# Support Vector Machine

**Keywords–** *Support vector, Classifier, Lagrange multiplier, Kernel function, Duality*

NCKU CSIE
Chia-Lung Hsu 徐嘉隆
kofzerozero@hotmail.com

September 13, 2012

簡報承接自Wen-Sheng Chu學長

# Outline

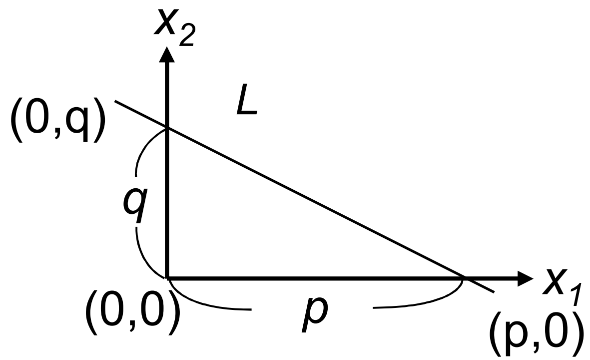
1. Basic Concepts
   - 1.1 Background Knowledge
   - 1.2 Classification Problem
   - 1.3 Nonlinearly Separable Data
   - 1.4 Standard SVM
2. Dual SVM Derivation
   - 2.1 SVM Reminder
   - 2.2 Lagrange Multiplier Method
   - 2.3 The Linearly Separable Case
   - 2.4 The Nonlinearly Separable Case
     - 2.4.1 Soft Margin
     - 2.4.2 The Nonlinear Mapping
3. Training Linear and Nonlinear SVMs
   - 3.1 Training Nonlinear SVMs
     - 3.1.1 Avoid Storage Problem
     - 3.1.2 Speedup Decomposition
   - 3.2 Training Linear SVMs
     - 3.2.1 Decomposition for SVM
     - 3.2.2 Approximations
4. Conclusion
5. Reference

# 1. Basic Concept

- Introduction
  - SVM is a classifier derived from statistical learning theory by Vapnik and Cortes (1995).
  - Relatively easy to use.
  - Suitable for pattern classification or nonlinear regression problems.

# 1.1 Background Knowledge: Linear Equation (1/3)

- Linear Equation Representation

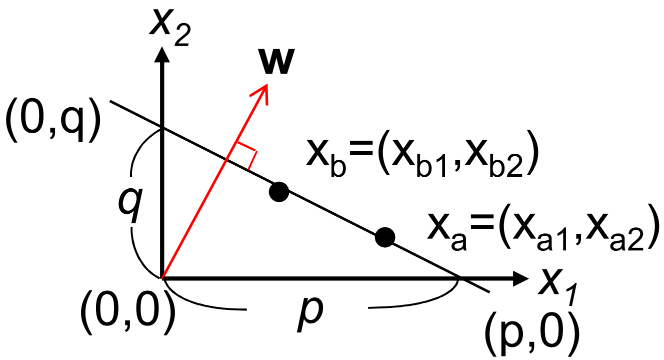



- Assume a line *L* passing two axes at (p,0) and (0,q).
- This line could be represented by $\frac{x_1}{p}+\frac{x_2}{q}=1$
- Reformulate:

$$\boxed{\begin{aligned}&\frac{x_1}{p}+\frac{x_2}{q}=1 \Rightarrow qx_1+px_2-pq=0\\&\Rightarrow [q \quad p]\begin{bmatrix}x_1\\x_2\end{bmatrix}-pq=0 \Rightarrow \mathrm{w}^T\mathrm{x}+b=0\end{aligned}}$$

# 1.1 Background Knowledge: Linear Equation (2/3)

- Normal vector



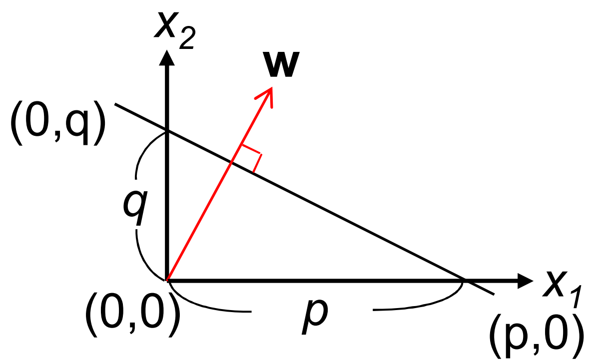

- Assume two points $x_a$ and $x_b$ at the line.
- So we have a vector $(x_a-x_b)$

$$\left.\begin{array}{l} w^T x_a + b = 0 \\ w^T x_b + b = 0 \end{array}\right\} \Rightarrow w^T x_a - w^T x_b = 0 \Rightarrow w^T(x_a - x_b) = 0$$

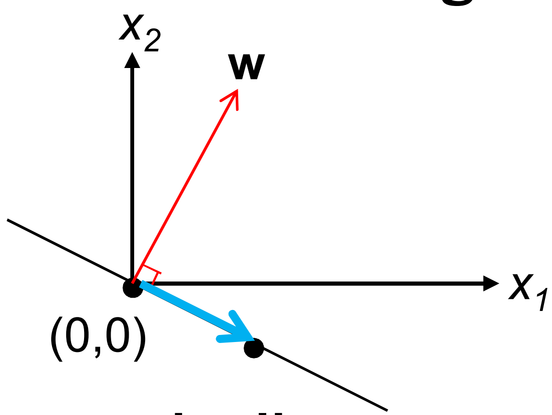
- w is the normal vector of line equation.

# 1.1 Background Knowledge: Linear Equation (3/3)

- How about b = 0?



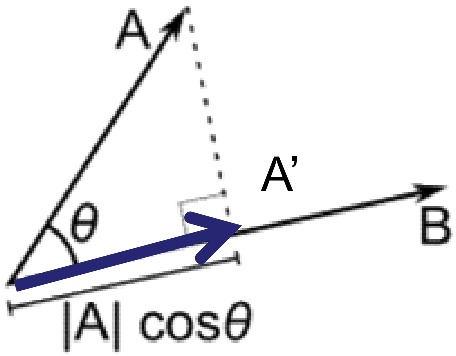

  - Assume b=0, so we get $\mathbf{w}^T\mathbf{x} + b = 0 \Rightarrow \mathbf{w}^T\mathbf{x} = 0 \Rightarrow \mathbf{w}^T(\mathbf{x} - \vec{0}) = 0$
  - It means that x is origin or that x $\perp$w




  - b=0 $\Longrightarrow$ the line passes through origin

# 1.1 Background Knowledge: Inner Product

- Inner Product



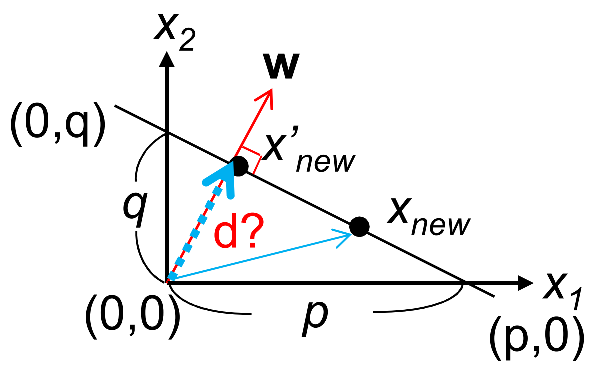

$$< A, B > = A^T B = \|A\| \cdot \|B\| \cos\theta$$

$$\Rightarrow \|A\| \cos\theta = \frac{A^T B}{\|B\|}$$

– The length of projected vector A' is $\dfrac{A^T B}{\|B\|}$

# 1.1 Background Knowledge: Distance to Origin

- What is the distance to the origin?



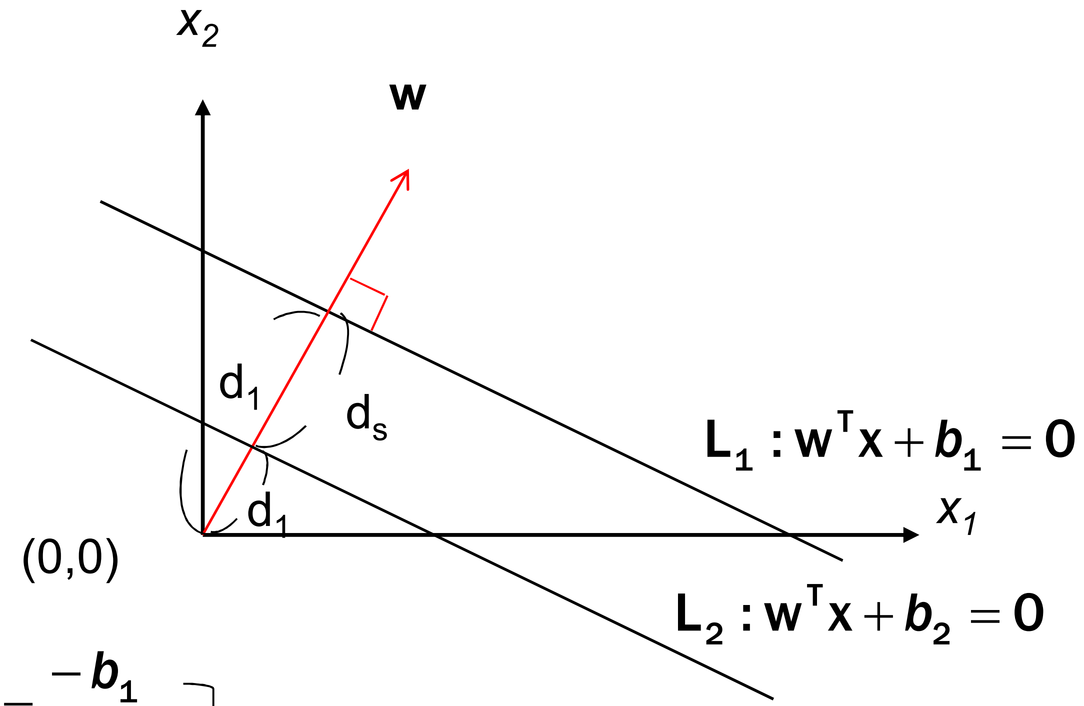

  - Assume there is a point $\mathbf{x}_{new}$ at this line
  - We know the length of projected point $\mathbf{x}'_{new}$ along *w* is d= $\frac{\mathbf{w}^T\mathbf{x}_{new}}{\|\mathbf{w}\|}$
  - And $\mathbf{x}_{new}$ satisfies the equation: $\mathbf{w}^T\mathbf{x}_{new} + b = 0$
  - Combine two terms:

$$\mathbf{w}^T\mathbf{x}_{new} + b = 0 \Rightarrow \frac{\mathbf{w}^T\mathbf{x}_{new} + b}{\|\mathbf{w}\|} = 0 \Rightarrow \frac{\mathbf{w}^T\mathbf{x}_{new}}{\|\mathbf{w}\|} = \frac{-b}{\|\mathbf{w}\|}$$

  - d = $\frac{-b}{\|\mathbf{w}\|}$ = $\frac{\mathbf{w}^T\mathbf{x}_{new}}{\|\mathbf{w}\|}$

# 1.1 Background Knowledge: Distance between Parallel Lines

- What is the distance $d_s$ between two lines $L_1$ and $L_2$?

