Regression and Classification

by Prof. Seungchul Lee
Industrial AI Lab
http://isystems.unist.ac.kr/
POSTECH

Table of Contents

1. Linear Regression

- $\hat{y}_i = f(x_i, \theta)$ in general
- In many cases, a linear model to predict y_i is assumed

Given $\begin{cases} x_i : \text{inputs} \\ y_i : \text{outputs} \end{cases}$, Find θ_1 and θ_2

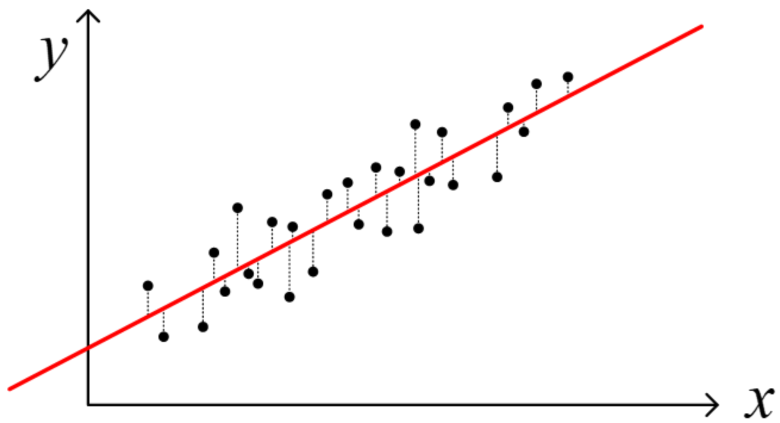
$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \approx \hat{y}_i = \theta_1 x_i + \theta_2$$

- $\hat{y}_i$: predicted output
- $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$: Model parameters

$$\hat{y}_i = f(x_i, \theta) \ \text{ in general}$$

- in many cases, a linear model to predict y_i used

$$\hat{y}_i = \theta_1 x_i + \theta_2 \ \text{ such that } \min_{\theta_1, \theta_2} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$



Linear Regression as Optimization</fotn>

- How to find model parameters, $\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix} \approx \hat{y}_i = \theta_1 x_i + \theta_2$$

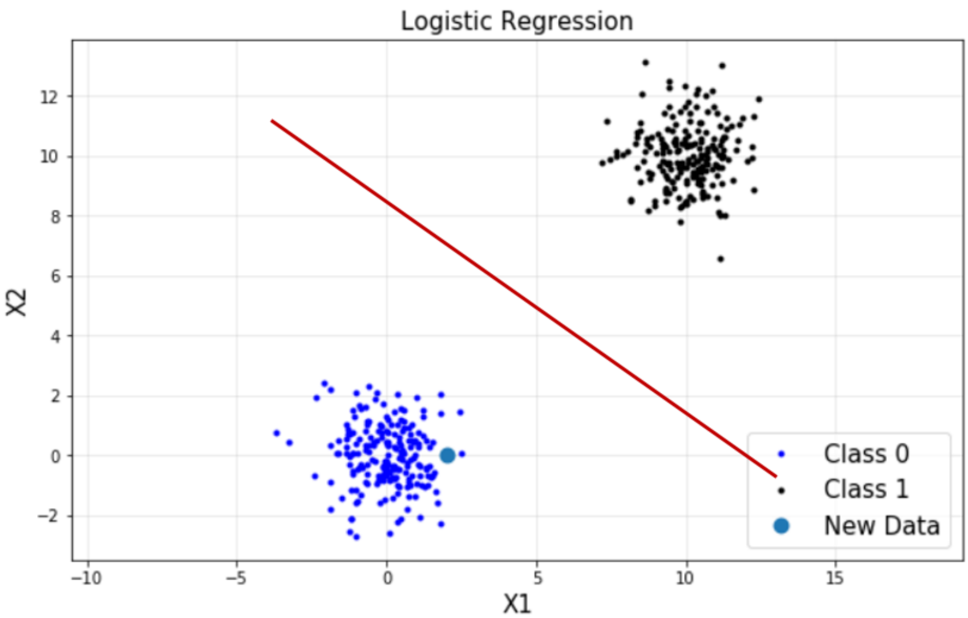
$$\hat{y}_i = \theta_1 x_i + \theta_2 \quad \text{such that } \min_{\theta_1, \theta_2} \sum_{\theta_1, \theta_2}^{m} (\hat{y}_i - y_i)^2$$

2. Classification

- Perceptron: make use of sign of data
  - discuss it later

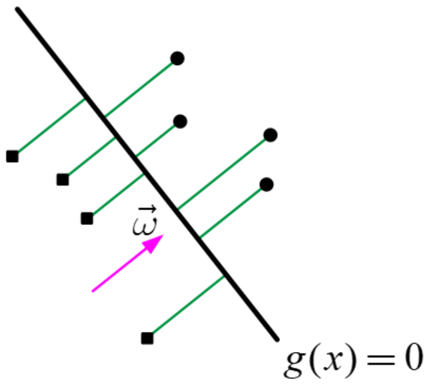
- Logistic regression is a classification algorithm
  - don't be confused

