POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Week 9-2

Machine Learning 4: Linear Classification, ANN, Linear SVM

Big Data

Prof. Hwanjo Yu
POSTECH

POSTECH
POHANG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Linear Classification, Artificial Neural Network (ANN)

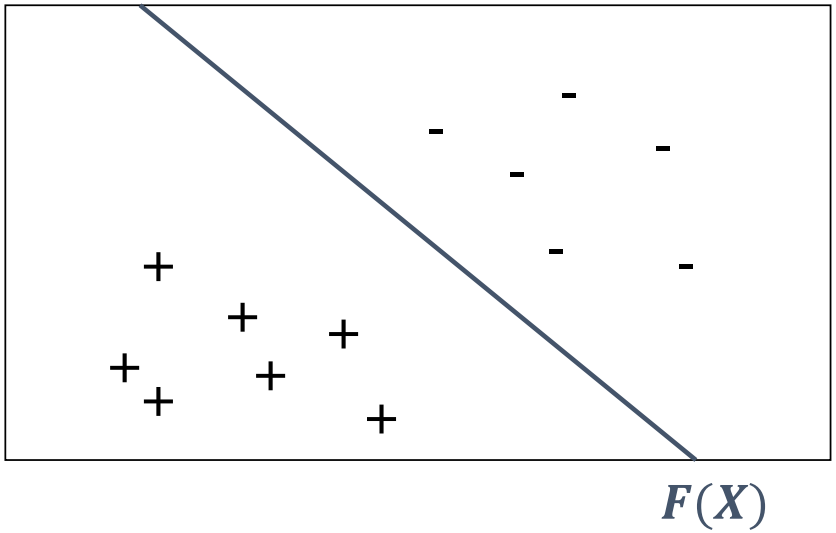
Big Data

Linear discriminant function

- A linear function classifies an object (n-dimensional vector) $X = [x_1, x_2, \dots x_n]$ based on

$$F(X) = b + \sum_{i}^{n} w_i x_i$$

  - b is bias
  - w_i is the weight of a feature x_i
  - X is positive If $F(X) > 0$
  - X is negative if $F(X) < 0$



Linear vs. Quadratic vs. Polynomial

- Linear: a sum of weighted features

$$F(X) = b + \sum_i w_i x_i$$

- Quadratic: considers also concatenations of two features

$$F(X) = b + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i \, x_j$$

- Polynomial: considers concatenations of multiple features

  (degree p denotes the size of concatenations)

$$F(X) = b + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i \, x_j + \sum_{ijk} w_{ijk} x_i \, x_j x_k + \cdots$$

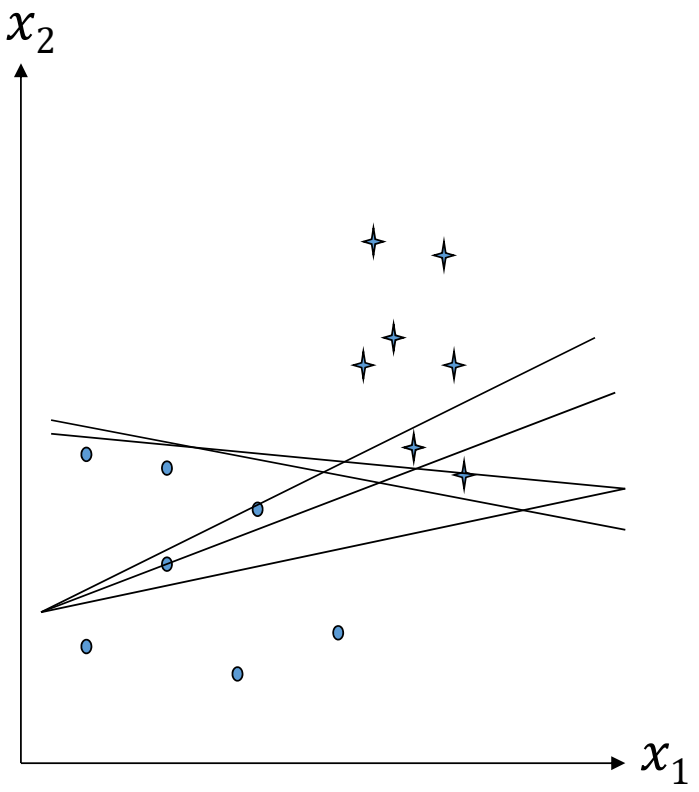
- Linear => Polynomial with degree = 1
- Quadratic => Polynomial with degree = 2

