# Data Mining
# Classification: Alternative Techniques

Lecture Notes for Chapter 5

(PART 2)

정보보호 및 지능형 IoT 연구실
Information Security & Intelligent IoT

# Agenda

- Rule Based Classifier
- Bayesian Classifier

PART 1

- Artificial Neural Network
- Support Vector Machine
- Ensemble, Bagging, Boosting

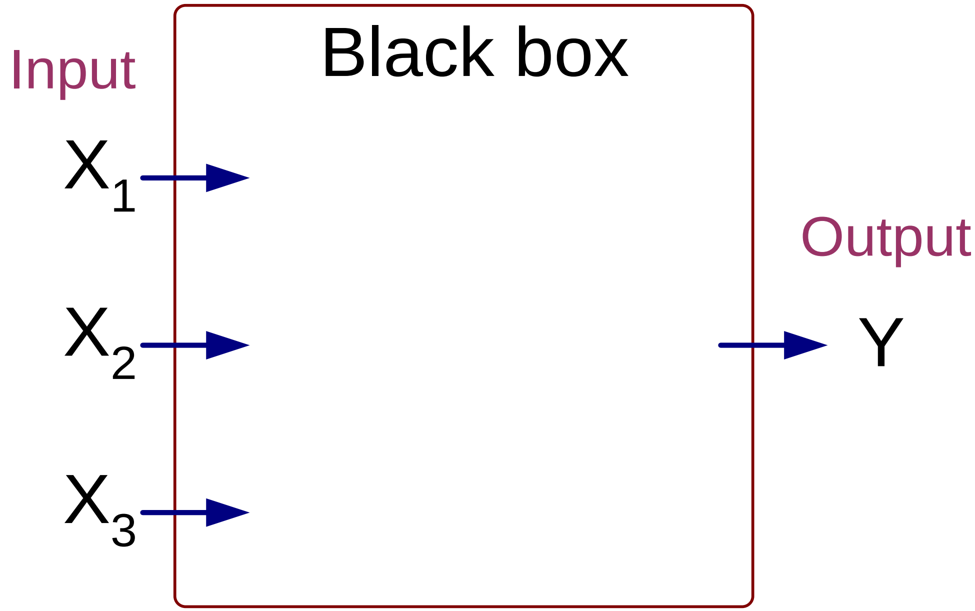
PART 2

# Contents

Artificial Neural Network

# Artificial Neural Networks (ANN)

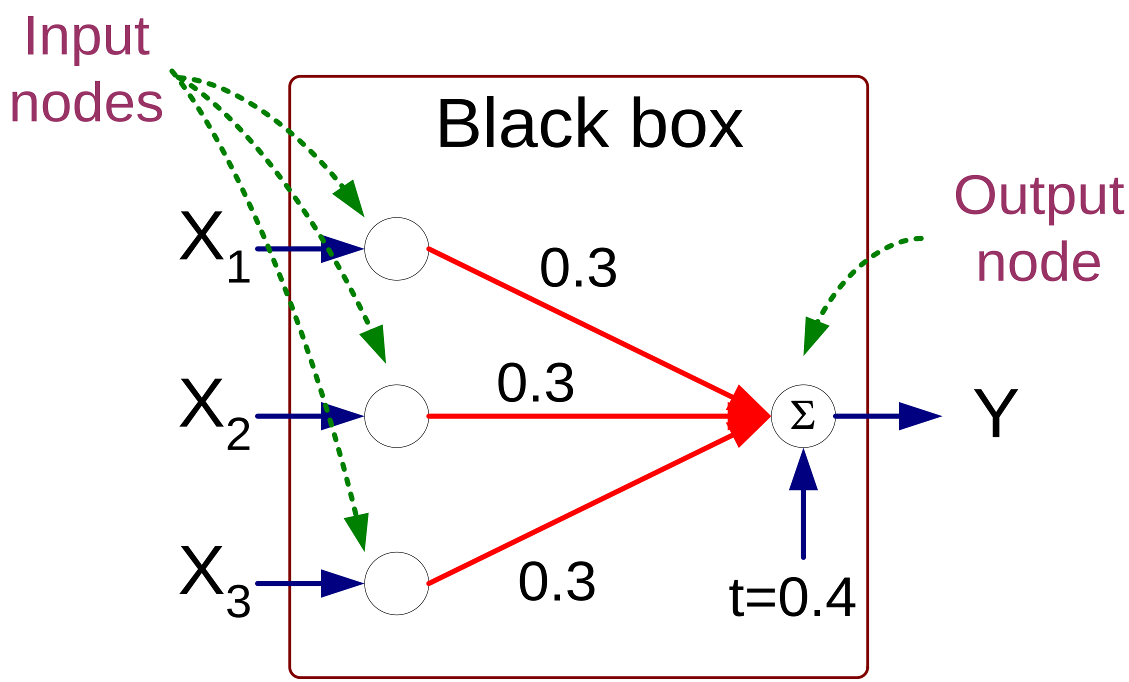
| $X_1$ | $X_2$ | $X_3$ | Y |
|---|---|---|---|
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 |




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

# Artificial Neural Networks (ANN)

| $X_1$ | $X_2$ | $X_3$ | 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 |

