

INTRODUCTORY APPLIED MACHINE LEARNING

Yan-Fu Kuo

Dept. of Biomechatronics Engineering

National Taiwan University

Today:

- Support vector machine

Outline

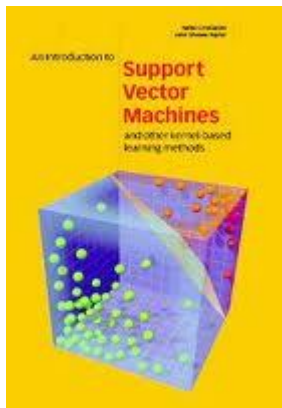
- History of SVM
- Linear SVM
- Soft margin SVM
- Kernel tricks
- SVM regression

What Is Support Vector Machine?

- Support vector machine (SVM) is a two-class classifier that maximizes the width of the margin between classes
- The margin is the empty area around the decision boundary defined by the distance to the nearest training patterns

History and Background

- “Generalized Portrait” algorithm, a special case of SVM, was introduced by Vapnik and Lerner in 1963
- SVM officially introduced with a paper at the Computational Learning Theory (COLT) conference in 1992 by Boser, Guyon and Vapnik

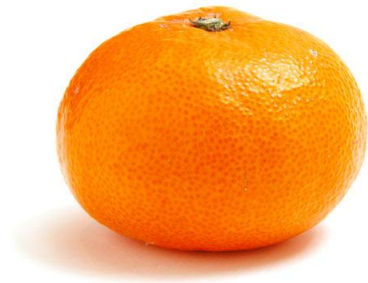


- A central website of information on kernel based methods: www.kernel-machines.org
- [*An introduction to Support Vector Machines*](#) by Cristianini and Shawe-Taylor

Classification Problem Statement

- Suppose there exists $\mathcal{X} = [\mathbf{x}_i] \in \mathbb{R}^m$, $i = 1 \dots n$, samples
- Each of the samples is associated with a class label $\mathbf{y} = [y_i] \in \mathbb{R}$
- We want to learn the mapping $\mathcal{X} \mapsto \mathbf{y}$

Example



- Suppose we have 100 photos (x_i) of oranges and bananas, i.e., $i = 1 \dots 100$
- We digitize them into 50 x 50 pixel images, i.e., $x_i \in \mathbb{R}^m$ where $m = 2500$
- Given a new photo, we want to answer the question – is it an **orange** or a **banana**? (2-class)

Classification Problem Statement (Cont'd)

- Input set: \mathcal{X} / Output set: \mathcal{Y}
- Training data points $(\mathbf{x}_1, y_1) \dots (\mathbf{x}_n, y_n)$
- Problem: given a $\mathbf{x}_i \in \mathcal{X}$, find a suitable mapping (model) such that it gives $y_i \in \mathcal{Y}$
- Simplify the case to a 2-class classification, i.e., $y_i \in \{+1, -1\}$
- The objective of the first phase is to learn a classifier: $\hat{y} = f(\mathbf{x}, \alpha)$, where α are the parameters of the function

Classification Error

- Zero-one loss function:

$$l(y, \hat{y}) = \begin{cases} 1 & \text{if } y \neq \hat{y} \\ 0 & \text{if } y = \hat{y} \end{cases}$$

- Training error:

$$E_{training}(\alpha) = \frac{1}{n} \sum_{i=1}^n l(y_i, f(\mathbf{x}_i, \alpha))$$

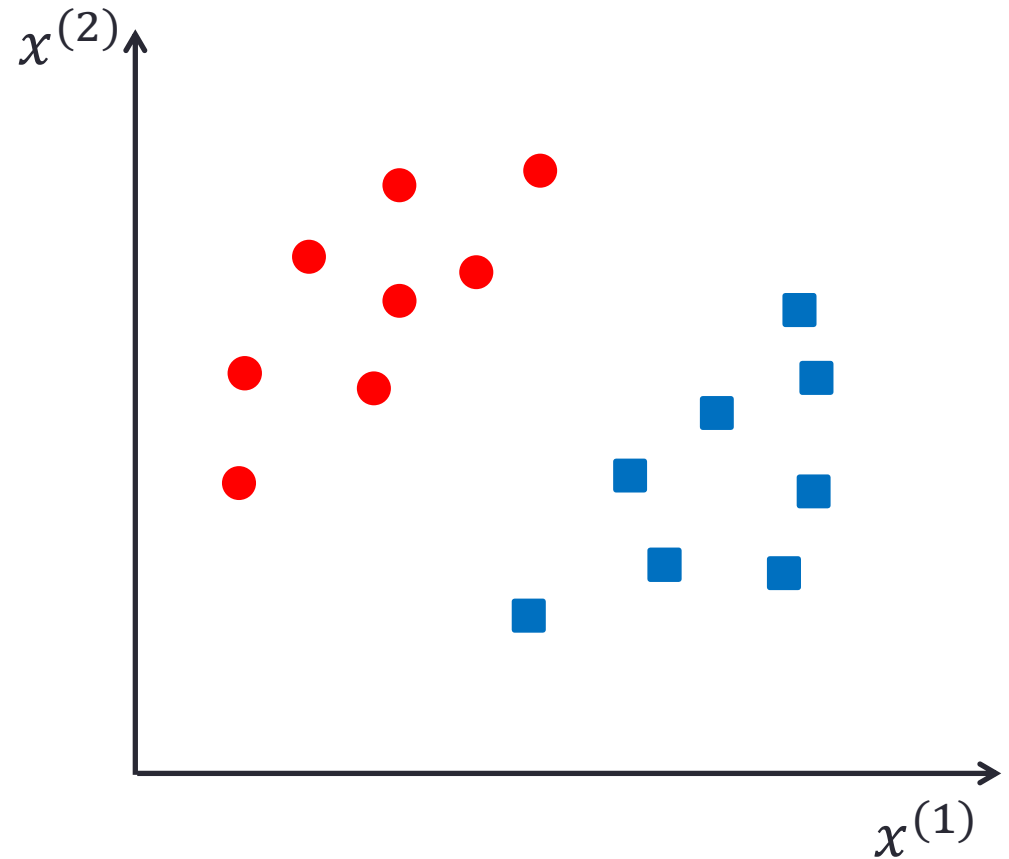
- True error:

$$E_{true}(\alpha) = \int l(y, f(\mathbf{x}, \alpha)) dP(\mathbf{x}, y),$$

where $P(\mathbf{x}, y)$ is the joint distribution function of \mathbf{x} and y

Linearly Separable Case

- Simplify the problem to 2-dimensional, i.e., $\mathbf{x}_i \in \mathbb{R}^2$



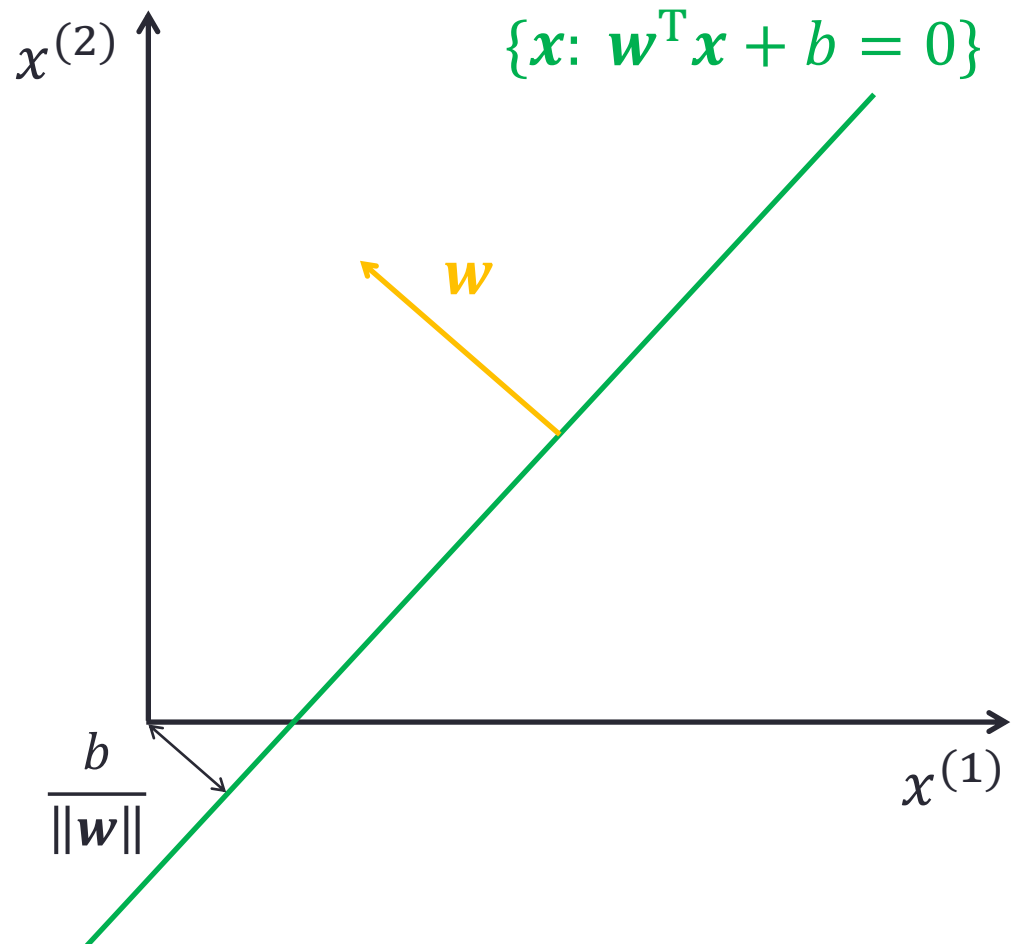
Review of High School Linear Algebra

- A hyperplane (line) can be represented as:

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

- Distance from a point \mathbf{x}_0 to this plane is:

$$\frac{\mathbf{w}^T \mathbf{x}_0 + b}{\|\mathbf{w}\|}$$

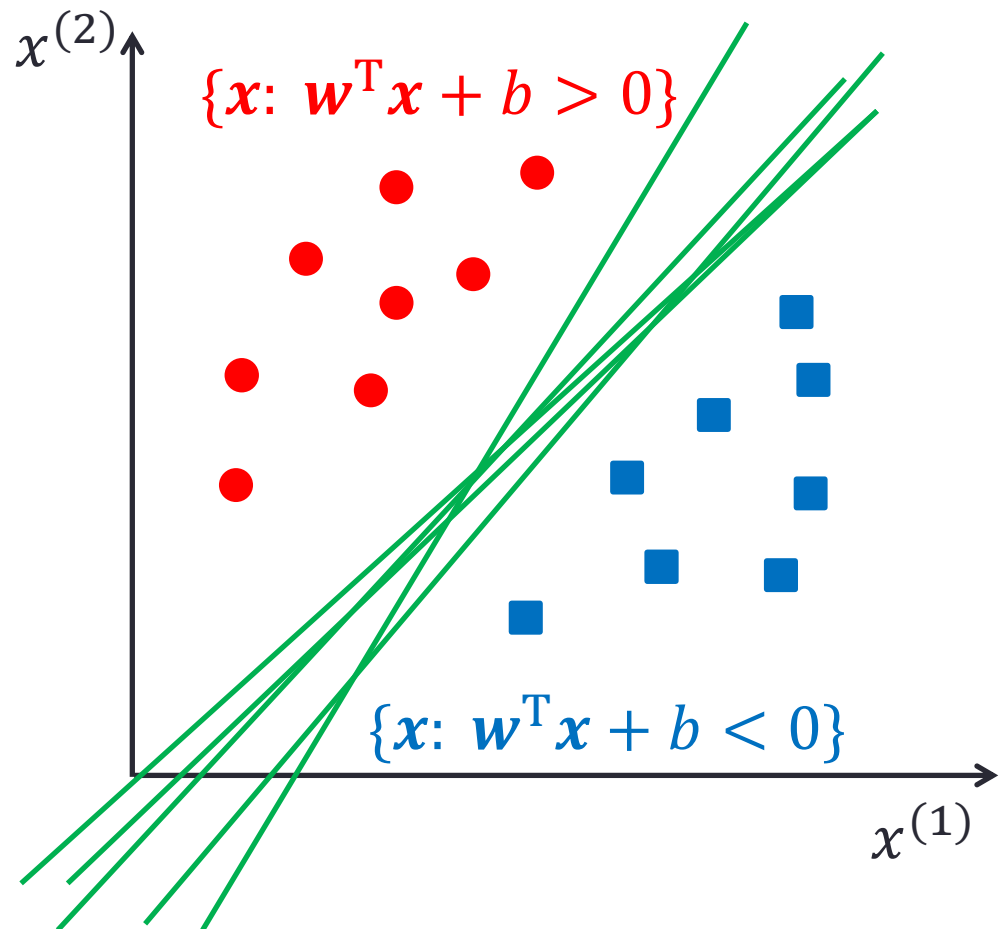


Linearly Separable Case

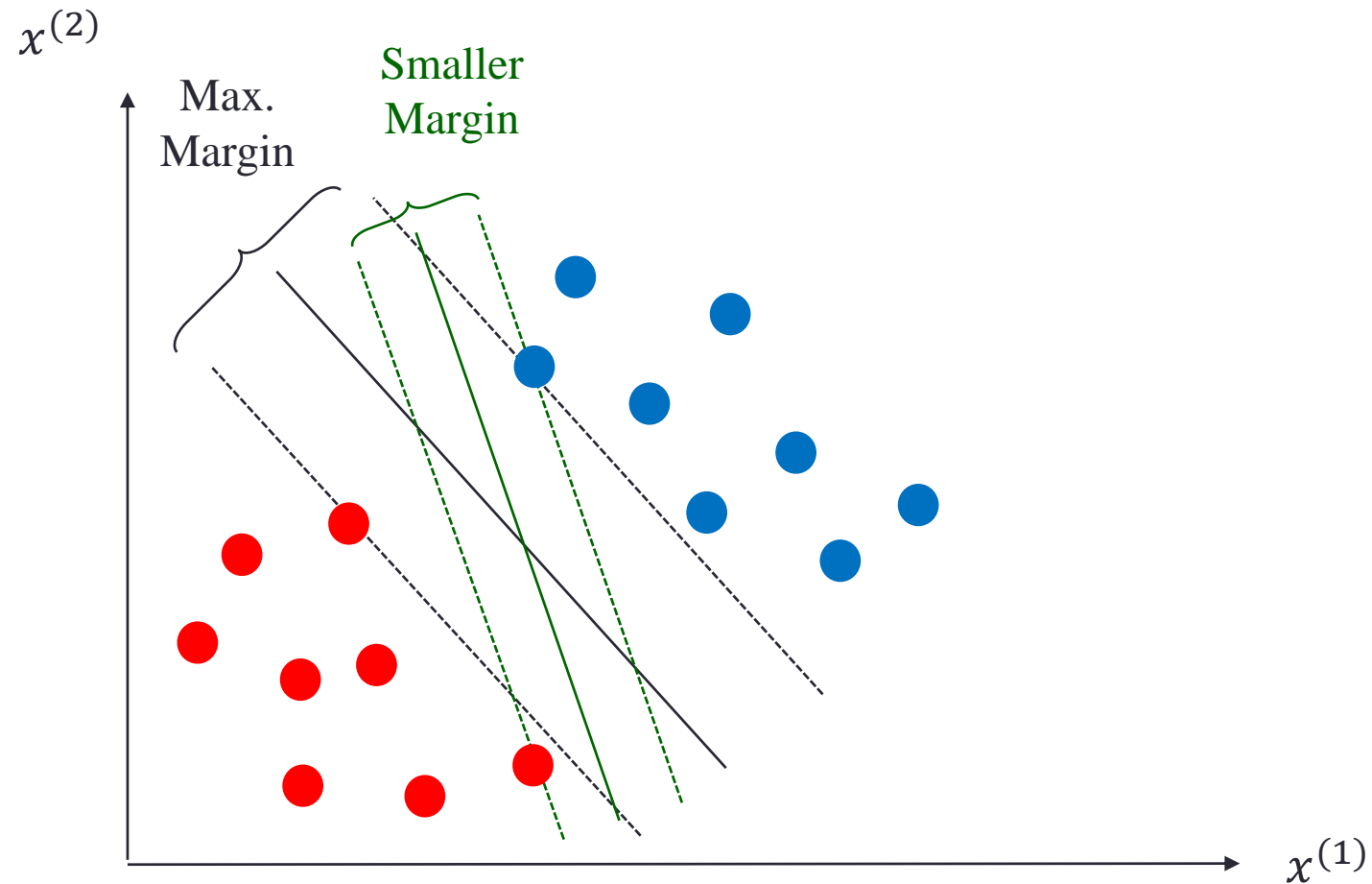
- Necessary condition of a separating hyperplane:

$$f(\mathbf{x}_i) = \text{sgn}(\mathbf{w}^T \mathbf{x}_i + b)$$
$$= \begin{cases} +1, \forall \text{ red} \\ -1, \forall \text{ blue} \end{cases}$$

- Any of these would be fine, but which one is the best?



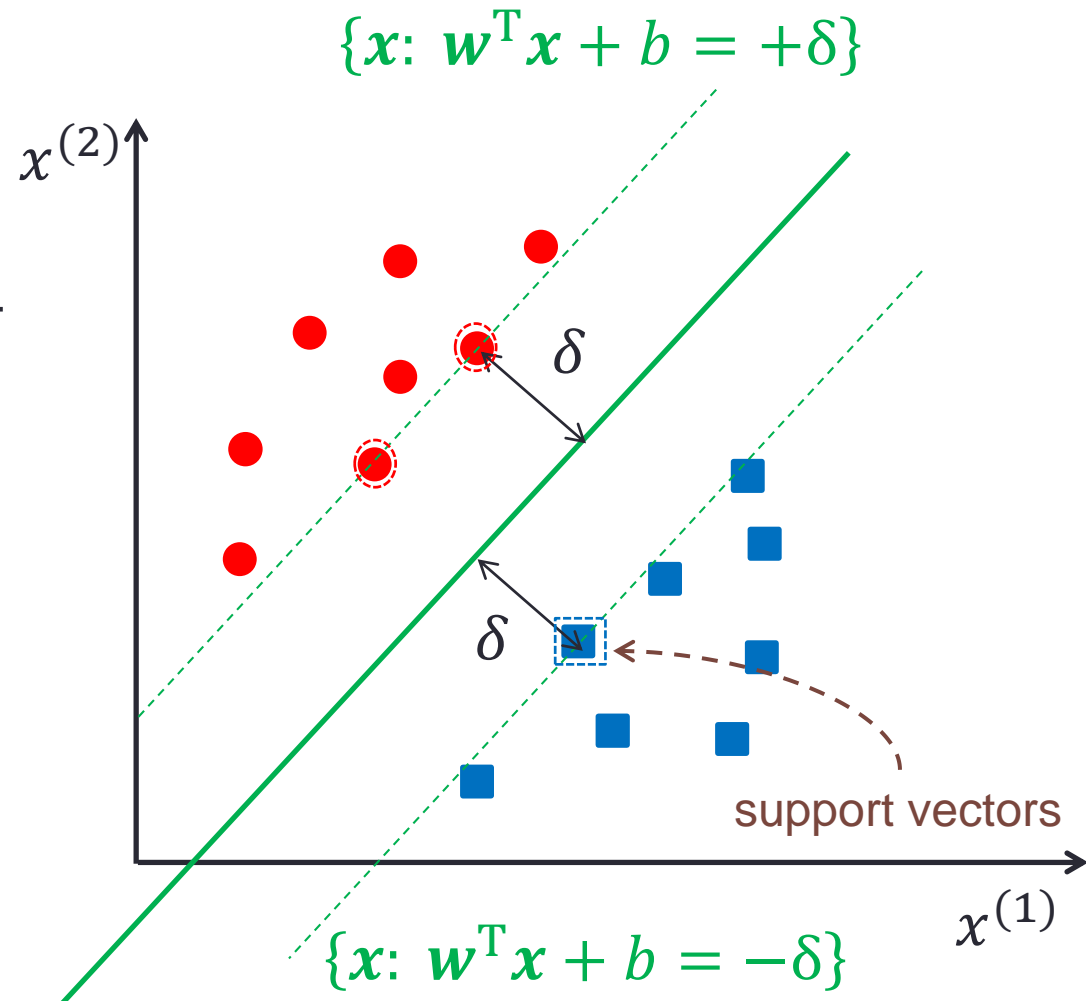
What Hyperplane Is the Best?



Optimal Separating Hyperplane

- Optimal separating hyperplane must maximizes the minimum distance from any sample x_i
- Suppose the minimum distance is δ , the margins are:

$$\begin{cases} \mathbf{w}^T \mathbf{x} + b = +\delta \\ \mathbf{w}^T \mathbf{x} + b = -\delta \end{cases}$$



Why Max-min?

1. If there is a small (measurement) error in the location of the boundary, this gives the least chance of causing a misclassification
2. Empirically it works well
3. It is easy since the model is immune to removal of any nonsupport-vector data points – this means that only the support vectors matter!

Over-parameterized Constraint

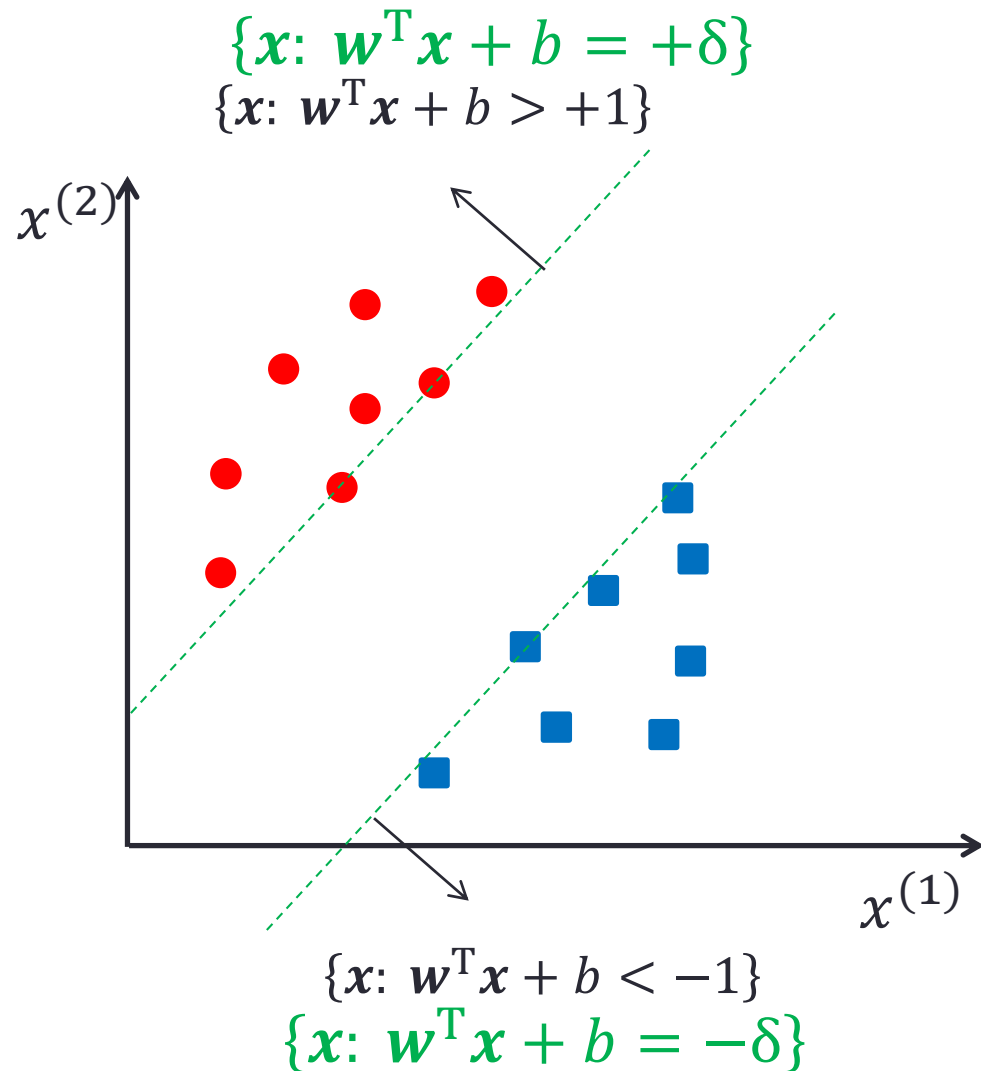
- Constraint for the optimal separating hyperplane

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq +\delta, \forall y_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -\delta, \forall y_i = -1 \end{cases}$$

- However, this can be equally expressed by all sets $(\alpha \mathbf{w}, \alpha b, \alpha \delta)$ for any $\alpha \in \mathbb{R}^+$

- Canonical constraint

$$\begin{cases} \mathbf{w}^T \mathbf{x}_i + b \geq +1, \forall y_i = +1 \\ \mathbf{w}^T \mathbf{x}_i + b \leq -1, \forall y_i = -1 \end{cases}$$



Maximum Margin Classifier

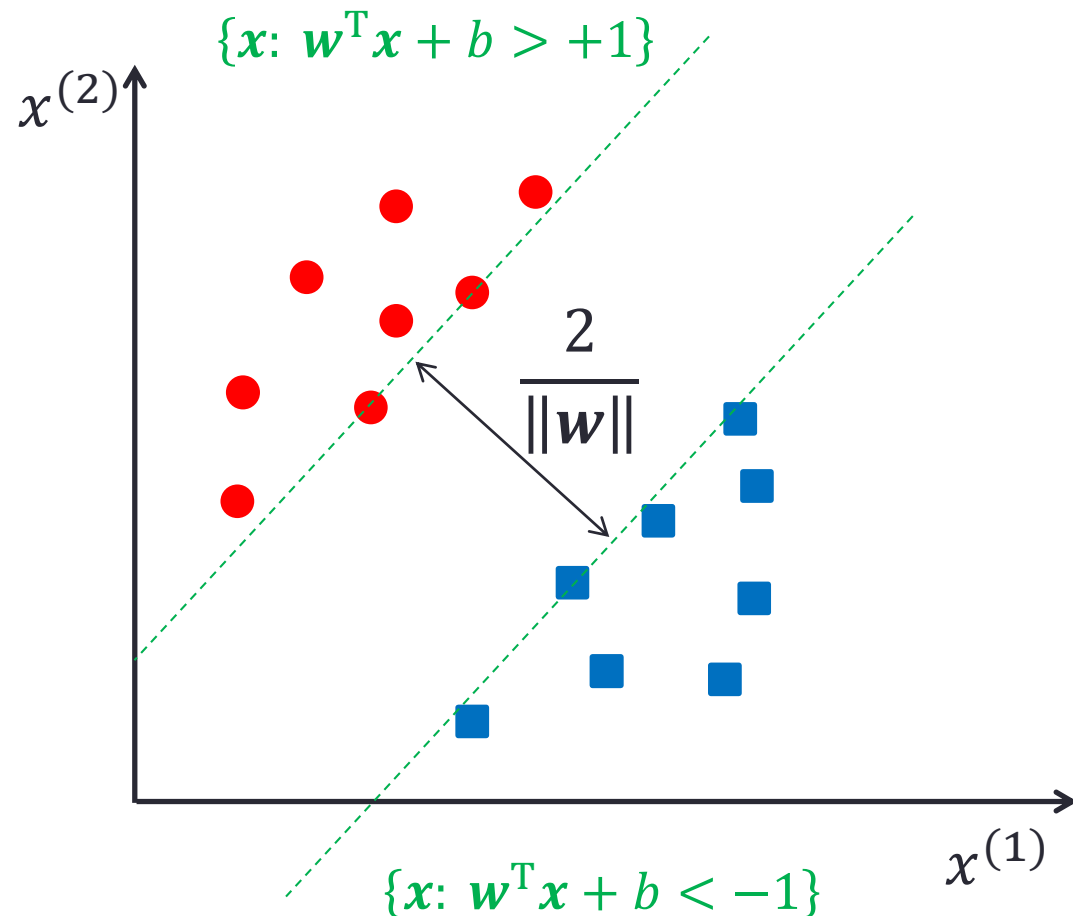
- Maximize the distance between two support hyperplanes:

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

- Subject to the constraint:

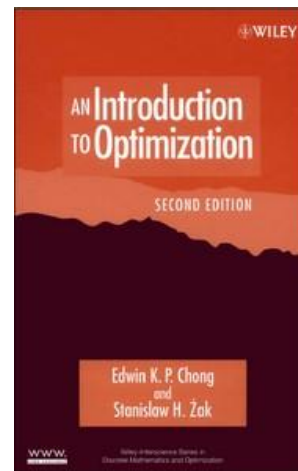
$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$

- Alter the cost to $\frac{1}{2} \|\mathbf{w}\|^2$
- A constraint quadratic programming problem!



Quadratic Programming

- Quadratic programming (QP) is a type of mathematical optimization problem to optimize (minimize or maximize) a quadratic function of several variables subject to linear constraints on these variables
- [*An Introduction to Optimization*](#) by Chong and Zak



- There exists algorithms of finding solutions for QP problems

Solving Optimal Hyperplane Using Lagrangian

- Problem:

Minimize $\frac{1}{2} \|\mathbf{w}\|^2$, subject to $y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$

- Lagrangian:

$$\begin{aligned} L(\mathbf{w}, b, \lambda_i) &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \lambda_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1] \\ &= \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \lambda_i y_i (\mathbf{w}^T \mathbf{x}_i + b) + \sum_{i=1}^n \lambda_i \quad \dots (a) \end{aligned}$$

Karush-Kuhn-Tucker (KKT) Condition

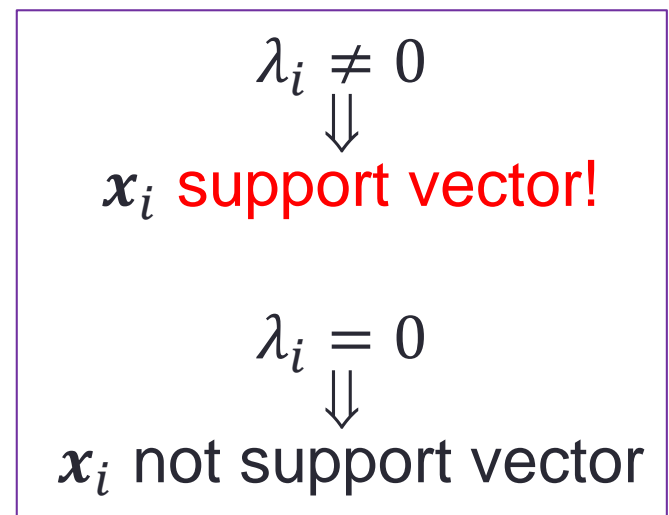
1. $\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i = \mathbf{0} \quad \dots(b)$

2. $\frac{\partial L}{\partial b} = \sum_{i=1}^n \lambda_i y_i = 0 \quad \dots(c)$

3. $y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0, \forall i$

4. $\lambda_i \geq 0, \forall i$

5. $\lambda_i [y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1] = 0, \forall i$



w of the Support Vector Machine

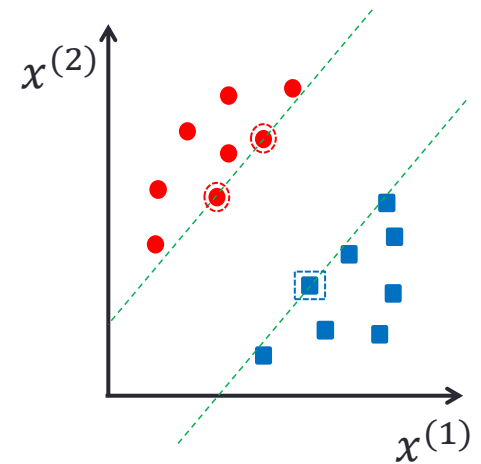
- w is determined by Eq. 1:

$$w = \sum_{i=1}^n \lambda_i y_i x_i$$

- Remember the condition:

x_i not support vector $\Rightarrow \lambda_i = 0$

(w is only determined by the support vectors)



b of the Support Vector Machine

- b can be determined by Eq. 5:

$$\lambda_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1] = 0, \forall i$$

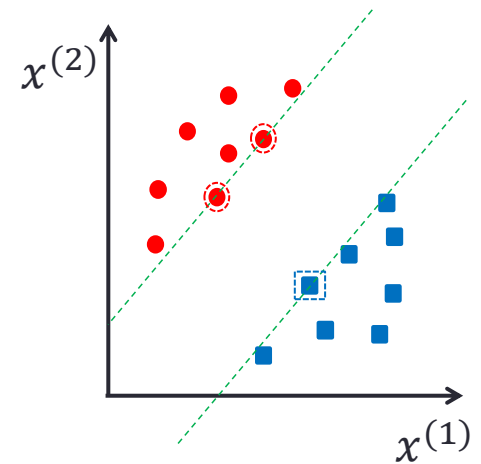
- Remember the condition:

\mathbf{x}_i not support vector $\Rightarrow \lambda_i = 0$

(b is only determined by the support vectors)

- Suppose $\exists \mathbf{x}_i$ s.t. $y_i = +1$

$$b = 1 - \mathbf{w}^T \mathbf{x}_i$$



Linear SVM Solution

- The solution has the form:

$$\mathbf{w} = \sum_{i=1}^n \lambda_i y_i \mathbf{x}_i \quad \text{and} \quad b = 1 - \mathbf{w}^T \mathbf{x}_i$$

Note:

For a SV \mathbf{x}_i with label $y_i = +1$

- The classifier will have the form:

$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b) = \text{sgn}\left(\sum_{i=1}^n \overbrace{\lambda_i y_i}^{\text{Weight}} \mathbf{x}_i^T \mathbf{x} + \underbrace{(1 - \mathbf{w}^T \mathbf{x}_i)}_{\text{Bias}}\right)$$

- Note that $\lambda_i y_i$ is the weight

(Wolfe) Dual Form

- Substitute (b) and (c) into (a):

$$L_d = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

subject to

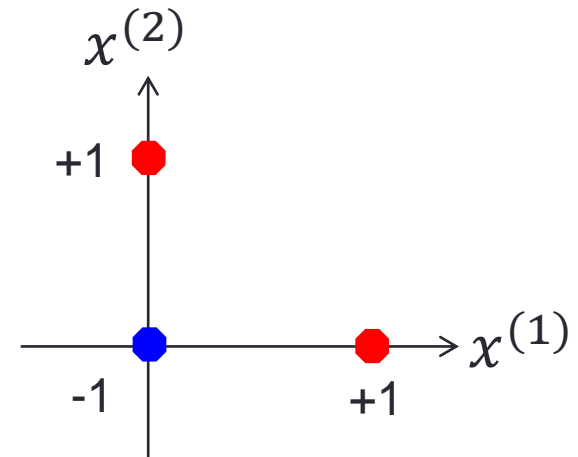
$$\begin{cases} \sum_{i=1}^m \lambda_i y_i = 0 \\ \lambda_i \geq 0 \quad \forall i \end{cases}$$

- SVM parameters can be determined using the dual form

Example: Analytically Solving SVM

- Data:

Input x	Output y
$x_1 = [0 \ 0]^T$	$y_1 = -1$
$x_2 = [1 \ 0]^T$	$y_2 = +1$
$x_3 = [0 \ 1]^T$	$y_3 = +1$



- Strategies:

- Apply the dual form (a constraint optimization problem)
- Introduce another Lagrange multiplier α

Analytically Solving SVM

Input x

$$\mathbf{x}_1 = [0 \ 0]^T$$

$$\mathbf{x}_2 = [1 \ 0]^T$$

$$\mathbf{x}_3 = [0 \ 1]^T$$

$$\begin{aligned}
 L_n(\lambda_i, \alpha) &= f(\lambda_i) - \alpha g(\lambda_i) \\
 &= \sum_{i=1}^3 \lambda_i - \frac{1}{2} \sum_{i=1}^3 \sum_{j=1}^3 \lambda_i \lambda_j y_i y_j \mathbf{x}_i^T \mathbf{x}_j - \alpha \sum_{i=1}^3 \lambda_i y_i \\
 &= \lambda_1 + \lambda_2 + \lambda_3 \\
 &\quad - \frac{1}{2} \left(\cancel{\lambda_1 \lambda_1 y_1 y_1 \mathbf{x}_1^T \mathbf{x}_1} + \lambda_2 \lambda_2 y_2 y_2 \mathbf{x}_2^T \mathbf{x}_2 + \lambda_3 \lambda_3 y_3 y_3 \mathbf{x}_3^T \mathbf{x}_3 \right. \\
 &\quad \left. + \cancel{2\lambda_1 \lambda_2 y_1 y_2 \mathbf{x}_1^T \mathbf{x}_2} + \cancel{2\lambda_1 \lambda_3 y_1 y_3 \mathbf{x}_1^T \mathbf{x}_3} + \cancel{2\lambda_2 \lambda_3 y_2 y_3 \mathbf{x}_2^T \mathbf{x}_3} \right) \\
 &\quad - \alpha (\lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3)
 \end{aligned}$$

Analytically Solving SVM (Cont'd)

$$L_n(\lambda_i, \alpha) = \lambda_1 + \lambda_2 + \lambda_3 - \frac{1}{2}\lambda_2^2 - \frac{1}{2}\lambda_3^2 - \alpha(-\lambda_1 + \lambda_2 + \lambda_3)$$

$$\Rightarrow \begin{cases} \frac{\partial L_n}{\partial \lambda_1} = 1 + \alpha = 0 \\ \frac{\partial L_n}{\partial \lambda_2} = 1 - \lambda_2 - \alpha = 0 \end{cases} \quad \begin{cases} \frac{\partial L_n}{\partial \lambda_3} = 1 - \lambda_3 - \alpha = 0 \\ \frac{\partial L_n}{\partial \alpha} = -\lambda_1 + \lambda_2 + \lambda_3 = 0 \end{cases}$$

$$\Rightarrow \begin{cases} \alpha = -1 \\ \lambda_1 = 4 \end{cases} \quad \begin{cases} \lambda_2 = 2 \\ \lambda_3 = 2 \end{cases}$$

Solving w and b

Input x	Output y
$x_1 = [0 \ 0]^T$	$y_1 = -1$
$x_2 = [1 \ 0]^T$	$y_2 = +1$
$x_3 = [0 \ 1]^T$	$y_3 = +1$

- Now solve w :

$$w = \sum_{i=1}^3 \lambda_i y_i x_i = 4(-1) \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

- Solve b :

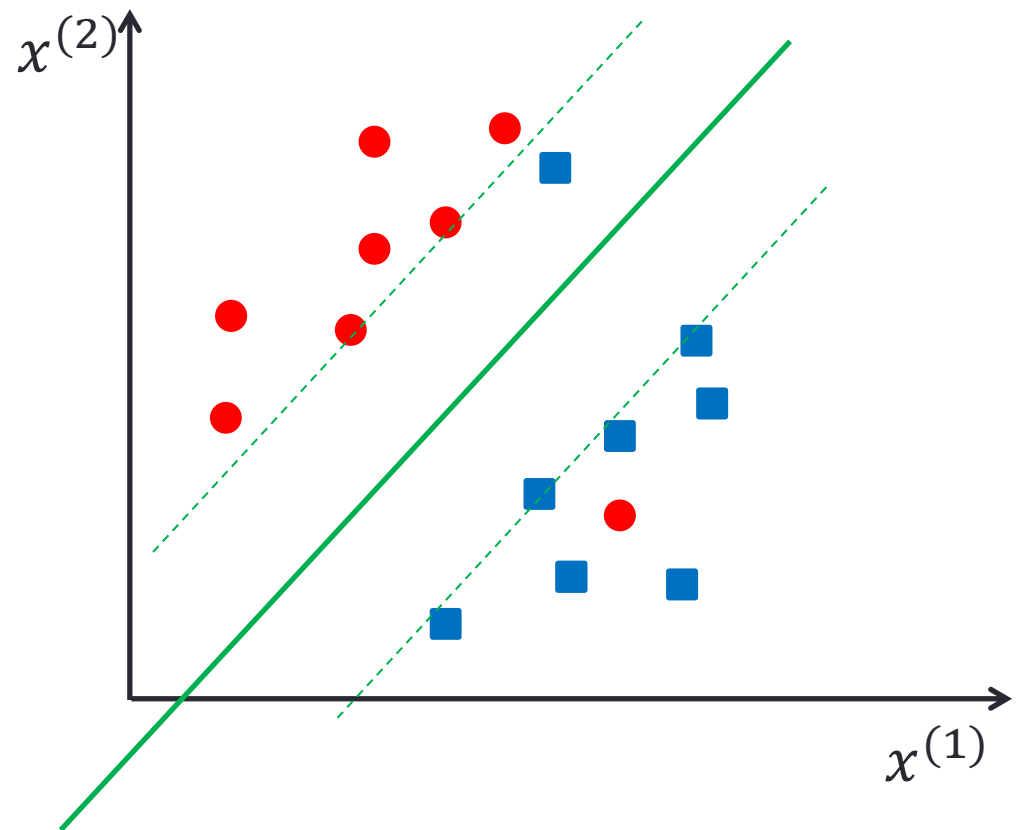
$$b = 1 - w^T x_i = 1 - [2 \ 2] \begin{bmatrix} 1 \\ 0 \end{bmatrix} = -1$$

- SVM:

$$f(x) = \text{sgn}(w^T x + b) = \text{sgn}([2 \ 2] x - 1)$$

Data with Noise

- So far we assume that the data points are linearly separable
- What if the training data is noisy?