Perceptron (and Winnow)

- Perceptron (and Winnow): Algorithms to learn a linear function

$$F(X) = b + \sum_{i}^{n} w_i x_i$$

- Steps:
  1. Initiate F by setting w and b randomly
  2. For each training point x_i, If F misclassifies x_i, adjust w and b so that new F correctly classifies x_i
  3. Repeat Step 2



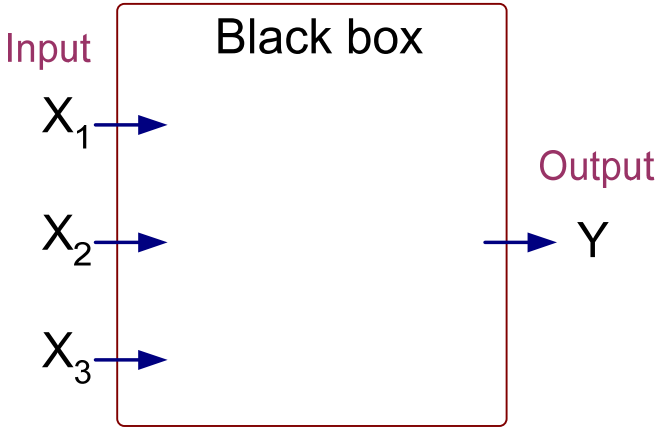
Perceptron and Winnow

- Nondeterministic
  - For the same training set, they could generate different F depending the data ordering.
  - Several hyper-parameters to tune such as # of iterations, how much adjust w.
- Limited to linear functions
  - # of w for polynomial function is too large to train in a nondeterministic way
  - Inherently requires $O(n^p)$ to learn a polynomial function
- SVM can learn a polynomial function in $O(n)$

Artificial Neural Networks (ANN) : Perceptron

Output Y is 1 if at least two of the three inputs are equal to 1.

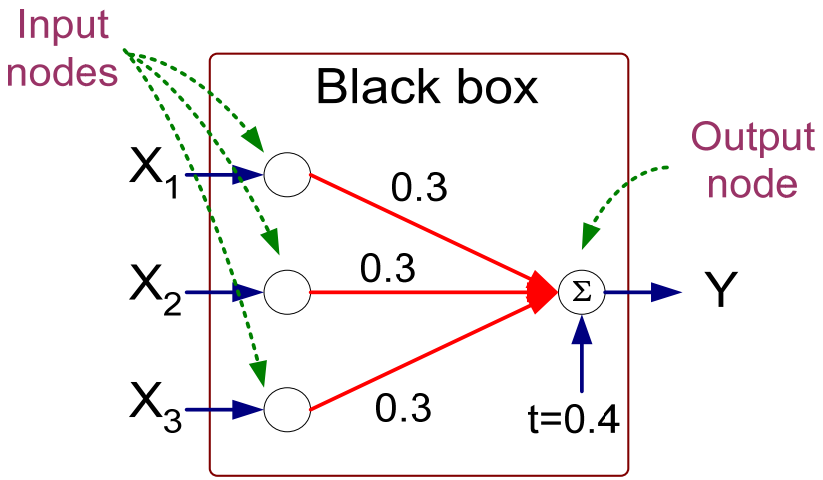
X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Artificial Neural Networks (ANN) : Perceptron

Output Y is 1 if at least two of the three inputs are equal to 1.

X_1	X_2	X_3	Y
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	0
0	1	0	0
0	1	1	1
0	0	0	0



Perceptron model :