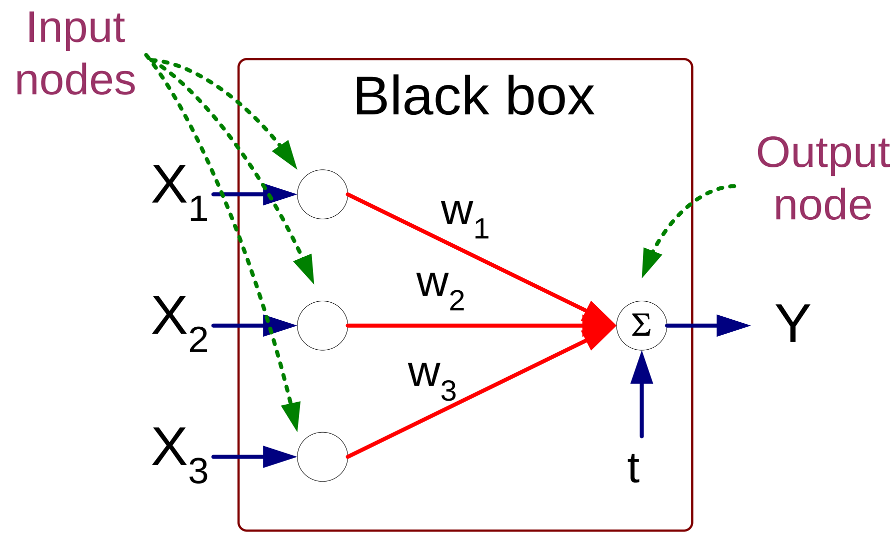



$$Y = I(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4 > 0)$$

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

# Artificial Neural Networks (ANN)

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t

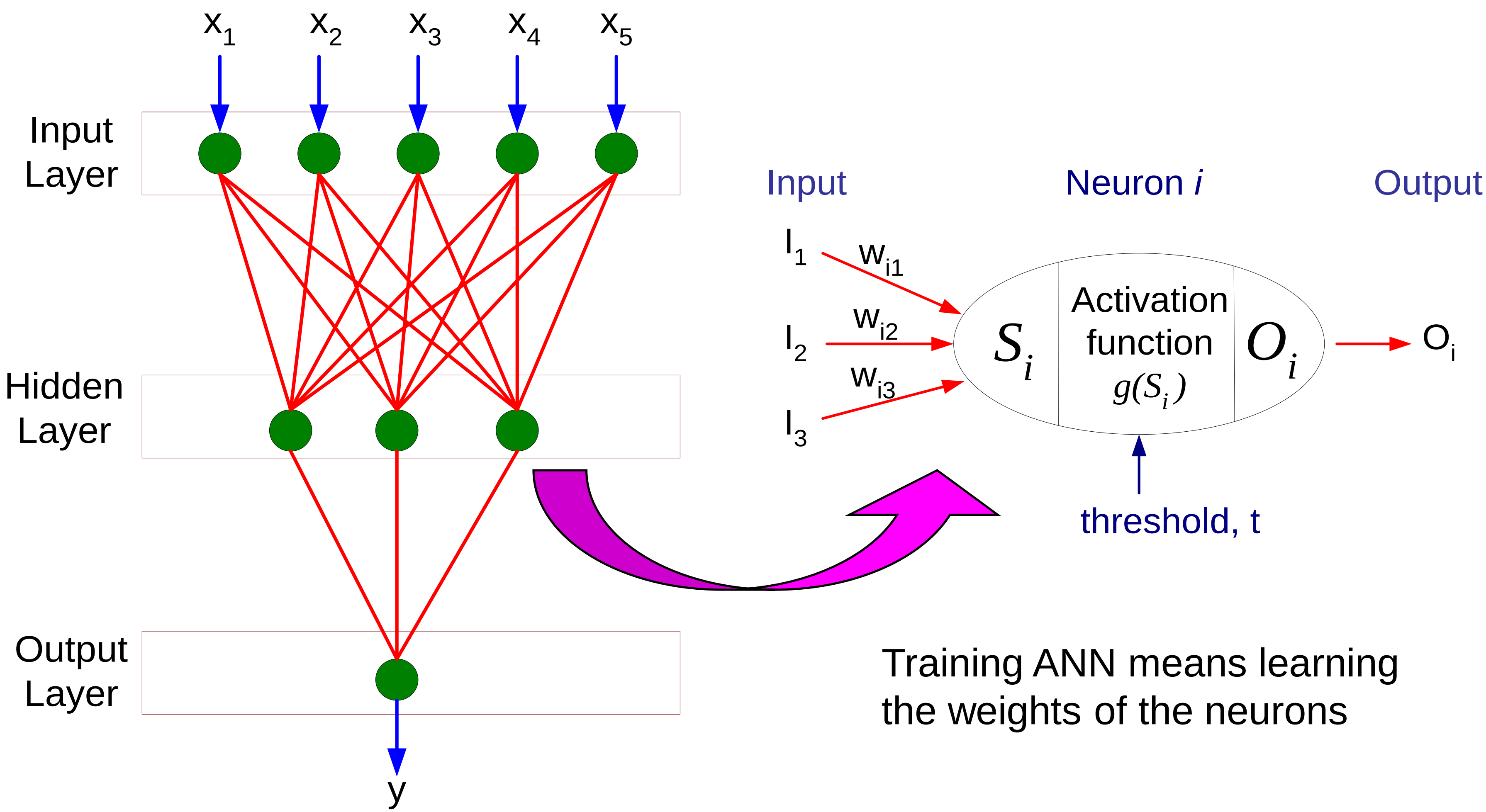



**Perceptron Model**

$$Y = I(\sum_i w_i X_i - t) \quad \text{or}$$

$$Y = sign(\sum_i w_i X_i - t)$$

# General Structure of ANN




Training ANN means learning the weights of the neurons

# Algorithm for learning ANN

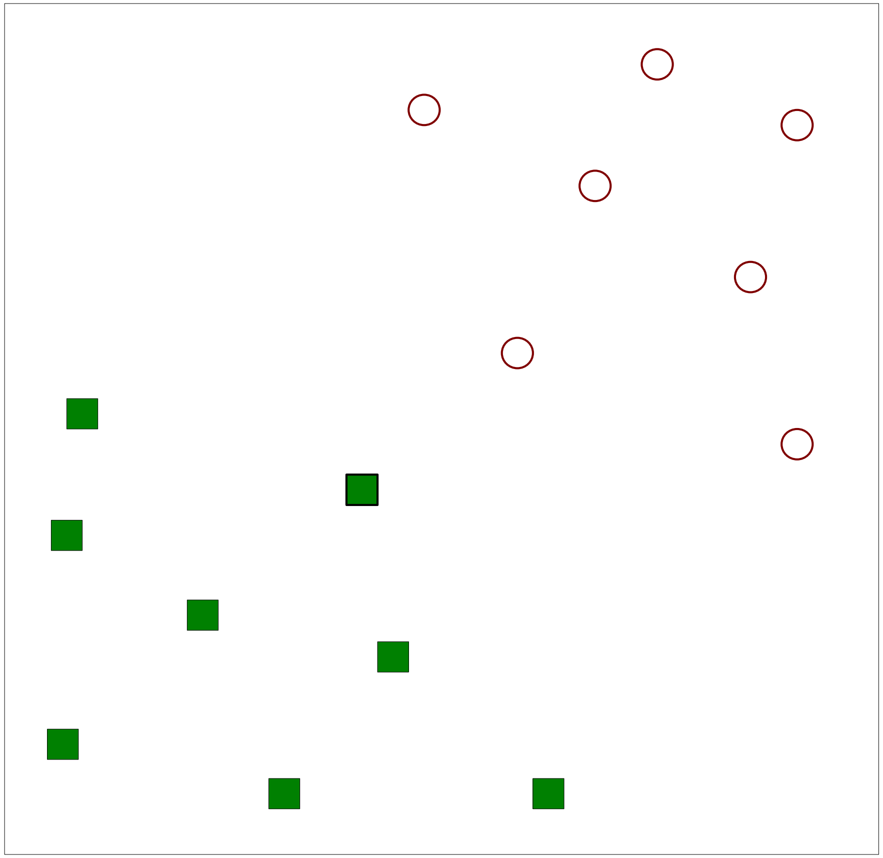
- Initialize the weights ($w_0$, $w_1$, …, $w_k$)

- Adjust the weights in such a way that the output of ANN is consistent with class labels of training examples
  - Objective function: $E = \sum_i [Y_i - f(w_i, X_i)]^2$
  - Find the weights $w_i$'s that minimize the above objective function
    - e.g., backpropagation algorithm

# Contents

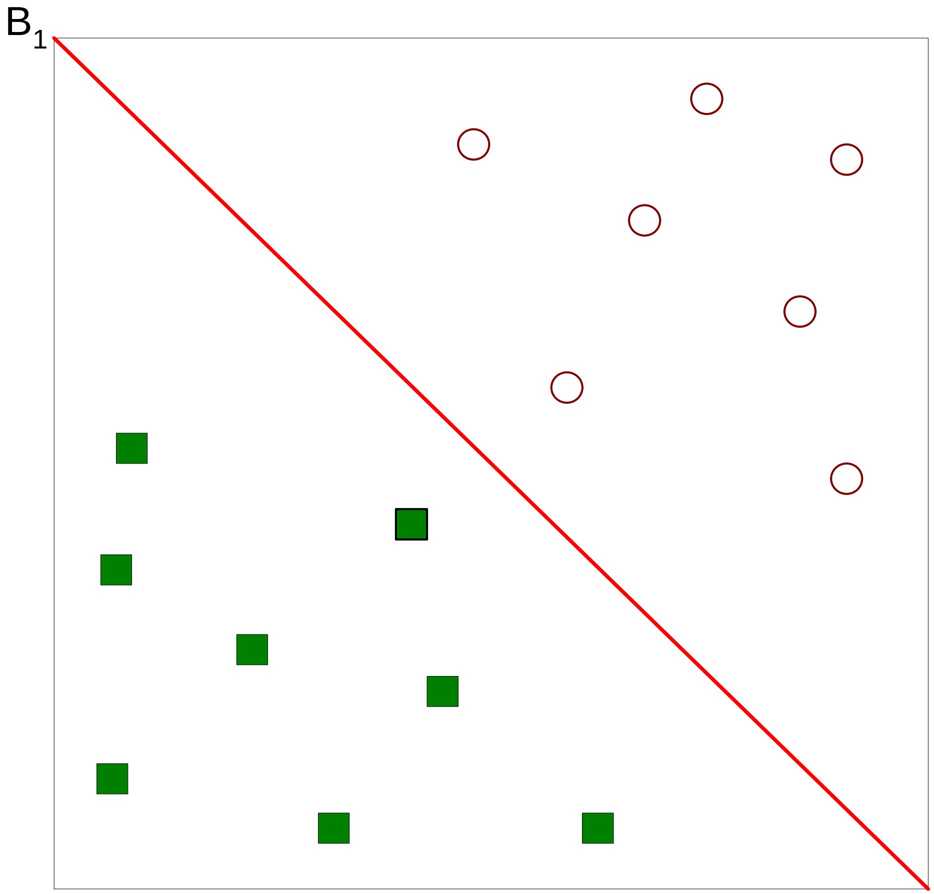
Support Vector Machine

# Support Vector Machines



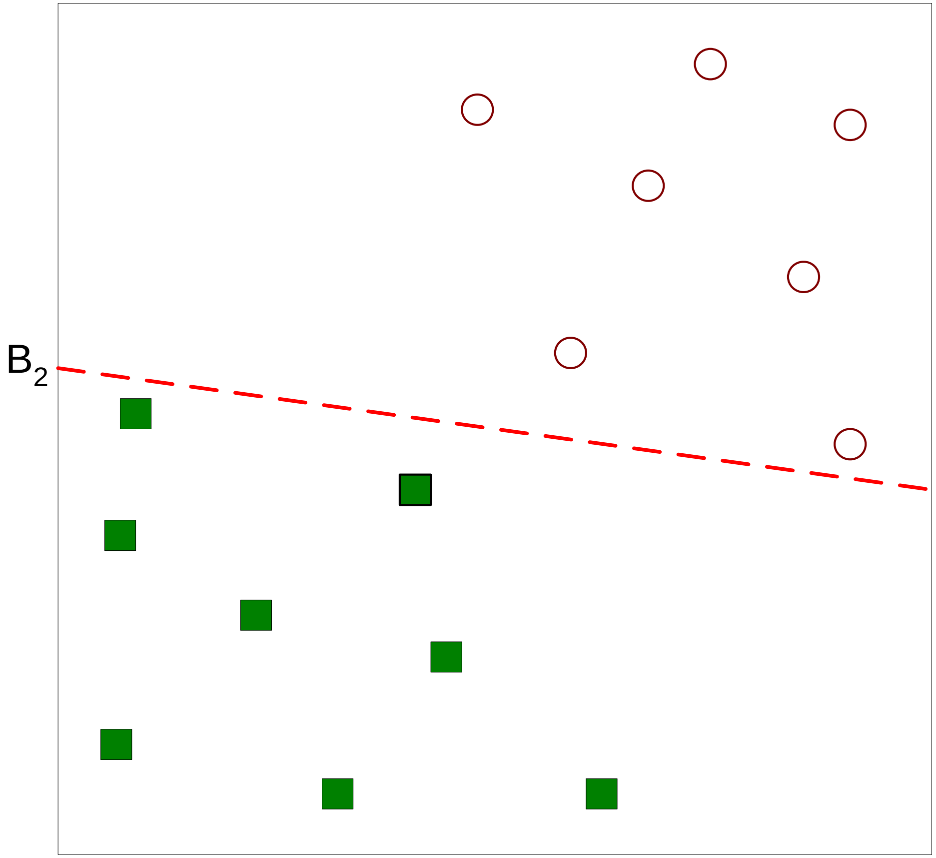
- Find a linear hyperplane (decision boundary) that will separate the data