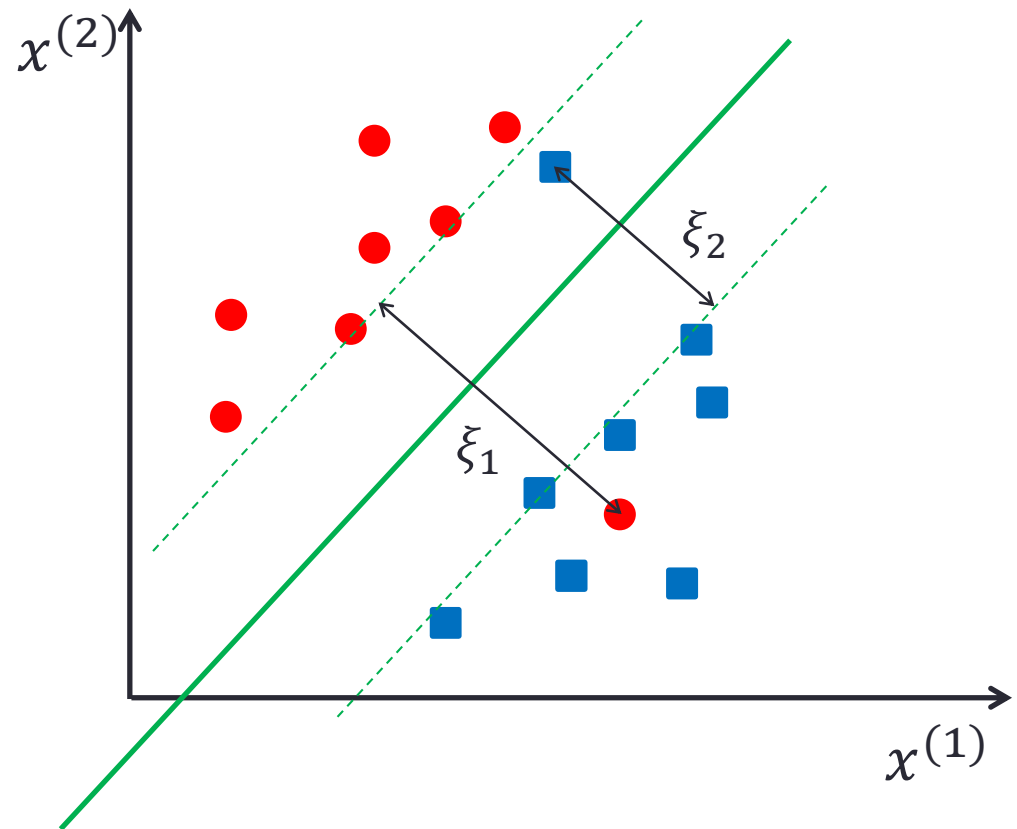


Slack Variable

- Slack variables ξ_i to allow misclassification of difficult or noisy examples
- Suppose there exists $r \in \mathbb{N}$ misclassification samples
- Cost function:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^r \xi_i$$

where $c > 0$



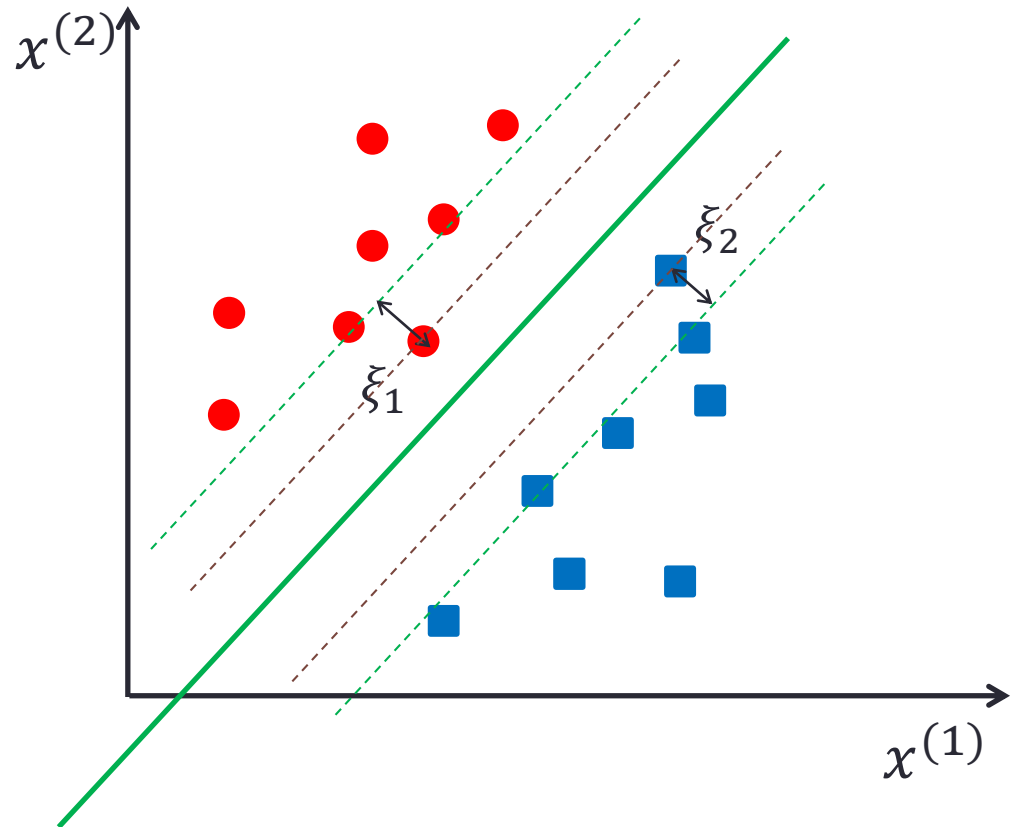
Soft Margin SVM Problem Statement

- Cost function:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^r \xi_i$$

subject to

$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

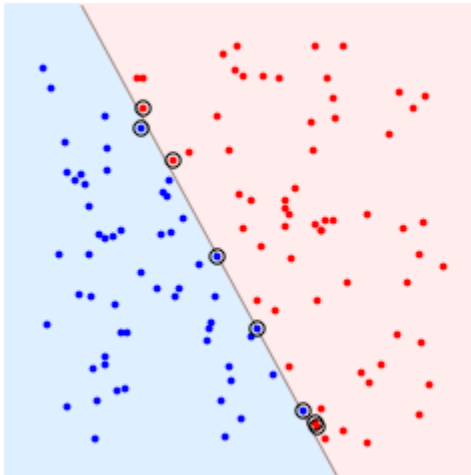


How Does c Impact the Margin?

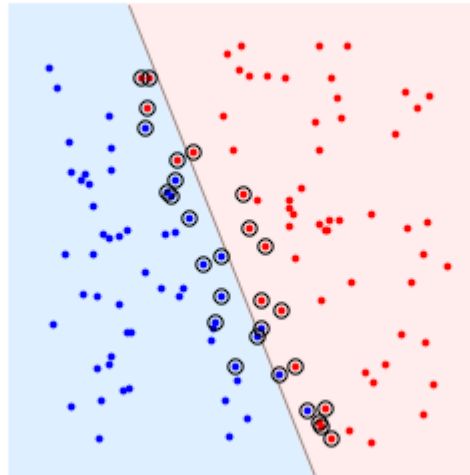
- Cost function: $\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^r \xi_i$

Large $c \Rightarrow$ Small $\sum_{i=1}^r \xi_i \Rightarrow$ Small $\xi_i \Rightarrow$ Small margin

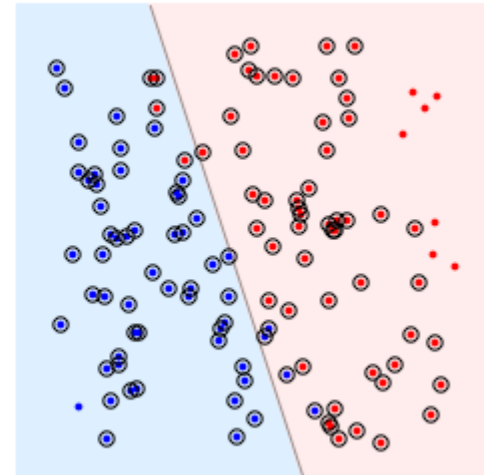
$c = 1000$



$c = 10$



$c = 0.1$



Lagrangian of Soft-margin SVM

- One cost function and two constraints
- Lagrangian:

$$\begin{aligned} & L(\mathbf{w}, b, \lambda_i, \xi_i, \gamma_i) \\ &= \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^r \xi_i - \sum_{i=1}^n \lambda_i [y_i (\mathbf{w}^T \mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^r \gamma_i \xi_i \end{aligned}$$

Hard Margin v.s. Soft Margin SVM

- Hard margin SVM formulation:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2$$
$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \forall i$$

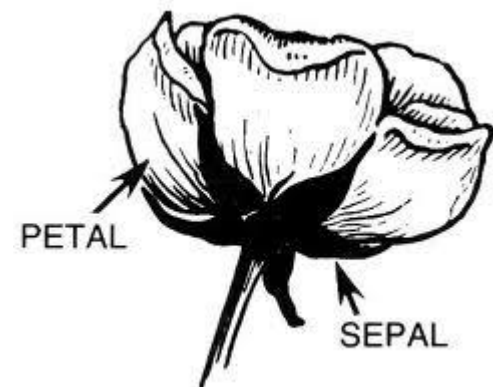
- Soft margin SVM formulation:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^r \xi_i$$
$$y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1 - \xi_i \quad \forall i$$
$$\xi_i \geq 0 \quad \forall i$$

- Parameter c can be viewed as a way to control overfitting

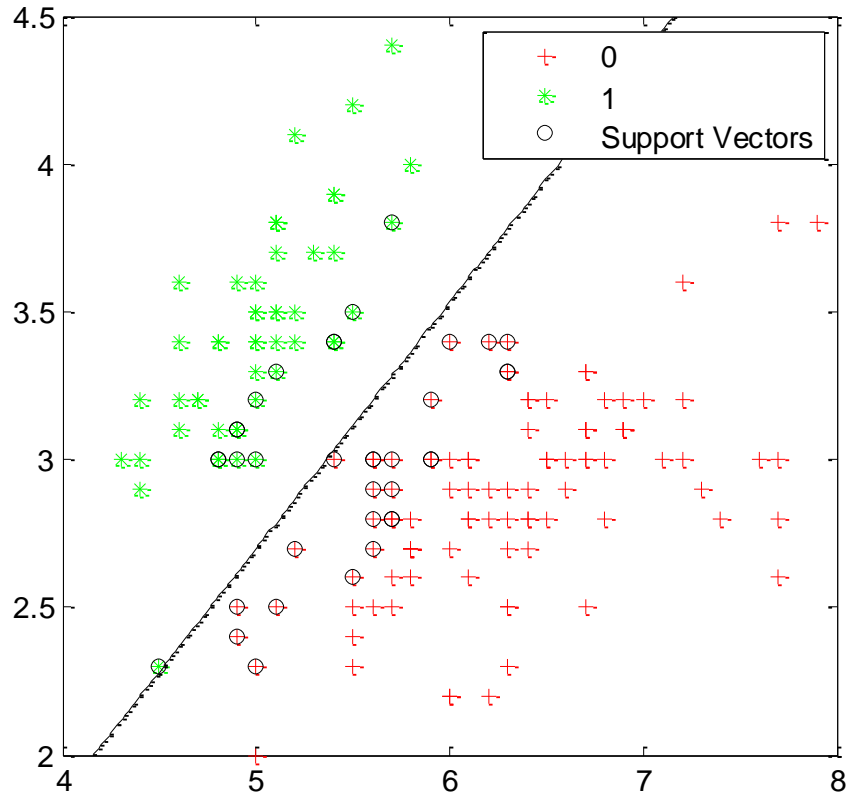
Example – Fisher's Iris Data

- Three flower types (classes):
Setosa, Virginica, Versicolour
- Four (non-class) attributes:
Sepal width and length,
Petal width and length
- Want to distinguish between
 - Setosa
 - Virginica and Versicolour

