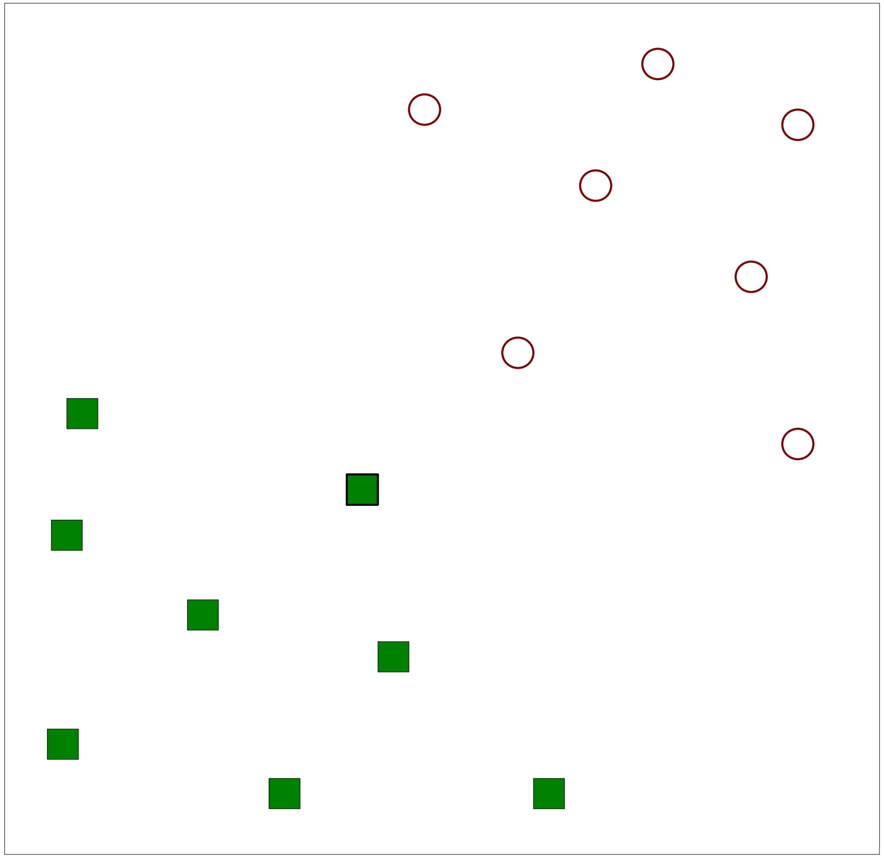
Support Vector Machines

Introduction to Data Mining, 2nd Edition

by

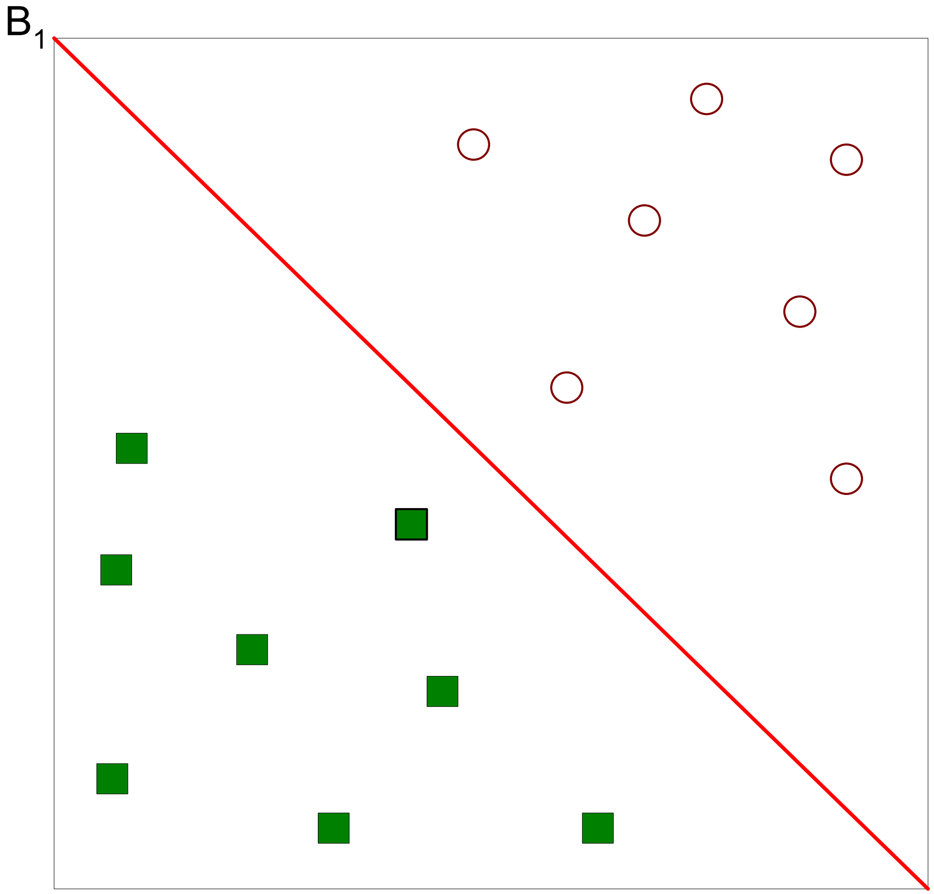