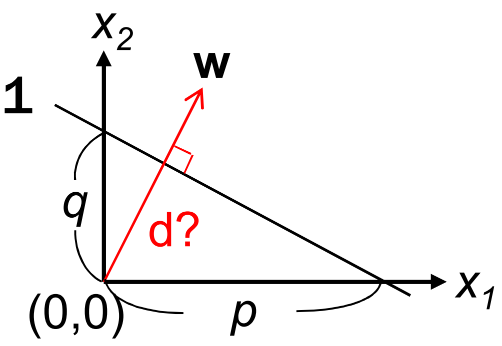



$$\left.\begin{aligned} d_1 &= \frac{-b_1}{\|w\|} \\ d_2 &= \frac{-b_2}{\|w\|} \end{aligned}\right\} \Rightarrow d_s = d_1 - d_2 = \frac{-b_1}{\|w\|} - \frac{-b_2}{\|w\|} = \frac{-(b_1 - b_2)}{\|w\|}$$

# 1.1 Background Knowledge

- **Linear Equation**

Linear equation: $\frac{x_1}{p} + \frac{x_2}{q} = 1$




$$\frac{x_1}{p} + \frac{x_2}{q} = 1 \Rightarrow qx_1 + px_2 - pq = 0$$

$$\Rightarrow \begin{bmatrix} q & p \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - pq = 0$$

$$\Rightarrow \mathbf{w}^T \mathbf{x} - b = 0$$

where w = [$q$  $p$]$^T$, x = [$x_1$  $x_2$]$^T$, and $b$ = $pq$.

James:
W: scaling and rotation
b: translation
b=0?
W: rotation, Scaling = 1,

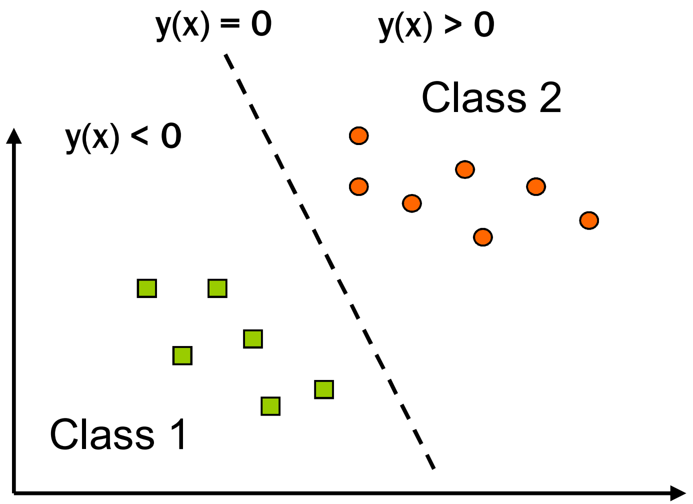
W=w/||w|| * ||w||

Unit vector = direction    magnitude

d=??

# 1.2 Classification Problem

- Classification Problem
  - Assume two classes of data: circles and squares
  - Find a hyperplane (dim>2) to separate two classes



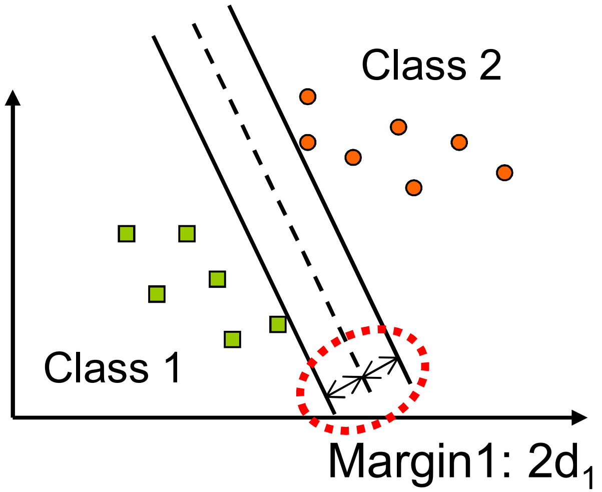

  - A separating hyperplane: $y(x) = w^T x + b = 0$

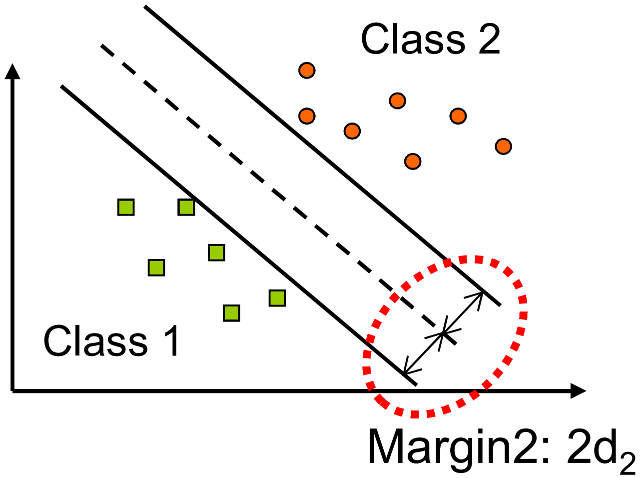
$(w^T x_i) + b > 0$ if $x_i$ is of class 2

$(w^T x_i) + b < 0$ if $x_i$ is of class 1

# 1.2 Classification Problem: Optimal Separating Plane (1/2)

- Optimal Separating Plane




$d_2 > d_1$

- Which is a better classifier?
- James: Why?

Margin 2d=? Margin 2d'=?

$W^T x + b = 0 \Rightarrow w'^T x + b' = 0$

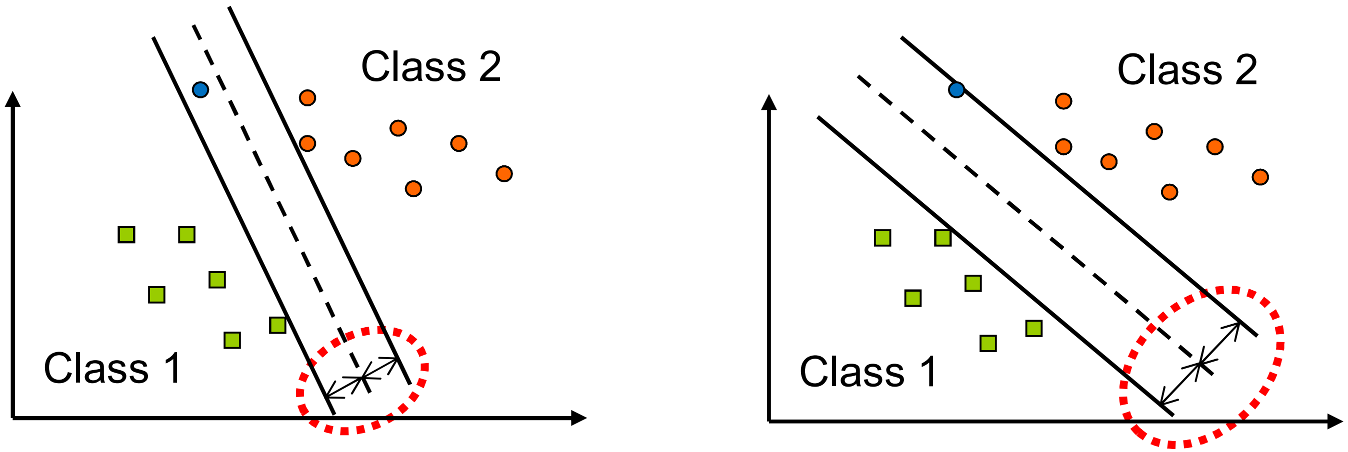
$W^T x + b = a \Rightarrow w'^T x + b' = 1$

$W^T x + b = -a \Rightarrow w'^T x - b' = -1$

Because of scaling and merge into w and b to become w' and b'

