Artificial Intelligence

Lecture 11. Logistic Regression Part 2

Spring 2022

Prof. Jonghoon Chun, Ph.D.

E-mail: jchun@mju.ac.kr
Lecture Note: http://lms.mju.ac.kr

Agenda

- Classification concept review
- Linear Regression
- Logistic Regression

SIMPLIFIED COST FUNCTION

Logistic Regression Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

$$Cost(h_\theta(x), y) = -y\, log(h_\theta(x)) - (1 - y)log(1 - h_\theta(x))$$

If y = 1: $Cost(h_\theta(x), y) = -\,log(h_\theta(x))$

If y = 0: $Cost(h_\theta(x), y) = -\,log(1 - h_\theta(x))$

Logistic Regression Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

- Derived from statistics using the principle of maximum likelihood estimation.
  - An idea in statistics for how to efficiently find parameters' data for different models
- It is “convex”!
- Used mostly when fitting logistic regression models

Fitting Parameters

To fit parameters θ :

$$\min_{\theta} J(\theta)$$

To make a prediction given new x:

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ $\quad P(y = 1 \mid x ; \theta)$

Gradient Descent

$$J(\theta) = -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_n \end{bmatrix}$$

$$\frac{d}{d\theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

$$h_\theta(x) = \theta^T x$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Algorithm looks identical to linear regression!

Except for $h_\theta(x)$!

Implementation

- For loop vs. vectorized implementation
  - Can update all m+1 parameters at once

- Feature Scaling
  - Use it when features are on very different scale
  - can help gradient descent converge faster for linear regression as well as logistic regression

- Logistic regression is a very powerful, and probably the most widely used classification algorithm in the world!

MULTI-CLASS CLASSIFICATION

Multiclass Classification

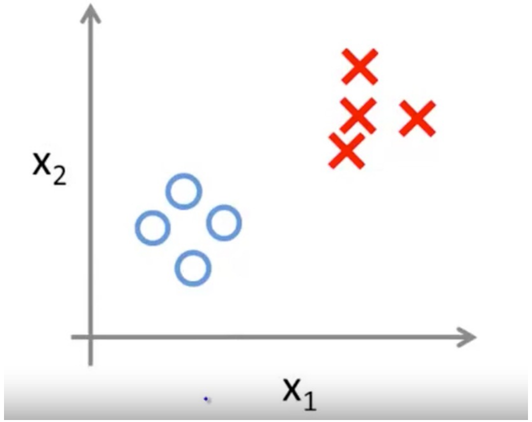
- Email Foldering/Tagging
  – Works, Friends, Family, Hobby