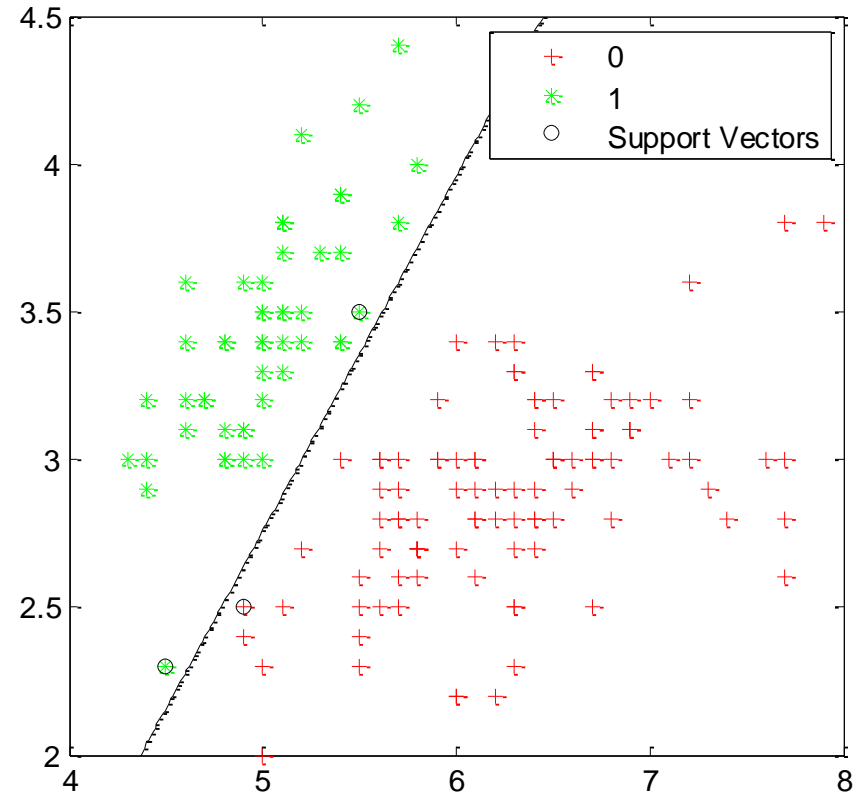


SVM Classifiers for Fisher's Iris

Linear Kernel with $C = 0.1$



Linear Kernel with $C = 100$



Example MATLAB Code

```
load fisheriris; %Load the data
data = [meas(:,1), meas(:,2)];
groups = ismember(species,'setosa'); % Setosa class

%Use a linear support vector machine classifier
subplot(1,2,1);
svmStruct =
svmtrain(data,groups,'boxconstraint',0.1,'showplot',true);
title('Linear Kernel with C = 0.1');

%Use a linear support vector machine classifier
subplot(1,2,2);
svmStruct =
svmtrain(data,groups,'boxconstraint',100,'showplot',true);
title('Linear Kernel with C = 100');
```

Note that this is using Matlab built-in SVM, NOT libSVM

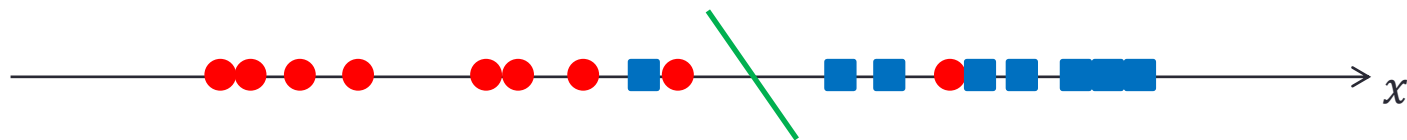
Review of Linear SVM

- The classifier is a separating hyperplane
- Most “important” data points are support vectors – they define the hyperplane (\mathbf{w} and b)
- Quadratic optimization algorithms can identify which data points \mathbf{x}_i are support vectors with non-zero Lagrangian multipliers λ_i
- In the formulation of the classifier, it appears only the inner products of the data points $\mathbf{x}_i^T \mathbf{x}$, i.e.,

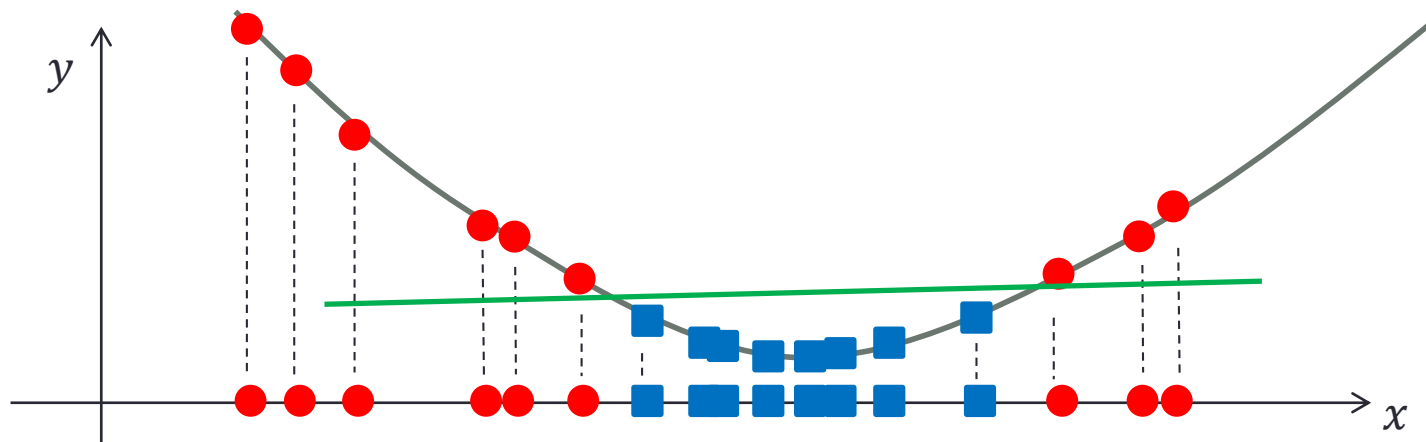
$$f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b) = \text{sgn}\left(\sum_{i=1}^n \lambda_i y_i \mathbf{x}_i^T \mathbf{x} + b\right)$$

Non-linear SVM

- Soft margin may work on datasets that are linearly separable with some noise:



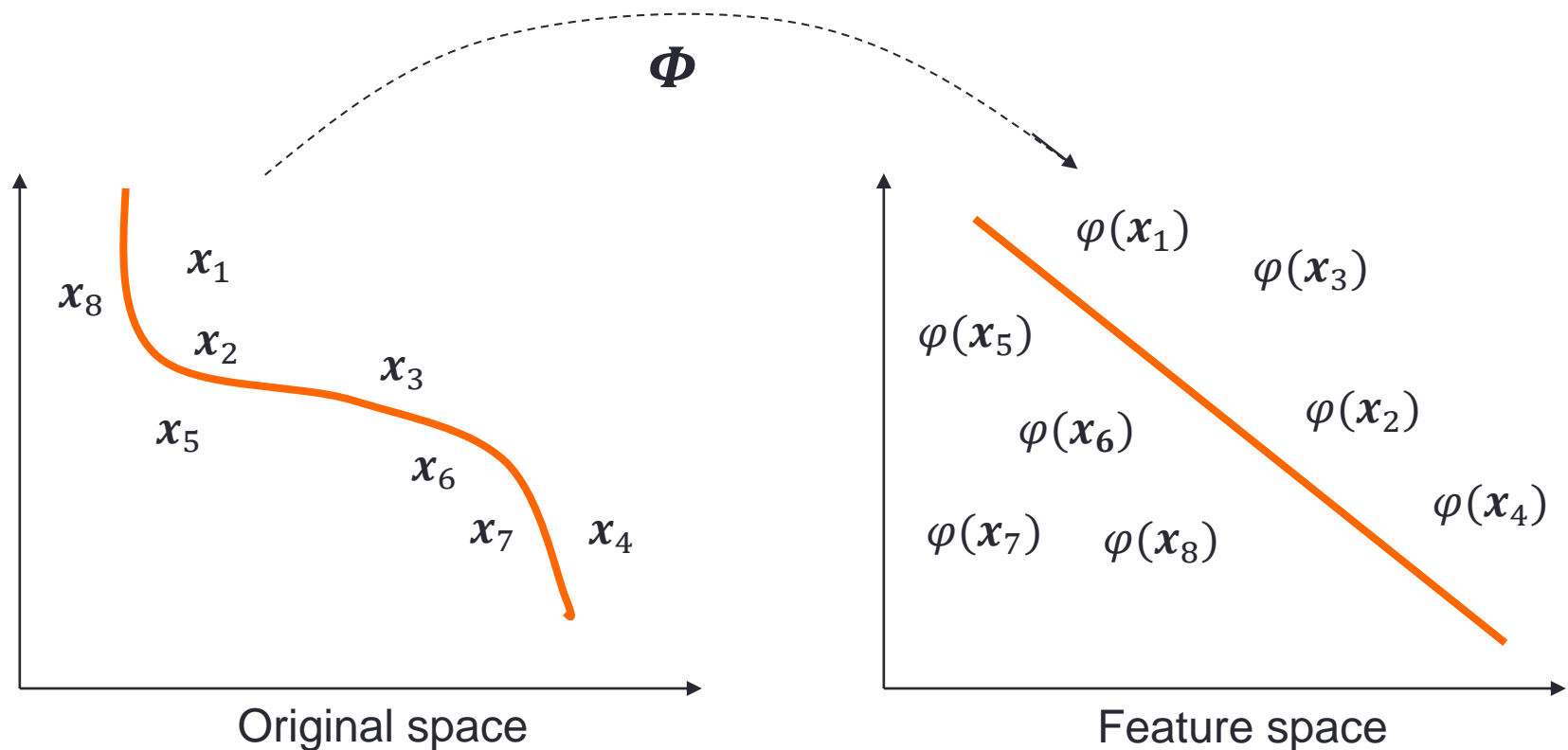
- What if the dataset is not just “noisy”?



- Strategy – mapping data to a higher-dimensional space

Feature Space

- Kernel function $\Phi: \mathbf{x}_i \rightarrow \varphi(\mathbf{x}_i)$ is used to map data into higher-dimensional feature space where they may be linearly separable

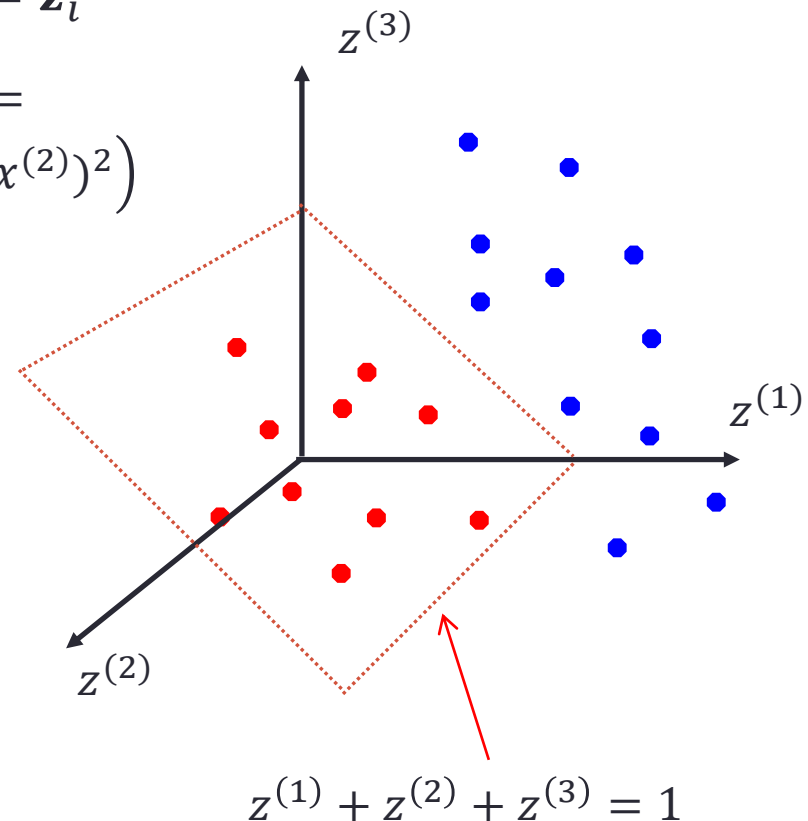
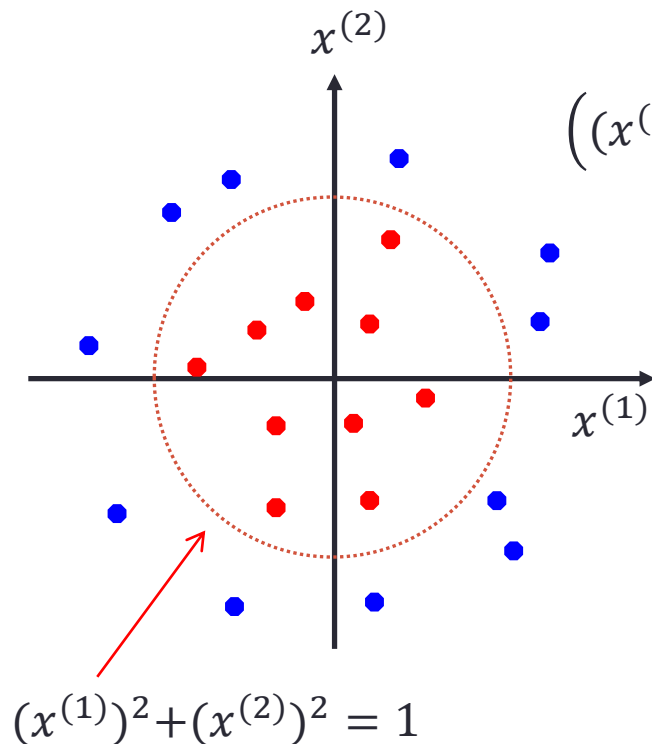


