Machine Learning Study Week 3

Logistic Regression
&
Regularization

구 본 규 (protocolstack9@gmail.com)

Machine Learning

Logistic Regression

Classification

Classification

- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

$y \in \{0, 1\}$

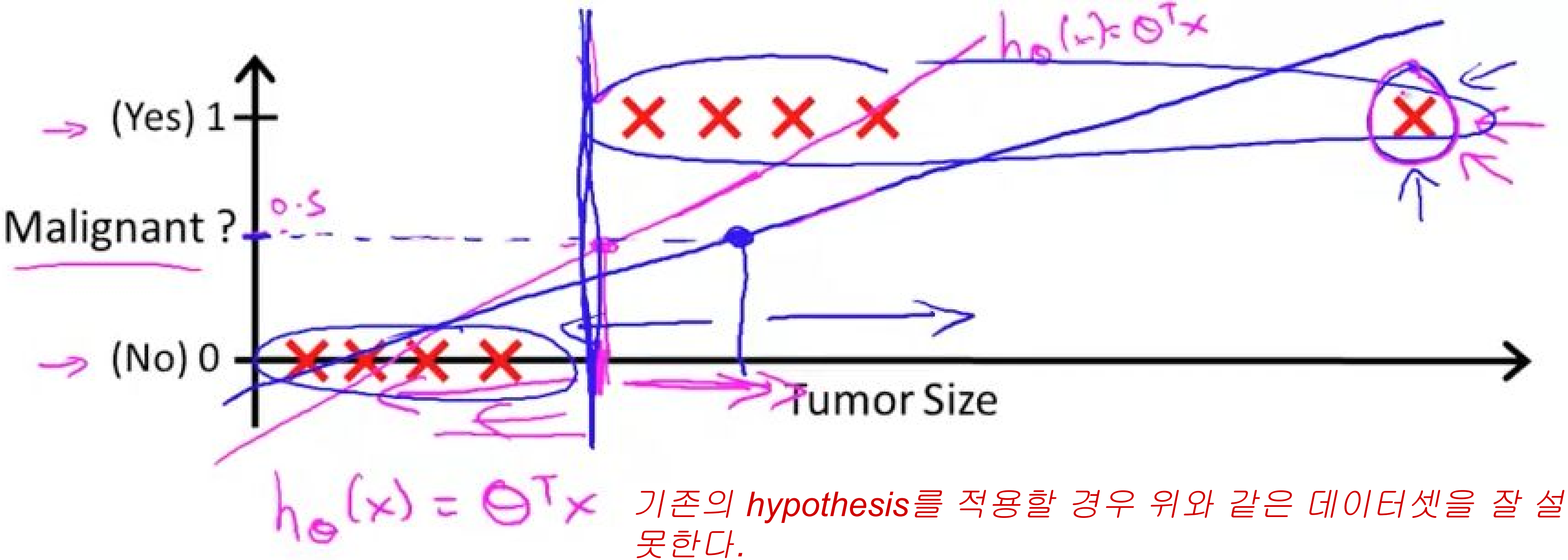
0: "Negative Class" (e.g., benign tumor)

1: "Positive Class" (e.g., malignant tumor)

$y \in \{0, 1, 2, 3\}$

Regression: the output variable takes continuous values.

Classification: the output variable takes class labels.



*기존의 **hypothesis**를 적용할 경우 위와 같은 데이터셋을 잘 설명하지 못한다.*
***hypothesis**의 결과에서 **0.5**를 기준으로 잡아 클래스를 분류한다.*

→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

Classification: y = 0 or 1

$h_\theta(x)$ can be > 1 or < 0

Logistic Regression: $0 \leq h_\theta(x) \leq 1$

Classification

linear regression은 결과값이 1 초과, 0 미만일 수도 있다.
logistic regression은 hypothesis의 예측범위를 0과 1 사이로 제한한다.

Machine Learning

Logistic Regression

Hypothesis Representation

Logistic Regression Model

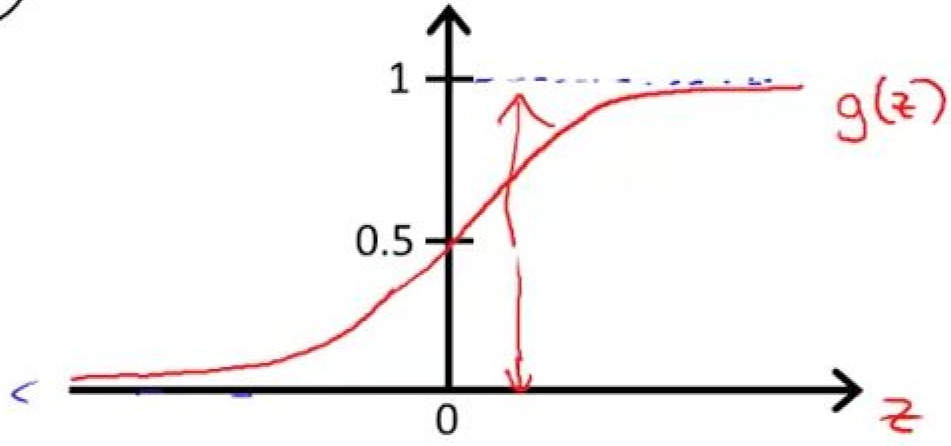
Want $\boxed{0 \le h_\theta(x) \le 1}$

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

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

$\theta^T x$

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



Sigmoid function

Logistic function

Parameters θ.

hypothesis의 범위를 제한하기 위해 sigmoid function을 적용한다.