- Medical Diagnosis
  – Not ill, Cold, Flu

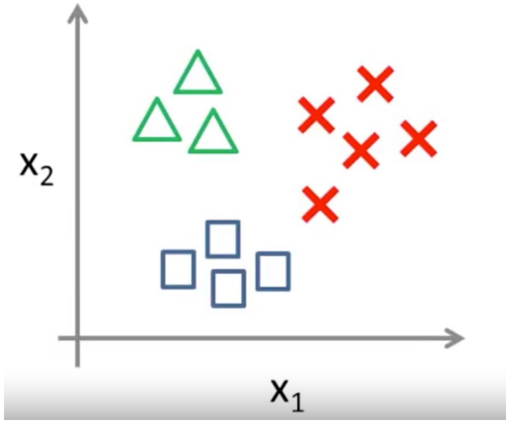
- Weather
  – Sunny, Cloudy, Rain, Snow

Binary vs. Multi-class Classification

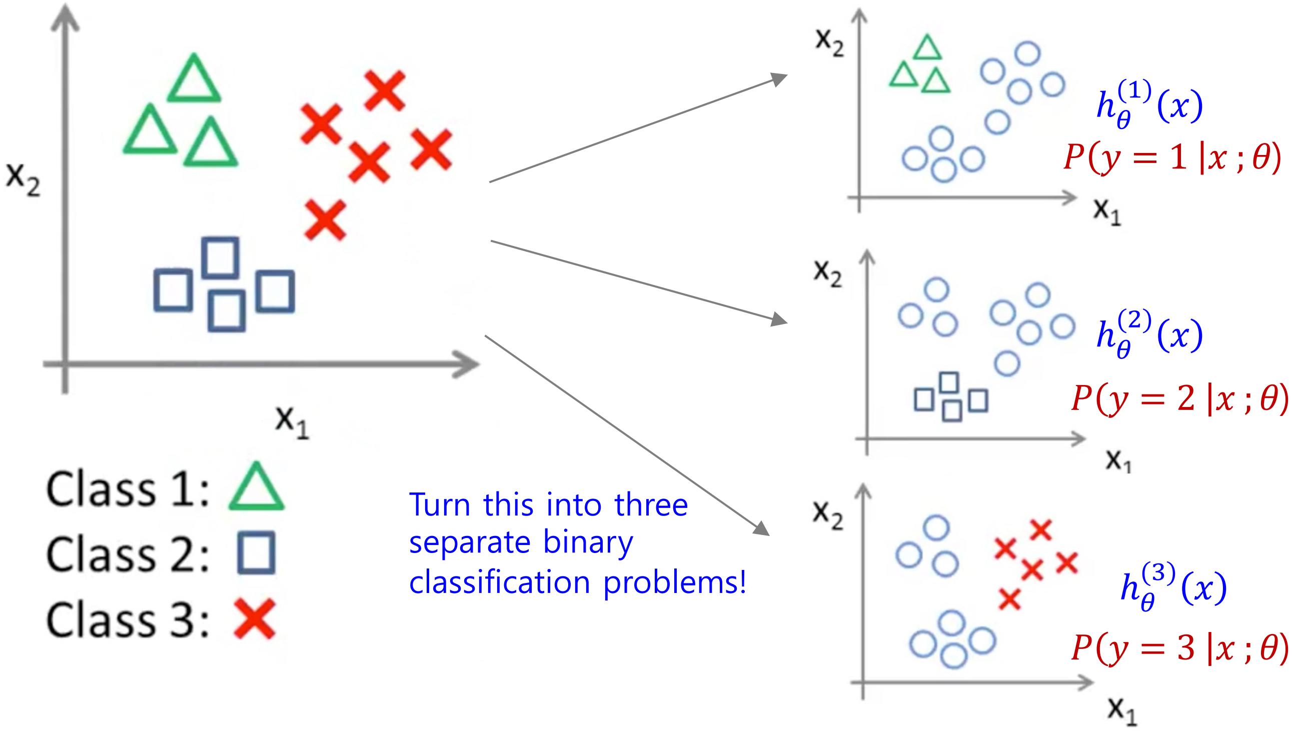
Binary Classification:



Multi-class Classification:



One-vs-all (one-vs-rest)



$$h_\theta^{(i)}(x) = P(y=i\,|x\,;\theta) \quad (i=1,2,3)$$

One-vs-all Summary

- Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

- On a new input x, to make a prediction, pick the class i that maximizes

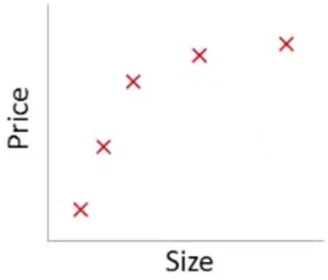
$$\max_i h_\theta^{(i)}(x)$$

OVERFITTING PROBLEM

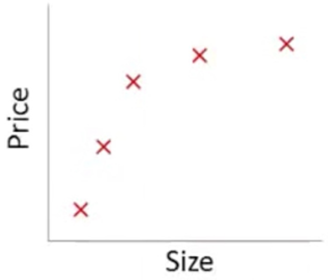
Regularization: Reduce the overfitting problem

Overfitting Example

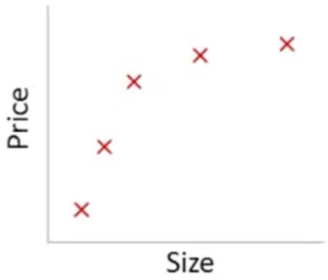
- Linear Regression on housing prices



$$\theta_0 + \theta_1 x$$



$$\theta_0 + \theta_1 x + \theta_2 x^2$$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

