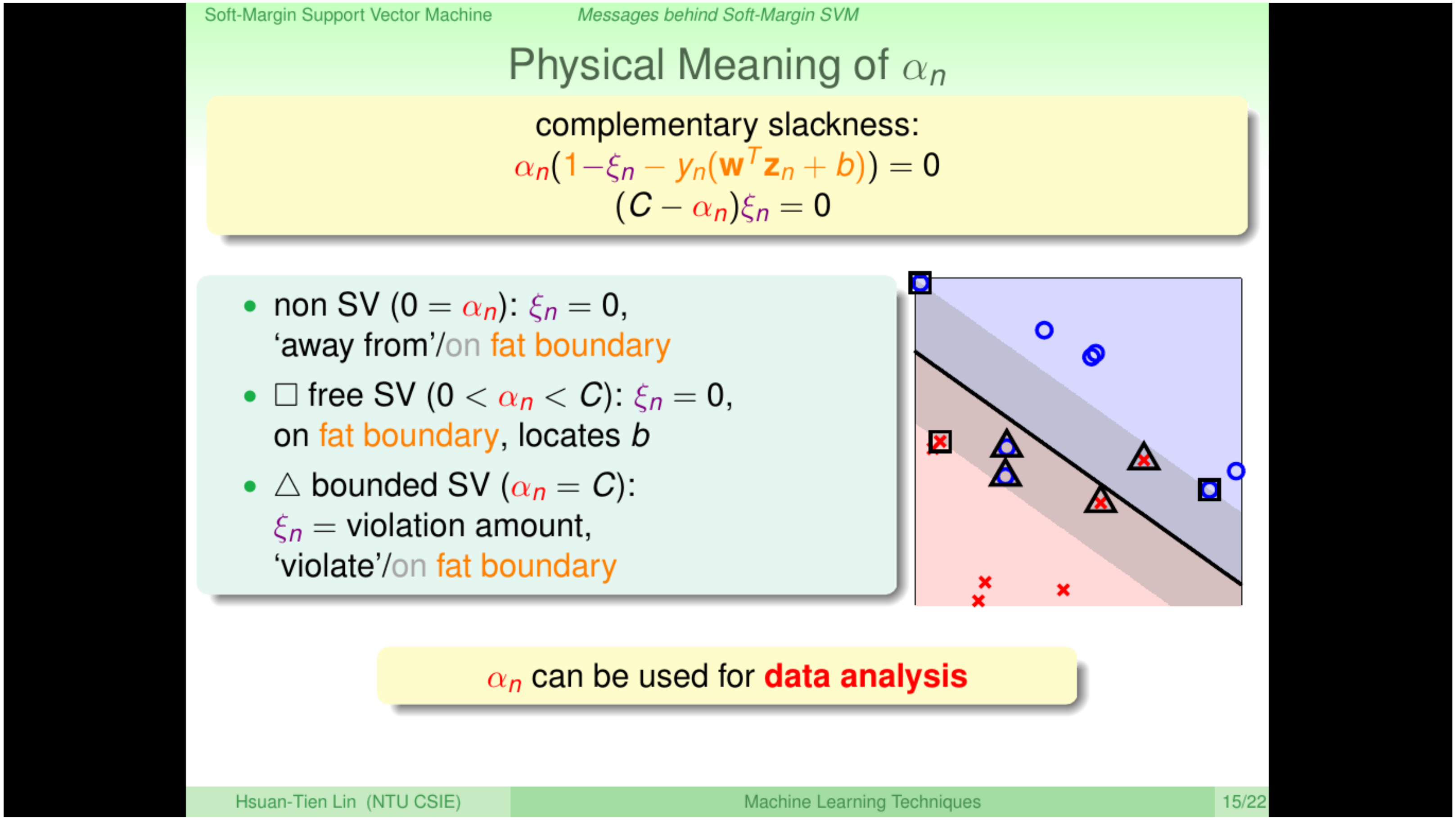
Support vectors?

解決二次規劃問題：

Solve quadratic programming problem:

$$0 < \alpha_i < C$$

使用特出的 alpha 來計算出 w 及 b，這些 alpha 隱含著什麼物理意義呢？其中那些 alpha=0 的，就是對於 margin 沒有意義的點。alpha 大於 0 小於 C 的就是在邊界上的點。alpha = C 的，就是代表 xi 有值的點，就代表有違反邊界的點。我們可以利用 alpha 的性質來做一些資料分析。

