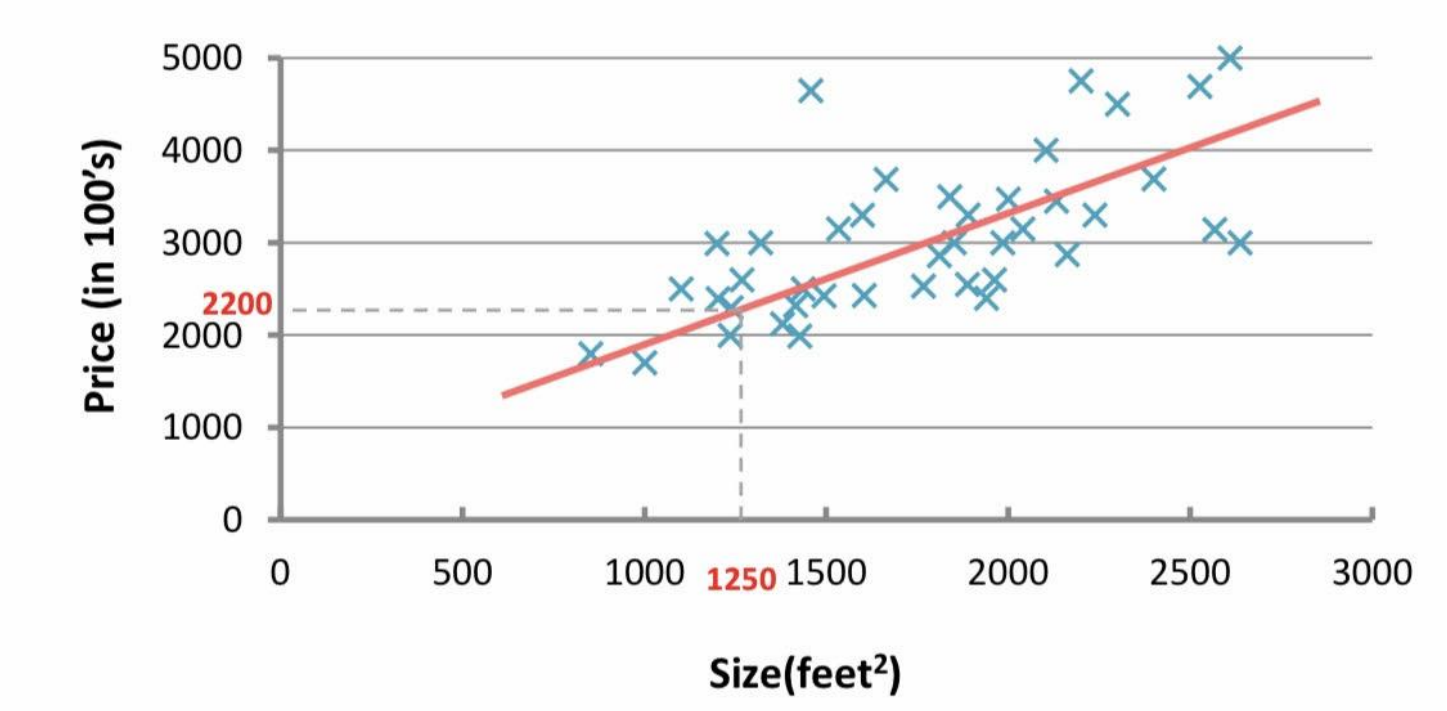
Machine Learning

By Ghazal Lalooha

Supervised learning: Regression

Table of Contents

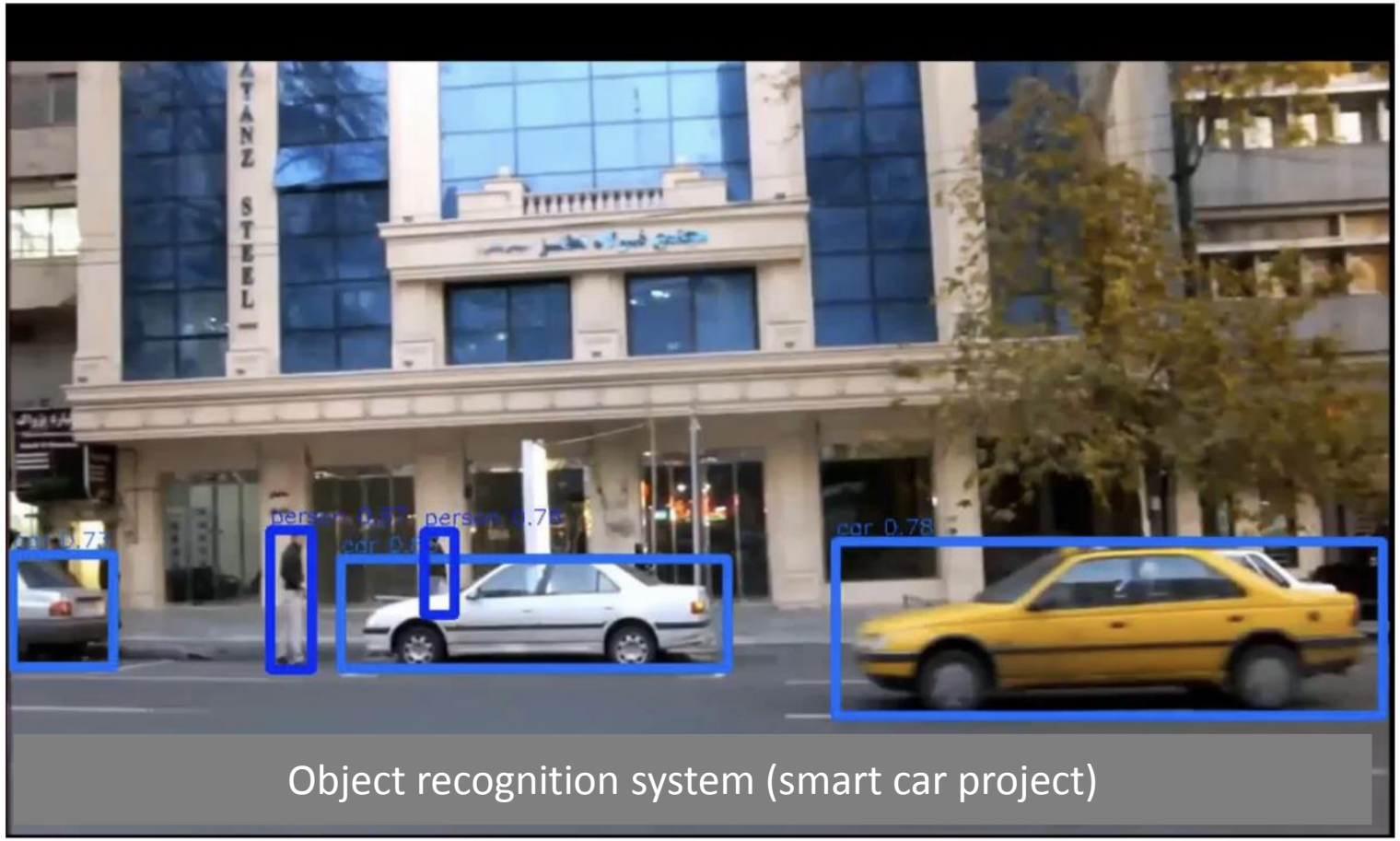
Univariate linear regression



Supervised learning:
For each training example,
a "correct answer" is given.