Tan, Steinbach, Karpatne, Kumar

Support Vector Machines



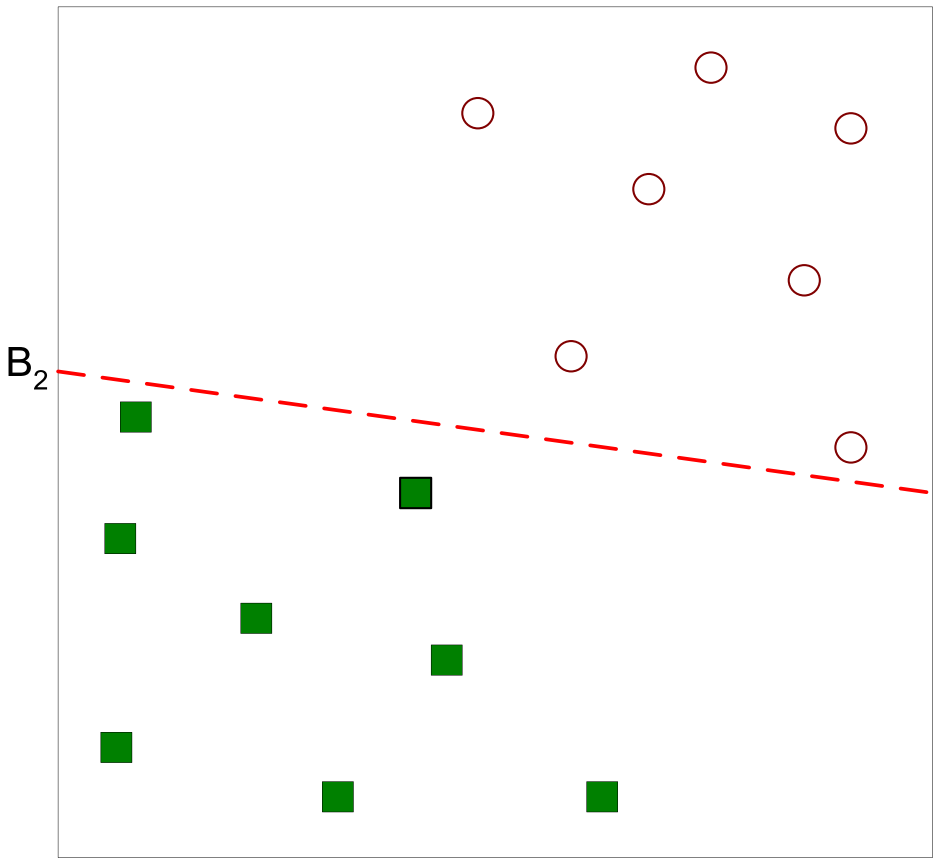
- Find a linear hyperplane (decision boundary) that will separate the data