# 1.2 Classification Problem: Optimal Separating Plane (2/2)

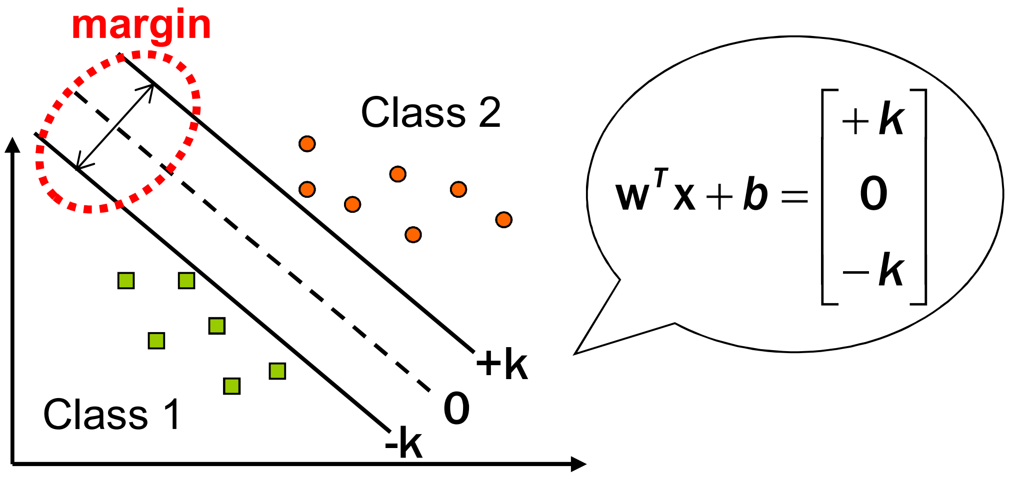
- Consider the outlier data
  - A new point: Blue circle




- The margin provides the potential tolerance to outlier data. Therefore margin2 is better than magin1

# 1.2 Classification Problem: Margin Classifier (1/2)

- Margin Classifier
  - Consider the margin



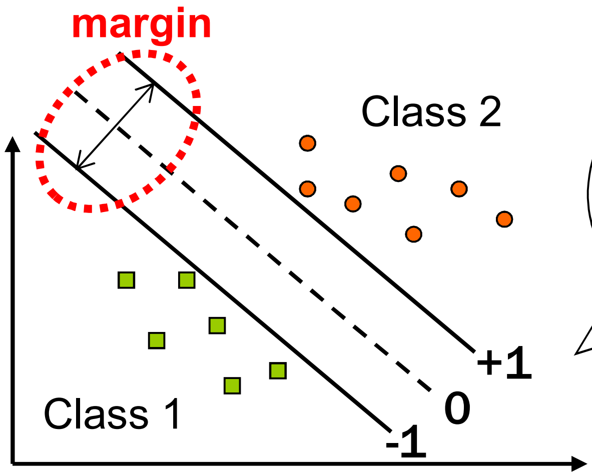

  - Reformulate:

$$w^T x + b = k \Rightarrow \frac{w^T x + b}{k} = 1 \Rightarrow \left(\frac{w}{k}\right)^T x + \frac{b}{k} = 1 \Rightarrow w'^T x + b' = 1$$

$$\textit{where}\ \mathrm{w'} = \frac{\mathrm{w}}{\mathrm{k}} \text{ and } \mathrm{b'} = \frac{\mathrm{b}}{\mathrm{k}}$$

# 1.2 Classification Problem: Margin Classifier (2/2)

- Margin Classifier




$$w^T x + b = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix}$$

$$w^T x + b = 1 \Rightarrow w^T x + b - 1 = 0$$

$$w^T x + b = -1 \Rightarrow w^T x + b + 1 = 0$$

$$\Rightarrow d_s = d_1 - d_2 = \frac{-b_1}{\|w\|} - \frac{-b_2}{\|w\|}$$

$$\Rightarrow \frac{-(b-1)}{\|w\|} - \frac{-(b+1)}{\|w\|} = \frac{-(-2)}{\|w\|}$$

  - Training vectors: $x_i$, $i = 1, \ldots, l$

    Define an indicator/labeling vector y

$$y_i = \begin{cases} 1 & \text{, if } x_i \text{ in class 2} \\ -1 & \text{, if } x_i \text{ in class 1} \end{cases}$$

  - A separating hyperplane: $w^T x + b = 0$

$$\begin{cases} (w^T x_i) + b > 0 & \text{if } y_i = 1 \\ (w^T x_i) + b < 0 & \text{if } y_i = -1 \end{cases}$$

**Brief Conclusion:**

w: Rotation (+ scaling) of decision line

b: Translation of decision line

# 1.2 Classification Problem: Maximal Margin

- Maximal Margin
  - Width of the margin between $w^T x + b = 1$ and -1:

Margin 2d'=? $2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T\mathbf{w}} \longrightarrow \max$

  - The decision boundary *L* should classify all data correctly.

$$\Rightarrow y_i(\mathbf{w}^T x_i + b) \geq 1$$

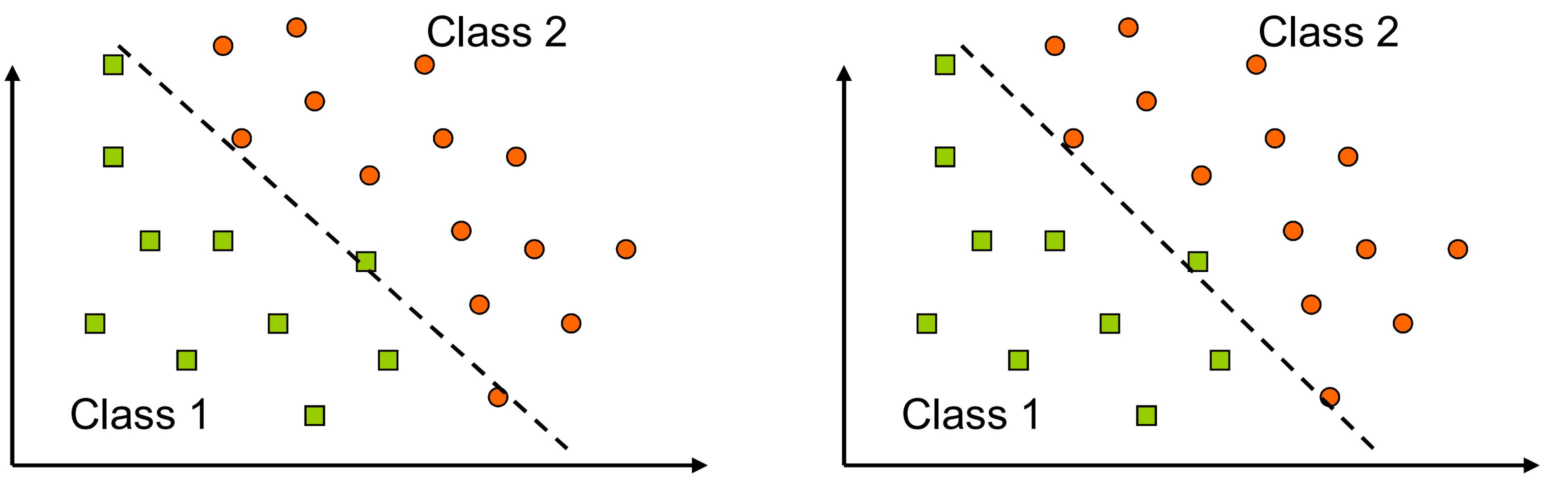
  - The first SVM formula: a constrained optimization problem [Boser et al., 1992]

$$\max 2/\|\mathbf{w}\| = 2/\sqrt{\mathbf{w}^T\mathbf{w}}$$
$$\Rightarrow \max 4/\mathbf{w}^T\mathbf{w}$$
$$\Rightarrow \min \mathbf{w}^T\mathbf{w}$$

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

# 1.3 Nonlinearly Separable Data

- Nonlinearly Separable Data
  - No linear plane could separate data perfectly




  - Solution

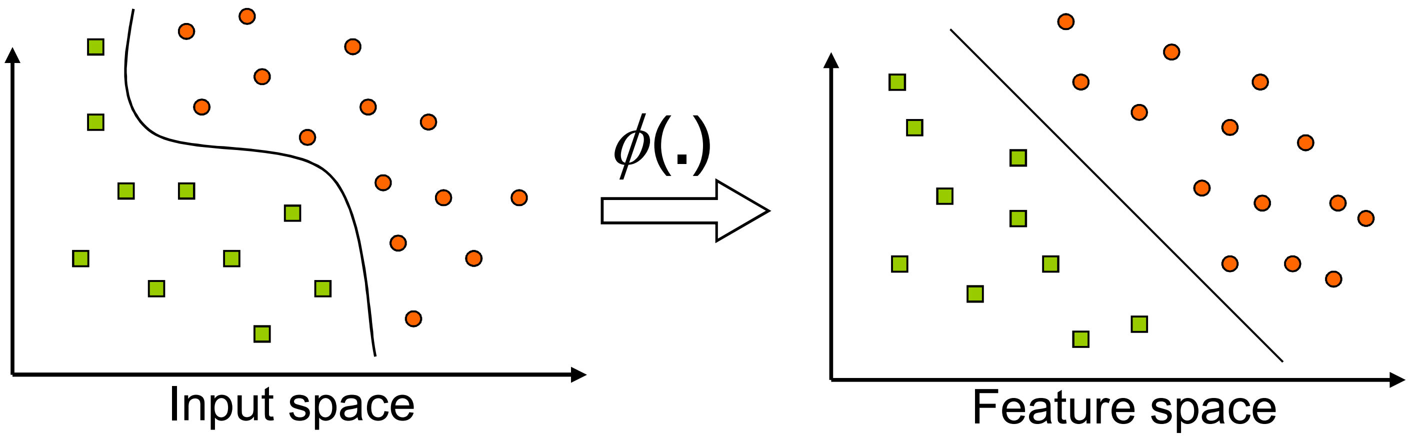
1) Nonlinear transformation => (james:)

1.1) To higher dimension $\phi(.)$ via kernel function and then linear separation

1.2) Or transfer to, for example, inside circle and outside circle. That is, decision boundary is nonlinear??

