INTRODUCTORY APPLIED MACHINE LEARNING

Yan-Fu Kuo

Dept. of Bio-industrial Mechatronics Engineering

National Taiwan University

Today:

- Support vector machine

Outline

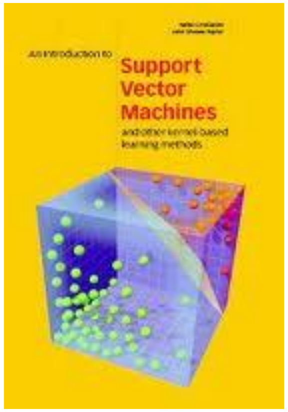
- History of SVM
- Linear SVM
- Lagrange multiplier
- Soft margin SVM
- Kernel tricks
- SVM regression

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 in1963
- 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 $\boldsymbol{\mathcal{X}} = [\boldsymbol{x}_i] \in \Re^n$, $i = 1 \dots m$, samples
- Each of the samples is associated with a class label $\boldsymbol{\mathcal{Y}} = [y_i] \in \Re$
- We want to learn the mapping $\boldsymbol{\mathcal{X}} \mapsto \boldsymbol{\mathcal{Y}}$

Example

- Suppose we have 100 photos ($\boldsymbol{x}_i$) of oranges and bananas, i.e., $i = 1 \dots 100$
- We digitize them into 50 x 50 pixel images, i.e., $\boldsymbol{x}_i \in \Re^n$ where $n = 2500$
- 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: $\boldsymbol{\mathcal{X}}$ / Output set: $\boldsymbol{\mathcal{Y}}$
- Training data points $(\boldsymbol{x}_1, y_1) \dots (\boldsymbol{x}_m, y_m)$
- Problem: given a $\boldsymbol{x}_i \in \boldsymbol{\mathcal{X}}$, find a suitable mapping (model) such that it gives $y_i \in \boldsymbol{\mathcal{Y}}$
- 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: $\hat{y} = f(\boldsymbol{x}, \alpha)$, 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:

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

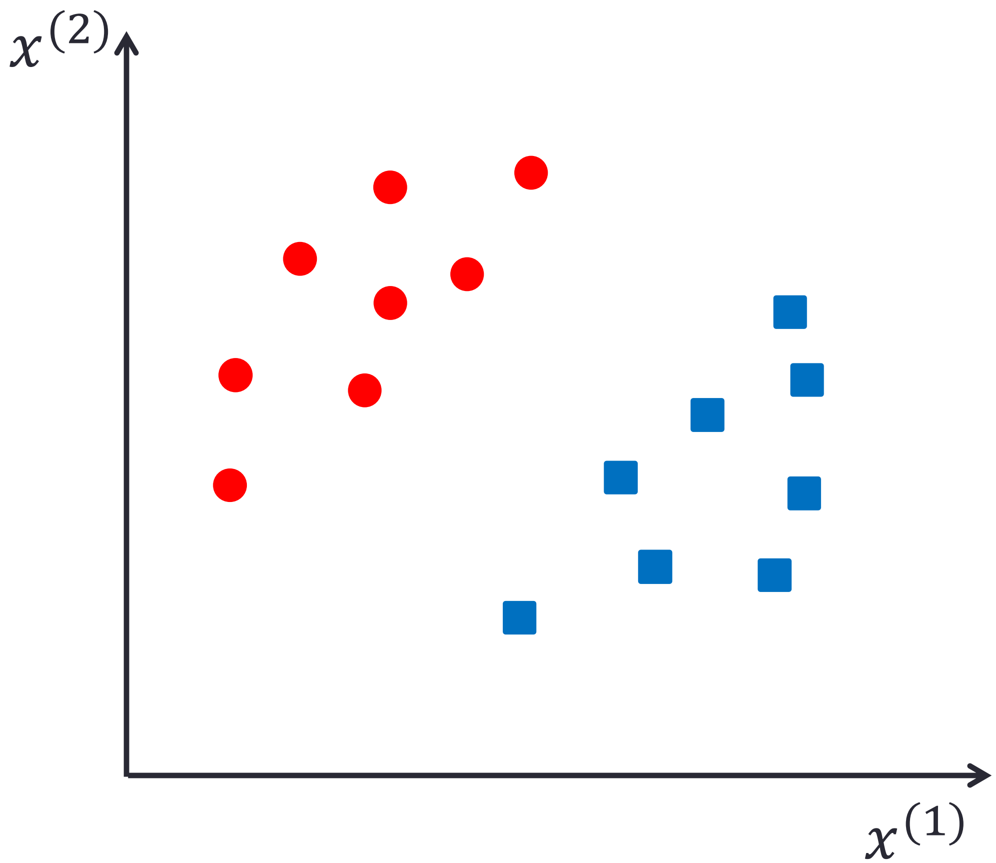
- True error:

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

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

Linearly Separable Case

- Simplify the problem to 2-dimensional, i.e., $\boldsymbol{x}_i \in \Re^2$



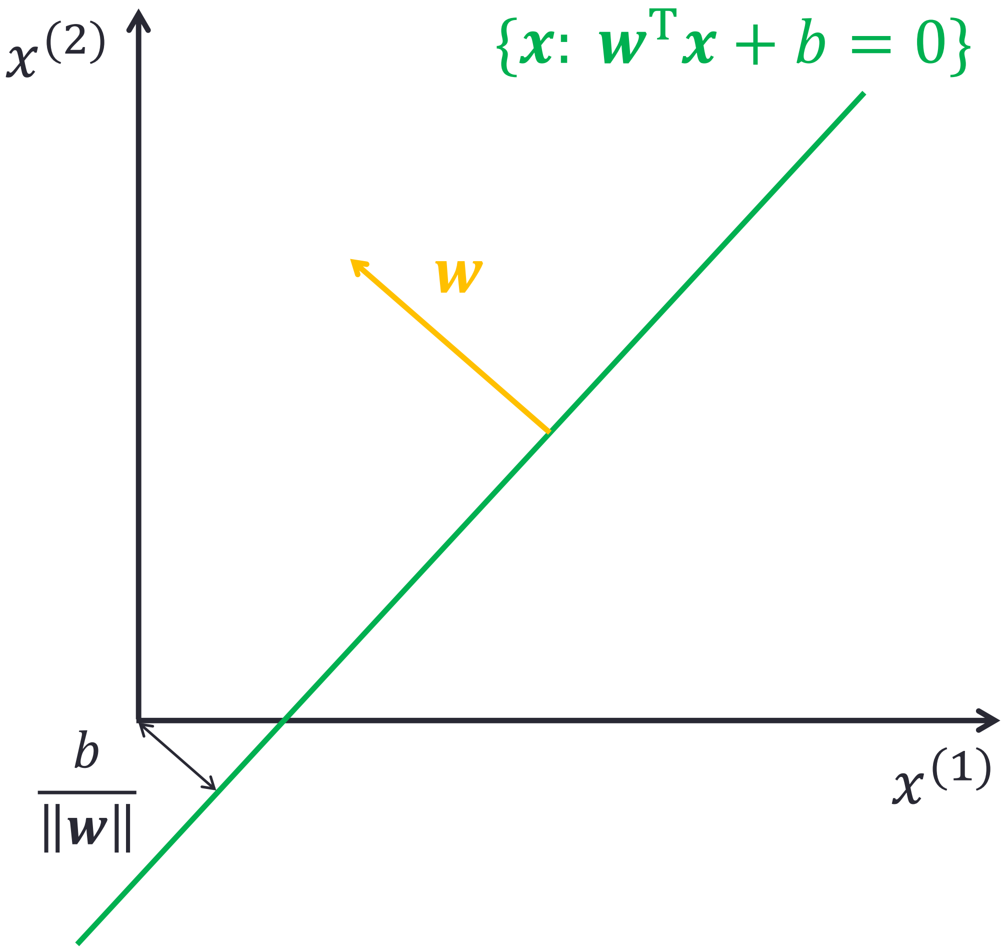
Review of High School Linear Algebra

- A hyperplane (line) can be represented as:

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

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

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

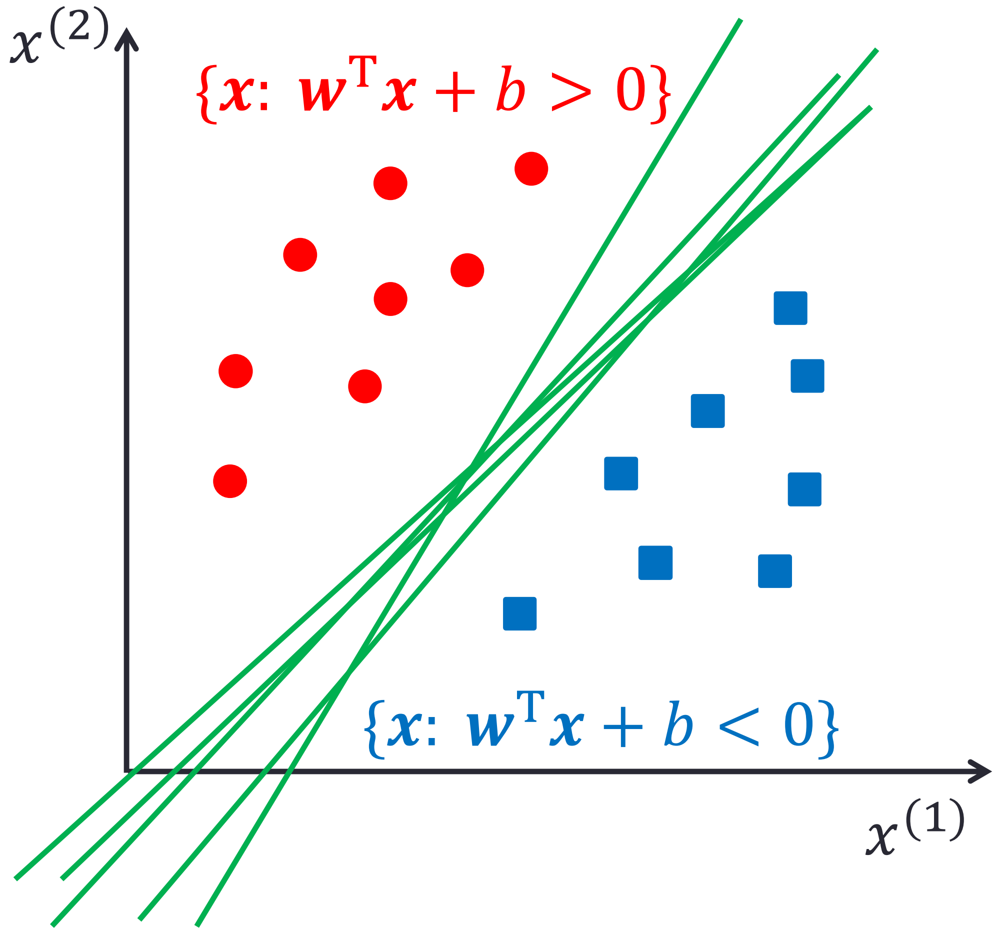


Linearly Separable Case

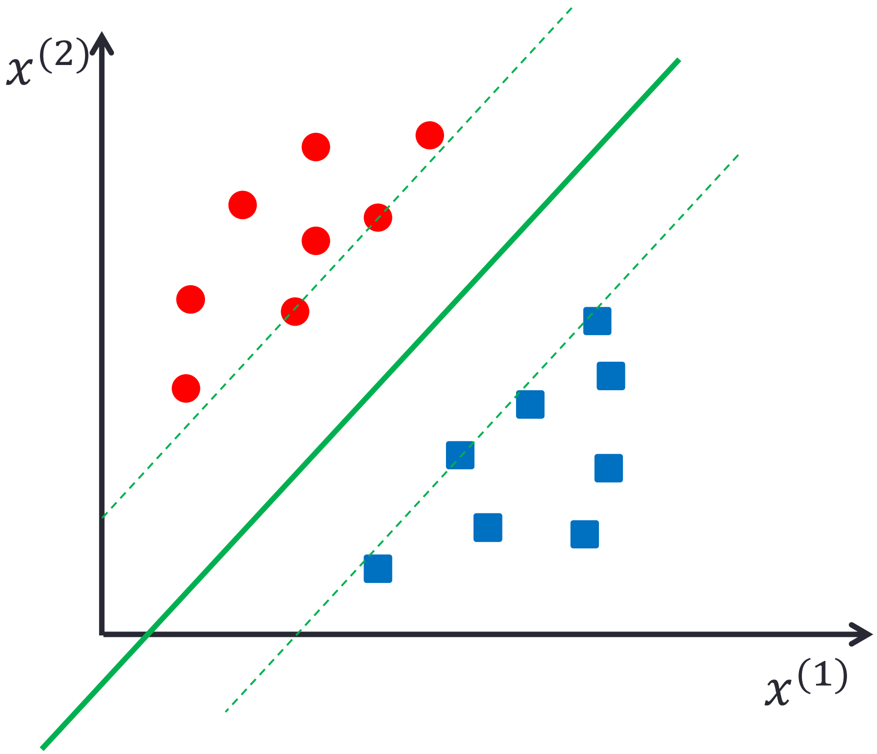
- Necessary condition of a separating hyperplane:

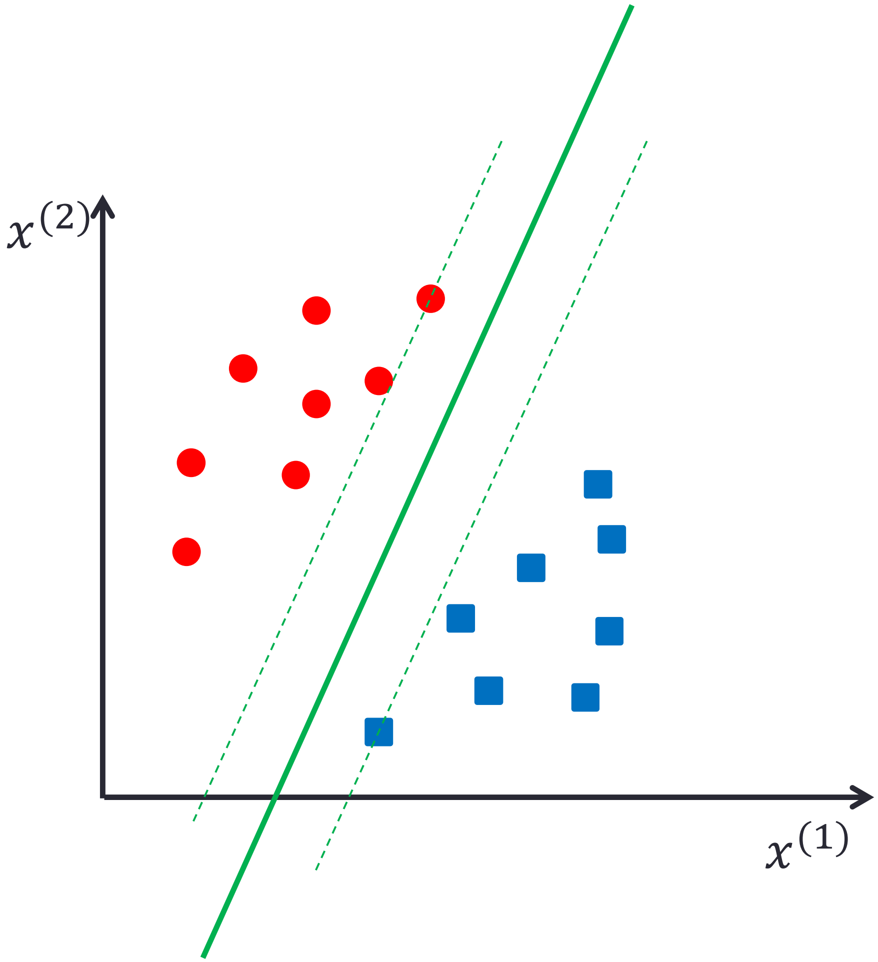
$$f(\boldsymbol{x}_i) = \operatorname{sgn}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) = \begin{cases} +1, \forall \text{ red} \\ -1, \forall \text{ blue} \end{cases}$$

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



What Hyperplane Is the Best?

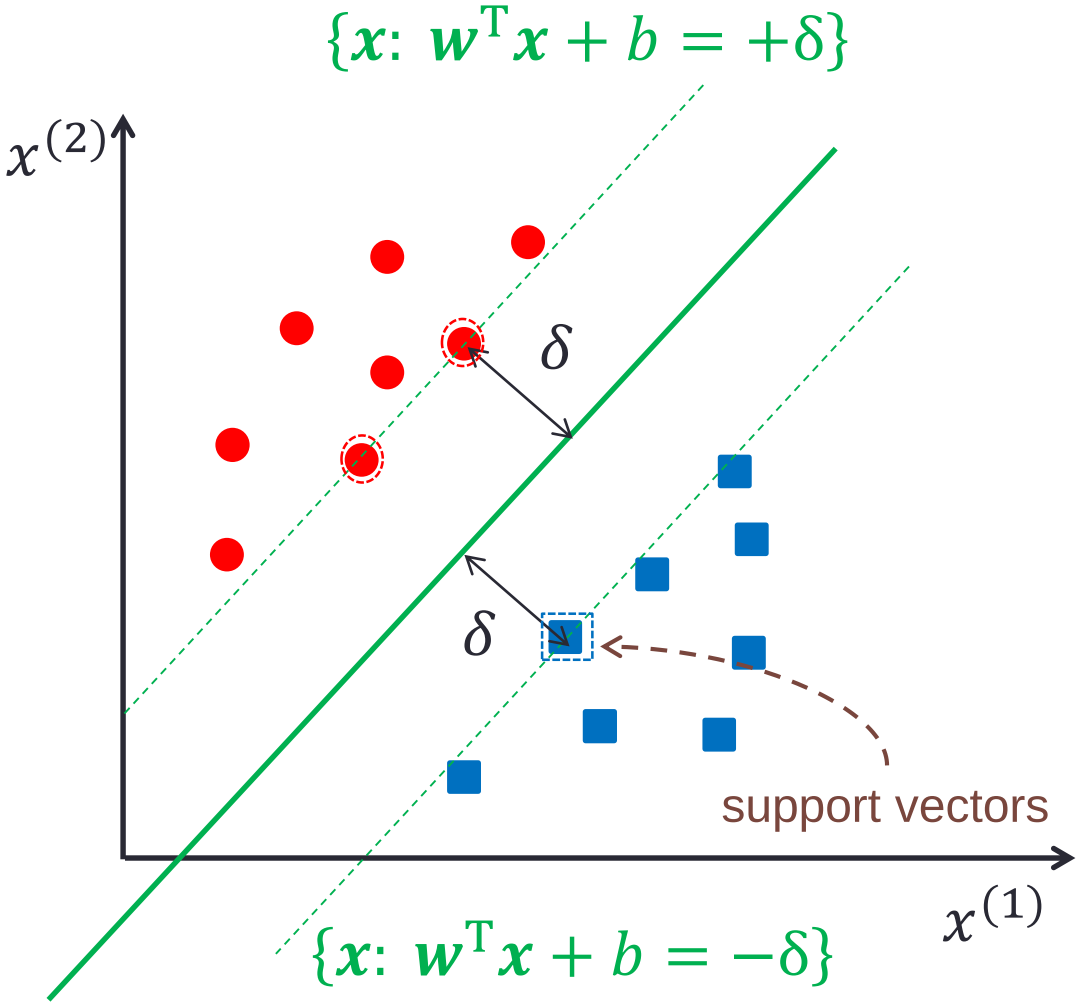




Optimal Separating Hyperplane

- Optimal separating hyperplane must <u>maximizes</u> the <u>minimum</u> distance from any sample $\boldsymbol{x}_i$
- Suppose the minimum distance is δ, the optimal separating hyperplane must satisfy

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



Why Max-min?

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

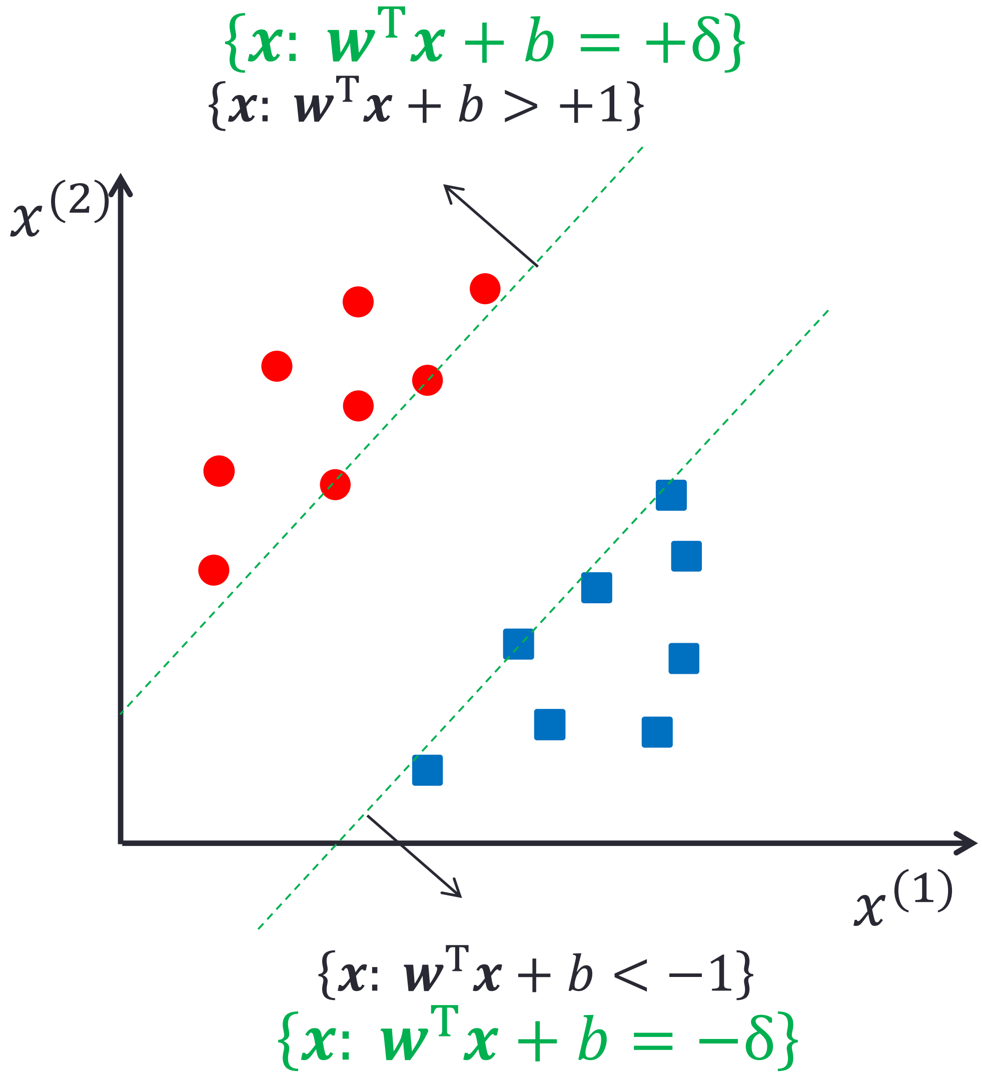
Over-parameterized Constraint

- Constraint for the optimal separating hyperplane

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

- However, this can be equally expressed by all sets $(\alpha\boldsymbol{w}, \alpha b, \alpha\delta)$ for any $\alpha \in \Re^+$
- Canonical constraint