Support Vector Machines



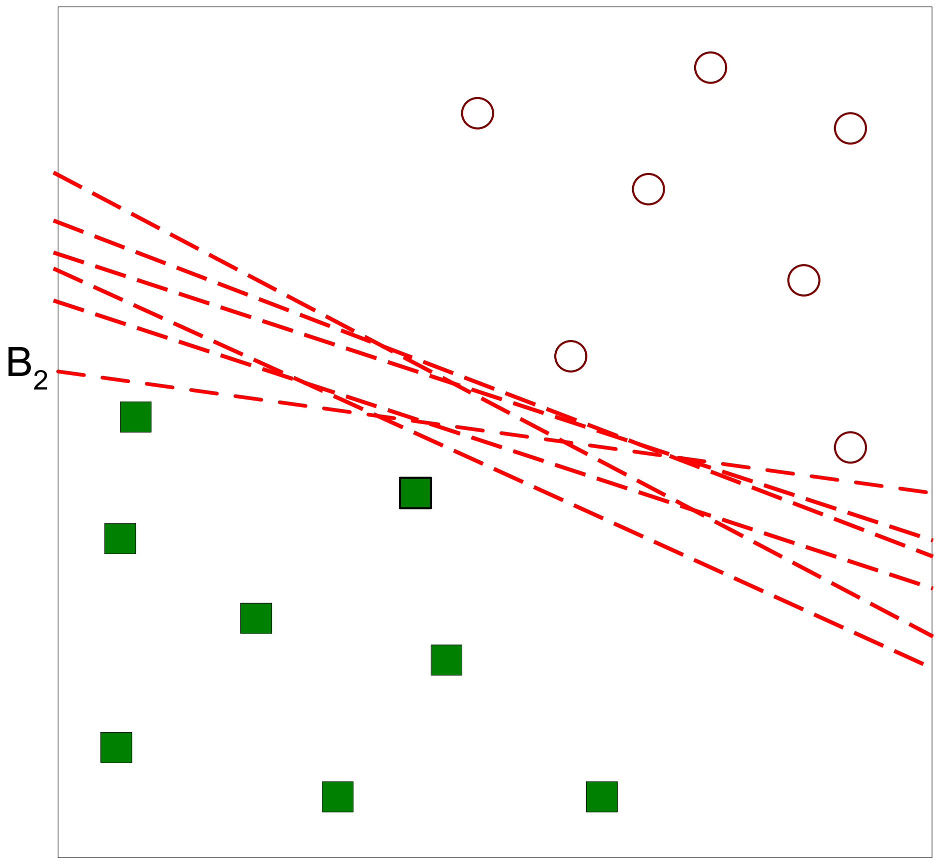
- One Possible Solution

Support Vector Machines



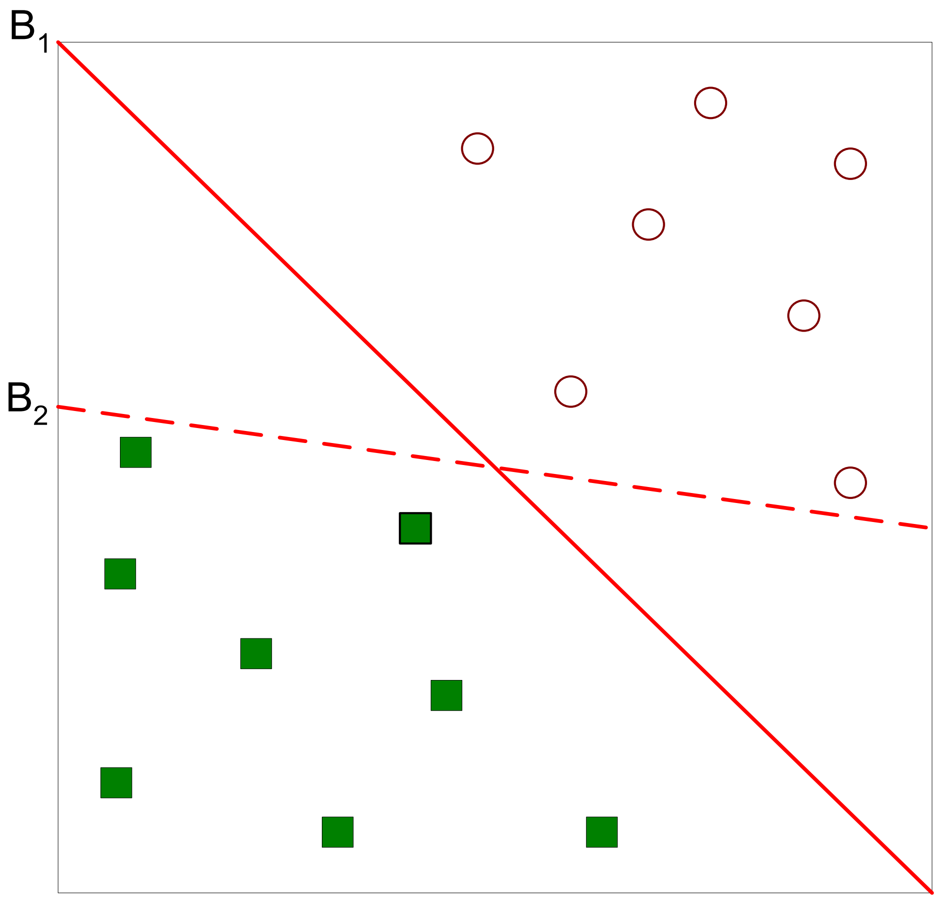
- Another possible solution

Support Vector Machines



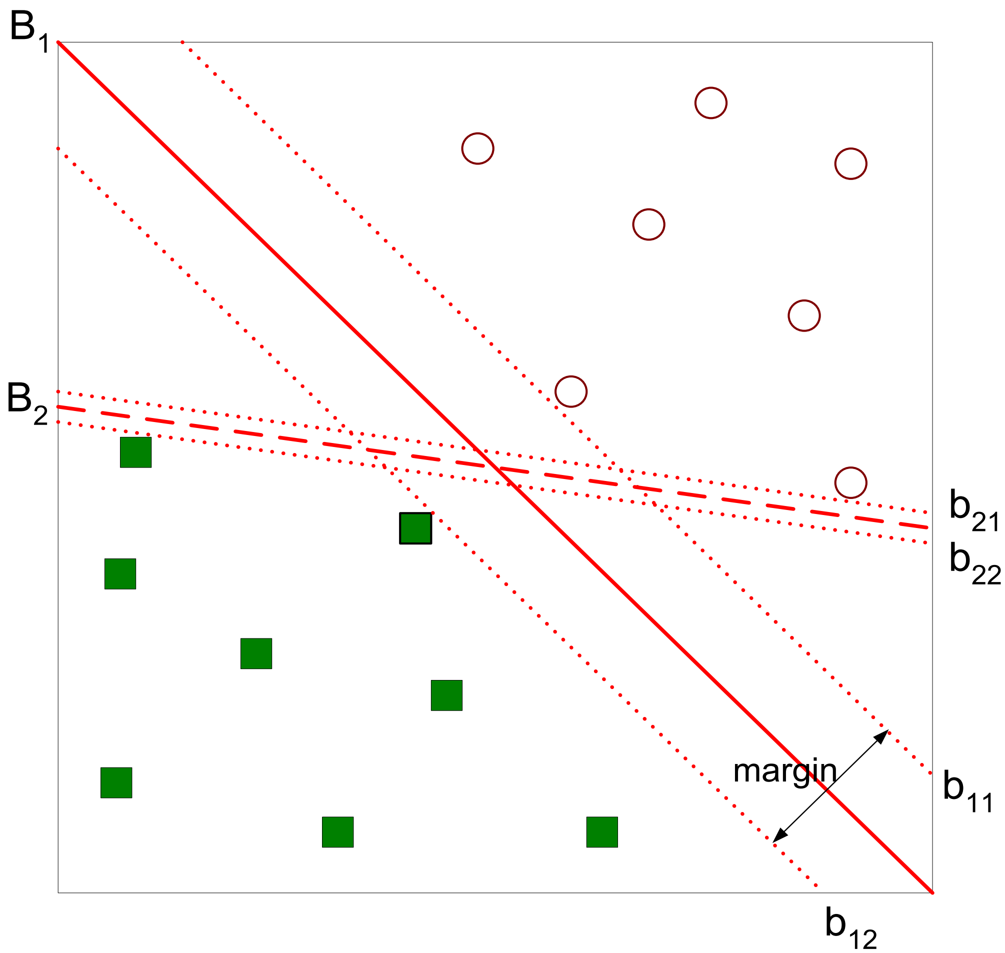
- Other possible solutions

Support Vector Machines



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

Support Vector Machines

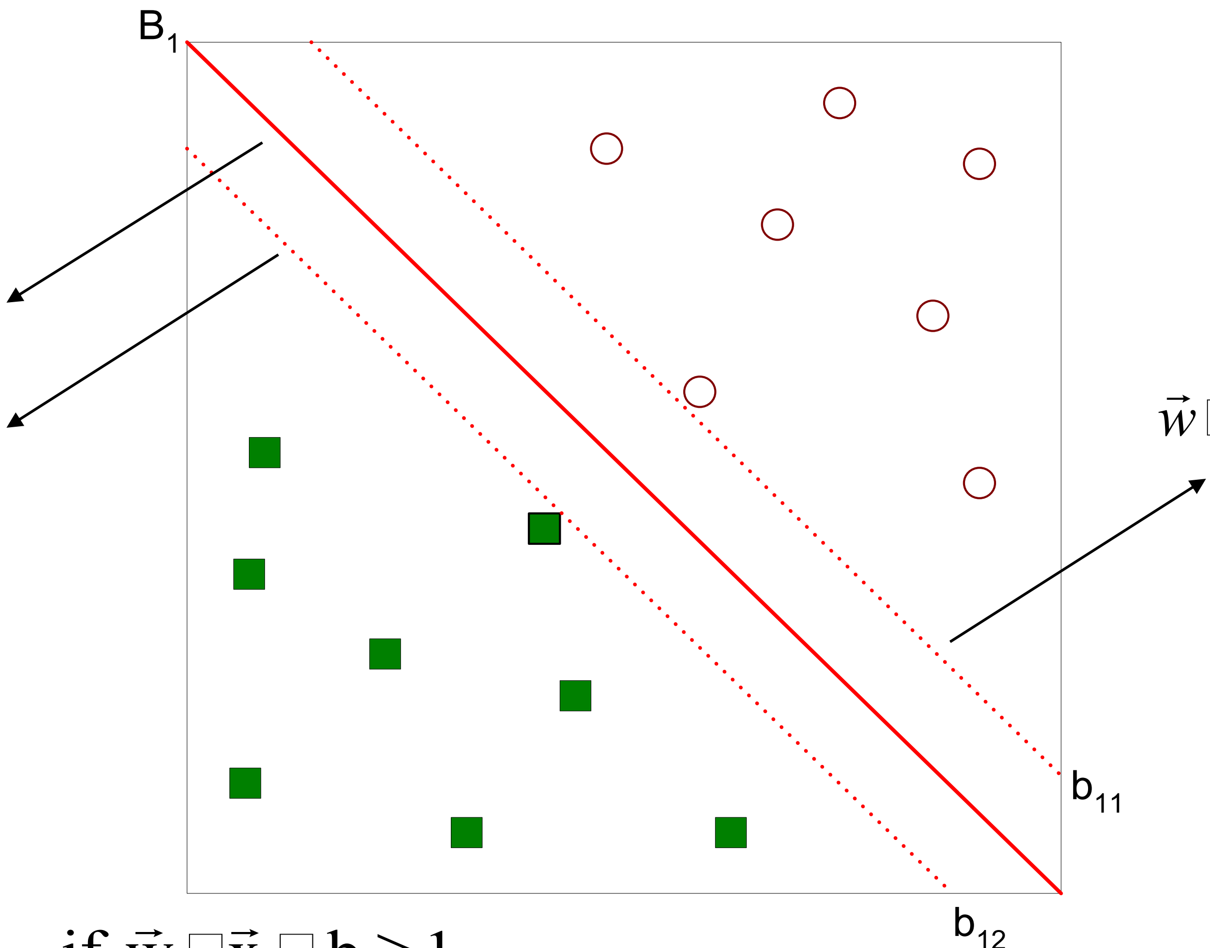


- Find hyperplane maximizes the margin => B1 is better than B2

Support Vector Machines

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

$$\vec{w} \cdot \vec{x} + b = -1$$



$$\vec{w} \cdot \vec{x} + b = +1$$

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

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

Linear SVM

- Linear model:

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

- Learning the model is equivalent to determining the values of $\vec{w}$ and b
  – How to find $\vec{w}$ and b from training data?

