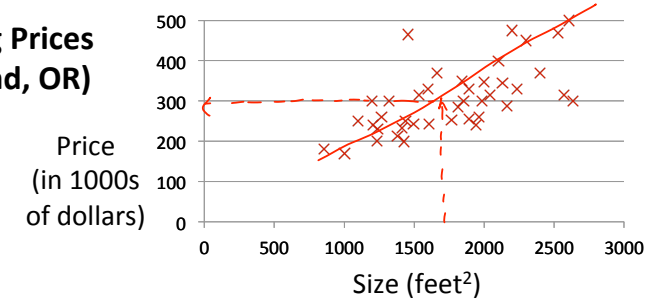
Internet Applications

Linear Regression with One Variable

Basic Concepts

Housing Prices (Portland, OR)



Supervised Learning

Given the "right answer" for each example in the data.

Regression Problem

Predict real-valued output

Training set of housing prices (Portland, OR)

Size in feet² (x)	Price ($) in 1000's (y)
$x^{(1)}$ = 2104	$y^{(1)}$ = 460
$x^{(2)}$ = 1416	$y^{(2)}$ = 232
= 1534	315
852	178
$x^{(m)}$ …	$y^{(m)}$ …

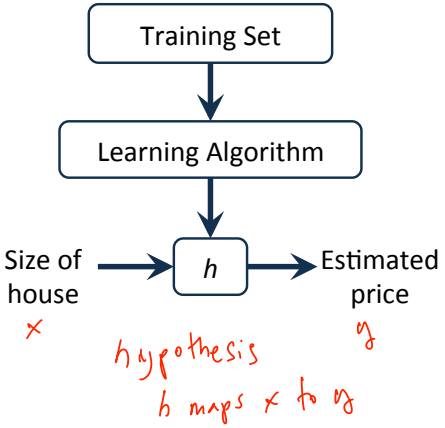
Notation:

m = Number of training examples

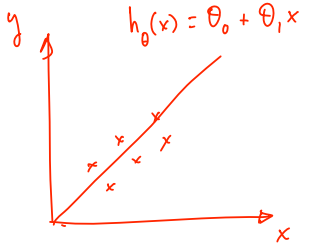
x's = "input" variable / features

y's = "output" variable / "target" variable

i^{th} training example = $(x^{(i)}, y^{(i)})$, $i = 1, \dots, m$

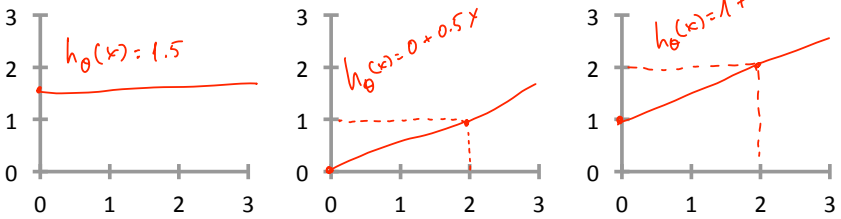


How do we represent *h* ?



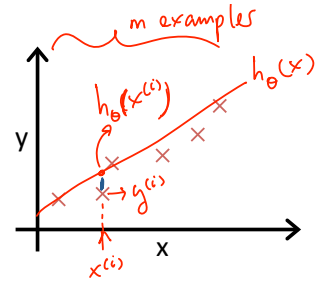
Linear regression with one variable.
Univariate linear regression.

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$ θ_0 y θ_1: parameters



$\theta_0 = 1.5$, $\theta_1 = 0$ | $\theta_0 = 0$, $\theta_1 = 0.5$ | $\theta_0 = 1$, $\theta_1 = 0.5$

How to choose θ_0 θ_1?



$$\underbrace{\frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}_{J(\theta_0,\theta_1)} \quad J(\theta)$$

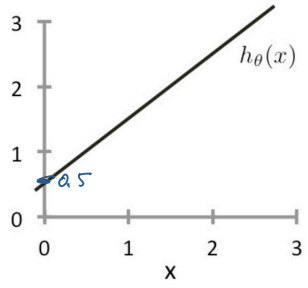
$J(\theta_0, \theta_1)$ = cost function

$\frac{1}{2}$ * mean squared error

Idea: Choose θ_0, θ_1 so that $h_\theta(x)$ is close to y for our training examples (x, y)