$$Y = I(\sum_i w_i x_i - t)$$

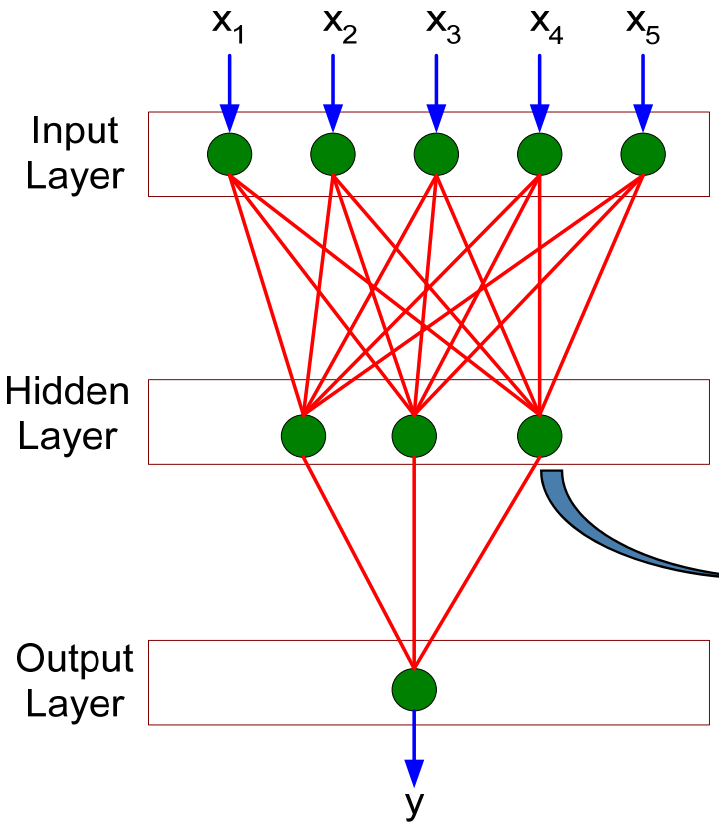
A Neuron :

$$Y = sigmoid(\sum_i w_i x_i - t)$$

$$Y = I(0.3x_1 + 0.3x_2 + 0.3x_3 - 0.4 > 0)$$

where $I(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{otherwise} \end{cases}$

General structure of ANN

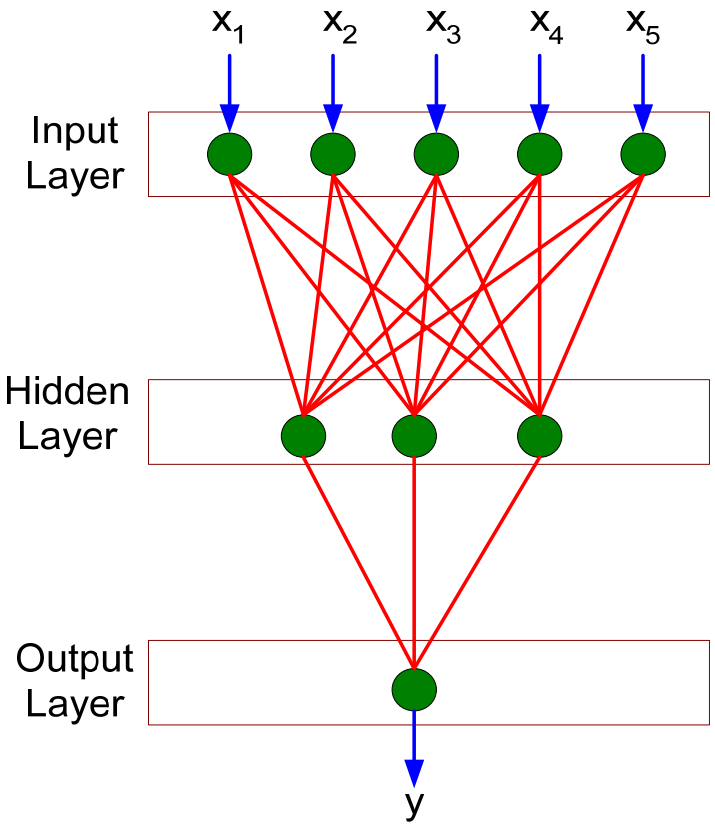


A Neuron :

$$Y = sigmoid(\sum_i w_i x_i - t)$$

Training ANN means learning the weights of the neurons

Learning ANN



$$f(X) = S(\sum_b w_b S(\sum_a w_a x_a - t_b) - t)$$

- Initialize the weights $(w_0, w_1, \ldots)$
- Minimize error by *Backpropagation*
  - Input each tuple X_i
  - Compute error $E = \sum_i (Y_i - f(X_i))^2$
  - Update weights backward to minimize E
  - Repeat
- Each *epoch* (one iteration through the training set) takes $O(|D|*|w|)$, with $|D|$ tuples and $|w|$ weights

ANN: Output encoding

- Single output nodes applicable when target classes ordered
- For example, classify elementary-level reading ability