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



Maximum Margin Classifier

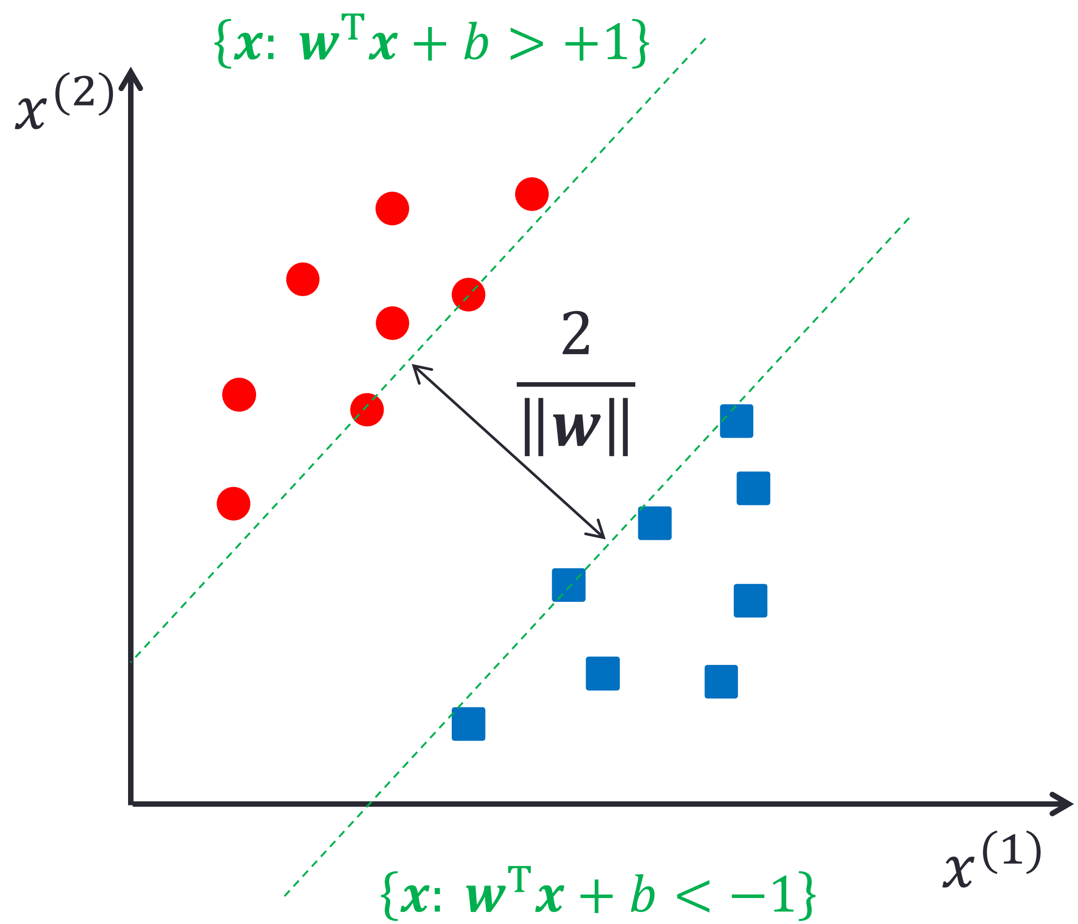
- Maximize the distance between two support hyperplanes:

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

- Subject to the constraint:

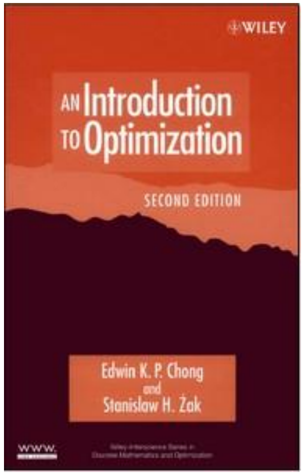
$$y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 1, \forall i$$

- Alter the cost to $\frac{1}{2}||\boldsymbol{w}||^{2}$
- 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 $\frac{1}{2}||\boldsymbol{w}||^2$, subject to $y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1, \forall i$

- Lagrangian:

$$L(\boldsymbol{w}, b, \lambda_i) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{m} \lambda_i\left[y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1\right]$$

$$= \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{m} \lambda_i y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) + \sum_{i=1}^{m} \lambda_i \quad \ldots\text{(a)}$$

Karush-Kuhn-Tucker (KKT) Condition

1. $$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{m} \lambda_i y_i \boldsymbol{x}_i = \boldsymbol{0} \qquad \text{…(b)}$$

2. $$\frac{\partial L}{\partial b} = \sum_{i=1}^{m} \lambda_i y_i = 0 \qquad \text{…(c)}$$

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

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

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

$\lambda_i \neq 0$

$\Downarrow$

$\boldsymbol{x}_i$ support vector!

$\lambda_i = 0$

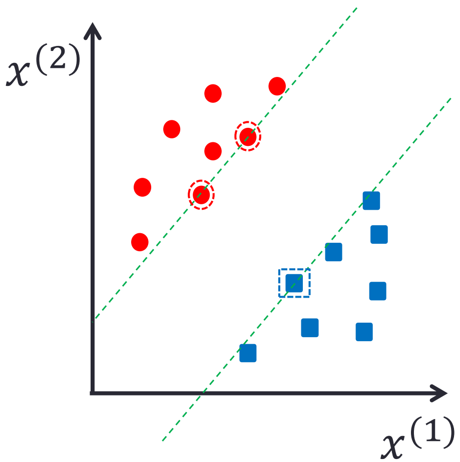
$\Downarrow$

$\boldsymbol{x}_i$ not support vector

$\boldsymbol{w}$ of the Support Vector Machine

- $\boldsymbol{w}$ is determined by Eq. 1:

$$\boldsymbol{w} = \sum_{i=1}^{m} \lambda_i y_i \boldsymbol{x}_i$$



- Remember the condition:

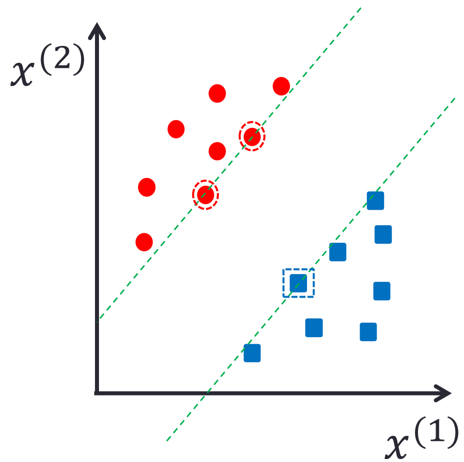
$\boldsymbol{x}_i$ not support vector $\Rightarrow \lambda_i = 0$

(<u>$\boldsymbol{w}$ is only determined by the support vectors</u>)

b of the Support Vector Machine

- b can be determined by Eq. 5:

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



- Remember the condition:

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

  (<u>b is only determined by the support vectors</u>)

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

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

Linear SVM Solution

- The solution has the form:

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

Note:
For a SV $\boldsymbol{x}_i$ with label $y_i = +1$

- The classifier will have the form:

$$f(\boldsymbol{x}) = \operatorname{sgn}(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b) = \operatorname{sgn}\left(\sum_{i=1}^{m} \overbrace{\lambda_i y_i}^{\text{Weight}} \boldsymbol{x}_i{}^{\mathrm{T}} \boldsymbol{x} + \overbrace{(1 - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i)}^{\text{Bias}} \right)$$

- Note that $\lambda_i y_i$ is the weight

(Wolfe) Dual Form

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

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

subject to

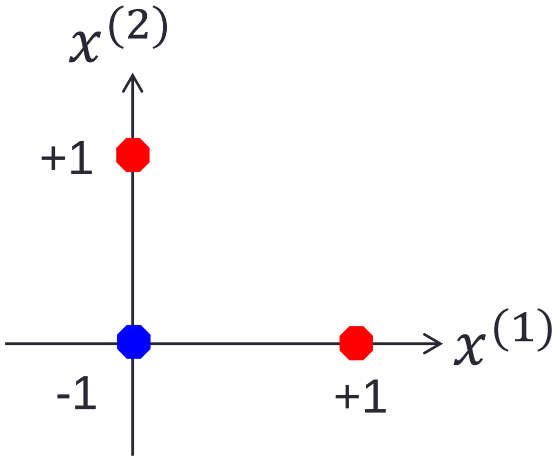
$$\begin{cases} \sum_{i=1}^{m} \lambda_i y_i = 0 \\ \lambda_i \geq 0 \ \forall i \end{cases}$$

- SVM parameters can be determined using the dual form

Example: Analytically Solving SVM

- Data:

Input x	Output y
$\boldsymbol{x}_1 = [0\ 0]^{\mathrm{T}}$	$y_1 = -1$
$\boldsymbol{x}_2 = [1\ 0]^{\mathrm{T}}$	$y_2 = +1$
$\boldsymbol{x}_3 = [0\ 1]^{\mathrm{T}}$	$y_3 = +1$



- Strategies:
  - Apply the duel form (a constraint optimization problem)
  - Introduce another Lagrange multiplier α

Analytically Solving SVM

Input x
$\boldsymbol{x}_1 = [0\ 0]^{\mathrm{T}}$
$\boldsymbol{x}_2 = [1\ 0]^{\mathrm{T}}$
$\boldsymbol{x}_3 = [0\ 1]^{\mathrm{T}}$

$$L_n(\lambda_i, \alpha) = f(\lambda_i) - \alpha g(\lambda_i)$$

$$= \sum_{i=1}^{3} \lambda_i - \frac{1}{2} \sum_{i=1}^{3} \sum_{j=1}^{3} \lambda_i \lambda_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j - \alpha \sum_{i=1}^{3} \lambda_i y_i$$

$$= \lambda_1 + \lambda_2 + \lambda_3$$

$$-\frac{1}{2}\Big(\cancel{\lambda_1 \lambda_1 y_1 y_1 \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{x}_1} + \lambda_2 \lambda_2 y_2 y_2 \boldsymbol{x}_2^{\mathrm{T}} \boldsymbol{x}_2 + \lambda_3 \lambda_3 y_3 y_3 \boldsymbol{x}_3^{\mathrm{T}} \boldsymbol{x}_3$$

$$+\boldsymbol{2}\cancel{\lambda_1 \lambda_2 y_1 y_2 \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{x}_2} + 2\cancel{\lambda_1 \lambda_3 y_1 y_3 \boldsymbol{x}_1^{\mathrm{T}} \boldsymbol{x}_3} + 2\cancel{\lambda_2 \lambda_3 y_2 y_3 \boldsymbol{x}_2^{\mathrm{T}} \boldsymbol{x}_3}\Big)$$

$$-\alpha(\lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3)$$

Analytically Solving SVM (Cont'd)

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

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

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

Solving $\boldsymbol{w}$ and b

Input x	Output y
$\boldsymbol{x}_1 = [0\ 0]^{\mathrm{T}}$	$y_1 = -1$
$\boldsymbol{x}_2 = [1\ 0]^{\mathrm{T}}$	$y_2 = +1$
$\boldsymbol{x}_3 = [0\ 1]^{\mathrm{T}}$	$y_3 = +1$

- Now solve $\boldsymbol{w}$:

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

- Solve b:

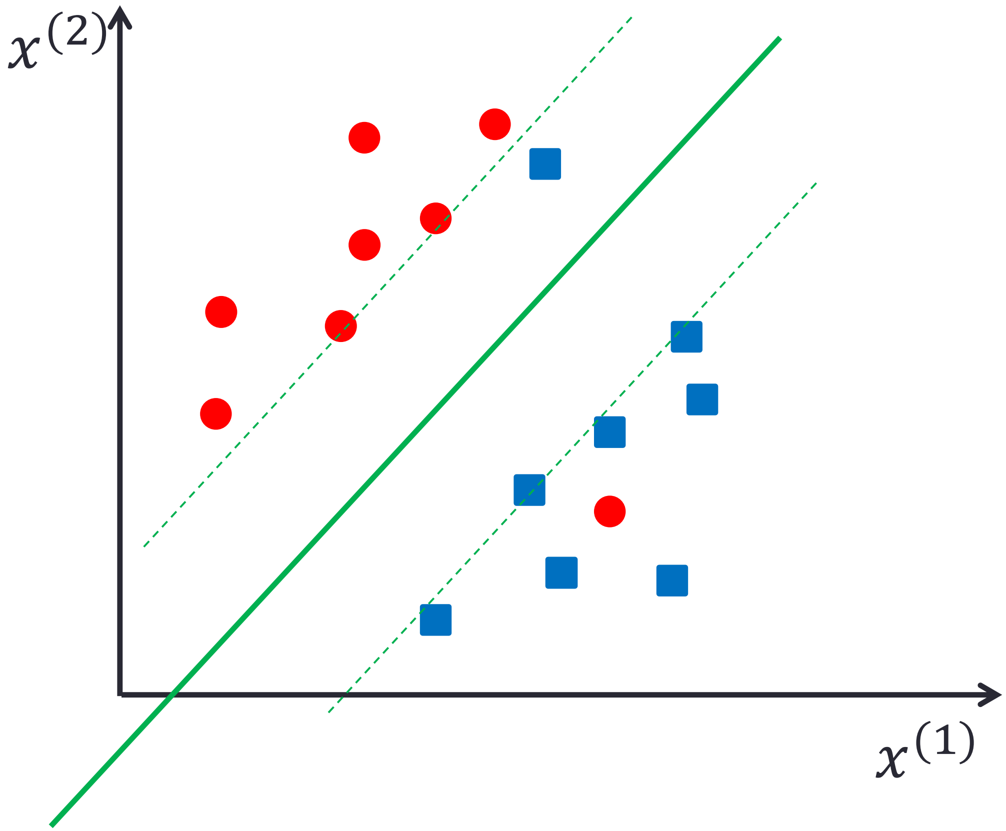
$$b = 1 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i = 1 - [2\ 2]\begin{bmatrix}1\\0\end{bmatrix} = -1$$