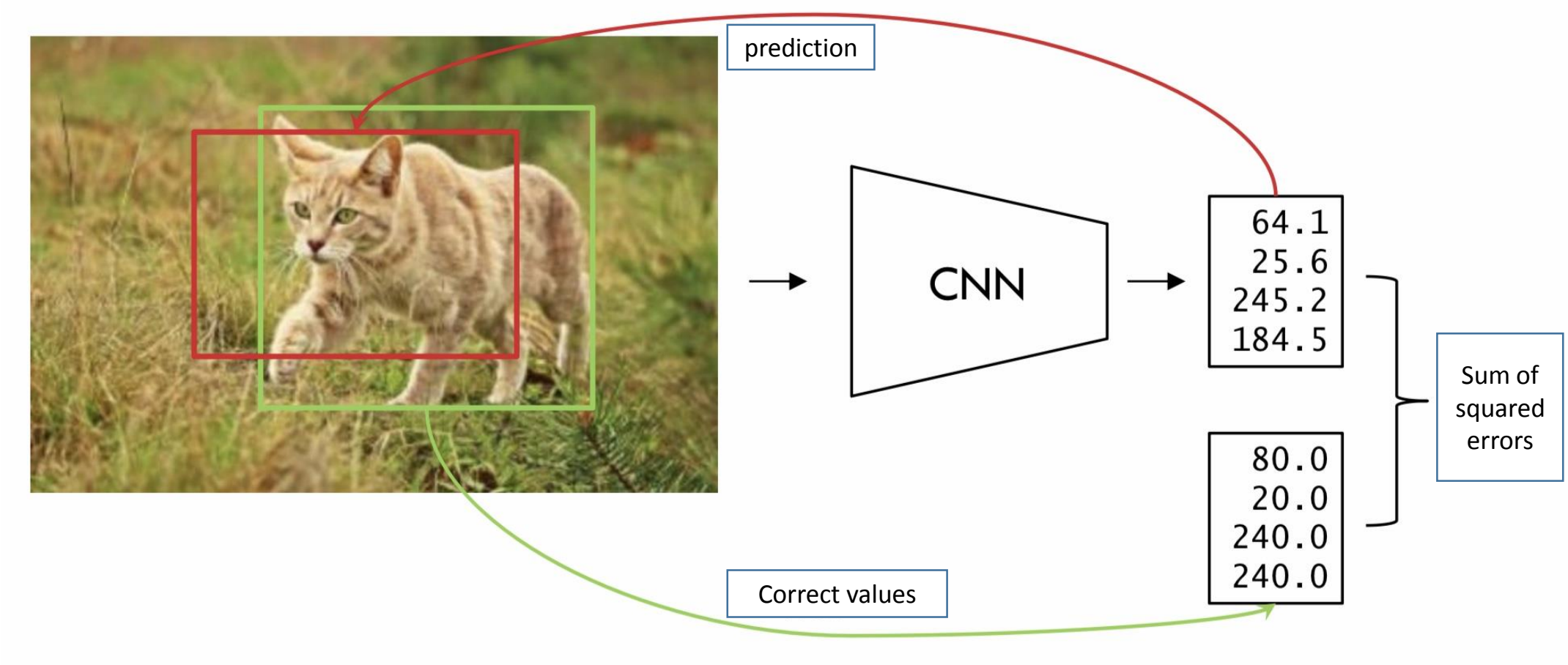
regression:
Predicting quantities with continuous values (such as the price of a house)

Regression: Things Detection



Object recognition system (smart car project)

Locating as regression



Symbolization

Training set

Price (in thousand dollars) (y)	Meterage (squared foot) (x)
460	2104
232	1416
315	1534
178	852
…	…

m = 47

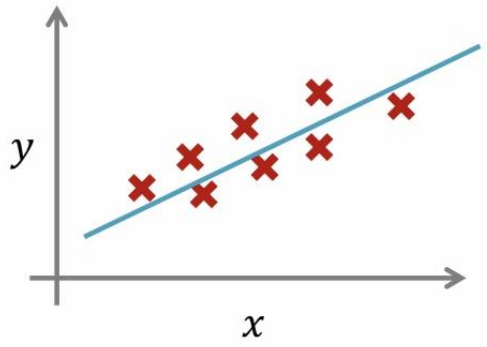
Symbols:
m = the number of training examples
x = “input” variable, features
y = “output” variable, “target” variable

(x, y): a training example
($x^{(i)}$, $y^{(i)}$): ith training example

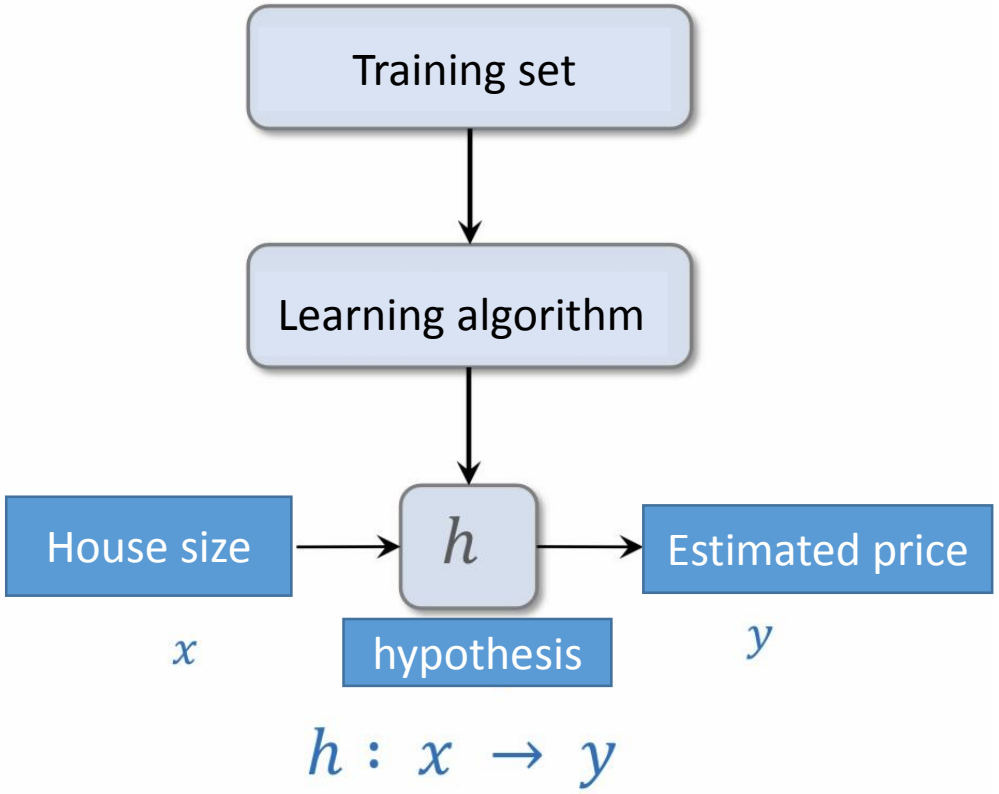
Model representation

Display hypothesis h

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



Univariate linear regression



Cost function

Hypothesis evaluation

Training set

Price (in thousand dollars) (y)	Meterage (squared foot) (x)
460	2104
232	1416
315	1534
178	852
…	…