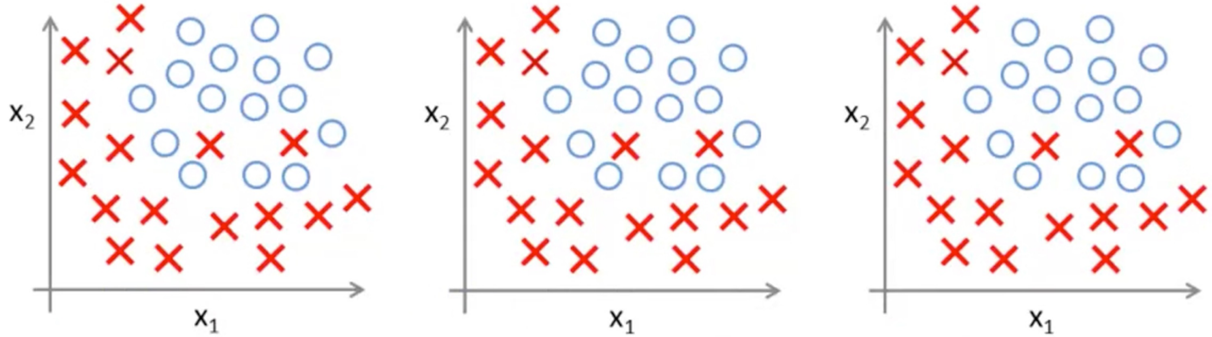
"underfit" "High bias"

"overfit" "High variance"

- **Overfitting:** If we have too many features, the learned hypothesis may fit the training set very well, but fail to generalize to new examples (e.g., predict prices on new examples).

Overfitting Example

- Logistic Regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

(g = sigmoid function)

"underfit"

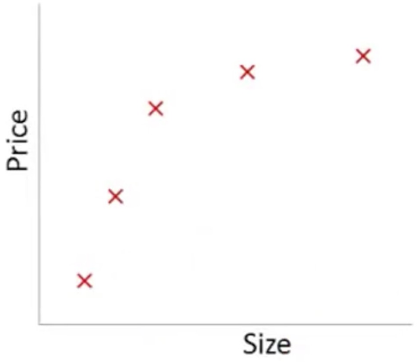
$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 \\ + \theta_3 x_1^2 + \theta_4 x_2^2 \\ + \theta_5 x_1 x_2)$$

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 \\ + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 \\ + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

"overfit"

Addressing Overfitting

$x_1 \ = size\ of\ house$
$x_2 \ = no\ of\ bedrooms$
$x_3 \ = no\ of\ floors$
$x_4 \ = age\ of\ house$
$x_5 \ = average\ income\ in\ neighborhood$
$x_6 \ = kitchen\ size$
....
x_{100}



With so many features,

- becomes much harder to plot the data
- much harder to visualize it
- much harder to decide what features to keep or not

Plotting the hypothesis, could be one way to try to decide what degree polynomial to use.

Lot of features, and, very little training data, then, overfitting can become a problem!

Addressing Overfitting

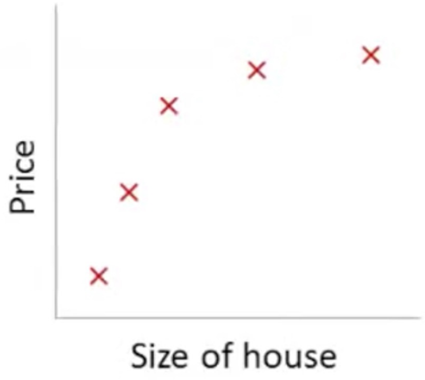
- Reduce number of features
  - Manually select which features to keep
  - Model selection algorithm: automatically decide which features to keep
  - throwing away some of the features, is also throwing away some of the information you have about the problem (i.e., maybe, all of those features are actually useful for predicting the price of a house)