2) Soft margin => allow training error, i.e, ERR != 0, occurs

# 1.3 Nonlinearly Separable Data: Nonlinear Transformation (1/2)




- Why nonlinear transform $\phi(.)$?
  1) Data are more easily separated in higher dimensional (maybe infinite) feature space .
  2) Linear operation in the feature space is equivalent to nonlinear operation in input space.
- Ex:

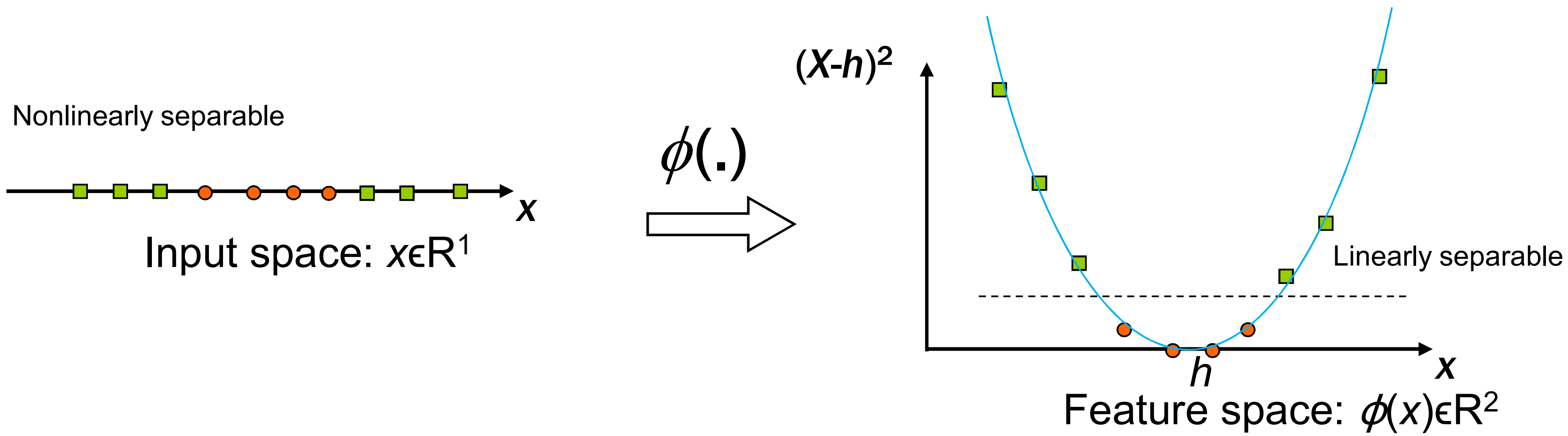
$$\mathbf{x} \in \mathbf{R}^3 \quad \phi(\mathbf{x}) \in \mathbf{R}^{10}$$

$$\phi(\mathbf{x}) = (1, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_3, x_1^2, x_2^2, x_3^2, \sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \sqrt{2}x_2x_3)$$

$(x_1,x_2,x_3)$

# 1.3 Nonlinearly Separable Data: Nonlinear Transformation (2/2)

- Example




  - The circle points spread between the square points.
  - After the nonlinear mapping, there explicitly exists a linear plane separating circle points from square points.

# 1.3 Nonlinearly Separable Data: Kernel Function

- Kernel Function
  - The relationship between the kernel function *K* and the mapping $\phi(.)$ is

$$K(x,y) = \langle \phi(x), \phi(y) \rangle = \phi(x)^T \phi(y)$$

  - In practice, we specify *K* instead of choosing $\phi(.)$ directly.
  - Intuitively, *K*(x,y) represents the similarity of $\phi(x)$ and $\phi(y)$ as we desired.
  - *K*(x,y) needs to satisfy ***Mercer's Condition*** (described later) to make sure that $\phi(.)$ exists.
- SVM solves two issues simultaneously
  - Nonlinear transformation using kernels
  - Minimize $\|w\|$

# 1.3 Nonlinearly Separable Data: Typical Kernel Function

- Typical Kernel Function
  1) Polynomial kernel

     $K(\mathrm{x},\mathrm{y}) = (\mathrm{x}^T\mathrm{y}+1)^d$, *d:* degree

  2) Radial basis function (Gaussian kernel)

     $$K(\mathrm{x},\mathrm{y}) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right),\ \sigma : \text{width}$$

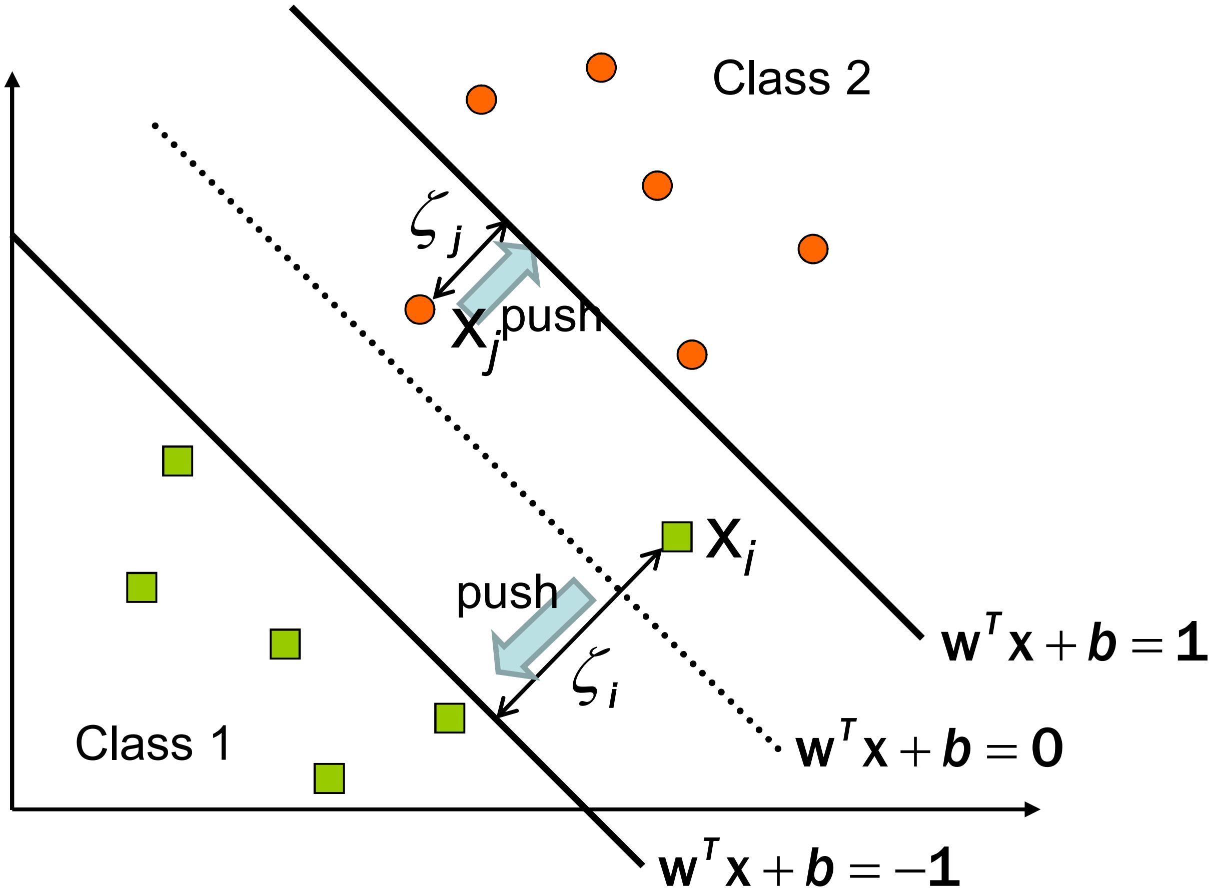
  3) Sigmoid function

     $K(\mathrm{x},\mathrm{y}) = \tanh(\kappa \mathrm{x}^T\mathrm{y}+\theta)$, $\kappa, \theta$ : parameters

  – The choice of different kernel functions is **problem-dependent**.

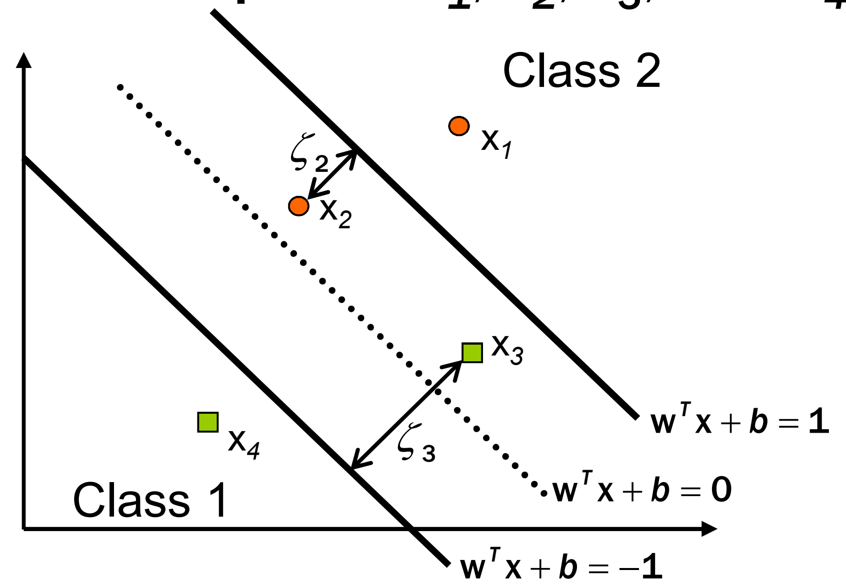
# 1.3 Nonlinearly Separable Data: Soft Margin Hyperplane (1/3)

- Soft Margin Hyperplane
  - To avoid overfitting, we allow "training errors" $\zeta_i$ in classification



# 1.3 Nonlinearly Separable Data: Soft Margin Hyperplane (2/3)

- The Inequality Constraint $y_i(w^T x_i + b) \geq 1 - \zeta_i$
  - Consider four points $x_1$, $x_2$, $x_3$, and $x_4$




$x_1$: $y_1 = 1$, $(w^T x_1 + b) > 1$ (ex: 1.5), so $y_1(w^T x_1 + b) > 1$ (ex: 1.5), and then $\zeta_1 = 0$ (no error)
satisfy constraint