Interpretation of Hypothesis Output

$h_\theta(x)$

$h_\theta(x)$ = estimated probability that $\boxed{y = 1}$ on input x

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$h_\theta(x) = 0.7$

$y=1$

Tell patient that 70% chance of tumor being malignant

$h_\theta(x) = P(y=1|x;\theta)$

"probability that y = 1, given x, parameterized by θ"

$y = 0$ or 1

$\rightarrow P(y=0|x;\theta) + P(y=1|x;\theta) = 1$

$P(y=0|x;\theta) = 1 - P(y=1|x;\theta)$

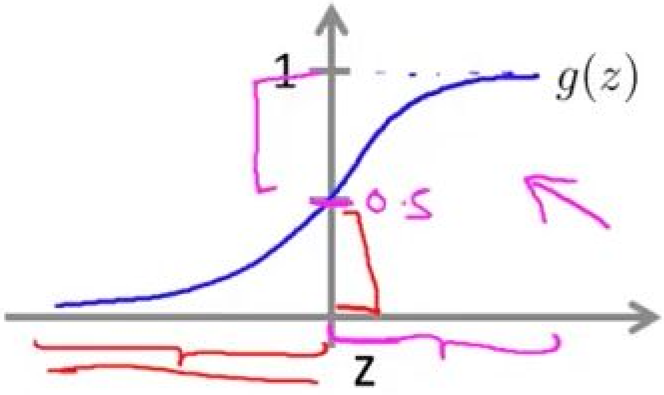
파라미터 theta를 사용해
입력 x에 대해 y가 1일 확률을 추정하는
hypothesis.

Machine Learning

Logistic Regression

Decision boundary

Logistic regression



$\rightarrow h_\theta(x) = g(\theta^T x) = P(y=1 \mid x;\theta)$

$\rightarrow g(z) = \frac{1}{1+e^{-z}}$

Suppose predict $\boxed{\text{“}y = 1\text{“ if } h_\theta(x) \geq 0.5}$

$\rightarrow \theta^T x \geq 0$

$g(z) \geq 0.5$

when $z \geq 0$

$h_\theta(x) = g(\theta^T x) \geq 0.5$

whenever $\theta^T x \geq 0$

z

predict $\boxed{\text{“}y = 0\text{“ if } h_\theta(x) < 0.5}$

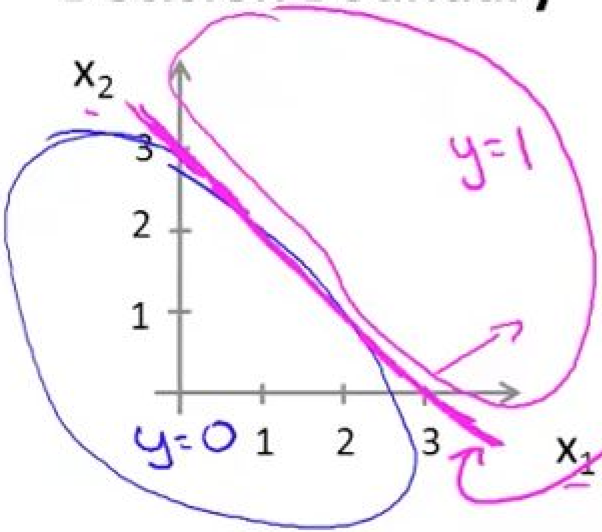
$h_\theta(x) = g(\theta^T x)$

$\rightarrow \theta^T x < 0$

$g(z) < 0.5$

hypothesis 결과가 0.5 이상이면 y=1로 예상하고,
0.5 미만이면 y=0으로 예상한다.

Decision Boundary



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$\boxed{h_\theta(x) = g(\underset{-3}{\theta_0} + \underset{1}{\theta_1} x_1 + \underset{1}{\theta_2} x_2)}$$

Decision boundary

x_1, x_2

$h_\theta(x) = 0.5$

$\boxed{x_1 + x_2 = 3}$

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2 \geq 0}_{\theta^T x}$

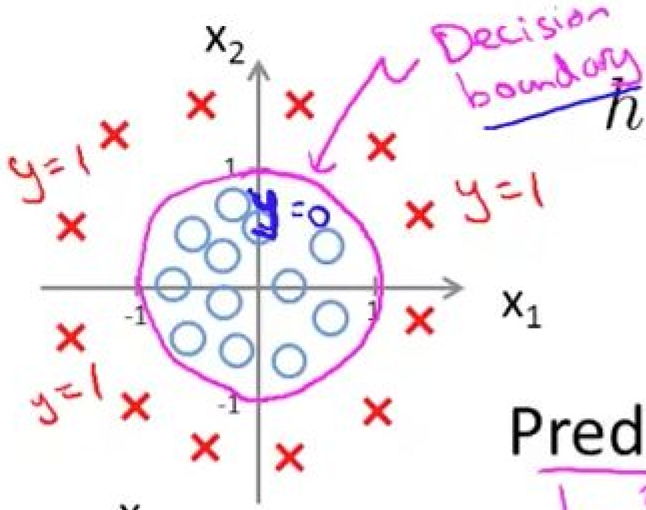
$x_1 + x_2 \geq 3$

$x_1 + x_2 < 3$

$y = 0$

logistic regression에서 hypothesis의 결과가 1이 되는 경계가 존재한다.