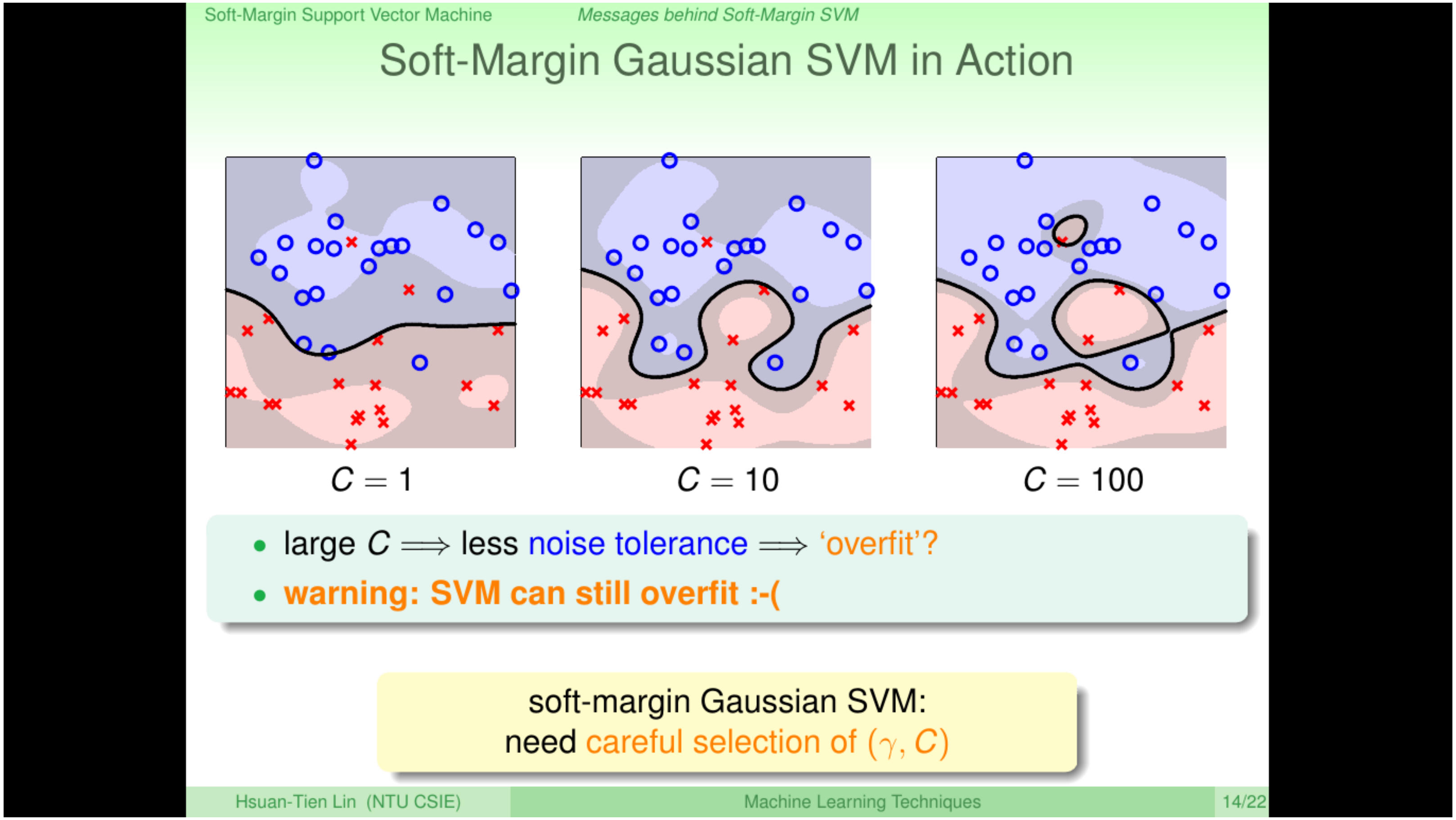


coursera, 《機器學習技法》 - 林軒田

高斯

Soft-margin Gaussian SVM

觀察一下 Soft Margin Gaussian SVM，其實如果使用不當還是會有 Overfitting 的現象產生。



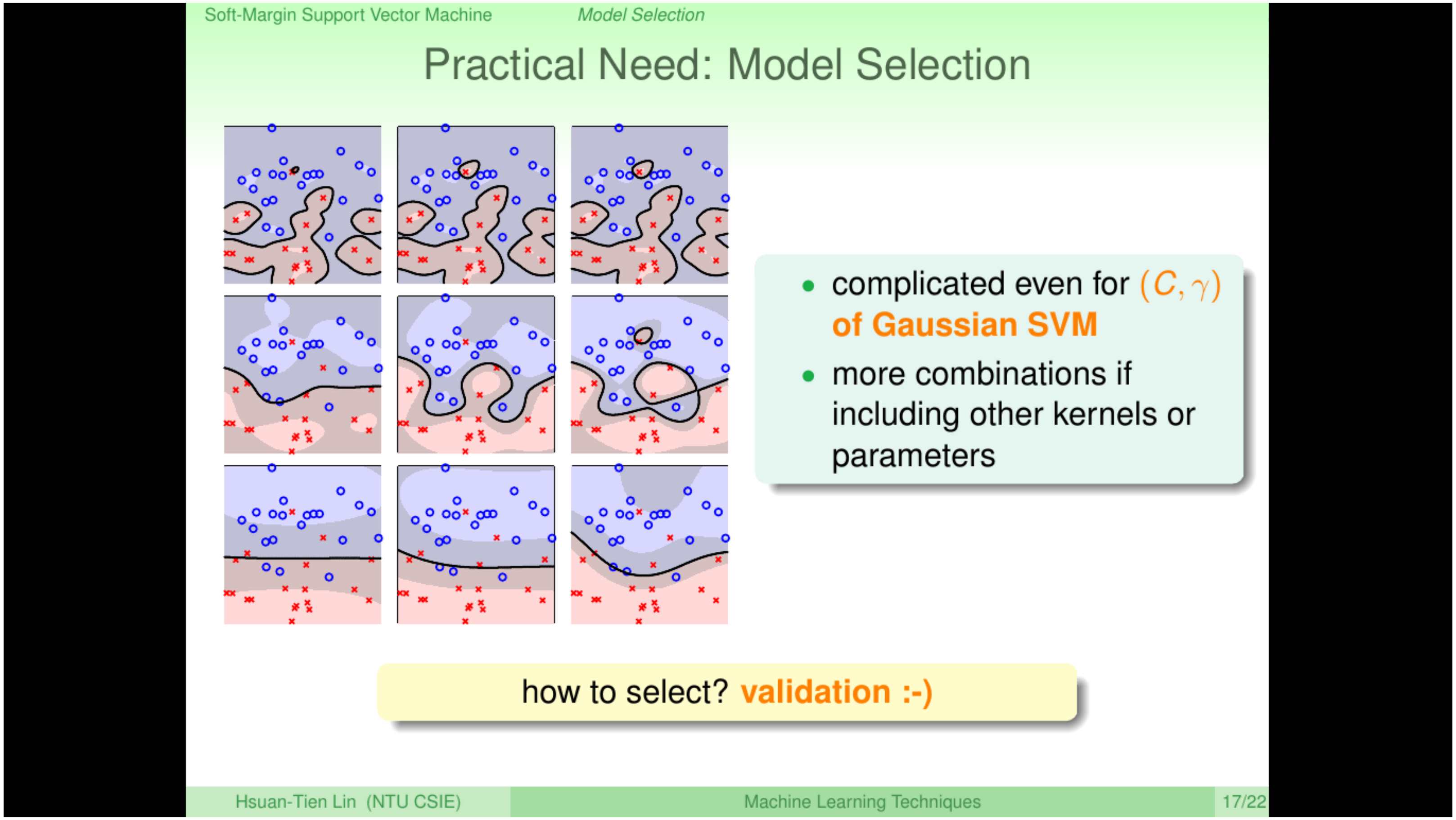
強度調整

coursera, 《機器學習技法》 - 林軒田

模式選擇

Model selection

我們可以使用 C 及 gamma 來挑整 Soft Gaussian SVM，那怎麼挑選 C 及 gamma 參數呢？



coursera, 《機器學習技法》 - 林軒田

模型訓練 - 模型測試
模型訓練 - 驗證 - 模型測試

鬆弛變量

Slack variables

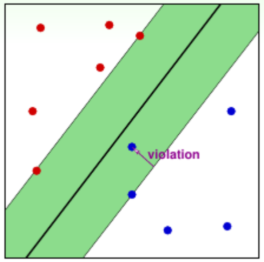
觀察一下 Soft Margin SVM 的容錯項，我們可以把原本的限制式整合到要最小化的式子裡來看看，如下圖所示，如此就沒有限制式了。