Define thresholds	Classify
“if 0.00 <= output < 0.25”	“first-grade”
“if 0.25 <= output < 0.50”	“second-grade”
“if 0.50 <= output < 0.75”	“third-grade”
“if output >= 0.75”	“fourth-grade”

- Fine-tuning of thresholds may be required

ANN: Output encoding

- Single output node not applicable to all classification problems
- For example, target variable is Marital_Status, unordered categories
- Use 1-of-n Encoding, with single output node for each target class
- Network has six output nodes for values “Divorced”, “Married”, “Separated”, “Single”, “Widowed”, and “Unknown”
- Output node with highest value chosen for classification
- Approach provides measure of confidence in classification
- Confidence is difference between highest and second-highest values in output nodes

ANN

- Nonlinear model
  - Produce nonlinear classification functions $f(X) = S(\sum_b w_b S(\sum_a w_a x_a - t_b) - t)$
  - Poor interpretability
- Nondeterministic
  - Topology (or structure) must be designed based on the data type and problem
    - Too many hidden layer nodes could result in overfitting
  - Many parameters to tune
    - Topology, # of iterations, how much adjust w, etc.
  - Works on continuous attributes => a nominal attribute must be converted to binary attributes
    - E.g. A color attribute of {red, blue, grey} is converted to three binary attributes, that are, red_color = {0,1}, blue_color = {0,1}, and grey_color = {0,1}

Support Vector Machine (SVM)

Big Data

SVM vs. ANN

SVM

- Deterministic algorithm
- Nice Generalization properties with few parameters to tune
- Hard to learn – learned in batch mode using quadratic programming techniques
- Using kernels can learn very complex functions

ANN

- Nondeterministic algorithm
- Generalizes well but need a lot of tuning
- Can be learned in incremental fashion
- To learn complex functions
  —use multilayer perceptron

Linear vs Quadratic vs Polynomial

- Linear: a sum of weighted features

$$F(X) = b + \sum_i w_i x_i$$

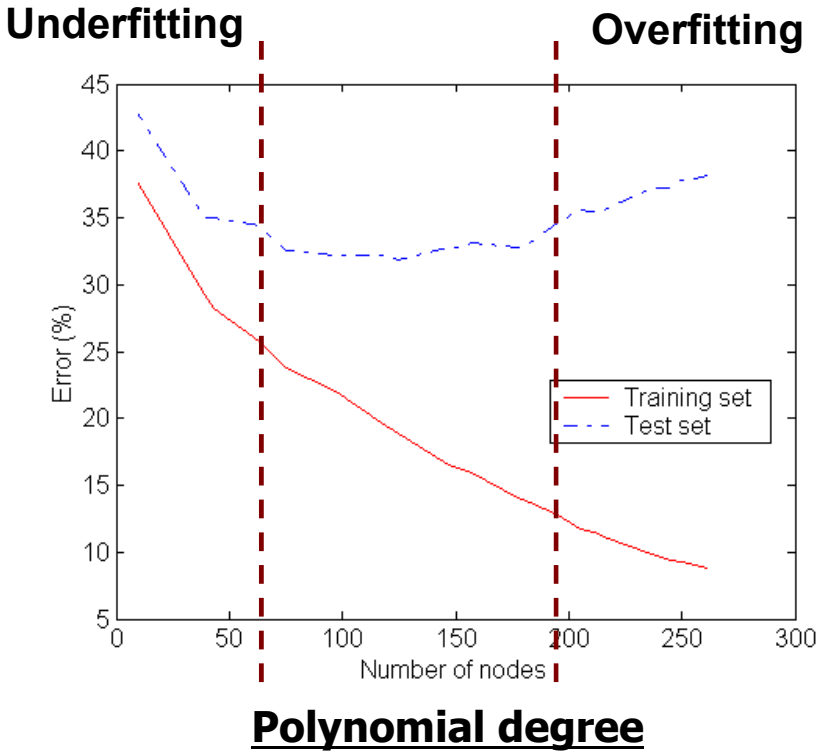
- Quadratic: considers also concatenations of two features

$$F(X) = b + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j$$

- Polynomial: considers concatenations of multiple features

(degree p denotes the size of concatenations)