Learning Linear SVM

- Objective is to maximize: $\text{Margin} \square \dfrac{2}{\|\vec{w}\|}$
  - Which is equivalent to minimizing: $L(\vec{w}) \square \dfrac{\|\vec{w}\|^2}{2}$
  - Subject to the following constraints:

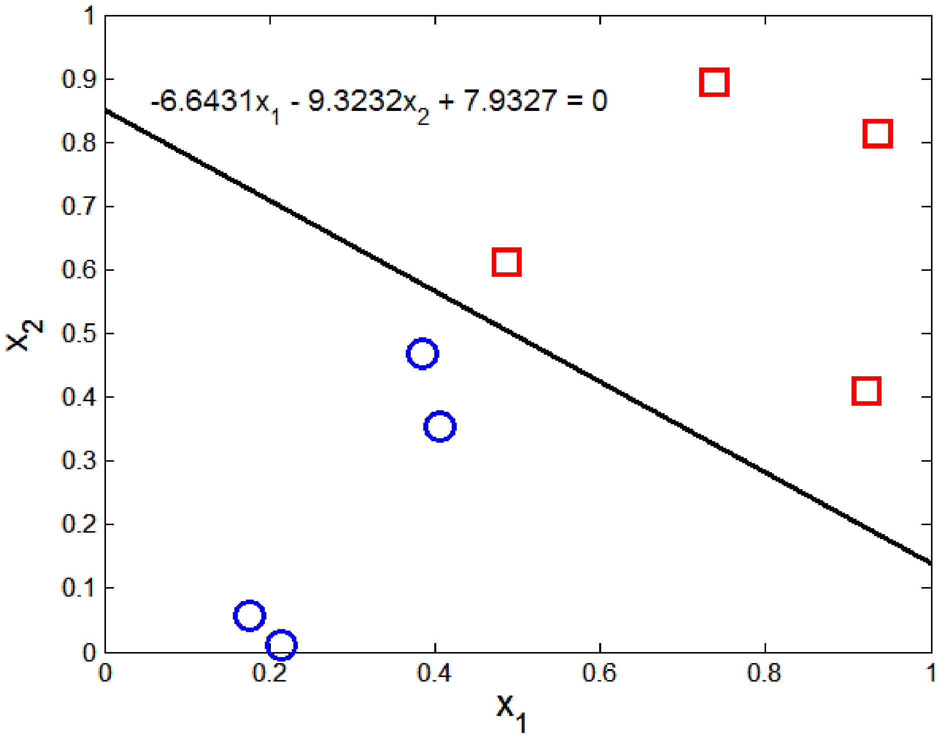
$$y_i \square \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \square \vec{\mathrm{x}}_\mathrm{i} \square \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \square \vec{\mathrm{x}}_\mathrm{i} \square \mathrm{b} \leq -1 \end{cases}$$

or

$$y_i(\mathbf{w} \square \mathbf{x}_i \square b) \geq 1, \quad i \square 1,2,\ldots,N$$

- This is a constrained optimization problem
  - Solve it using Lagrange multiplier method

Example of Linear SVM



Support vectors

x1	x2	y	λ
0.3858	0.4687	1	65.5261
0.4871	0.611	-1	65.5261
0.9218	0.4103	-1	0
0.7382	0.8936	-1	0
0.1763	0.0579	1	0
0.4057	0.3529	1	0
0.9355	0.8132	-1	0
0.2146	0.0099	1	0

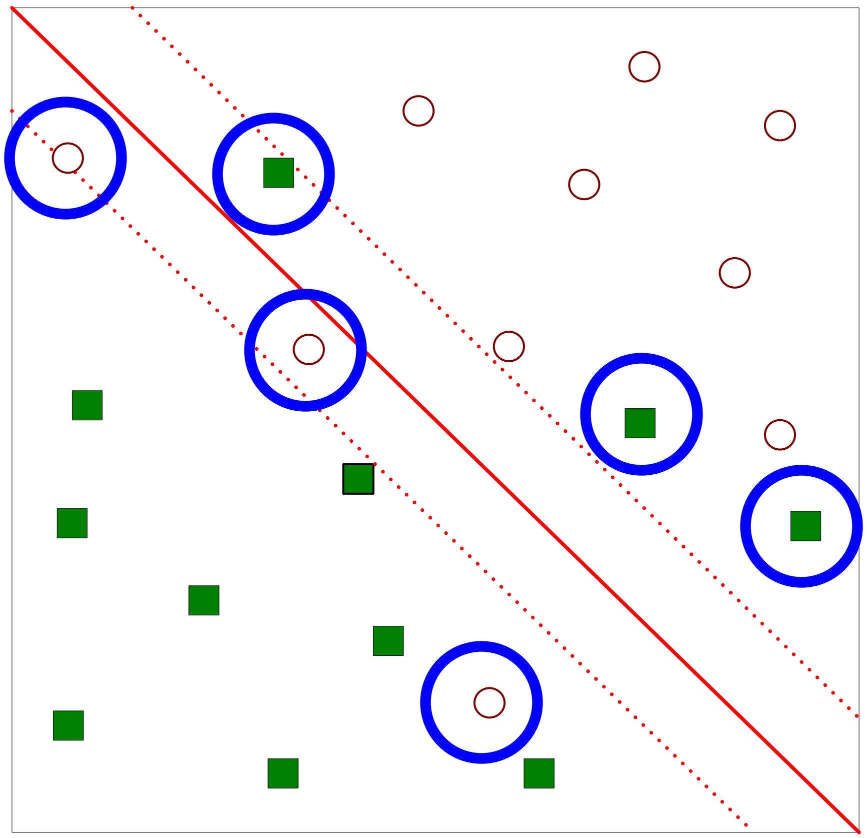
Learning Linear SVM

- Decision boundary depends only on support vectors
  - If you have data set with same support vectors, decision boundary will not change
  - How to classify using SVM once **w** and *b* are found? Given a test record, x_i

$$f(\vec{x}_i) = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \cdot \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 \\ -1 & \text{if } \vec{\mathrm{w}} \cdot \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 \end{cases}$$

Support Vector Machines

- What if the problem is not linearly separable?