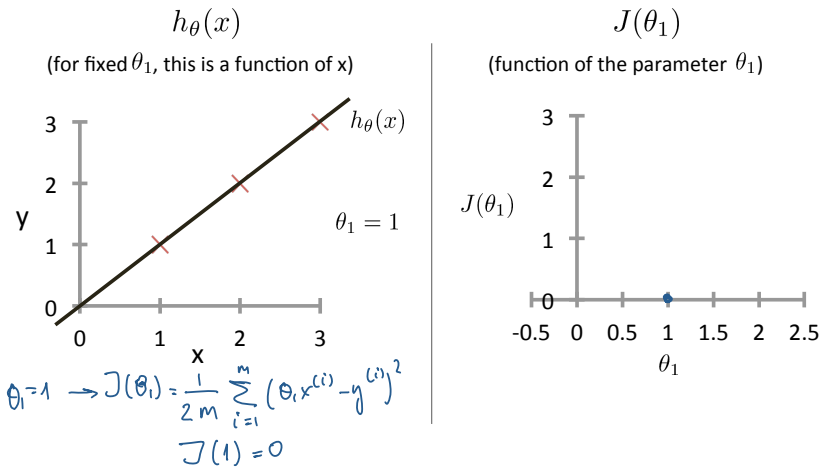
$$\min_{\theta_0,\theta_1} J(\theta_0, \theta_1)$$

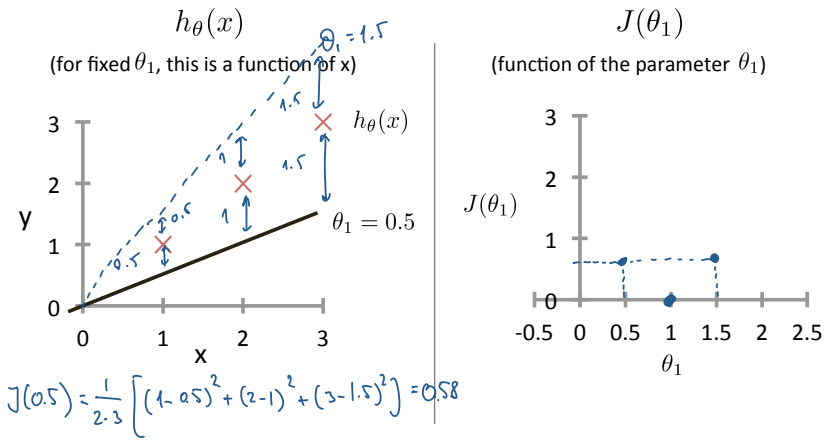
Given the hypothesis $h_\theta(x) = \theta_0 + \theta_1 x$. in the figure, select the right parametes values:



- ○ $\theta_0 = 0, \theta_1 = 1$
- ⊗ $\theta_0 = 0.5, \theta_1 = 1$
- ○ $\theta_0 = 1, \theta_1 = 0.5$
- ○ $\theta_0 = 1, \theta_1 = 1$

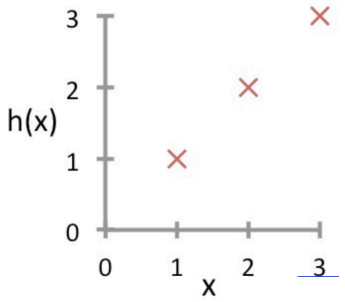
	Simplified
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$	$h_\theta(x) = \theta_1 x$
Parameters: θ_0, θ_1	θ_1
Cost Function: $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$	$J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$
Goal: $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$	$\underset{\theta_1}{\text{minimize}}\ J(\theta_1)$

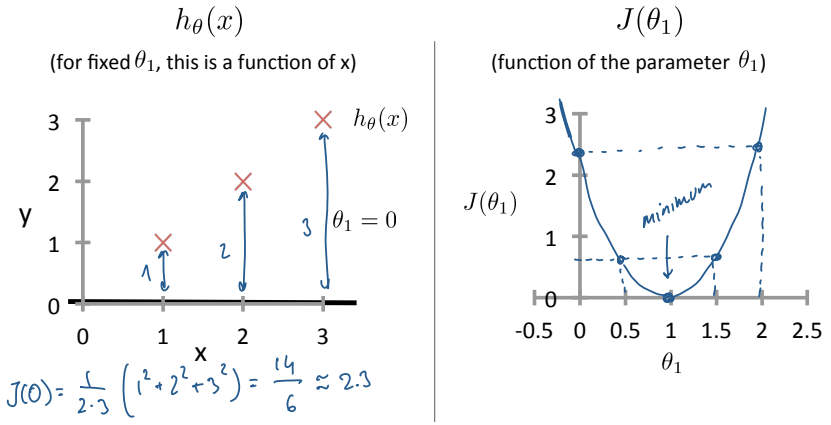




Consider m=3 inputs, shown in the figure.
The hypothesis function is $h_\theta(x) = \theta_1 x$
And the cost function is $J(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2$
Select $J(0)$

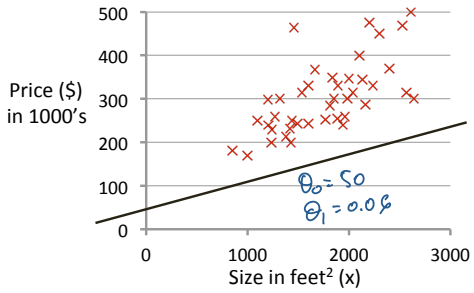
- 0
- 1/6
- 1
- 14/6





$h_\theta(x)$

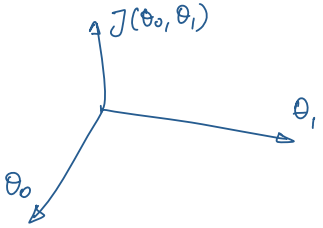
(for fixed θ_0, θ_1, this is a function of x)

