

AI6102: Machine Learning Methodologies & Applications

L4: Linear Models: Classification


ZHANG Hanwang

hanwangzhang@ntu.edu.sg

Nanyang Technological University, Singapore

Homepage: <https://mreallab.github.io/>

Outline

- Linear regression revisit
 - Linear models for classification
 - Logistic Regression
 - Support Vector Machines (basic idea, details will be taught in next lecture)
- 
- A decorative graphic consisting of several overlapping, wavy, curved lines in shades of pink and light red, located in the bottom right corner of the slide.

Linear Regression Revisit

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \lambda \Omega(\boldsymbol{\theta})$$

- $f(\mathbf{x}_i; \boldsymbol{\theta})$ is defined as

$$f(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{w} \cdot \mathbf{x} + b$$

OR $f(\mathbf{x}; \boldsymbol{\theta}) = \mathbf{w} \cdot \mathbf{x}$ by introducing additional w_0 and x_0 to absorb b in \mathbf{w}

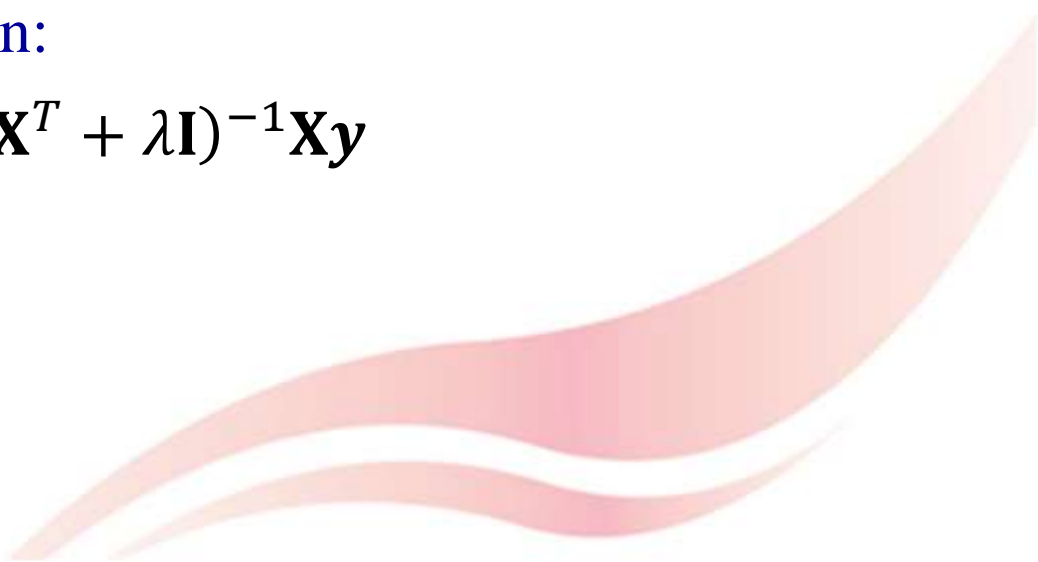
- The loss function $\ell(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i)$ is defined as the squared difference between $f(\mathbf{x}_i; \boldsymbol{\theta})$ and y_i
- The regularization term is defined as the L2 norm

Linear Regression Revisit (cont.)

- With a set of N labeled data $\{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$, the optimization problem is specified as

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \frac{1}{2} \sum_{i=1}^N (\mathbf{w} \cdot \mathbf{x}_i - y_i)^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2$$

- We have derived that there is a closed-form solution for regularized linear regression:

$$\mathbf{w} = (\mathbf{X}\mathbf{X}^T + \lambda\mathbf{I})^{-1}\mathbf{X}\mathbf{y}$$


Linear Models for Classification

$$\hat{\boldsymbol{\theta}} = \arg \min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f(\mathbf{x}_i; \boldsymbol{\theta}), y_i) + \lambda \Omega(\boldsymbol{\theta})$$

- In general, for classification, $f(\mathbf{x}_i; \mathbf{w})$ is defined as

$$f(\mathbf{x}; \boldsymbol{\theta}) = h(\mathbf{w} \cdot \mathbf{x} + b)$$

where $h(z)$ is a function to map continuous values to discrete values (denoting different categories)

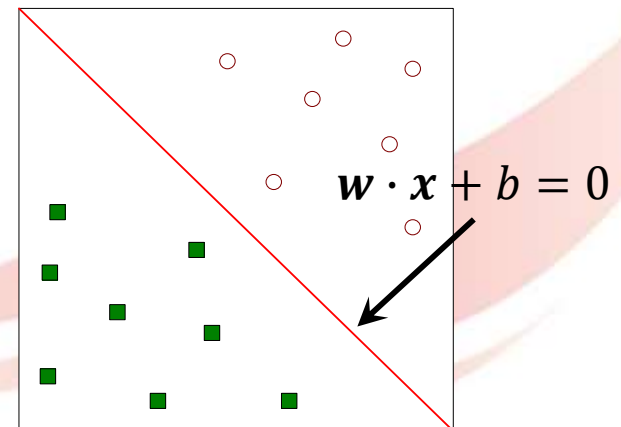
- For binary classification,

$$h(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

Hyperplane

- In linear algebra, from the geometric point of view,
$$\mathbf{w} \cdot \mathbf{x} + b = 0, \text{ where } \mathbf{w}, \mathbf{x} \in \mathbb{R}^m$$
defines a hyperplane in the m -dimensional space, and separate the m -dimensional space into two regions
- For a data instance \mathbf{x}_i
 - If $\mathbf{w} \cdot \mathbf{x}_i + b = 0$, then \mathbf{x}_i is on the hyperplane
 - If $\mathbf{w} \cdot \mathbf{x}_i + b > 0$, then \mathbf{x}_i is on one side of the hyperplane
 - If $\mathbf{w} \cdot \mathbf{x}_i + b < 0$, then \mathbf{x}_i is on the other side of the hyperplane

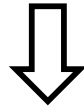
In the 3-dimensional space, hyperplanes are 2-dimensional planes, while in the 2-dimensional space, hyperplanes are lines



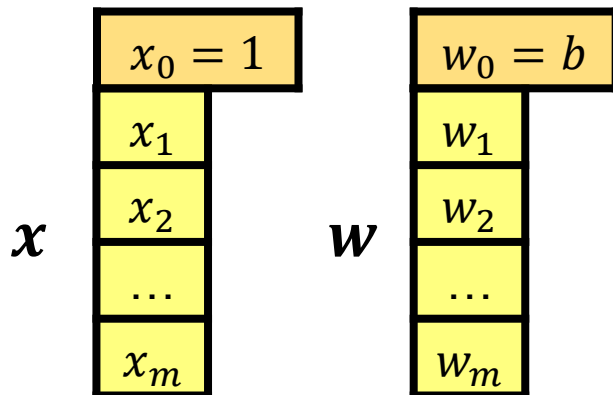
A Compact Form of Hyperplane

- Recall from Lecture 3

$$\mathbf{w} \cdot \mathbf{x} + b = 0, \text{ where } \mathbf{w}, \mathbf{x} \in \mathbb{R}^m$$



$$\sum_{i=1}^m w_i x_i + b = 0$$



$$\mathbf{w} \cdot \mathbf{x} = 0, \text{ where } \mathbf{w}, \mathbf{x} \in \mathbb{R}^{m+1}$$



$$\sum_{i=0}^m w_i x_i = 0 \Rightarrow \sum_{i=1}^m w_i x_i + \boxed{w_0 x_0} = 0$$

The term $w_0 x_0$ in the second equation is enclosed in a red box, and the b in the original equation is highlighted in red.

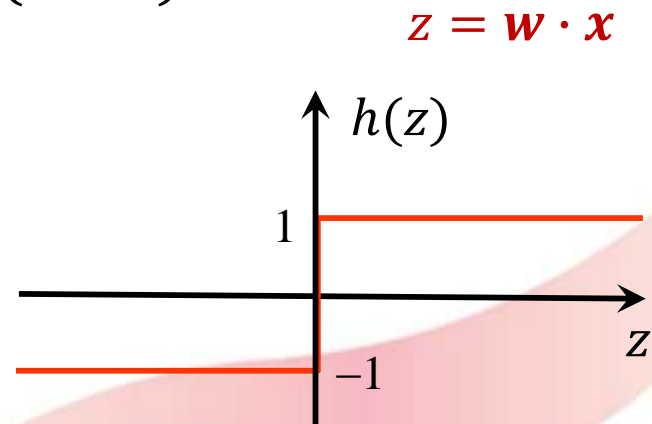
Hyperplane as A Classifier

- Consider binary classification
- We can learn a hyperplane $\mathbf{w} \cdot \mathbf{x} = 0$ in terms of \mathbf{w} to separate data instances of two different classes
- With the learned hyperplane, we can design a function $h(z)$ to generate classification result

$$\hat{y} = f(\mathbf{x}; \boldsymbol{\theta}) = h(\mathbf{w} \cdot \mathbf{x})$$

$$h(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

This is also known as a
sign or threshold function



Hyperplane as A Classifier (cont.)

