AI6102: Machine Learning Methodologies & Applications

L4: Linear Models: Classification

Dacheng Tao

dacheng.tao@ntu.edu.sg

Nanyang Technological University, Singapore

Outline

- Linear regression revisit
- Linear models for classification
  – Logistic Regression
  – Support Vector Machines (basic idea, details will be taught in next lecture)

Linear Regression Revisit

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

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

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

OR $f(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{w} \cdot \boldsymbol{x}$ by introducing additional w_0 and x_0 to absorb b in $\boldsymbol{w}$
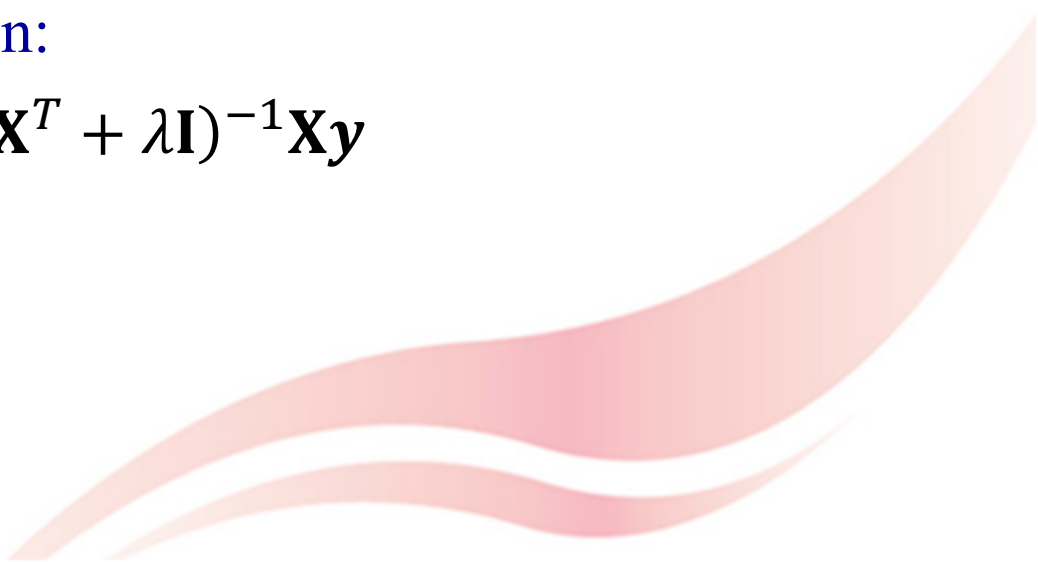
- The loss function $\ell(f(\boldsymbol{x}_i; \boldsymbol{\theta}), y_i)$ is defined as the squared difference between $f(\boldsymbol{x}_i; \boldsymbol{\theta})$ and y_i

- The regularization term is defined as the L2 norm

Linear Regression Revisit (cont.)

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

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

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

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

Linear Models for Classification

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

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

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

  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,
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0, \text{where } \boldsymbol{w}, \boldsymbol{x} \in \mathbb{R}^m$$
  defines a hyperplane in the m-dimensional space, and separate the m-dimensional space into two regions

- For a data instance $\boldsymbol{x}_i$
  - If $\boldsymbol{w} \cdot \boldsymbol{x}_i + b = 0$, then $\boldsymbol{x}_i$ is on the hyperplane
  - If $\boldsymbol{w} \cdot \boldsymbol{x}_i + b > 0$, then $\boldsymbol{x}_i$ is on one side of the hyperplane
  - If $\boldsymbol{w} \cdot \boldsymbol{x}_i + b < 0$, then $\boldsymbol{x}_i$ 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
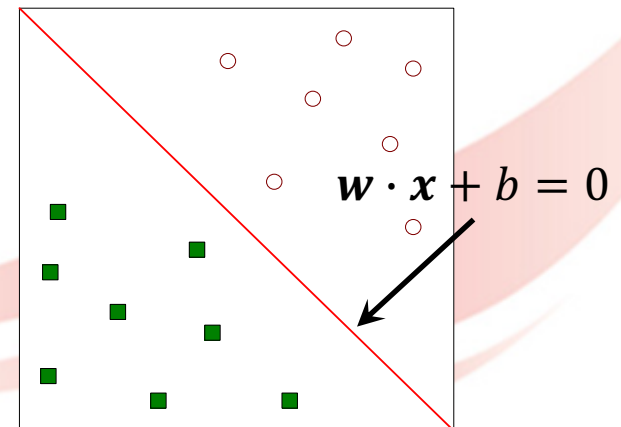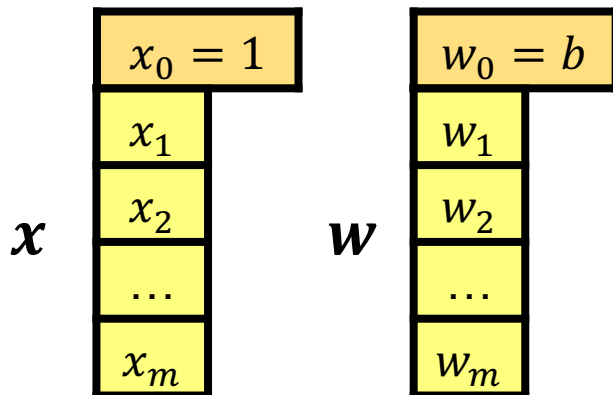
A Compact Form of Hyperplane

- Recall from Lecture 3

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

⇩

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

$$\boldsymbol{x} \quad \begin{array}{|c|} \hline x_0 = 1 \\ \hline x_1 \\ \hline x_2 \\ \hline \dots \\ \hline x_m \\ \hline \end{array} \qquad \boldsymbol{w} \quad \begin{array}{|c|} \hline w_0 = b \\ \hline w_1 \\ \hline w_2 \\ \hline \dots \\ \hline w_m \\ \hline \end{array}$$

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

⇩

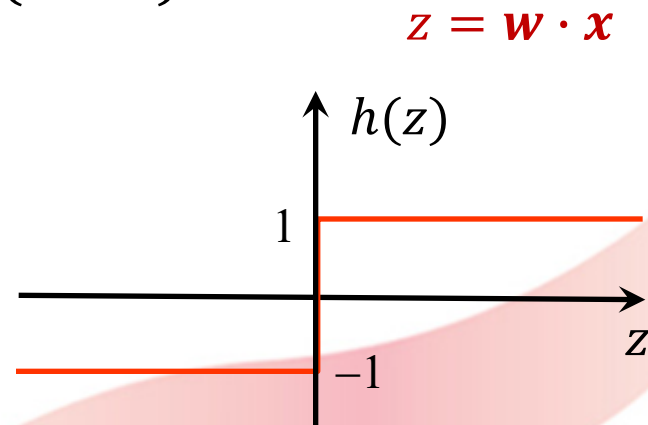$$\sum_{i=0}^{m} w_i x_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{m} w_i x_i + \boxed{w_0 x_0}^{\,b} = 0$$

Hyperplane as A Classifier

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

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

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

$$h(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$
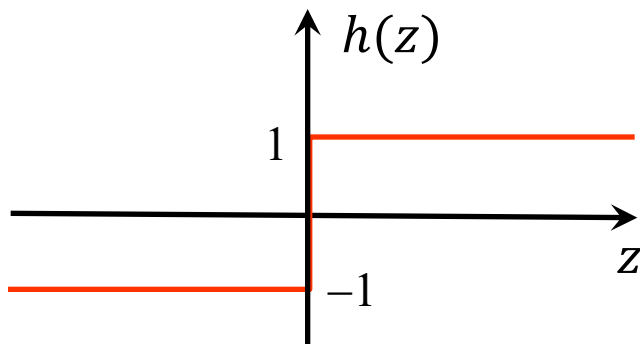
This is also known as a sign or threshold function

Hyperplane as A Classifier (cont.)

