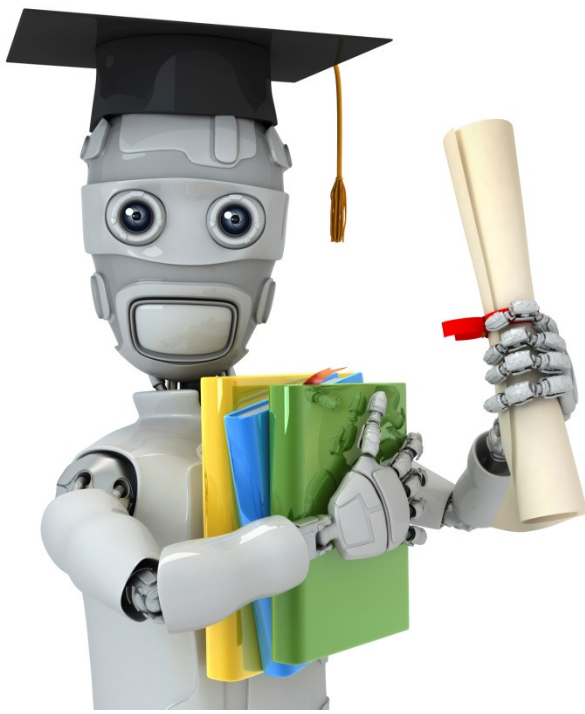
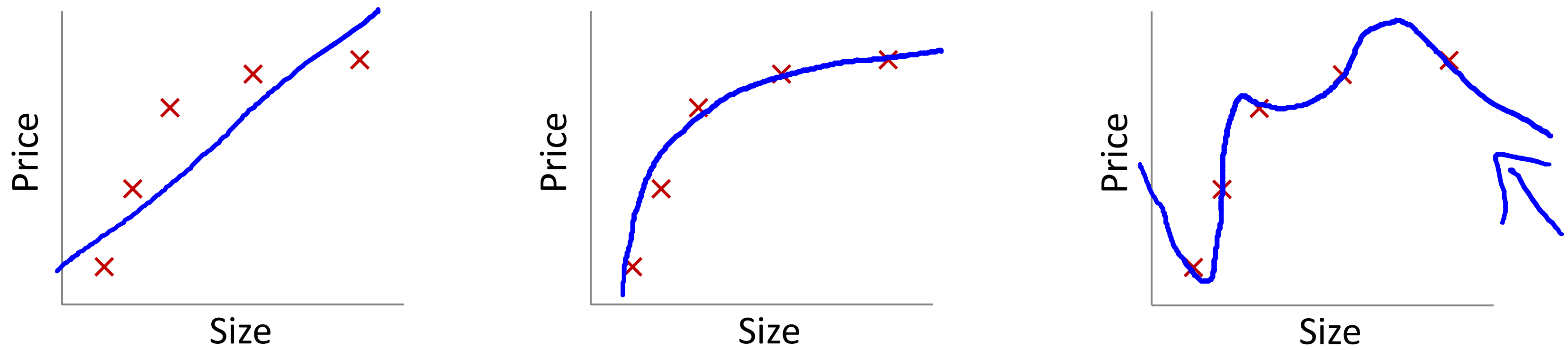


Machine Learning

Regularization

The problem of overfitting

Example: Linear regression (housing prices)