# Support Vector Machines



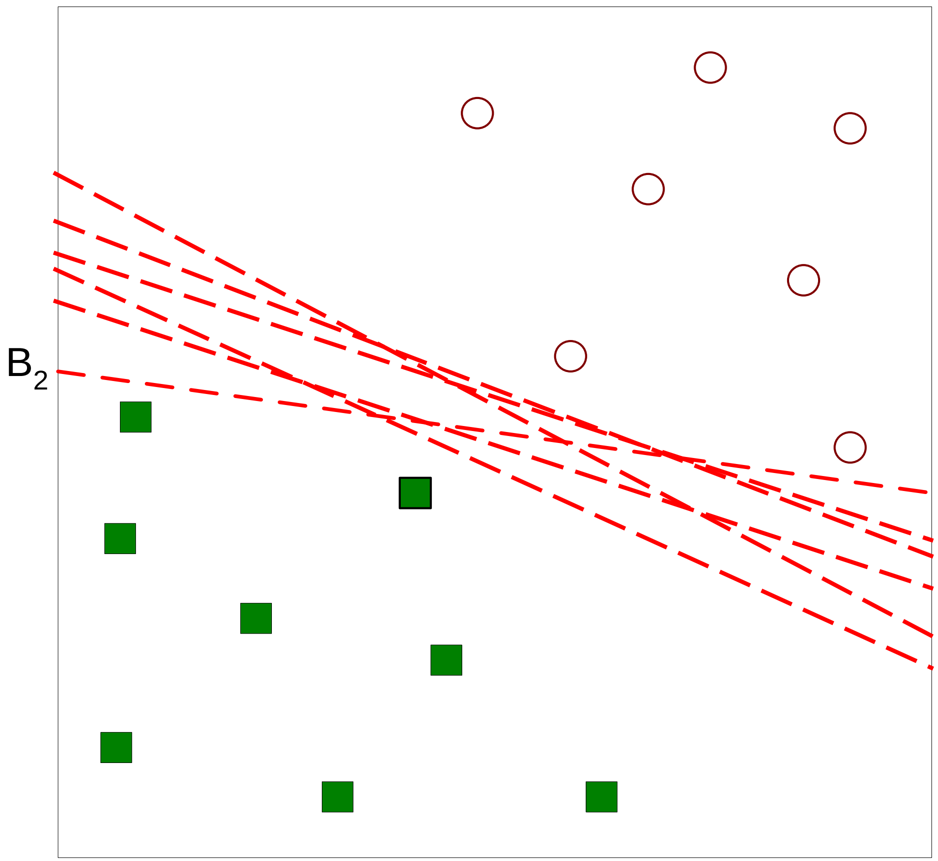

- One Possible Solution

# Support Vector Machines



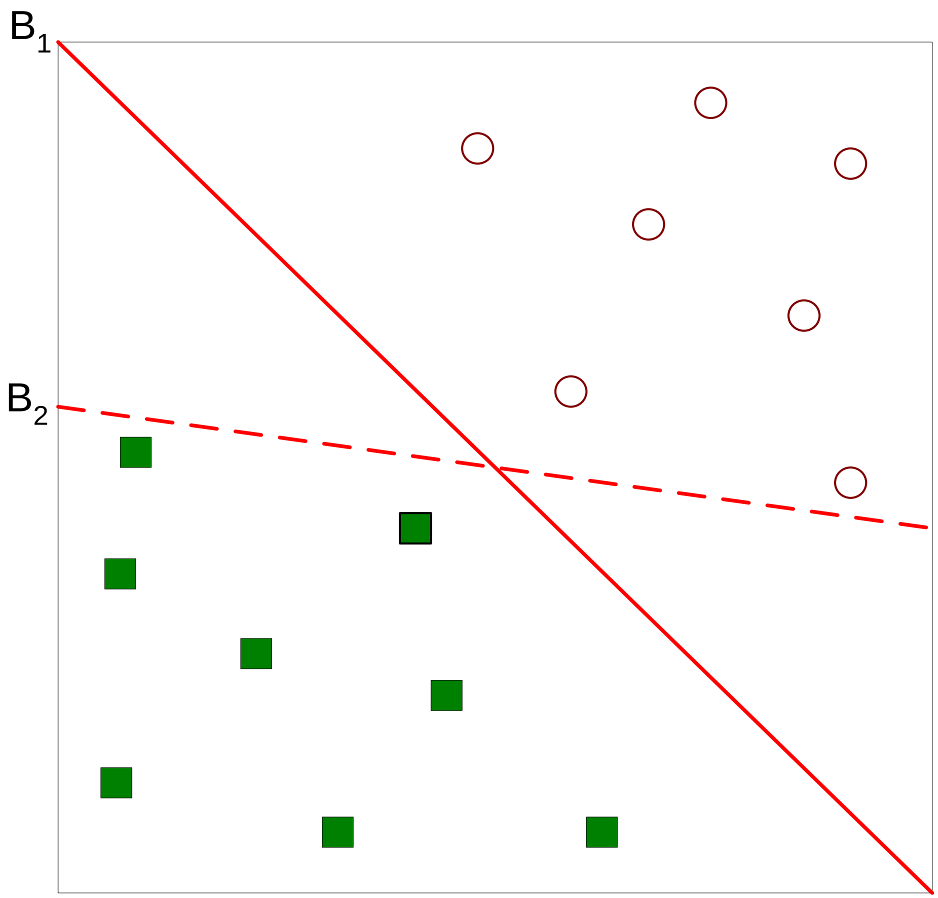

- Another possible solution

# Support Vector Machines



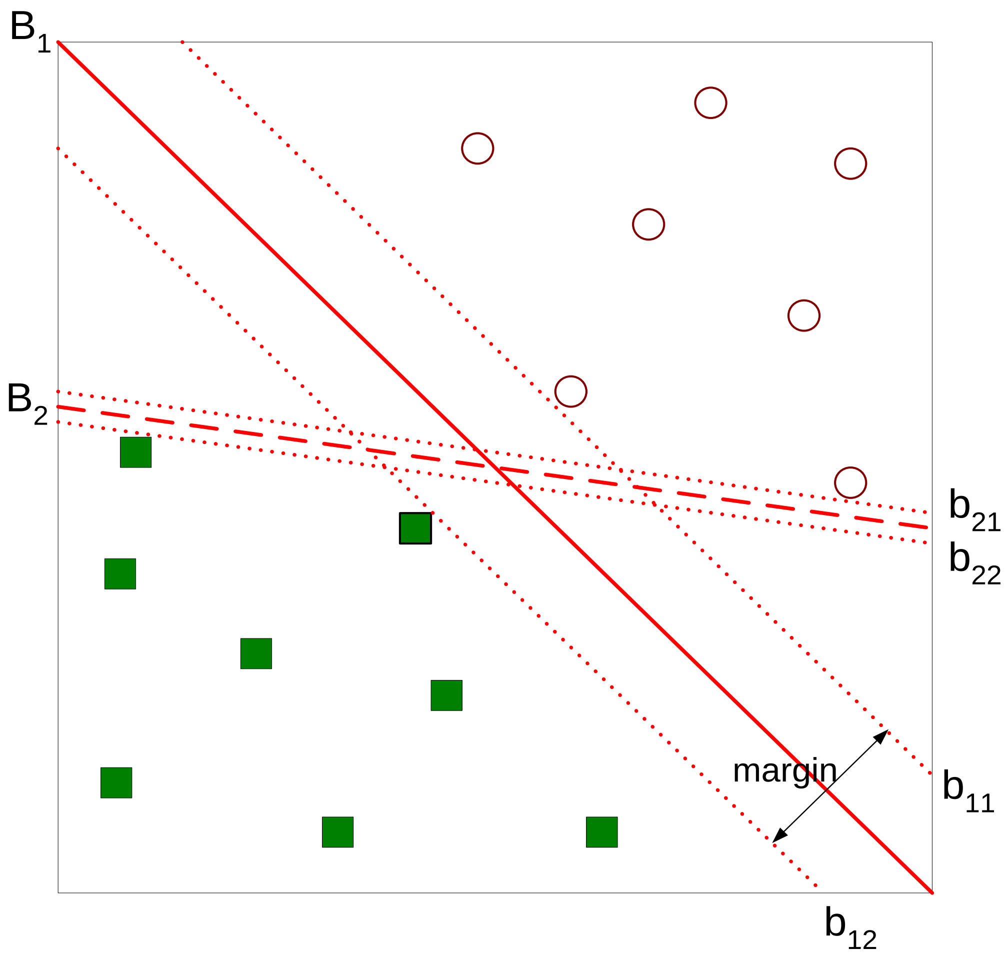

- Other possible solutions

# Support Vector Machines



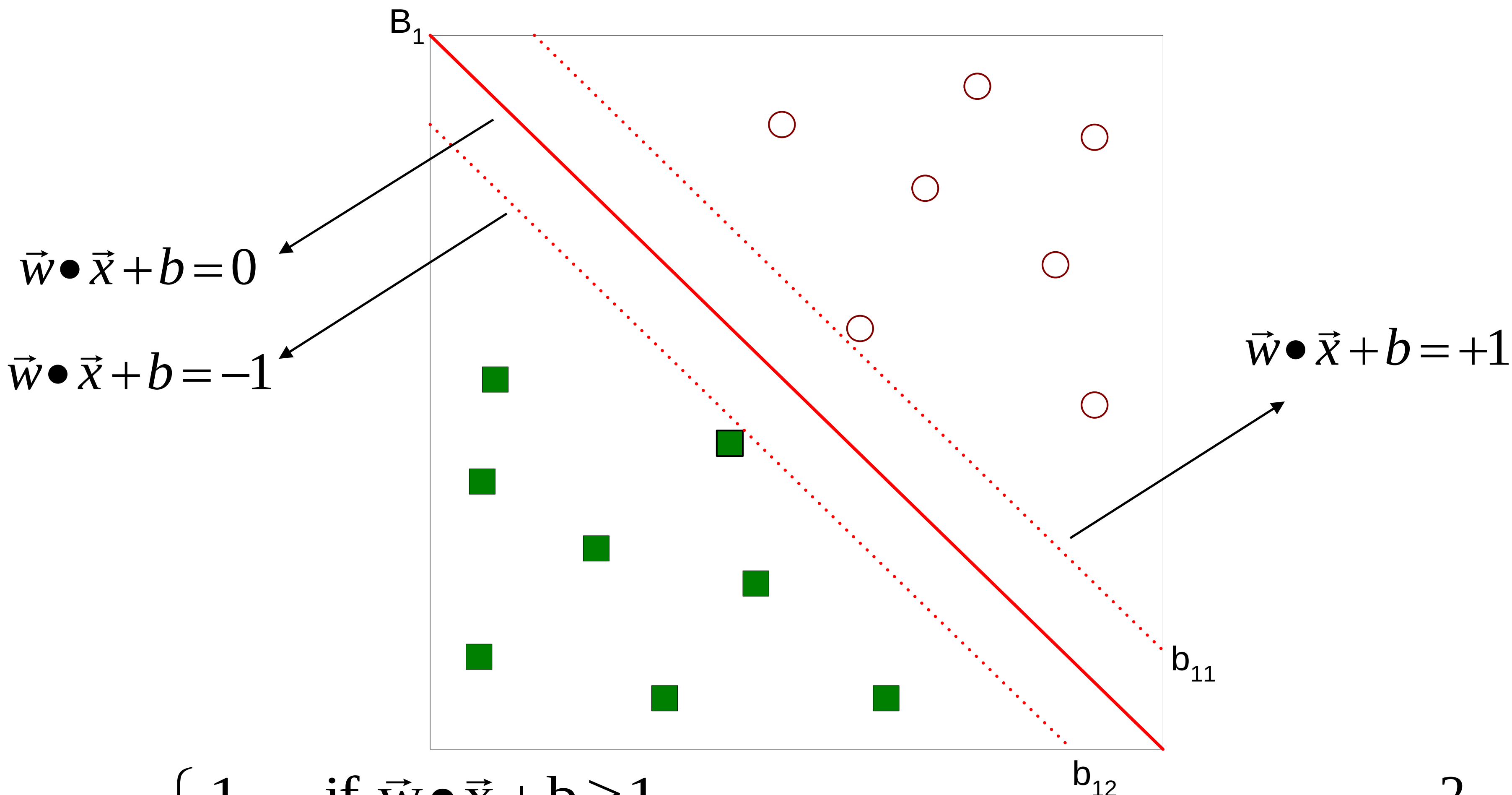

- Which one is better? B1 or B2?
- How do you define better?

# Support Vector Machines




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

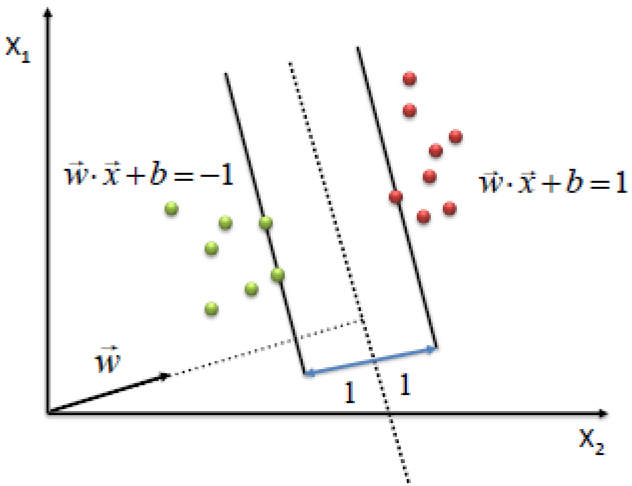
# Support Vector Machines




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

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

# Support Vector Machines

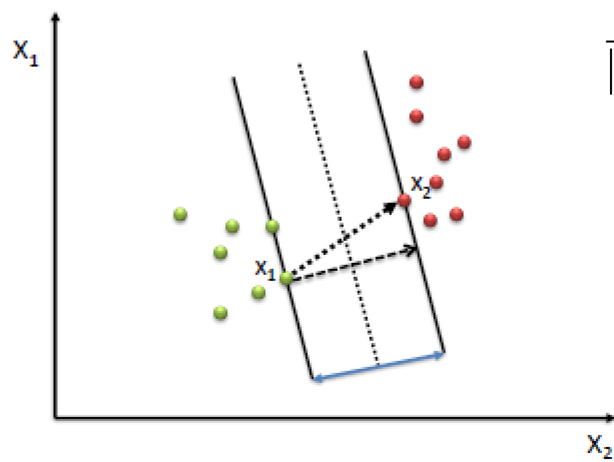



$$\max \frac{2}{\|w\|}$$

s.t.

$(w\cdot x+b)\geq 1, \forall x$ of class 1