Slack Variables ξ_n

- record '**margin violation**' by ξ_n
- penalize with **margin violation**

$$\min_{b,\mathbf{w},\xi} \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C \cdot \sum_{n=1}^{N} \xi_n$$

$$\text{s.t.} \quad y_n(\mathbf{w}^T\mathbf{z}_n + b) \geq 1 - \xi_n \text{ and } \xi_n \geq 0 \text{ for all } n$$



on any $(b, \mathbf{w})$, ξ_n = **margin violation** $= \max(1 - y_n(\mathbf{w}^T\mathbf{z}_n + b), 0)$

- $(\mathbf{x}_n, y_n)$ violating margin: $\xi_n = 1 - y_n(\mathbf{w}^T\mathbf{z}_n + b)$
- $(\mathbf{x}_n, y_n)$ not violating margin: $\xi_n = 0$

'unconstrained' form of soft-margin SVM:

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

coursera, 《機器學習技法》 - 林軒田

杜：無實用性

l_2 regularized model

再仔細觀察一下沒有限制式的 SVM 數學式，發現形式跟 L2 regularized Logistic Regression 有點像，只是沒有限制式的 SVM 數學式有個 max 的函數在裡面，這樣的數學式不再是一個 QP 問題了，然後也不是一個可以微分的式子，因此很難最佳化。

Unconstrained Form

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

familiar? :-)

$$\min \quad \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum \widehat{\mathrm{err}}$$

just L2 regularization

$$\min \quad \frac{\lambda}{N}\mathbf{w}^T\mathbf{w} + \frac{1}{N}\sum \mathrm{err}$$

with shorter **w**, another parameter, and special err

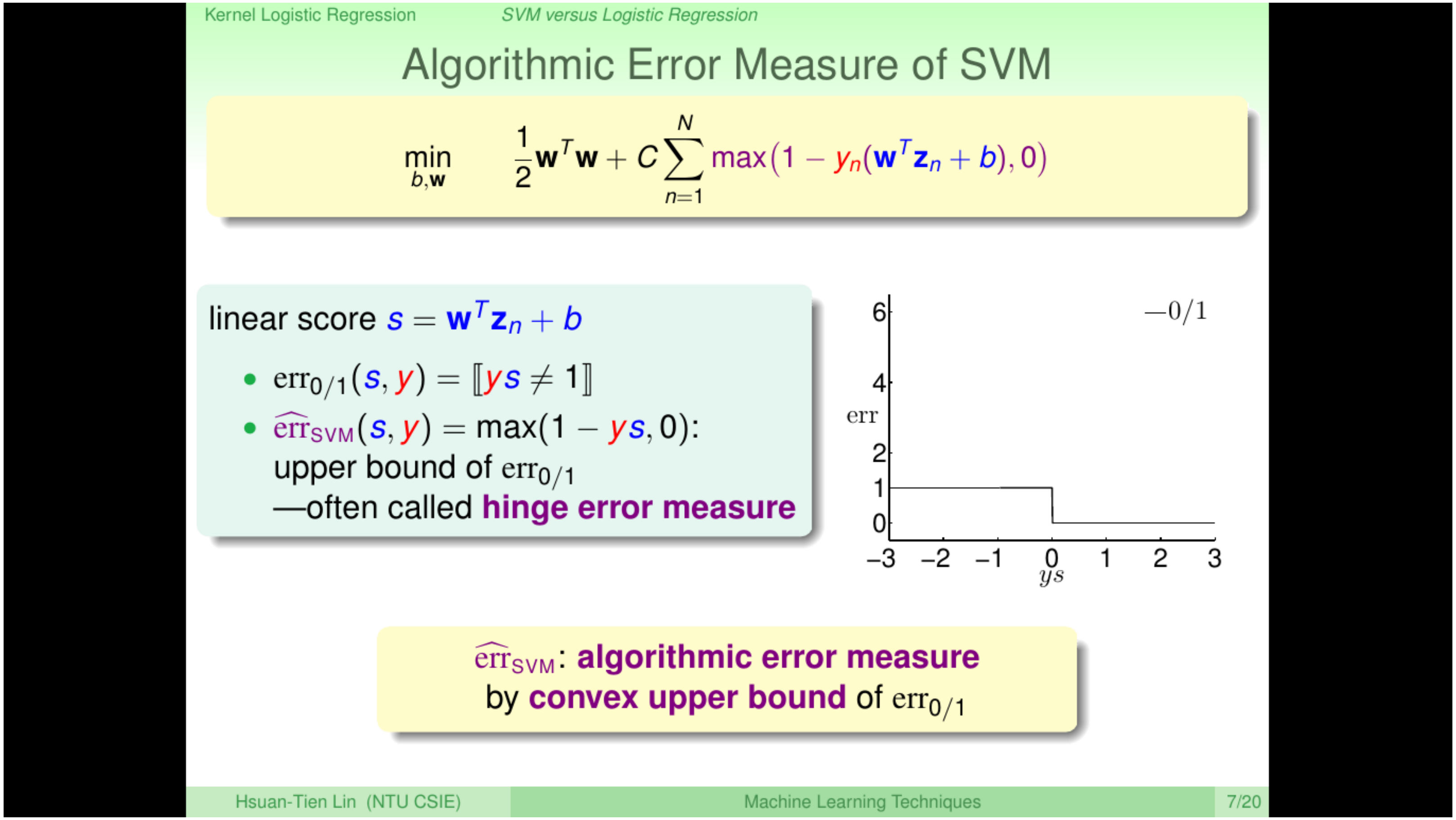
why not solve this? :-)

