기계학습 및 데이터마이닝
- Support Vector Machine -

손경아

아주대학교

Content

- Linear SVM
- Linear SVM: non-separable case
- Non-linear SVM
- Kernel trick

Introduction

- SVMs provide a learning technique for classification

- Solution provided SVM is
  - Theoretically elegant
  - Computationally Efficient
  - Very effective in many large practical problems

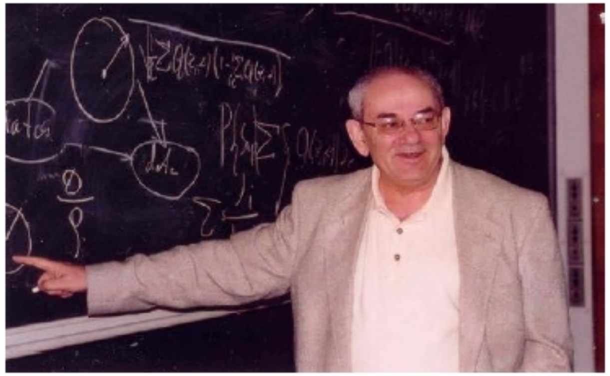
- It has a simple geometrical interpretation in a high-dimensional feature space that is nonlinearly related to input space

- By using kernels all computations keep simple.

History

- The Study on Statistical Learning Theory was started in the 1960s by Vapnik
- Support Vector Machine is a practical learning method based on Statistical Learning Theory
- A simple SVM could beat a sophisticated neural networks with elaborate features in a handwriting recognition task.
- Now deep neural networks could beat SVM

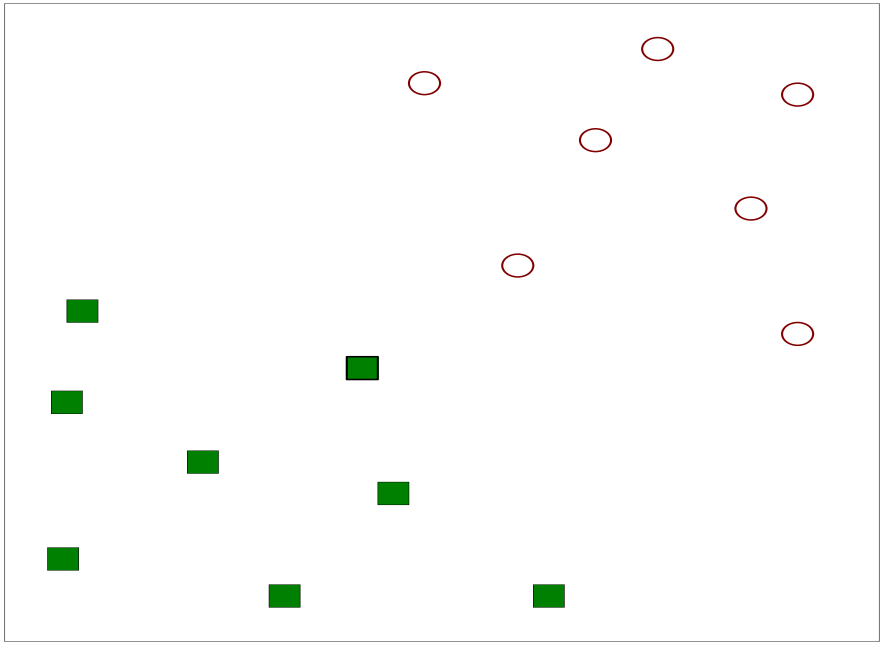
Vladimir Vapnik



Separable Hyperplanes

- Imagine a situation where you have a two class classification problem with two predictors X_1 and X_2.
- Suppose that the two classes are "linearly separable" i.e. one can draw a straight line in which all points on one side belong to the first class and points on the other side to the second class.
- Then a natural approach is to find the straight line that gives the biggest separation between the classes i.e. the points are as far from the line as possible
- This is the basic idea of a support vector classifier.