m = 47

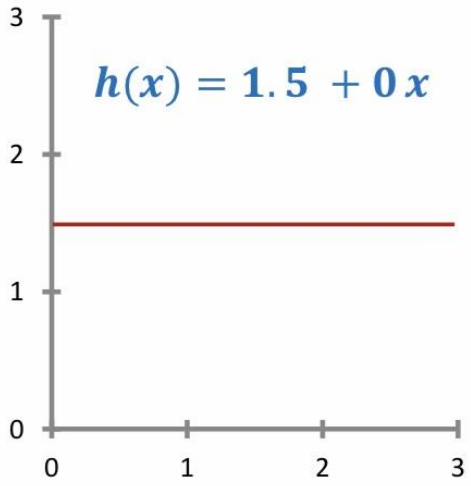
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

parameters: (θ_0, θ_1)

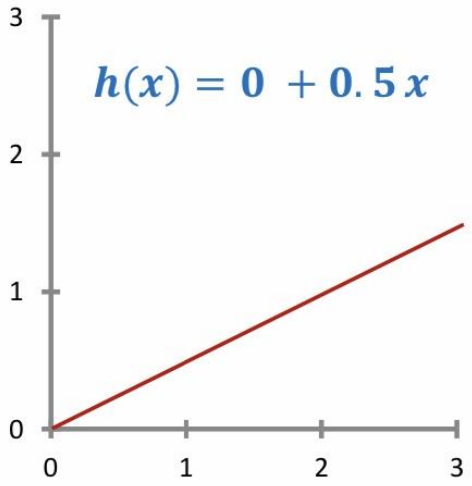
Question: How to choose the value of the parameters?

Hypothesis evaluation

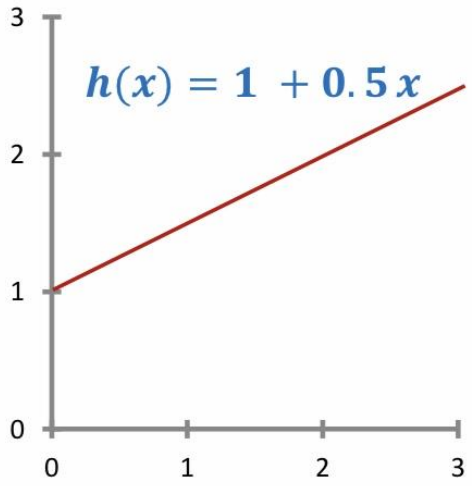
$$h_\theta(x) = \theta_0 + \theta_1 x$$



$\theta_0 = 1.5$
$\theta_1 = 0$



$\theta_0 = 0$
$\theta_1 = 0.5$

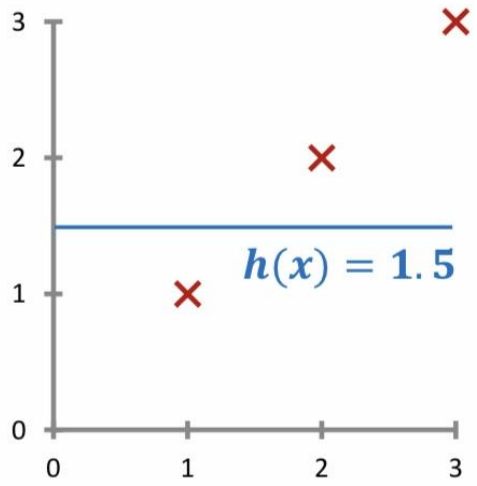


$\theta_0 = 1$
$\theta_1 = 0.5$

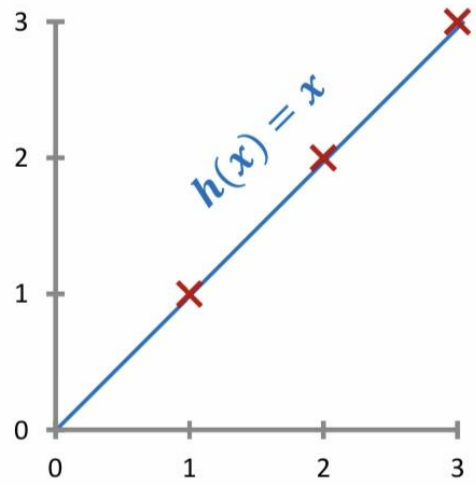
Hypothesis evaluation

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

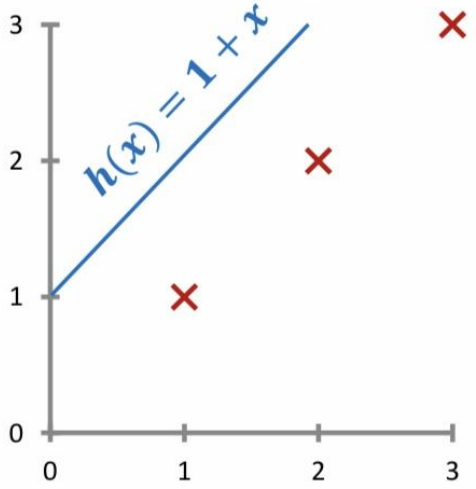
Question: Which hypothesis is better?



$\theta_0 = 1.5$
$\theta_1 = 0$



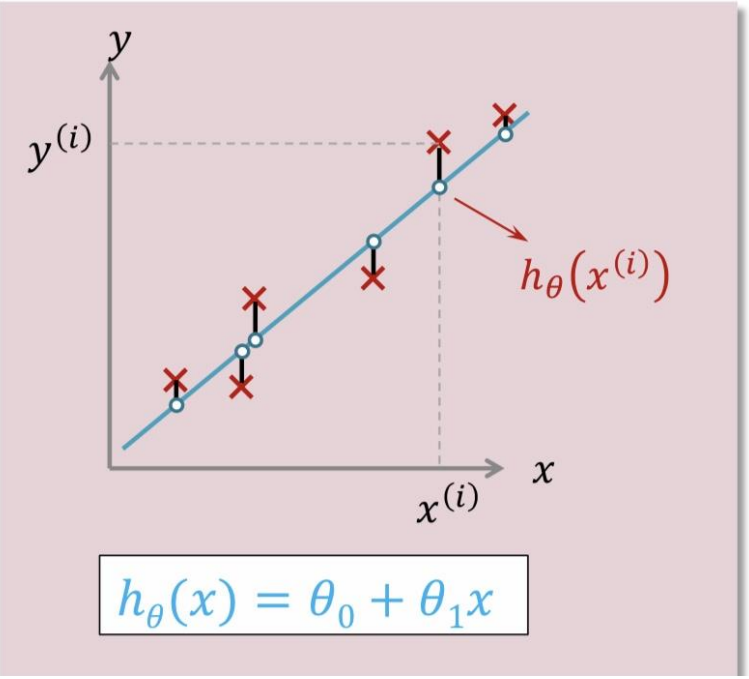
$\theta_0 = 0$
$\theta_1 = 1$



$\theta_0 = 1$
$\theta_1 = 1$

Cost function

- Idea: choosing the parameters so that for each training sample such as (x,y), the value of h_θ (x) is as close as possible to the value of y.