- not QP, **no (?) kernel trick**
- $\max(\cdot, 0)$ **not differentiable**, harder to solve

coursera,《機器學習技法》- 林軒田

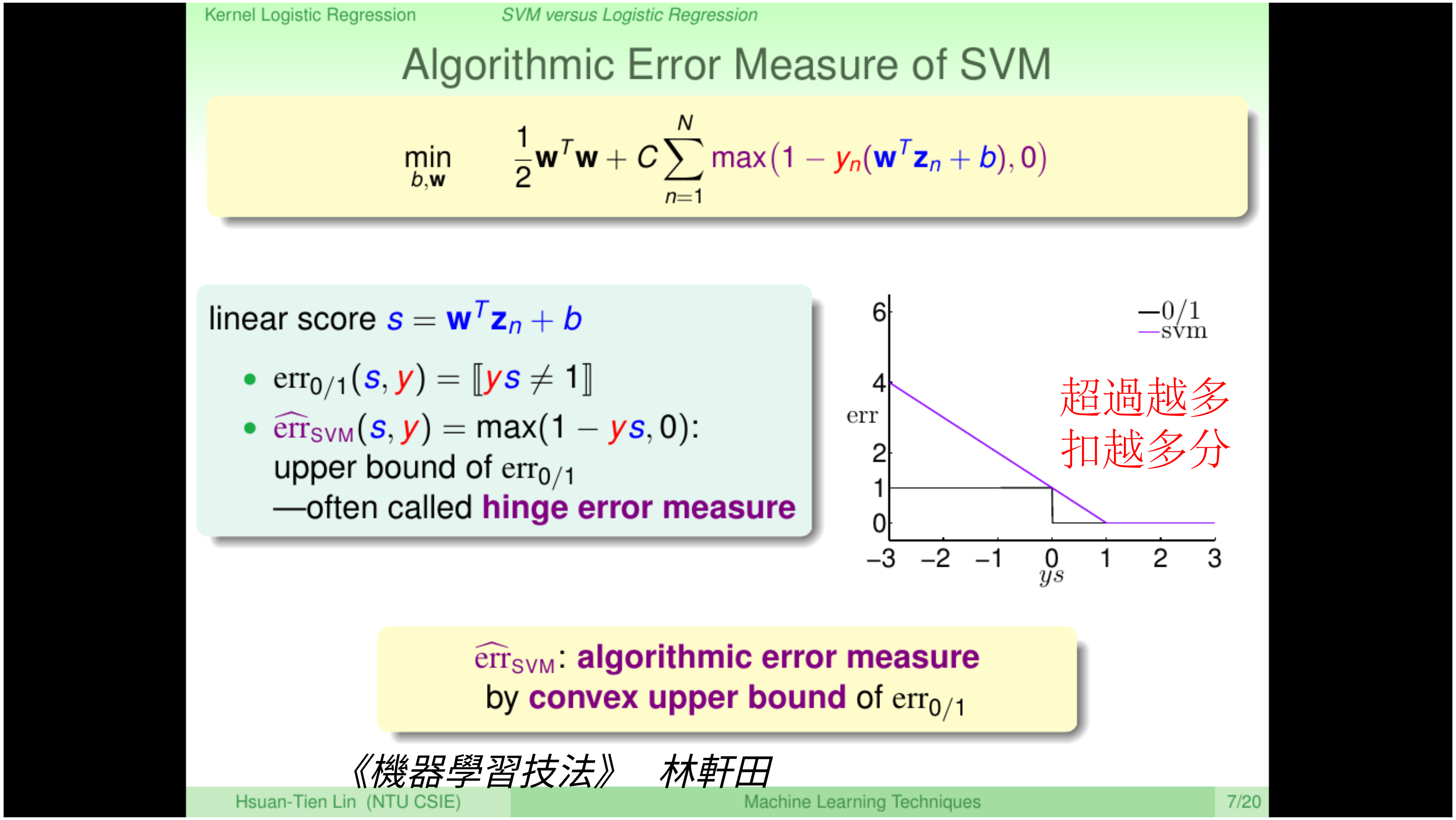
杜:PASS

Hinge error



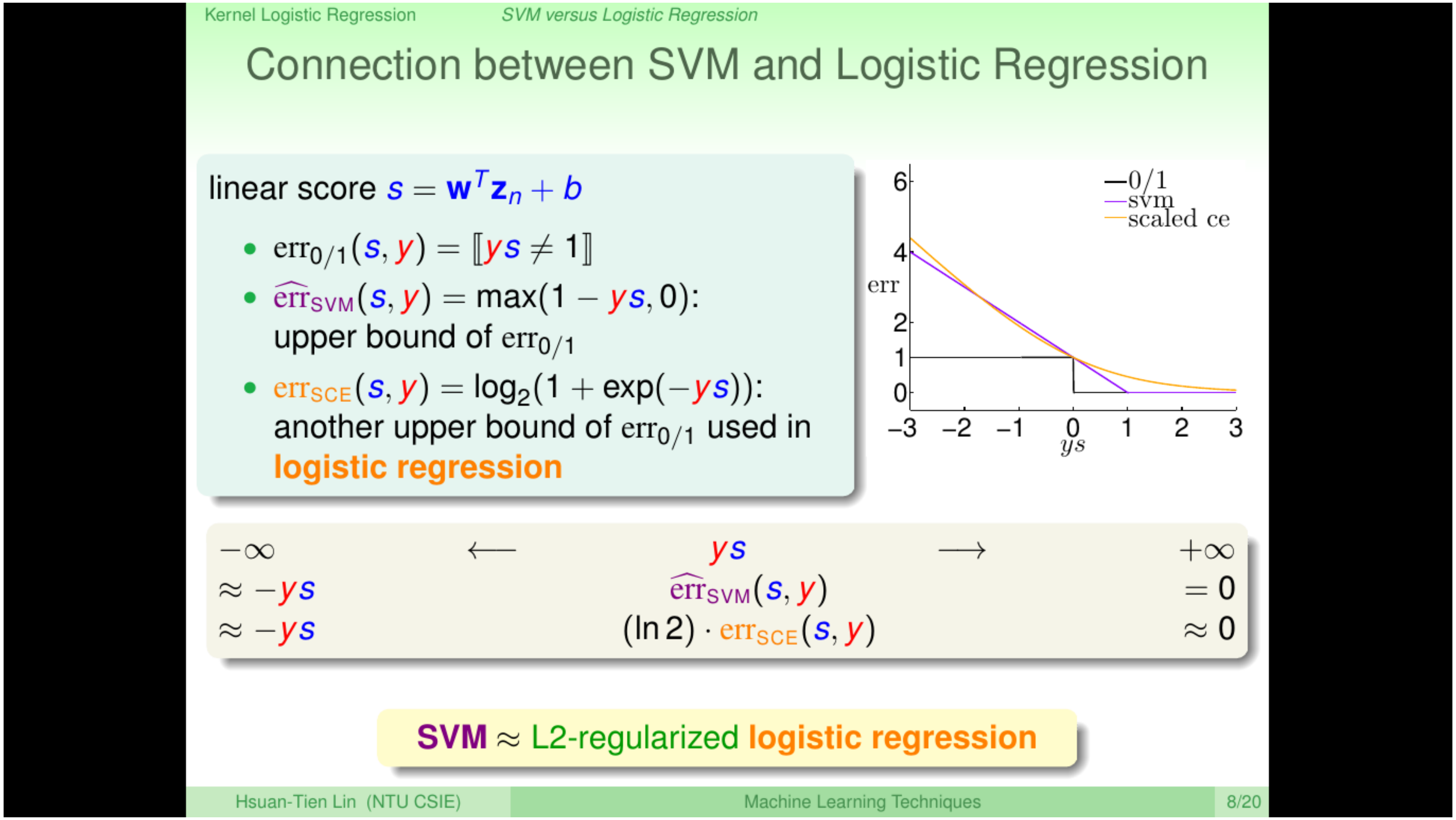