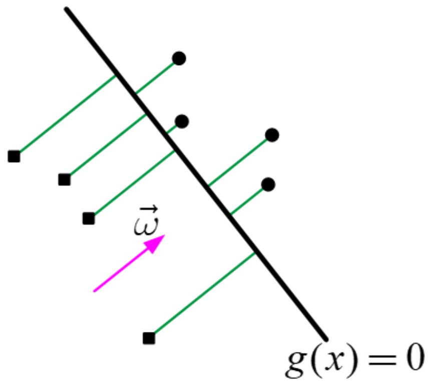
- To find a classification boundary



2.1. Using Distances

- basic idea: to find the decision boundary (hyperplane) of $g(x) = \omega^T x = 0$ such that maximizes $\prod_i |h_i| \rightarrow$ optimization



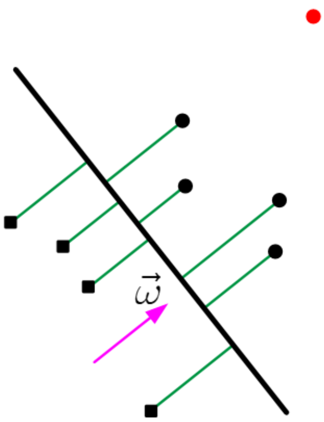


- Inequality of arithmetic and geometric means
$$\frac{x_1 + x_2 + \cdots + x_m}{m} \geq \sqrt[m]{x_1 \cdot x_2 \ldots x_m}$$
and that equality holds if and only if $x_1 = x_2 = \cdots = x_m$

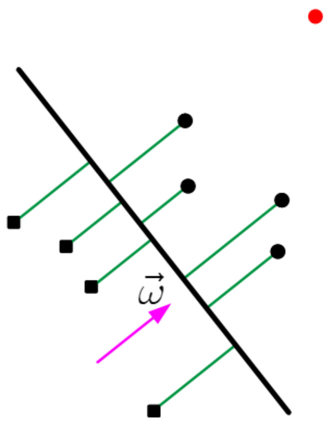
- Roughly speaking, this optimization of $\max \prod_i |h_i|$ tends to position a hyperplane in the middle of two classes
$$h = \frac{g(x)}{\|\omega\|} = \frac{\omega^T x}{\|\omega\|} \sim \omega^T x$$

2.2. Using all Distances with Outliers

- SVM vs. Logistic Regression



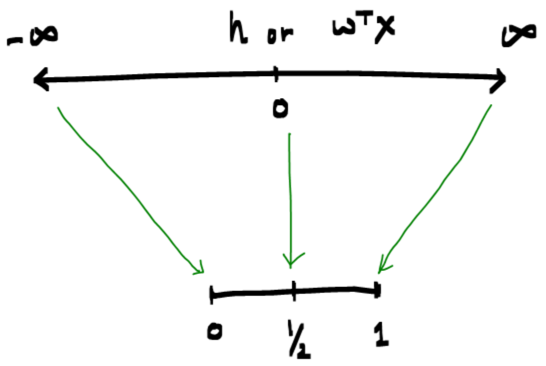
SVM

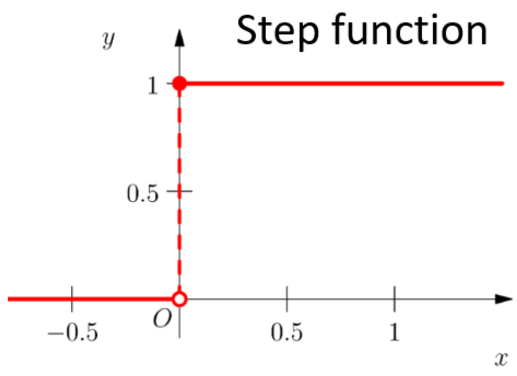


Logistic Regression

2.3. Sigmoid Function

- We link or squeeze $(-\infty, +\infty)$ to $(0, 1)$ for several reasons:

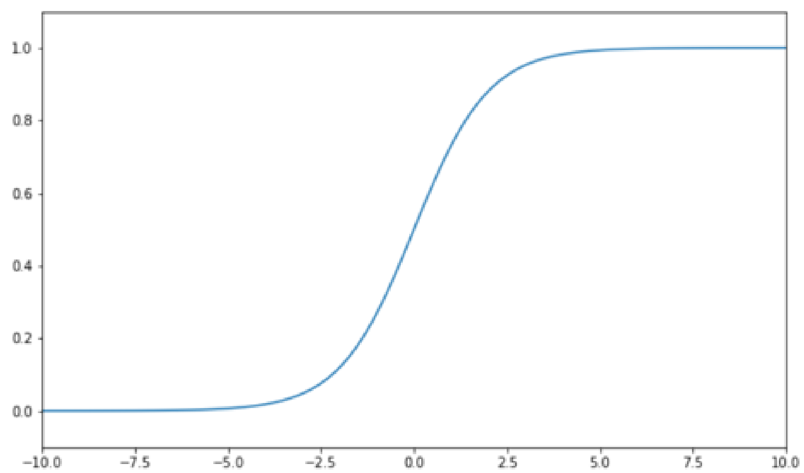




- If $\sigma(z)$ is the sigmoid function, or the logistic function

$$\sigma(z) = \frac{1}{1+e^{-z}} \implies \sigma(\omega^T x) = \frac{1}{1+e^{-\omega^T x}}$$

  - logistic function always generates a value between 0 and 1
  - Crosses 0.5 at the origin, then flattens out



- Benefit of mapping via the logistic function
  - monotonic: same or similar optimziation solution
  - continuous and differentiable: good for gradient descent optimization
  - probability or confidence: can be considered as probability

$$P(y = +1 \mid x, \omega) = \frac{1}{1+e^{-\omega^T x}} \quad \in \ [0, 1]$$

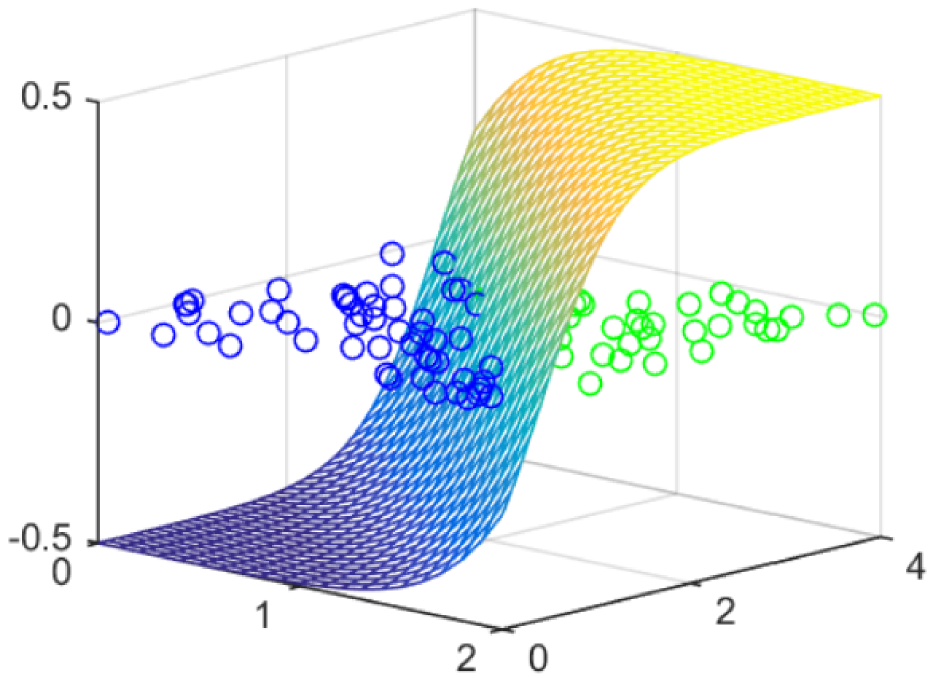
- Goal: we need to fit ω to our data

$$\max \prod_i |h_i|$$

- Again, it is an optimization problem

3. Logistic Regression

- Classified based on probability



3.1. Multiclass Classification: Softmax

- Generalization to more than 2 classes is straightforward
  - one vs. all (one vs. rest)
  - one vs. one

- Using the soft-max function instead of the logistic function (refer to UFLDL Tutorial (http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression/))
  - see them as probability

$$P\left(y=k \mid x,\omega\right) = \frac{\exp\left(\omega_k^T x\right)}{\sum_k \exp\left(\omega_k^T x\right)} \in [0,1]$$

- We maintain a separator weight vector ω_k for each class k

- Note: sigmoid function

$$P(y=+1 \mid x,\omega) = \frac{1}{1+e^{-\omega^T x}} \in [0,1]$$

4. Summary

- From parameter estimation of machine learning to optimization problems

Machine learning	**Optimization**
	Loss (or objective functions)
Regression	$$\min_{\theta_1,\theta_2} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$
Classification	$$\begin{aligned}\ell(\omega) = \log \mathscr{L} = \log P(y \mid x, \omega) &= \log \prod_{n=1}^{m} P(y_n \mid x_n, \omega) \\ &= \sum_{n=1}^{m} \log P(y_n \mid x_n, \omega)\end{aligned}$$

In [1]:

```
%%javascript
$.getScript('https://kmahelona.github.io/ipython_notebook_goodies/ipython_notebook_toc.js')
```