- The sign function is NOT differentiable <u>everywhere</u>
- Need to look for a more smooth function

$$f(x; \theta) = h(w \cdot x)$$

sigmoid/logistic function

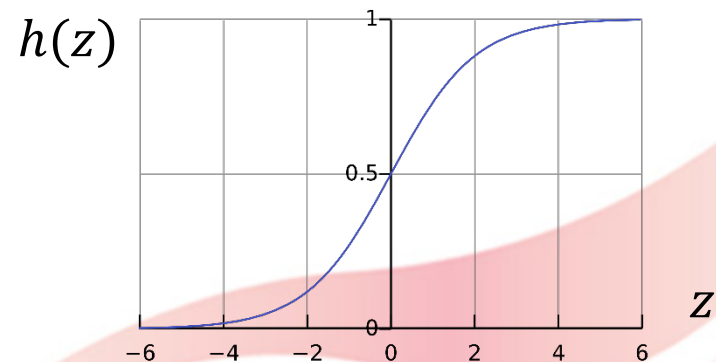$$h(z) = \begin{cases} +1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

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

$$z = w \cdot x$$
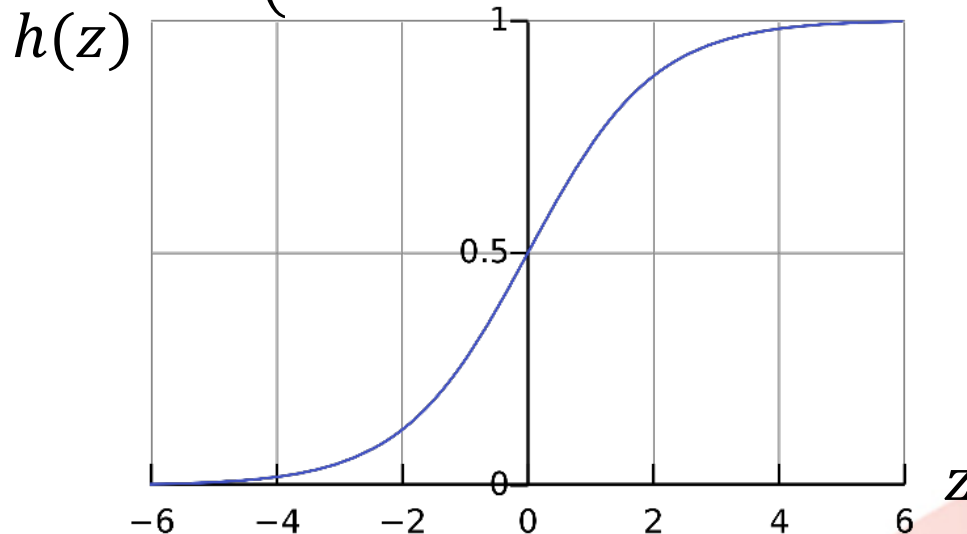
Hyperplane as A Classifier (cont.)

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

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

$h(z)$

$\{-1, +1\} \rightarrow \{0,1\}$

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

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 $\{\boldsymbol{x}_i, y_i\}, i = 1, \ldots, N$, where $\boldsymbol{x}_i$ is $(m + 1)$-dimensional $(x_{0i} = 1)$, and y_i is either 0 or 1
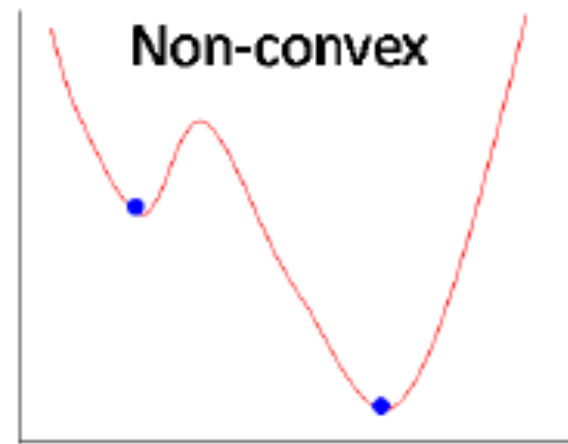
$$\sum_{i=1}^{N} \ell(h(\boldsymbol{x}_i; \boldsymbol{w}), y_i) = \sum_{i=1}^{N} (h(\boldsymbol{x}_i; \boldsymbol{w}) - y_i)^2$$

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

Loss Function (cont.)

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

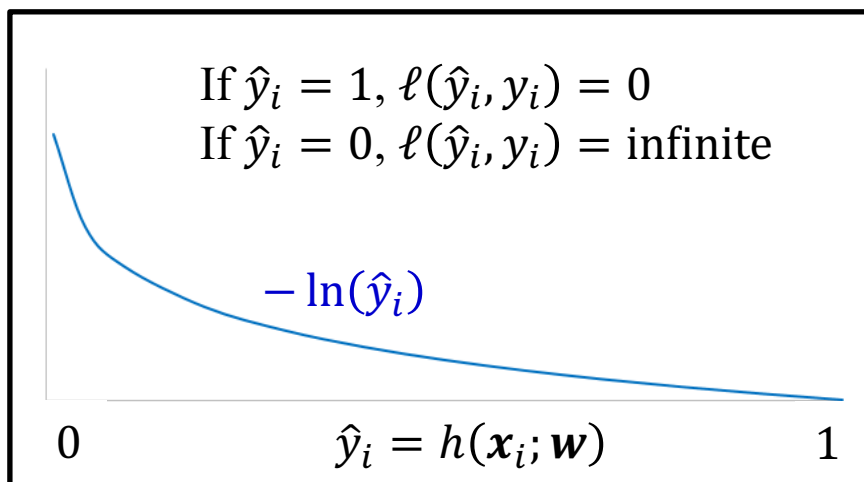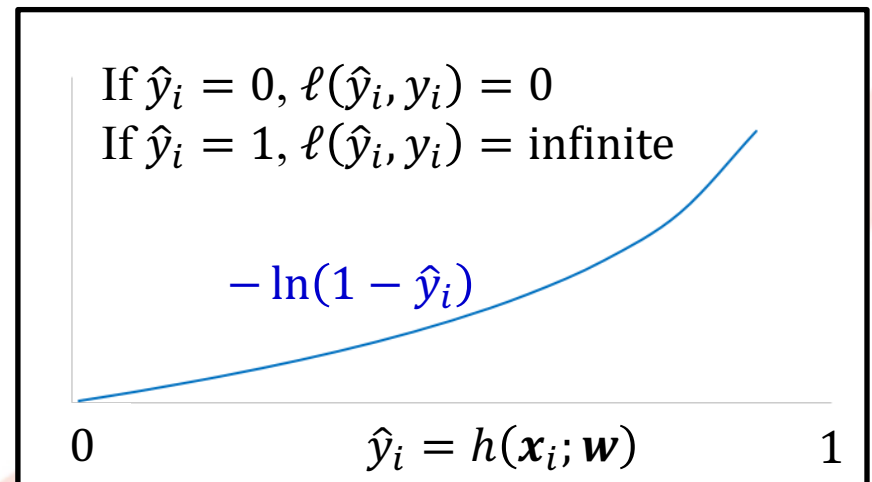- What is the problem if the objective is not convex?

Convex

Non-convex
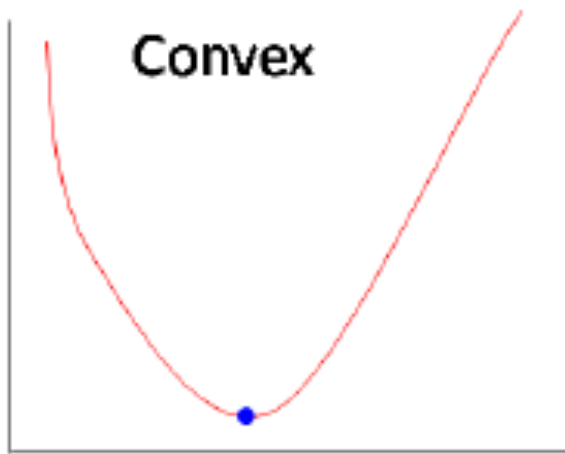
Logistic Regression Loss

- Define the loss function as

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

$$y_i = 1 \qquad\qquad y_i = 0$$

If $\hat{y}_i = 1$, $\ell(\hat{y}_i, y_i) = 0$
If $\hat{y}_i = 0$, $\ell(\hat{y}_i, y_i) = $ infinite

$-\ln(\hat{y}_i)$

$0 \qquad \hat{y}_i = h(\boldsymbol{x}_i; \boldsymbol{w}) \qquad 1$

If $\hat{y}_i = 0$, $\ell(\hat{y}_i, y_i) = 0$
If $\hat{y}_i = 1$, $\ell(\hat{y}_i, y_i) = $ infinite