Cost function: Sum of Squared Errors

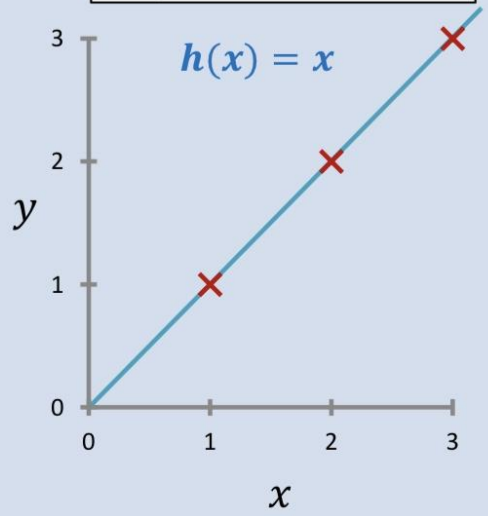
$$J(\theta_0, \theta_1) = \frac{1}{2}\sum^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

Target:

$$\underset{\theta_0,\theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$$

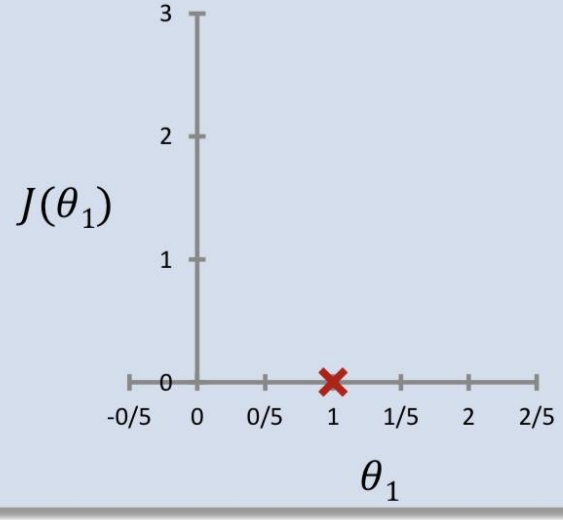
Simplified Cost Function ($\theta_0 = 0$)

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



$\theta_1 = 1$

$J(\theta_1)$

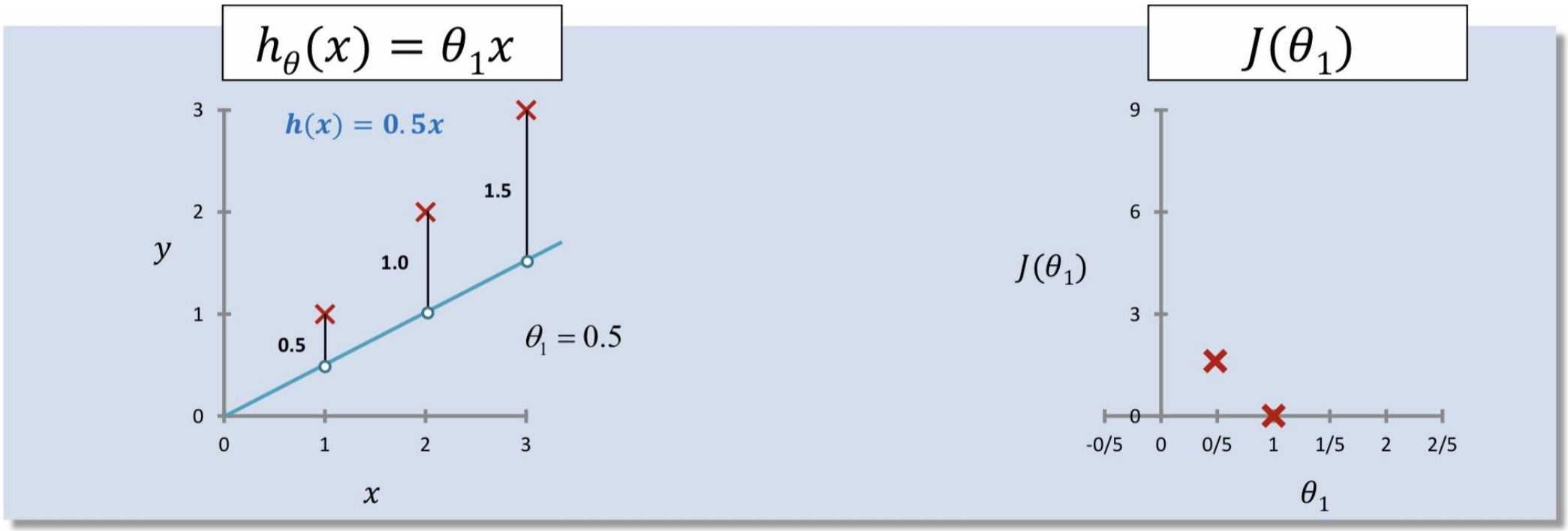


$$J(\theta_0, \theta_1) = \frac{1}{2}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$$= \frac{1}{2}\sum_{i=1}^{m}\left(x^{(i)} - y^{(i)}\right)^2 = \frac{1}{2}(0^2 + 0^2 + 0^2) = 0$$