Example Kernel Function

- Inside/outside unit circle to a 3-dimensional feature space

$$\Phi: \mathbf{x}_i \rightarrow \varphi(\mathbf{x}_i) = \mathbf{z}_i$$

$$\begin{aligned} (z^{(1)}, z^{(2)}, z^{(3)}) = \\ ((x^{(1)})^2, \sqrt{2}x^{(1)}x^{(2)}, (x^{(2)})^2) \end{aligned}$$



Kernel SVM

- The linear SVM relies on inner product between vectors $\mathbf{x}_i^T \mathbf{x}_j$
- The non-linear SVM relies on inner product between the inner product becomes $\varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$
- The classifier:

$$f(\varphi(\mathbf{x})) = \text{sgn} \left(\sum_{i=1}^n \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b \right)$$

- Note that we do NOT need to know the kernel function $\varphi(\mathbf{x})$ but the inner product of kernel $K(\mathbf{x}_i, \mathbf{x}_j)$ to calculate the classification

Typical Kernel Function

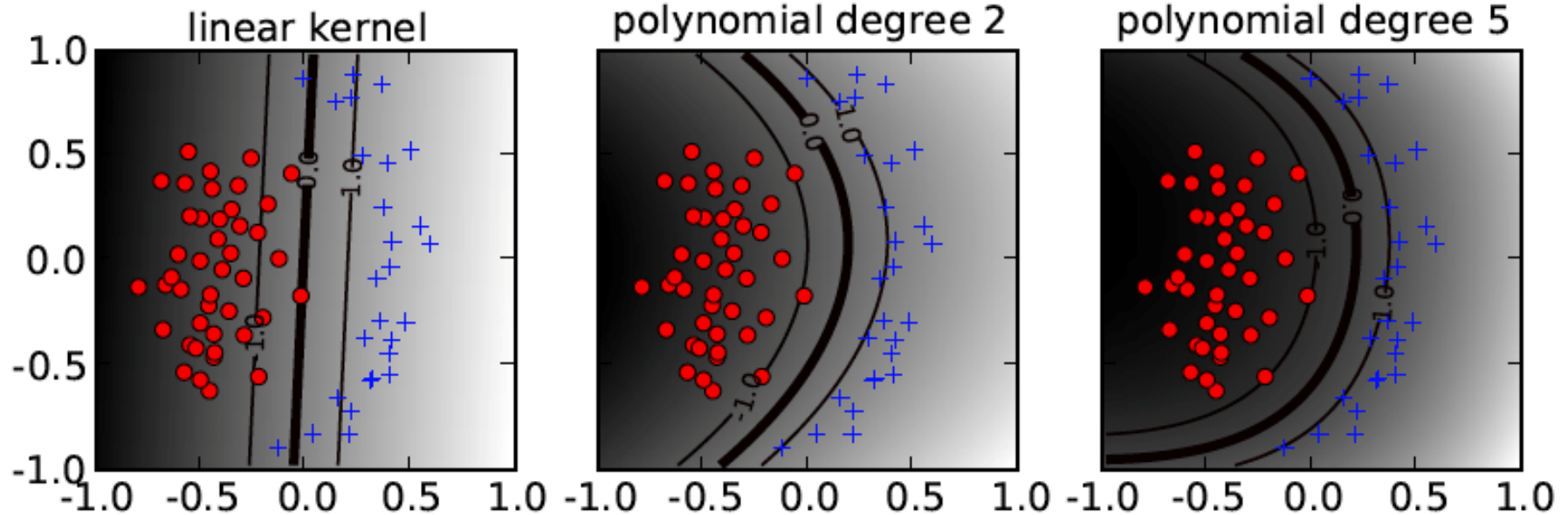
- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial of power p : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Gaussian (radial-basis function network):

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

- Sigmoid: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$

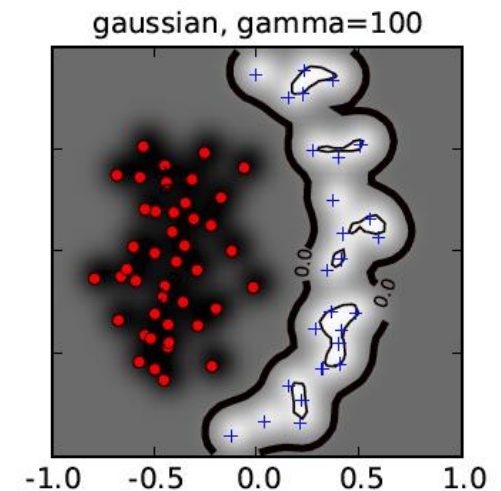
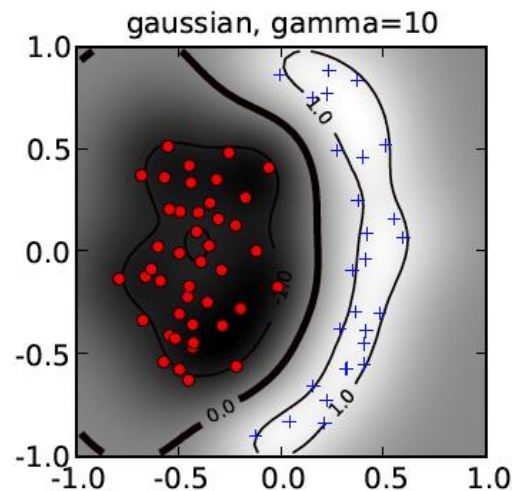
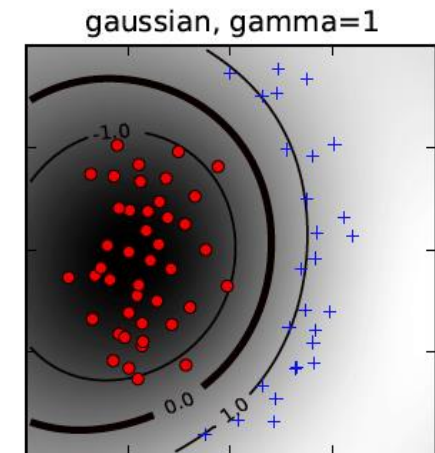
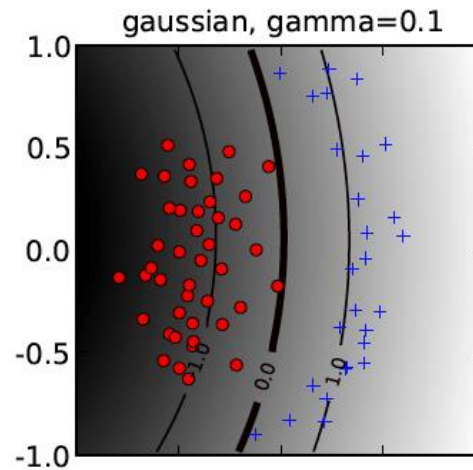
Polynomial Kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$$



RBF Kernel

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$



$\varphi(\mathbf{x}_i)$ of the Polynomial Kernel

- What is the kernel function $\varphi(\mathbf{x}_i)$ for 2nd-order polynomial kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^2$?
- Let $\mathbf{x} = [x^{(1)} \ x^{(2)}]^T$

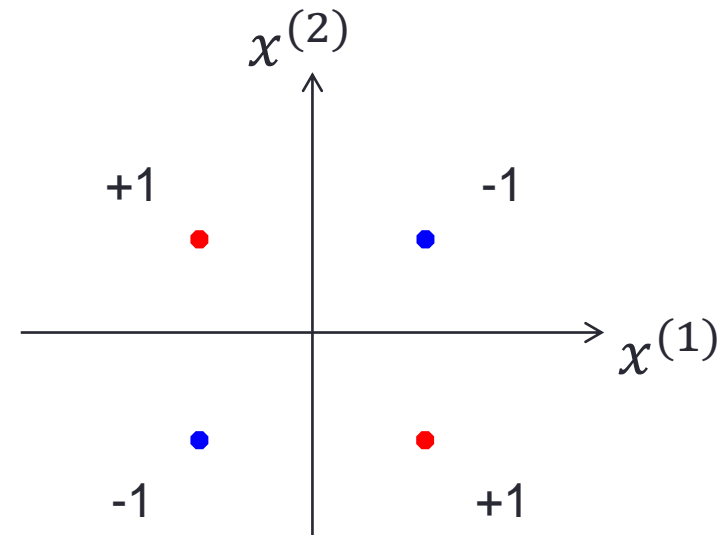
$$\begin{aligned} K(\mathbf{x}_i, \mathbf{x}_j) &= (1 + \mathbf{x}_i^T \mathbf{x}_j)^2 = \left(1 + [x_i^{(1)} \ x_i^{(2)}] \begin{bmatrix} x_j^{(1)} \\ x_j^{(2)} \end{bmatrix} \right)^2 \\ &= \begin{bmatrix} 1 & x_i^{(1)2} & \sqrt{2}x_i^{(1)}x_i^{(2)} & x_i^{(2)2} & \sqrt{2}x_i^{(1)} & \sqrt{2}x_i^{(2)} \end{bmatrix} \\ &\quad \begin{bmatrix} 1 & x_j^{(1)2} & \sqrt{2}x_j^{(1)}x_j^{(2)} & x_j^{(2)2} & \sqrt{2}x_j^{(1)} & \sqrt{2}x_j^{(2)} \end{bmatrix}^T \\ &= \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}_j) \end{aligned}$$

It is NOT always possible to decompose the inner product of the kernel function

Example: Kernel SVM

- XOR:

Input x	Output y
$x_1 = [+1 \ +1]^T$	$y_1 = -1$
$x_2 = [+1 \ -1]^T$	$y_2 = +1$
$x_3 = [-1 \ +1]^T$	$y_3 = +1$
$x_4 = [-1 \ -1]^T$	$y_4 = -1$



- Cannot be solved by linear SVM
- Choose Polynomial kernel function

$$K(x_i, x_j) = (1 + x_i^T x_j)^2$$

that maps $x = [x^{(1)}, x^{(2)}]^T$ into six-dimensional feature space

$$\varphi(x) = [1, (x^{(1)})^2, \sqrt{2}x^{(1)}x^{(2)}, (x^{(2)})^2, \sqrt{2}x^{(1)}, \sqrt{2}x^{(2)}]^T$$

XOR Problem Cost Function

- Dual form Lagrangian:

$$L_d = \sum_{i=1}^n \lambda_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \lambda_i \lambda_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

$$\begin{aligned} &= \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 - \frac{1}{2} (9(\lambda_1)^2 - 2\lambda_1\lambda_2 - 2\lambda_1\lambda_3 + 2\lambda_1\lambda_4 \\ &\quad + 9(\lambda_2)^2 + 2\lambda_2\lambda_3 - 2\lambda_2\lambda_4 + 9(\lambda_3)^2 - 2\lambda_3\lambda_4 + 9(\lambda_4)^2) \end{aligned}$$

- Differentiate against λ_i , the optimal Lagrange multiplier:

$$\lambda_1 = \lambda_2 = \lambda_3 = \lambda_4 = \frac{1}{8}$$

- This implies that all the four input vectors are support vectors

Review: Nonlinear SVM Solution

- Classifier of nonlinear SVM

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn} \left(\sum_{i=1}^n \lambda_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \\ &= \text{sgn} \left(\sum_{i=1}^n \lambda_i y_i \varphi(\mathbf{x}_i)^T \varphi(\mathbf{x}) + b \right) \\ &= \text{sgn} \left(\left(\sum_{i=1}^n \lambda_i y_i \varphi(\mathbf{x}_i) \right)^T \varphi(\mathbf{x}) + b \right) \end{aligned}$$

XOR Problem Solution

$$\begin{aligned}\sum_{i=1}^n \lambda_i y_i \varphi(\mathbf{x}_i) &= \frac{1}{8} (-\varphi(\mathbf{x}_1) + \varphi(\mathbf{x}_2) + \varphi(\mathbf{x}_3) - \varphi(\mathbf{x}_4)) \\ &= \frac{1}{8} \left(-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}\end{aligned}$$