$-\ln(1 - \hat{y}_i)$
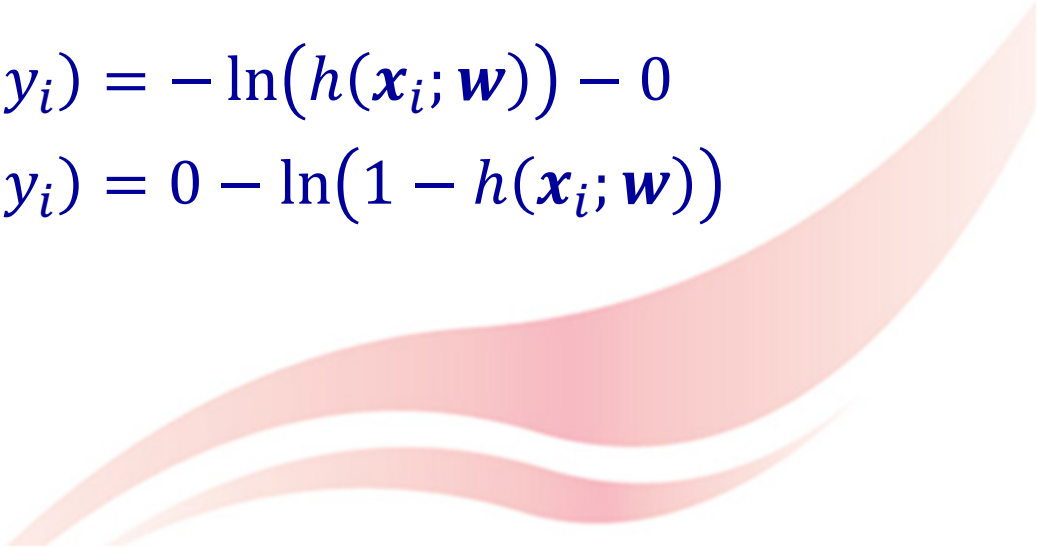
$0 \qquad \hat{y}_i = h(\boldsymbol{x}_i; \boldsymbol{w}) \qquad 1$

Logistic Regression Loss (cont.)

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

- The above loss can be simplified as

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

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

Logistic Regression Objective

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

$$\Downarrow$$

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

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

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

Probabilistic Point of View

$$z = \boldsymbol{w}^T \boldsymbol{x} \qquad y = \frac{1}{1 + \exp(-z)}$$
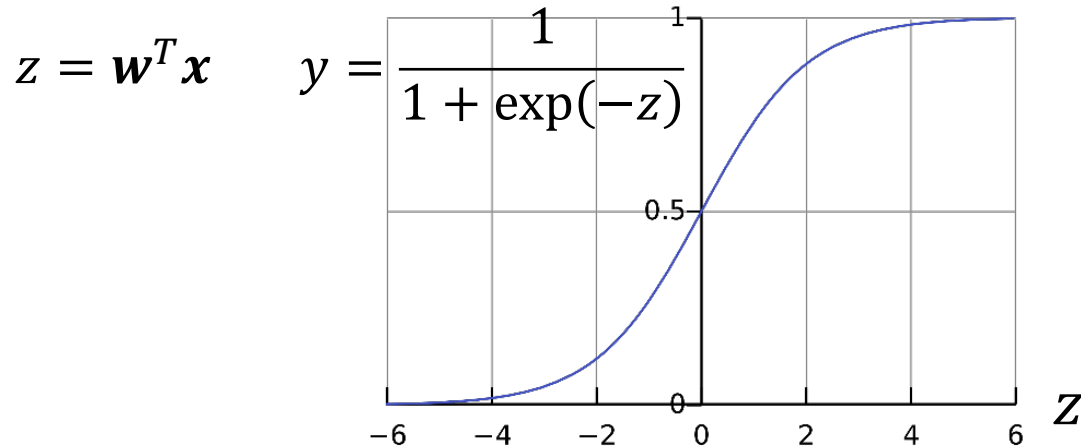
- Assume the conditional probability of class 1 is modeled as

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

- Thus, the conditional probability of class 0 is

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

Probability Review

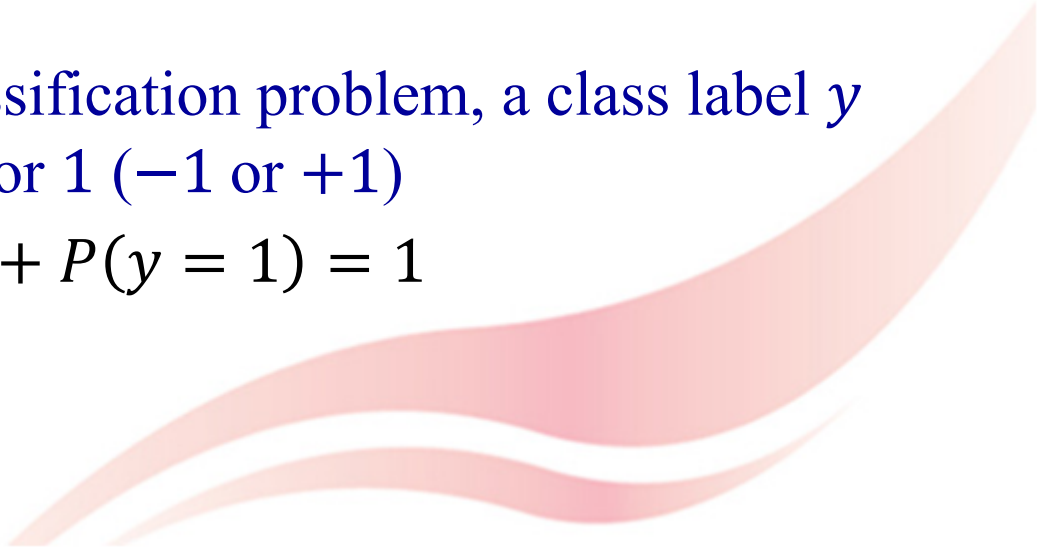- Let A be a random variable (a feature / a label in machine learning)

- 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 $(\boldsymbol{x}_i, y_i)$, its joint probability is
$$P(\boldsymbol{x} = \boldsymbol{x}_i, y = y_i)$$

OR

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

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 $\boldsymbol{x}_i$

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

Parametric Form

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

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

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

Maximum Likelihood Estimation

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

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

  - Ideally, for each training pair $\{\boldsymbol{x}_i, y_i\}$, $l(\boldsymbol{w}|\{\boldsymbol{x}_i, y_i\}) = P(y_i|\boldsymbol{x}_i; \boldsymbol{w}) = 1$

- Maximum likelihood estimation (MLE) aims to find a solution of $\boldsymbol{w}$ such that $P(y_i|\boldsymbol{x}_i; \boldsymbol{w})$ is maximized

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

  - 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 $\{\boldsymbol{x}_i, y_i\}, i = 1, \ldots, N$, which are i.i.d., the likelihood is defined as the product of the likelihoods of each individual data pairs

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

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

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

MLE (cont.)

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

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

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

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

$$\ln\mathcal{L}(\boldsymbol{w}) = \ln\left(\prod_{i=1}^{N} P(y_i | \boldsymbol{x}_i; \boldsymbol{w})\right) = \boxed{\sum_{i=1}^{N} \ln P(y_i | \boldsymbol{x}_i; \boldsymbol{w})}$$
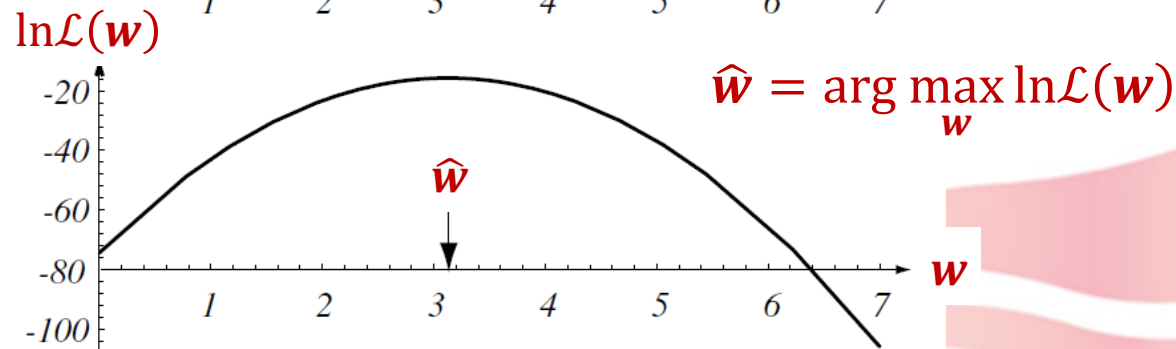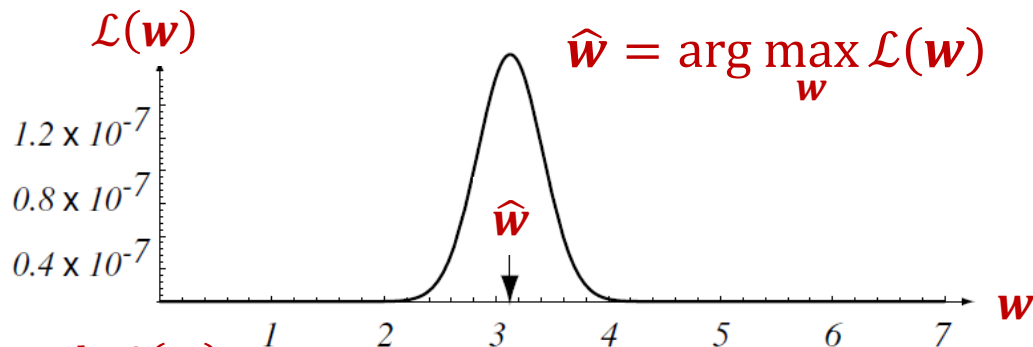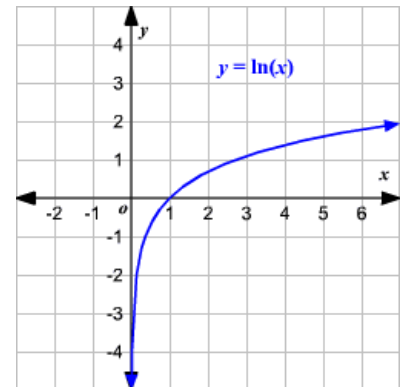
  2. The $\ln(\cdot)$ function is a strictly increasing function, the solution of $\boldsymbol{w}$ remains the same