- The sign function is NOT differentiable everywhere
- Need to look for a more smooth function

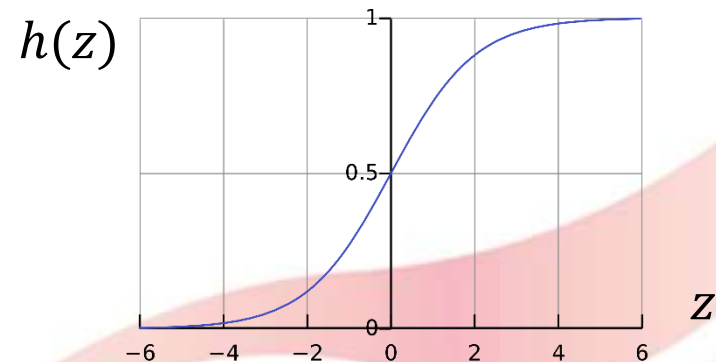
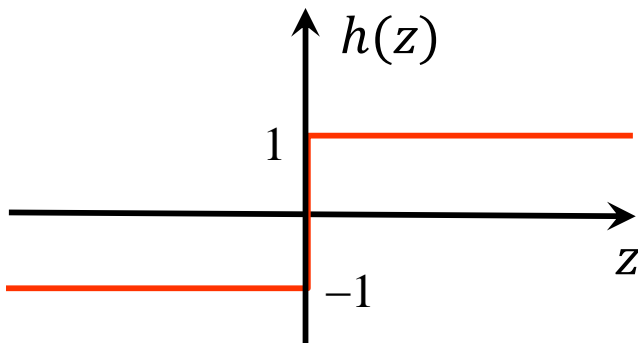
$$f(\mathbf{x}; \boldsymbol{\theta}) = h(\mathbf{w} \cdot \mathbf{x})$$

sigmoid function

$$h(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

$$h(z) = \frac{1}{1 + e^{-z}} = \frac{1}{1 + \exp(-z)}$$

$$z = \mathbf{w} \cdot \mathbf{x}$$



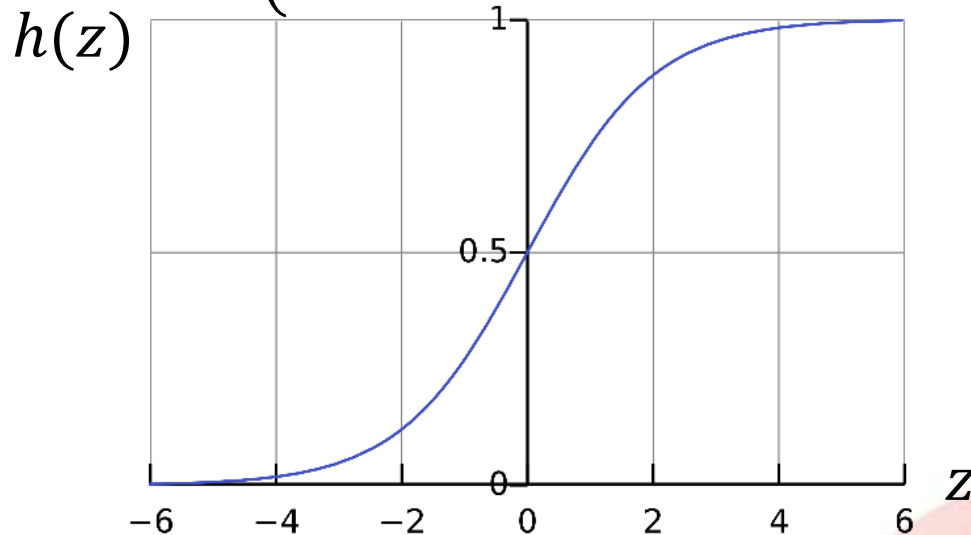
Hyperplane as A Classifier (cont.)

$$h(z) = \frac{1}{1 + \exp(-z)} \quad z = \mathbf{w} \cdot \mathbf{x}$$

Sigmoid function is non-convex, second derivative is non positive

In high dimension hessian is positive semi definite for convex function

$$\begin{cases} +1 & \text{if } h(z) \geq 0.5 \ (z \geq 0) \\ -1 & \text{if } h(z) < 0.5 \ (z < 0) \end{cases} \quad \{-1, +1\} \rightarrow \{0, 1\}$$



$$\begin{cases} 1 & \text{if } h(z) \geq 0.5 \ (z \geq 0) \\ 0 & \text{if } h(z) < 0.5 \ (z < 0) \end{cases}$$

More convenient to
design a loss function

Loss Function

- With the sigmoid function, the predicted values are in $[0,1]$, the ground-truth values are in $\{0,1\}$, can we use the square loss as in linear regression?
- Given N training data instances $\{\mathbf{x}_i, y_i\}, i = 1, \dots, N$, where \mathbf{x}_i is $(m + 1)$ -dimensional ($x_{0i} = 1$), and y_i is either 0 or 1

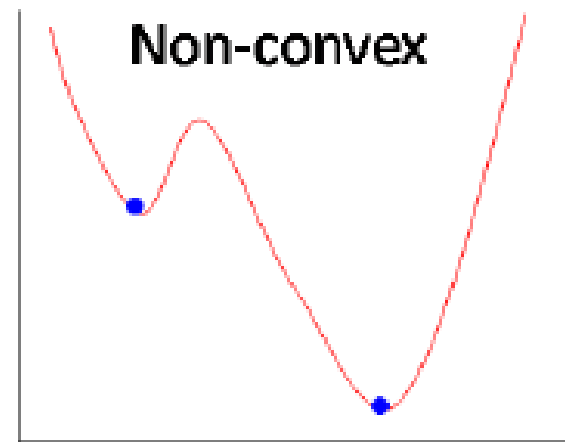
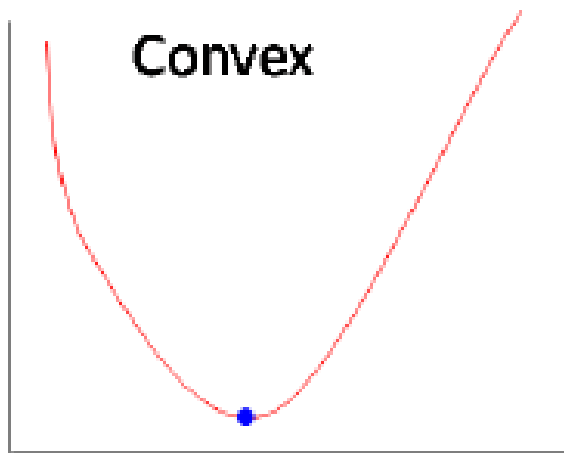
$$\sum_{i=1}^N \ell(h(\mathbf{x}_i; \mathbf{w}), y_i) = \sum_{i=1}^N (h(\mathbf{x}_i; \mathbf{w}) - y_i)^2$$

- However, $h(\mathbf{x}_i; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x}_i)}$ is nonlinear w.r.t. \mathbf{w} . It can be shown that the square loss is non-convex

Loss Function (cont.)

$$\arg \min_{\mathbf{w}} \sum_{i=1}^N \left(\frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x}_i)} - y_i \right)^2$$

- What is the problem if the objective is not convex?

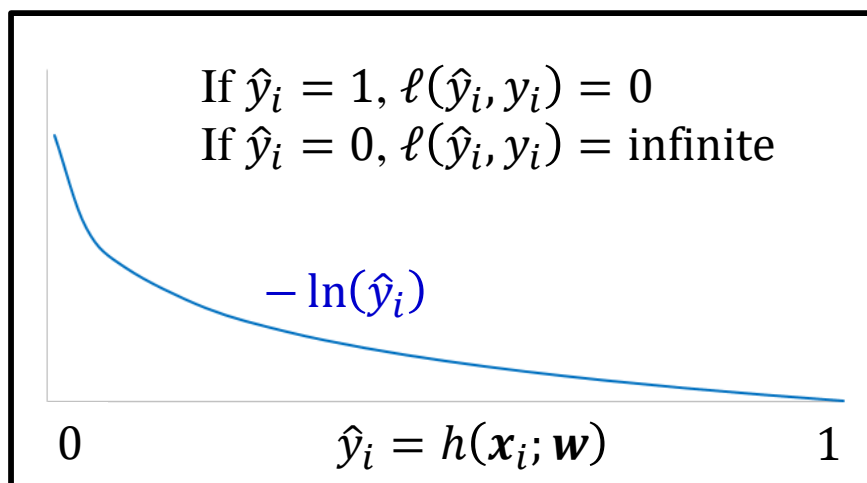


Logistic Regression Loss

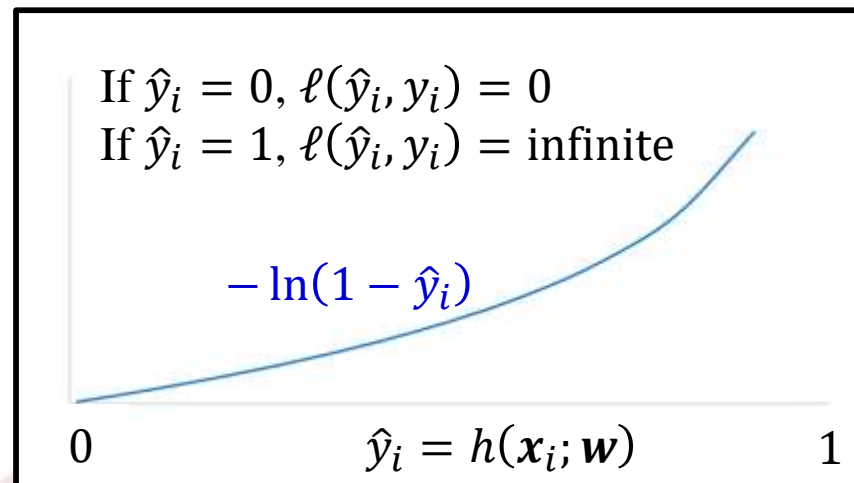
- Define the loss function as

$$\ell(h(\mathbf{x}_i; \mathbf{w}), y_i) = \begin{cases} -\ln(h(\mathbf{x}_i; \mathbf{w})) & \text{if } y_i = 1 \\ -\ln(1 - h(\mathbf{x}_i; \mathbf{w})) & \text{if } y_i = 0 \end{cases}$$