$(w\cdot x+b)\leq -1, \forall x$ of class 2




$$\frac{w}{\|w\|}\cdot(x_2-x_1)=\text{width}=\frac{2}{\|w\|}$$

$$w\cdot x_2+b=1$$

$$w\cdot x_1+b=-1$$

$$w\cdot x_2+b-w\cdot x_1-b=1-(-1)$$

$$w\cdot x_2-w\cdot x_1=2$$

$$\frac{w}{\|w\|}(x_2-x_1)=\frac{2}{\|w\|}$$



http://www.saedsayad.com/support_vector_machine.htm

# Support Vector Machines

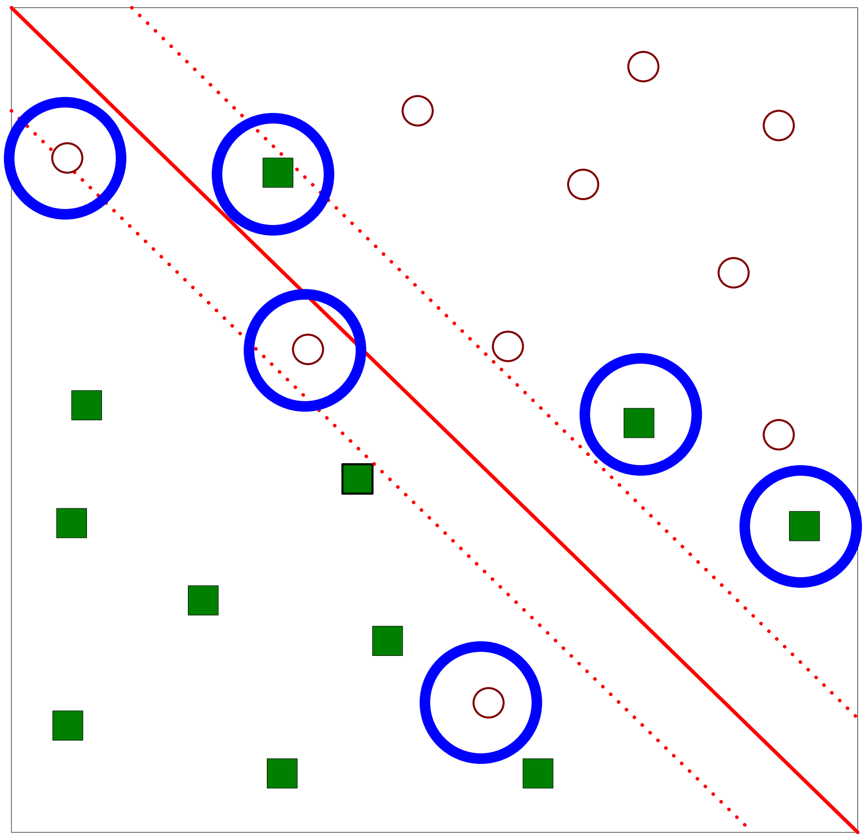
- We want to maximize: $\text{Margin} = \frac{2}{\|\vec{w}\|^2}$
  - Which is equivalent to minimizing: $L(w) = \frac{\|\vec{w}\|^2}{2}$
  - But subjected to the following constraints:

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

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

# 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:

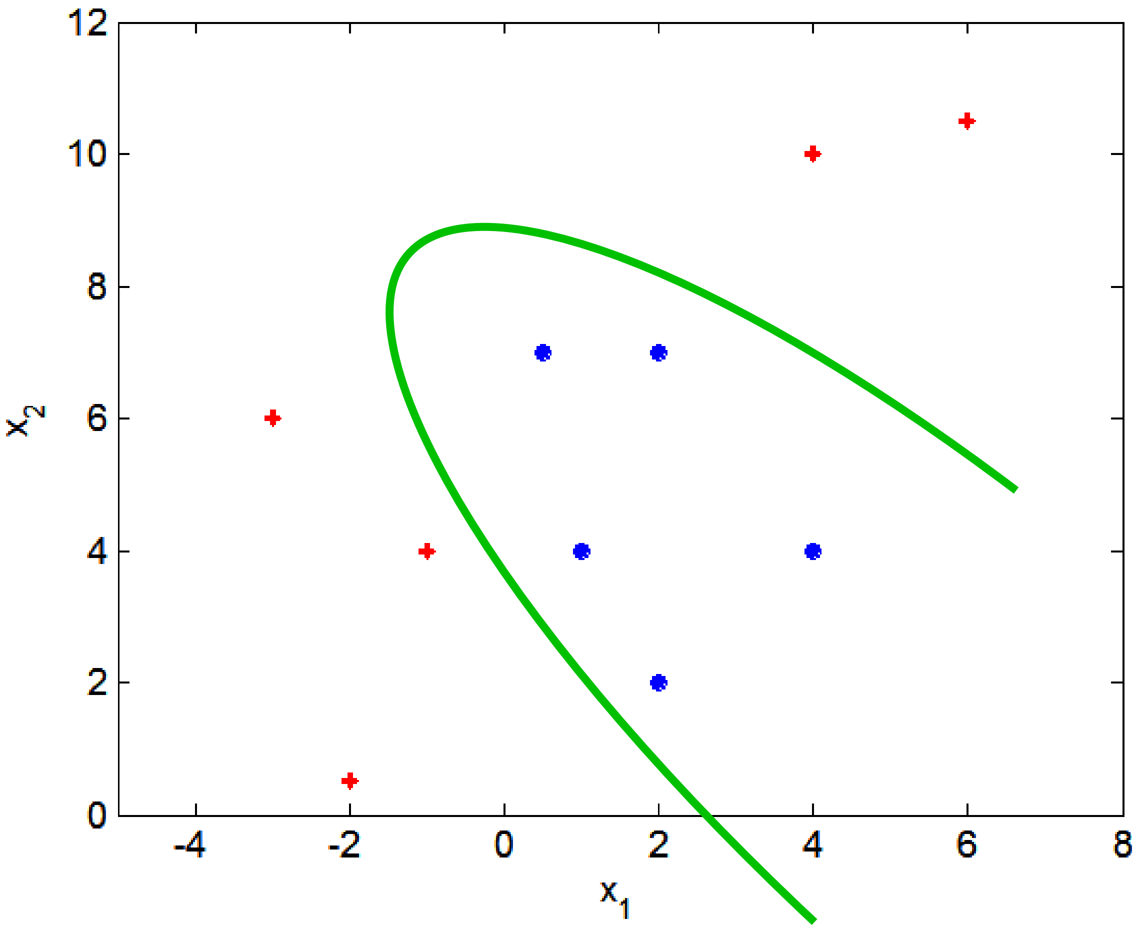
$$L(w) = \frac{\|\vec{w}\|^2}{2} + C\left(\sum_{i=1}^{N} \xi_i^k\right)$$