$$\widehat{\boldsymbol{w}} = \arg \max_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}) \Longleftrightarrow \widehat{\boldsymbol{w}} = \arg \max_{\boldsymbol{w}} \ln\mathcal{L}(\boldsymbol{w})$$

Maximum Log-Likelihood

- How to understand "the $\ln(\cdot)$ function is a strictly increasing function, the solution of w remains the same"

$$\widehat{w} = \arg\max_{w} \mathcal{L}(w) \iff \widehat{w} = \arg\max_{w} \ln\mathcal{L}(w)$$

Maximum Log-Likelihood (cont.)

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

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

$$\widehat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} \sum_{i=1}^{N} \ln\left(h(\boldsymbol{x}_i; \boldsymbol{w})^{y_i} (1 - h(\boldsymbol{x}_i; \boldsymbol{w}))^{1-y_i}\right)$$

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

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

Maximum Log-Likelihood (cont.)

$$\widehat{\boldsymbol{w}} = \arg\max_{\boldsymbol{w}} \sum_{i=1}^{N} \left[y_i \ln\big(h(\boldsymbol{x}_i; \boldsymbol{w})\big) + (1 - y_i) \ln\big(1 - h(\boldsymbol{x}_i; \boldsymbol{w})\big) \right]$$

Equivalent!
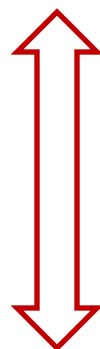
$$\widehat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} - \sum_{i=1}^{N} \left[y_i \ln\big(h(\boldsymbol{x}_i; \boldsymbol{w})\big) + (1 - y_i) \ln\big(1 - h(\boldsymbol{x}_i; \boldsymbol{w})\big) \right]$$

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 <u>odds</u> 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

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

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

Logit + Regression

- Denote by

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

- The logit of p is

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

logit

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

logistic regression

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

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

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

Approaches to Solution

Objective $E(\boldsymbol{w})$

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

- 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

$$w_{t+1} = w_t - \rho \frac{\partial E(w)}{\partial w}$$

$$\frac{\partial E(w)}{\partial w} = \frac{\partial\left(-\sum_{i=1}^{N}\left[y_i \ln\big(h(x_i; w)\big) + (1 - y_i)\ln\big(1 - h(x_i; w)\big)\right]\right)}{\partial w}$$

$$= -\sum_{i=1}^{N} \frac{\partial\left[y_i \ln\big(h(x_i; w)\big) + (1 - y_i)\ln\big(1 - h(x_i; w)\big)\right]}{\partial w}$$

$$= -\sum_{i=1}^{N}\left(\frac{\partial\big(y_i \ln\big(h(x_i; w)\big)\big)}{\partial w} + \frac{\partial\left((1 - y_i)\ln\big(1 - h(x_i; w)\big)\right)}{\partial w}\right)$$

Optimization (cont.)

$$\frac{\partial\big(y_i \ln\big(h(\boldsymbol{x}_i; \boldsymbol{w})\big)\big)}{\partial \boldsymbol{w}}$$

$$\textcolor{red}{\frac{\partial \ln f(z)}{\partial z} = \frac{\partial \ln f(z)}{\partial f(z)} \frac{\partial f(z)}{\partial z} = \frac{1}{f(z)} \frac{\partial f(z)}{\partial z}}$$

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

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

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

$$\textcolor{red}{\frac{\partial f(z)^k}{\partial z} = \frac{\partial f(z)^k}{\partial f(z)} \frac{\partial f(z)}{\partial z} = k f(z)^{k-1} \frac{\partial f(z)}{\partial z}}$$

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

Optimization (cont.)

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

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

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

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

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

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

Optimization (cont.)

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

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

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

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

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

Optimization (cont.)
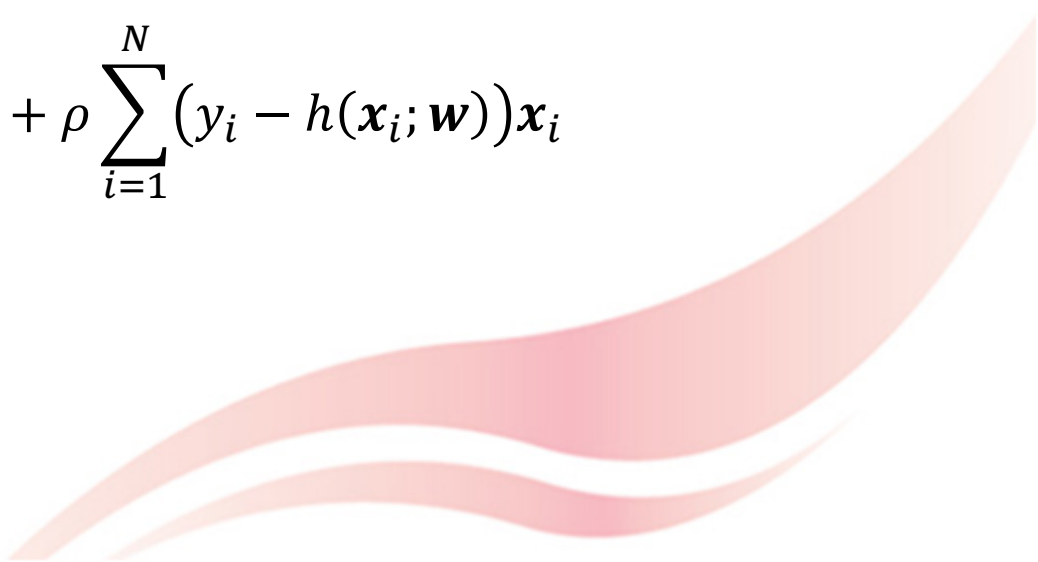
$$\frac{\partial \Big((1-y_i)\ln\big(1-h(\boldsymbol{x}_i;\boldsymbol{w})\big)\Big)}{\partial \boldsymbol{w}} = (y_i-1)\frac{1}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x}_i)}\,\boldsymbol{x}_i$$

$$= (y_i-1)h(\boldsymbol{x}_i;\boldsymbol{w})\boldsymbol{x}_i$$

$$\frac{\partial\big(y_i\ln\big(h(\boldsymbol{x}_i;\boldsymbol{w})\big)\big)}{\partial \boldsymbol{w}} = y_i\frac{\exp(-\boldsymbol{w}^T\boldsymbol{x}_i)}{1+\exp(-\boldsymbol{w}^T\boldsymbol{x}_i)}\,\boldsymbol{x}_i$$

$$= y_i\big(1-h(\boldsymbol{x}_i;\boldsymbol{w})\big)\boldsymbol{x}_i$$

$$\frac{\partial \Big((1-y_i)\ln\big(1-h(\boldsymbol{x}_i;\boldsymbol{w})\big)\Big)}{\partial \boldsymbol{w}} + \frac{\partial\big(y_i\ln\big(h(\boldsymbol{x}_i;\boldsymbol{w})\big)\big)}{\partial \boldsymbol{w}} = \big(y_i-h(\boldsymbol{x}_i;\boldsymbol{w})\big)\boldsymbol{x}_i$$

Optimization (cont.)