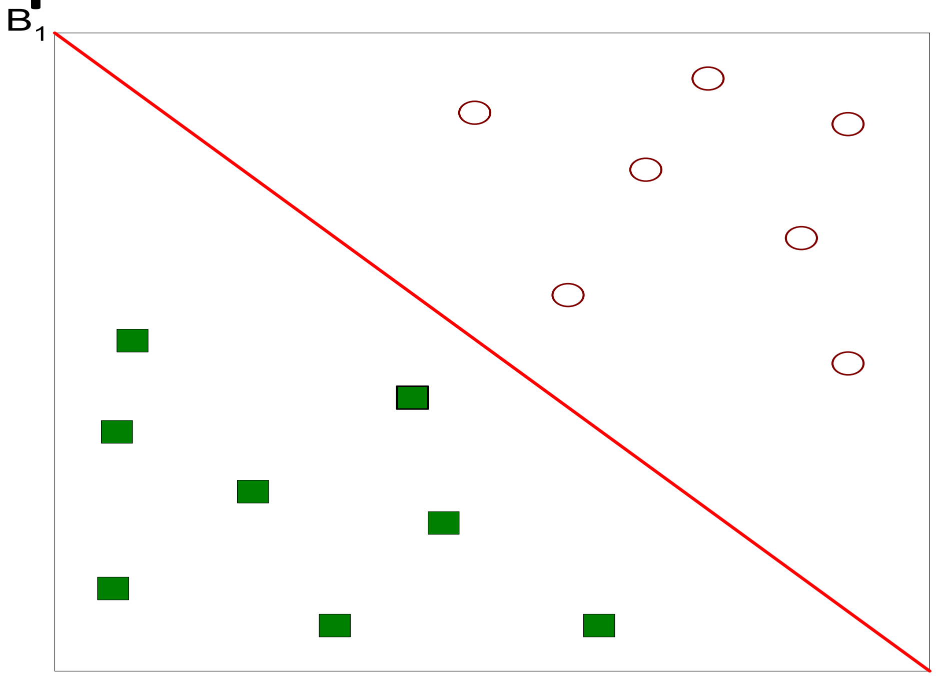
Support Vector Machines



Two dimensional training data

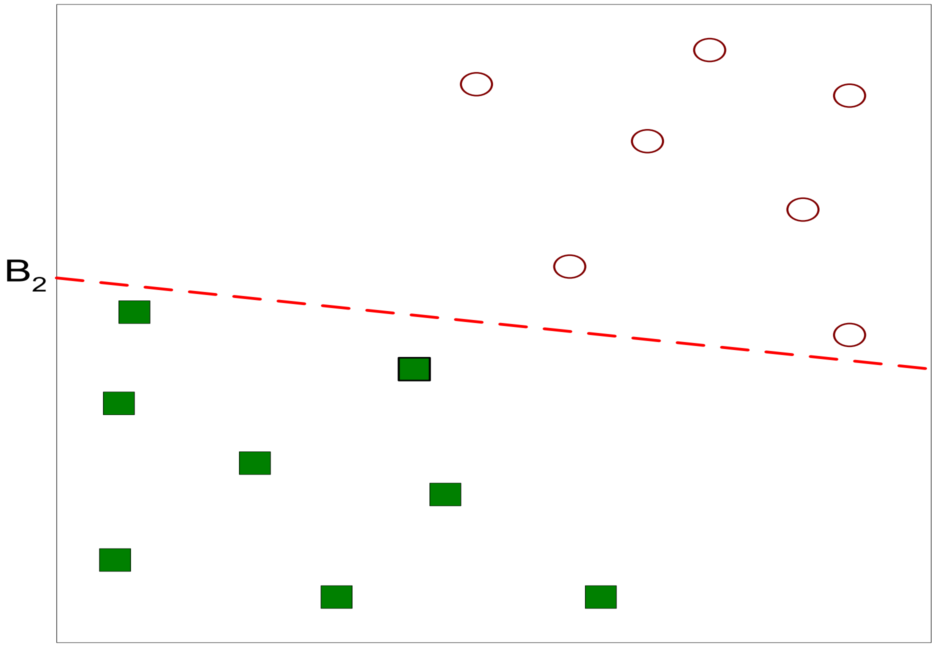
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