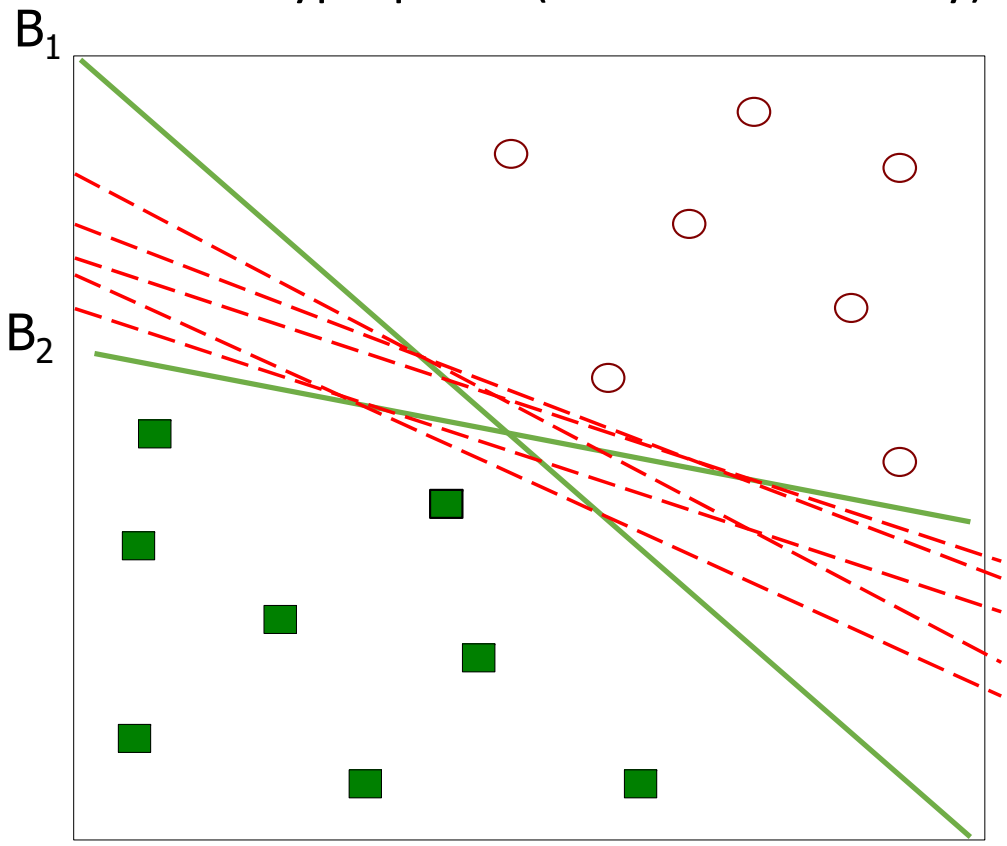
$$F(X) = b + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j + \sum_{ijk} w_{ijk} x_i x_j x_k + \cdots$$