Support Vector Machines

- What if the problem is not linearly separable?
  - Introduce slack variables
    - Need to minimize:

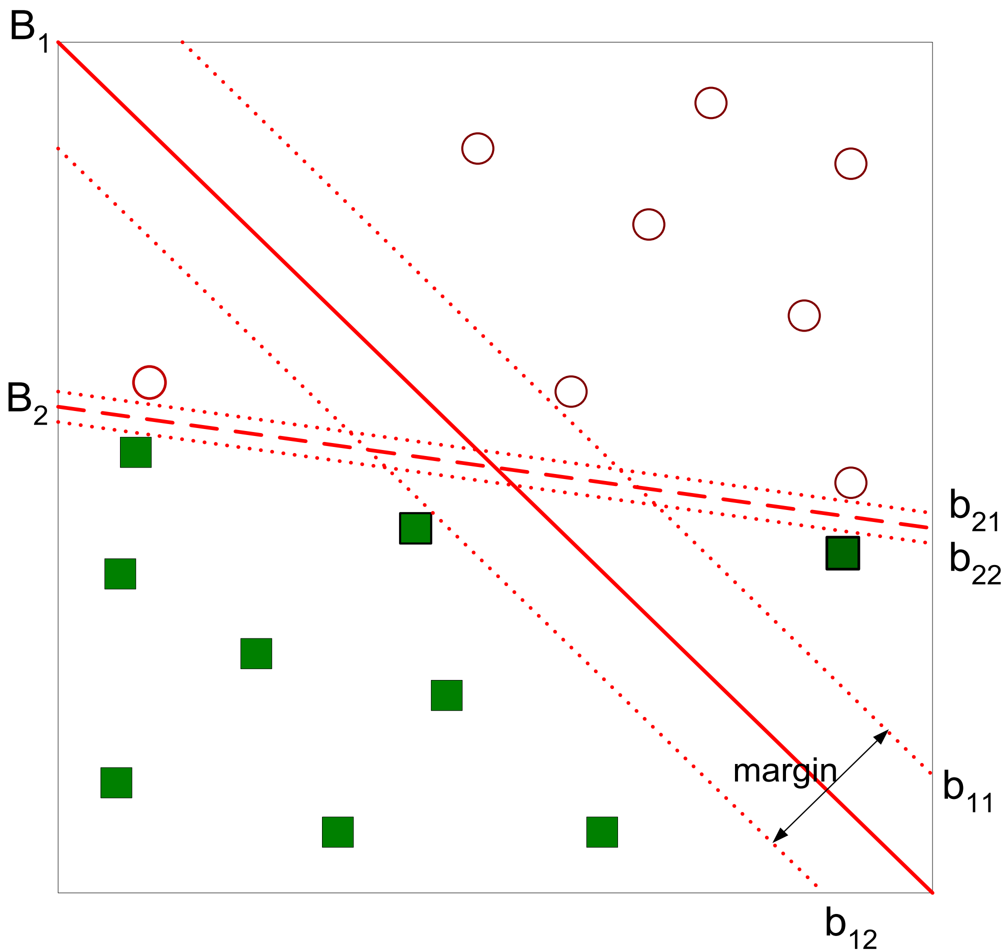
$$L(w) = \frac{\| \vec{w} \|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

    - Subject to:

$$y_i = \begin{cases} 1 & \text{if } \vec{\mathrm{w}} \cdot \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \geq 1 - \xi_{\mathrm{i}} \\ -1 & \text{if } \vec{\mathrm{w}} \cdot \vec{\mathrm{x}}_{\mathrm{i}} + \mathrm{b} \leq -1 + \xi_{\mathrm{i}} \end{cases}$$

    - If k is 1 or 2, this leads to same objective function as linear SVM but with different constraints (see textbook)