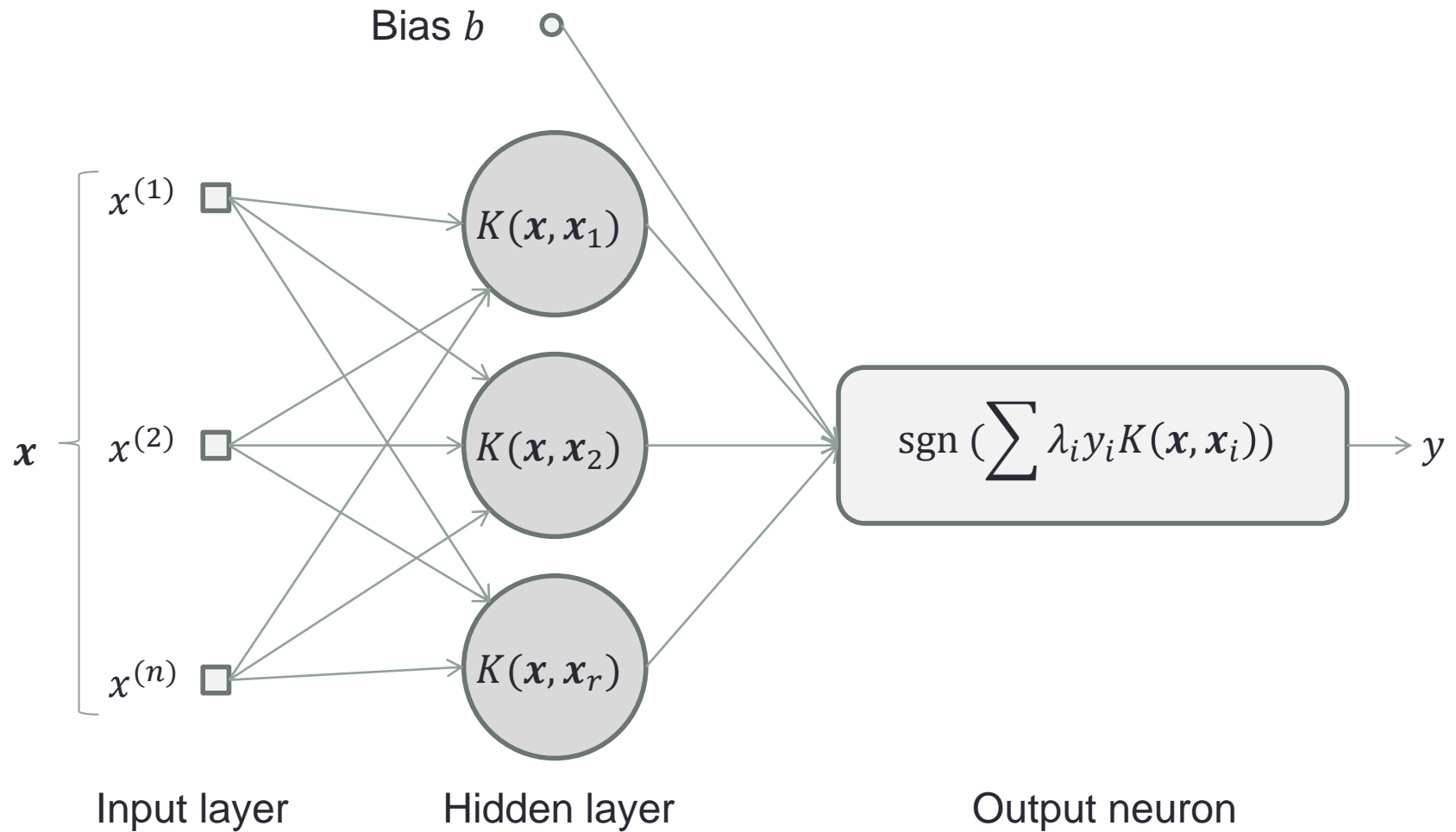
XOR Problem Solution (Cont'd)

$$\begin{aligned} f(\mathbf{x}) &= \text{sgn} \left(\left(\sum_{i=1}^n \lambda_i y_i \varphi(\mathbf{x}_i) \right)^T \varphi(\mathbf{x}) + b \right) \\ &= \text{sgn} \left(\left(\begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix} \right)^T \begin{bmatrix} 1 \\ (x^{(1)})^2 \\ \sqrt{2}x^{(1)}x^{(2)} \\ (x^{(2)})^2 \\ \sqrt{2}x^{(1)} \\ \sqrt{2}x^{(2)} \end{bmatrix} + 0 \right) \\ &= \text{sgn}(-x^{(1)}x^{(2)}) \end{aligned}$$

Review of Non-linear SVM

- SVM locates a separating hyperplane in the feature space and classify points in that space
- It does not need to represent the space explicitly, simply by defining a kernel function
- The kernel function plays the role of the inner product in the feature space

Architecture of Support Vector Machine



Why SVM Works?

- Why SVM doesn't have the curse of dimensionality since the feature space is often very high dimensional?
- Vapnik: the fundamental problem is NOT the number of parameters to be estimated; rather, it is about the flexibility (VC-dimension) of a classifier
- Another view: the term $\frac{1}{2} \|\mathbf{w}\|^2$ "shrinks" the parameters towards zero to avoid overfitting
- The maximum margin hyperplane is stable – there are usually few support vectors relative to the size of the training set

Nice Properties of SVM

- Nice mathematic property – a simple convex optimization problem which is guaranteed to converge to a single global solution
- Sparseness of solution when dealing with large data sets – only support vectors are used to specify the separating hyperplane
- Feature selection – some entries of \mathbf{w} could be zero
- Ability to handle large feature spaces – complexity does not depend on the dimensionality of the feature space but on the dimensionality of the inner product (kernel)

Strength of SVM

- Training is relatively easy – no local optimal, unlike in neural networks
- It scales relatively well to high dimensional data
- Tradeoff between classifier complexity and error can be controlled explicitly
- Flexible in input variables – non-traditional data, such as strings and trees, can be used as input to SVM

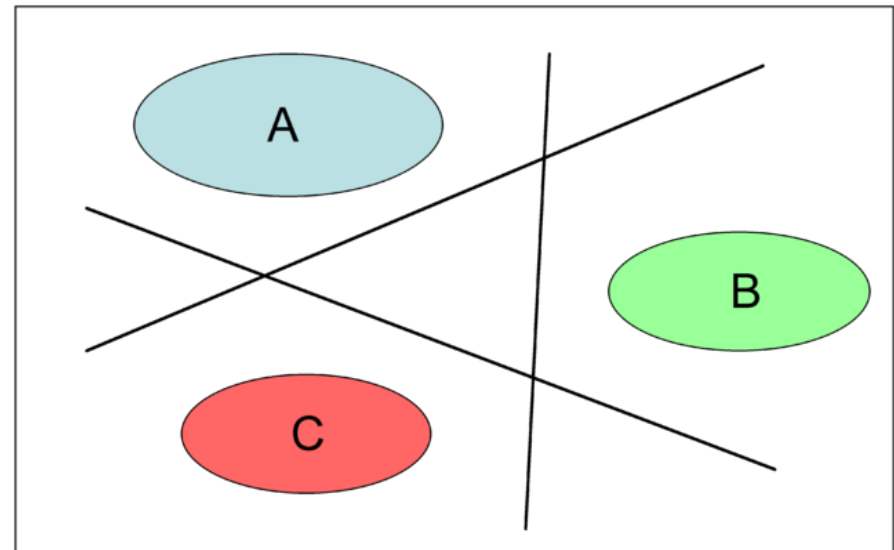
Weakness of SVM

- Sensitive to noise – a relatively small number of mislabeled examples can dramatically decrease the performance
- It only considers two classes – how to do multi-class classification with SVM?



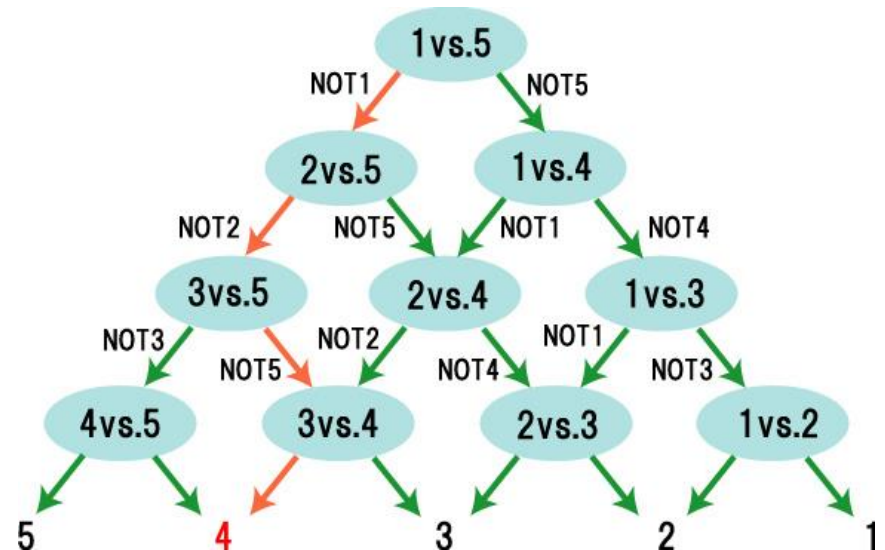
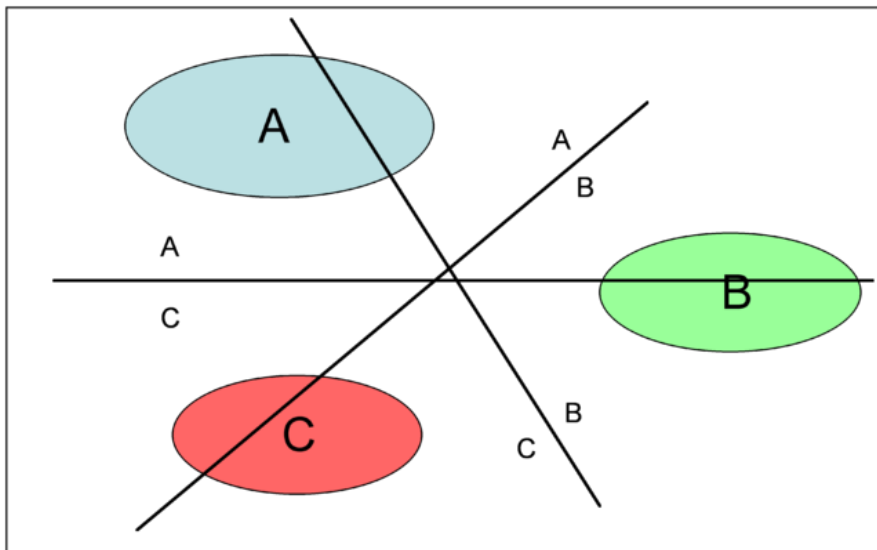
Multi-class SVM

- Two strategies – building binary classifiers which distinguish between (i) one of the classes to the rest (one-versus-all) or (ii) between every pair of classes (one-versus-one)
- One-versus-all: Train $q \in \aleph$ SVMs each of which separates a single class from all the others, and the classification is done by “winner takes all strategy”



Multi-class SVM (Cont'd)

- One-versus-one: Train $q(q - 1)/2 \in \mathbb{N}$ SVMs each of which separates a pair of classes, and the classification is done by “max-wins” voting strategy



- Experimentally no difference between the two

Some Other Issues

- Choice of kernel
 - What kernel should one choose, Gaussian or polynomial?
 - In practice, a low degree polynomial kernel or RBF kernel with a reasonable width is a good initial try for most applications
- Choice of kernel parameters
 - How does one choose parameters in kernel, e.g. σ in Gaussian kernel
- In the absence of reliable criteria, applications rely on the use of cross-validation to make such decisions

LibSVM

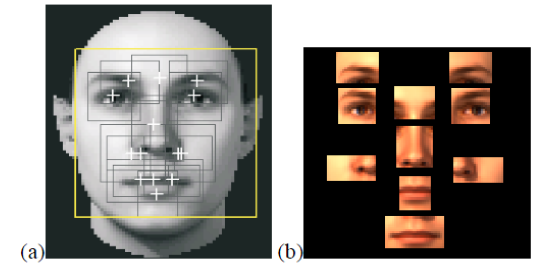
- A library for SVMs developed by NTU CSIE
- Library website: <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>
- Guide:
<http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf>
- Test data:
<http://www.csie.ntu.edu.tw/~cjlin/papers/guide/data/>
- Read “README” in the package first before act!