$J(1) = 0$

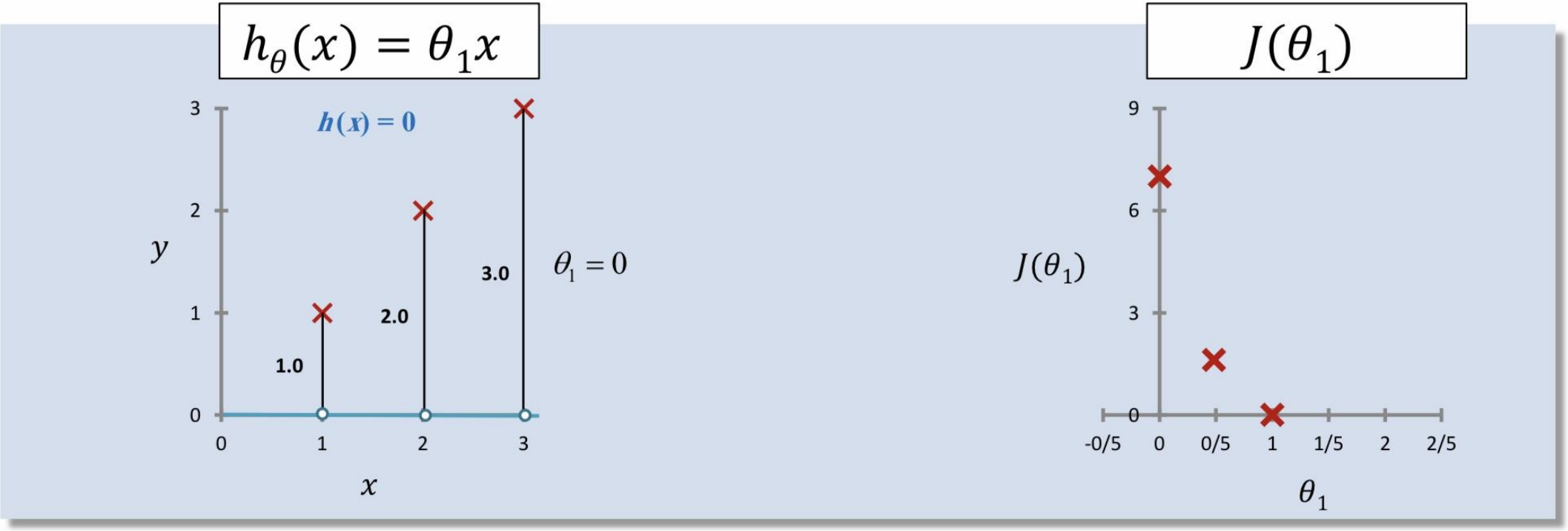
Cost function



$J(0.5) = \frac{1}{2}(0.5^2 + 1.0^2 + 1.5^2) = \frac{1}{2}(3.5) = 1.75$

$J(0.5) = 1.75$

Cost function

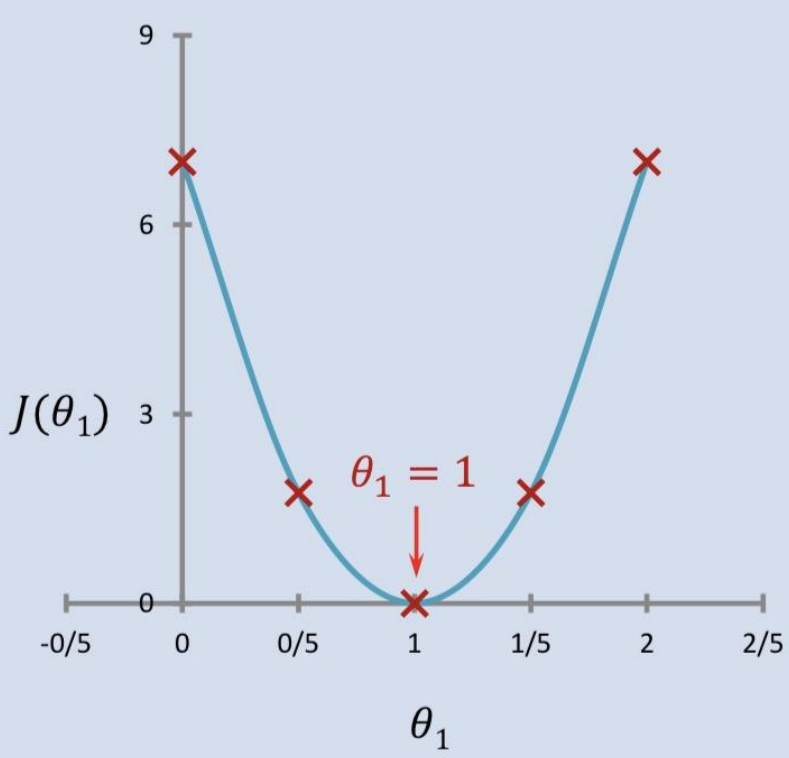


$J(0) = \frac{1}{2}(1.0^2 + 2.0^2 + 3.0^2) = \frac{1}{2}(14) = 7.0$

$J(0) = 7.0$

Cost function

$$\underset{\theta_1}{\text{minimize}} \; J(\theta_1)$$