    - Subject to:

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

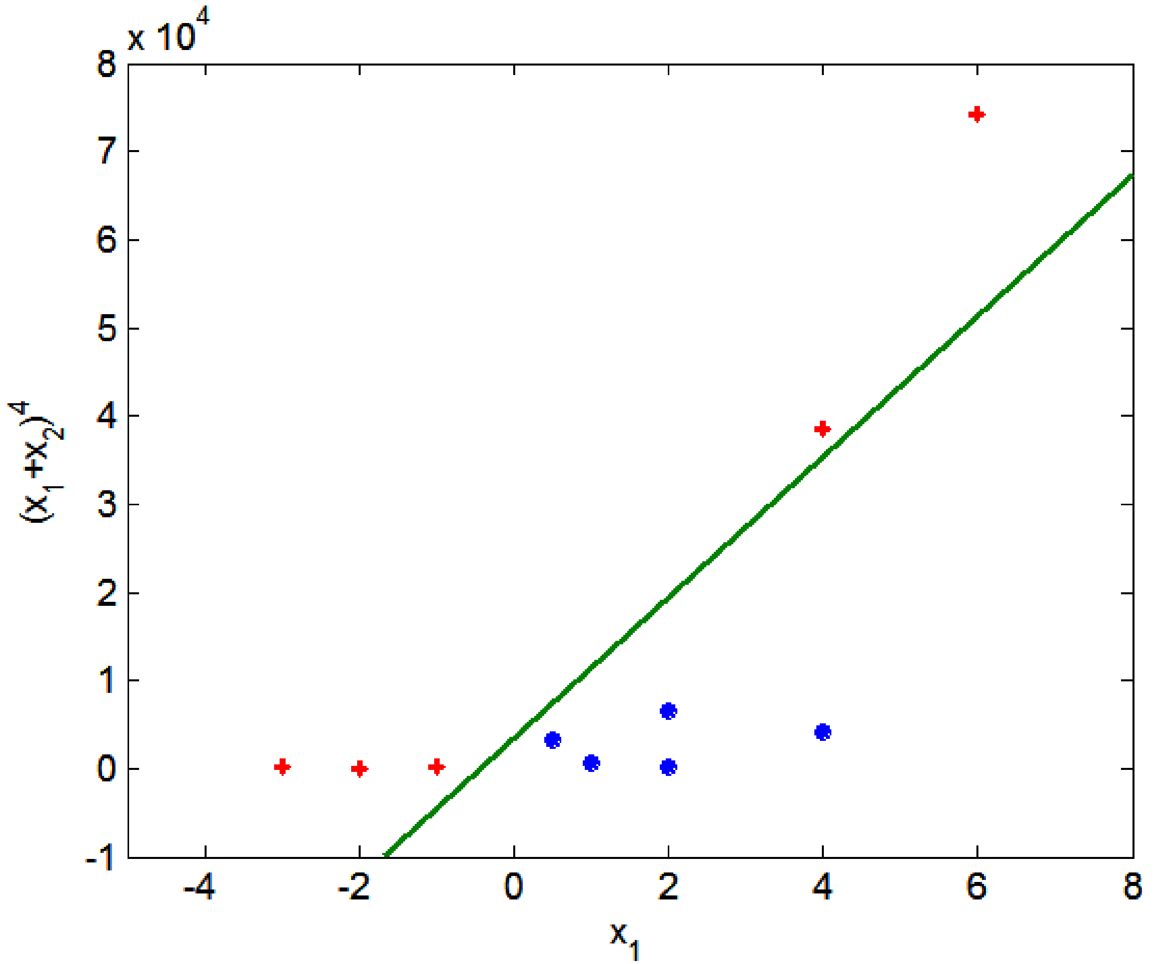
# Nonlinear Support Vector Machines

- What if decision boundary is not linear?



# Nonlinear Support Vector Machines

- Transform data into higher dimensional space



# Contents

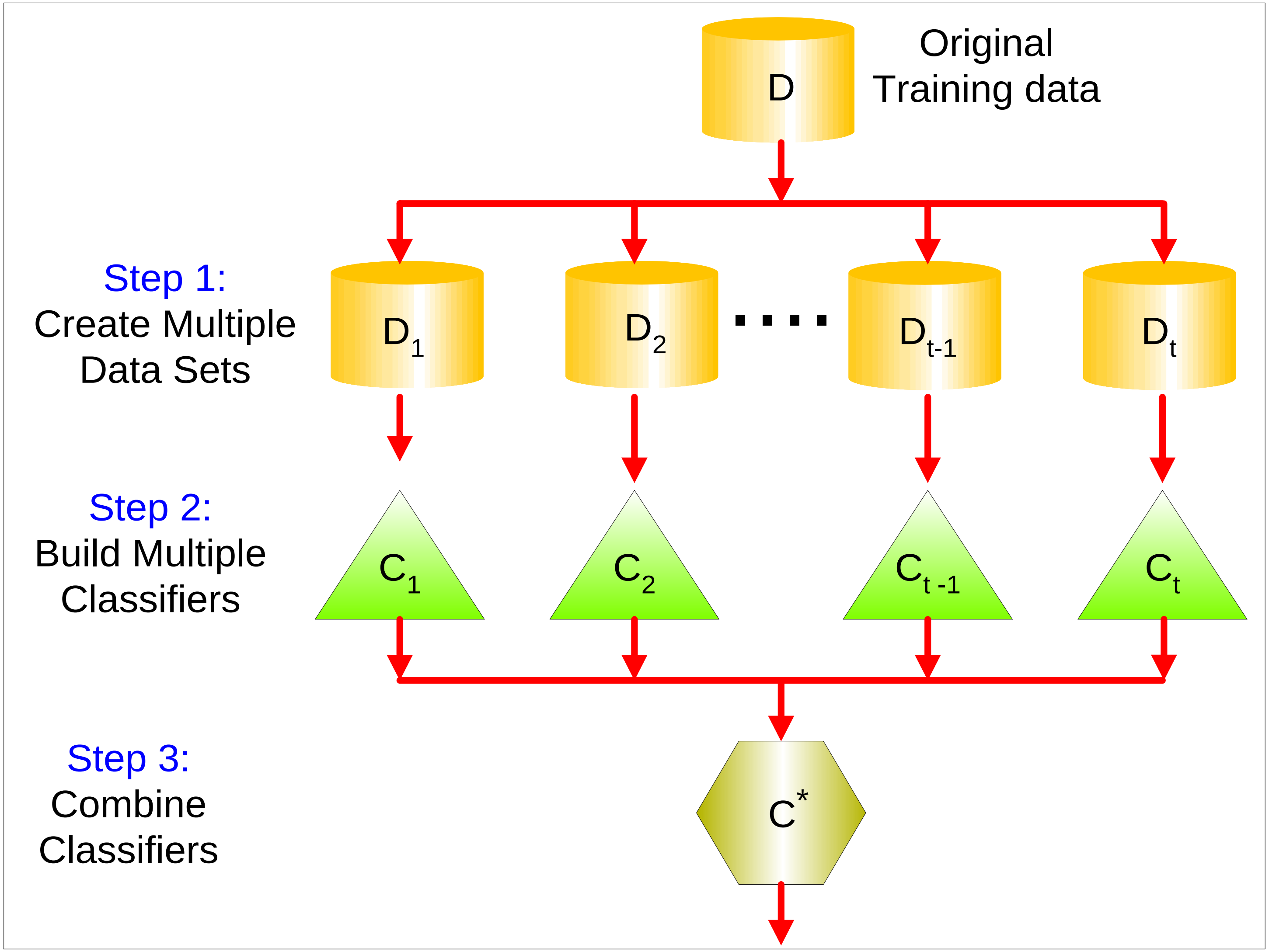
## 기타

# Ensemble Methods

- Construct a set of classifiers from the training data

- Predict class label of previously unseen records by aggregating predictions made by multiple classifiers

# General Idea



# Why does it work?

- Suppose there are 25 base classifiers
  - Each classifier has error rate, ε = 0.35
  - Assume classifiers are independent
  - Probability that the ensemble classifier makes a wrong prediction:

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06$$

**Ensemble classifier인 경우, 다수의 classifier를 통합해서 판단을 내리므로, 이 ensemble classifier가 잘못된 판단을 내리는 경우는, 25개의 기본 분류기 중에서, 반 이상의 기본 분류기가 잘못 예측할 경우이며, 이때의 오류율은 이 식과 같음**

# Examples of Ensemble Methods

- How to generate an ensemble of classifiers?
  - Bagging
  - Boosting

# Bagging

- Sampling with replacement

| Original Data | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Bagging (Round 1) | 7 | 8 | 10 | 8 | 2 | 5 | 10 | 10 | 5 | 9 |
| Bagging (Round 2) | 1 | 4 | 9 | 1 | 2 | 3 | 2 | 7 | 3 | 2 |
| Bagging (Round 3) | 1 | 8 | 5 | 10 | 5 | 5 | 9 | 6 | 3 | 7 |

- Build classifier on each bootstrap sample
- The probability of NOT being selected in any n trials is $(1 – 1/n)^n$
  - → The probability of being selected at least once in n trials is $1 - (1 – 1/n)^n$
  - – The probability of being selected in some particular trial is 1/n.
  - – The probability of **not** being selected in some particular trial is 1−1/n.

# Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
  – Initially, all N records are assigned equal weights
  – Unlike bagging, weights may change at the end of boosting round

# Boosting

- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased

| Original Data | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Boosting (Round 1) | 7 | 3 | 2 | 8 | 7 | 9 | 4 | 10 | 6 | 3 |
| Boosting (Round 2) | 5 | 4 | 9 | 4 | 2 | 5 | 1 | 7 | 4 | 2 |
| Boosting (Round 3) | 4 | 4 | 8 | 10 | 4 | 5 | 4 | 6 | 3 | 4 |

- Instance 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

# Algorithm AdaBoost.M1

1. initialize example weights $w_i = 1/N \quad (i = 1..N)$
2. for $m = 1$ to $t$ // $t$ ... number of iterations
   a) learn a classifier $C_m$ using the current example weights
   b) compute a weighted error estimate $err_m = \frac{\sum w_i \, of \, all \, incorrectly \, classified \, e_i}{\sum_{i=1}^{N} w_i}$

      ← = 1 because weights are normalized

   c) compute a classifier weight $\alpha_m = \frac{1}{2} \ln\left(\frac{1 - err_m}{err_m}\right)$
   d) for all correctly classified examples $e_i$: $w_i \leftarrow w_i e^{-\alpha_m}$
   e) for all incorrectly classified examples $e_i$: $w_i \leftarrow w_i e^{\alpha_m}$

      ← update weights so that sum of correctly classified examples equals sum of incorrectly classified examples

   f) normalize the weights $w_i$ so that they sum to $1$
3. for each test example
   a) try all classifiers $C_m$
   b) predict the class that receives the highest sum of weights $\alpha_m$

# Example: AdaBoost

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where $Z_j$ is the normalization factor

- Base classifiers: $C_1, C_2, \ldots, C_T$

**오류율이 큰 classifier (e=0.93, ai=-3)에서** ai (classifier importance) log 값은 음의값
- 이때, 해당 instance가 맞는 경우, 다음을 위한 가중치 크게올리고(예: exp-(-3.0) → exp(3.0)),
- 해당 instance로 틀리면 wi 가중치는 작은 값을 곱해서 weight를 줄임 (exp(-3.0))
- **맞다고 하는데 왜 다음을 위해 가중치를 높이나? 이는 오류율이 0.5를 넘어서면(random한 것보다 못함), 믿을 수 없기 때문에 해당 instance가 맞다고 하더라도, 이와 반대로 행동(즉, 해당 instance에서 맞음에도 가중치 높여서 다음에 다시 테스트함)**

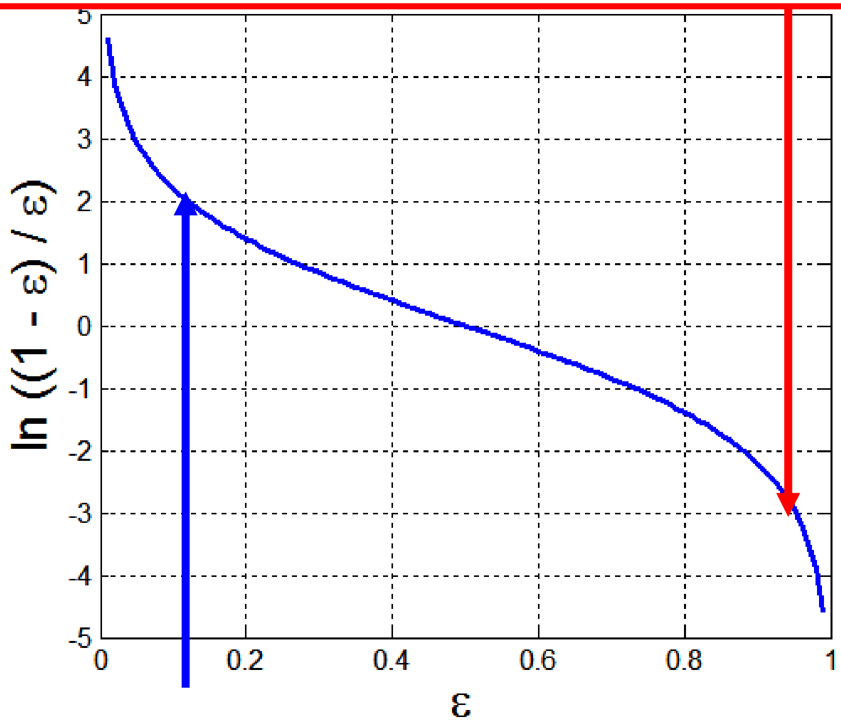
exp(3.0)=20 , exp(-3.0)=0.05

- Error rate:

$$\varepsilon_i = \frac{1}{N}\sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

- Importance of a classifier:

$$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$




**만약 오류율이 작은 classifier (e=0.12)에서** log 값은 약 2.0를 가짐.
이때, 해당 instance가 맞는 경우, 가중치는 작은 값이 곱해져서 (* exp(-2.0)) 줄고,
instance로 틀리면 wi 가중치는 큰 값이 곱해져서 (* exp(2.0)) 증가함. 참고) exp(-2.0)~ 0.14, exp(2.0)~ 7.4

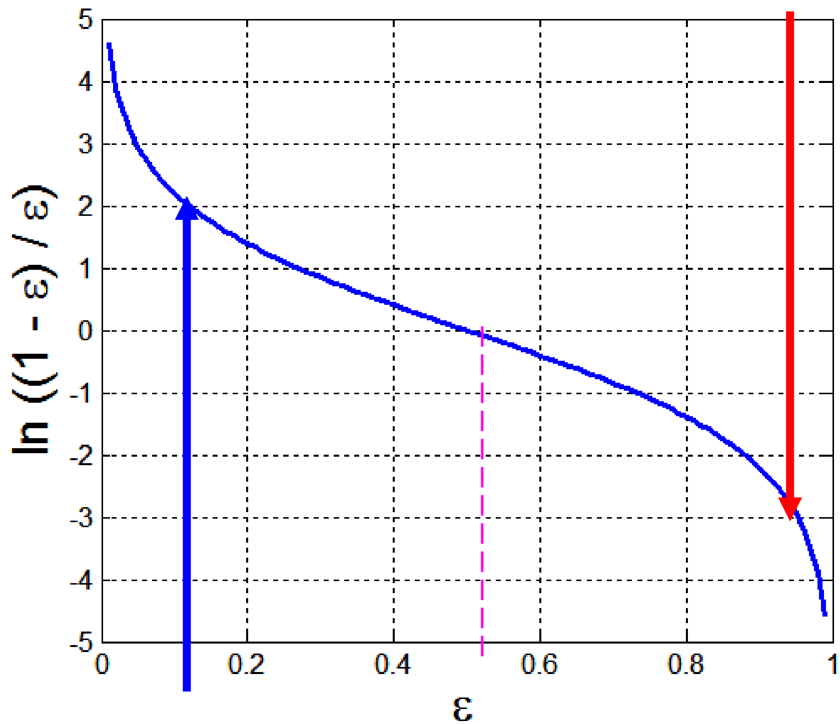
# Example: AdaBoost

$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

where $Z_j$ is the normalization factor

$$\varepsilon_i = \frac{1}{N} \sum_{j=1}^{N} w_j \delta\left(C_i(x_j) \neq y_j\right)$$

$$\alpha_i = \frac{1}{2} \ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$




- **There are three bits of intuition to take from this graph:**
- The classifier weight grows exponentially as the error approaches 0. **Better classifiers are given exponentially more weight.**
- **The classifier weight is zero if the error rate is 0.5**. A classifier with 50% accuracy is no better than random guessing, so we ignore it.
- **The classifier weight grows exponentially negative as the error approaches 1**. We give a negative weight to classifiers with worse than 50% accuracy.
- "Whatever that classifier says, do the opposite!".

- 분류기 가중치 그래프 값은 분류기 오류(e)이 0에 가까워질 수록 급격히 커짐 → 즉, 분류기 품질이 지수적으로 높아짐
- **오류율이 0.5이면, 가중치 그래프는 0이 됨**
- **오류율이 1에 가까워지면, log 값(ai)은 음수가 됨. → 이 경우, 분류가 애기한 경우의 반대로 행동함(즉, Cj(xi)=yi 이라도, 즉 분류기가 맞더라도 가중치는 적음 )**

# Example: AdaBoost

- Weight update:

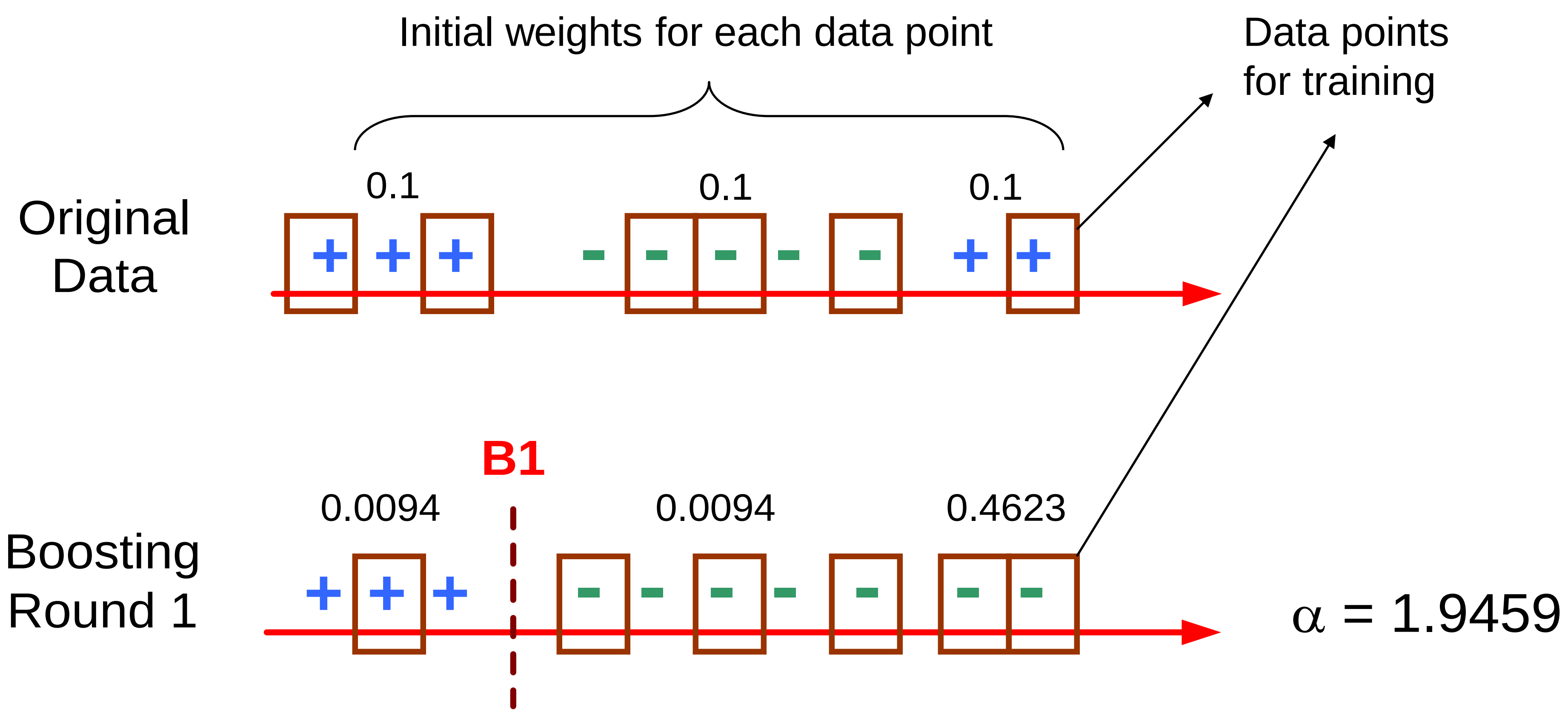
$$w_i^{(j+1)} = \frac{w_i^{(j)}}{Z_j} \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