$x_2$: $y_2 = 1$, $1 > (w^T x_2 + b) > 0$ (ex: 0.5), so $1 > y_2(w^T x_2 + b) > 0$ (ex: 0.5), and then $1 > \zeta_2 > 0$ (ex: 0.5)

$x_3$: $y_3 = -1$, $1 > (w^T x_3 + b) > 0$ (ex: 0.5), so $0 > y_3(w^T x_3 + b) > -1$ (ex: -0.5), and then $2 > \zeta_3 > 1$ (ex: 1.5)

$x_4$: $y_4 = -1$, $(w^T x_4 + b) < -1$ (ex: -1.5), so $y_4(w^T x_4 + b) > 1$ (ex: 1.5), and then $\zeta_4 = 0$ (no error)
satisfy constraint

# 1.3 Nonlinearly Separable Data: Soft Margin Hyperplane (3/3)

- Soft Margin Optimization Problem
  - Include an additional term of training errors $\sum_{i=1}^{l} \zeta_i$
  - Combine with margin term by multiplying a scalar $C$
  - Reformulate:

$$\min_{w,b,\zeta} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \zeta_i$$

$$\text{subject to } y_i (w^T x_i + b) \geq 1 - \zeta_i, \zeta_i \geq 0, \text{ for } i = 1, \cdots, l$$

# 1.4 Standard SVM ??

w: Line Rotation + Scaling
=> Line Rotation

- Standard SVM [Vapnik and Cortes, 1995]
  - Key Idea
    1) Higher dimensional feature space
    2) Allow training errors
  - Constrained Optimization Problem

**Margin Term:** Find the best rotation factor by margin w

**Training Error Term:**
Min Sum of $\zeta_i$ : Min sum of all training errors

C: Bigger C, margin???
If linear margin, the total training errors are big. If nonlinear margin such as curve, then the total training errors will reduce.

J + Carl: Cause nonlinear margin occur

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \zeta_i$$

subject to $y_i(w^T \phi(x_i) + b) \geq 1 - \zeta_i, \ \zeta_i \geq 0, i = 1, ..., l$

*C*: Tradeoff parameters between training error and margin *w* (need to tune)

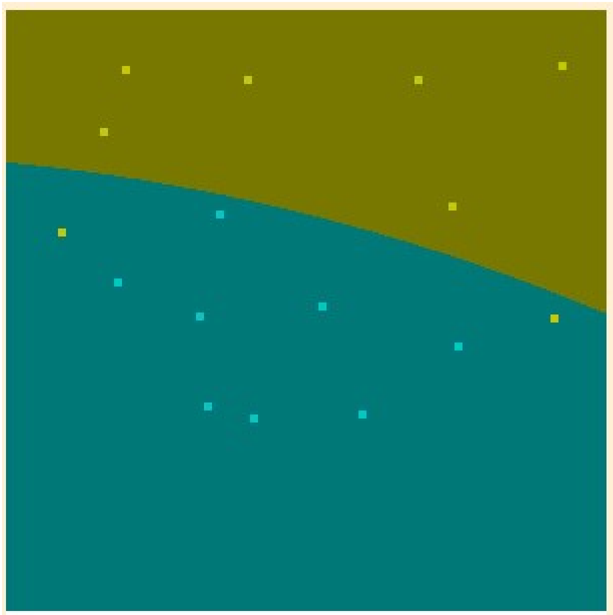
  - w: a vector in high dimensional space

$\Rightarrow$ maybe infinite variables. Difficult to solve

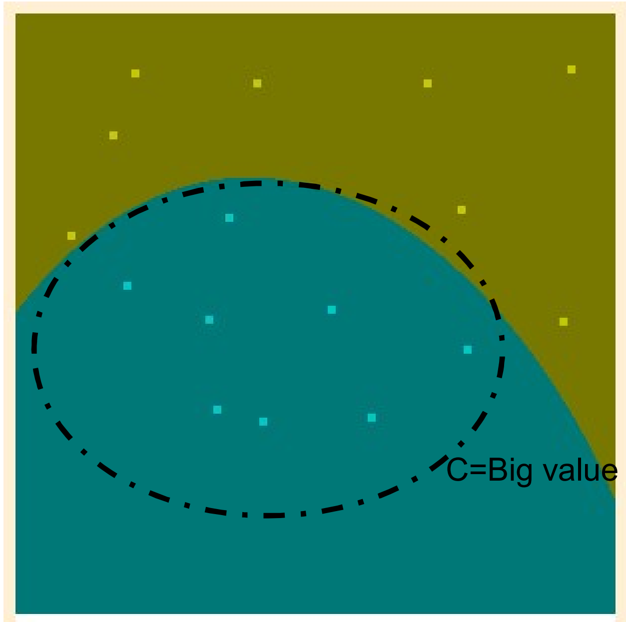
# 1.4 Standard SVM: Effect of Tradeoff Parameter

- Effect of Tradeoff Parameter
  - The effect of C can be observed using the SVM Toy on the libsvm webpage:
  - http://www.csie.ntu.edu.tw/~cjlin/libsvm/

C=1



C=100



# 1.4 Standard SVM: Duality

$$L(\mathrm{w}, b, \alpha) = \frac{1}{2}\mathrm{w}^{T}\mathrm{w} - \sum_{i=1}^{l} \alpha_i \left[ y_i(\mathrm{w}^{T}x_i + b) - 1 \right]$$

$$\frac{\partial L}{\partial \mathrm{w}} = \mathrm{w} - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \;\Rightarrow\; \mathrm{w} = \sum_{i=1}^{l} \alpha_i y_i x_i, \;\; \alpha_i \geq 0$$

- Duality
  - Transform the primal problem to the dual problem

$$\min L(w, b, \alpha) = \max_{\alpha \geq 0}(\min_{w,b} L(w, b, \alpha))$$

- The Dual Problem

A finite problem: # variables = #training data

$$\boxed{\min_{\alpha} \; \frac{1}{2}\alpha^T Q \alpha - e^T \alpha}$$

$$L(\mathrm{w}, b, \alpha) \Leftarrow (\mathrm{w} = \sum_{i=1}^{l} \alpha_i y_i x_i)$$

subject to $0 \leq \alpha_i \leq C, \; i = 1, \ldots, l, \;\; y^T \alpha = 0$

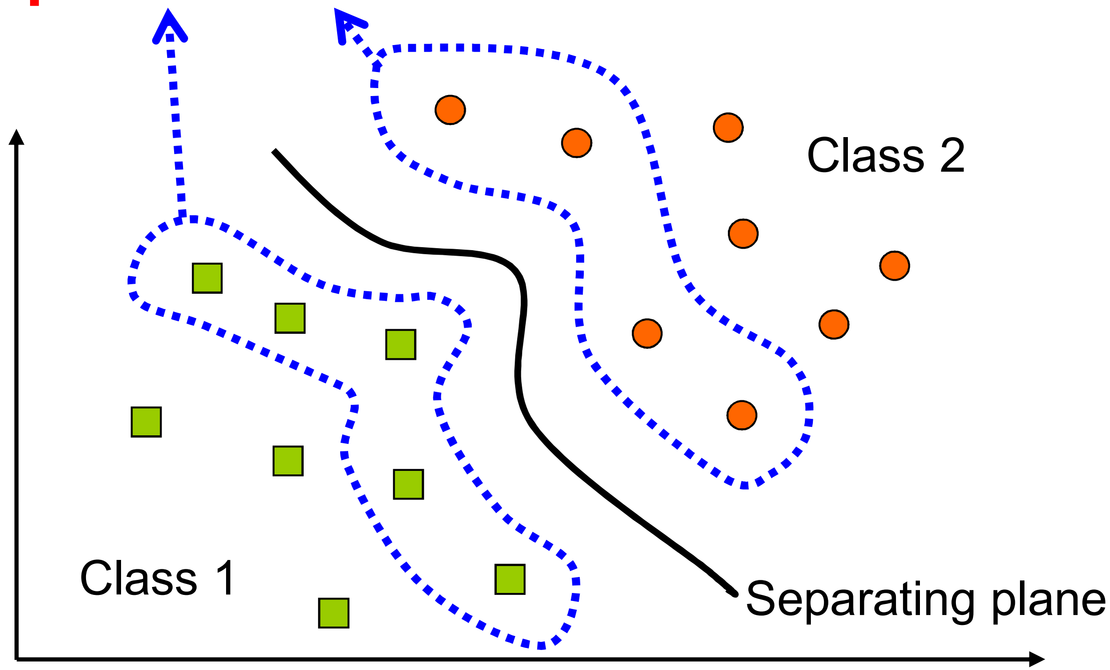
where $Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j)$ and e = $[1, \ldots, 1]^T$

  - At optimum, w is recovered as $\mathrm{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(x_i)$
  - The only difference with the linearly separable case is the upper bound C on $\alpha_i$
  - A quadratic programming solver can be applied to find $\alpha_i$

# 1.4 Standard SVM: Support Vectors

- Support Vectors
  - The support vectors are a subset of training data **closest to the separating plane** and therefore the most difficult to classify.

**Support Vectors**



# 1.4 Standard SVM: Decision Function

- Decision Function
  - At optimum, w is recovered as $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \phi(\mathbf{x}_i)$
  - Let $\phi(\mathbf{x})$ be the testing data, then decision function

$$
\begin{aligned}
& \mathbf{w}^T \phi(\mathbf{x}) + b \\
&= \sum_{i=1}^{l} \alpha_i y_i \underline{\phi(\mathbf{x}_i)^T \phi(\mathbf{x})} + b \\
&= \sum_{i=1}^{l} \alpha_i y_i \underline{K(\mathbf{x}_i, \mathbf{x})} + b
\end{aligned}
$$

  - By using the dual variable $\alpha_i$, it is no need to write down w.
  - Very often, $\alpha_i$ is optimized to zero. In other words, $x_i$ with non-zero $\alpha_i$ are so-called support vectors which determine the decision boundary.