coursera, 《機器學習技法》 - 林軒田

Hinge error



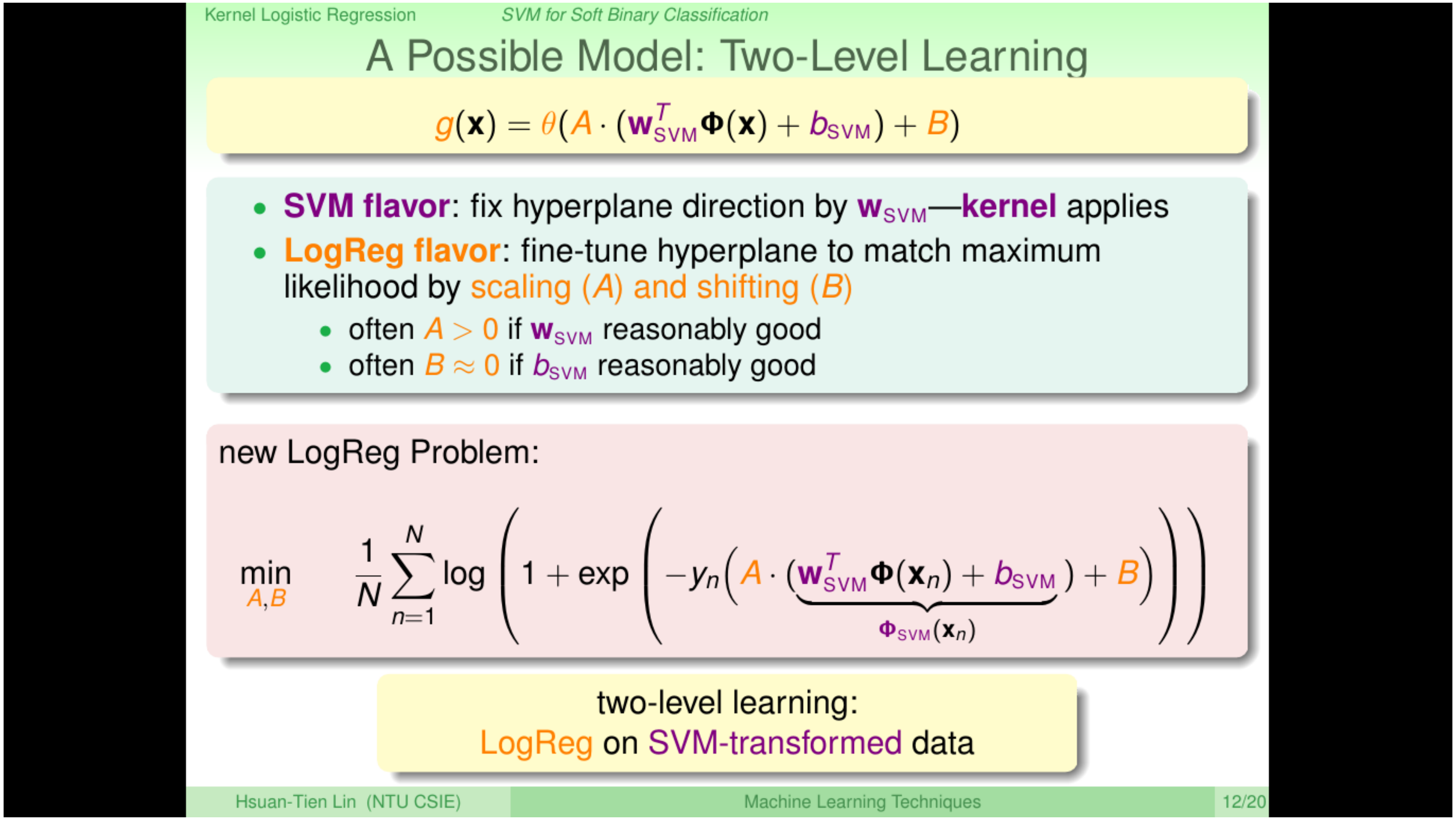
coursera, -

仔細觀察一下 SVM 錯誤衡量 function，其實 err_svm 跟 err_0/1 在數線圖上
err_svm 會是 err_0/1 的上界，且邊界也很接近，
所以我們可以說 SVM 與 L2-regularized logistic regression 是很接近的。
邏輯回歸