Non-linear decision boundaries

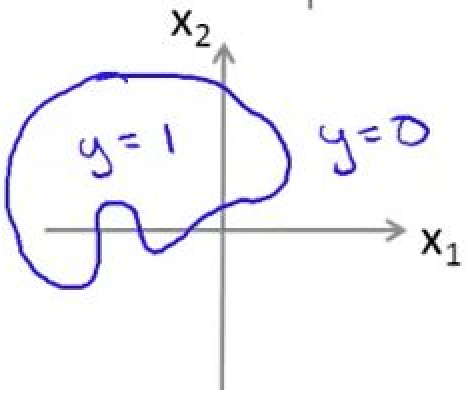


$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2 \geq 0}_{x_1^2 + x_2^2 \geq 1}$

$\boxed{x_1^2 + x_2^2 = 1}$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 \underline{x_1^2} + \theta_4 \underline{x_1^2 x_2} + \theta_5 \underline{x_1^2 x_2^2} + \theta_6 \underline{x_1^3 x_2} + \dots)$$

Machine Learning

Logistic Regression

Cost function

Training set: $\underline{\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}}$

m examples

$$x \in \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}}_{} \mathbb{R}^{n+1} \qquad x_0 = 1, y \in \{0,1\}$$

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

How to choose parameters θ ?

Cost function

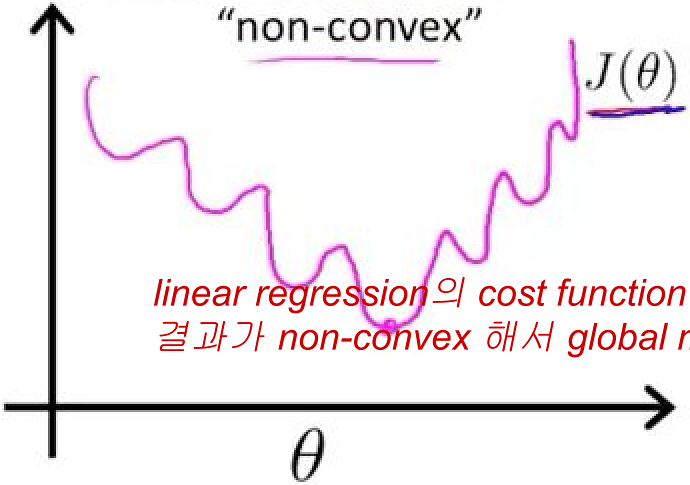
→ ~~Linear~~ regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

logistic

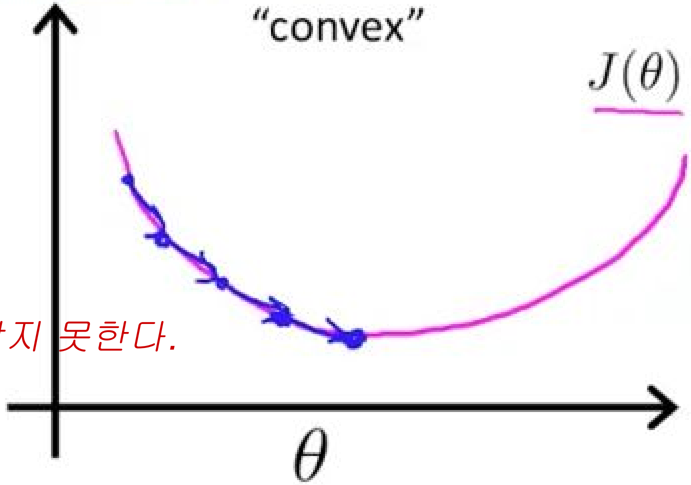
→ $\mathrm{cost}(h_\theta(x^{(i)}), y)$

→ $\mathrm{Cost}(h_\theta(x), y) = \boxed{\frac{1}{2}\left(h_\theta(x) - y\right)^2}$ ← $\boxed{\frac{1}{1+e^{-\theta^T x}}}$

"non-convex"