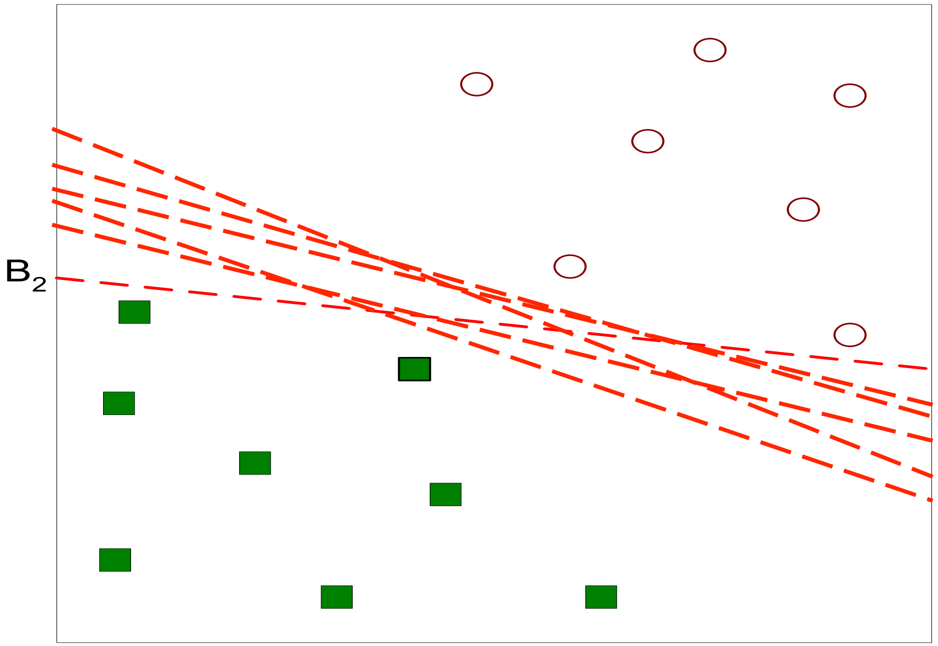
- One Possible Solution

Support Vector Machines



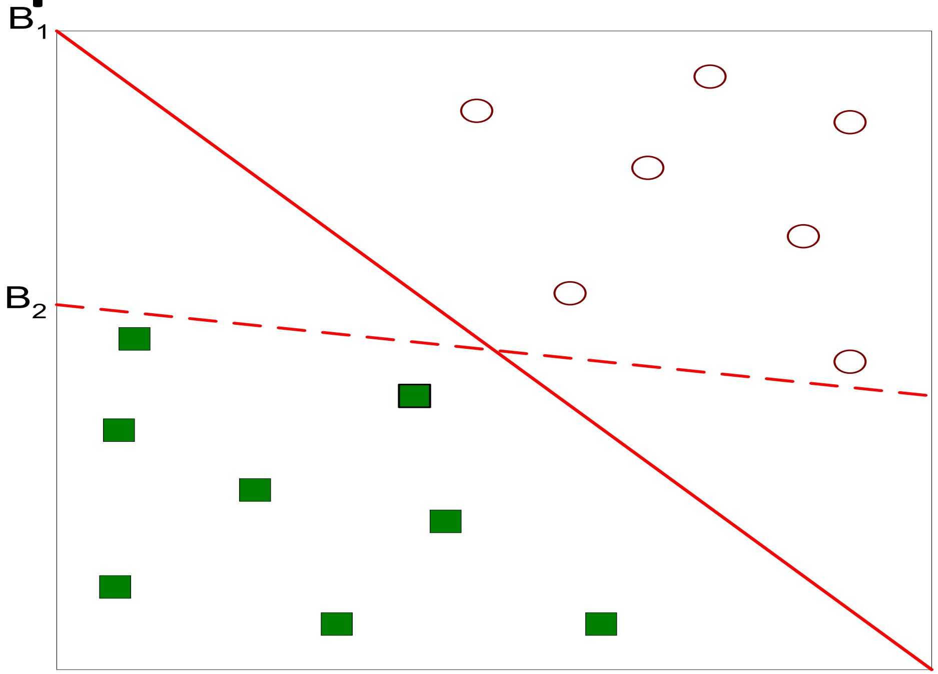
- Another possible solution

Support Vector Machines



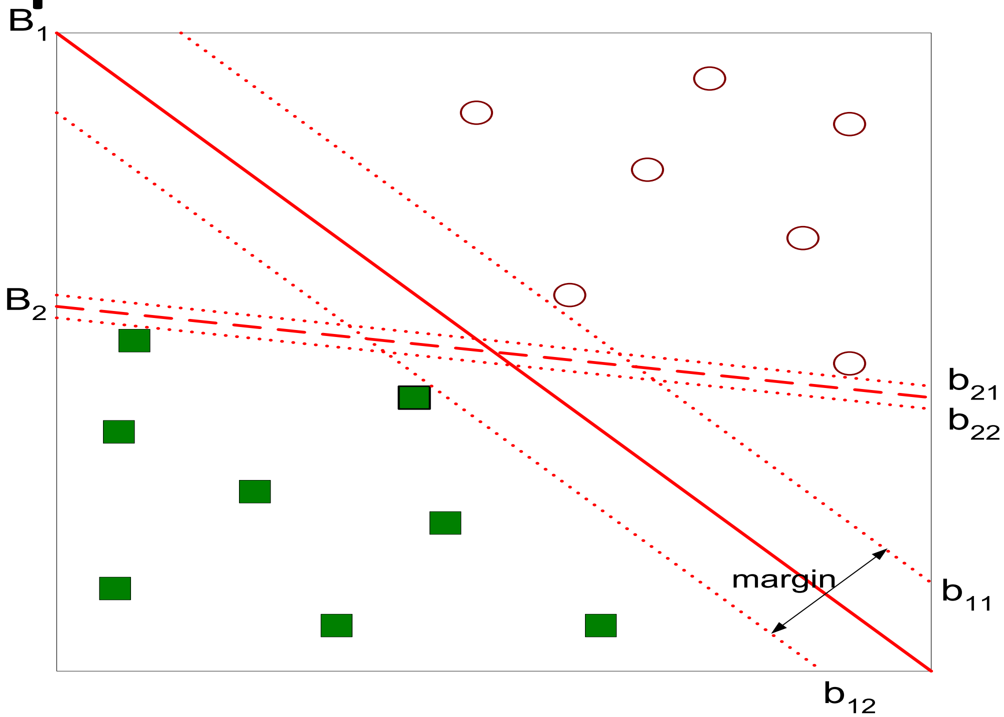
- Other possible solutions

Support Vector Machines



- Which one is better? B1 or B2?
- How do you define better?