Univariate linear regression

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

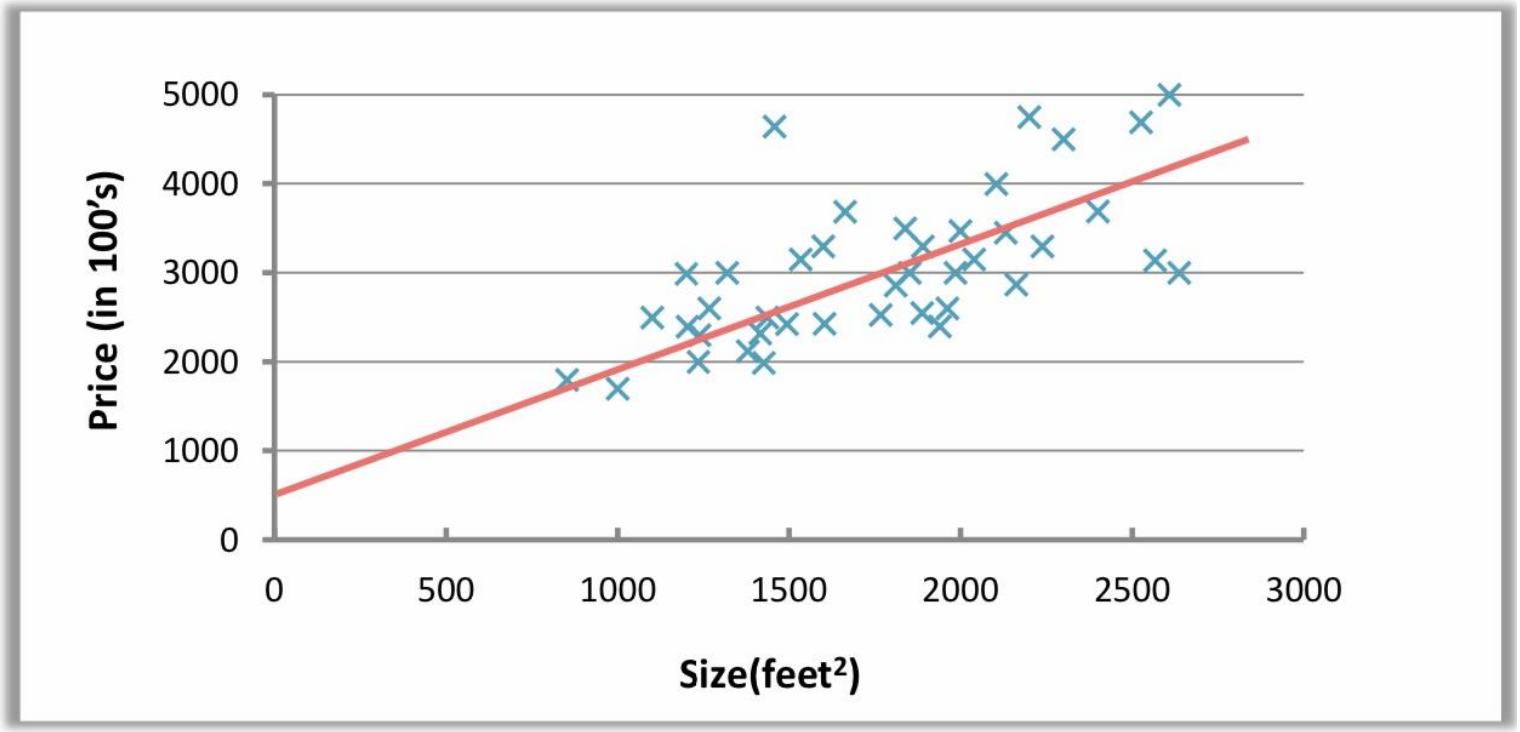
parameters θ_0, θ_1

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

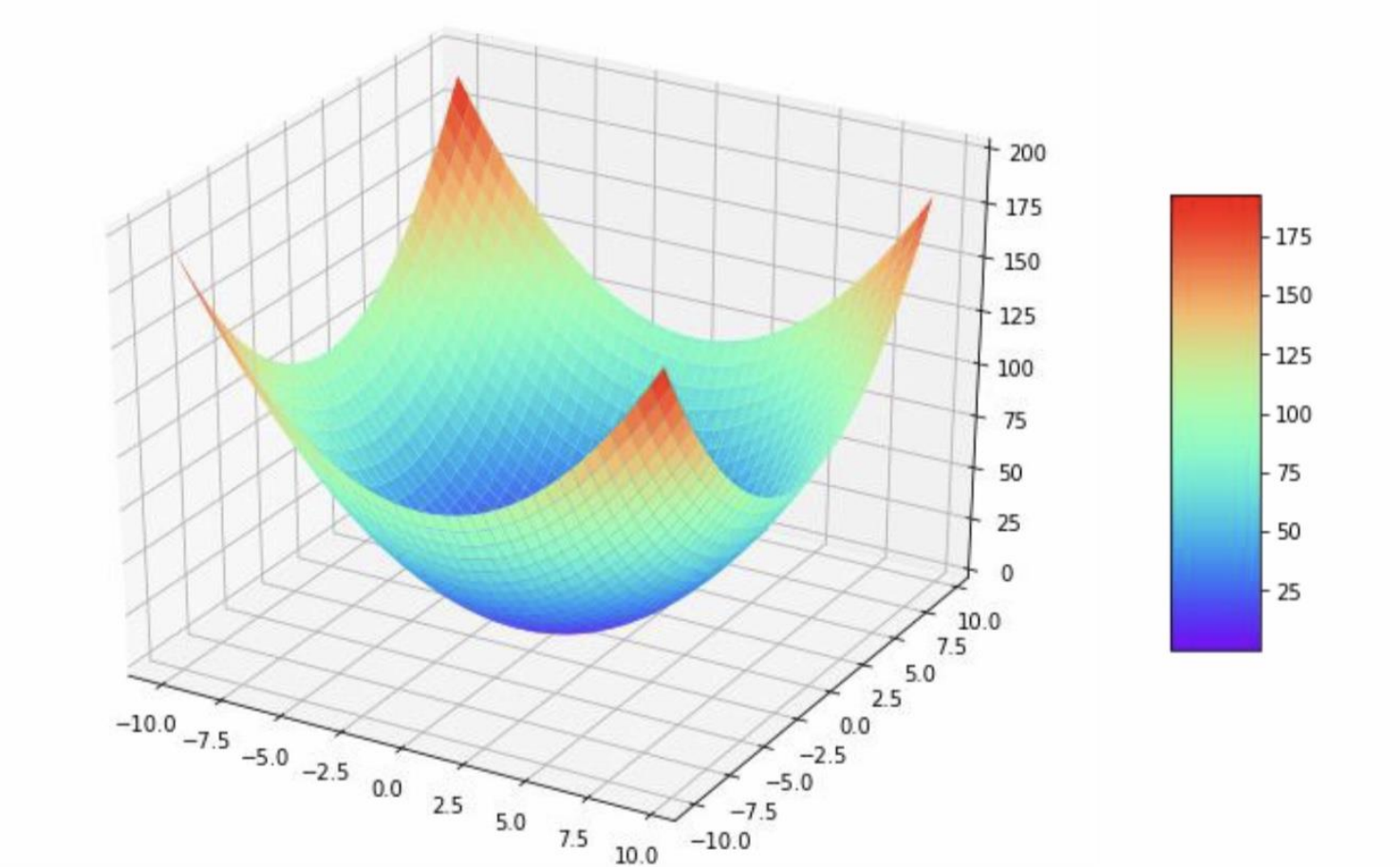
Target $\underset{\theta_0,\theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$