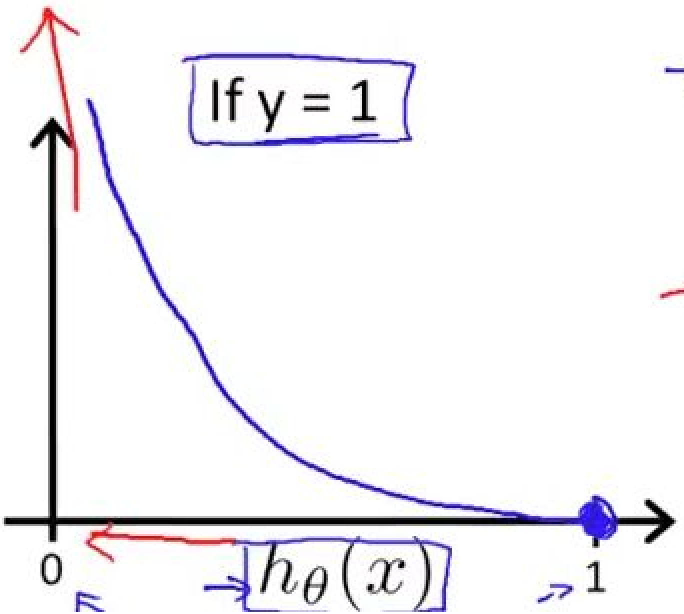
"convex"



linear regression의 cost function을 사용하면
결과가 non-convex 해서 global minimum을 찾지 못한다.

Logistic regression cost function

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



- Cost $= 0$ if $y = 1, h_\theta(x) = 1$

  But as $h_\theta(x) \to 0$

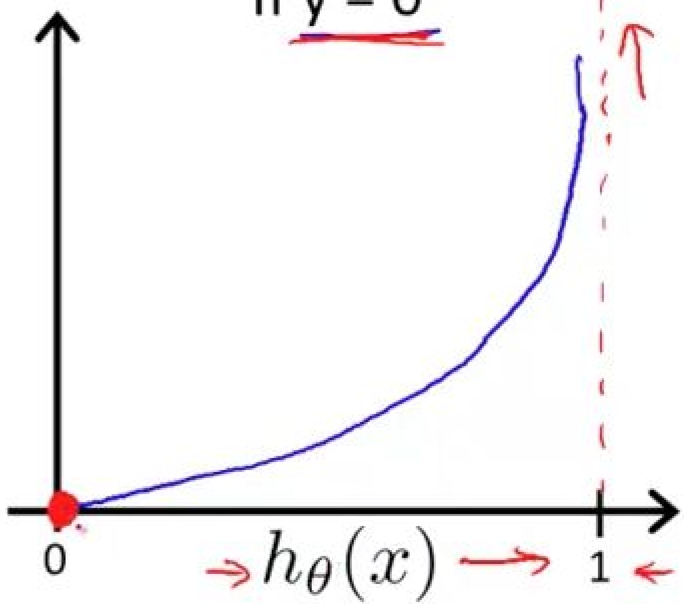
  $Cost \to \infty$

- Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x;\theta) = 0$), but $\boxed{y = 1}$, we'll penalize learning algorithm by a very large cost.

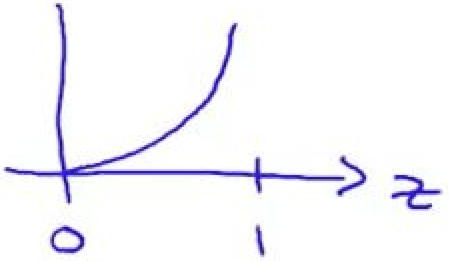
Logistic regression cost function

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

If y = 0



$-\log(1-z)$



Machine Learning

Logistic Regression

Simplified cost function and gradient descent

Logistic regression cost function

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

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

Note: $y = 0$ or 1 always

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

Logistic regression cost function

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

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

To fit parameters θ:

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

To make a prediction given new $\underline{x}$:

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

Gradient Descent

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

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

Repeat $\{$

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

$\}$ (simultaneously update all θ_j)

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

J_of_theta를 최저로 만드는 theta를 찾기 위해 gradient를 계산한다.

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

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \begin{matrix} \leftarrow \\ \leftarrow \\ \leftarrow \\ \\ \leftarrow \end{matrix}$$

for i = 0 to n

Repeat {

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

(simultaneously update all θ_j)

}

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

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

linear regression과 외형은 동일하지만, hypothesis가 다르다.

Algorithm looks identical to linear regression!

Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial\theta_j} J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

cost function을 최저로 만드는 theta를 찾는
여러 가지 알고리즘이 존재한다.

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

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

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \qquad \theta_1 = 5,\ \theta_2 = 5.$$

$$\boxed{J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2}$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```
function [jVal, gradient]
            = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
     = fminunc(@costFunction, initialTheta, options);
```

cost function을 정의하고, 초기값과 함께 전달하면 라이브러리 함수로 optimized theta를 찾아준다.

$$\texttt{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \begin{array}{l} \text{theta(1)} \\ \text{theta(2)} \\ \\ \text{theta(n+1)} \end{array}$$