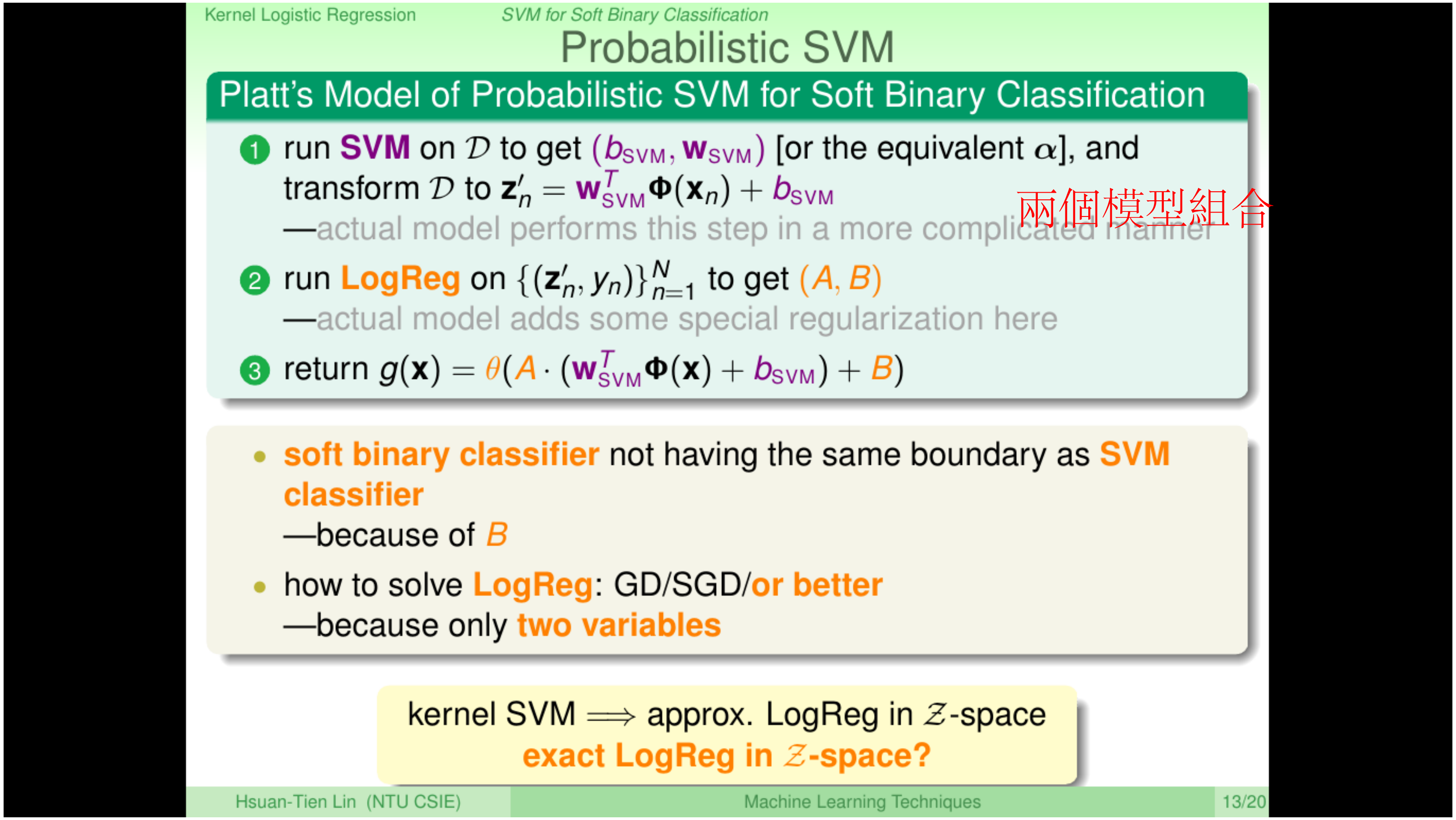
coursera, 《機器學習技法》 - 林軒田

怎麼讓 SVM 做 Logistic Regression 呢？
一個做法是使用 Two-Level Learning，也就是先做 SVM，
然後將原來的 X 計算分數（轉換到 SVM 的空間）之後，
再對新的 X 以及 Y 做 Logistic Regression 學習 A 與 B。



coursera,《機器學習技法》- 林軒田

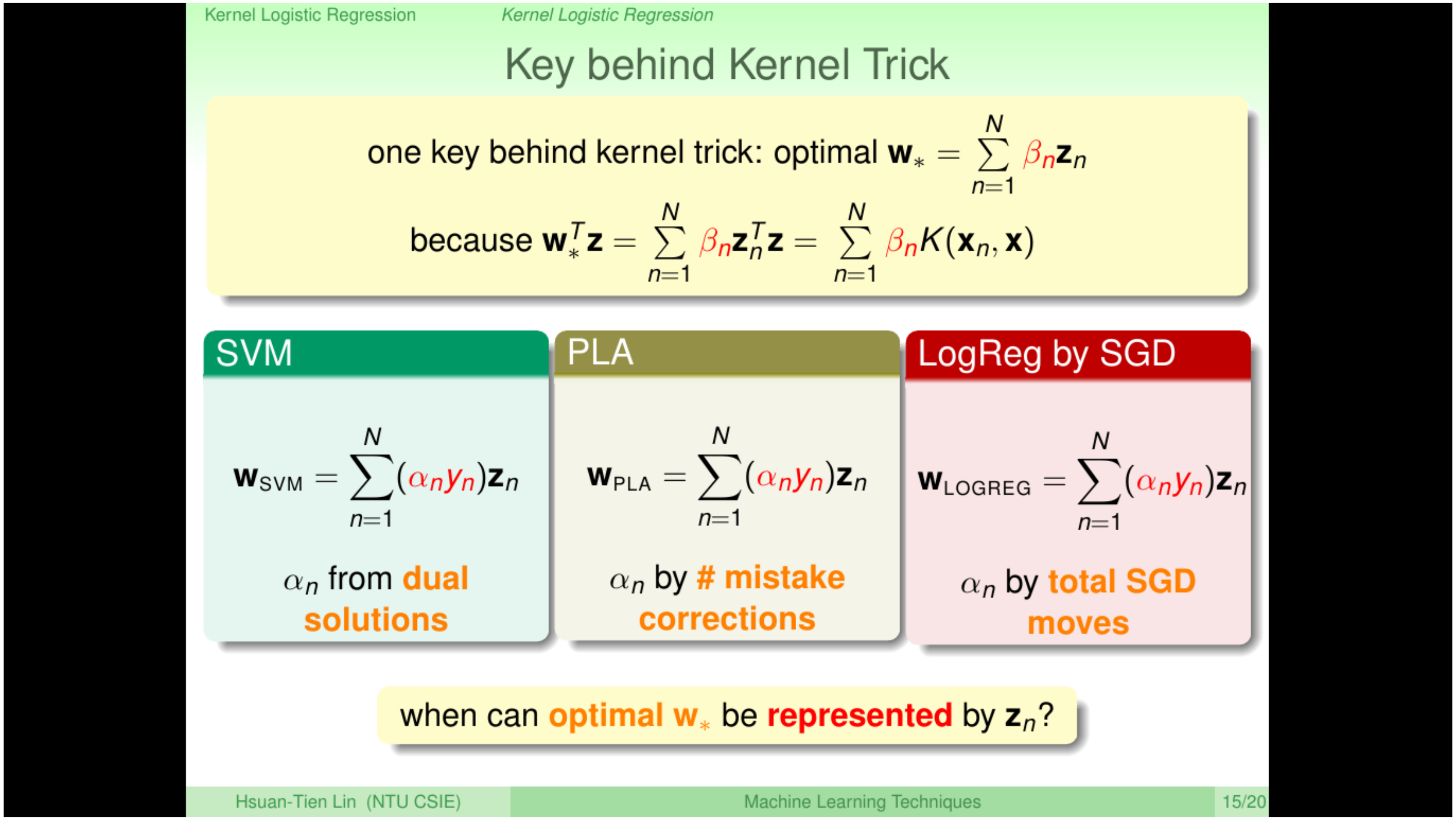
這就是 Probabilistic SVM，具體演算法如下，
但仔細研究這個算法的背後意涵，
這樣的做法並不是讓 Logistic Regression 在 z 空間做最佳解，
有其它方法可以讓 Logistic 真正在 z 空間算最佳解嗎？