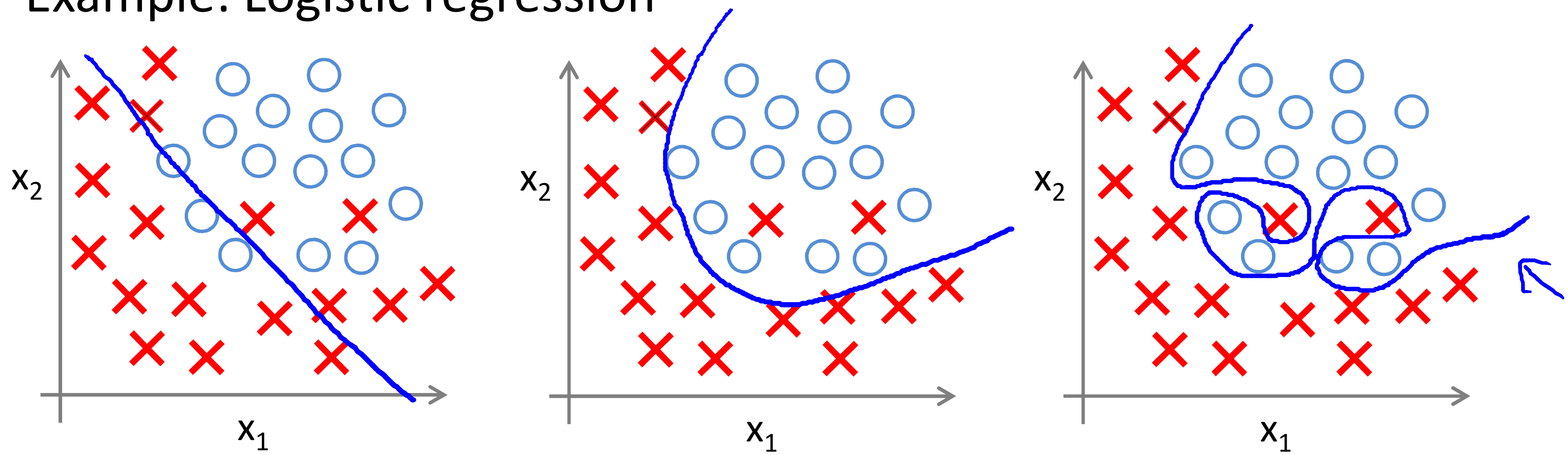
→ $\theta_0 + \theta_1 x$ → $\theta_0 + \theta_1 x + \theta_2 x^2$ → $\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Underfit" "High bias" "Just right" "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).

Andrew Ng

Example: Logistic regression



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

(g = sigmoid function)

"Underfit"

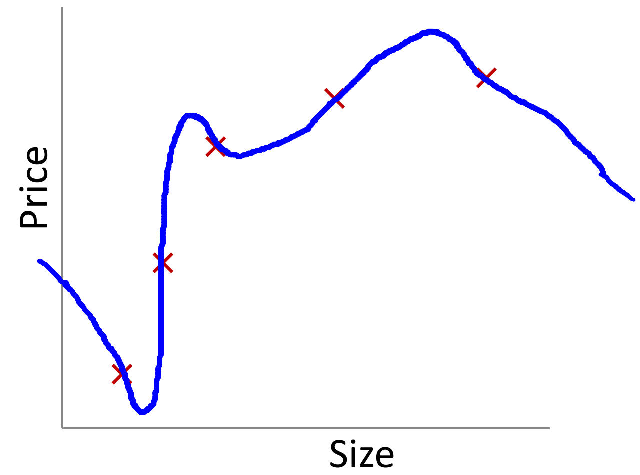
$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

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

"Overfit"

Addressing overfitting:

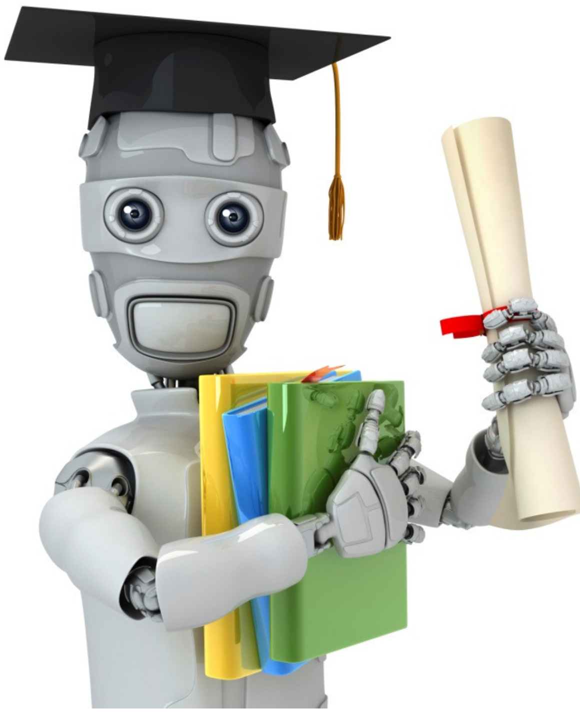
$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}



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.