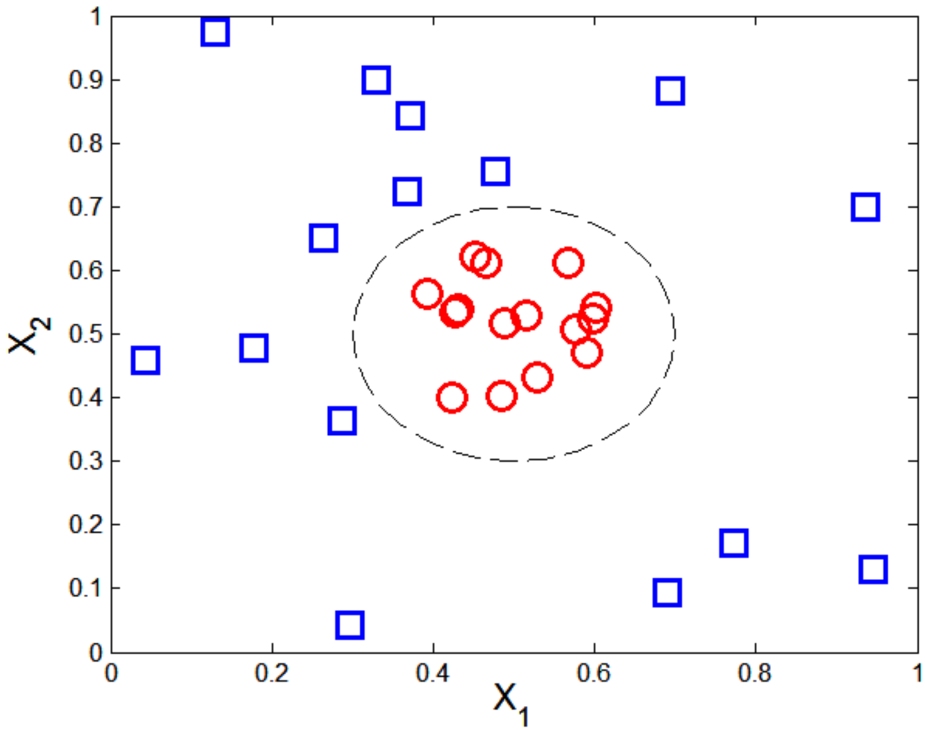
Support Vector Machines



- Find the hyperplane that optimizes both factors

Nonlinear Support Vector Machines

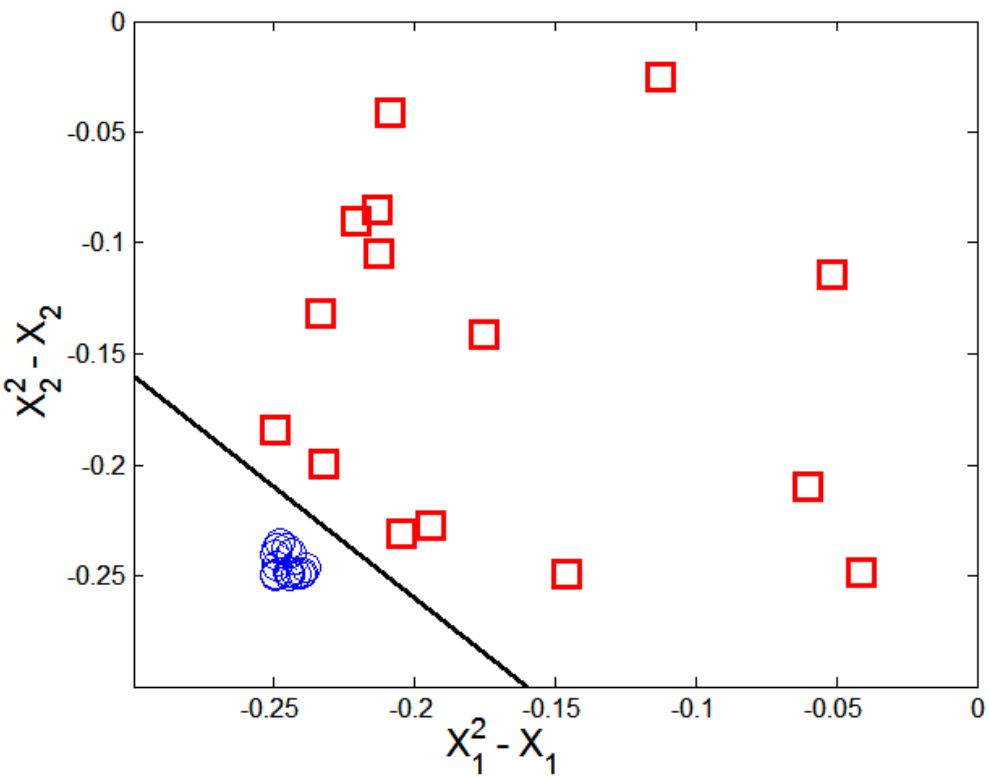
- What if decision boundary is not linear?



$$y(x_1, x_2) = \begin{cases} 1 & \text{if } \sqrt{(x_1 - 0.5)^2 + (x_2 - 0.5)^2} > 0.2 \\ -1 & \text{otherwise} \end{cases}$$

Nonlinear Support Vector Machines

- Trick: Transform data into higher dimensional space



$$x_1^2 - x_1 + x_2^2 - x_2 = -0.46.$$

$$\Phi : (x_1, x_2) \longrightarrow (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, 1).$$

$$w_4 x_1^2 + w_3 x_2^2 + w_2\sqrt{2}x_1 + w_1\sqrt{2}x_2 + w_0 = 0.$$

Decision boundary:

$$\vec{w} \cdot \Phi(\vec{x}) + b = 0$$

Learning Nonlinear SVM

- Optimization problem:

$$\min_{w} \frac{\|\mathbf{w}\|^2}{2}$$

$$\textit{subject to} \quad y_i(w \cdot \Phi(x_i) + b) \geq 1, \ \forall\{(x_i, y_i)\}$$

- Which leads to the same set of equations (but involve Φ(x) instead of x)

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

$$\mathbf{w} = \sum_i \lambda_i y_i \Phi(\mathbf{x}_i)$$

$$\lambda_i \{y_i(\sum_j \lambda_j y_j \Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i) + b) - 1\} = 0,$$

$$f(\mathbf{z}) = sign(\mathbf{w} \cdot \Phi(\mathbf{z}) + b) = sign(\sum_{i=1}^{n} \lambda_i y_i \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{z}) + b).$$

Learning NonLinear SVM

- Issues:
  – What type of mapping function Φ should be used?
  – How to do the computation in high dimensional space?
    - Most computations involve dot product $\Phi(x_i) \cdot \Phi(x_j)$
    - Curse of dimensionality?