LibSVM (Cont'd)

- Procedure suggested by the guild
 1. Transform data to the format of an SVM package
 2. Conduct simple scaling on the data
 3. Consider the RBF kernel: $K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$
 4. Use cross-validation to find the best parameter c and σ
 5. Use the best parameter c and σ to train the whole training set
 6. Test

Face Recognition Using SVM

- [Heisle, Ho and Poggio](#)
- One-versus-all strategy
- Two approaches – global and component
- Global approach – the gray values of a face picture are converted to a feature vector
- Component approach – facial components are detected, and the final detection is made by combining the results of the component classifiers
- Real-time face



Real-time Facial Expression Recognition

Real-time facial expression
recognition

30 training pictures / emotion

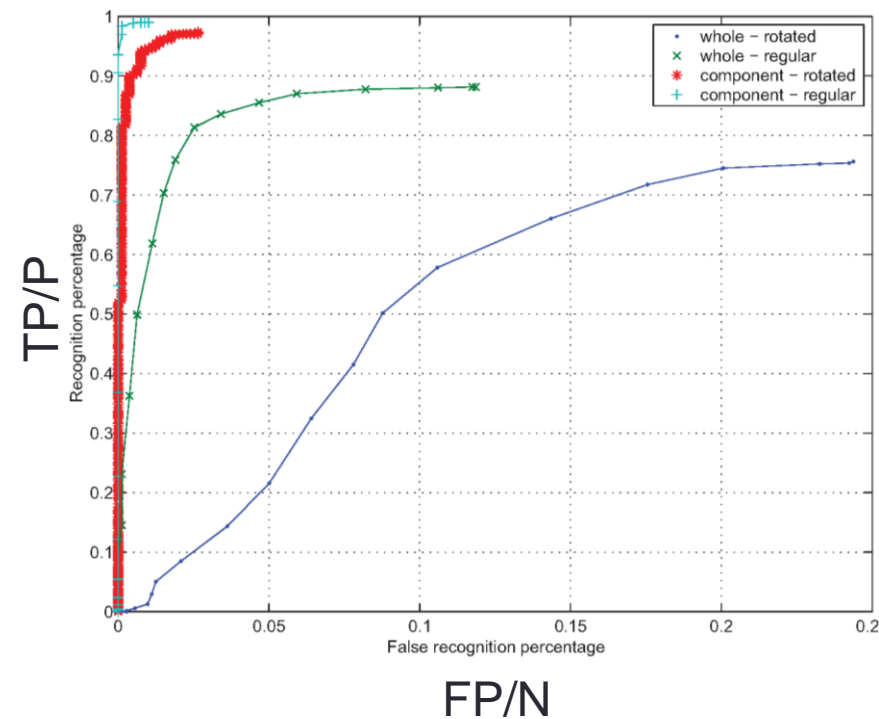
Mitchel Benovoy
Centre for Intelligent Machines
McGill University

2007

Result of Face Recognition

- Receiver operating characteristic (ROC) figure

		Actual value	
		<i>positive</i>	<i>negative</i>
Prediction outcome	<i>positive</i>	True Positive	False Positive
	<i>negative</i>	False Negative	True Negative
total		P	N



- The Component-based algorithm showed much better results than the Global approach

SVM Regression

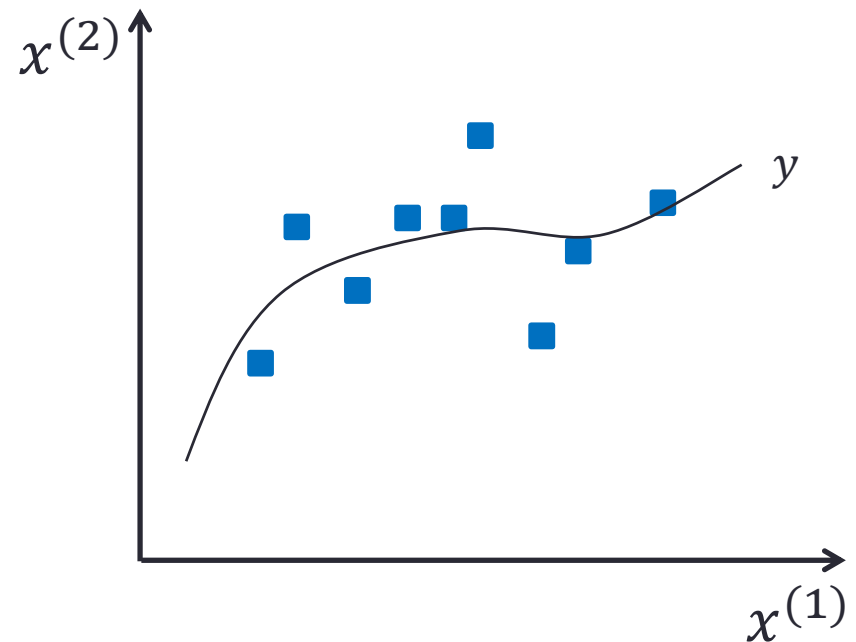
- Suppose we are given training data

$$(\mathbf{x}_i, y_i) \in \mathbb{R}^m \times \mathbb{R}$$

- Problem: find a hyperplane

$$y = f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

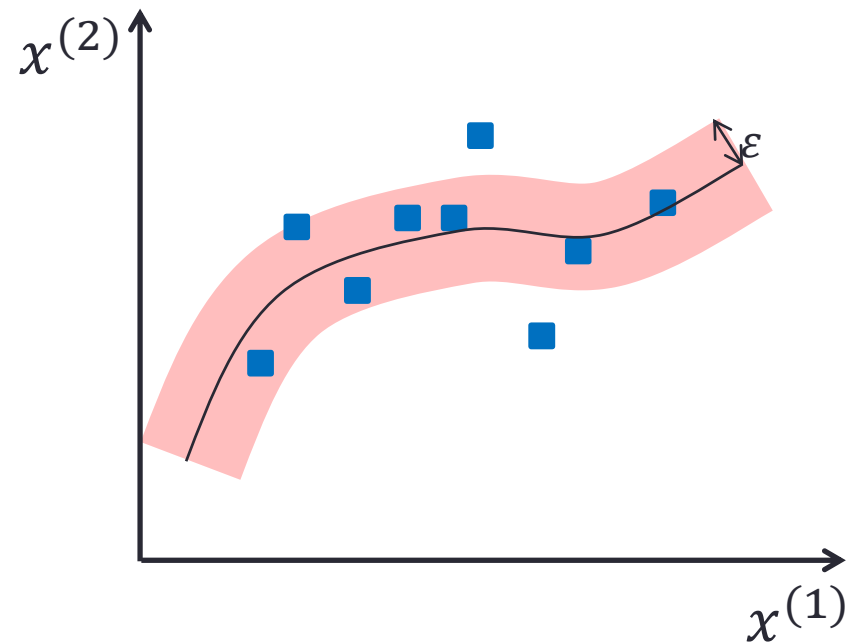
that predicts y_i for given \mathbf{x}_i



ε -insensitive Zone

- The bigger the ε , the fewer support vectors are selected
- Bigger ε -values results in more 'flat' estimates
- ε -insensitive zone:

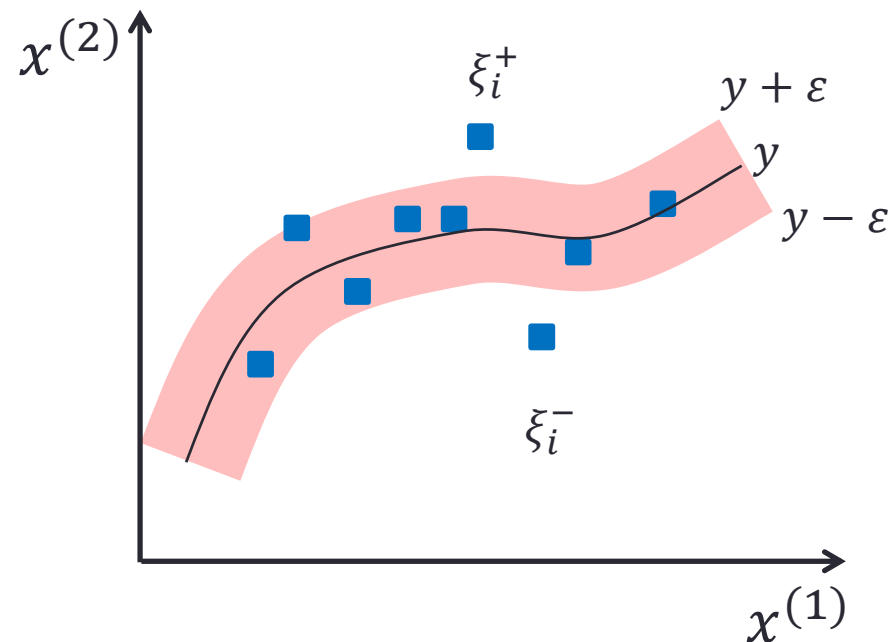
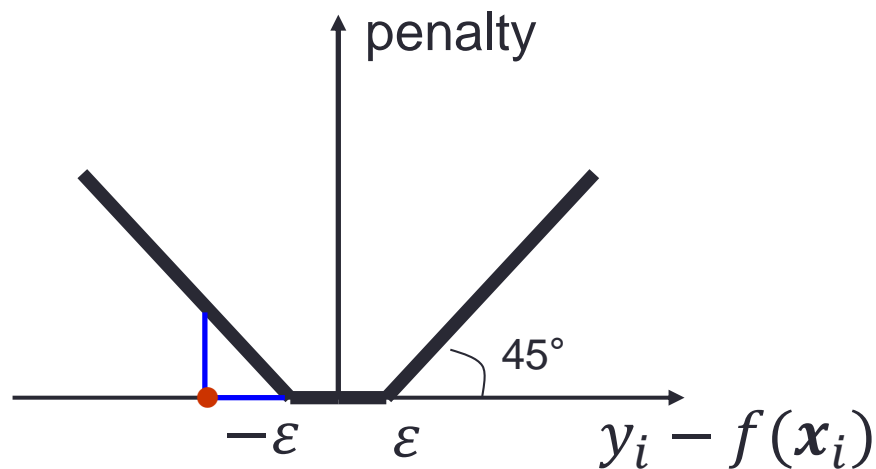
$$|y - f(\mathbf{x})| \leq \varepsilon \Rightarrow \begin{cases} y_i - \mathbf{w}^T \mathbf{x}_i - b \leq \varepsilon \\ \mathbf{w}^T \mathbf{x}_i - b - y_i \leq \varepsilon \end{cases}$$



Penalty Outside the ε -insensitive Zone

- Slack variables to allow points to lie outside the tube: ξ_i^+, ξ_i^-
- The loss function:

$$|\xi_i| = \begin{cases} 0, & \text{if } |\xi_i| \leq \varepsilon \\ |\xi_i| - \varepsilon, & \text{otherwise} \end{cases}$$



Problem Formulation

- SVM regression

$$\min_{\mathbf{w}} \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^n (\xi_i^+ + \xi_i^-)$$
$$\text{subject to } \begin{cases} y_i - \mathbf{w}^T \mathbf{x}_i - b \leq \varepsilon + \xi_i^+ \\ \mathbf{w}^T \mathbf{x}_i - b - y_i \leq \varepsilon + \xi_i^- \\ \xi_i^+, \xi_i^- > 0 \end{cases}$$

Lagrangian of SVM Regression

- Primal:

$$\begin{aligned} L_p = & \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{i=1}^n (\xi_i^+ + \xi_i^-) - \sum_{i=1}^n (\lambda_i^+ \xi_i^+ + \lambda_i^- \xi_i^-) \\ & - \sum_{i=1}^n \gamma_i^+ (\varepsilon + \xi_i^+ - y_i + \mathbf{w}^T \mathbf{x}_i + b) \\ & - \sum_{i=1}^n \gamma_i^- (\varepsilon + \xi_i^- + y_i - \mathbf{w}^T \mathbf{x}_i - b) \end{aligned}$$

Solving SVM Regression

- Derivate of L_p :

$$\frac{\partial L_p}{\partial \mathbf{w}} = 0 \Rightarrow \mathbf{w} = \sum_{i=1}^n (\gamma_i^+ - \gamma_i^-)$$

$$\frac{\partial L_p}{\partial b} = 0 \Rightarrow \sum_{i=1}^n (\gamma_i^- - \gamma_i^+) = 0$$

$$\frac{\partial L_p}{\partial \xi_i^+} = 0 \Rightarrow \sum_{i=1}^n (\lambda_i^+ + \gamma_i^+) = c$$

$$\frac{\partial L_p}{\partial \xi_i^-} = 0 \Rightarrow \sum_{i=1}^n (\lambda_i^- + \gamma_i^-) = c$$

Solution of SVM Regression

- Classifier:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b = \sum_{i=1}^n (\gamma_i^+ - \gamma_i^-) \mathbf{x}_i^T \mathbf{x} + b$$

- But what about b ?

Solving SVM Regression (Cont'd)

- Support vectors are points that lie on the boundary or outside the “tube” (Karush-Kuhn-Tucker conditions):

$$\gamma_i^+ (\varepsilon + \xi_i^+ - y_i + \mathbf{w}^T \mathbf{x}_i + b) = 0$$

$$\gamma_i^- (\varepsilon + \xi_i^- + y_i - \mathbf{w}^T \mathbf{x}_i - b) = 0$$

$$(c - \gamma_i^+) \xi_i^+ = 0$$

$$(c - \gamma_i^-) \xi_i^- = 0$$

$$\gamma_i^+ \gamma_i^- = 0$$

where vectors lie on the boundary: $\gamma_i^+ \neq 0$ or $\gamma_i^- \neq 0$

- For vectors inside the tube: $\gamma_i^+ = \gamma_i^- = 0$

Solving SVM Regression (Cont'd)

- This allows us to conclude that:

$$\begin{aligned} \varepsilon - y_i + \mathbf{w}^T \mathbf{x}_i + b &\geq 0 & \text{and} & \quad \xi_i^+ = 0 & \quad \text{if} & \quad \gamma_i^+ < c \\ \varepsilon - y_i + \mathbf{w}^T \mathbf{x}_i + b &\leq 0 & & & \quad \text{if} & \quad \gamma_i^+ > 0 \end{aligned}$$

- The range of b :

$$\begin{aligned} &\max\{-\varepsilon + y_i - \mathbf{w}^T \mathbf{x}_i \mid \gamma_i^+ < c \text{ or } \gamma_i^- > 0\} \\ &\leq b \leq \\ &\min\{-\varepsilon + y_i - \mathbf{w}^T \mathbf{x}_i \mid \gamma_i^+ > 0 \text{ or } \gamma_i^- < c\} \end{aligned}$$

Compared to Least-squares Regression

- Basic idea is the same as in least-squares regression – want to minimize error
- Difference:
 - Ignore errors smaller than ε and use absolute error instead of squared error
 - Simultaneously aim to maximize flatness of function
- User-specified parameter ε defines the “tube”
- If there are tubes that enclose all the training points, the flattest of them is used
- SVM requires trade-off between error and flatness

Further Reading

- C. J. C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition
- D. Klein, Lagrange Multipliers without Permanent Scarring
- A. J. Smola and B. Scholkopf, A Tutorial on Support Vector Regression

Acknowledgement

- Especially thank Dr. Andrew W. Moore for sharing his valuable teaching material in this course