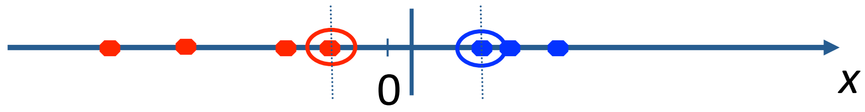
Support Vector Machines



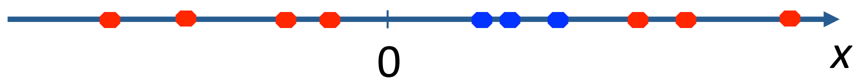
- Find hyperplane maximizes the margin => B1 is better than B2

Non-linear SVMs

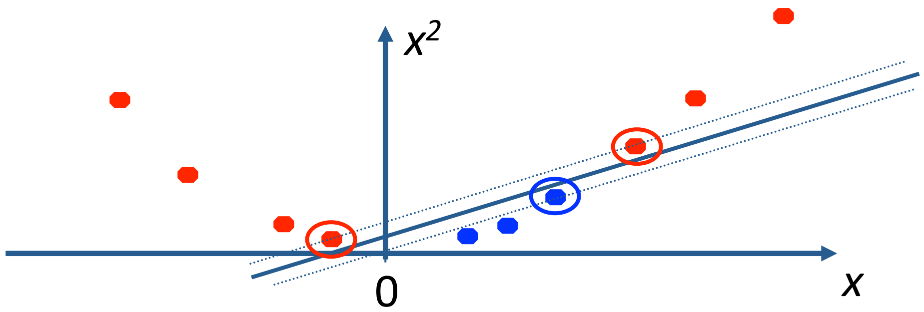
- Datasets that are linearly separable with some noise work out great:



- But what are we going to do if the dataset is just too hard?



- How about… mapping data to a higher-dimensional space:



Issues in nonlinear SVM

- It is not clear what type of mapping function to use

- Solving optimization problems in a high-dimensional space can be computationally expensive

➔ Use **Kernel Trick**

Linear SVM: Separable case

- Linear decision boundary

- Consider a binary classification problem consisting of N training samples ($\mathbf{x}_i$,y_i), i=1,…,N where
  - $\mathbf{x}_i=(x_{i1},x_{i2},\ldots,x_{id})^T$: the attributes
  - y_i : class labels (either -1 or 1)
  - Decision boundary: $\mathbf{w} \cdot \mathbf{x} + b = 0$

Linear Classifier

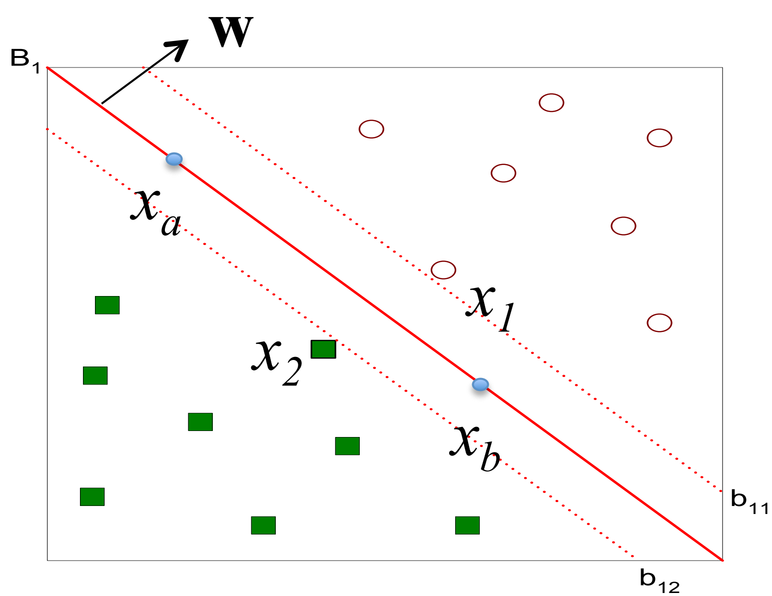
$$\mathbf{w} \cdot \mathbf{x_a} + b = 0$$

$$\mathbf{w} \cdot \mathbf{x_b} + b = 0$$

$$\Rightarrow \mathbf{w} \cdot (\mathbf{x_a} - \mathbf{x_b}) = 0$$

One possible linear classifier:

$$y = \begin{cases} 1 & \text{if } \mathrm{w} \bullet \mathrm{x} + \mathrm{b} > 0 \\ -1 & \text{if } \mathrm{w} \bullet \mathrm{x} + \mathrm{b} < 0 \end{cases}$$