- SVM:

$$f(\boldsymbol{x}) = \mathrm{sgn}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + \ b\right) = \mathrm{sgn}([2\ 2]\,\boldsymbol{x} - 1)$$

Data with Noise

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

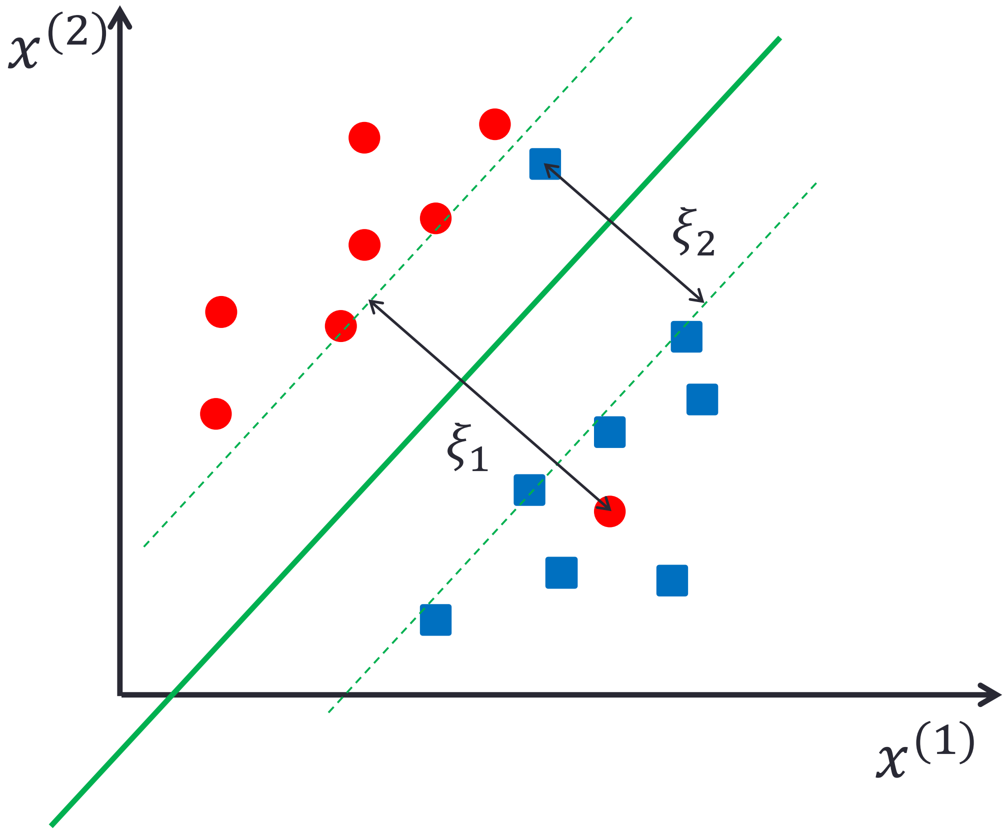


Slack Variable

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

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

where $c > 0$



Soft Margin SVM Problem Statement

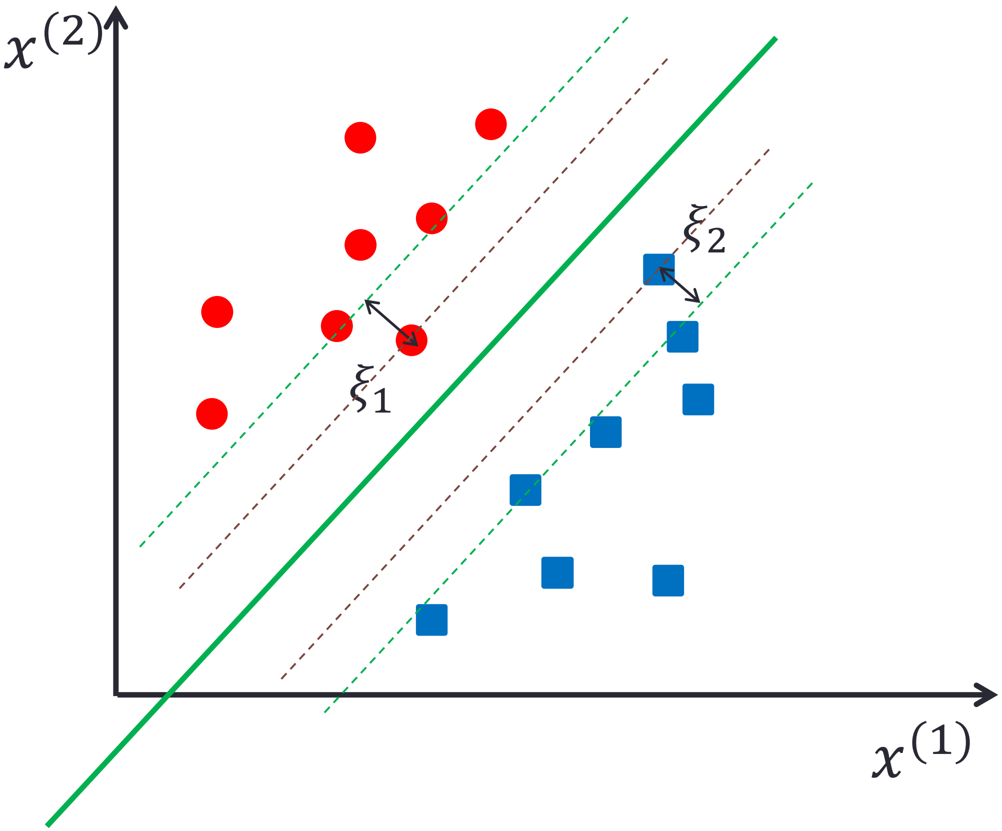
- Cost function:

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

subject to

$$y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 1 - \xi_i \quad \forall i$$

$$\xi_i \geq 0 \quad \forall i$$



Lagrangian of Soft-margin SVM

- One cost function and two constraints
- Lagrangian:

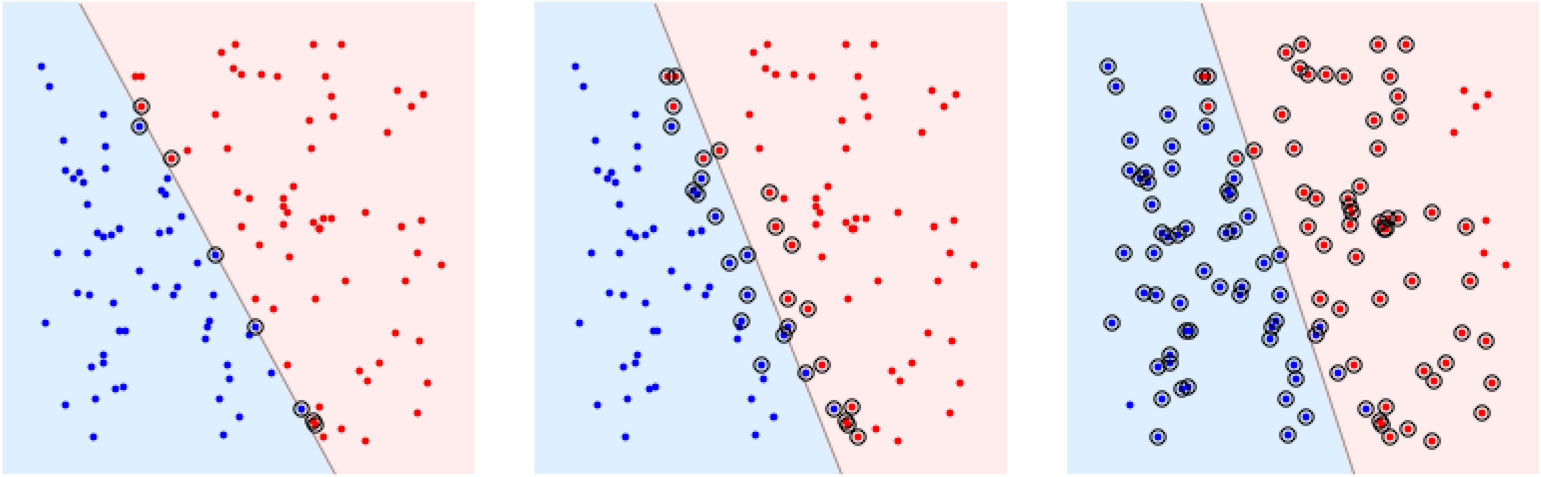
$$L(\boldsymbol{w}, b, \lambda_i, \xi_i, \gamma_i)$$
$$= \frac{1}{2}\|\boldsymbol{w}\|^2 + c\sum_{i=1}^{r}\xi_i - \sum_{i=1}^{m}\lambda_i\left[y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{r}\gamma_i\xi_i$$

How Does c Impact the Margin?

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

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

c = 1000 c = 10 c = 0.1



Hard Margin v.s. Soft Margin SVM

- Hard margin SVM formulation:

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

$$y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 1, \forall i$$

- Soft margin SVM formulation:

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

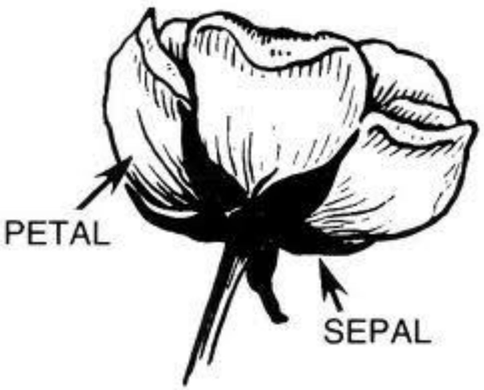
$$y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 1 - \xi_i \;\; \forall i$$

$$\xi_i \geq 0 \;\; \forall i$$

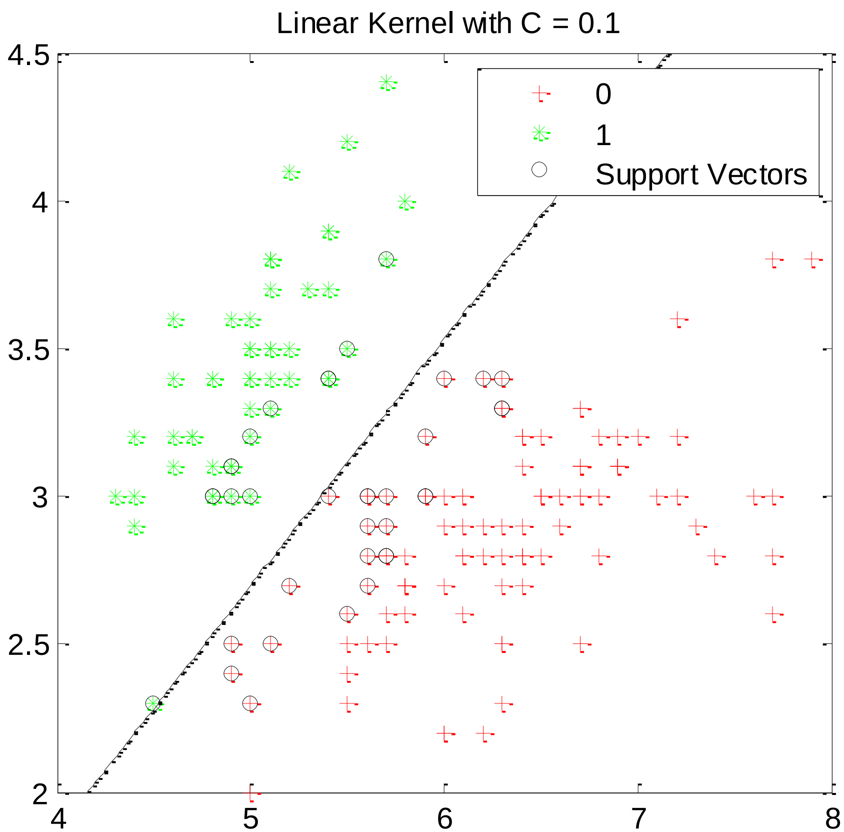
- Parameter c can be viewed as a way to control overfitting