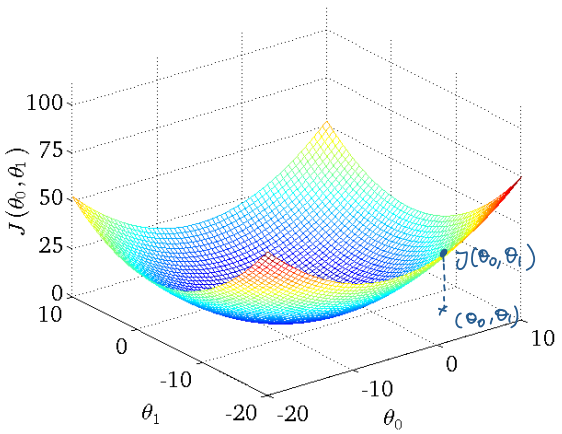


$h_\theta(x) = 50 + 0.06x$

$J(\theta_0, \theta_1)$

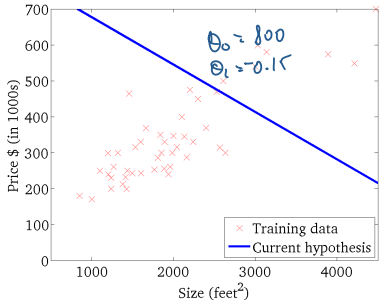
(function of the parameters θ_0, θ_1)





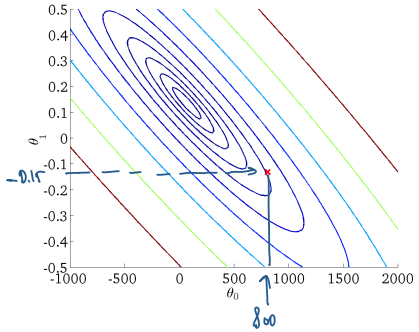
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



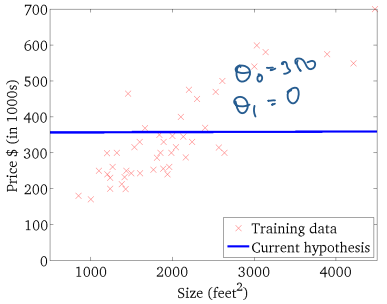
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



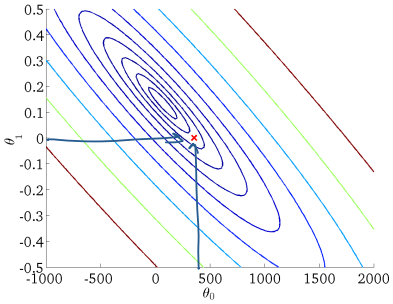
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



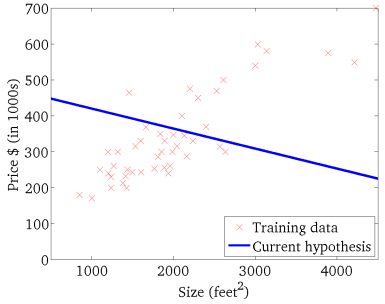
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



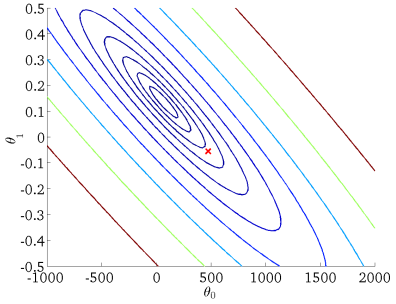
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



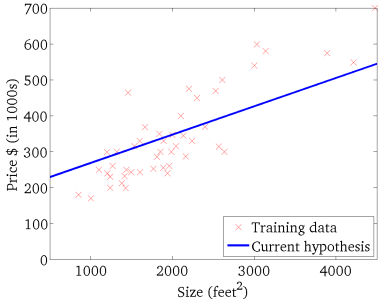
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



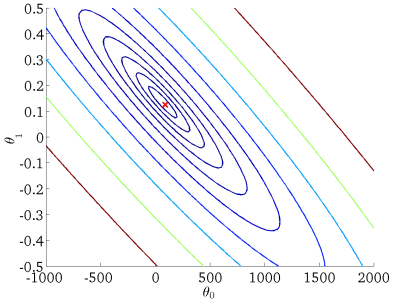
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