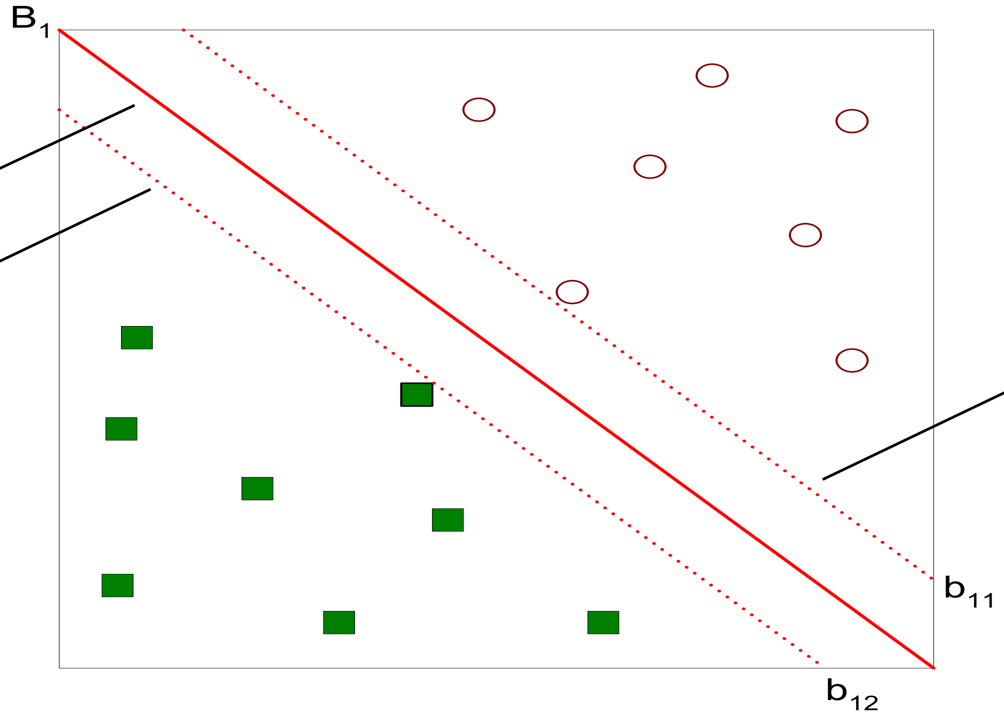


$$\mathbf{w} \cdot \mathbf{x} + b = 0$$

Linear SVM

$$w \bullet x + b = 0$$

$$w \bullet x + b = -1$$

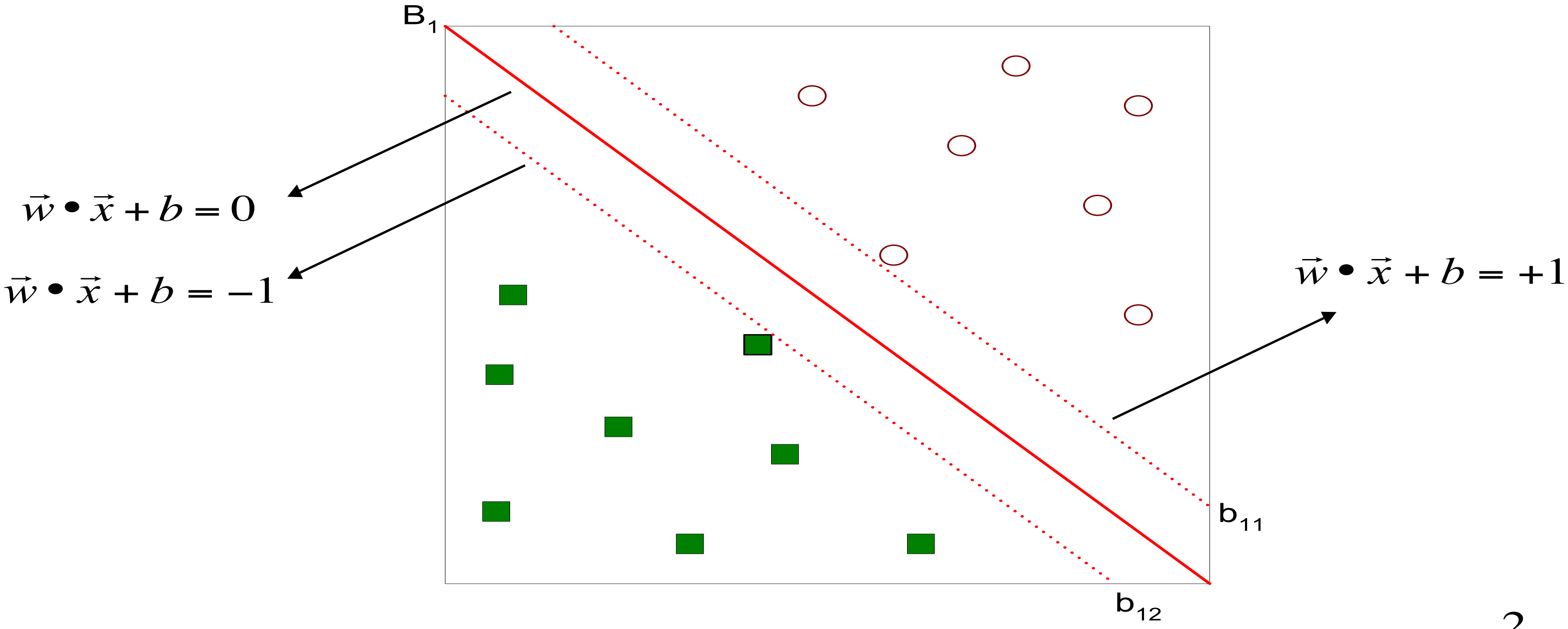


$$w \bullet x + b = +1$$

$$y = \begin{cases} 1 & \text{if } w \bullet x + b \geq 1 \\ -1 & \text{if } w \bullet x + b \leq -1 \end{cases}$$