SVM learns a polynomial function with an arbitrary degree in linear time

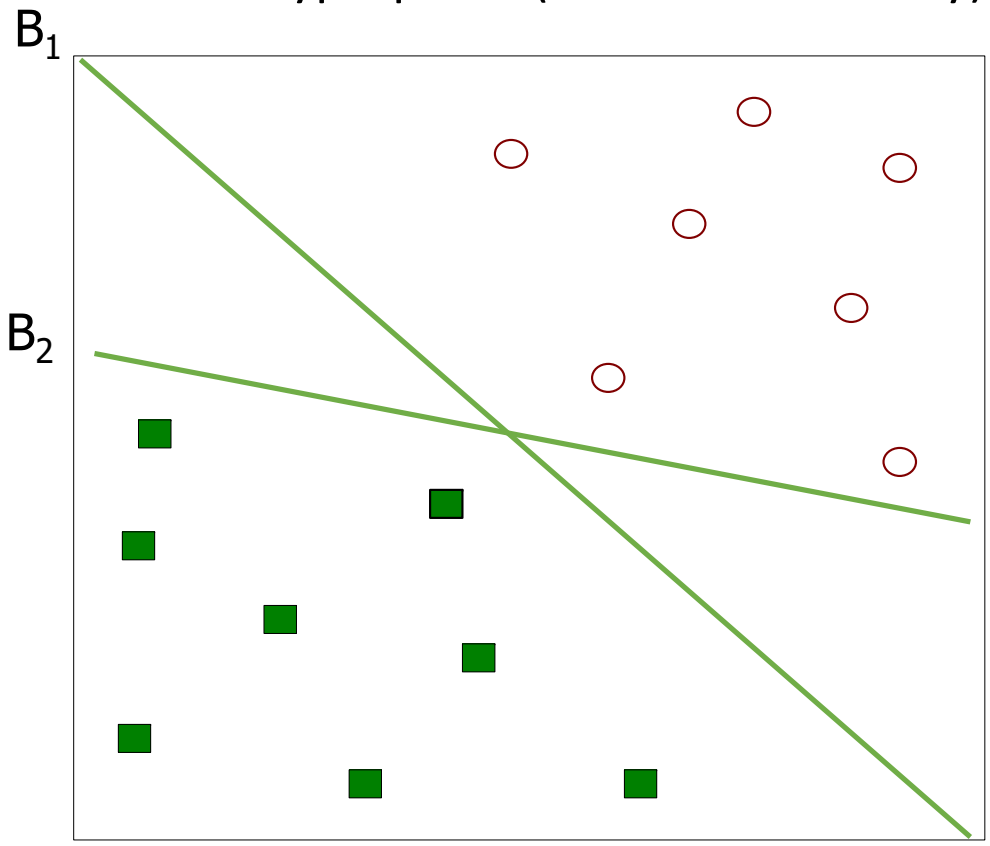
Linear support vector machines

- Find a linear hyperplane (decision boundary) that separates the data: $F(X) = b + WX$



Linear support vector machines

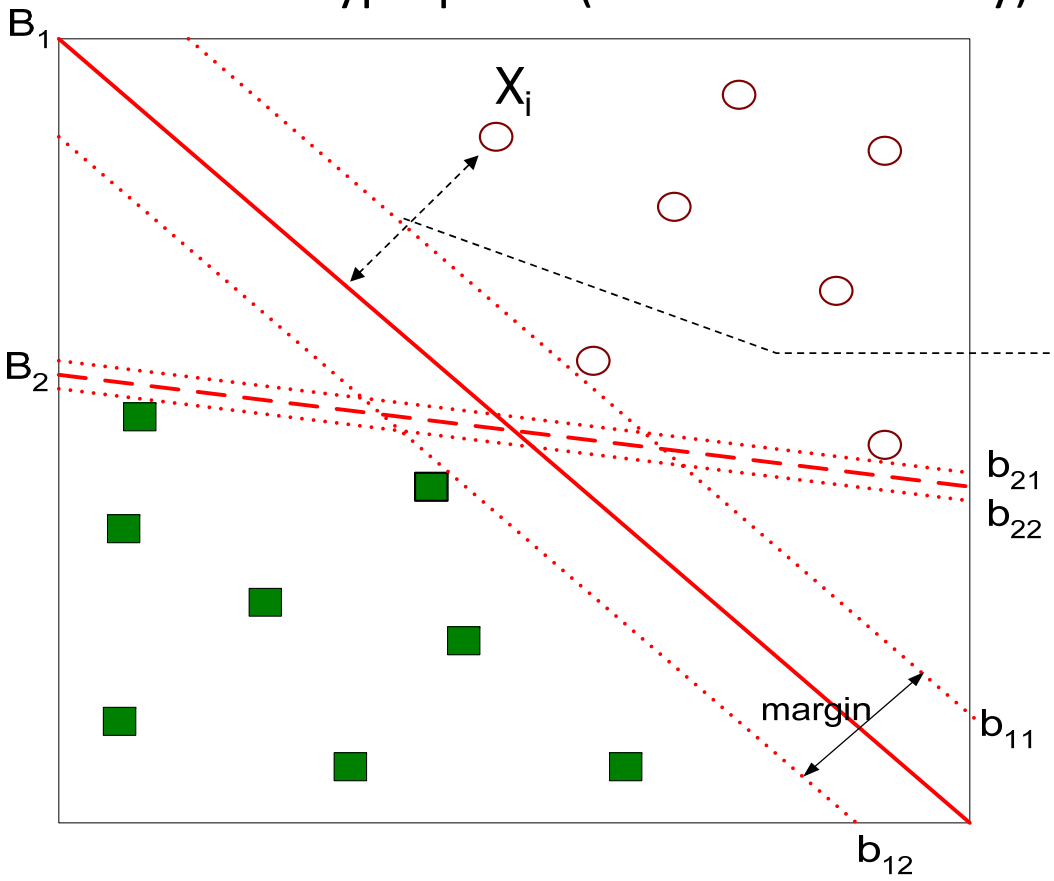
- Find a linear hyperplane (decision boundary) that separates the data: $F(X) = b + WX$



- Which one is better? B1 or B2?
- How do you quantify the “goodness”?