$y_i = 1$



$y_i = 0$

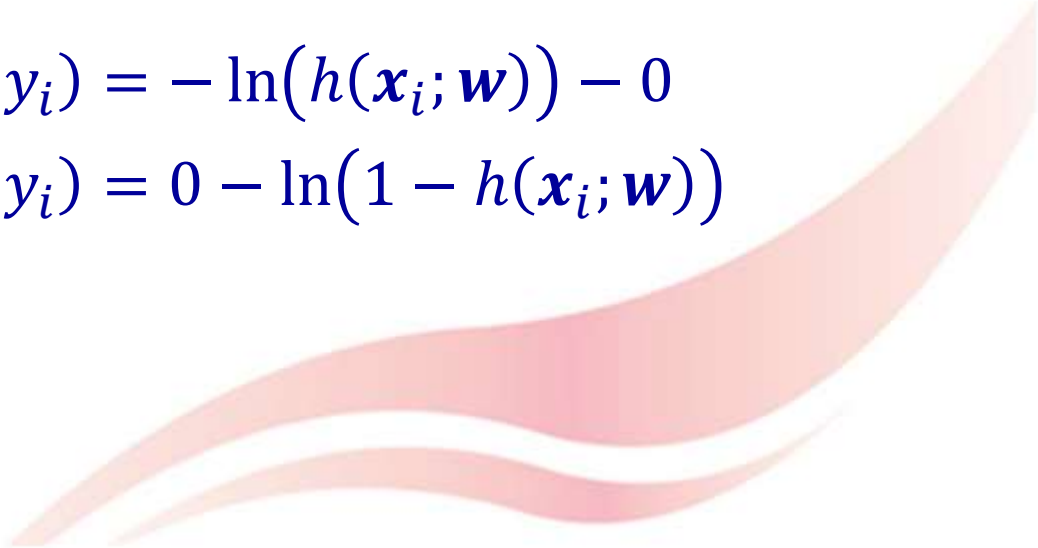


Logistic Regression Loss (cont.)

$$\ell(h(\mathbf{x}_i; \mathbf{w}), y_i) = \begin{cases} -\ln(h(\mathbf{x}_i; \mathbf{w})) & \text{if } y_i = 1 \\ -\ln(1 - h(\mathbf{x}_i; \mathbf{w})) & \text{if } y_i = 0 \end{cases}$$

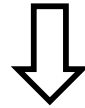
- The above loss can be simplified as

$$\ell(h(\mathbf{x}_i; \mathbf{w}), y_i) = -y_i \ln(h(\mathbf{x}_i; \mathbf{w})) - (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))$$

- If $y_i = 1$, then $\ell(h(\mathbf{x}_i; \mathbf{w}), y_i) = -\ln(h(\mathbf{x}_i; \mathbf{w})) - 0$
 - If $y_i = 0$, then $\ell(h(\mathbf{x}_i; \mathbf{w}), y_i) = 0 - \ln(1 - h(\mathbf{x}_i; \mathbf{w}))$
- 

Logistic Regression Objective

$$\arg \min_{\mathbf{w}} \sum_{i=1}^N [-y_i \ln(h(\mathbf{x}_i; \mathbf{w})) - (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]$$

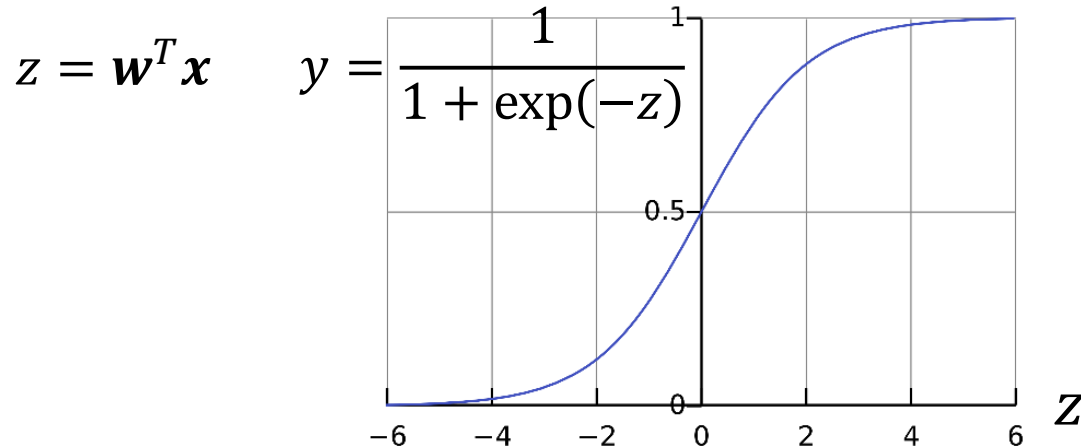


$$\arg \min_{\mathbf{w}} - \sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]$$

$$\text{where } h(\mathbf{x}_i; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x}_i)}$$

The objective is convex w.r.t. \mathbf{w} , and differentiable

Probabilistic Point of View



- Assume the conditional probability of class 1 is modeled as

$$P(y = 1|\mathbf{x}) = h(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

- Thus, the conditional probability of class 0 is

$$P(y = 0|\mathbf{x}) = 1 - P(y = 1|\mathbf{x}) = 1 - h(\mathbf{x}; \mathbf{w}) = \frac{\exp(-\mathbf{w}^T \mathbf{x})}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

Probability Review

- Let A be a random variable (a feature / a label in machine learning)

- Marginal probability $0 \leq P(A = a) \leq 1$

$$P(A = a)$$

refers to the probability that variable $A = a$

$$\sum_{a_i} P(A = a_i) = 1$$

- For example, in binary classification problem, a class label y has two possible values, 0 or 1 (-1 or $+1$)

$$P(y = 0) + P(y = 1) = 1$$


Probability Review (cont.)

- Let A and B be a pair of random variables (features/labels in machine learning).
- Their joint probability

$$P(A = a, B = b)$$

refers to the probability that variable $A = a$, and at the same time variable $B = b$

- E.g., for pair of input data instance and output label (\mathbf{x}_i, y_i) , its joint probability is

$$P(\mathbf{x} = \mathbf{x}_i, y = y_i)$$

OR

$$P(\mathbf{x}_i, y_i) \text{ for simplicity}$$

Probability Review (cont.)

- Conditional probability

$$P(B = b|A = a)$$

refers to the probability that variable B will take on the value b , given that the variable A is observed to have the value a

$$\sum_{b_i} P(B = b_i|A = a) = 1$$

- For example, in binary classification, given a data instance \mathbf{x}_i

$$P(y_i = 1|\mathbf{x}_i) + P(y_i = 0|\mathbf{x}_i) = 1$$


Parametric Form

$$P(y = 1|\mathbf{x}) = h(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

$$P(y = 0|\mathbf{x}) = 1 - h(\mathbf{x}; \mathbf{w}) = \frac{\exp(-\mathbf{w}^T \mathbf{x})}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

- We can define more compact form as

$$P(y|\mathbf{x}; \mathbf{w}) = h(\mathbf{x}; \mathbf{w})^y (1 - h(\mathbf{x}; \mathbf{w}))^{1-y}$$

- If $y = 1$, then $P(y|\mathbf{x}; \mathbf{w}) = h(\mathbf{x}; \mathbf{w})^1 (1 - h(\mathbf{x}; \mathbf{w}))^0$
- If $y = 0$, then $P(y|\mathbf{x}; \mathbf{w}) = h(\mathbf{x}; \mathbf{w})^0 (1 - h(\mathbf{x}; \mathbf{w}))^1$
- To find \mathbf{w} that makes sampling y_i conditioned on \mathbf{x}_i from $P(y|\mathbf{x}; \mathbf{w})$ as likely as possible
 - Maximum likelihood estimation

Maximum Likelihood Estimation

Notice the different kind of l, likelihood l is notationally different from loss L.

- For each training pair $\{\mathbf{x}_i, y_i\}$, the likelihood of parameter \mathbf{w} of the conditional probability $P(y|\mathbf{x}; \mathbf{w})$ is

$$l(\mathbf{w}|\{\mathbf{x}_i, y_i\}) \triangleq P(y_i|\mathbf{x}_i; \mathbf{w}) = h(\mathbf{x}_i; \mathbf{w})^{y_i}(1 - h(\mathbf{x}_i; \mathbf{w}))^{1-y_i}$$

- Ideally, for each training pair $\{\mathbf{x}_i, y_i\}$, $l(\mathbf{w}|\{\mathbf{x}_i, y_i\}) = P(y_i|\mathbf{x}_i; \mathbf{w}) = 1$
- Maximum likelihood estimation (MLE) aims to find a solution of \mathbf{w} such that $P(y_i|\mathbf{x}_i; \mathbf{w})$ is maximized

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} l(\mathbf{w}|\{\mathbf{x}_i, y_i\}) = \arg \max_{\mathbf{w}} P(y_i|\mathbf{x}_i; \mathbf{w})$$

- The parametric form of the conditional probability fits the pair of input data instance and corresponding output well

MLE (cont.)