```
function [jVal, gradient] = costFunction(theta)
```

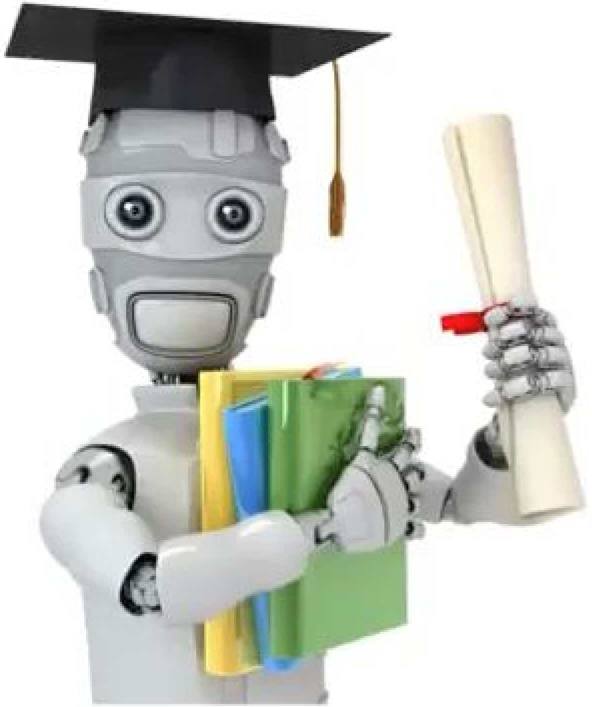
`jVal = [`code to compute $J(\theta)$`];`

`gradient(1) = [`code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$`];`

`gradient(2) = [`code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$`];`

$\vdots$

`gradient(n+1) = [`code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$`];`



Machine Learning

Logistic Regression

Multi-class classification: One-vs-all

Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

$y=1$ $y=2$ $y=3$ $y=4$

Medical diagrams: Not ill, Cold, Flu

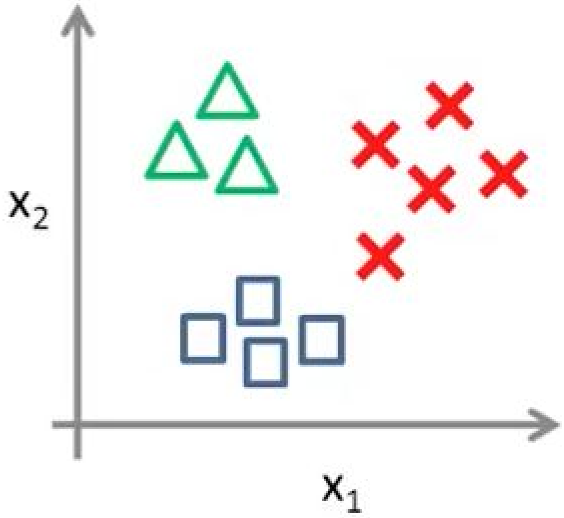
$y=1$ 2 3

Weather: Sunny, Cloudy, Rain, Snow

$y=1$ 2 3 4 $\Leftarrow$

0 1 2 3

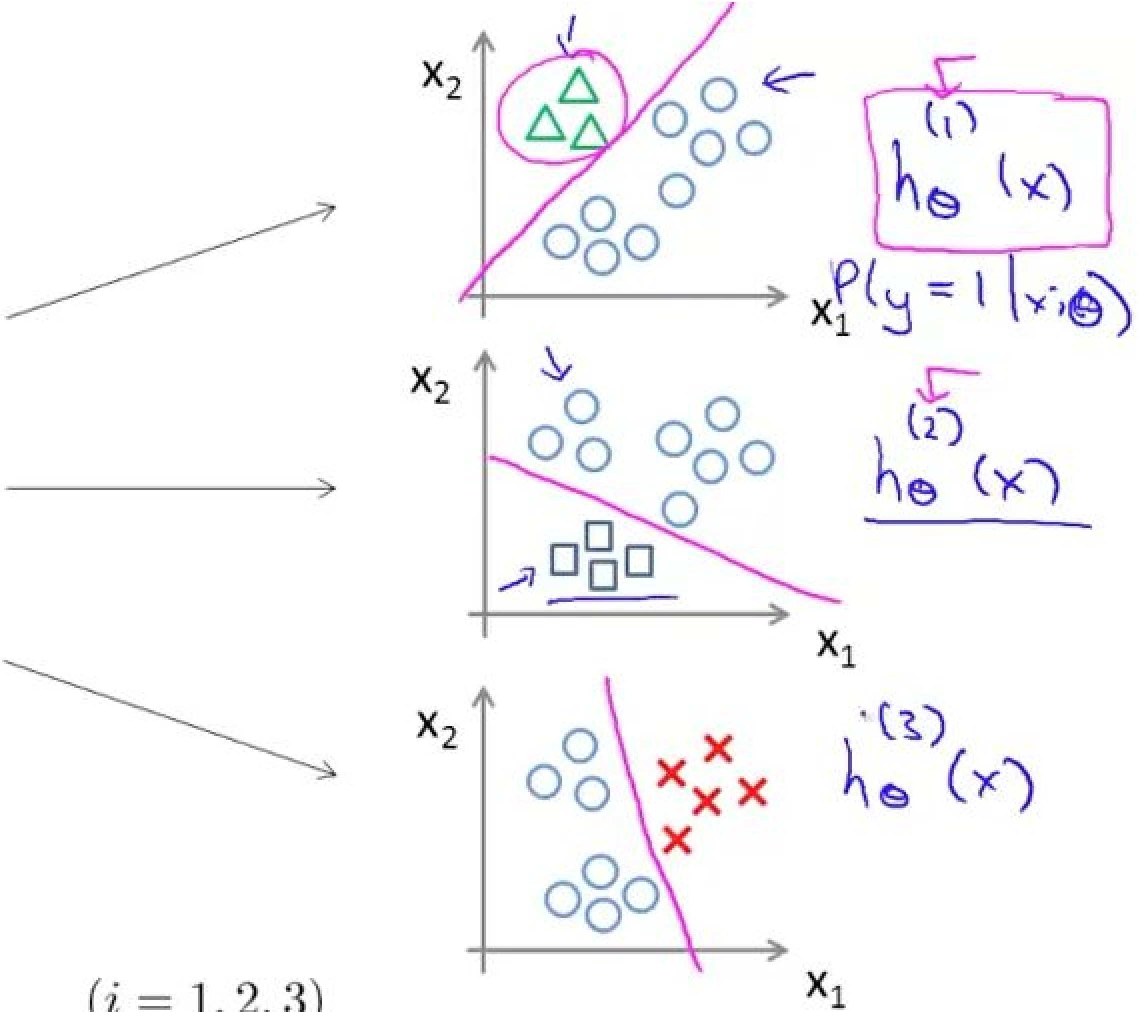
One-vs-all (one-vs-rest):



Class 1: △

Class 2: □

Class 3: ✕



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

One-vs-all

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

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

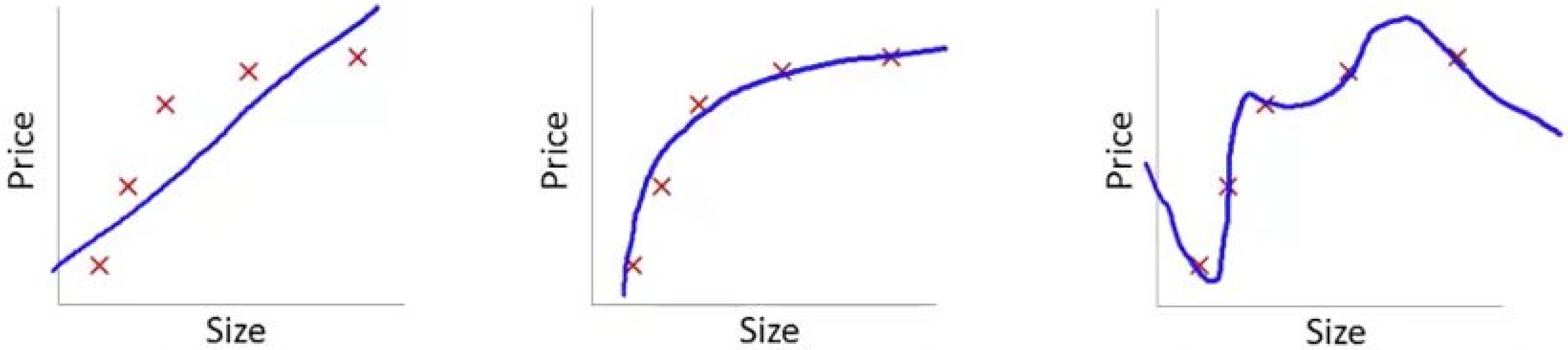
각 class에 대해서 one-vs-all로 확률을 구하고,
그 중 가장 높은 hypothesis를 선택한다.

Machine Learning

Regularization

The problem of overfitting

Example: Linear regression (housing prices)



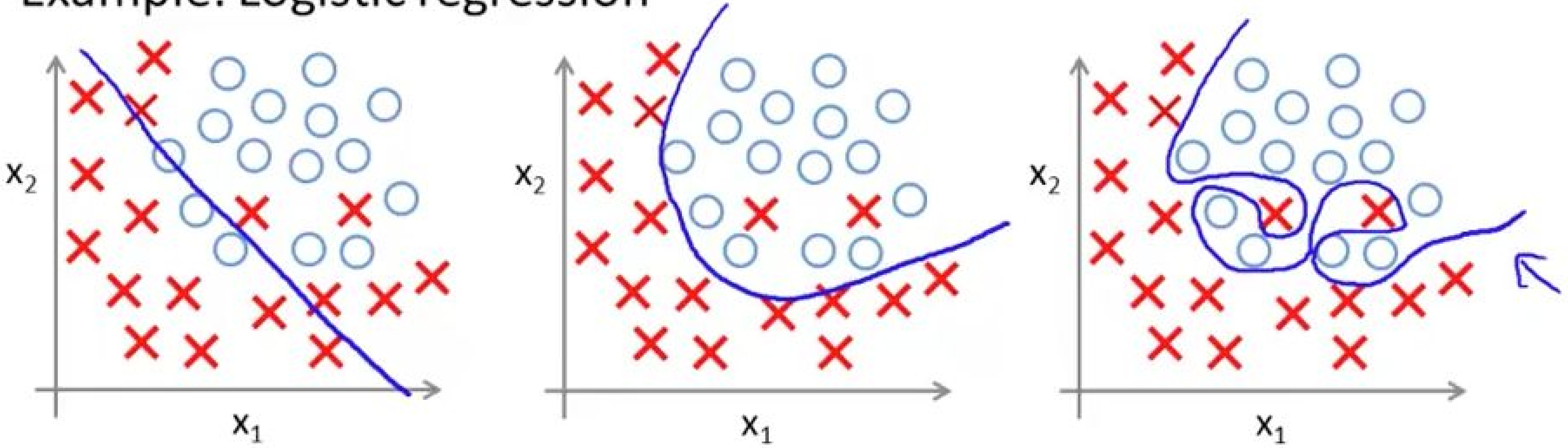
$\rightarrow \theta_0 + \theta_1 x$ "Underfit" "High bias"

$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$ "Just right"

$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$ "Overfit" "High variance"

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

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:

Options:

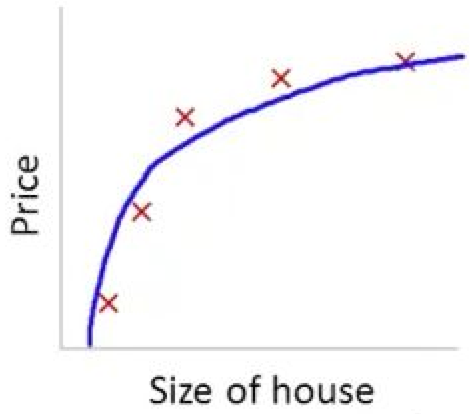
1. Reduce number of features.
   - Manually select which features to keep.
   - Model selection algorithm (later in course).