$$\frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = -\sum_{i=1}^{N}\left(\frac{\partial\big(y_i \ln(h(\boldsymbol{x}_i; \boldsymbol{w}))\big)}{\partial \boldsymbol{w}} + \frac{\partial\Big((1-y_i)\ln\big(1-h(\boldsymbol{x}_i; \boldsymbol{w})\big)\Big)}{\partial \boldsymbol{w}}\right)$$

$$= -\sum_{i=1}^{N}\big(y_i - h(\boldsymbol{x}_i; \boldsymbol{w})\big)\boldsymbol{x}_i$$

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

Recall: Approaches to Solution

Objective $E(\boldsymbol{w})$

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

- 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 $\{x_t\}, t = 0, 1, ...,$ 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

$$x^* = \arg\min_x g(x)$$

$g(x)$

x^* x_2 x_1
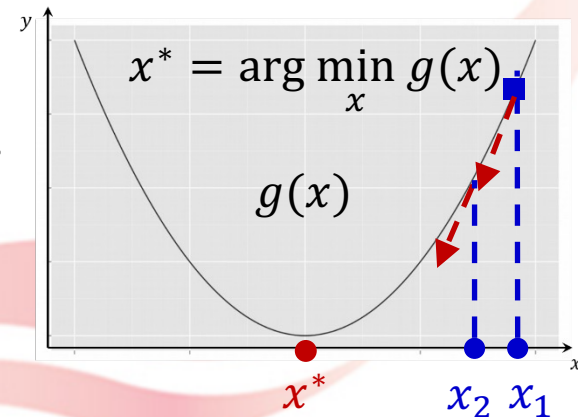
Regularized Logistic Regression

Objective $E(\boldsymbol{w})$

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

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

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

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

Extension to Multiple Classes

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

$$\text{For } c > 0: \quad P(y = c | \boldsymbol{x}) = \frac{\exp\left(-\boldsymbol{w}^{(c)T}\boldsymbol{x}\right)}{1 + \sum_{c=1}^{C-1} \exp\left(-\boldsymbol{w}^{(c)T}\boldsymbol{x}\right)}$$

$$\text{For } c = 0: \quad P(y = 0 | \boldsymbol{x}) = \frac{1}{1 + \sum_{c=1}^{C-1} \exp\left(-\boldsymbol{w}^{(c)T}\boldsymbol{x}\right)}$$

$$\left. \right\} \quad \sum_{c=0}^{C-1} P(y = c | \boldsymbol{x}) = 1$$

- For learning each $\boldsymbol{w}^{(c)}$, the procedure is basically the same as what we derived!

- For a test data instance $\boldsymbol{x}^*$, $y^* = \arg\max_{c \in \{0, \ldots, C-1\}} P(y = c | \boldsymbol{x}^*)$

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

⬇
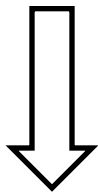
- To find a hyperplane (also known as decision boundary) such that all the squares reside on one side of the hyperplane and all the circles reside on the other side
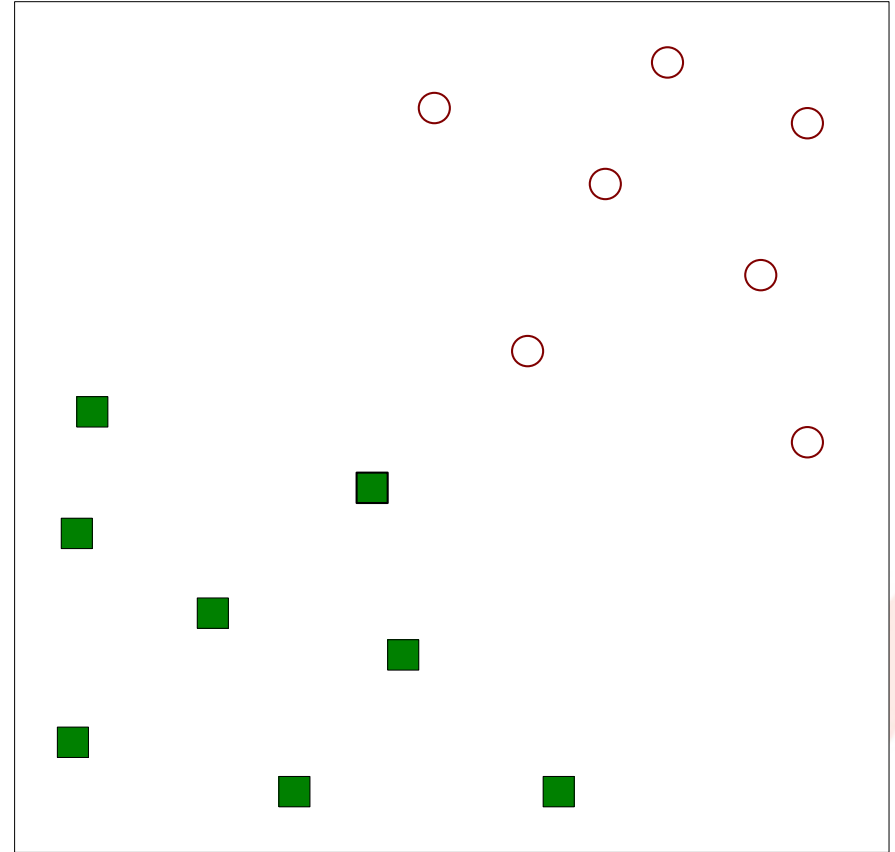
A Possible Decision Boundary
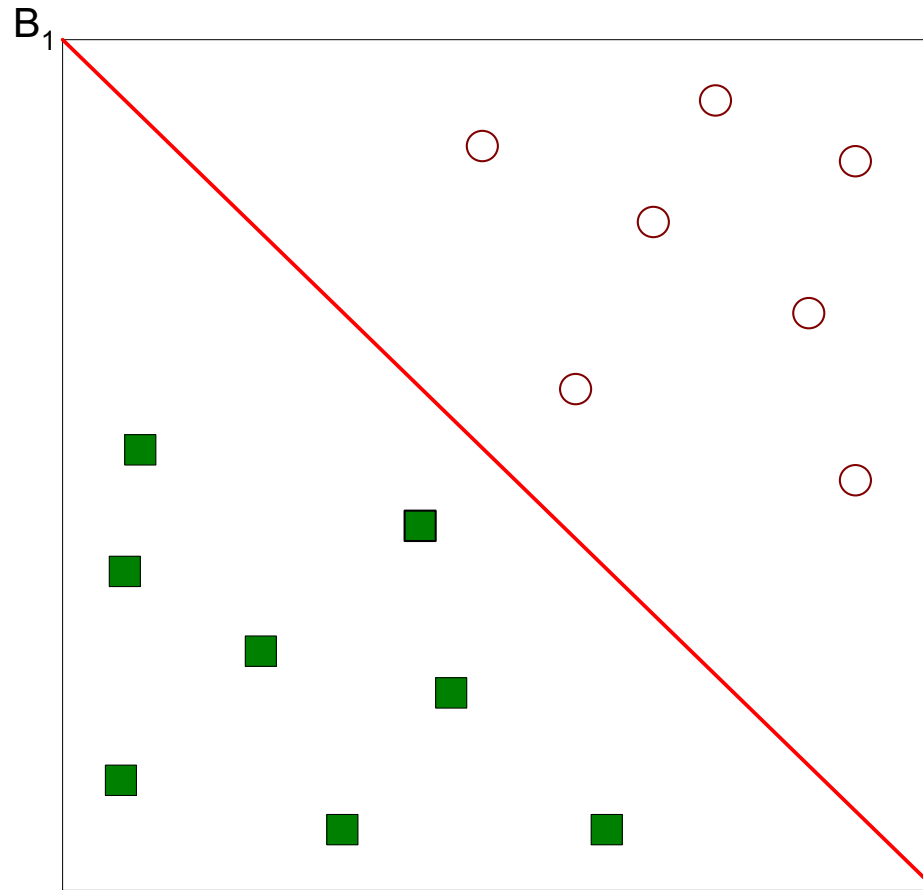
One Possible Solution