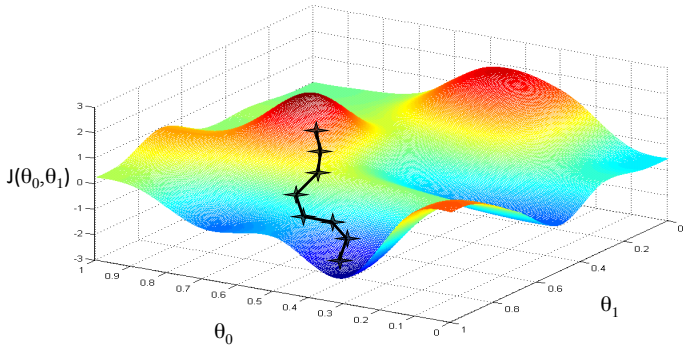
Gradient Descent Algorithm

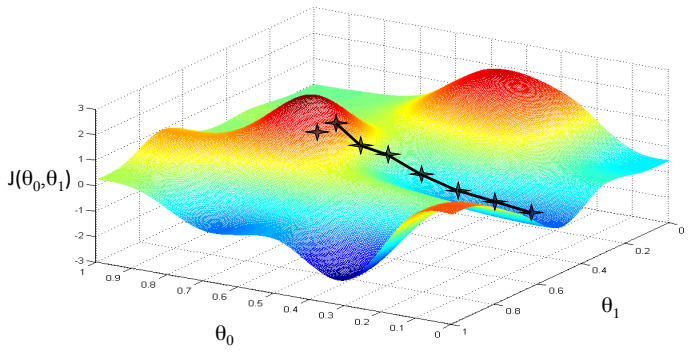
Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum





Gradient descent algorithm

$\theta := \theta - \alpha \nabla_\theta J(\theta)$

$\begin{pmatrix}\theta_0\\ \theta_1\end{pmatrix} := \begin{pmatrix}\theta_0\\ \theta_1\end{pmatrix} - \alpha \begin{pmatrix}\frac{\partial J(\theta)}{\partial \theta_0}\\ \frac{\partial J(\theta)}{\partial \theta_1}\end{pmatrix}$

repeat until convergence {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ (for $j = 0$ and $j = 1$)

}

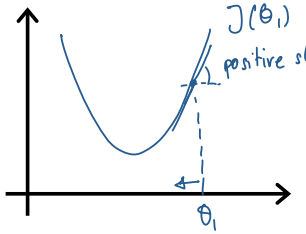
α: learning rate

Correct: Simultaneous update

$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_0 := \text{temp0}$
$\theta_1 := \text{temp1}$

Incorrect:

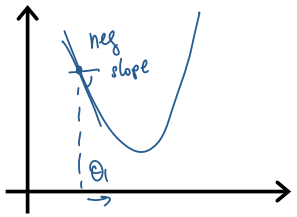
$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$
$\theta_0 := \text{temp0}$
$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$
$\theta_1 := \text{temp1}$



$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

>0

$\theta_1 := \theta_1 - \alpha$ (positive value)



$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$

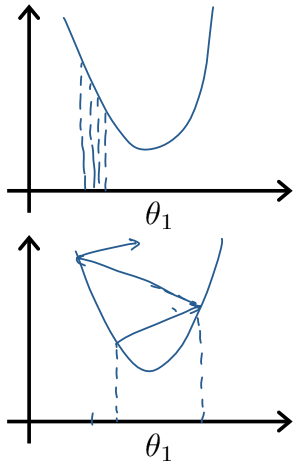
<0

$\theta_1 := \theta_1 - \alpha$ (negative)

$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$

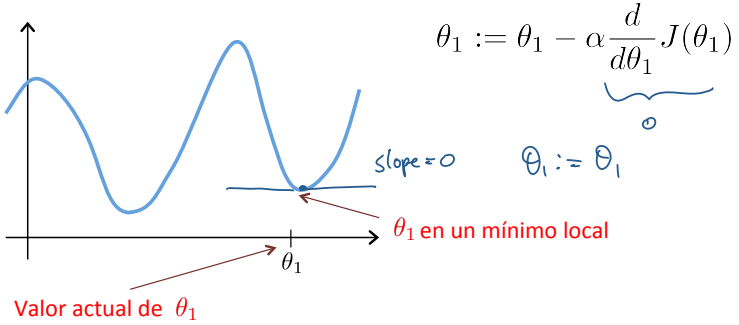
If α is too small, gradient descent can be slow.

If α is too large, gradient descent can overshoot the minimum. It may fail to converge, or even diverge.



Aspectos Prácticos

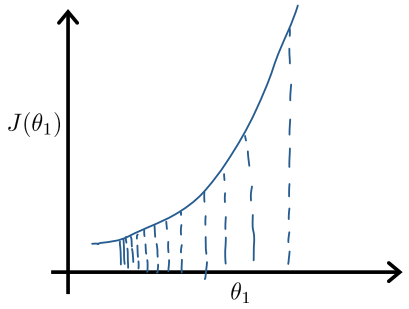
What happens in the gradient descent method if θ_1 is already at a local minimum?