Margin $=$

Linear SVM



$$f(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \bullet \vec{x} + b \geq 1 \\ -1 & \text{if } \vec{w} \bullet \vec{x} + b \leq -1 \end{cases}$$

Maximize $\text{Margin} = \frac{2}{\|\vec{w}\|}$

Learning linear SVM

- Training phase: estimate the parameters w and b from the training data
- Objective: maximize margin
- Constraints:

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 \text{ if } y_i = 1$$
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 \text{ if } y_i = -1$$

$$\Longrightarrow \quad y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, i = 1, ..., N$$

Optimization problem

- We want to maximize: $\text{Margin} = \dfrac{2}{\|\vec{w}\|^2}$
  - Which is equivalent to minimizing: $L(w) = \dfrac{\|\vec{w}\|^2}{2}$
  - But subjected to the following constraints:

$$y_i(\mathbf{w} \cdot \mathbf{x_i} + b) \geq 1, i = 1, ..., N$$

- This is a constrained (convex) optimization problem
  - Numerical approaches to solve it (e.g., quadratic programming)
  - Lagrange multiplier method:

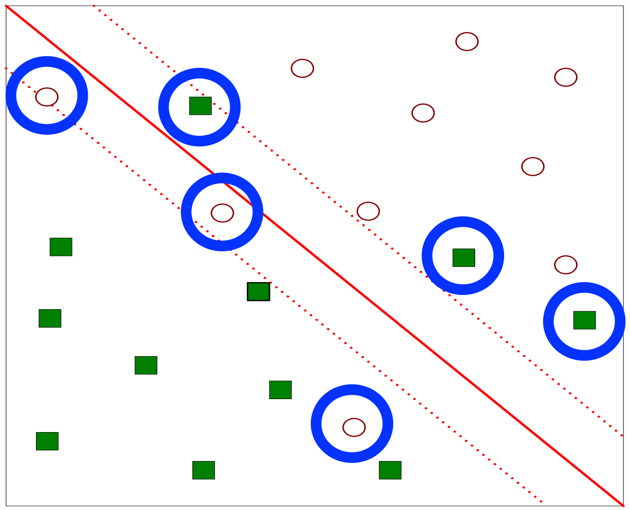
Linear SVM

- Test phase
  - Once the parameters of the decision boundary are found, a test instance z is classified as follows:

$$f(z) = 1 \text{ if } \mathbf{w} \cdot \mathbf{z} + b \geq 0$$
$$f(z) = -1 \text{ otherwise}$$

Linear SVM: non-separable case

- What if the problem is not linearly separable?