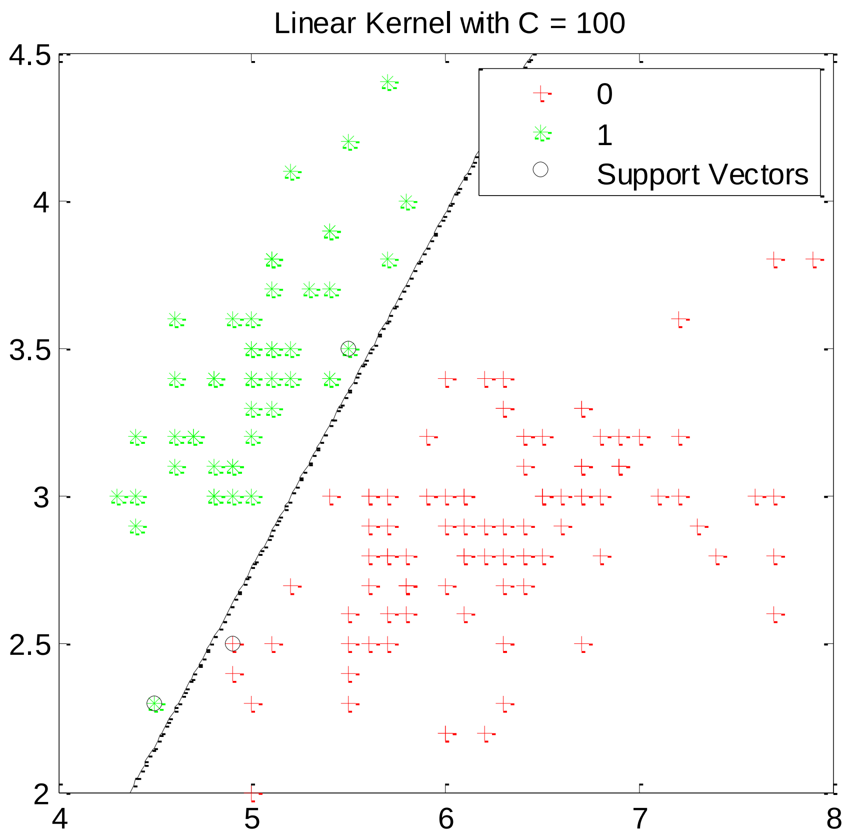
Example – Fisher's Iris Data

- 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





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

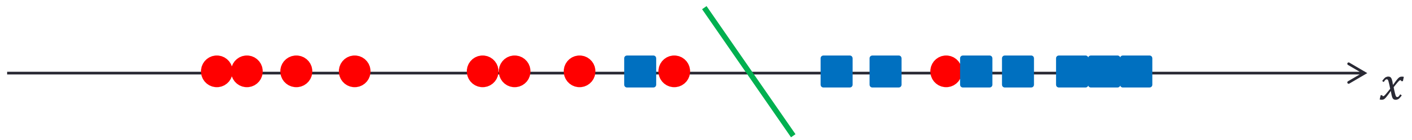
Review of Linear SVM

- The classifier is a separating hyperplane
- Most “important” data points are support vectors – they define the hyperplane ($\boldsymbol{w}$ and b)
- Quadratic optimization algorithms can identify which data points $\boldsymbol{x}_i$ 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 $\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}$, i.e.,

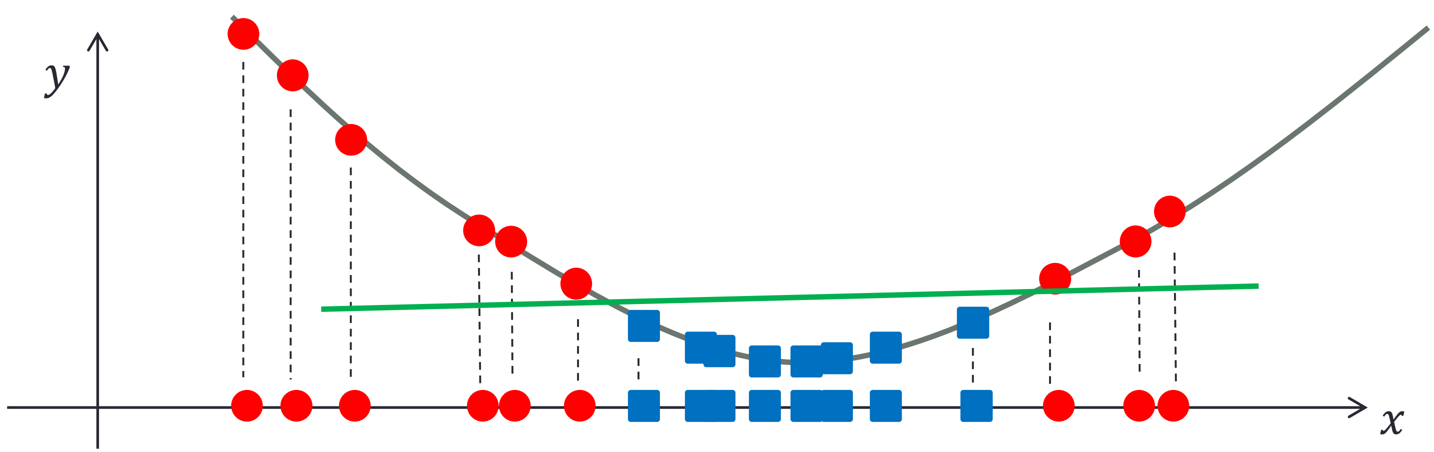
$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b\right) = \operatorname{sgn}\left(\sum_{i=1}^{m} \lambda_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} + b\right)$$

Non-linear SVM

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



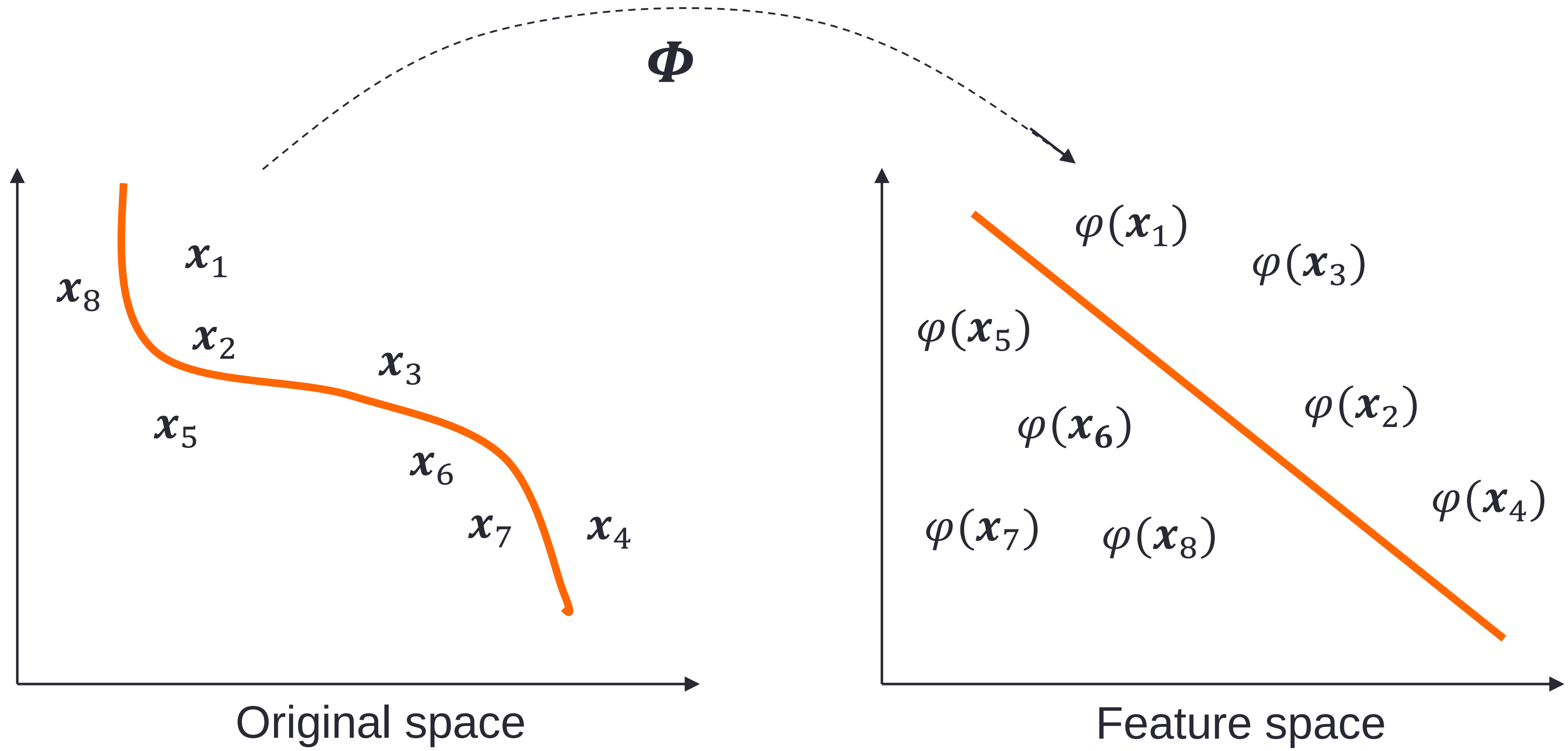
- What if the dataset is not just "noisy"?



- Strategy – mapping data to a higher-dimensional space

Feature Space

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

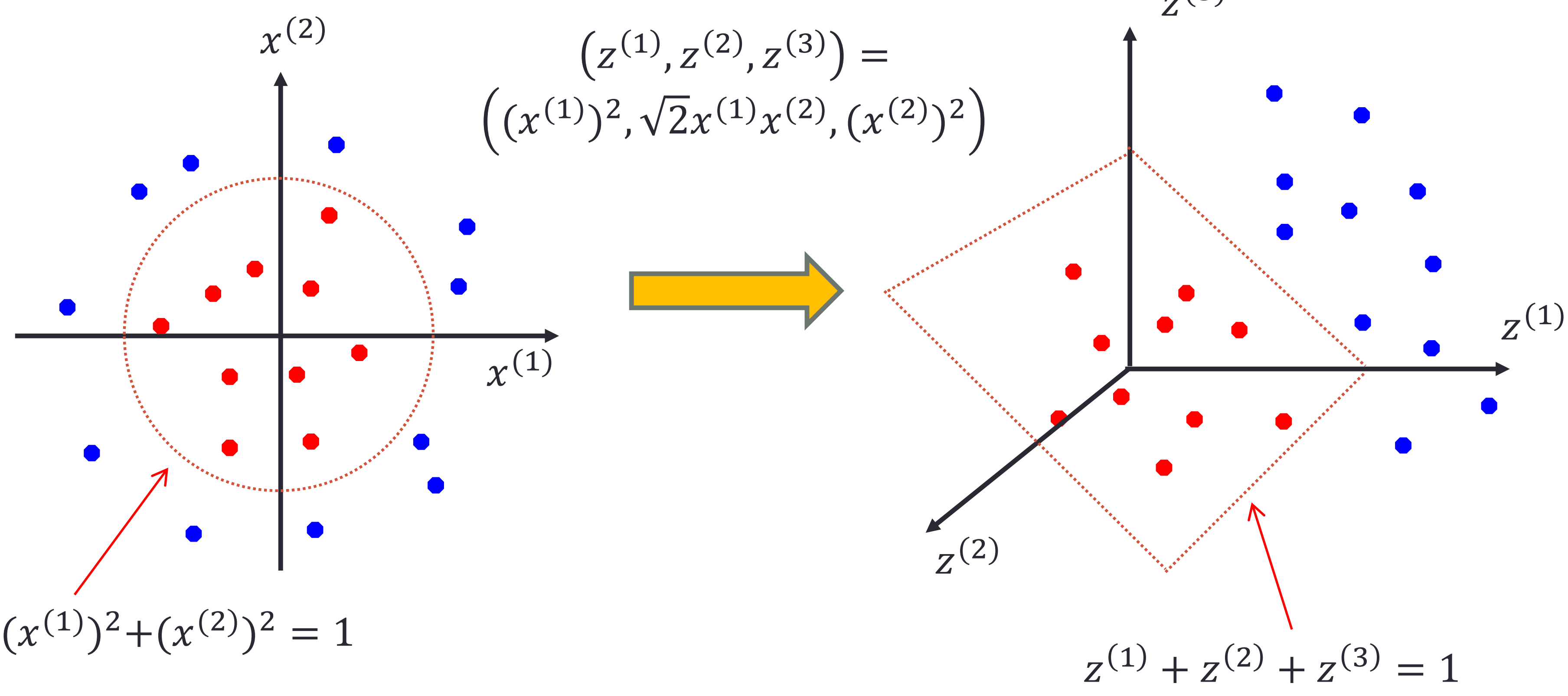


Example Kernel Function

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

$$\boldsymbol{\Phi}: \boldsymbol{x}_i \longrightarrow \varphi(\boldsymbol{x}_i) = \boldsymbol{z}_i$$

$$\left(z^{(1)}, z^{(2)}, z^{(3)}\right) = \left((x^{(1)})^2, \sqrt{2}x^{(1)}x^{(2)}, (x^{(2)})^2\right)$$