Gradient descent can converge to a local minimum, even with the learning rate α fixed.

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

As we approach a local minimum, gradient descent will automatically take smaller steps. So, no need to decrease α over time.



Gradient descent for linear regression

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$$

(for $j = 1$ and $j = 0$)

}

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Gradient descent for linear regression

$$\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} \left(\frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2 \right) =$$

$$\frac{\partial}{\partial \theta_j} \left(\frac{1}{2m} \sum_{i=1}^{m} \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)^2 \right)$$

$j = 0$: $\frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} 2 \left(\theta_0 + \theta_1 x^{(i)} - y^{(i)} \right)$

$j = 0 : \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$

$j = 1 : \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x^{(i)}$

Gradient descent for linear regression

Algorithm:

repeat until convergence {

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) \cdot x^{(i)}$$

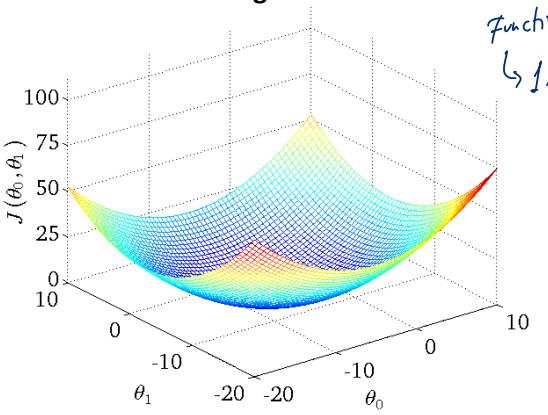
}

$\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_0}$ $\frac{\partial J(\theta_0,\theta_1)}{\partial \theta_1}$

Update θ_0 and θ_1 simultaneously

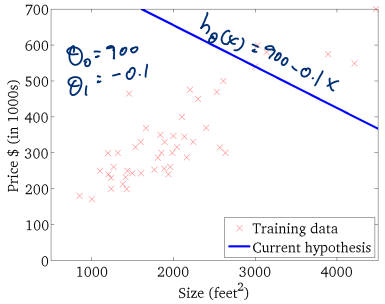
Gradient descent for linear regression

Convex function → 1 global minimum



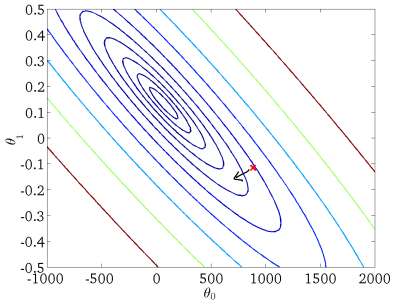
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



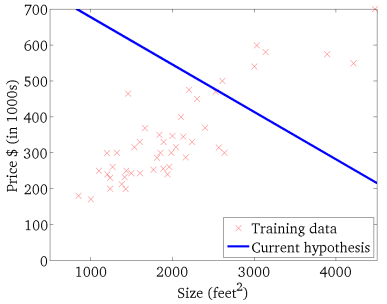
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



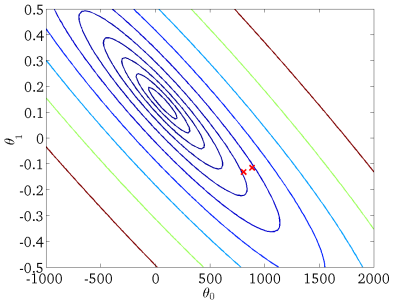
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



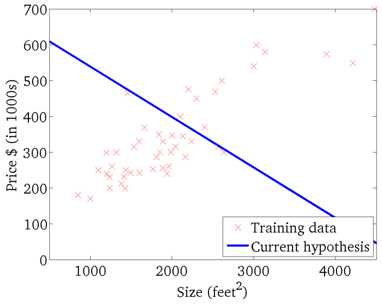
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



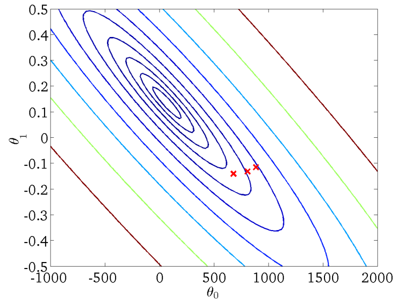
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



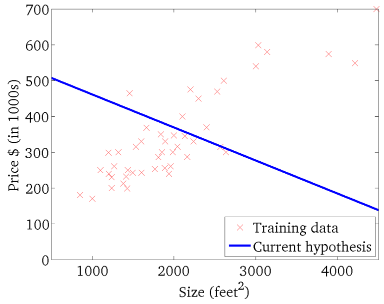
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



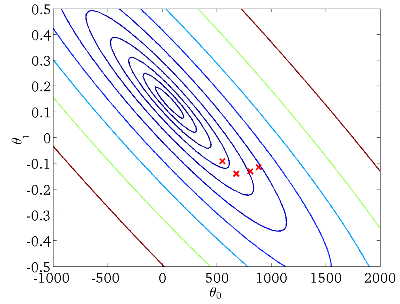
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