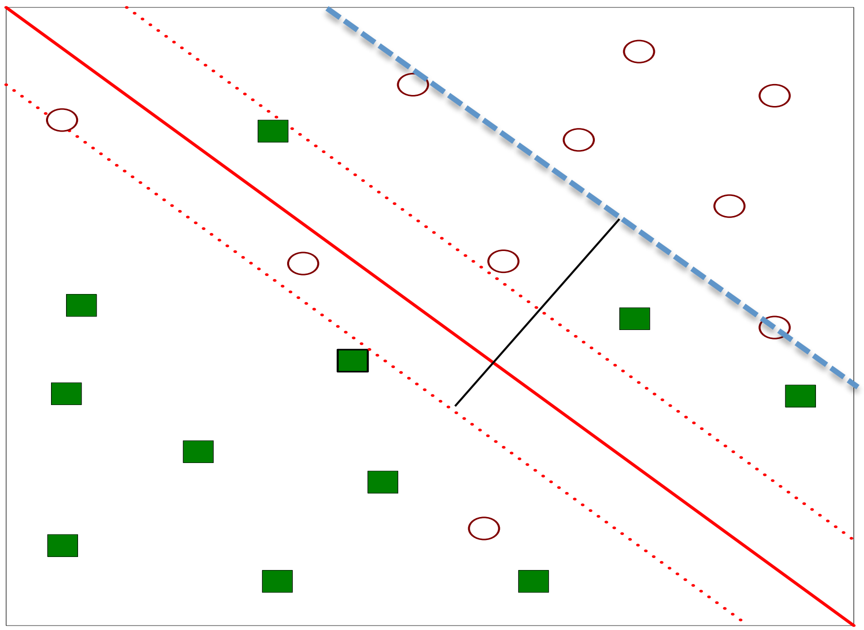
Use a soft margin approach

Slack variables for non-separable data

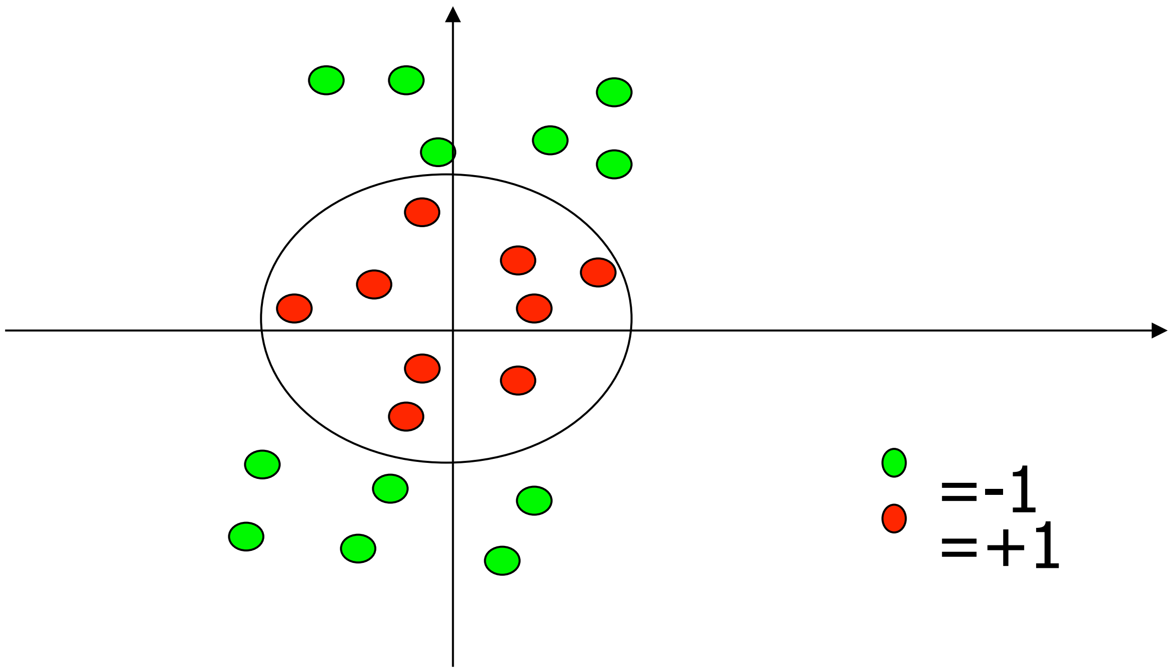
Relax the constraints

$$\mathbf{w} \cdot \mathbf{x_i} + b \geq 1 - \xi \text{ if } y_i = 1$$
$$\mathbf{w} \cdot \mathbf{x_i} + b \leq -1 + \xi \text{ if } y_i = -1$$

minimize $L(w) = \dfrac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i\right)^k$



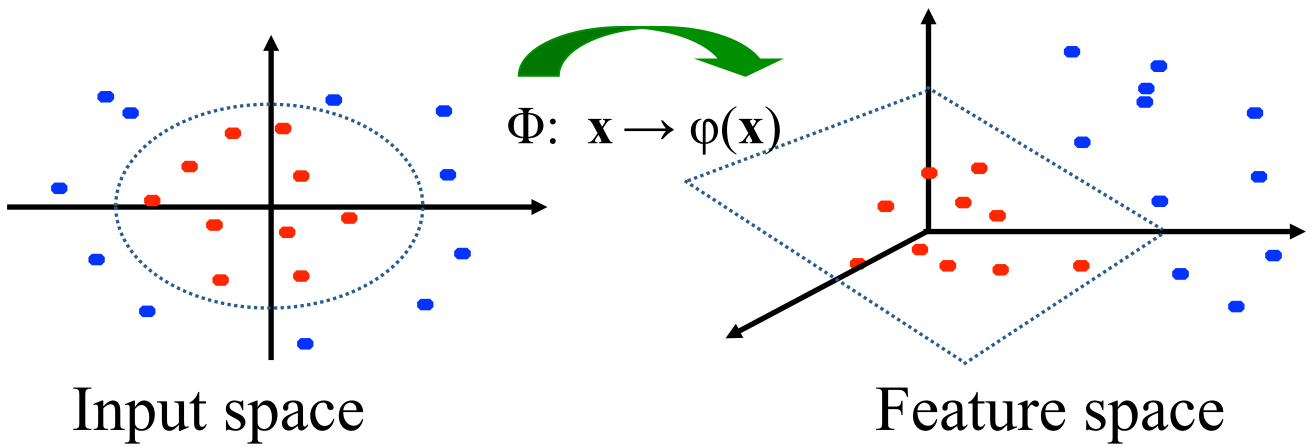
Problems with linear SVM



What if the decison function is not a linear?