- Given a set of N training input-output pairs $\{\mathbf{x}_i, y_i\}, i = 1, \dots, N$, which are i.i.d., the likelihood is defined as the product of the likelihoods of each individual data pairs

$$\mathcal{L}(\mathbf{w}) = \prod_{i=1}^N l(\mathbf{w}|\{\mathbf{x}_i, y_i\}) = \prod_{i=1}^N P(y_i|\mathbf{x}_i; \mathbf{w})$$

- The goal of MLE is to find a solution of \mathbf{w} such that $\mathcal{L}(\mathbf{w})$ is maximized, ideally for all $\{\mathbf{x}_i, y_i\}, i = 1, \dots, N, P(y_i|\mathbf{x}_i; \mathbf{w}) = 1$, and thus $\mathcal{L}(\mathbf{w}) = 1$

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^N P(y_i|\mathbf{x}_i; \mathbf{w})$$

MLE (cont.)

\ln is a monotonically increasing function, which is important because it does not change the trend, ie the higher is still better.

$$\ln(ab) = \ln a + \ln b$$

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \arg \max_{\mathbf{w}} \prod_{i=1}^N P(y_i | \mathbf{x}_i; \mathbf{w})$$

- In practice, we maximize $\ln \mathcal{L}(\mathbf{w})$ instead. Why?

1. The $\ln(\cdot)$ function converts the product into a sum

$$\ln \mathcal{L}(\mathbf{w}) = \ln \left(\prod_{i=1}^N P(y_i | \mathbf{x}_i; \mathbf{w}) \right) = \sum_{i=1}^N \ln P(y_i | \mathbf{x}_i; \mathbf{w})$$

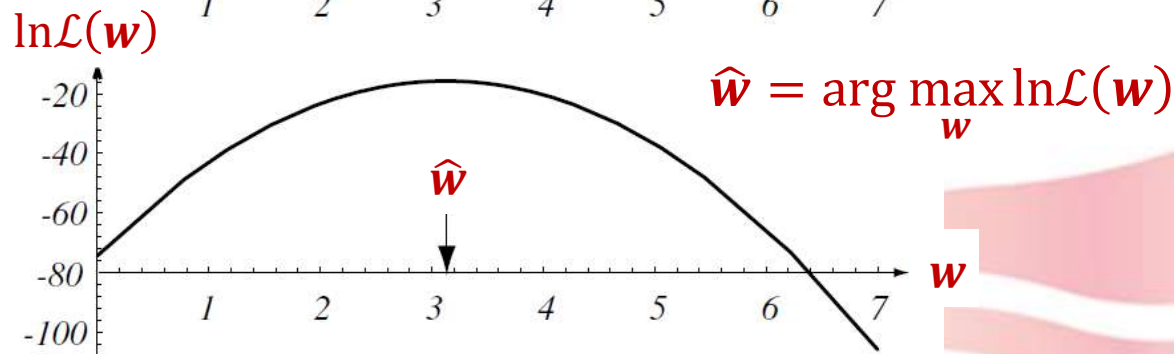
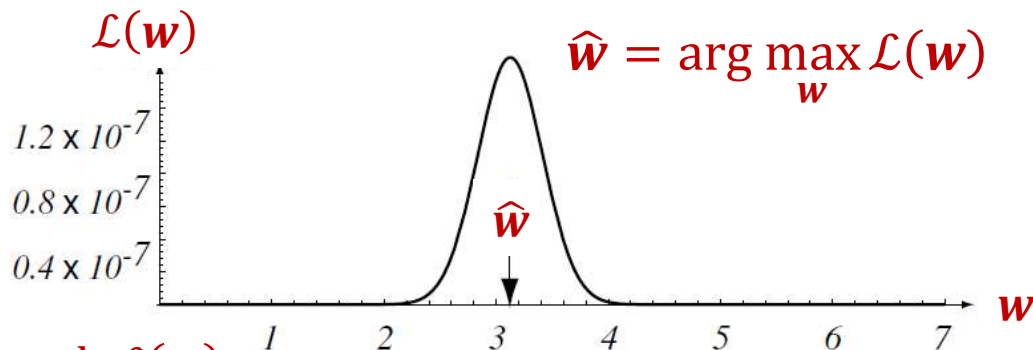
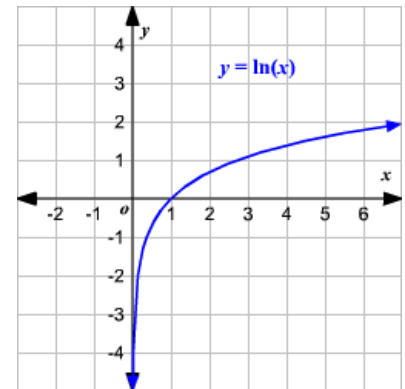
2. The $\ln(\cdot)$ function is a strictly increasing function, the solution of \mathbf{w} remains the same

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \Leftrightarrow \hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \ln \mathcal{L}(\mathbf{w})$$

Maximum Log-Likelihood

- How to understand “the $\ln(\cdot)$ function is a strictly increasing function, the solution of \mathbf{w} remains the same”

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \Leftrightarrow \hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \ln \mathcal{L}(\mathbf{w})$$



Maximum Log-Likelihood (cont.)

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \ln \mathcal{L}(\mathbf{w}) = \arg \max_{\mathbf{w}} \sum_{i=1}^N \ln P(y_i | \mathbf{x}_i; \mathbf{w})$$

Recall that $P(y_i | \mathbf{x}_i; \mathbf{w}) = h(\mathbf{x}_i; \mathbf{w})^{y_i} (1 - h(\mathbf{x}_i; \mathbf{w}))^{1-y_i}$

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \sum_{i=1}^N \ln(h(\mathbf{x}_i; \mathbf{w})^{y_i} (1 - h(\mathbf{x}_i; \mathbf{w}))^{1-y_i})$$

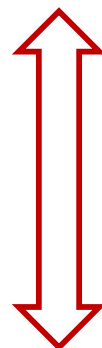
$$\ln(a^c b^d) = \ln a^c + \ln b^d = c \ln a + d \ln b$$

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]$$

Maximum Log-Likelihood (cont.)

$$\hat{\mathbf{w}} = \arg \max_{\mathbf{w}} \sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]$$

Equivalent!



$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} - \sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]$$

The objective induced on Page 15 based on empirical risk minimization

Why Called Logistic Regression?

- In statistics, given a variable z that has two outcomes $\{0, 1\}$, denoted by $p = P(z = 1)$, then $1 - p = P(z = 0)$
- The odds of the probability p is defined as the ratio of p and $1 - p$

$$\text{odds} = \frac{P(z = 1)}{P(z = 0)} = \frac{p}{1 - p}$$

- The logit of the probability p is the logarithm of the odds:

$$\text{logit}(p) = \ln(\text{odds}) = \ln\left(\frac{p}{1 - p}\right)$$

- Recall that

$$P(y = 1|\mathbf{x}) = h(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

$$P(y = 0|\mathbf{x}) = 1 - h(\mathbf{x}; \mathbf{w}) = \frac{\exp(-\mathbf{w}^T \mathbf{x})}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

Logit + Regression

- Denote by

$$p = P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}$$

- The logit of p is

$$\boxed{\ln\left(\frac{p}{1-p}\right)} = \ln\left(\frac{\frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}}{\frac{\exp(-\mathbf{w}^T \mathbf{x})}{1 + \exp(-\mathbf{w}^T \mathbf{x})}}\right)$$

logit

$$= \ln(\exp(-\mathbf{w}^T \mathbf{x}))^{-1}$$

logistic regression

$$\ln(\exp(a)^{-1}) = -1 \ln \exp(a) = -a$$


$$= -1 \times (-\mathbf{w}^T \mathbf{x})$$

$$= \boxed{\mathbf{w}^T \mathbf{x}} \text{ Linear regression}$$

Approaches to Solution

Objective $E(\mathbf{w})$

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} - \sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]$$

- The objective is convex, differentiable, without constraints
-  • Can we set derivatives equal to zero and solve the resultant equations to get a closed form solution?
- Optimization methods
 - Gradient descent – first order methods
 - Newton's method – second order methods

Optimization

- Gradient descent

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \rho \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}}$$

$$\begin{aligned} \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= \frac{\partial \left(-\sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))] \right)}{\partial \mathbf{w}} \\ &= - \sum_{i=1}^N \frac{\partial [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]}{\partial \mathbf{w}} \\ &= - \sum_{i=1}^N \left(\frac{\partial (y_i \ln(h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} + \frac{\partial ((1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} \right) \end{aligned}$$

Optimization (cont.)

$$\frac{\partial(y_i \ln(h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} \quad \frac{\partial \ln f(z)}{\partial z} = \frac{\partial \ln f(z)}{\partial f(z)} \frac{\partial f(z)}{\partial z} = \frac{1}{f(z)} \frac{\partial f(z)}{\partial z}$$

$$= y_i \frac{1}{h(\mathbf{x}_i; \mathbf{w})} \frac{\partial(h(\mathbf{x}_i; \mathbf{w}))}{\partial \mathbf{w}} \quad P(y = 1 | \mathbf{x}_i) = h(\mathbf{x}_i; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)}$$

$$= y_i \frac{1}{h(\mathbf{x}_i; \mathbf{w})} \frac{\partial((1 + \exp(-\mathbf{w}^T \mathbf{x}_i))^{-1})}{\partial \mathbf{w}} \quad \frac{\partial f(z)^k}{\partial z} = \frac{\partial f(z)^k}{\partial f(z)} \frac{\partial f(z)}{\partial z} = k f(z)^{k-1} \frac{\partial f(z)}{\partial z}$$

$$= y_i \frac{1}{h(\mathbf{x}_i; \mathbf{w})} (-1)(1 + \exp(-\mathbf{w}^T \mathbf{x}_i))^{-2} \frac{\partial \exp(-\mathbf{w}^T \mathbf{x}_i)}{\partial \mathbf{w}}$$