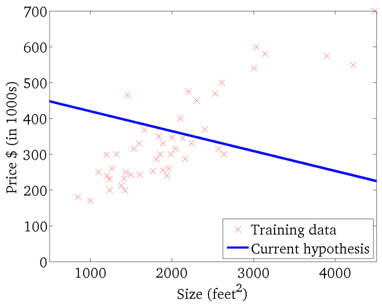
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



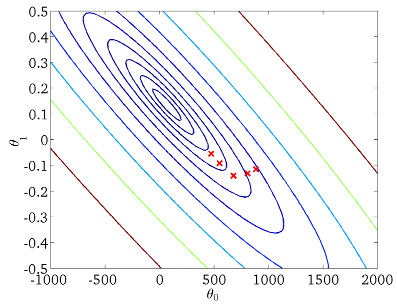
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



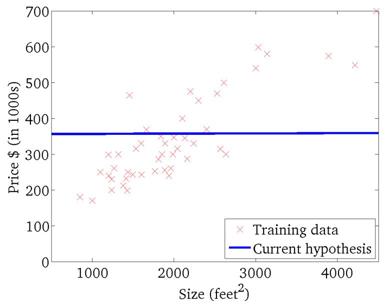
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



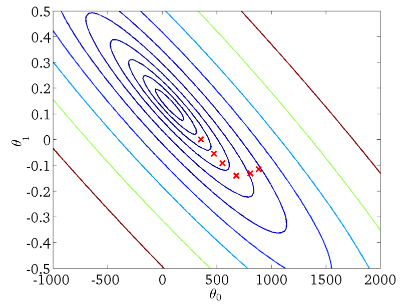
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



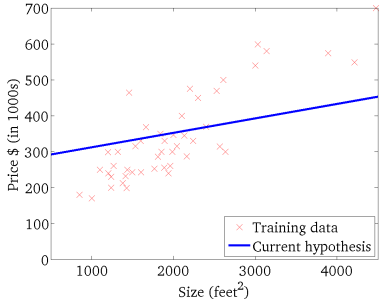
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



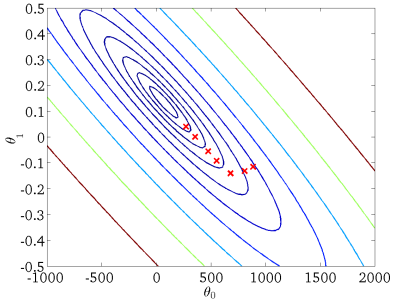
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



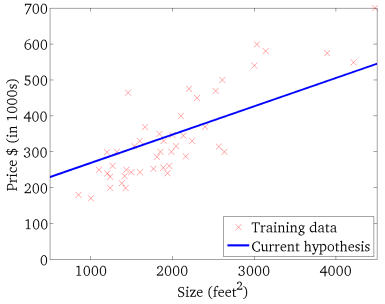
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



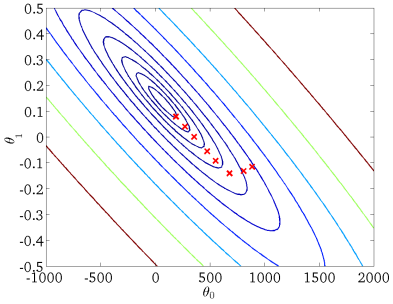
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



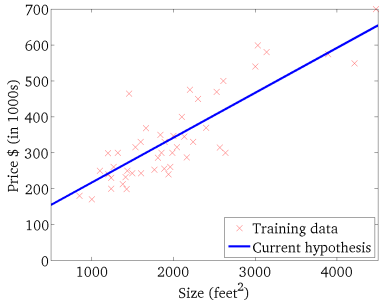
$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



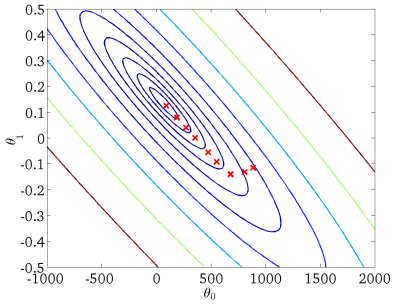
$h_\theta(x)$

(for fixed θ_0, θ_1, this is a function of x)



$J(\theta_0, \theta_1)$

(function of the parameters θ_0, θ_1)



Which of the following statements are correct?

- ☐ In order for the gradient descent to converge, we must gradually decrease over time.
- ☐ The gradient descent ensures that an overall minimum is reached for any J function J(θ_0, θ_1)
- ☒ The gradient descent can converge even if α it remains steady (although α shouldn't be too large or the algorithm may diverge)
- ☒ For function J(θ_0, θ_1) associated with linear regression, there are no local optimum, there is only one global optimal.

Multiple features

Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$

Multiple features (variables).