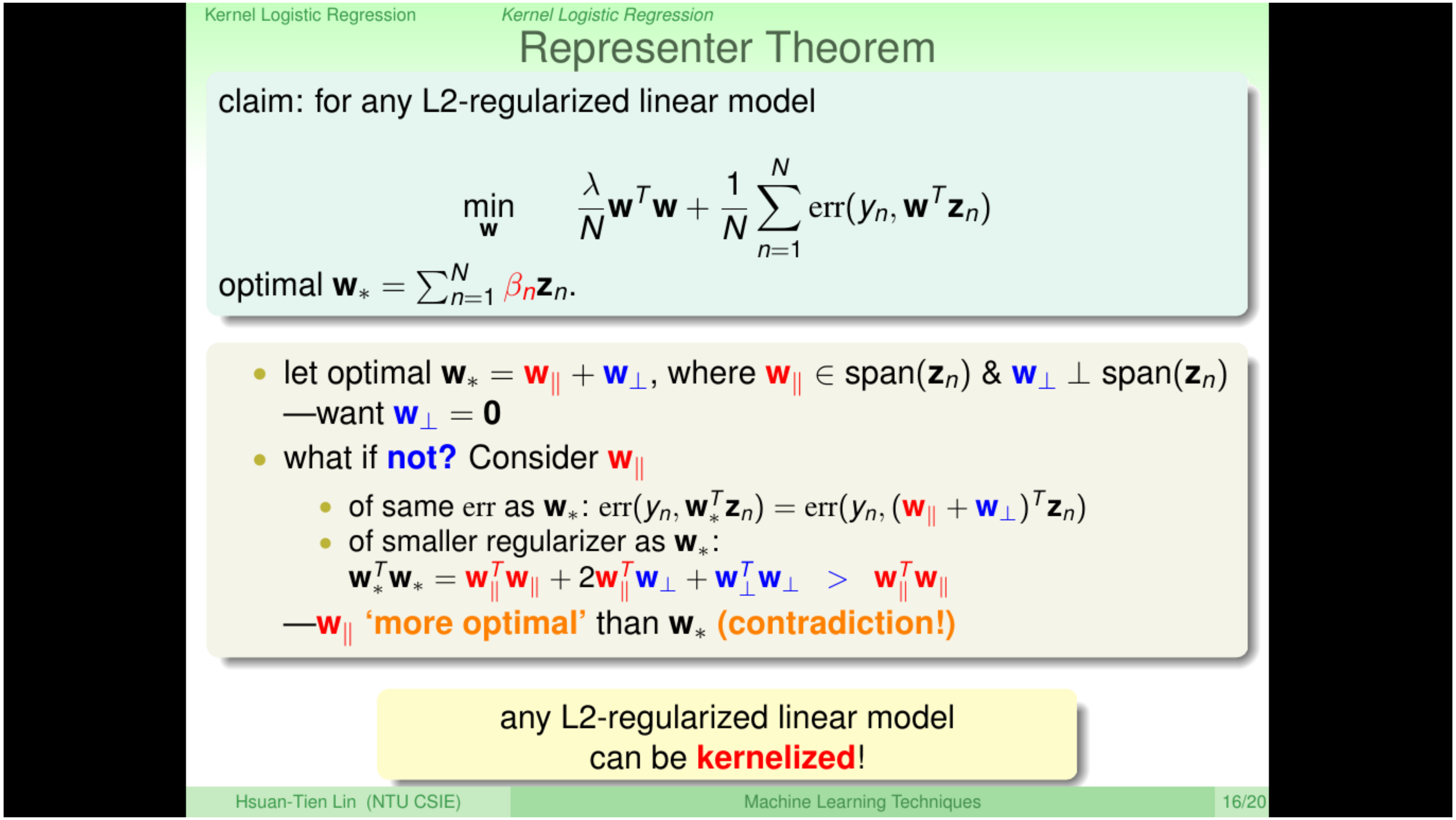
coursera, 《機器學習技法》 - 林軒田

Kernel Trick 背後的關鍵

我們了解一下 SVM 使用的 Kernel Trick，SVM 其實有在 z 空間算最佳解，只是用了 Kernel Trick 來省下計算時間，然後算出的 w 其實就是某種 z 空間的資料線性組合。SVM 是取 support vector 的線性組合、PLA 是取錯誤資料的線性組合、Logistic Regression 是取梯度下降的線性組合，所以只要 w 是一種 z 空間的線性組合的形式，那就可以使用 Kernel Trick。