2. Regularization.
   - Keep all the features, but reduce magnitude/values of parameters θ_j.
   - Works well when we have a lot of features, each of which contributes a bit to predicting y.

Machine Learning

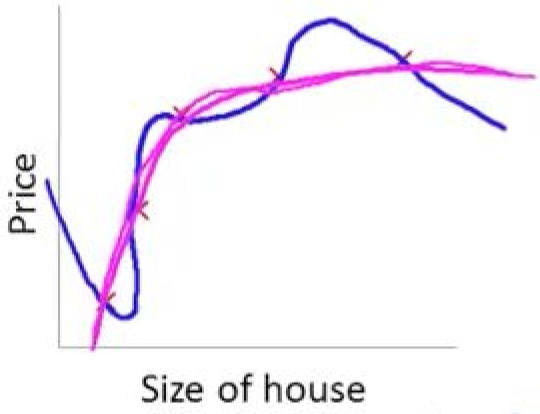
Regularization

Cost function

Intuition



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



$$\underline{\theta_0 + \theta_1 x + \theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$$

Suppose we penalize and make θ_3, θ_4 really small.

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

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

Regularization.

Small values for parameters $\boxed{\theta_0, \theta_1, \ldots, \theta_n}$

- "Simpler" hypothesis
- Less prone to overfitting

$\boxed{\theta_3, \theta_4}$

≈ 0

Housing:

- Features: $x_1, x_2, \ldots, x_{100}$
- Parameters: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{100}$

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

$\theta_1, \theta_2, \theta_3, \ldots, \theta_{100}$

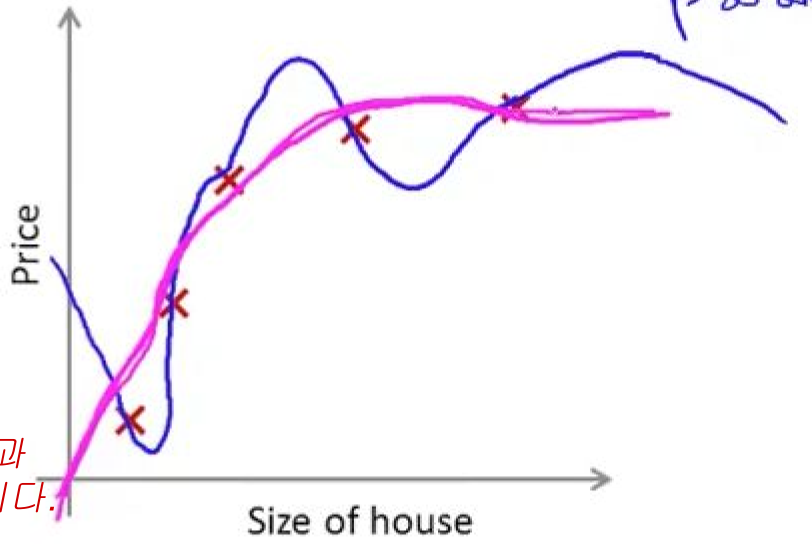
$\cancel{\theta_0}$

Regularization.

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

regularization parameter

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



파라미터의 값은 *hypothesis*가 잘 피팅되는 것과 오버피팅을 *shrink* 시키는 것 사이의 *trade off*이다.

In regularized linear regression, we choose θ to minimize

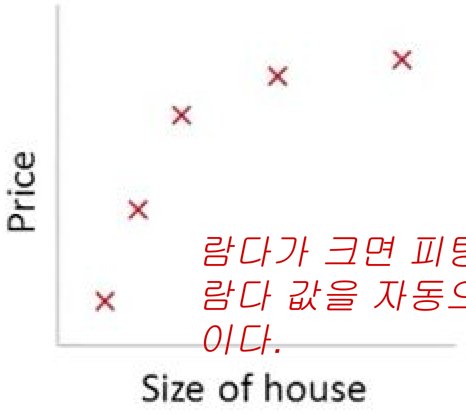
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \underline{\lambda}\sum_{j=1}^{n}\theta_j^2\right]$$

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

$\underline{\theta_1}, \underline{\theta_2}, \underline{\theta_3}, \underline{\theta_4}$

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

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