Another Possible Decision Boundary
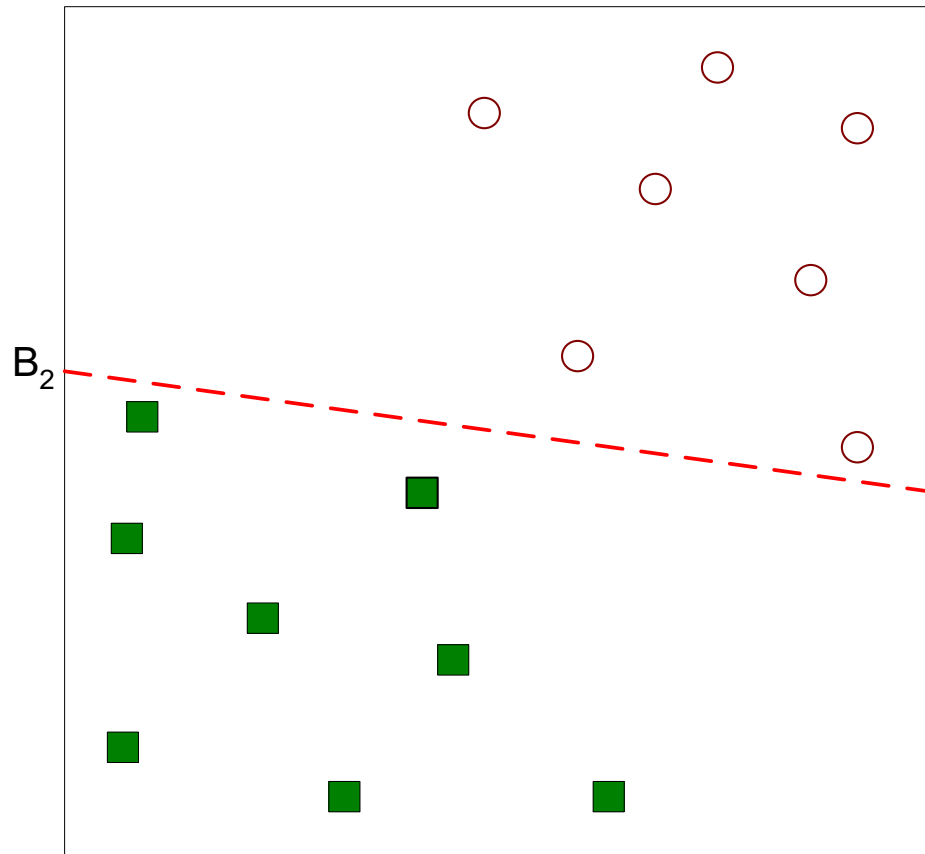
Another possible solution

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?

B_2
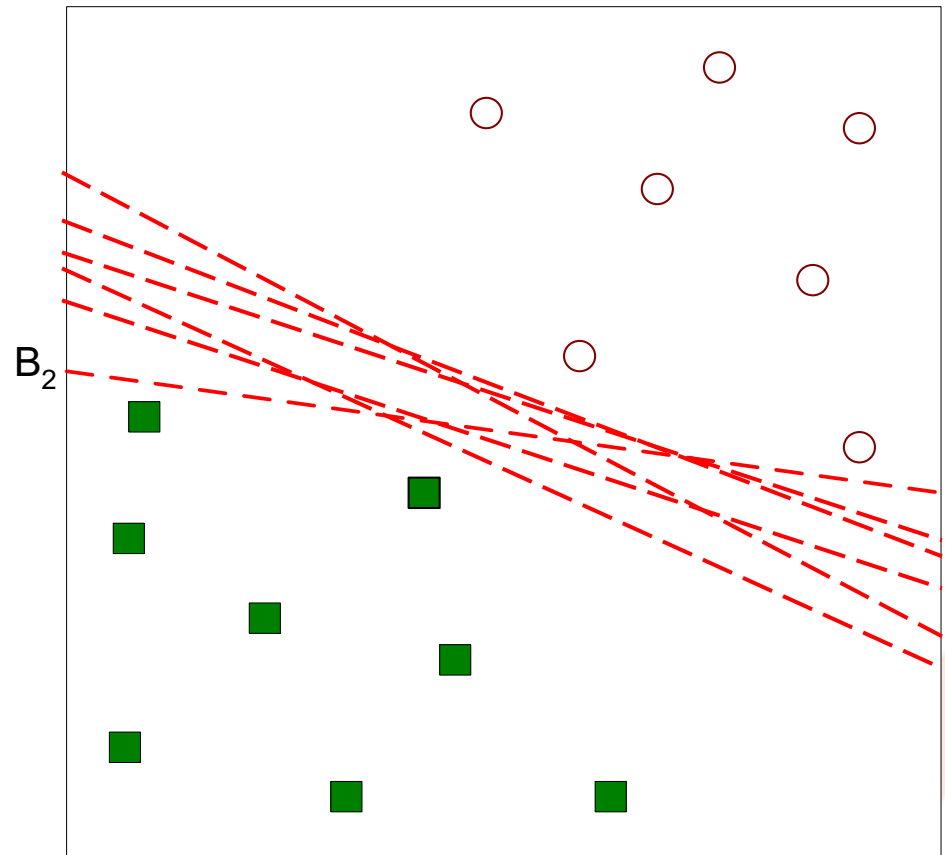
Decision Boundaries Comparison
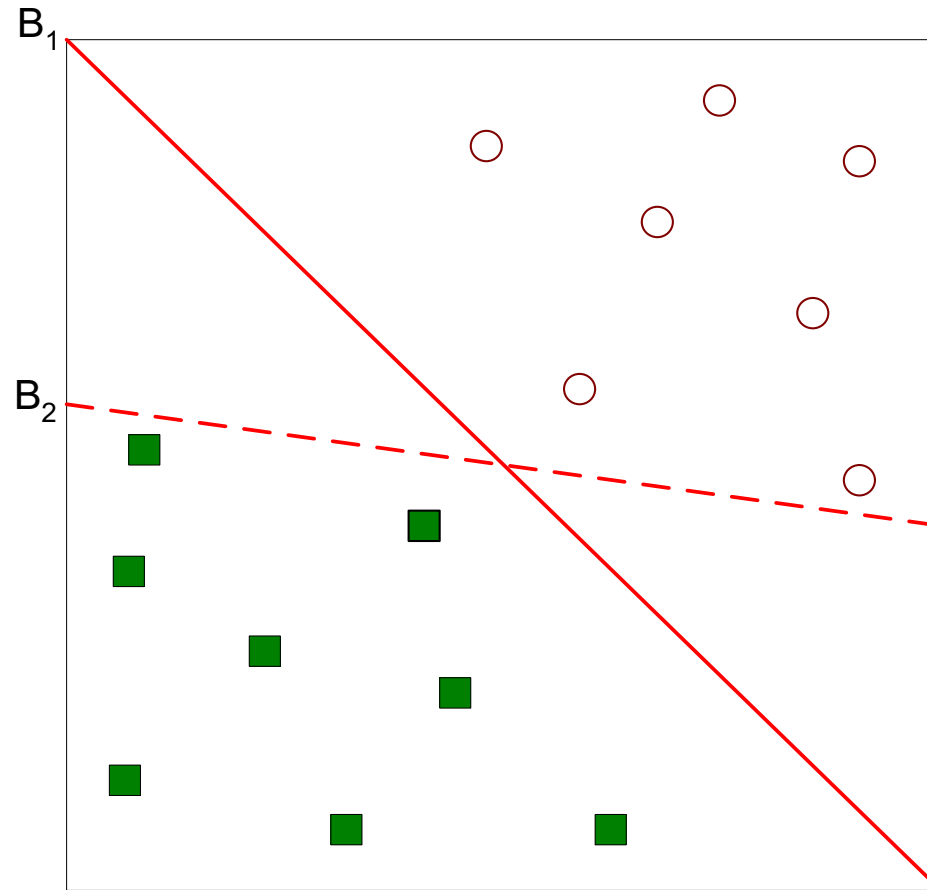
Which one is better? B1 or B2?