Linear support vector machines

- Find a linear hyperplane (decision boundary) that separates the data: $F(X) = b + WX$



- Which one is better? B1 or B2?
- How do you quantify the "goodness"?

$$F(X_i)/||W||$$

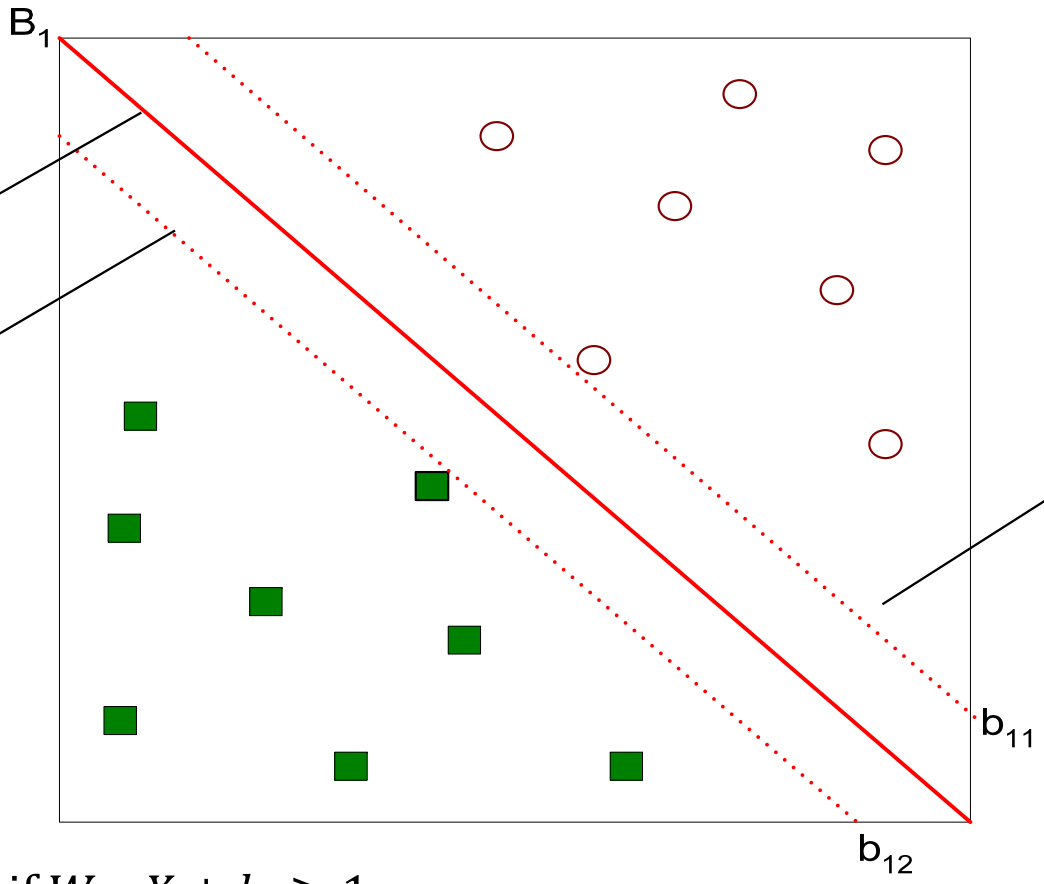
- Find hyperplane maximizes the margin
  => B1 is better than B2

Linear support vector machines

- Given a labeled dataset $\mathrm{D}=\{(X_1, y_1), \ldots, (X_m, y_m)\}$ where X_i is a data vector, and y_i is class label $(+1 \text{ or } -1)$ of X_i

- D is "linearly separable",
  - if there exists a linear function $F(X) = b + WX$ that correctly classifies every data vector in D, that is,
  - if there exists b and W that satisfies $y_i(b + WX_i) > 0$ for all $(X_i, y_i) \in \mathrm{D}$, where b is a scalar and W is a weight vector.
  - Then, there exist b' and W' that satisfies $y_i(b' + W'X_i) \geq 1$ for all $(X_i, y_i) \in \mathrm{D}$?