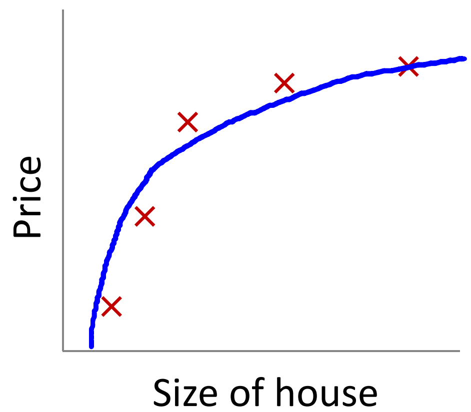


Machine Learning

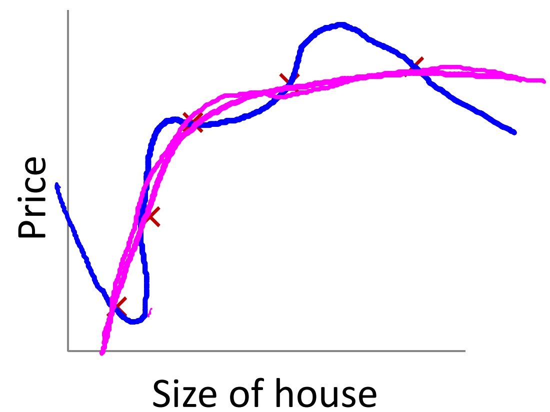
Regularization

Cost function

Intuition



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



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

Suppose we penalize and make θ_3, θ_4 really small.

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

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

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: $\theta_0, \theta_1, \theta_2, \ldots, \theta_{100}$

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

$\cancel{\theta_0}$, $\theta_1, \theta_2, \theta_3, \ldots, \theta_{100}$

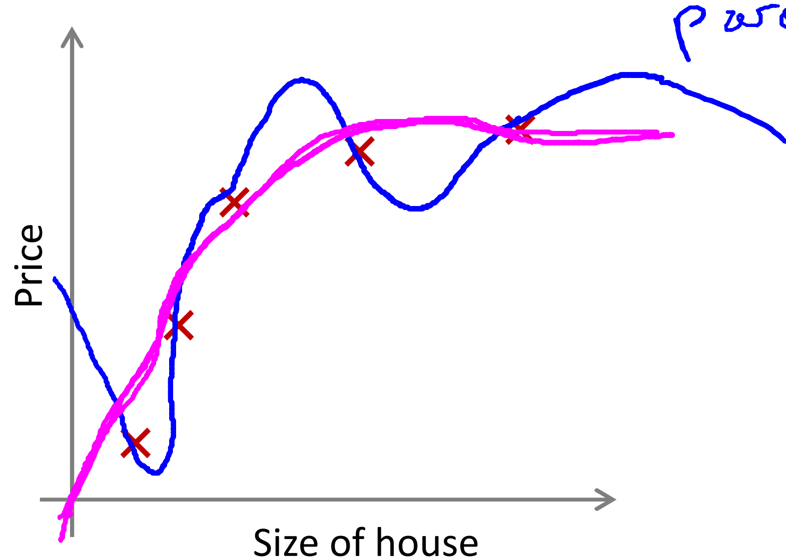
$\cancel{\theta_0}$

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

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



In regularized linear regression, we choose θ to minimize

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

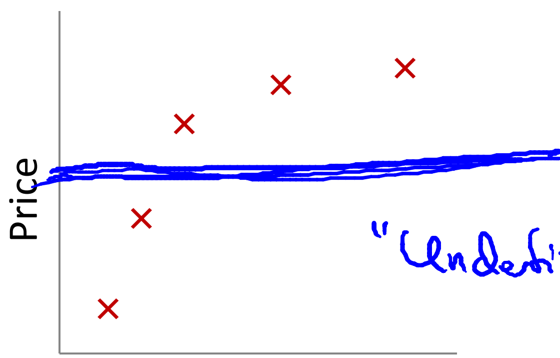
What if λ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- Algorithm works fine; setting λ to be very large can't hurt it
- Algortihm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

In regularized linear regression, we choose θ to minimize

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

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



"Underfit"