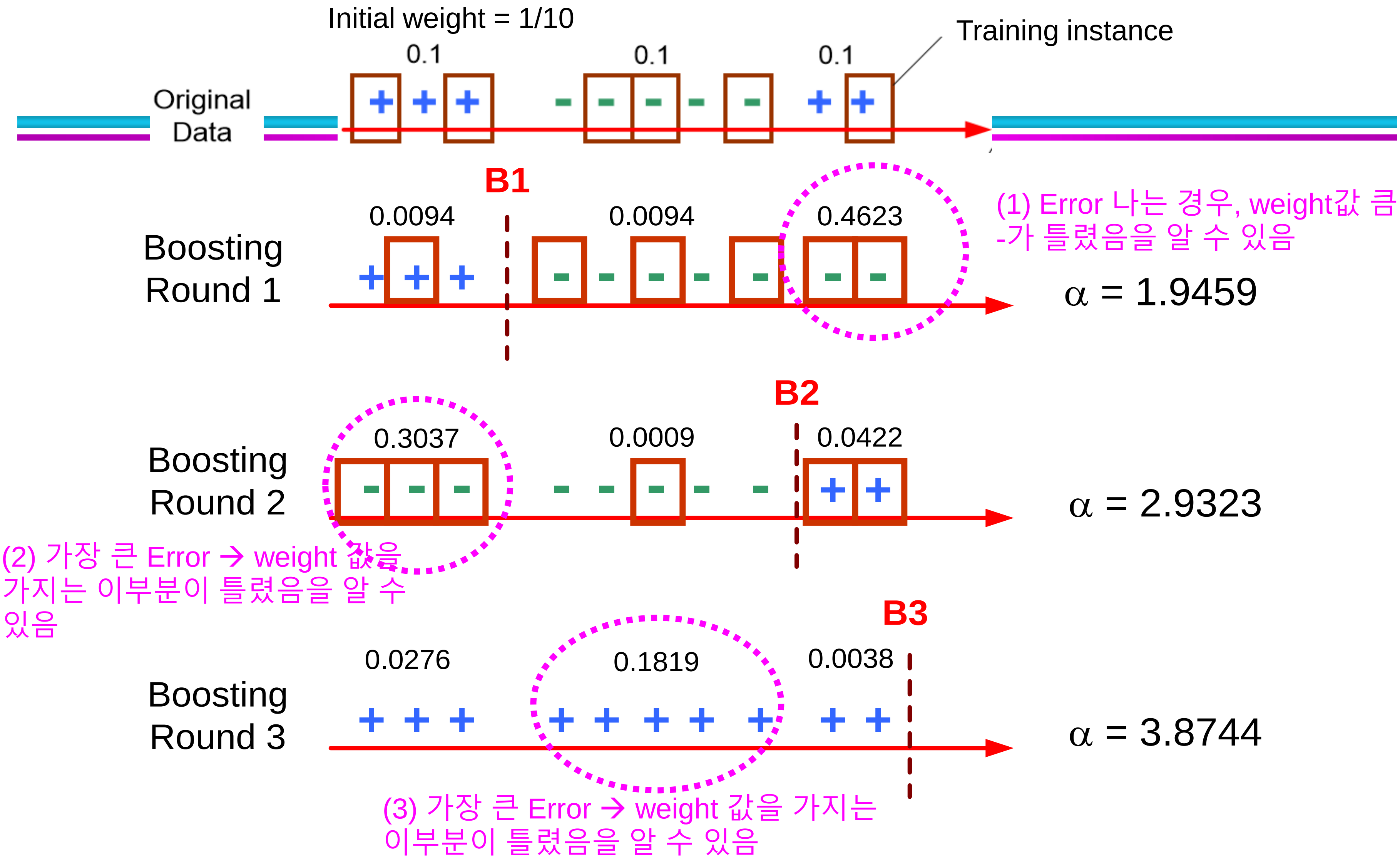
where $Z_j$ is the normalization factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to 1/n and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg\max_y \sum_{j=1}^{T} \alpha_j \delta\left(C_j(x) = y\right)$$

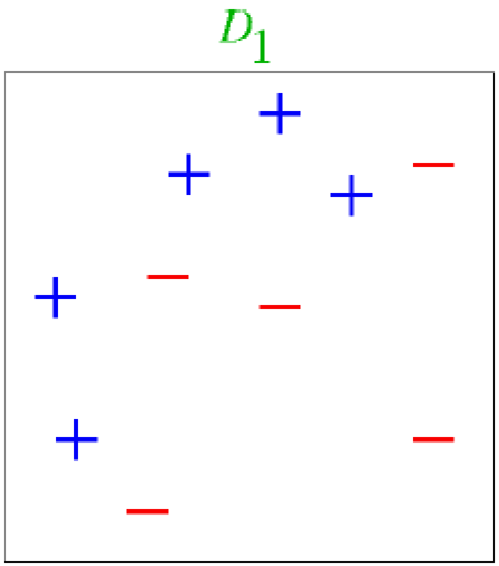
# Illustrating AdaBoost






Round 1,2,3의 틀린 부분 (1),(2),(3)과 맞는 부분을 고려하여 최종 classifier가 만들어짐

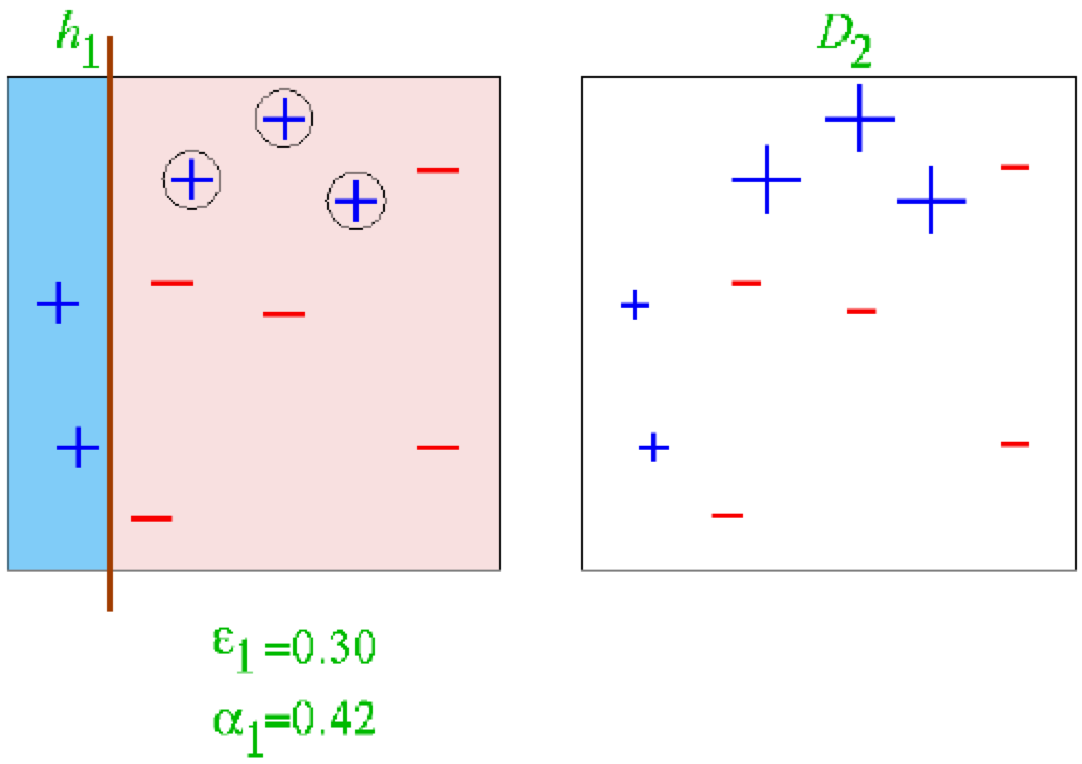
# AdaBoost Example 2




Reference) Data Mining und Maschinelles Lernen, Ensemble Methods from Darmstadt University