- Regularization
  - Keep all the features, but reduce magnitude/values of parameters θ_j
  - Works well when we have a lot of features, each of which contributes a bit to predicting y

REGULARIZATION

An Intuitive Example

$$\theta_0 + \theta_1 x + \theta_2 x^2$$



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Suppose we penalize and make θ_3 , θ_4 very small:

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \quad + 1000\theta_3^2 + 1000\ \theta_4^2$$

$$\theta_3 \approx 0 \qquad \theta_4 \approx 0$$

Regularization

- Small values for parameters θ_0, θ_2, ..., θ_n
  - "Simpler" hypothesis
  - Less prone to overfitting

- Housing example
  - Features: x_1, x_2, ..., x_{100}
  - Parameters: θ_0, θ_1, ..., θ_{100}

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

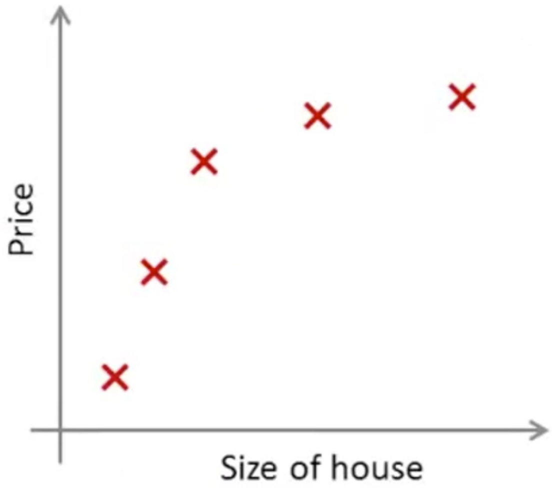
Sometimes called "L2" Regularization!

Regularization

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

Regularization parameter

$$\min_\theta J(\theta)$$

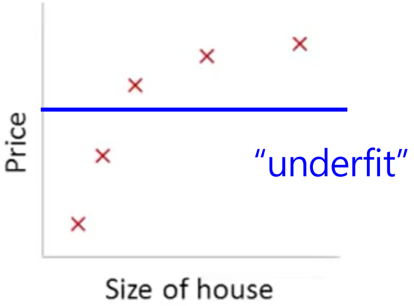


Regularization Parameter

In regularized linear regression, we choose θ to minimize

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?



$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

$\theta_1, \theta_2, \theta_3, \theta_4$

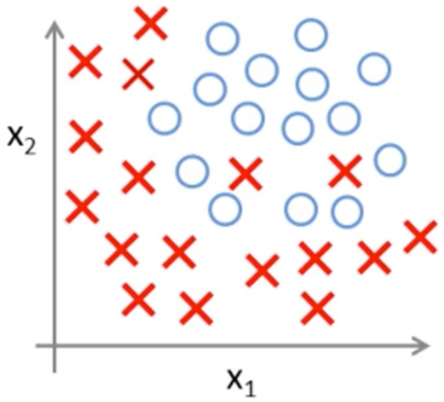
$\theta_1 \approx 0, \theta_2 \approx 0$

$\theta_3 \approx 0 , \theta_4 \approx 0$

$h_\theta(x) = \theta_0$

REGULARIZED LOGISTIC REGRESSION

Regularized Logistic Regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost Function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$
$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

Regularized Logistic Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)}\log h_\theta(x^{(i)}) + (1 - y^{(i)})\log(1 - h_\theta(x^{(i)}))\right]$$

$$+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

Gradient Descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m}\theta_j \right]$$

$$(j = 0, 1, 2, 3, \ldots, n)$$

}

$$\frac{\partial}{\partial \theta_j} J(\theta)$$

Regularized!

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$1 - \alpha \frac{\lambda}{m} < 1 \ (e.g., 0.99)$$

ACKNOWLEDGMENTS

- Many of the slides and examples are borrowed from the open course by Coursera "**Machine Learning**" by A. Ng which were modified into their current form.

END