# 1.4 Standard SVM: Mercer's Condition

- Mercer's Condition [1903]
  - What kind of $K_{ij}$ can be represented as $\phi(x_i)^T \phi(x_j)$ ?
  - $K(x, y) = \phi(x)^T \phi(y)$ if and only if $\forall g$ s.t.

$$\int g(x)^2 dx \text{ finite} \Rightarrow \int K(x, y) g(x) g(y) dx dy \geq 0$$

  - It is useful for some kernel. However, still not easy to check.

# 2. Dual SVM Derivation

- Duality
  - Transform the primal problem to the dual problem

$$\min L(w,b,\alpha) = \max_{\alpha \geq 0}(\min_{w,b} L(w,b,\alpha))$$

Lagrange:

$$L(\mathrm{w},b,\alpha) = \frac{1}{2}\mathrm{w}^T\mathrm{w} - \sum_{i=1}^{l}\alpha_i\left[y_i(\mathrm{w}^T x_i + b) - 1\right]$$

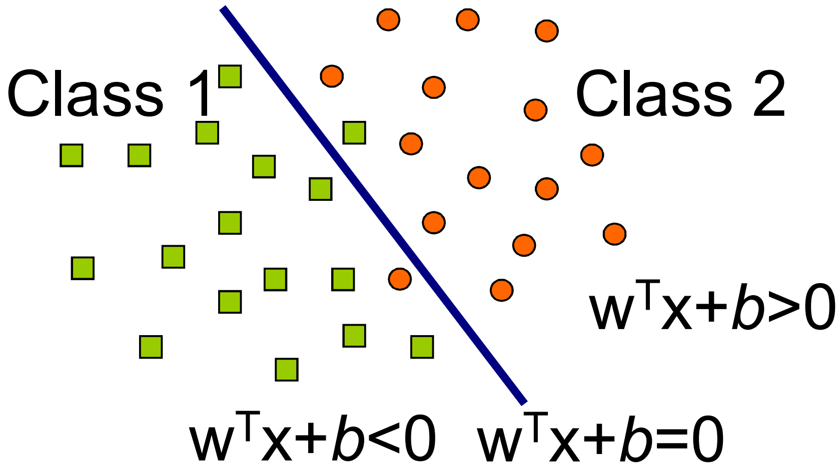
Derivation of w and b

Max(Lamda):

$$\tilde{L}(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

# 2.1 SVMs Reminder

- Original SVM Problem




- Consider the problem without $\zeta_i$ and $C$

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

subject to $y_i(w^T x_i + b) \geq 1$ , $i$ = 1, ..., $l$

- A constrained optimization problem: Use lagrange multiplier method to solve

# 2.2 Lagrange Multiplier Method

- Constrained Optimization Problem
  - Find x=[$x_1$ $x_2$ … $x_n$] which minimizes f(x) subject to the inequality constraints: $g_j(x) \leqq 0$, j=1, 2, …, *m.*

- Lagrange Function
  - Transform the inequality constraints to equality constraints by using $G_j(x,y)=g_j(x)+y_j^2=0$, where y=[$y_1$ $y_2$ … $y_m$] is the vector of slack variables.
  - Lagrange function: $L(x,y,\lambda)=f(x)+\Sigma_j\lambda_jG_j(x,y)$

# 2.2 Lagrange Multiplier Method

- Lagrange Function
  - The solution of L is given by solving

$$\begin{cases} \dfrac{\partial L}{\partial x_i} = \dfrac{\partial f}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \dfrac{\partial g_j}{\partial x_i} = 0, \ i = 1,2,\ldots,n \\ \dfrac{\partial L}{\partial \lambda_j} = g_j(x) + y_j^2 = 0, \ \ j = 1,2,\ldots,m \\ \dfrac{\partial L}{\partial y_j} = 2\lambda_j y_j = 0, \ \ j = 1,2,\ldots,m \end{cases}$$

# 2.3 The Linearly Separable Case (1/2)

- The Primal Problem

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

subject to $y_i(\mathbf{w}^T x_i + b) \geq 1$, $i = 1, \ldots, l$

- The Lagrange function

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{l} \alpha_i \left[ y_i(\mathbf{w}^T x_i + b) - 1 \right]$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \ \Rightarrow \ \mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i x_i, \ \ \alpha_i \geq 0$$

where $\alpha_i$ is the weight of data point $x_i$.

# 2.3 The Linearly Separable Case (2/2)

- Notice the value of $\alpha_i$:
  - $\alpha_i = 0$, don't care about the constraints!
  - $\alpha_i > 0$, the *i*-th point $x_i$ is close to the hyperplane.

    Support vector

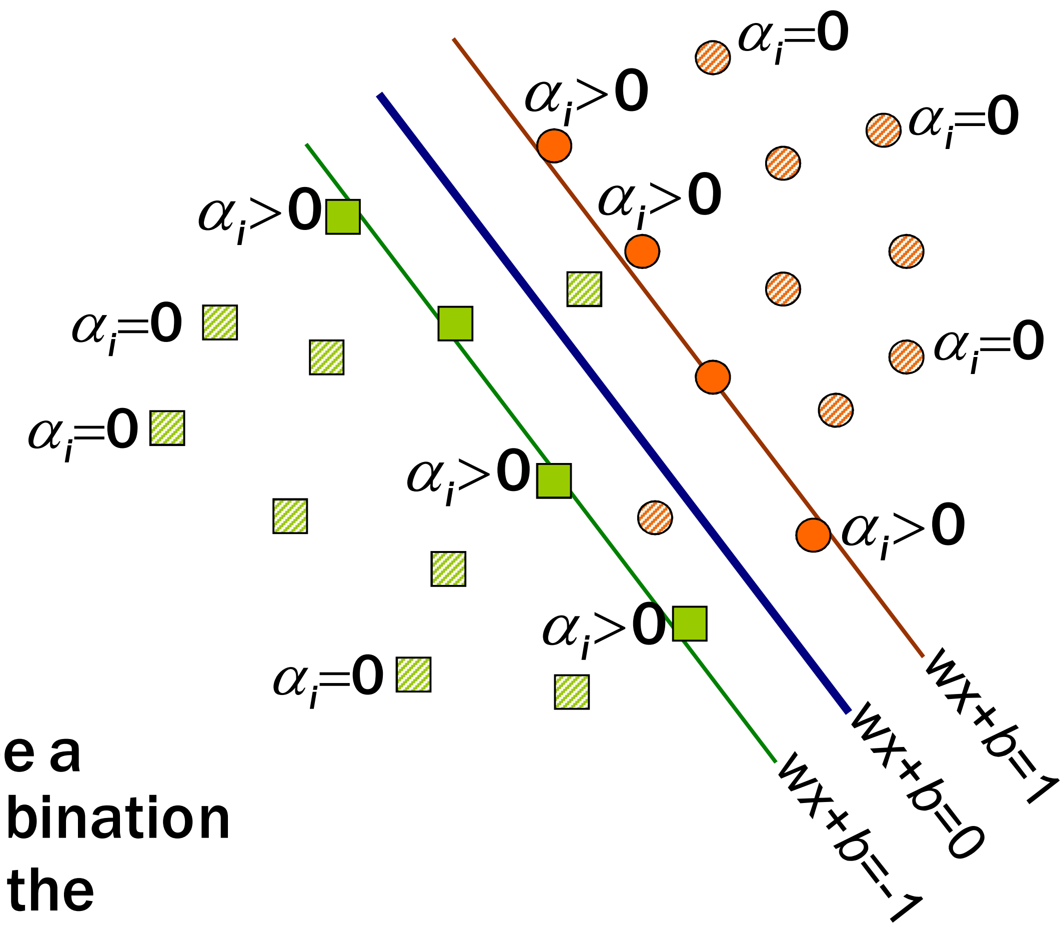
    At optimum, $\frac{\partial L}{\partial \alpha_i} = y_i(w^T x_i + b) - 1 = 0$

    $$y_i(w^T x_i + b) = 1 \;\Rightarrow\; b = \frac{1}{y_i} - w^T x_i$$

    notice $y_i = 1 / y_i, \;\; y_i = \{-1, +1\}$
- Therefore, we can obtain *b* by $b = y_i - w^T x_i$, for any *i* where $\alpha_i > 0$.

  (Average *b* over all points where $\alpha_i > 0$)

# 2.3 The Linearly Separable Case: Dual SVM Interpretation




$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i,$$

**w is going to be a weighted combination of points near the hyperplane.**

# 2.3 The Linearly Separable Case: Dual Problem (1/2)

- Dual Problem
  - Substitute $\mathbf{w}=\sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i$ into $L(\mathbf{w},b,\alpha)$ to get $\tilde{L}(\alpha)$

$$\tilde{L}(\alpha)$$

$$=\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i\Big)^T\Big(\sum_{j=1}^{l}\alpha_j y_j \mathbf{x}_j\Big)-\sum_{i=1}^{l}\alpha_i\left\{y_i\Big[\Big(\sum_{j=1}^{l}\alpha_j y_j \mathbf{x}_j\Big)^T\mathbf{x}_i+b\Big]-1\right\}$$

$$=\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i \mathbf{x}_i^T\alpha_j y_j \mathbf{x}_j-\sum_{i=1}^{l}\alpha_i y_i\Big(\sum_{j=1}^{l}\alpha_j y_j \mathbf{x}_j\Big)^T\mathbf{x}_i \qquad \text{hint}: \sum_{i=1}^{l}\alpha_i y_i=0$$

$$=\underline{\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j-\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i\alpha_j y_j \mathbf{x}_j^T\mathbf{x}_i}-\boxed{\sum_{i=1}^{l}\alpha_i y_i b}+\sum_{i=1}^{l}\alpha_i$$

$$=\sum_{i=1}^{l}\alpha_i-\underline{\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j}$$