# AdaBoost Example 2

- Round 1

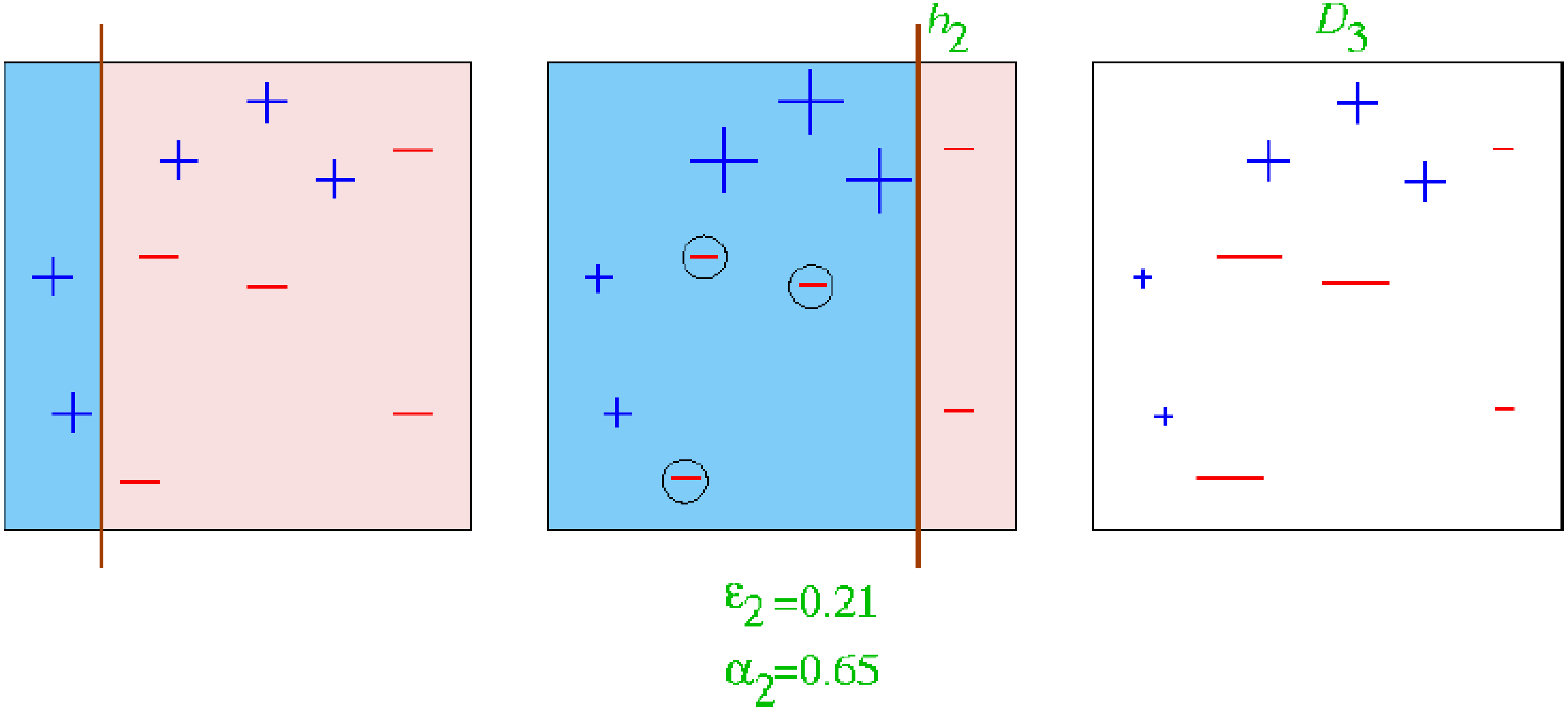



첫번째 분류 수행→ error 값이 0.3이며, h1처럼 분류됨

정보보호 및 지능형 IoT 연구실
Information Security & Intelligent IoT

# AdaBoost Example 2

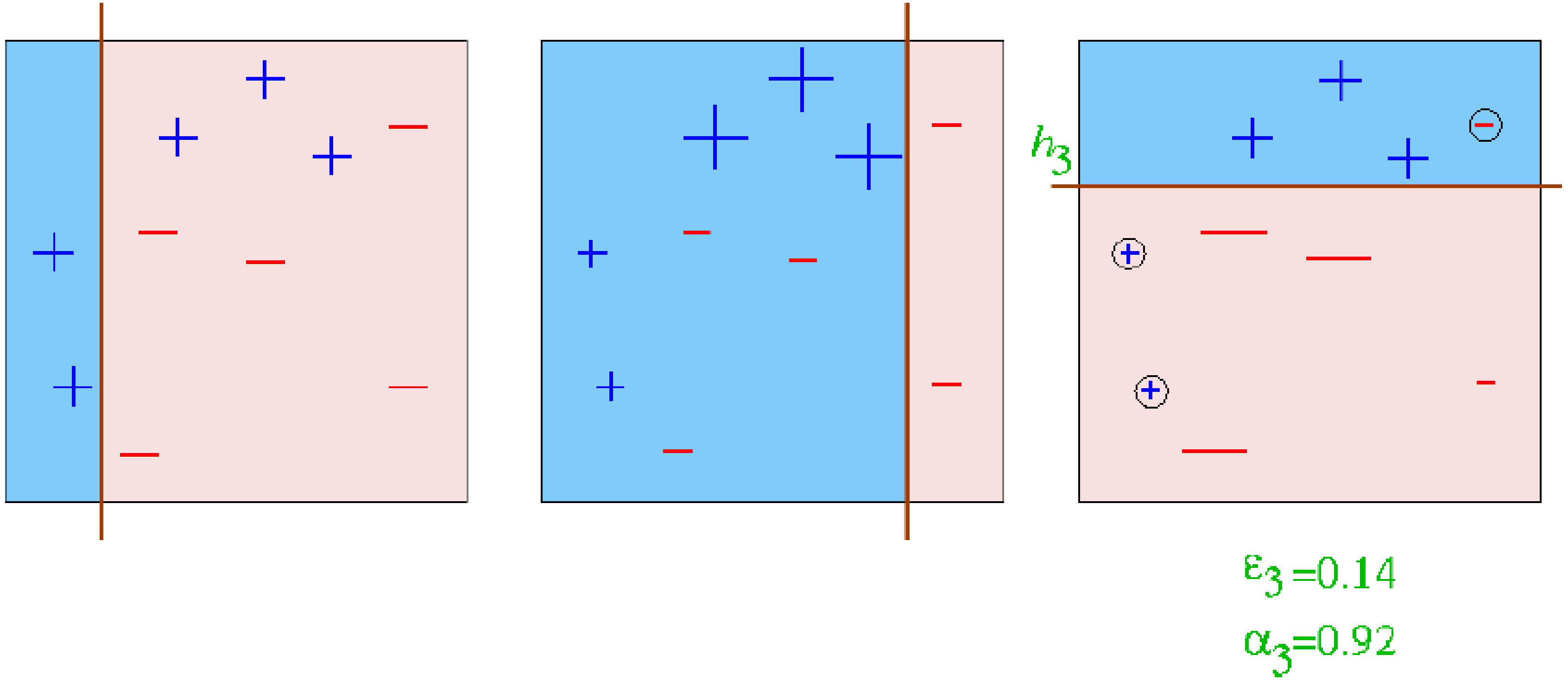
- Round 2




두번째 분류 수행→ error 값이 0.21이며, h2처럼 분류됨
이때 (-)에 의해 error가 생김
다음번 분류때에는 (-)를 강조하여 분류됨

# AdaBoost Example 2

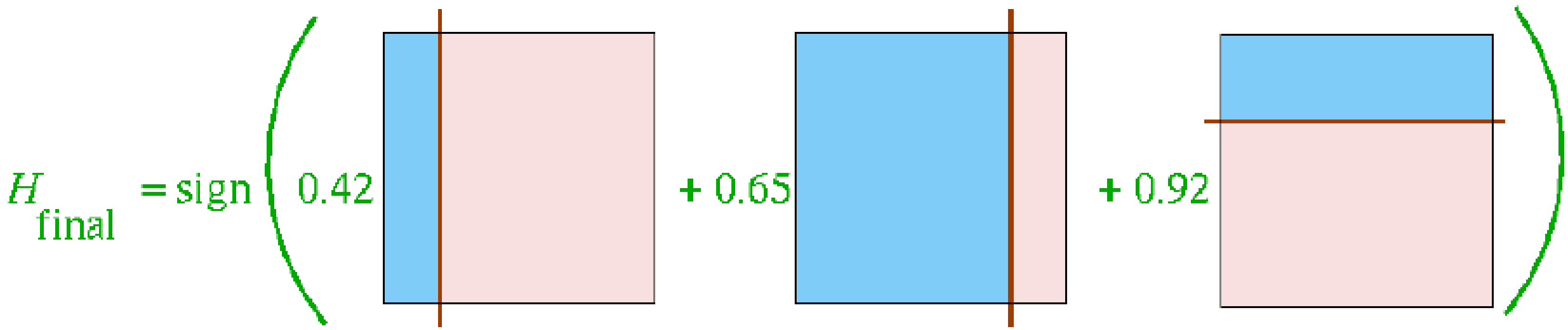
- Round 3

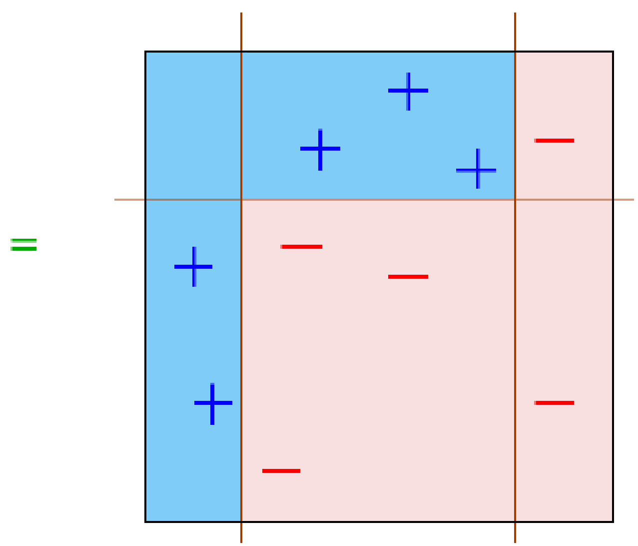



세번째 분류 수행→ error 값이 0.14이며, h3처럼 분류됨

# AdaBoost Example 2

- Final Hypothesis




최종적으로 세가지 분류와 각각의 alpha 값을 고려하여 분류기가 결정됨