Optimization (cont.)

$$y_i \frac{1}{h(\mathbf{x}_i; \mathbf{w})} (-1) (1 + \exp(-\mathbf{w}^T \mathbf{x}_i))^{-2} \frac{\partial \exp(-\mathbf{w}^T \mathbf{x}_i)}{\partial \mathbf{w}}$$

$$\frac{\partial \exp(f(z))}{\partial z} = \frac{\partial \exp(f(z))}{\partial f(z)} \frac{\partial f(z)}{\partial z} = \exp(f(z)) \frac{\partial f(z)}{\partial z}$$

$$= y_i \frac{1}{h(\mathbf{x}_i; \mathbf{w})} (-1) (1 + \exp(-\mathbf{w}^T \mathbf{x}_i))^{-2} \exp(-\mathbf{w}^T \mathbf{x}_i) (-\mathbf{x}_i)$$

$$P(y = 1 | \mathbf{x}_i) = h(\mathbf{x}_i; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)}$$

$$= y_i (1 + \exp(-\mathbf{w}^T \mathbf{x}_i)) \frac{1}{(1 + \exp(-\mathbf{w}^T \mathbf{x}_i))^2} \exp(-\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i$$

$$= y_i \frac{\exp(-\mathbf{w}^T \mathbf{x}_i)}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)} \mathbf{x}_i$$

Optimization (cont.)

$$\frac{\partial \left((1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w})) \right)}{\partial \mathbf{w}}$$

$$= (1 - y_i) \frac{1}{1 - h(\mathbf{x}_i; \mathbf{w})} (-1) \frac{\partial(h(\mathbf{x}_i; \mathbf{w}))}{\partial \mathbf{w}}$$

$$P(y = 1|\mathbf{x}_i) = h(\mathbf{x}_i; \mathbf{w}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)}$$

$$= (y_i - 1) \frac{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)}{\exp(-\mathbf{w}^T \mathbf{x}_i)} \frac{1}{(1 + \exp(-\mathbf{w}^T \mathbf{x}_i))^2} \exp(-\mathbf{w}^T \mathbf{x}_i) \mathbf{x}_i$$

$$= (y_i - 1) \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)} \mathbf{x}_i$$

Optimization (cont.)

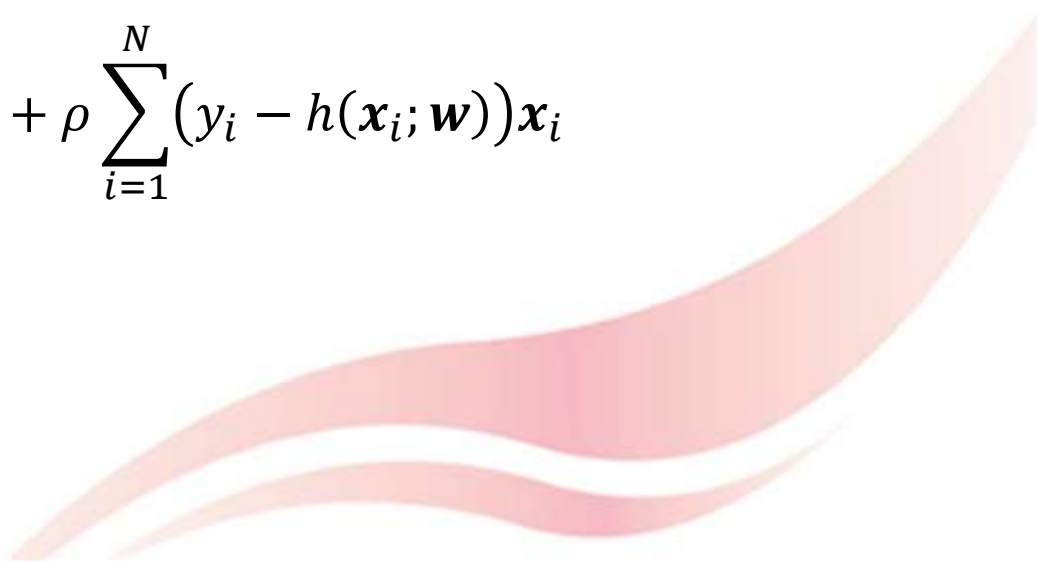
$$\begin{aligned}\frac{\partial \left((1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w})) \right)}{\partial \mathbf{w}} &= (y_i - 1) \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)} \mathbf{x}_i \\ &= (y_i - 1) h(\mathbf{x}_i; \mathbf{w}) \mathbf{x}_i\end{aligned}$$

$$\begin{aligned}\frac{\partial (y_i \ln(h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} &= y_i \frac{\exp(-\mathbf{w}^T \mathbf{x}_i)}{1 + \exp(-\mathbf{w}^T \mathbf{x}_i)} \mathbf{x}_i \\ &= y_i (1 - h(\mathbf{x}_i; \mathbf{w})) \mathbf{x}_i\end{aligned}$$

$$\frac{\partial \left((1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w})) \right)}{\partial \mathbf{w}} + \frac{\partial (y_i \ln(h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} = (y_i - h(\mathbf{x}_i; \mathbf{w})) \mathbf{x}_i$$

Optimization (cont.)

$$\begin{aligned}\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} &= - \sum_{i=1}^N \left(\frac{\partial (y_i \ln(h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} + \frac{\partial ((1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} \right) \\ &= - \sum_{i=1}^N (y_i - h(\mathbf{x}_i; \mathbf{w})) \mathbf{x}_i\end{aligned}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \rho \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t + \rho \sum_{i=1}^N (y_i - h(\mathbf{x}_i; \mathbf{w})) \mathbf{x}_i$$


Recall: Approaches to Solution

Objective $E(\mathbf{w})$

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} - \sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))]$$

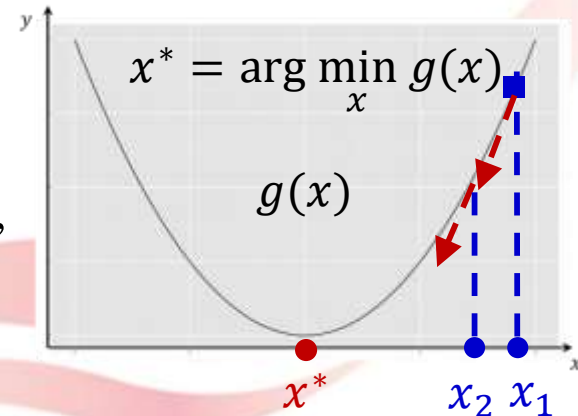
- The objective is convex, differentiable, without constraints
- Can we set derivatives equal to zero and solve the resultant equations to get a closed form solution?
- Optimization methods
 - Gradient descent – first order methods
 - Newton's method – second order methods



Newton's Method in Optimization

- Consider one dimension case, and our goal is to find an optimal solution x^* that minimizes a convex objective $g(x)$
- Gradient descent methods only use the first derivative together with a learning rate (step size) to generate a series $\{x_t\}, t = 0, 1, \dots$, iteratively, which converges to the optimal solution x^*
 - May take many steps to converge to the optimum
- Newton's method aims to exploit the second order derivative to “guess” a better x_t at each iteration t
 - Needs fewer steps to converge to the optimum
 - Assumption: $g(x)$ is at least twice differentiable, and $g''(x) \neq 0$

Out of scope



Regularized Logistic Regression

Objective $E(\mathbf{w})$

$$\min_{\mathbf{w}} \left[- \sum_{i=1}^N [y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w}))] + \frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right]$$

$$\frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = - \sum_{i=1}^N \left(\frac{\partial (y_i \ln(h(\mathbf{x}_i; \mathbf{w})) + (1 - y_i) \ln(1 - h(\mathbf{x}_i; \mathbf{w})))}{\partial \mathbf{w}} \right) + \frac{\partial \left(\frac{\lambda}{2} \|\mathbf{w}\|_2^2 \right)}{\partial \mathbf{w}}$$

$$= - \sum_{i=1}^N (y_i - h(\mathbf{x}_i; \mathbf{w})) \mathbf{x}_i + \lambda \mathbf{w}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \rho \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{w}_t + \rho \left(\sum_{i=1}^N (y_i - h(\mathbf{x}_i; \mathbf{w})) \mathbf{x}_i - \lambda \mathbf{w} \right)$$

Extension to Multiple Classes

- Suppose there are C classes, $\{0, 1, \dots, C - 1\}$
- Each class except class 0 is associated with a specific $\mathbf{w}^{(c)}$, $c = 1, \dots, C - 1$

$$\left. \begin{array}{l} \text{For } c > 0: P(y = c|\mathbf{x}) = \frac{\exp(-\mathbf{w}^{(c)T} \mathbf{x})}{1 + \sum_{c=1}^{C-1} \exp(-\mathbf{w}^{(c)T} \mathbf{x})} \\ \text{For } c = 0: P(y = 0|\mathbf{x}) = \frac{1}{1 + \sum_{c=1}^{C-1} \exp(-\mathbf{w}^{(c)T} \mathbf{x})} \end{array} \right\} \sum_{c=0}^{C-1} P(y = c|\mathbf{x}) = 1$$

- For learning each $\mathbf{w}^{(c)}$, the procedure is basically the same as what we derived!
- For a test data instance \mathbf{x}^* , $y^* = \arg \max_{c \in \{0, \dots, C-1\}} P(y = c|\mathbf{x}^*)$

Assignment

Implementation using scikit-learn

- API: `sklearn.linear_model`: Linear Models

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.linear_model

Linear classifiers

`linear_model.LogisticRegression`([penalty, ...]) Logistic Regression (aka logit, MaxEnt) classifier.