람다가 크면 피팅이 잘 안 될 수 있다(underfit).
람다 값을 자동으로 잡는 방법은 나중에 다룰 것이다.

$\underline{h_\theta(x) = \theta_0}$

$h_\theta(x) \rightarrow \theta_0 + \cancel{\theta_1 x} + \cancel{\theta_2 x^2} + \cancel{\theta_3 x^3} + \cancel{\theta_4 x^4}$

Andrew N

Machine Learning

Regularization

Regularized linear regression

Gradient descent

$\theta_0 \quad \theta_1, \theta_2, \dots, \theta_n$

Repeat {

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

$\frac{\partial}{\partial \theta_0} J(\theta)$

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

$$(j = \cancel{0}, 1, 2, 3, \dots, n)$$

}

regularization을 위해 수식을 변형해도
theta가 약간 줄어들고, Gradient descent는 동일하게 유지된다.

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

$1 - \alpha \frac{\lambda}{m} < 1$

0.99

$\theta_j \times 0.99$

θ_j^2

Normal equation

- *cost function을 최소로 만드는 theta를 방정식으로 푸는 방법*

$$X = \begin{bmatrix} (x^{(1)})^T \\ \vdots \\ (x^{(m)})^T \end{bmatrix} \leftarrow \qquad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \mathbb{R}^m$$

$m \times (n+1)$

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

$$\frac{\partial}{\partial \theta_j} J(\theta) \stackrel{\text{set}}{=} 0$$

$$\rightarrow \theta = \Big(X^T X + \lambda \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & & 0 \\ 0 & & 1 & \\ 0 & & & \ddots \\ 0 & 0 & \cdots & 1 \end{bmatrix}}_{(n+1)\times(n+1)} \Big)^{-1} X^T y$$

E.g. $n=2$ $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Normal equation에 대한 regularization도 가능하다.

Non-invertibility (optional/advanced).

Suppose $m \leq n$,

(#examples) (#features)

$$\theta = \underbrace{(X^T X)^{-1}}_{\text{non-invertible / singular}} X^T y$$

pinv inv

If $\lambda > 0$,

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \right)^{-1} X^T y$$