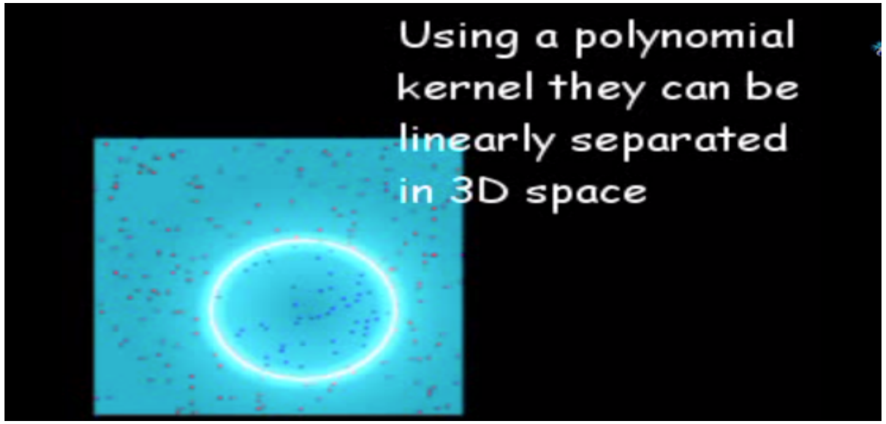
Extension to Non-linear Decision Boundary

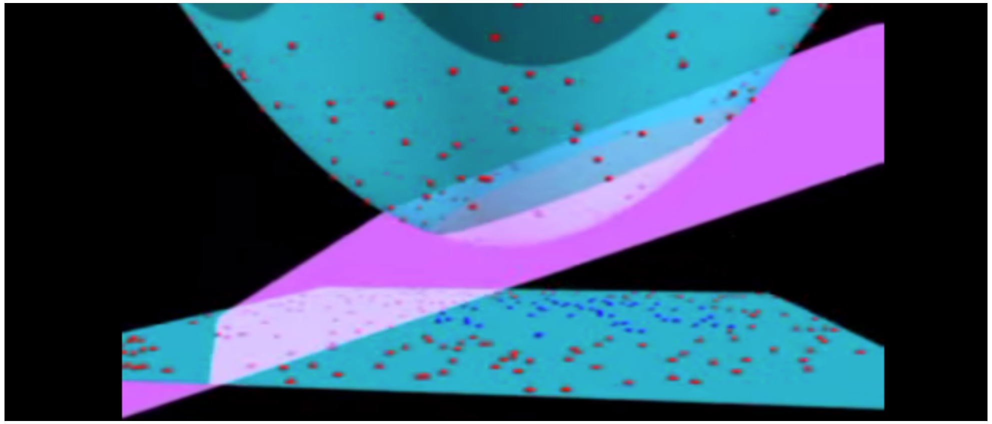
- Possible problem of the transformation
  - High computation burden and hard to get a good estimate
- SVM solves these two issues simultaneously
  - Kernel tricks for efficient computation
  - Minimizing $||\mathbf{w}||^2$ can lead to a "good" classifier



SVM with polynomial kernel visualization

http://www.youtube.com/watch?v=3liCbRZPrZA





Issues in nonlinear SVM

- what type of mapping function to use

- Computation in a high-dimensional space can be expensive

➔ Use **Kernel Trick**

Kernel trick

$$\mathbf{w} \cdot \mathbf{\Phi}(\mathbf{z}) + b = \sum_{i=1}^{n} \lambda_i y_i \Phi(x_i) \cdot \mathbf{\Phi}(\mathbf{z}) + b$$

Dot product in a high-dimensional space…
A kind of similarity measure

- There may exist a kernel function K such that

$$K(\mathbf{u}, \mathbf{v}) = \mathbf{\Phi}(\mathbf{u})\mathbf{\Phi}(\mathbf{v})$$

  – The dot product in a the transformed space can be expressed in terms of a similarity in the original space
  – e.g. K(u,v) = (u • v + 1)^2 for $\mathbf{\Phi}(\mathbf{u}) = (u_1^2, u_2^2, \sqrt{2}u_1, \sqrt{2}u_2, 1)$

Kernel Trick

- K: kernel function
  - The kernel functions can be expressed as the dot product between two input vectors in some high-dimensional space
  - Computing the dot product using kernel functions is considerably cheaper than using the transformed attribute
  - We do not have to know the exact form of the mapping function Φ

Examples of Kernel Functions

- Polynomial kernel with degree *d*

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

- Radial basis function kernel with width σ

$$K(\mathbf{x}, \mathbf{y}) = \exp(-||\mathbf{x} - \mathbf{y}||^2 / (2\sigma^2))$$

  – Closely related to radial basis function neural networks

- Sigmoid with parameter κ and θ

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

SVM

- Convex optimization problem in which efficient algorithms are available
- Maximizing margin of the decision boundary
- Attribute transformation to a high-dimensional space and kernel trick
- The user must still provide other parameters such as the type of kernel function and the cost function C for slack variables
- For binary classification. Can be extended to multi-class problems