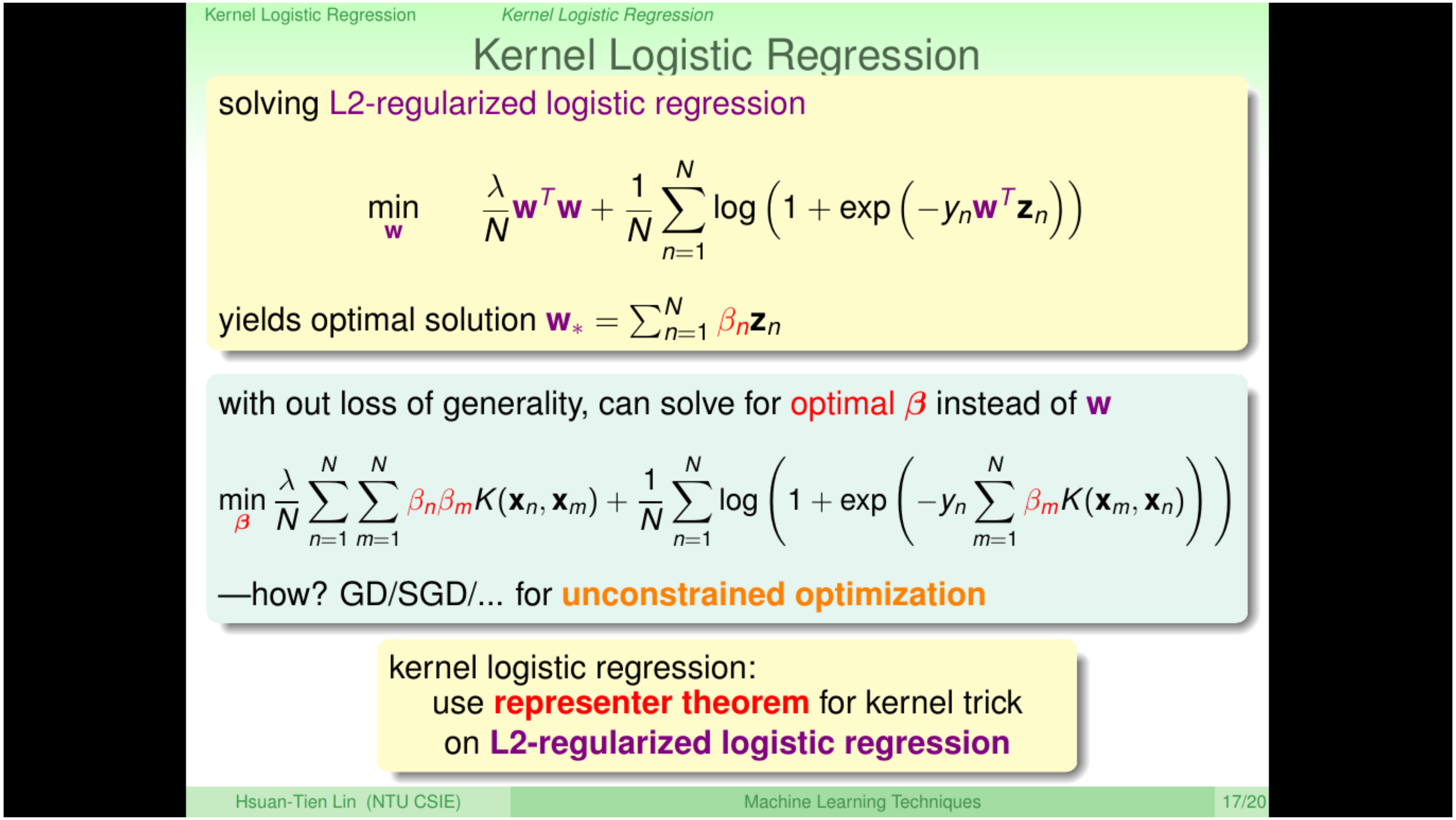
coursera, 《機器學習技法》 - 林軒田



coursera, 《機器學習技法》 - 林軒田

Kernel Logistic Regression

我們將原本的 L2-Regularized Logistic Regression 數學式使用 w 是一種 z 空間線性組合的形式帶進去，得到如下圖數學式，而這數學式是可以最佳化的，所以我們可以使用之前的梯度下降法、隨機梯度下降法來求得最佳的 beta



coursera, 《機器學習技法》 - 林軒田

Kernel Logistic Regression (KLR) : Another View

$$\min_{\boldsymbol{\beta}} \frac{\lambda}{N}\sum_{n=1}^{N}\sum_{m=1}^{N}\beta_n\beta_m K(\mathbf{x}_n,\mathbf{x}_m) + \frac{1}{N}\sum_{n=1}^{N}\log\left(1+\exp\left(-y_n\sum_{m=1}^{N}\beta_m K(\mathbf{x}_m,\mathbf{x}_n)\right)\right)$$

- $\sum_{m=1}^{N}\beta_m K(\mathbf{x}_m,\mathbf{x}_n)$: inner product between variables $\boldsymbol{\beta}$ and transformed data $(K(\mathbf{x}_1,\mathbf{x}_n), K(\mathbf{x}_2,\mathbf{x}_n), \ldots, K(\mathbf{x}_N,\mathbf{x}_n))$
- $\sum_{n=1}^{N}\sum_{m=1}^{N}\beta_n\beta_m K(\mathbf{x}_n,\mathbf{x}_m)$: a special regularizer $\boldsymbol{\beta}^T \mathrm{K}\boldsymbol{\beta}$
- KLR = linear model of $\boldsymbol{\beta}$
  with kernel as transform & kernel regularizer;
  = linear model of $\mathbf{w}$
  with embedded-in-kernel transform & L2 regularizer
- similar for SVM

warning: unlike coefficients α_n in SVM,
coefficients β_n in KLR often non-zero!

coursera, 《機器學習技法》 - 林軒田

Summary

如何使用 SVM 來解 Logistic Regression 的問題，
一個是使用 SVM 做轉換的 Probabilistic SVM，
一個是使用 SVM Kernel Trick 所啟發的 Kernel Logistic Rregression。