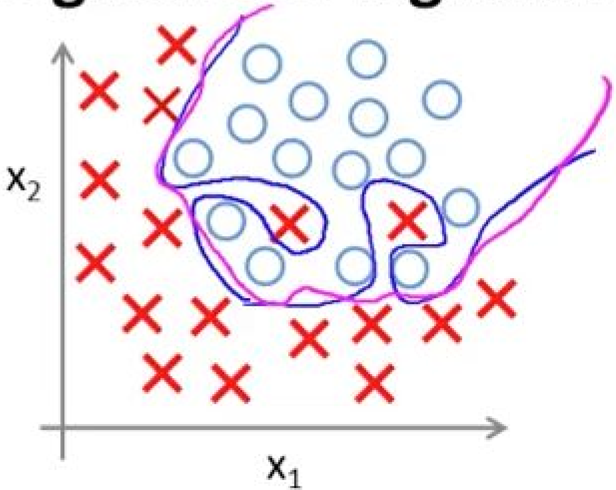
non-invertible 했던 문제가 regularization으로 invertible 해졌다.

Machine Learning

Regularization

Regularized logistic regression

Regularized logistic regression.



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

logistic regression에도 적용 가능하다.
feature가 많은 경우에도 regularization이 효과가 있다.

Cost function:

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

$$\boxed{\theta_1, \theta_2, \dots, \theta_n}$$

Gradient descent

Repeat $\{$

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

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

$(j = \cancel{0}, \underbrace{1, 2, 3, \ldots, n}_{\theta_1 \ldots \theta_n})$

$\}$

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

Gradient descent에서 linear regression과 같은 외형을 보이지만 hypothesis가 다르다.

fminunc(@costFunction...)

$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$ theta(1) theta(2) theta(n+1)

Advanced optimization

```
function [jVal, gradient] = costFunction(theta)
    jVal = [code to compute J(θ)];
```

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

gradient(1) = [code to compute $\boxed{\frac{\partial}{\partial\theta_0}J(\theta)}$];

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

gradient(2) = [code to compute $\boxed{\frac{\partial}{\partial\theta_1}J(\theta)}$];

$$\left(\boxed{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}}\right) + \frac{\lambda}{m}\theta_1$$

gradient(3) = [code to compute $\boxed{\frac{\partial}{\partial\theta_2}J(\theta)}$];

$$\left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}\right) + \frac{\lambda}{m}\theta_2$$

⋮

gradient(n+1) = [code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$];

regularization으로 적용한 gradient를 advanced optimization에도 적용 가능하다.