`linear_model.LogisticRegressionCV`(*[, Cs, ...]) Logistic Regression CV (aka logit, MaxEnt) classifier.

`sklearn.linear_model.LogisticRegression`

```
class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True,
intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0,
warm_start=False, n_jobs=None, l1_ratio=None)
```

[\[source\]](#)

C : float, default=1.0

Inverse of regularization strength; must be a positive float. Like in support vector machines, smaller values specify stronger regularization.

Example

```
>>> from sklearn.linear_model import LogisticRegression  
>>> import numpy as np
```

```
>>> n_samples, n_features = 10, 5  
>>> rng = np.random.RandomState(0)  
>>> y = rng.integers(2, n_samples)  
>>> X = rng.randn(n_samples, n_features)
```

```
>>> logisticR = LogisticRegression()  
>>> logisticR.fit(X, y)  
>>> pred= logisticR.predict(X)
```

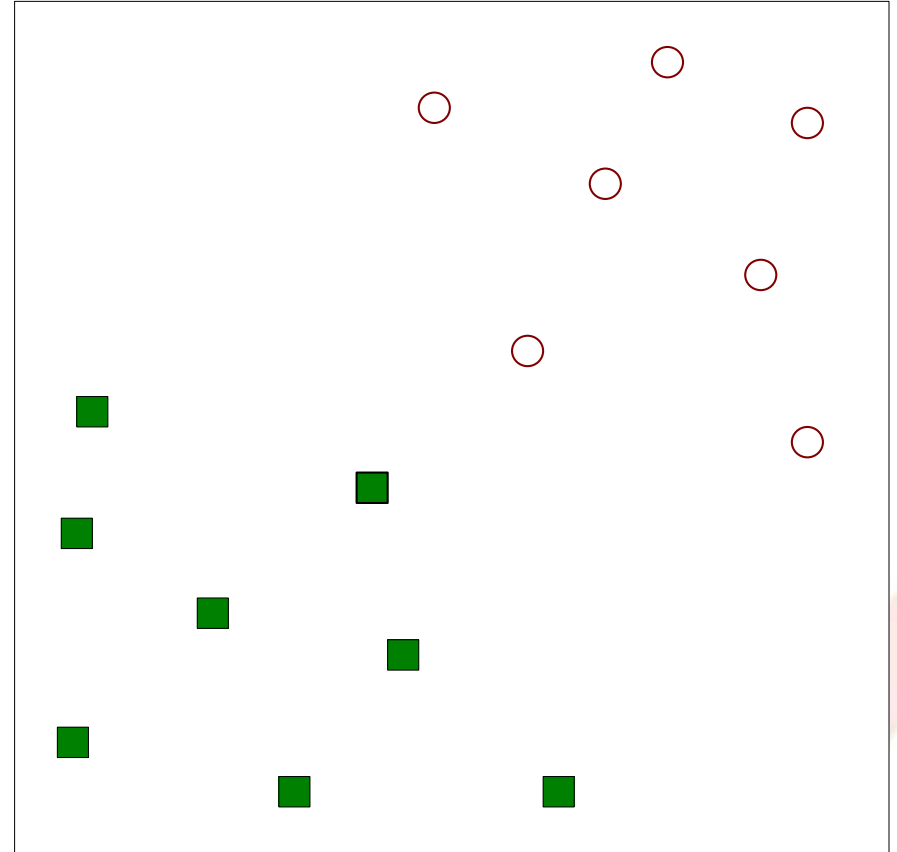
Model training and testing

Support Vector Machines

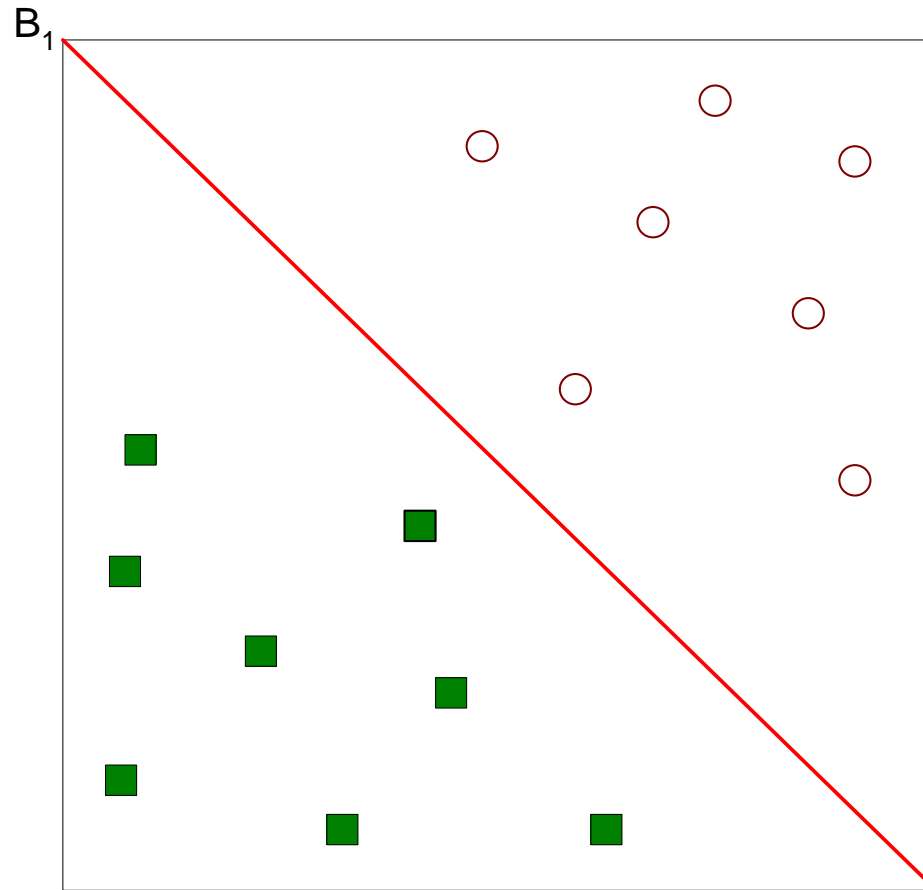
- To learn a binary classifier



- To find a hyperplane (also known as decision boundary) such that all the squares reside on one side of the hyperplane and all the circles reside on the other side