Example: house pricing

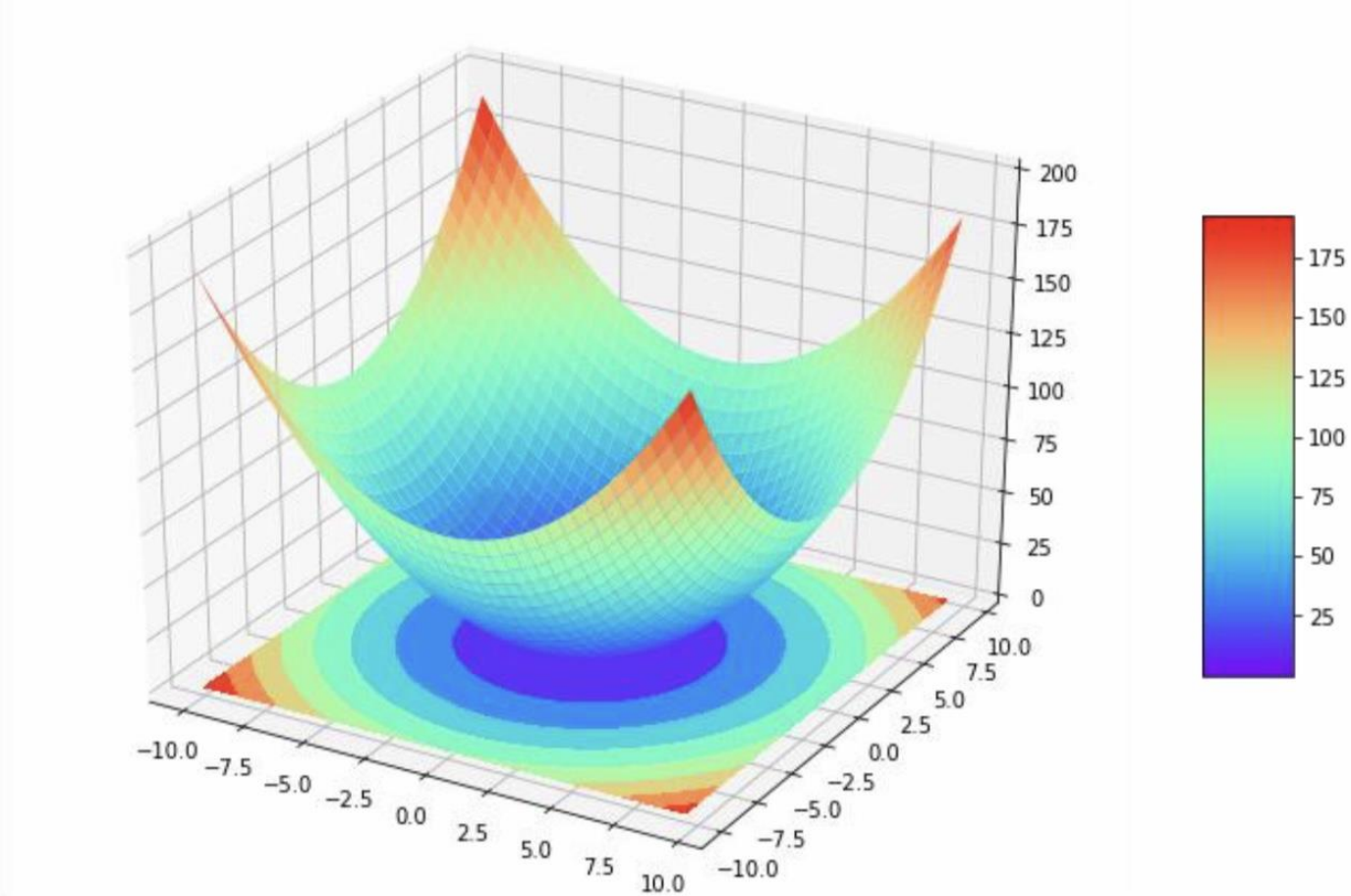
$$h_\theta(x) = \theta_0 + \theta_1 x$$



Cost function

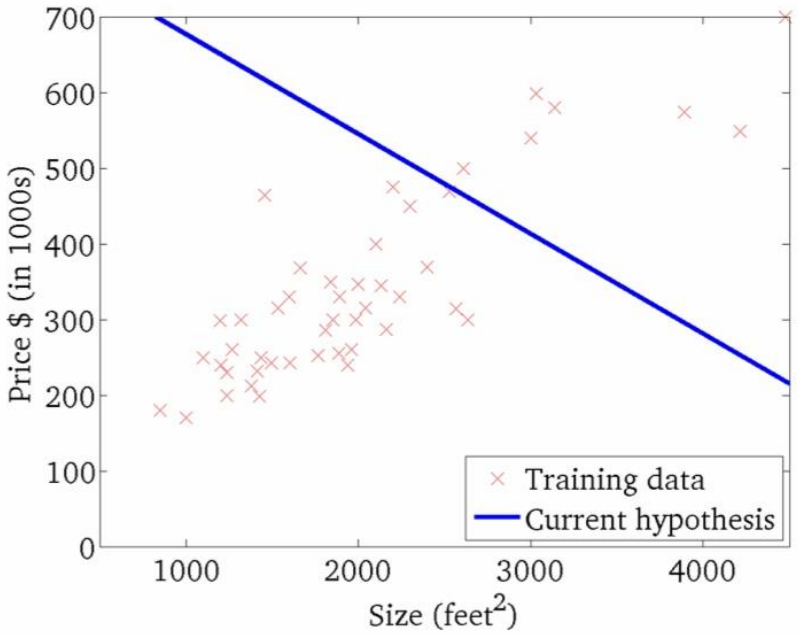


Cost function

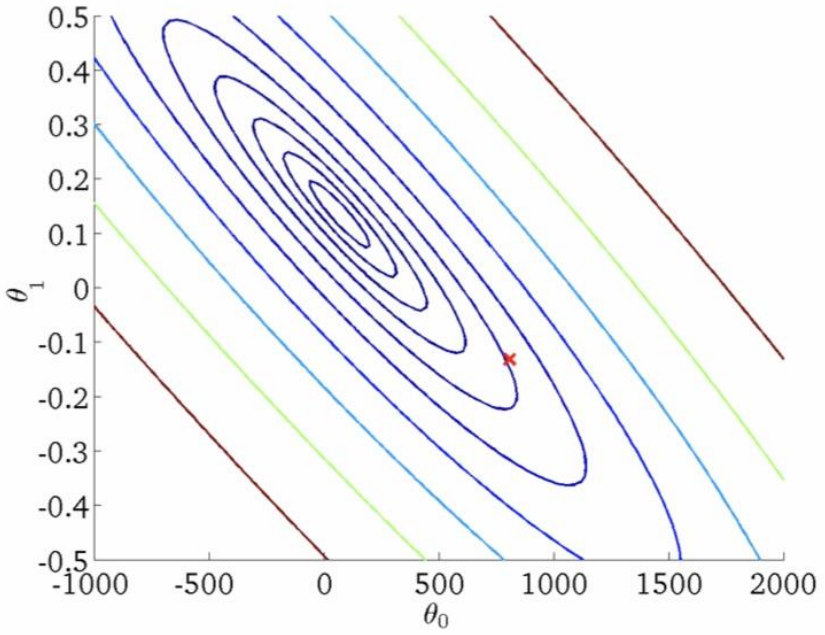


Cost function: Contour curve

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

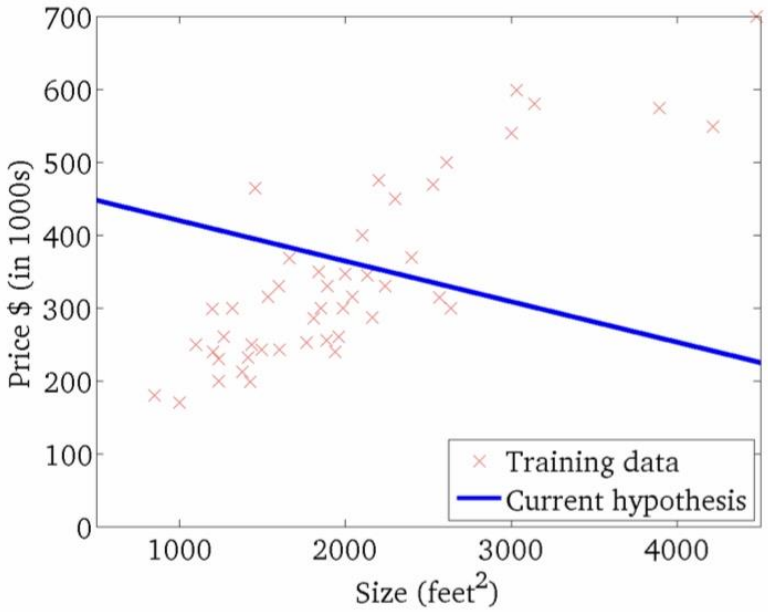


$J(\theta_0, \theta_1)$