Size (feet²)	Number of bedrooms	Number of floors	Age of home (years)	Price ($1000)
2104	5	1	45	460
1416	3	2	40	232
1534	3	2	30	315
852	2	1	36	178
...	...	...	...	...

$x^{(2)} \rightarrow$ (1416, 3, 2, 40) $\rightarrow y^{(2)}$ = 232

Notation:

n = number of features

$x^{(i)}$ = input (features) of i^{th} training example.

$x_j^{(i)}$ = value of feature j in i^{th} training example.

example: $n = 4$

$x_1^{(2)} = 1416$

$x_2^{(2)} = 3$

$x_3^{(2)} = 2$

Hypothesis:

Previously: $h_\theta(x) = \theta_0 + \theta_1 x$

Now: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

For convenience of notation, define: $x_0 = 1$

$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$ $\quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$

$\theta^T \cdot x = [\theta_0 \theta_1 \cdots \theta_n] \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$ $\quad = x^T \cdot \theta$

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

Multivariate linear regression.

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$ $\rightarrow \theta$ an $n+1$ dimensional vector

Cost function:

$J(\theta) = J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Gradient descent:

Repeat $\{$

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$ $\quad J(\theta)$

$\}$ (simultaneously update for every $j = 0, \ldots, n$)

Gradient Descent

Previously (n=1):

Repeat $\{$

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x^{(i)}$$

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$$

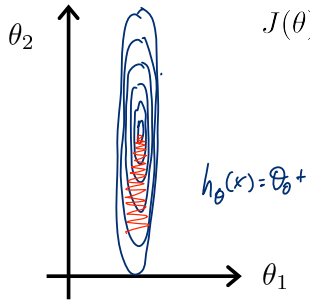
$\ldots$

Practical aspects I: Feature Scaling

Idea: Make sure features are on a similar scale.

E.g. x_1 = size (0-2000 feet^2)

x_2 = number of bedrooms (1-5)

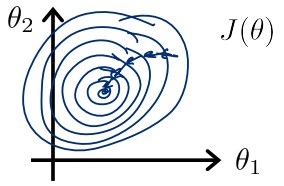


$$x_1 = \frac{\text{size (feet}^2\text{)}}{2000}$$

$$x_2 = \frac{\text{number of bedrooms}}{5}$$

$0 \leq x_1 \leq 1 \quad 0 \leq x_2 \leq 1$

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



Practical aspects II: Mean normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean (Do not apply to $x_0 = 1$).

E.g. $x_1 = \frac{size - 1000}{2000}$

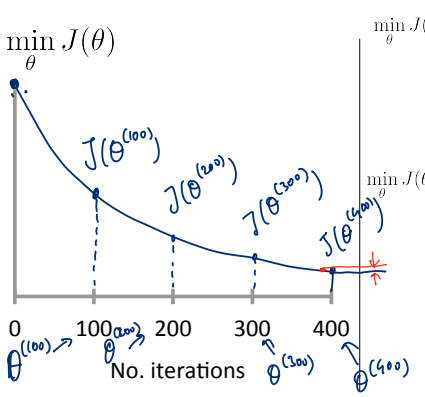
$x_2 = \frac{\#bedrooms - 2}{5}$

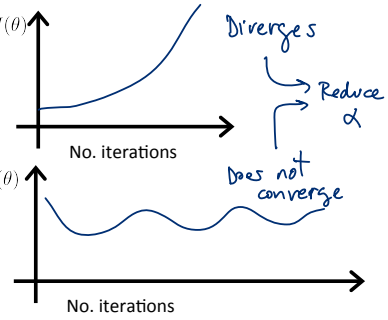
$$-0.5 \leq x_1 \leq 0.5, -0.5 \leq x_2 \leq 0.5$$

$x_j = \frac{x_j - \mu_j}{s_j}$ s_j { range (max - min) / std dev.

It let us interpret θ_0 as the expected value of y when the features have the average value

Practical aspects III: Adjust learning rate α





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

Summary:

- If α is too small: slow convergence.
- If α is too large: $J(\theta)$ may not decrease on every iteration; may not converge.

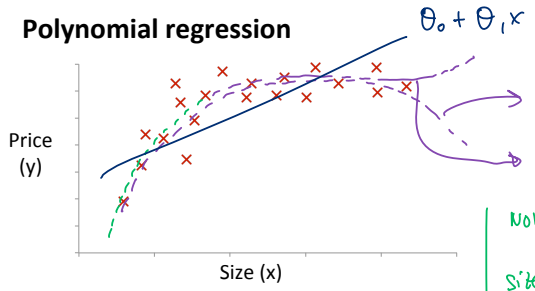
To choose α, try

$$\ldots, 0.001, \quad 0.003, \quad 0.01, \quad 0.03, \quad 0.1, \quad 0.3, \quad 1, \ldots$$

$\times 10$ $\times 10$ $\times 10$

$\times 3$ $\approx \times 3$ $\times 3$ $\approx \times 3$ $\times 3$ ...