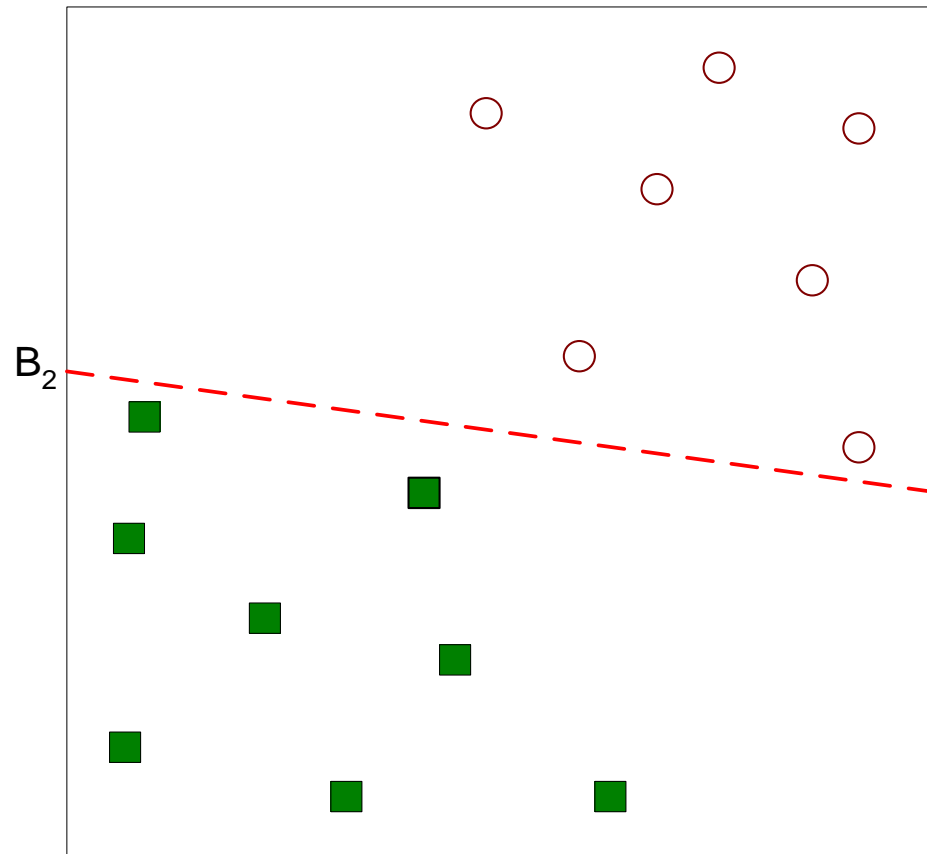


A Possible Decision Boundary



One Possible Solution

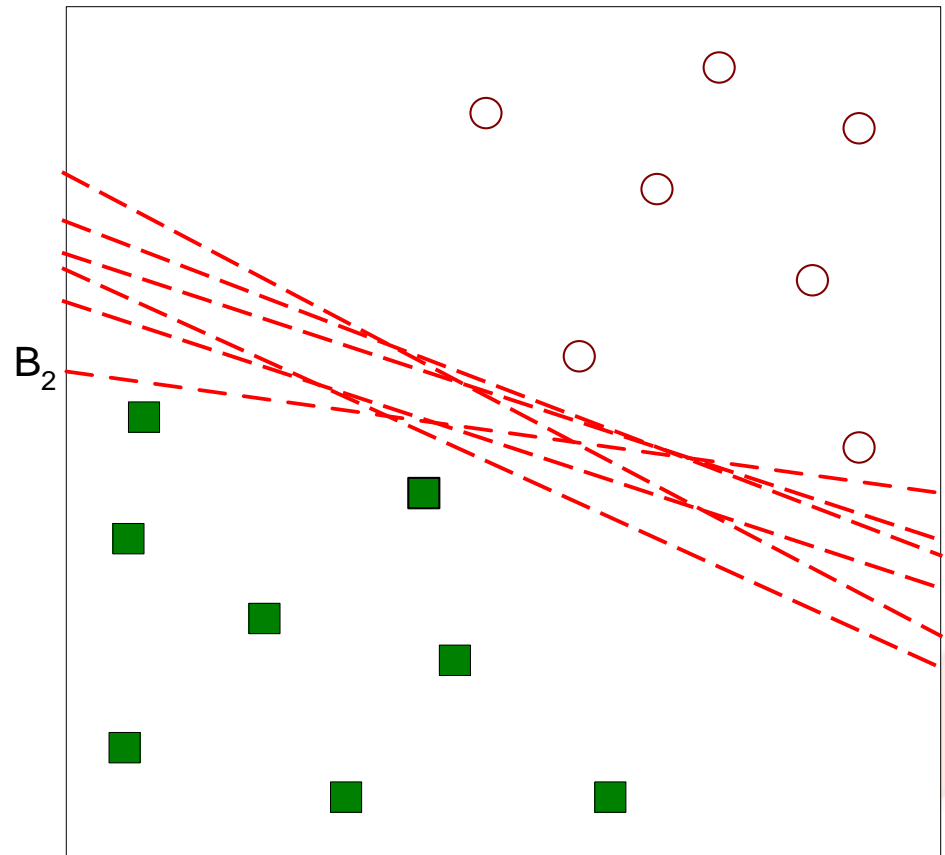
Another Possible Decision Boundary



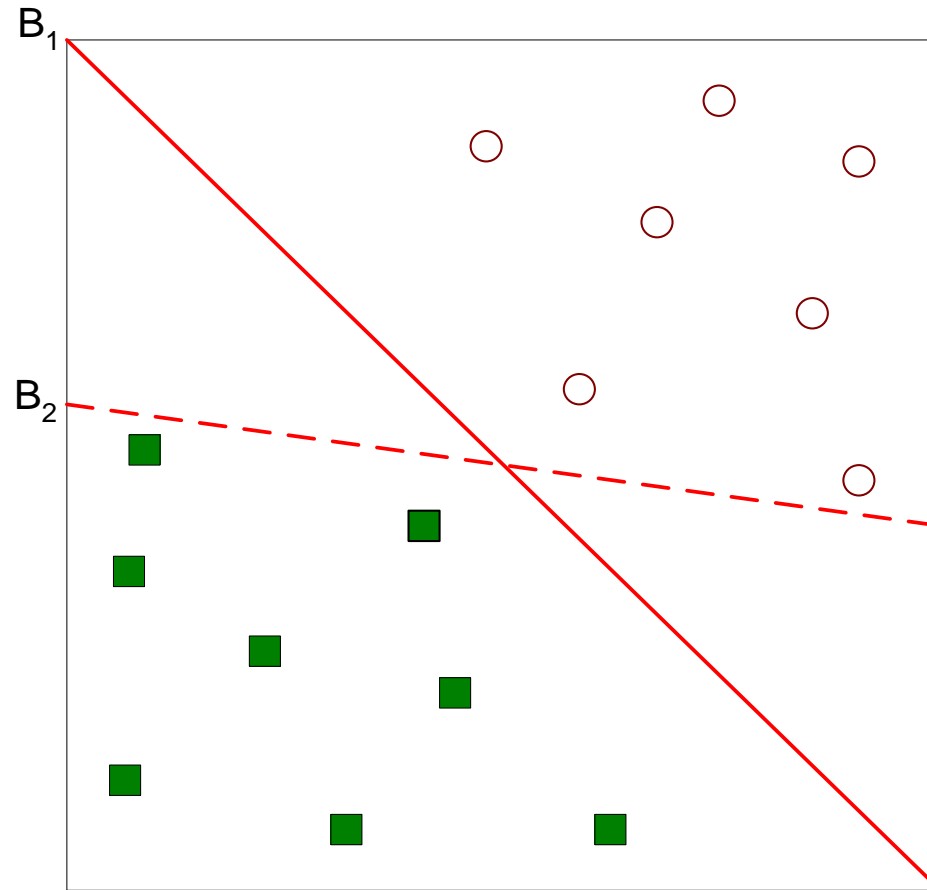
Another possible solution

Many Possible Decision Boundary

- Though all the shown decision boundaries can separate training examples perfectly, their test errors may be different
- Which one should be used to construct the classifier?

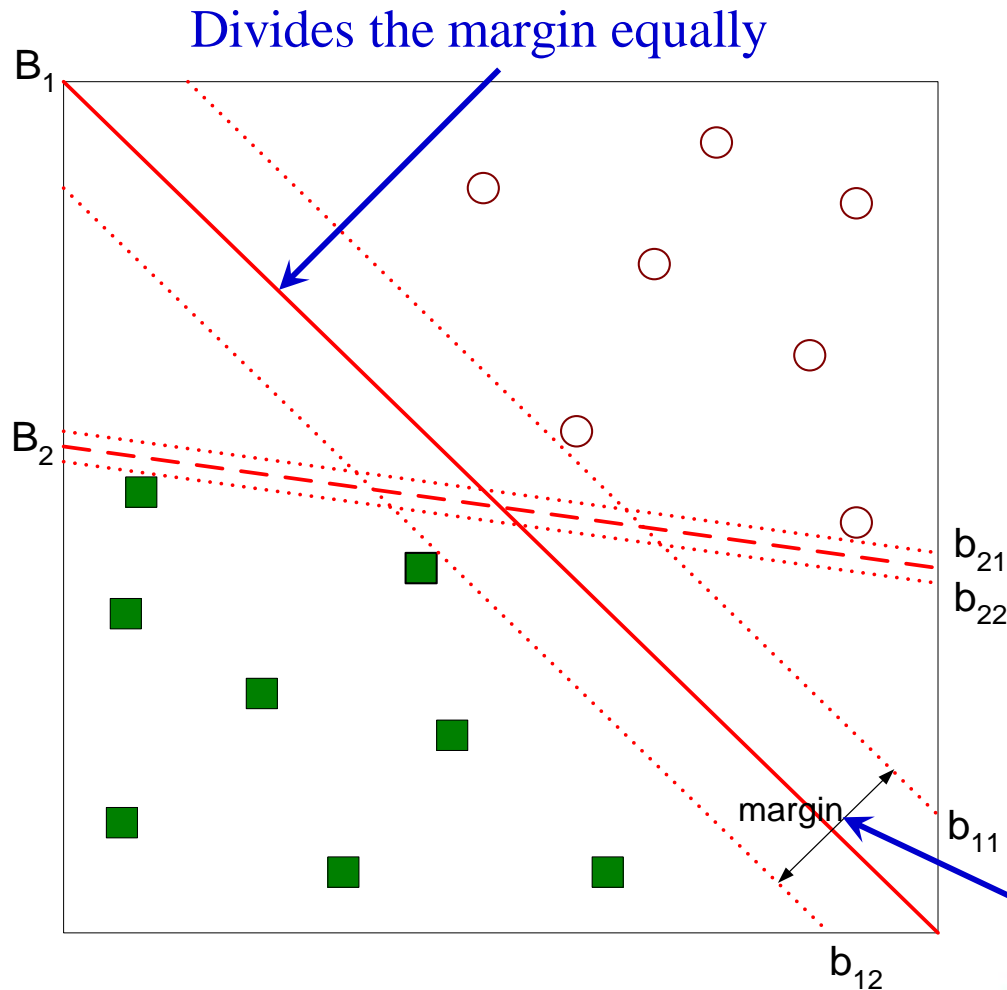


Decision Boundaries Comparison



Which one is better? B1 or B2?