Kernel SVM

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

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

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

Typical Kernel Function

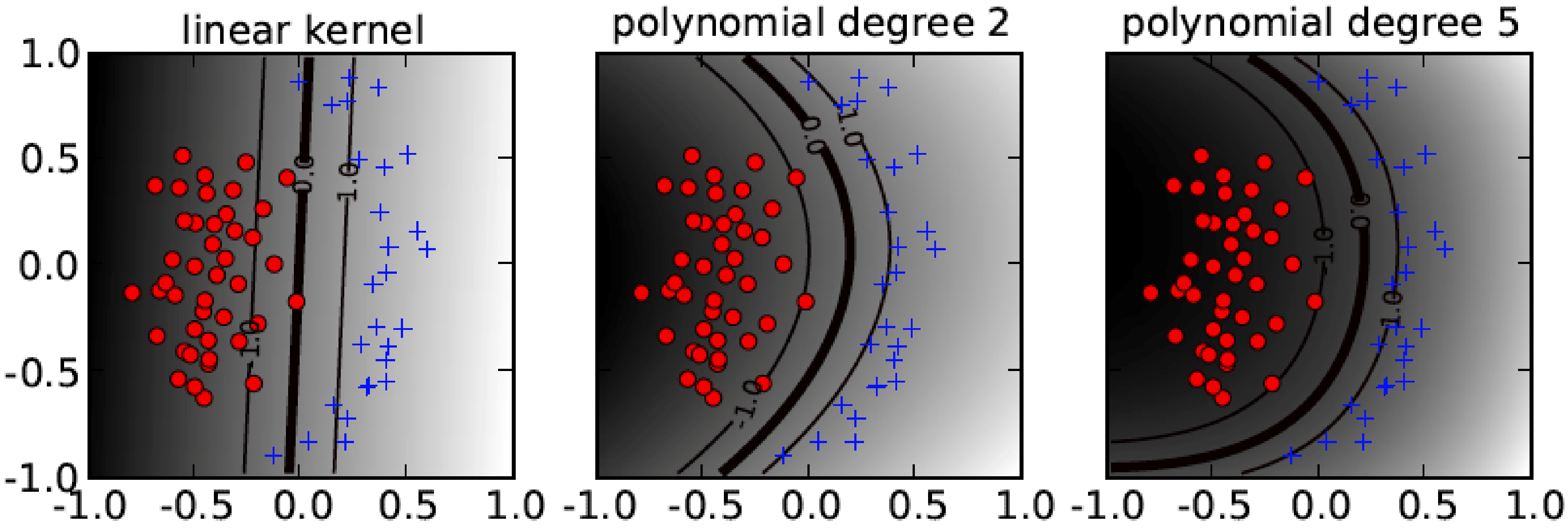
- Linear: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$
- Polynomial of power p: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j)^p$
- Gaussian (radial-basis function network):

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

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

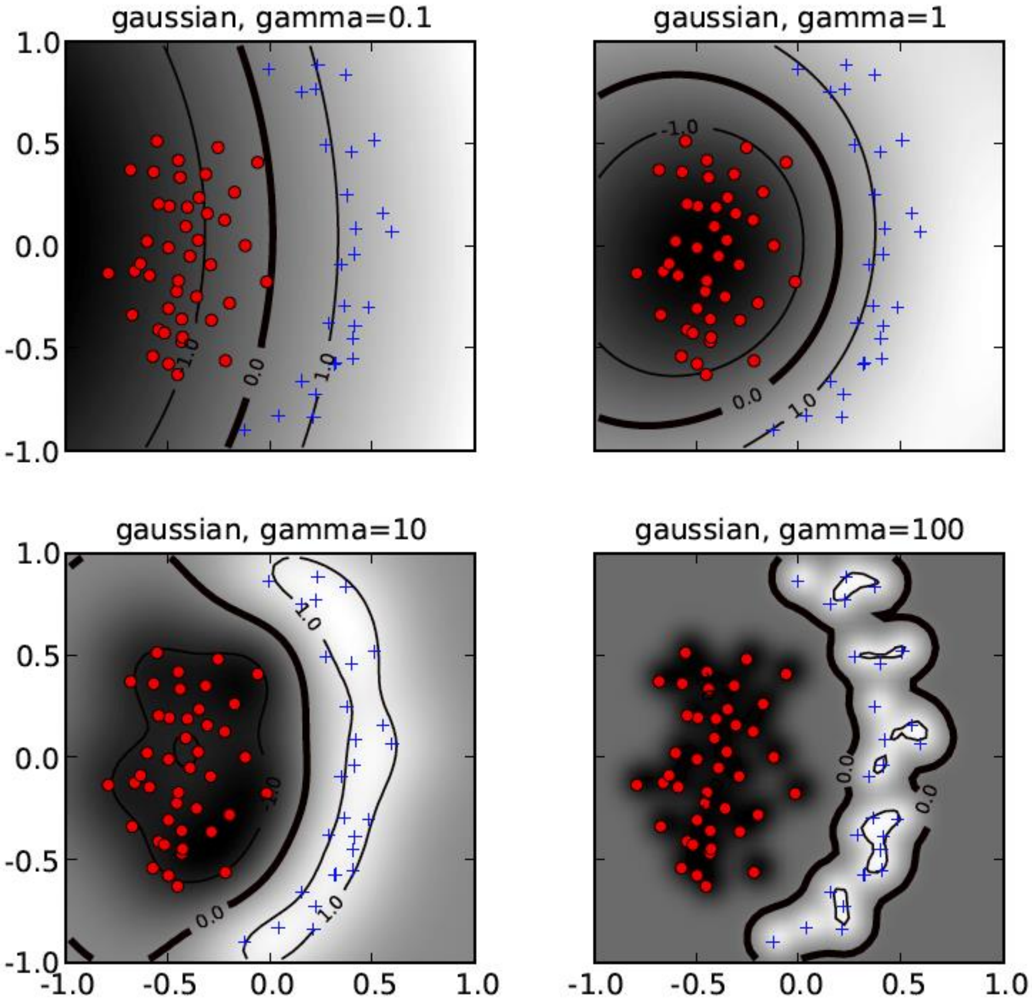
Polynomial Kernel

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(1 + \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j\right)^p$$



RBF Kernel

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



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

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

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j)^2 = \left(1 + [x_i{}^{(1)} \;\; x_i{}^{(2)}]\begin{bmatrix} x_j{}^{(1)} \\ x_j{}^{(2)} \end{bmatrix}\right)^2$$

$$= \left[1 \quad x_i{}^{(1)2} \quad \sqrt{2}x_i{}^{(1)}x_i{}^{(2)} \quad x_i{}^{(2)2} \quad \sqrt{2}x_i{}^{(1)} \quad \sqrt{2}x_i{}^{(2)}\right]^{\mathrm{T}}$$
$$\left[1 \quad x_j{}^{(1)2} \quad \sqrt{2}x_j{}^{(1)}x_j{}^{(2)} \quad x_j{}^{(2)2} \quad \sqrt{2}x_j{}^{(1)} \quad \sqrt{2}x_j{}^{(2)}\right]$$

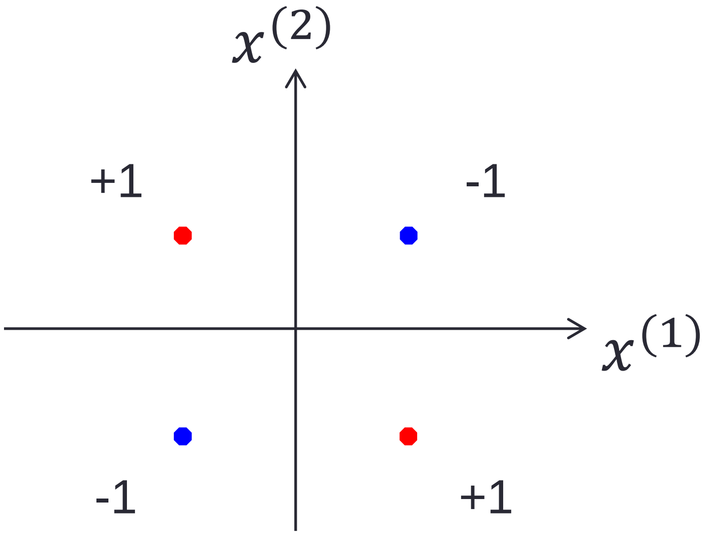
$$= \varphi(\boldsymbol{x}_i)^{\mathrm{T}}\varphi(\boldsymbol{x}_j)$$

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

Example: Kernel SVM

- XOR:

Input x	Output y
$\boldsymbol{x}_1 = [+1 \quad +1]^{\mathrm{T}}$	$y_1 = -1$
$\boldsymbol{x}_2 = [+1 \quad -1]^{\mathrm{T}}$	$y_2 = +1$
$\boldsymbol{x}_3 = [-1 \quad +1]^{\mathrm{T}}$	$y_3 = +1$
$\boldsymbol{x}_4 = [-1 \quad -1]^{\mathrm{T}}$	$y_4 = -1$



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

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \left(1 + \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j\right)^2$$

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

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

XOR Problem Cost Function

- Dual form Lagrangian:

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

$$= \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 - \frac{1}{2}(9(\lambda_1)^2 - 2\lambda_1\lambda_2 - 2\lambda_1\lambda_3 + 2\lambda_1\lambda_4$$

$$+9(\lambda_2)^2 + 2\lambda_2\lambda_3 - 2\lambda_2\lambda_4 + 9(\lambda_3)^2 - 2\lambda_3\lambda_4 + 9(\lambda_4)^2)$$

- 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

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m} \lambda_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right)$$

$$= \operatorname{sgn}\left(\sum_{i=1}^{m} \lambda_i y_i \varphi(\boldsymbol{x}_i)^{\mathrm{T}} \varphi(\boldsymbol{x}) + b\right)$$

$$= \operatorname{sgn}\left(\left(\sum_{i=1}^{m} \lambda_i y_i \varphi(\boldsymbol{x}_i)\right)^{\mathrm{T}} \varphi(\boldsymbol{x}) + b\right)$$

XOR Problem Solution

$$\sum_{i=1}^{m} \lambda_i y_i \varphi(\boldsymbol{x}_i) = \frac{1}{8}\left(-\varphi(\boldsymbol{x}_1) + \varphi(\boldsymbol{x}_2) + \varphi(\boldsymbol{x}_3) - \varphi(\boldsymbol{x}_4)\right)$$

$$= \frac{1}{8}\left(-\begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ \sqrt{2} \\ \sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ \sqrt{2} \\ -\sqrt{2} \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ -\sqrt{2} \\ 1 \\ -\sqrt{2} \\ \sqrt{2} \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ \sqrt{2} \\ 1 \\ -\sqrt{2} \\ -\sqrt{2} \end{bmatrix} \right) = \begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

XOR Problem Solution (Cont'd)

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\left(\sum_{i=1}^{m} \lambda_i y_i \varphi(\boldsymbol{x}_i)\right)^{\mathrm{T}} \varphi(\boldsymbol{x}) + b\right)$$