Polynomial regression



$\theta_0 + \theta_1 x$

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

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

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1 (size) + \theta_2 (size)^2 + \theta_3 (size)^3$$

$x_1 = (size)$
$x_2 = (size)^2$
$x_3 = (size)^3$

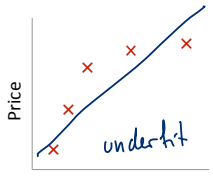
NORMALIZATION is required

$size \in \{1 \ldots 1000\}$

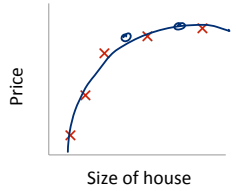
$size^2 \in \{1 \ldots 1000000\}$

$size^3 \in \{1 \ldots 10^9\}$

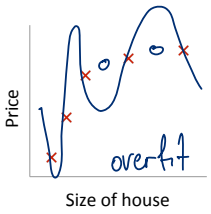
Regularization



$\theta_0 + \theta_1 x$



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



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

Assume we penalize the size of θ_3 θ_4

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

Regularization

Smaller values for ~~θ_0~~, $\theta_1, \ldots, \theta_n$

- "simpler" hypothesis
- Less prone to overfitting

Example:

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

regularization factor

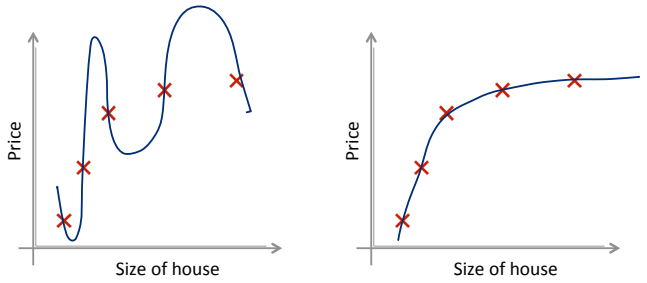
$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \cdot \sum_{j=1}^{n} \theta_j^2 \right]$$

other option $+ \lambda \sum_{j=1}^{n} |\theta_j|$

Regularization

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

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



In regularized linear regression, we choose θ to minimize

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

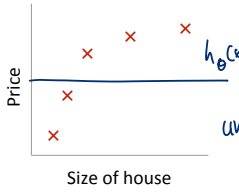
When λ is set to an extremely high value, for example, $\lambda = 10^{10}$ what happens?

- ❑ The algorithm works well, increasing λ can't hurt it.
- ❑ The algorithm fails to eliminate overfitting.
- ❑ The algorithm suffers underfitting. (It can't even adjust the training data.
- ❑ The descent of the gradient doesn't manage to converge.

In regularized linear regression, we choose θ to minimize

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

When λ is set to an extremely high value, for example, $\lambda = 10^{10}$ what happens? $\theta_1 \approx 0 \quad \theta_2 \approx 0 \quad \theta_3 \approx 0 \quad \theta_4 \approx 0$



$h_\theta(x) = \theta_0 \rightarrow$ empirical mean of $y^{(1)} \dots y^{(m)}$

underfit

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

Regresión lineal regularizada

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

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

$j = 0 \quad \frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})\underbrace{x_0^{(i)}}_{1}\right]$

$j \neq 0 \quad \frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \lambda \theta_j\right] \quad j = 1 \dots n$

this is the only change

Gradient descent

Repeat $\{$

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

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

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

$\}$

θ_j common factor

$$\theta_j := \underbrace{\theta_j (1 - \alpha \frac{\lambda}{m})}_{\text{Reduce } \theta_j} - \underbrace{\alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}}_{\text{update with gradient descent step}}$$

m training examples

- m' → training set: $x^{(1)}, y^{(1)}$; $x^{(2)}, y^{(2)}$; …; $x^{(m')}, y^{(m')}$
- t → test set: $x^{(m'+1)}, y^{(m'+1)}$; …; $x^{(m)}, y^{(m)}$

$m' \gg t$

training set $(x^{(i)}, y^{(i)})$ $i = 1 \ldots m'$

test set $(x^{(i)}, y^{(i)})$ $i = m'+1 \ldots m$

For a given λ

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

$$\theta^{\lambda} = \arg\min_{\theta} J_{train}^{\lambda}(\theta)$$

$$MSE_{train}(\lambda) = \frac{1}{m'} \left(\sum_{i=1}^{m'} \left(h_{\theta^\lambda}(x^{(i)}) - y^{(i)} \right)^2 \right)$$

$$MSE_{test}(\lambda) = \frac{1}{t} \left(\sum_{i=m'+1}^{m} \left(h_{\theta^\lambda}(x^{(i)}) - y^{(i)} \right)^2 \right)$$

Regularization factor adjustment

$$\lambda^* = \arg\min_{\lambda} MSE_{test}(\lambda)$$