Margin of Decision Boundary



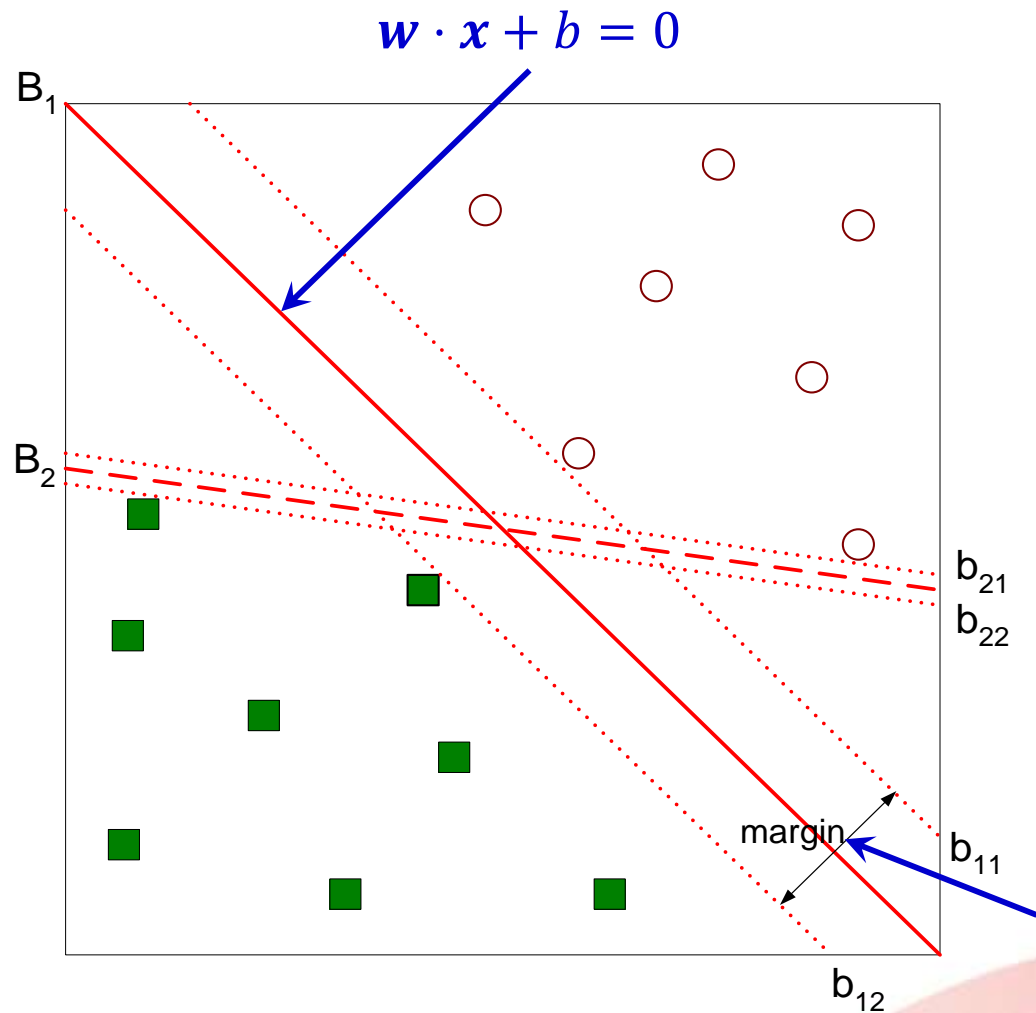
- Each decision boundary B_i is associated with a pair of parallel hyperplanes: b_{i1} and b_{i2}
- b_{i1} is obtained by moving the hyperplane until it touches the closest circle(s)
- b_{i2} is obtained by moving a hyperplane away from the decision boundary until it touches the closest square(s)

The distance between the parallel hyperplanes is known as the margin of the decision boundary

Support Vector Machines

- Support Vector Machines (SVMs) aim to learn a linear decision boundary whose margin is largest over the training data instances
 - SVMs are one of the most classical machine learning methods
 - In the past (in 90's and 00's), SVMs have shown promising empirical results in many practical applications, such as computer vision, sensor networks and text mining
- 

How to Represent A Margin?



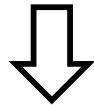
Detailed induction will
be presented next week

Optimization Problem

Detailed induction will
be presented next week

- Optimization problem of linear SVMs (separable case)

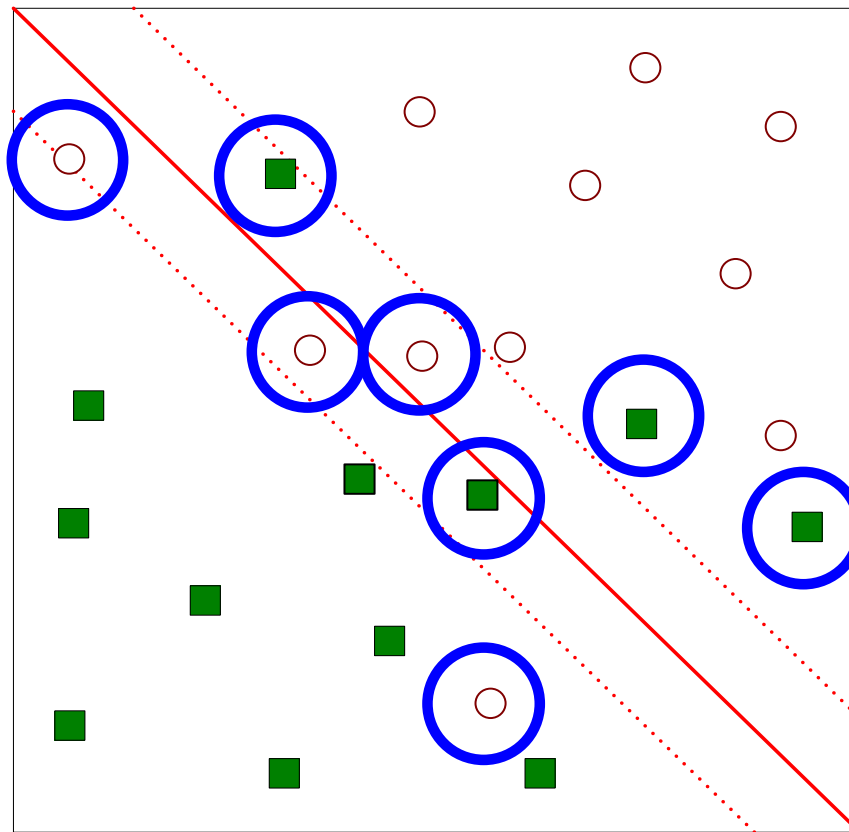
$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{\|\mathbf{w}\|_2^2}{2} \\ \text{s.t.} \quad & \mathbf{w} \cdot \mathbf{x}_i + b \geq 1, \text{ if } y_i = 1, \\ & \mathbf{w} \cdot \mathbf{x}_i + b \leq -1, \text{ if } y_i = -1, \\ & i = 1, \dots, N \end{aligned}$$



$$\begin{aligned} \min_{\mathbf{w}, b} \quad & \frac{\|\mathbf{w}\|_2^2}{2} \\ \text{s.t.} \quad & y_i \times (\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad i = 1, \dots, N \end{aligned}$$

Linear SVMs: Non-separable Case

- What if data instances of the two class are not separable?



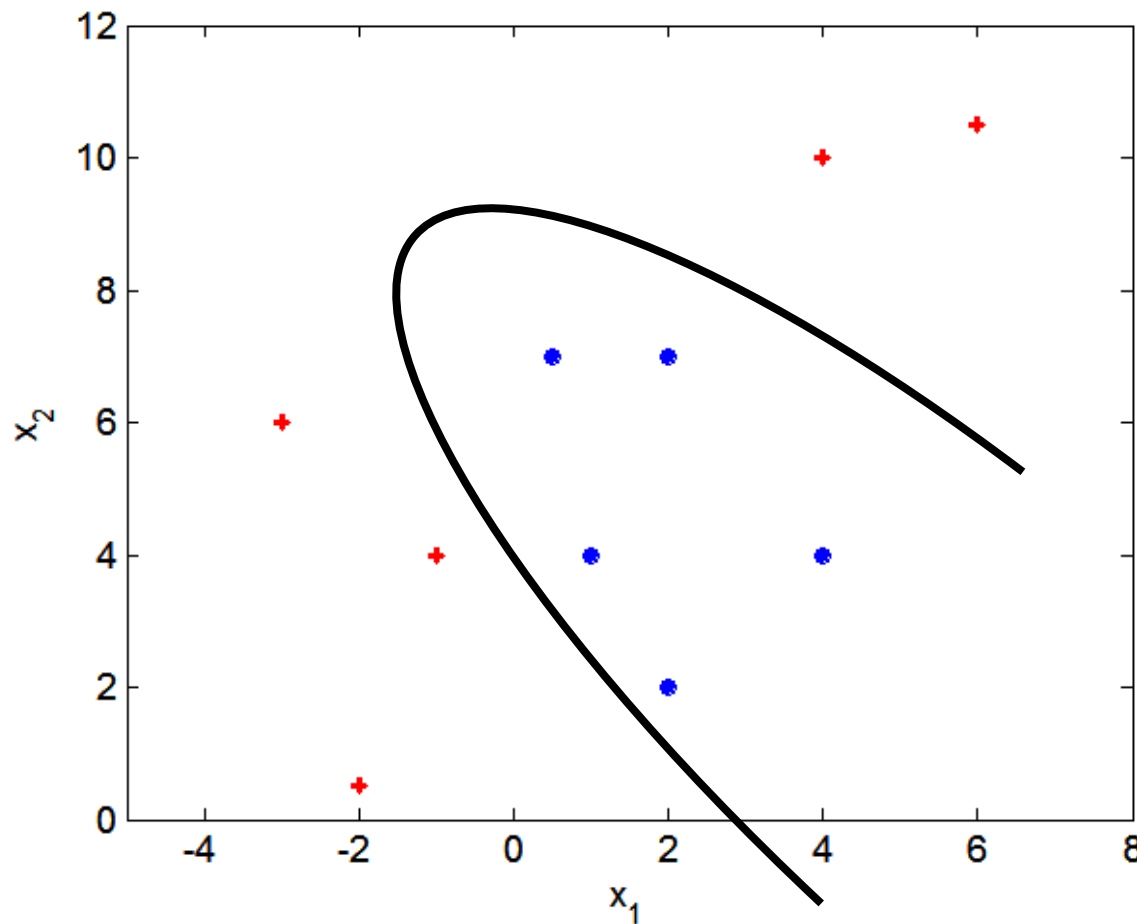
Slack variables $\xi_i \geq 0$ need to be introduced to absorb errors

Detailed induction will be presented next week

Nonlinear SVMs

Detailed induction will be presented next week

- What if the decision boundary is not linear?



Kernel methods

Thank you!