Linear support vector machines



$W \bullet X + b = 0$

$W \bullet X + b = -1$

$W \bullet X + b = +1$

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

$$Margin = \frac{1}{\|W\|}$$

Linear support vector machines

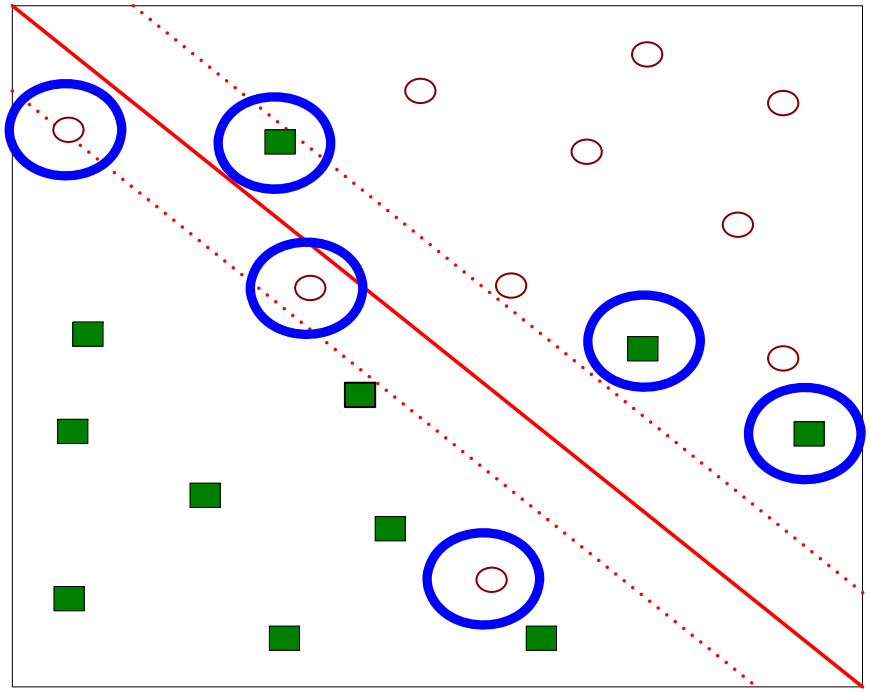
- We want to maximize: $Margin = \frac{1}{\|W\|}$
- Instead, we minimize: $L(W) = \frac{\|W\|^2}{2}$
- But subjected to the following constraints:

$$y_i(W \bullet X_i + b) \geq 1, \text{for all } (X_i, y_i) \in \text{D}$$

- This is a constrained optimization problem
- Numerical approaches to solve it (e.g. quadratic programming)

Linear support vector machines

- What if the problem is not linearly separable?



Linear support vector machines

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

$$L(W, \xi) = \frac{\|W\|^2}{2} + C\left(\sum_{i=1}^{m} \xi_i\right)$$

  - Subject to:

$$y_i(W \bullet X_i + b) \geq 1 - \xi_i, \text{for all } (X_i, y_i) \in \mathrm{D}$$

Linear support vector machines

- Primal form:
  - Minimize:

$$L(W, \xi) = \frac{\|W\|^2}{2} + C\left(\sum_{i=1}^{m} \xi_i\right)$$

  - Subject to:

$$y_i(W \bullet X_i + b) \geq 1 - \xi_i, \text{for all } (X_i, y_i) \in \mathrm{D}$$

  - To solve this, transform the primal to the dual: see the next slide

Linear support vector machines

- Dual form:

$$\text{max. } M(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1,j=1}^{m} \alpha_i \alpha_j y_i y_j X_i^T X_j$$

$$\text{Subject to } C \geq \alpha_i \geq 0, \ \sum_{i=1}^{m} \alpha_i y_i = 0$$

- W can be recovered by $W = \sum_{i=1}^{m} \alpha_i y_i X_i$

Characteristics of the solution

- Many of the α_i are zero
  - w is a linear combination of a small number of data points

- X_i with non-zero α_i are called support vectors (SV)
  - The decision boundary is determined only by the SV

$$W = \sum_{i=1}^{m} \alpha_i y_i X_i$$

- For testing with a new data Z
  - Compute $F(Z) = b + \sum \alpha_i y_i X_i Z$ and classify Z as class 1 if the sum is positive, and class -1 otherwise
  - Note: w need not be formed explicitly

SVM model

- SVM model
  - Bias b, a list of support vectors and their coefficients α
- Where is ξ and How C affects the model?
- Why don't we compute w explicitly?