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

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

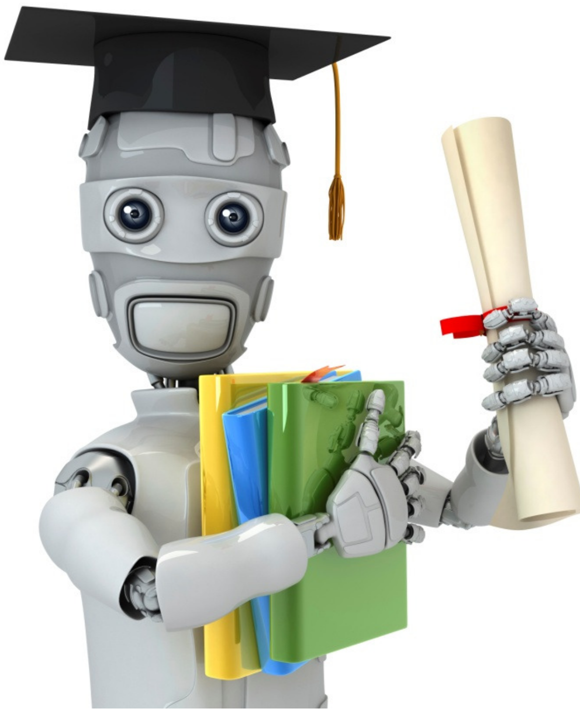
$\theta_3 \approx 0, \theta_4 \approx 0$

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

$h_\theta(x)$

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

Andrew Ng



Machine Learning

Regularization

Regularized linear regression

Regularized linear regression

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

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

Gradient descent

$\theta_0 \qquad \theta_1, \theta_2, \ldots, \theta_n$

Repeat {

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

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

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

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

}

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

$\rightarrow J(\theta)$

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

0.99

$\theta_j \times 0.99$

$\boxed{\theta_j^2}$

Andrew Ng

Normal equation

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

$m \times (n+1)$

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

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

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

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

Andrew Ng

Non-invertibility (optional/advanced).

Suppose $m \leq n$,

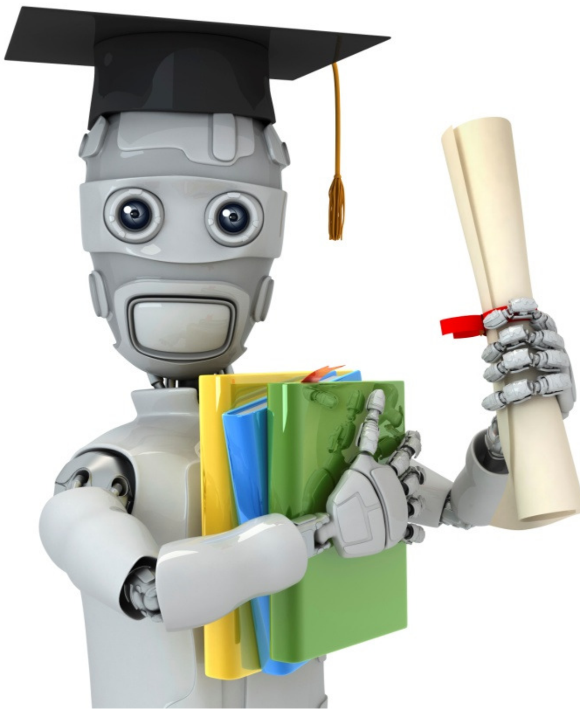
(#examples) (#features)

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

pinv inv

If $\lambda > 0$,

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

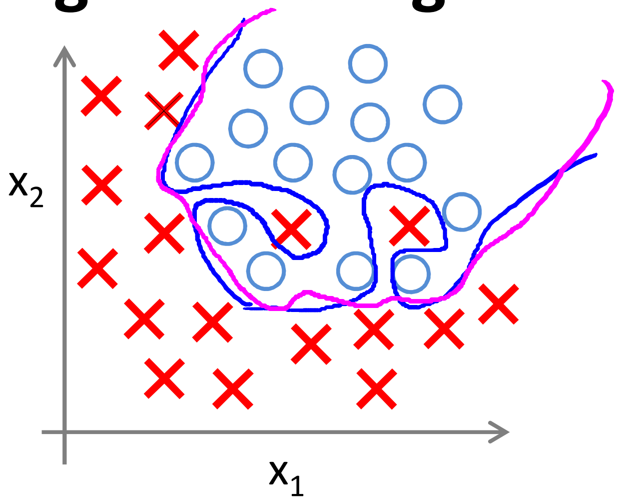


Machine Learning

Regularization

Regularized logistic regression

Regularized logistic regression.



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

Cost function:

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

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

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

Andrew Ng

Gradient descent

Repeat $\{$

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

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

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

$\}$

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

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 $J(\theta)$];

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

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 $\boxed{\frac{\partial}{\partial \theta_1} J(\theta)}$];

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

J(θ)

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 $\frac{\partial}{\partial \theta_n} J(\theta)$];