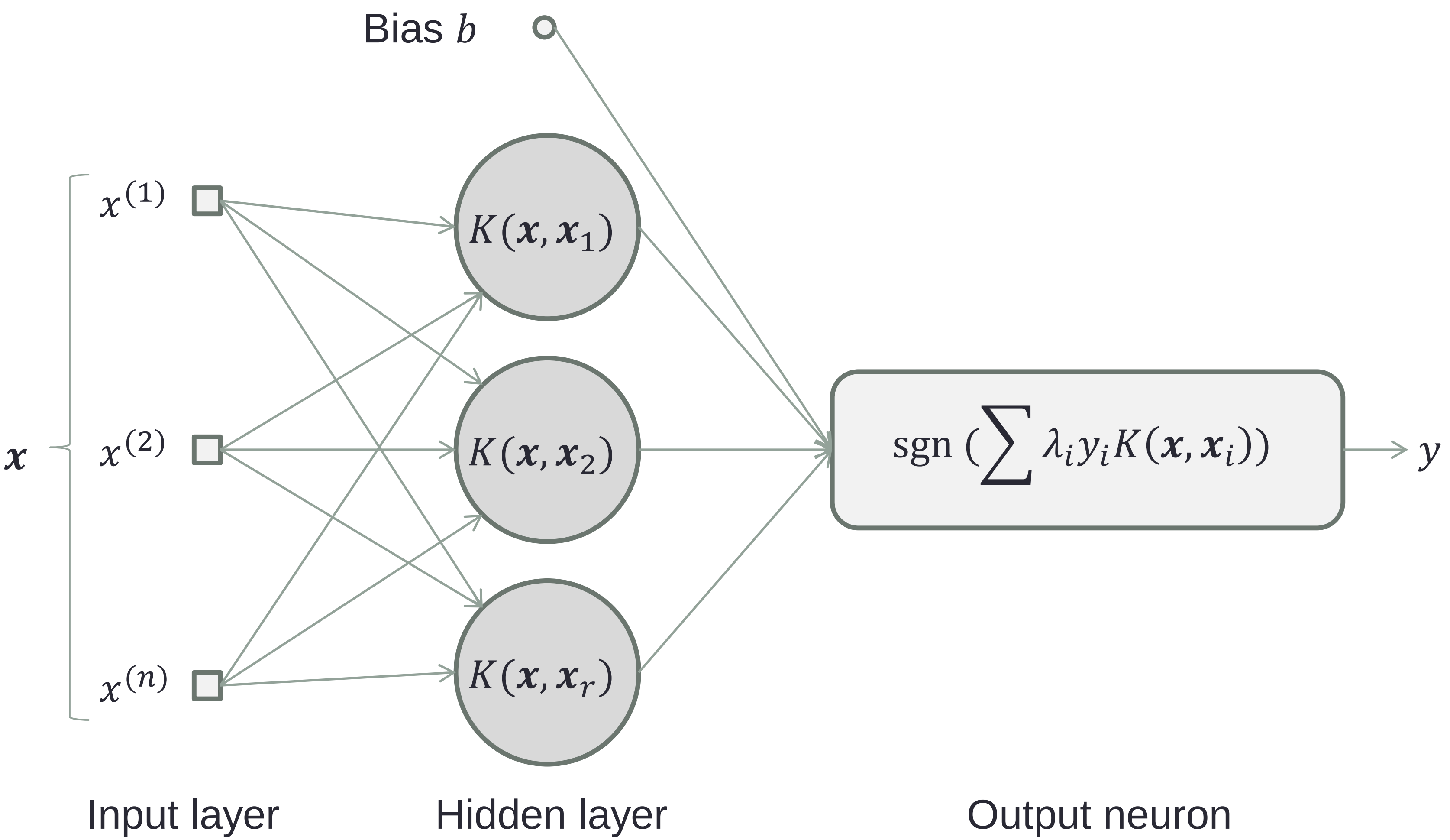
$$= \operatorname{sgn}\left(\left(\begin{bmatrix} 0 \\ 0 \\ -1/\sqrt{2} \\ 0 \\ 0 \\ 0 \end{bmatrix}\right)^{\mathrm{T}} \begin{bmatrix} 1 \\ \left(x^{(1)}\right)^2 \\ \sqrt{2}x^{(1)}x^{(2)} \\ \left(x^{(2)}\right)^2 \\ \sqrt{2}x^{(1)} \\ \sqrt{2}x^{(2)} \end{bmatrix} + 0\right)$$

$$= \operatorname{sgn}\left(-x^{(1)}x^{(2)}\right)$$

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 $\frac{1}{2}||\boldsymbol{w}||^2$ “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 $\boldsymbol{w}$ 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

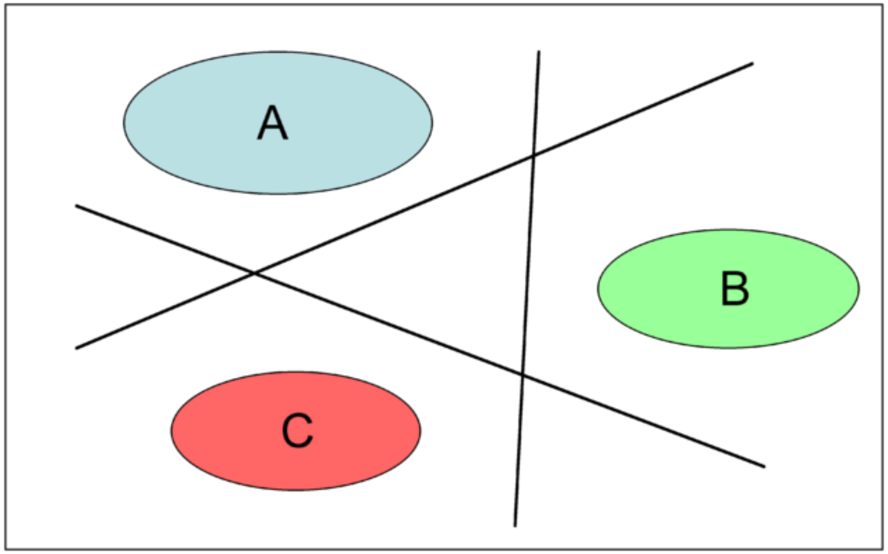
Weakness of SVM

- 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?



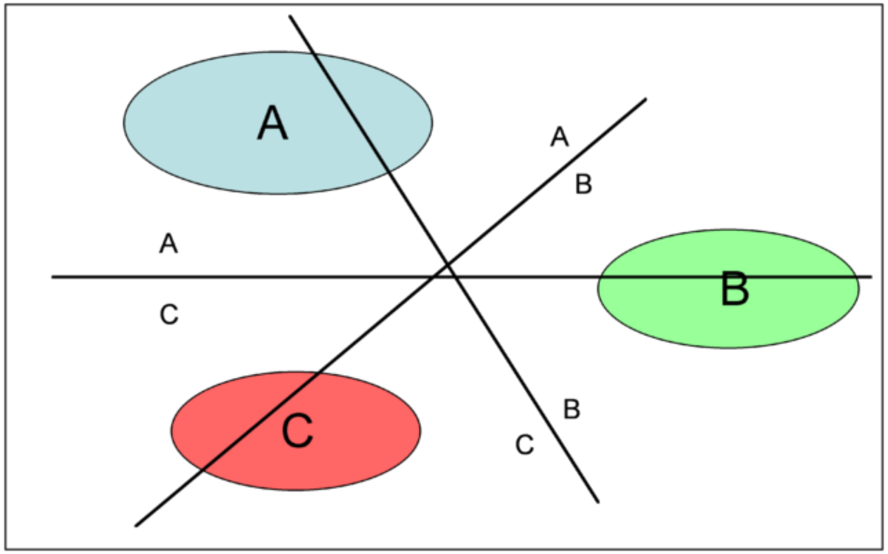
Multi-class SVM

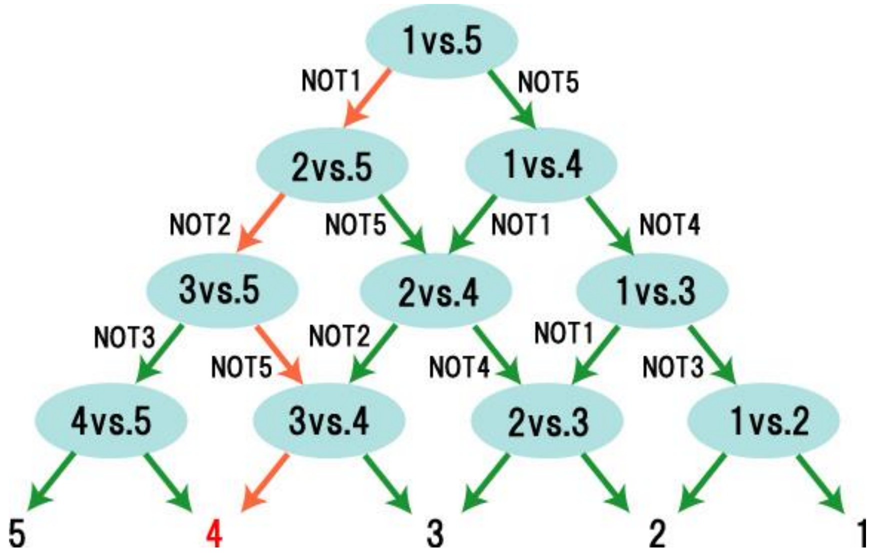
- Two strategies – building binary classifiers which distinguish between (i) one of the classes to the rest (one-versus-all) or (ii) between every pair of classes (one-versus-one)
- One-versus-all: Train $q \in \aleph$ SVMs each of which separates a single class from all the others, and the classification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



Multi-class SVM (Cont'd)

- One-versus-one: Train $q(q-1)/2 \in \aleph$ 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

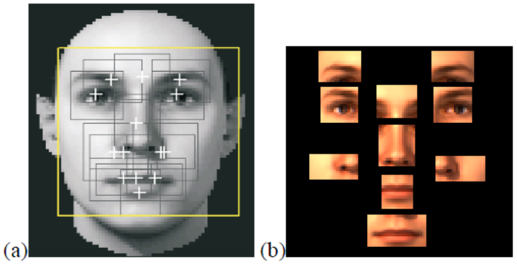
- A library for SVMs developed by NTU CSIE
- Library website: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
- Guide: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
- Test data: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/data/
- Read “README” in the package first before act!

LibSVM (Cont'd)

- Procedure suggested by the guild

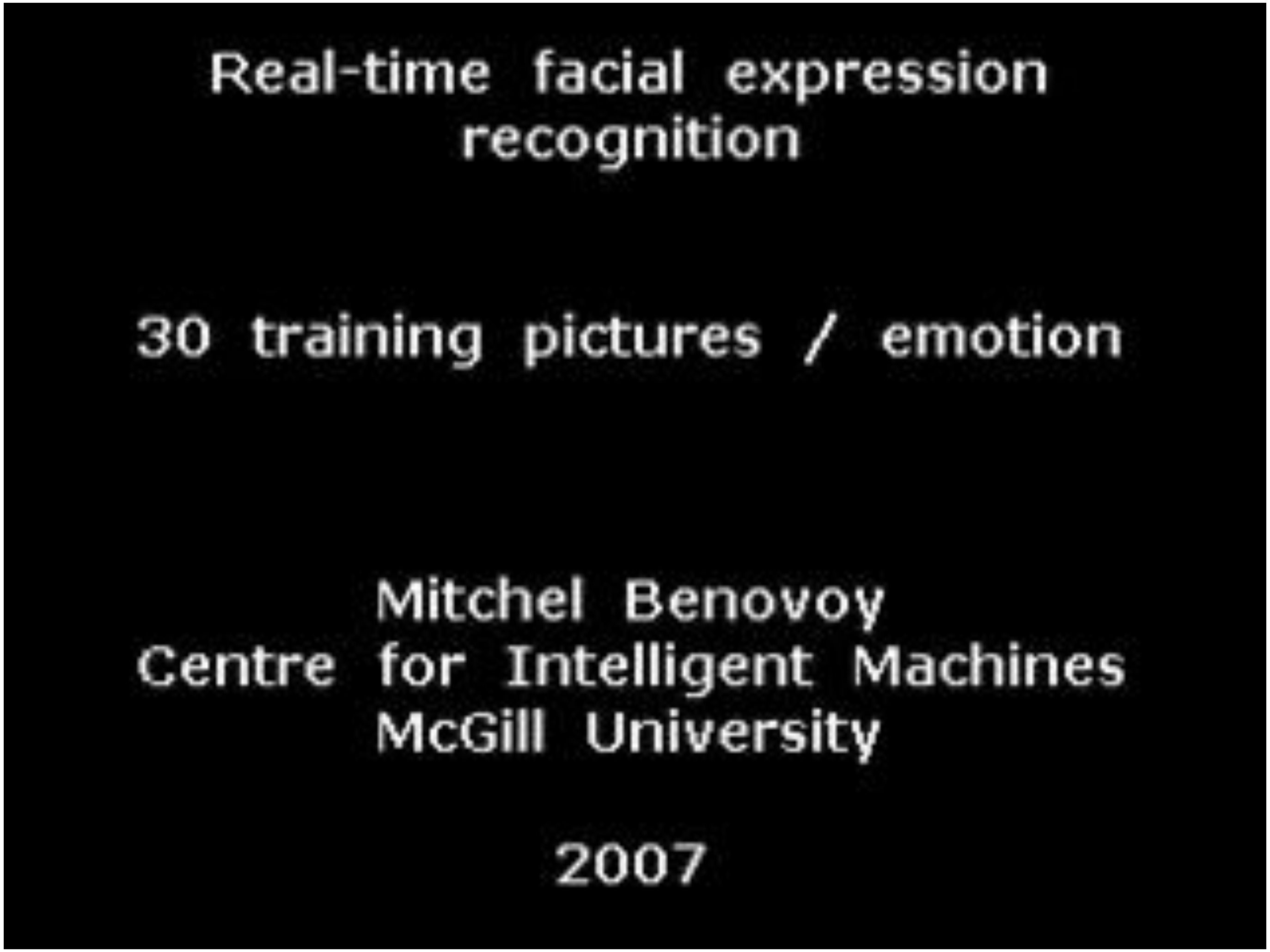
1. Transform data to the format of an SVM package
2. Conduct simple scaling on the data
3. Consider the RBF kernel: $K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = e^{-\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}}$
4. Use cross-validation to find the best parameter c and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