# 2.3 The Linearly Separable Case: Dual Problem (2/2)

- Dual Problem
  - Reformulate $\tilde{L}(\alpha)$ to a quadratic programming problem

$$\min_{\alpha} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

subject to $\alpha_i \geq 0$ for i=1,...,l and $y^T\alpha = 0$

where $Q \in R^{l\times l}$, $Q_{ij} = y_i y_j x_i^T x_j$, $e = [1 \quad \cdots \quad 1]^T \in R^{l\times 1}$, $\alpha = [\alpha_1 \quad \ldots \quad \alpha_l]^T \in R^{l\times 1}$, and $y = [y_1 \quad \ldots \quad y_l]^T \in R^{l\times 1}$

  - We can apply quadratic programming solver to find $\alpha$

$$k(x_i, x_j) = x_i^T x_j$$

  - Use kernel tricks to find decision function…Substitute support vectors to get b, without knowing w

# 2.3 The Linearly Separable Case: Dual SVM Formulation

- Lagrange function has to be

$$L(\mathrm{w}, b, \alpha) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} - \sum_{i=1}^{l} \alpha_i \left[ y_i (\mathrm{w}^T x_i + b) - 1 \right]$$

- Dual problem is given by

$$\min_{\alpha \geq 0} \mathrm{Primal} = \max_{\alpha \geq 0} \min_{\mathrm{w}, b} L(\mathrm{w}, b, \alpha)$$

- Solution is given by

$$\alpha = \arg\min_{\alpha} \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i^T x_j, \sum_{i=1}^{l} \alpha_i y_i = 0, \alpha_i \geq 0$$

- Thus w and *b* can be obtained by

$$\mathrm{w} = \sum_{i=1}^{l} \alpha_i y_i x_i$$

$$b = y_i - \mathrm{w}^{\mathrm{T}} x_i, \text{ for any } k \text{ where } \alpha_k \geq 0$$

# Q: Why should $\Sigma_i \alpha_i y_i = 0$?

# A:

- If $\sum_{i=1}^{l} \alpha_i y_i \neq 0$, move $b$ to ∞, then $-b\sum_{i=1}^{l} \alpha_i y_i$ will be -∞. That is, $L(\mathrm{w}, b, \alpha)$ decreases to -∞.

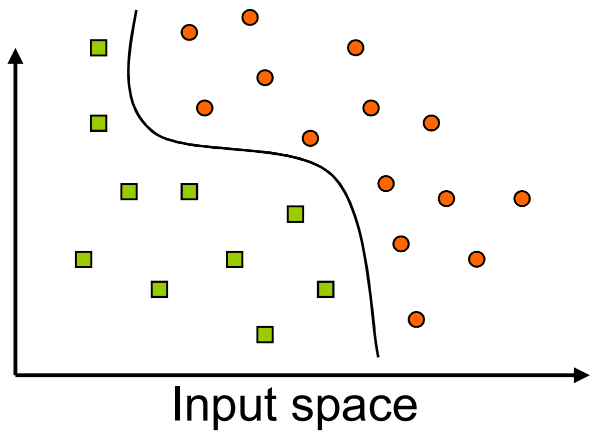
- $\min_{\mathrm{w},b} L(\mathrm{w}, b, \alpha) =$

$$\begin{cases} -\infty & \text{if } \sum_{i=1}^{l} \alpha_i y_i \neq 0 \\ \min_{\mathrm{w}} \frac{1}{2}\mathrm{w}^T\mathrm{w} - \sum_{i=1}^{l} \alpha_i \left[ y_i \mathrm{w}^T x_i - 1 \right] & \text{if } \sum_{i=1}^{l} \alpha_i y_i = 0 \end{cases}$$

- Hence, we have w only when $\sum_{i=1}^{l} \alpha_i y_i = 0$.

# 2.4 The Nonlinearly Separable Case

- Nonlinearly Separable Data
  - More Often than not, the data could not separated by a linear plane



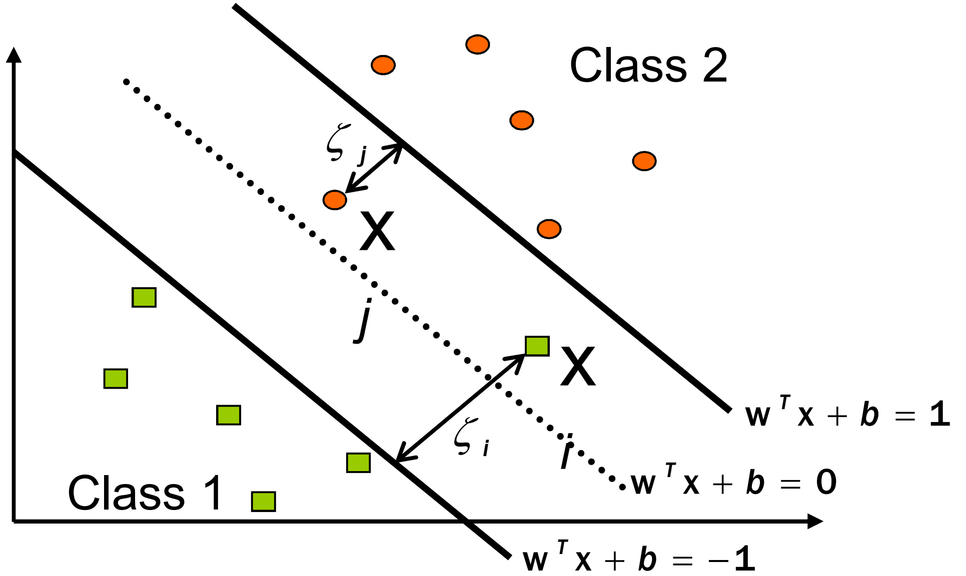

  - Solution for nonlinearly separable case

    1) Soft Margin

    2) Nonlinear Mapping to High dimensional space

# 2.4.1 Soft Margin (1/2)

- Soft Margin
  - Allow training error $\zeta_i$




- Primal problem:

$$\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \zeta_i$$

subject to $y_i(w^T x_i + b) \geq 1 - \zeta_i, \ \zeta_i \geq 0$

# 2.4.1 Soft Margin (2/2)

- **Lagrange function:**

$$L(\mathrm{w}, b, \alpha, \zeta_i) = \frac{1}{2}\mathrm{w}^{\mathrm{T}}\mathrm{w} + C\sum_{i=1}^{l}\zeta_i - \sum_{i=1}^{l}\alpha_i\left[y_i(\mathrm{w}^T \mathbf{x}_i + b) - 1\right] + \sum_{i=1}^{l}\mu_i(\zeta_i - 0)$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i = 0 \;\Rightarrow\; \mathbf{w} = \sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i, \;\; \alpha_i \geq 0$$

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

$$\frac{\partial L}{\partial \zeta_i} = C - \alpha_i + \mu_i = 0 \Rightarrow \alpha_i = C - \mu_i$$

# 2.4.1 Soft Margin: Dual Problem (1/2)

- **Dual Problem:**

$$
\begin{aligned}
&\tilde{L}(\alpha)\\
&=\frac{1}{2}\Big(\sum_{i=1}^{l}\alpha_i y_i x_i\Big)^T\Big(\sum_{j=1}^{l}\alpha_j y_j x_j\Big)+C\sum_{i=1}^{l}\zeta_i-\sum_{i=1}^{l}\alpha_i\left\{y_i\Big[\Big(\sum_{j=1}^{l}\alpha_j y_j x_j\Big)^T x_i+b\Big]-1+\zeta_i\right\}-\sum_{i=1}^{l}\mu_i\zeta_i\\
&=\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i^T x_j+\sum_{i=1}^{l}C\zeta_i-\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i y_i\alpha_j y_j x_j^T x_i\boxed{-\sum_{i=1}^{l}\alpha_i y_i b}+\sum_{i=1}^{l}\alpha_i-\sum_{i=1}^{l}\alpha_i\zeta_i-\sum_{i=1}^{l}\mu_i\zeta_i\\
&=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i^T x_j+\boxed{\sum_{i=1}^{l}(C-\mu_i)}\zeta_i-\sum_{i=1}^{l}\alpha_i\zeta_i\\
&=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i^T x_j+\boxed{\sum_{i=1}^{l}(\alpha_i)}\zeta_i-\sum_{i=1}^{l}\alpha_i\zeta_i\\
&=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i^T x_j
\end{aligned}
$$

hint : $\sum_{i=1}^{l}\alpha_i y_i = 0$

hint : $\alpha_i = C-\mu_i$

# 2.4.1 Soft Margin: Dual Problem (2/2)

- Dual Problem
  - $\alpha_i = C - \mu_i$ with $\mu_i \geq 0$ implies $C \geq \alpha_i$
  - Combine $C \geq \alpha_i$ and $\alpha_i \geq 0 \Rightarrow C \geq \alpha_i \geq 0$
  - Reformulate $\tilde{L}(\alpha)$ to a quadratic programming problem

$$\min_{\alpha} \frac{1}{2}\alpha^T Q \alpha - e^T \alpha$$

subject to $C \geq \alpha_i \geq 0$ for i=1,…,l and $y^T \alpha = 0$

where $Q \in R^{l \times l}$, $Q_{ij} = y_i y_j x_i^T x_j$, $e = [1 \quad \cdots \quad 1]^T \in R^{l \times 1}$

, $\alpha = [\alpha_1 \quad \ldots \quad \alpha_l]^T \in R^{l \times 1}$, and $y = [y_1 \quad \ldots \quad y_l]^T \in R^{l \times 1}$

  - We can apply quadratic programming solver to find $\alpha$

# 2.4.2 Nonlinear Mapping (1/2)

- Project data onto high dimensional space $\phi(x_i)$
  - The hard margin case

$$\min_{\alpha} \frac{1}{2}\alpha^T Q \alpha - e^T \alpha, \text{ subject to } \boxed{\alpha_i \geq 0} \text{ for } i = 1, \cdots, l \text{ and } y^T \alpha = 0$$

$$\text{where } Q \in R^{l \times l}, Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j), e = [1 \quad \cdots \quad 1]^T \in R^{l \times 1}$$

$$, \alpha = [\alpha_1 \quad \cdots \quad \alpha_l]^T \in R^{l \times 1} \text{ and } y = [y_1 \quad \cdots \quad y_l]^T \in R^{l \times 1}$$