Margin of Decision Boundary

Divides the margin equally

B_1

B_2

margin

b_{21}
b_{22}

b_{11}

b_{12}
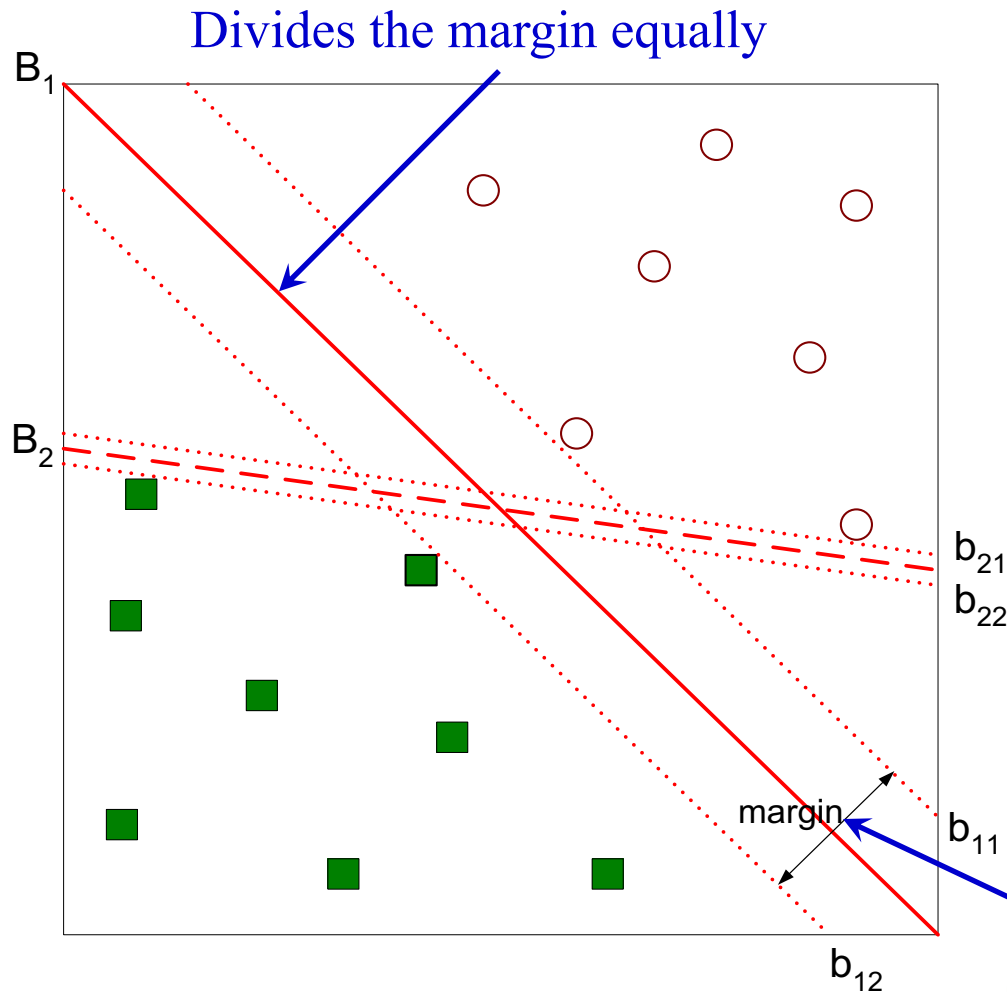
- Each decision boundary B_i is associated with a pair of parallel hyperplanes: b_{i1} and b_{i2}

- 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?

$w \cdot x + b = 0$

B_1

B_2

b_{21}
b_{22}

margin

b_{11}

b_{12}

$\dfrac{2}{\|w\|_2}$
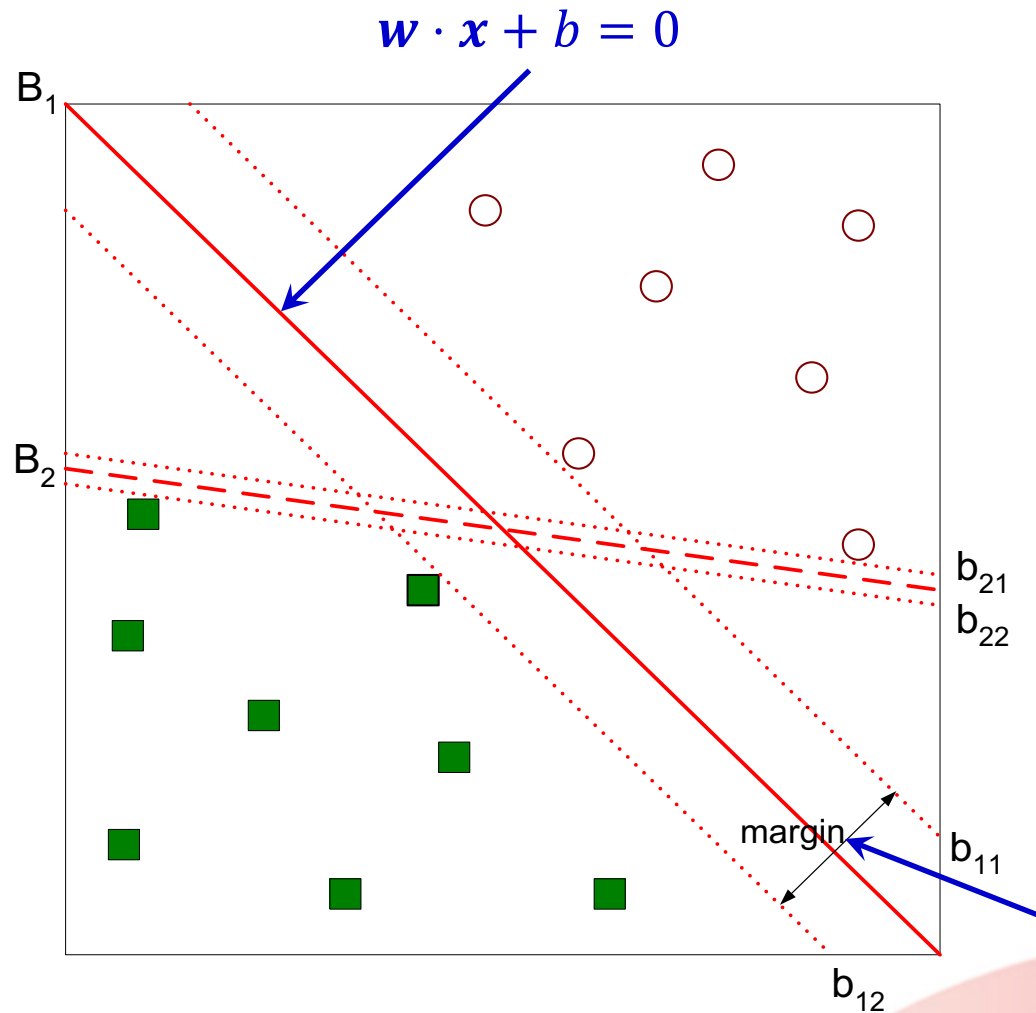
Detailed induction will be presented next week

Optimization Problem

- Optimization problem of linear SVMs (separable case)

$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$

$$\text{s.t.} \quad \boldsymbol{w} \cdot \boldsymbol{x}_i + b \geq 1, \text{if } y_i = 1,$$

$$\boldsymbol{w} \cdot \boldsymbol{x}_i + b \leq -1, \text{if } y_i = -1,$$

$$i = 1, \dots, N$$

$$\Downarrow$$
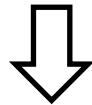
$$\min_{\boldsymbol{w},b} \frac{\|\boldsymbol{w}\|_2^2}{2}$$

$$\text{s.t.} \quad y_i \times (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1, \ i = 1, \dots, N$$