Face Recognition with SVM



- [Heisle, Ho and Pogio]
- 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

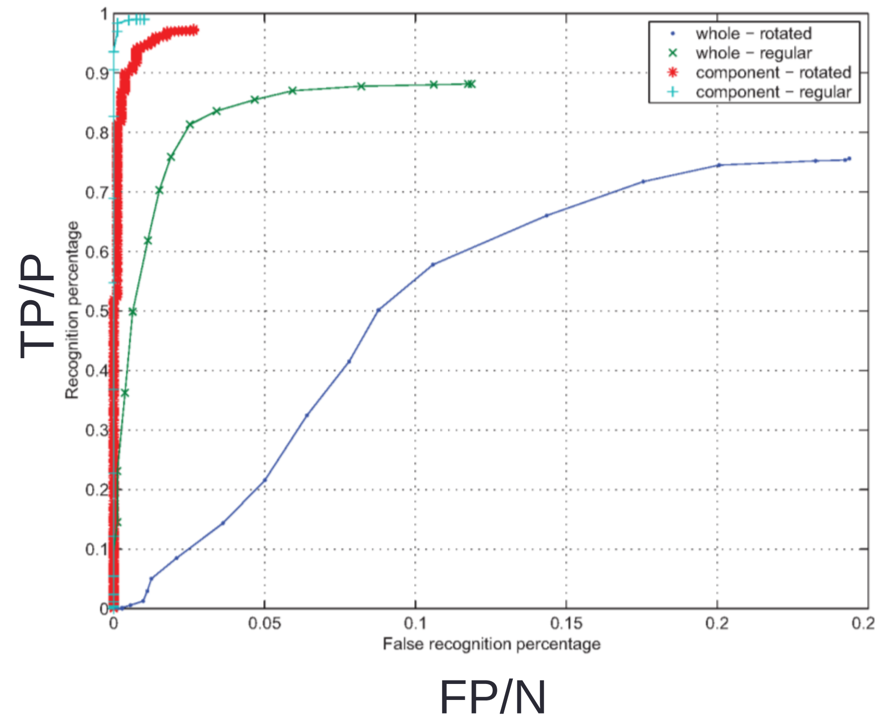
Real-time Facial Expression Recognition



Result of Face Recognition

- Receiver operating characteristic (ROC) figure

		Actual value	
		positive	*negative*
Prediction outcome	*positive*	True Positive	False Positive
	negative	False Negative	True Negative
	total	P	N



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

SVM Regression

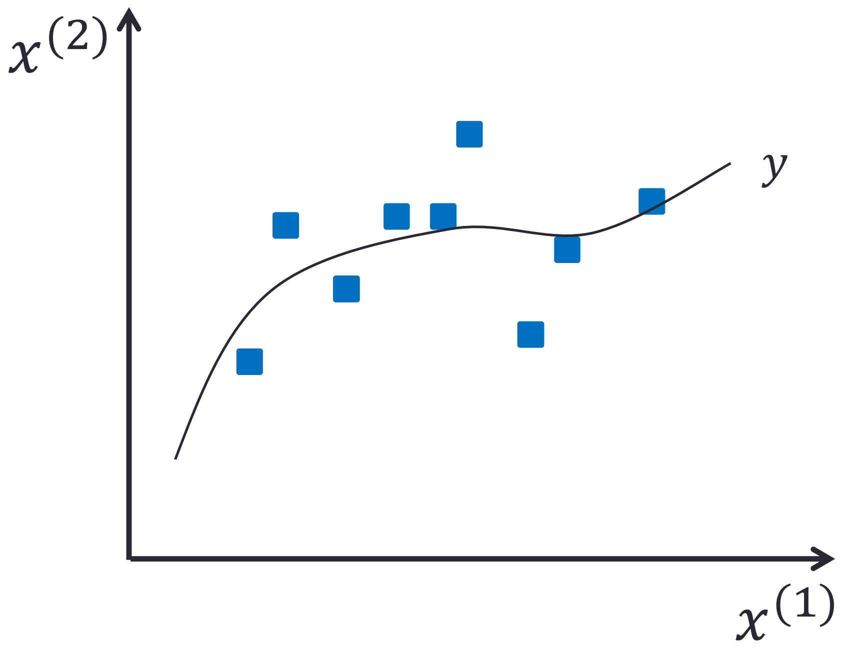
- Suppose we are given training data

$$(\boldsymbol{x}_i, y_i) \in \Re^n \times \Re$$

- Problem: find a <u>hyperplane</u>

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

that predicts y_i for given $\boldsymbol{x}_i$

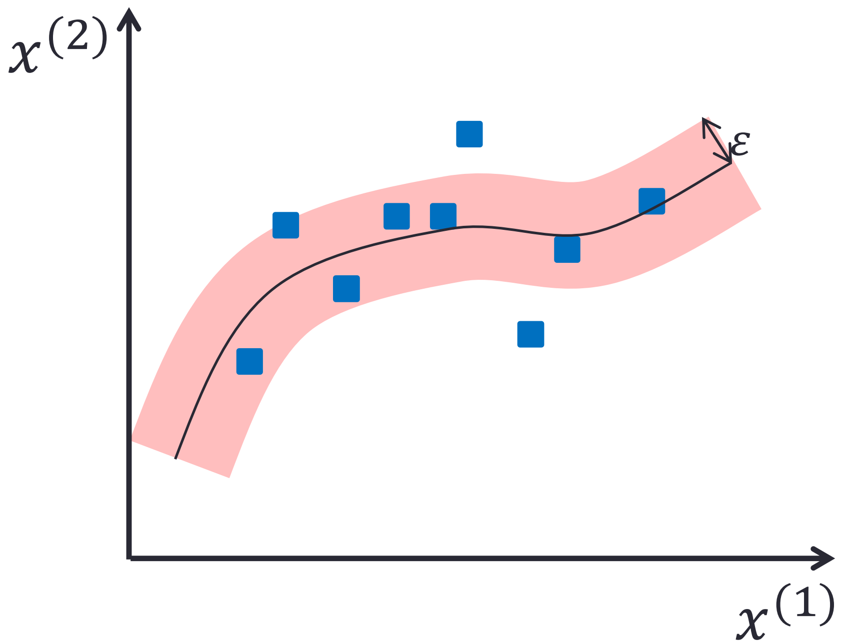


ε-insensitive Zone

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

$$|y - f(\boldsymbol{x})| \leq \varepsilon \;\; \Rightarrow$$

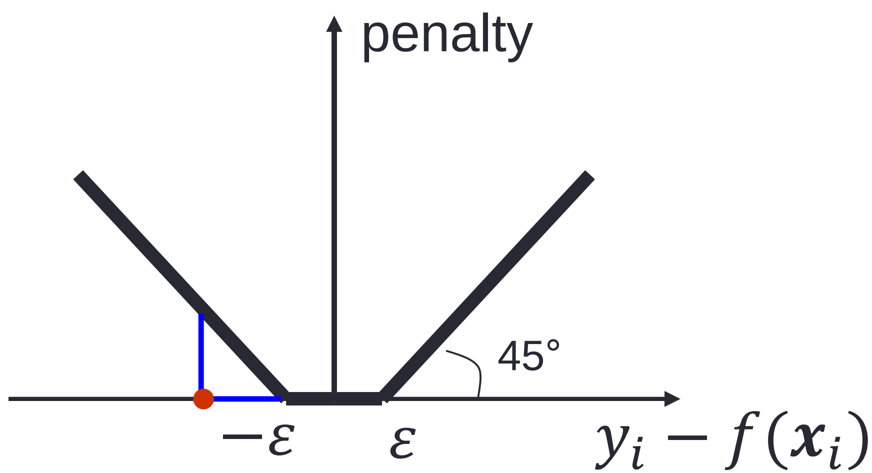
$$\begin{cases} y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b \leq \varepsilon \\ \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b - y_i \leq \varepsilon \end{cases}$$

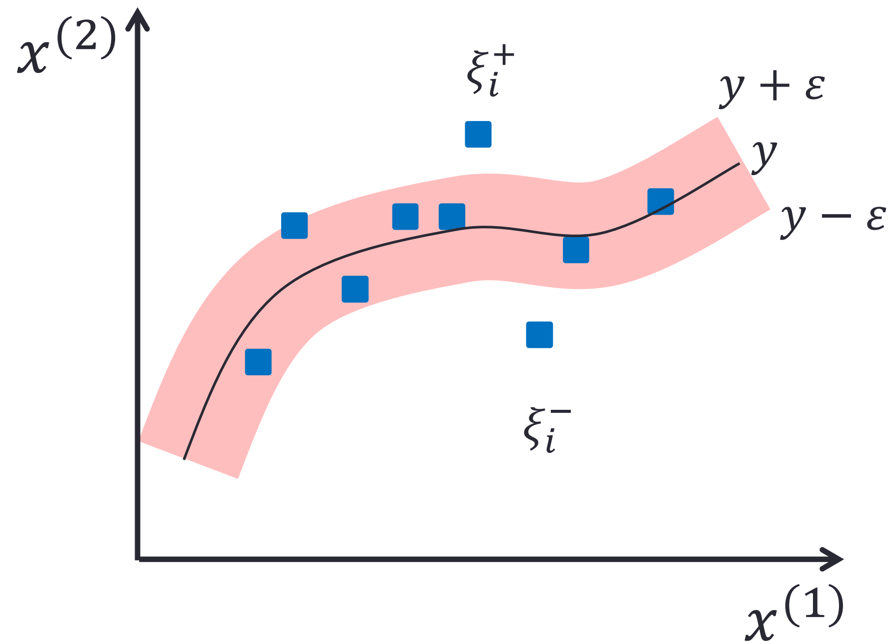


Penalty Outside the ε-insensitive Zone

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

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





Problem Formulation

- SVM regression

$$\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + c\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-)$$

$$\text{subject to } \begin{cases} y_i - \boldsymbol{w}^\mathrm{T}\boldsymbol{x}_i - b \le \varepsilon + \xi_i^+ \\ \boldsymbol{w}^\mathrm{T}\boldsymbol{x}_i - b - y_i \le \varepsilon + \xi_i^- \\ \xi_i^+, \xi_i^- > 0 \end{cases}$$

Lagrangian of SVM Regression

- Primal:

$$L_p = \frac{1}{2}\|\boldsymbol{w}\|^2 + c\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) - \sum_{i=1}^{n}(\lambda_i{}^+\xi_i^+ + \lambda_i{}^-\xi_i^-)$$
$$-\sum_{i=1}^{n}\gamma_i{}^+\left(\varepsilon + \xi_i^+ - y_i + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right)$$
$$-\sum_{i=1}^{n}\gamma_i{}^-\left(\varepsilon + \xi_i^- + y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i - b\right)$$

Solving SVM Regression

- Derivate of L_p:

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

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

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

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

Solution of SVM Regression

- Classifier:

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

- 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):

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

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

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

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

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

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:

$$\varepsilon - y_i + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \geq 0 \quad \text{and} \quad \xi_i^+ = 0 \quad \text{if} \quad {\gamma_i}^+ < c$$

$$\varepsilon - y_i + \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b \leq 0 \quad \text{if} \quad {\gamma_i}^+ > 0$$

- The range of b:

$$\max\{-\varepsilon + y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i | {\gamma_i}^+ < c \ \text{ or } \ {\gamma_i}^- > 0\}$$
$$\leq b \leq$$
$$\min\{-\varepsilon + y_i - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i | {\gamma_i}^+ > 0 \ \text{ or } \ {\gamma_i}^- < c\}$$

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