  - The soft margin case

$$\min_{\alpha} \frac{1}{2}\alpha^T Q \alpha - e^T \alpha, \text{ subject to } \boxed{C \geq \alpha_i \geq 0} \text{ for } i = 1, \cdots, l \text{ and } y^T \alpha = 0$$

$$\text{where } Q \in R^{l \times l}, Q_{ij} = y_i y_j \phi(x_i)^T \phi(x_j), e = [1 \quad \cdots \quad 1]^T \in R^{l \times 1}$$

$$, \alpha = [\alpha_1 \quad \cdots \quad \alpha_l]^T \in R^{l \times 1} \text{ and } y = [y_1 \quad \cdots \quad y_l]^T \in R^{l \times 1}$$

# 2.4.2 Nonlinear Mapping (2/2)

- Kernel Trick
  - Because the dimension of $\phi(\mathrm{x}_i)$ may be infinity, we have problem on calculating the inner product of two points in the high dimensional space.
  - We define $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ as kernel function.
  - Use the kernel function (ex: $K(\mathrm{x}, \mathrm{y}) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$ ), we can get the inner product value directly without computing the mapping $\phi(\mathrm{x}_i)$.
- The decision function would be reformulated:

$$\mathrm{w}^T \phi(\mathrm{x}) + b = \sum_{i=1}^{l} \alpha_i \mathrm{y}_i \underline{\phi(\mathrm{x}_i)^T \phi(\mathrm{x})} + b = \sum_{i=1}^{l} \alpha_i \mathrm{y}_i \underline{K(\mathrm{x}_i, \mathrm{x})} + b$$

# 3. Training Linear and Nonlinear SVMs

- Training Nonlinear SVMs Technique
  - Save storage
  - Speedup
    1) Caching
    2) Shrinking

- Training Linear SVMs Technique
  - Decomposition
  - Approximation

# 3.1 Training Nonlinear SVM

- Training Nonlinear SVM
  - The dual

$$\min_{\alpha} \quad \frac{1}{2}\alpha^{\mathsf{T}} \mathbf{Q} \alpha - \mathbf{e}^{\mathsf{T}} \alpha$$

subject to $0 \le \alpha_i \le C,\ i = 1, \ldots, l,\ y^{\mathsf{T}} \alpha = 0$

where $Q_{ij} = y_i y_j \phi(x_i)^{\mathsf{T}} \phi(x_j)$ and $\mathbf{e} = [1, \ldots, 1]^{\mathsf{T}}$

  - $Q_{ij} \neq 0$, Q: an $l$ by $l$ symmetric and fully dense matrix.

    In practice, 30,000 training data: 30,000 variables

    $\Rightarrow$ size(Q) = $30{,}000^2 * 8/2$ = 3GB, cause storage problem!
  - Traditional methods such as Newton and Quasi-Newton are hard to be applied.

# 3.1 Training Nonlinear SVM: Decomposition Method

- Decomposition Method
  - B: selected working set, N: the remaining set
  - $B^k$*:* B in *k*-th iteration
  - Sub-problem in each iteration

$$\min_{\alpha_B} \frac{1}{2}\begin{bmatrix}\alpha_B^{\mathrm{T}} & (\alpha_N^k)^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}Q_{BB} & Q_{BN} \\ Q_{NB} & Q_{NN}\end{bmatrix}\begin{bmatrix}\alpha_B \\ \alpha_N^k\end{bmatrix}$$

$$-\begin{bmatrix}\mathbf{e}_B^{\mathrm{T}} & (\mathbf{e}_N^k)^{\mathrm{T}}\end{bmatrix}\begin{bmatrix}\alpha_B \\ \alpha_N^k\end{bmatrix}$$

subject to $0 \le \alpha_t \le C,\ \ t \in B,\ \ y_B^{\mathrm{T}}\alpha_B = -y_N^{\mathrm{T}}\alpha_N^k$

where $\alpha_B$ is the only variable related to B.

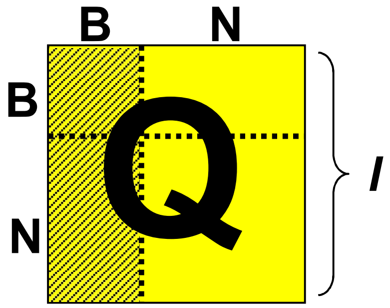
# 3.1.1 Avoid Storage Problem (1/3)

- Avoid Storage Problem
  - Consider min only with respect to $\alpha_B$

    $\Rightarrow$ Remove several terms related to $\alpha_N$
  - The new objective function

    $$\frac{1}{2}\alpha_B^T Q_{BB}\alpha_B + (-e_B + Q_{BN}\alpha_N^k)^T\alpha_B + \text{const}$$

    the part out of working set is regarded as constant.




  - To avoid the storage problem, *B* columns of *Q* are stored only when needed

# 3.1.1 Avoid Storage Problem (2/3)

- How Does It Work?
  - It converges slowly compared to some optimization methods, e.g. Newton and Quasi-Newton.
  - The decision function

$$\operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b\right)$$

  - It is no need to obtain accurate $\alpha$

    $\Rightarrow$ It is also no need to apply many iterations.
  - If #support vectors << #training data, training will be fast.
  - $\alpha$ is usually initialized to be 0.

# 3.1.1 Avoid Storage Problem (3/3)

- Example
  - An example of training 50,000 data using LIBSVM on a Pentium M 1.4G laptop.
  - Converge in 5m1.456s, while calculating Q may have taken more than 5 minutes.
  - #SVs = 3,370 << 50,000 = #training data
  - We can observe that it is a good case where many remain zero all the time.

# 3.1.2 Speedup Decomposition (1/3)

- Speedup Decomposition
  - Caching [Joachims, 1998]

    Store recently used kernel columns as the real computer cache.

  - Ex. (in LIBSVM)

    100K cache: 11.463s

    40M cache: 7.817s
  - Note that SVM is a quadratic optimization problem, so the size of cache is not proportional to the converging time.

# 3.1.2 Speedup Decomposition (2/3)

- Speedup Decomposition
  - Shrinking [Joachims, 1998]

    Some bounded elements do not change anymore until the end. Thus we can heuristically resize it to a smaller problem by removing these elements.
  - After certain iterations, most bounded elements are identified and do not change anymore. [Lin, 2002]

- Caching and shrinking are useful.

# 3.1.2 Speedup Decomposition (3/3)

- Caching: Issues
  - Goal: minimize the total number of calculating columns among *k* iterations
  - A simple way:

    Store recently used columns
  - A better usage of cache:

    Deliberately select those in cache
  - Idea: The columns in cache have been calculated, so it is no need to spend more effort to calculate new kernel columns.

# 3.2 Training Linear SVMs

- Training Linear SVMs
  - Linear kernel:

$$\min_{\mathbf{w},b,\zeta} \ \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\zeta_i$$

subject to $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \zeta_i, \ \ \zeta_i \geq 0$

  - An optimum

$$\zeta_i = \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

# 3.2 Training Linear SVMs

- Training Linear SVMs
  - Remaining variable: w, *b*

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b))$$

  - The maximum term is not differentiable
  - #variables = #features + 1
  - Traditional optimization methods can be applied.
  - Although data set is large, if #features is small, it is easier to solve.
  - It is challenging if #features and #data is large.

# 3.2.1 Decomposition Methods for SVMs

- Decomposition Methods
  - Upper bounded components are related to training errors.
  - When *C* is large enough, w does not change anymore. [Keerthi and Lin, 2003]
  - Recall $\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \in R^n, \quad b \in R^1$

    $$\#(0 < \alpha_i < C) \leq n + 1$$

  - Starting from small *C*, faster convergence [Kao et al., 2004]
  - Using *C* = 1, 2, 4, 8, ...

# 3.2.2 Approximations (1/2)

- Approximations
  - Solving the dual is difficult when #data is large and using nonlinear kernels.
  - A simple and effective way: subsampling (e.g. k-NN or hierarchical settings)
  - Incremental way:

    Randomly separate data into 10 parts

    Train 1st part $\Rightarrow SV^1$, then train ($SV^1$ + 2nd part), ... until 10 parts are trained
  - Select good points, i.e. remove some unnecessary points first: k-NN
  - Goal: process smaller data set at the same time

# 3.2.2 Approximations (2/2)

- How to select B?
  - Random [Lee and Mangasarian, 2001]
  - Incremental [Keerthi et al., 2006]: starting from a small subset then add points to it in each iteration
- In machine learning, it is very often to balance between simplification and performance

# 4. Conclusion

- Conclusion
  - SVM could find a hyperplane which separate the different classes of data.
  - In the nonlinearly separable case, we can use the soft margin and/or nonlinear mapping to solve this problem.
  - Using the kernel trick, we can avoid the complex computation in high dimensional space.

- More Problem
  - How about multiclasses?

# 5. Reference (1/3)


- Chistianini and Shawe-Taylor, 2000
- Scholkopf and Smola, 2002
- www.csie.ntu.edu.tw/~cjlin/ (or lecture of Lin's talk in MLSS2006 pp.4-17)
- www.kernel-machines.org/phpbb/
- svm.cs.rhul.ac.uk/pagesnew/GPat.shtml

# 5. Reference (2/3)

- Lecture of Lin's talk in MLSS2006 pp.19-25
- Lecture of C. Guestrin, Machine Learning of CMU

# 5. Reference (3/3)


- Lecture of Lin's talk in MLSS2006 pp.28-48
- SVM Applet
  www.site.uottawa.ca/%7Egcaron/applets.htm