Linear SVMs: Non-separable Case

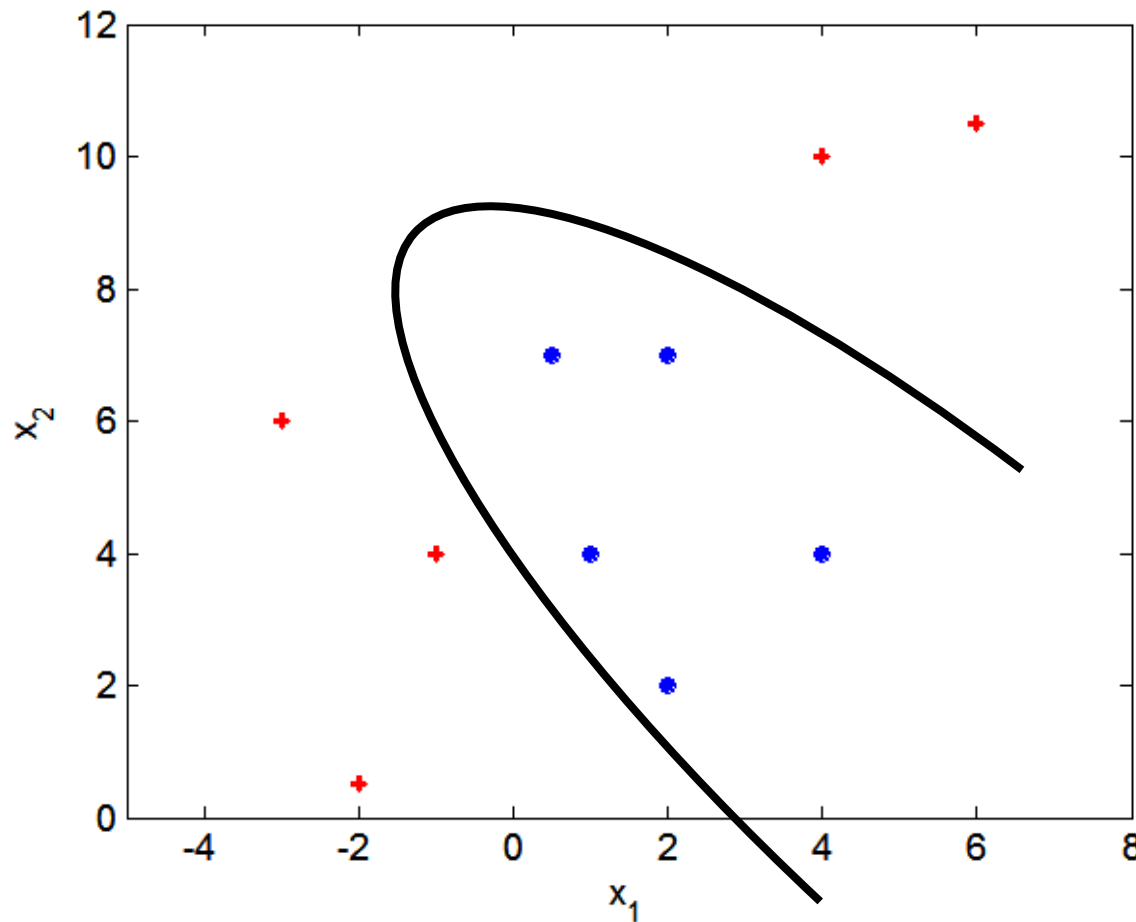- What if data instances of the two class are not separable?
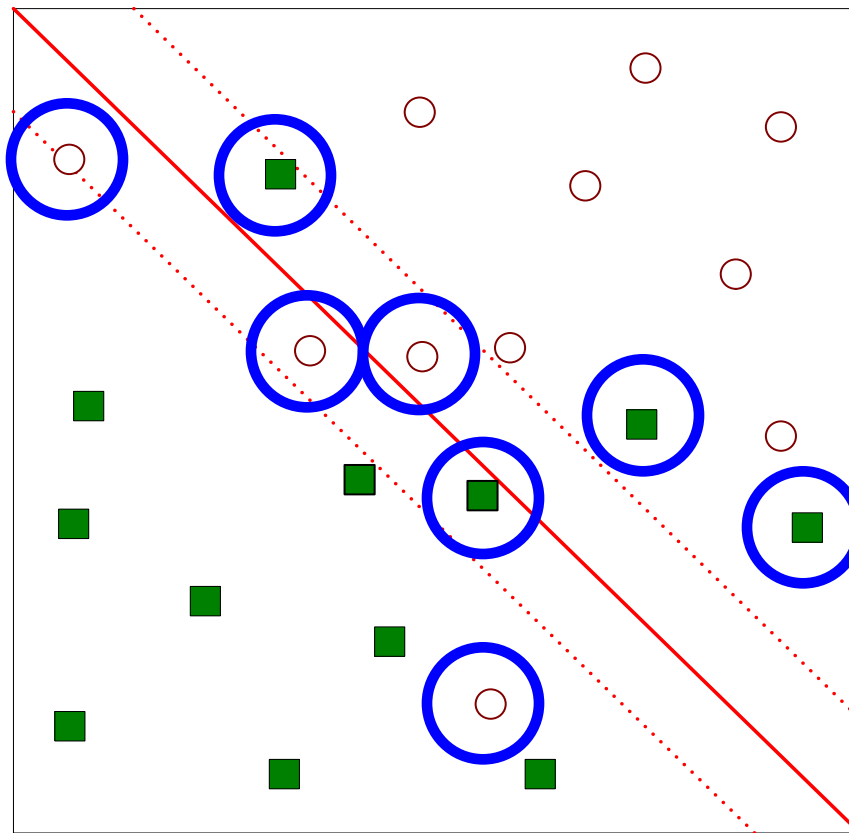
Slack variables $\xi_i \geq 0$ need to be introduced to absorb errors

Detailed induction will be presented next week

Nonlinear SVMs

- What if the decision boundary is not linear?

Kernel methods

1) From Linear Scoring to a Probabilistic Binary Model

We consider binary classification:

$$y \in \{0, 1\}, \quad x \in \mathbb{R}^d.$$

We start with a linear score (also called a *logit*):

$$z = w^\top x + b.$$

Since $z \in \mathbb{R}$ is not a probability, we need a function that maps real numbers to $(0, 1)$.
The standard choice is the **sigmoid (logistic) function**:

$$\sigma(z) = \frac{1}{1 + e^{-z}}.$$

Define:

$$p(y = 1 \mid x) = \sigma(w^\top x + b), \qquad p(y = 0 \mid x) = 1 - \sigma(w^\top x + b).$$

2) The Core Assumption: Linear Log-Odds

Define the **odds**:

$$\text{odds}(x) = \frac{p(y = 1 \mid x)}{p(y = 0 \mid x)}.$$

If $p = \sigma(z)$, then:

$$\frac{p}{1 - p} = \frac{\frac{1}{1+e^{-z}}}{1 - \frac{1}{1+e^{-z}}} = e^z.$$

Taking logarithms yields the **log-odds (logit)**:

$$\log \frac{p}{1 - p} = z = w^\top x + b.$$

3) Likelihood and Cross-Entropy Loss

Given a dataset $\{(x_i, y_i)\}_{i=1}^{n}$, assume:

$$p_i := p(y_i = 1 \mid x_i) = \sigma(z_i), \quad z_i = w^\top x_i + b.$$

Each y_i follows a Bernoulli distribution:

$$p(y_i \mid x_i) = p_i^{y_i}(1 - p_i)^{1 - y_i}.$$

Assuming independence, the likelihood is:

$$L(w, b) = \prod_{i=1}^{n} p_i^{y_i}(1 - p_i)^{1 - y_i}.$$

Taking logarithms:

$$\ell(w, b) = \sum_{i=1}^{n} \left(y_i \log p_i + (1 - y_i) \log(1 - p_i) \right).$$

Minimizing the **negative log-likelihood** gives:

$$J(w, b) = -\ell(w, b) = -\sum_{i=1}^{n} \left(y_i \log p_i + (1 - y_i) \log(1 - p_i) \right),$$

which is exactly the **binary cross-entropy loss**.

4) Logistic Loss Form ($y \in \{-1, +1\}$)

Let $\tilde{y} = 2y - 1 \in \{-1, +1\}$. Then:

$$p(\tilde{y} \mid x) = \sigma(\tilde{y}z).$$

The loss becomes the classic **logistic loss**:

$$\mathcal{J}(w, b) = \sum_{i=1}^{n} \log\left(1 + \exp(-\tilde{y}_i z_i)\right).$$

This form is often preferred for theoretical analysis and convexity proofs.

5) Gradient Derivation and Its Simplicity

For a single sample:

$$\mathcal{J}_i = -(y_i \log p_i + (1 - y_i) \log(1 - p_i)), \quad p_i = \sigma(z_i).$$

Using $\sigma'(z) = \sigma(z)(1 - \sigma(z))$, we compute:

$$\frac{\partial \mathcal{J}_i}{\partial z_i} = \frac{\partial \mathcal{J}_i}{\partial p_i} \cdot \frac{\partial p_i}{\partial z_i}.$$

After simplification:

$$\frac{\partial \mathcal{J}_i}{\partial z_i} = p_i - y_i.$$

Thus the gradients are:

$$\nabla_w \mathcal{J} = \sum_{i=1}^{n} (p_i - y_i) x_i, \qquad \frac{\partial \mathcal{J}}{\partial b} = \sum_{i=1}^{n} (p_i - y_i).$$

> Interpretation:
> $(p_i - y_i)$ is the **probability error**, scaled by the feature vector.

6) Hessian and Convexity

Let $X \in \mathbb{R}^{n \times d}$ be the design matrix. The Hessian is:

$$\nabla_w^2 \mathcal{J} = X^\top W X, \quad W = \mathrm{diag}(p_i(1 - p_i)) \succeq 0.$$

Hence the loss is **convex**, implying:

- no local minima,
- stable optimization,
- effective Newton / quasi-Newton (L–BFGS) methods,
- scalability with SGD-type methods.

7) Regularization and MAP Interpretation

With L2 regularization:

$$\min_{w,b} \ \mathcal{J}(w, b) + \frac{\lambda}{2} \|w\|_2^2.$$

This corresponds to a Gaussian prior:

$$w \sim \mathcal{N}(0, \lambda^{-1}I),$$

and the optimization becomes **maximum a posteriori (MAP)** estimation.

L1 regularization corresponds to a Laplace prior and induces sparsity.

8) Decision Boundary

The usual decision rule is $p(y = 1 \mid x) \geq 0.5$.

Since $p \geq 0.5 \iff z \geq 0$, the boundary is:

$$w^\top x + b = 0,$$

a linear hyperplane.

May the Year of the Horse bring you the strength to reach your academic goals!

HAPPY
Lunar New Year
2026

Thank you!