Learning Nonlinear SVM

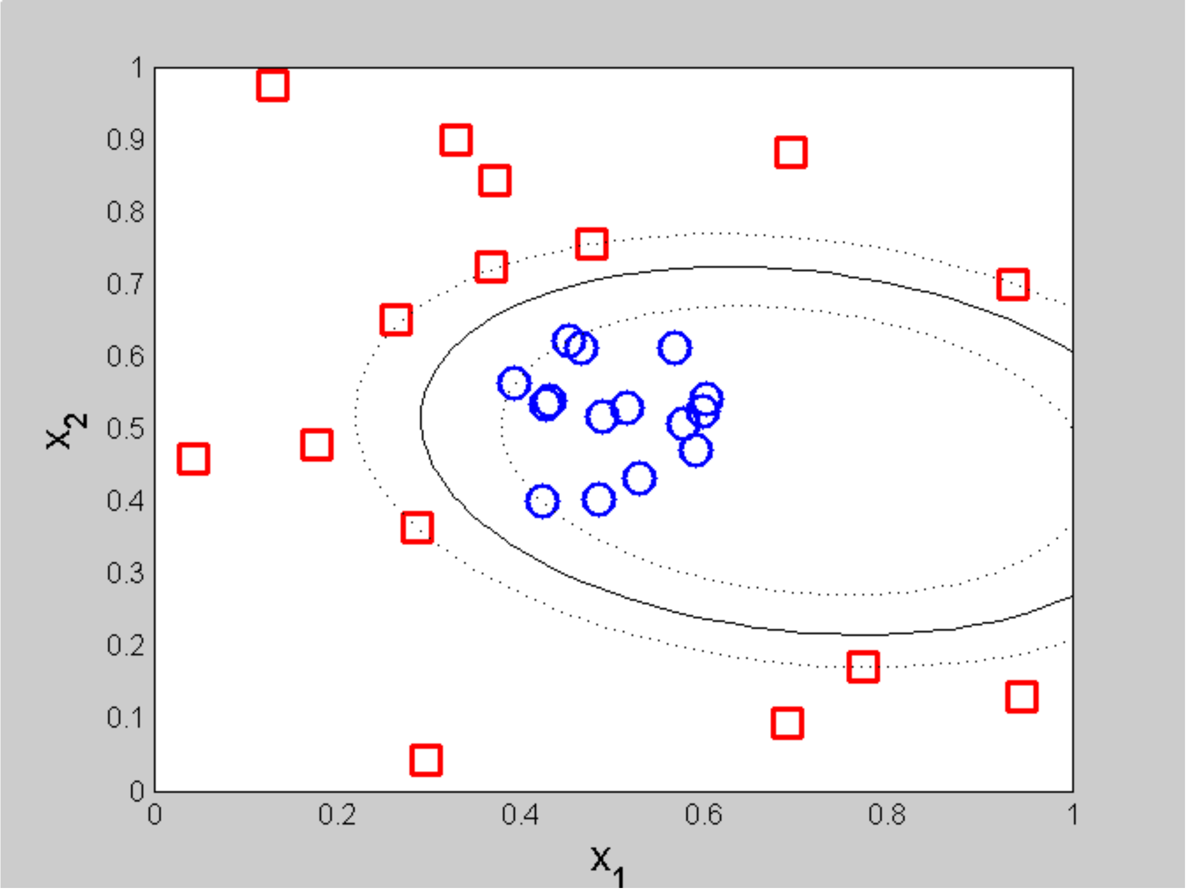
- Kernel Trick:
  - $\Phi(x_i)$□$\Phi(x_j) = K(x_i, x_j)$
  - $K(x_i, x_j)$ is a kernel function (expressed in terms of the coordinates in the original space)
    - Examples:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$$

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

$$K(\mathbf{x}, \mathbf{y}) = \tanh(k\mathbf{x} \cdot \mathbf{y} - \delta)$$

Example of Nonlinear SVM



SVM with polynomial degree 2 kernel

Learning Nonlinear SVM

- Advantages of using kernel:
  - Don’t have to know the mapping function Φ
  - Computing dot product $\Phi(x_i) \cdot \Phi(x_j)$ in the original space avoids curse of dimensionality

- Not all functions can be kernels
  - Must make sure there is a corresponding Φ in some high-dimensional space
  - Mercer’s theorem (see textbook)

Characteristics of SVM

- Since the learning problem is formulated as a convex optimization problem, efficient algorithms are available to find the global minima of the objective function (many of the other methods use greedy approaches and find locally optimal solutions)
- Overfitting is addressed by maximizing the margin of the decision boundary, but the user still needs to provide the type of kernel function and cost function
- Difficult to handle missing values
- Robust to noise
- High computational complexity for building the model