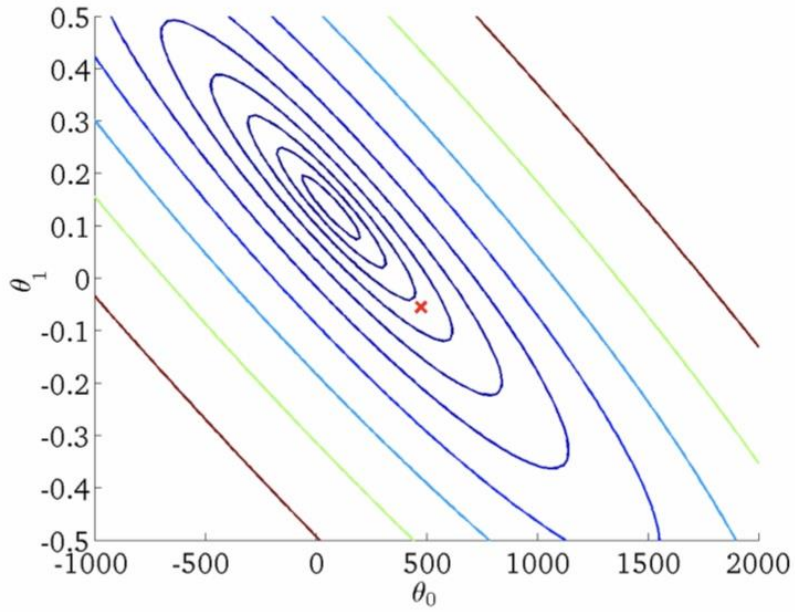


Cost function: Contour curve

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

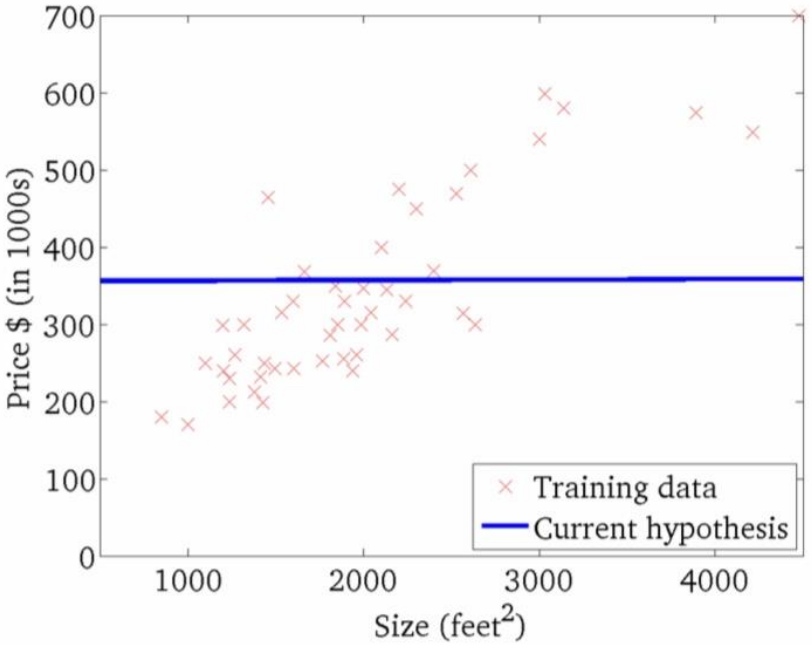


$$J(\theta_0, \theta_1)$$

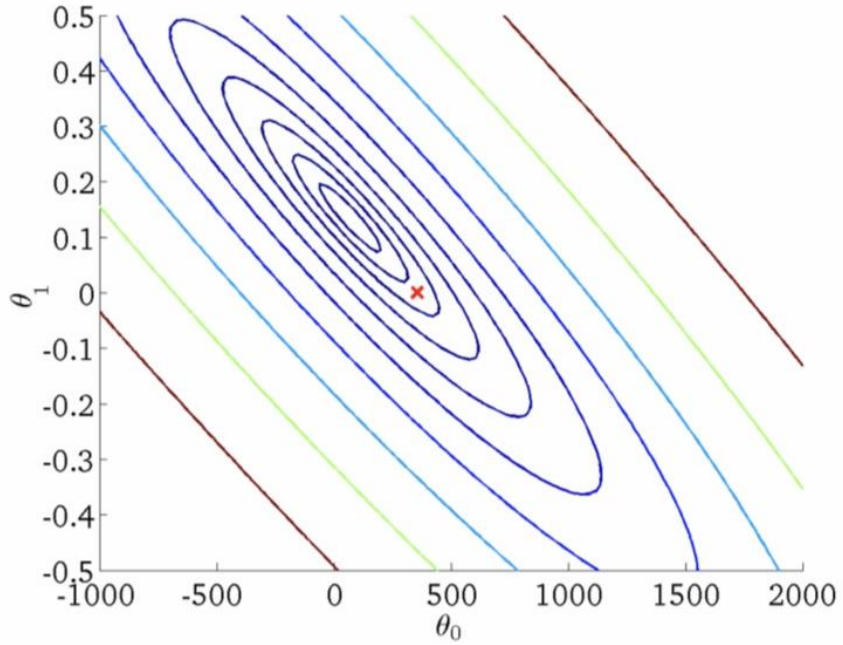


Cost function: Contour curve

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

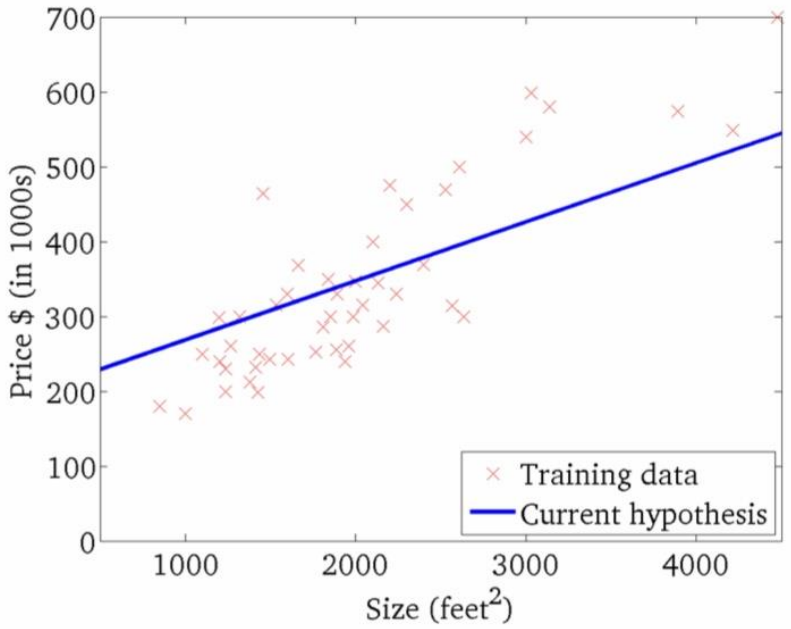


$J(\theta_0, \theta_1)$

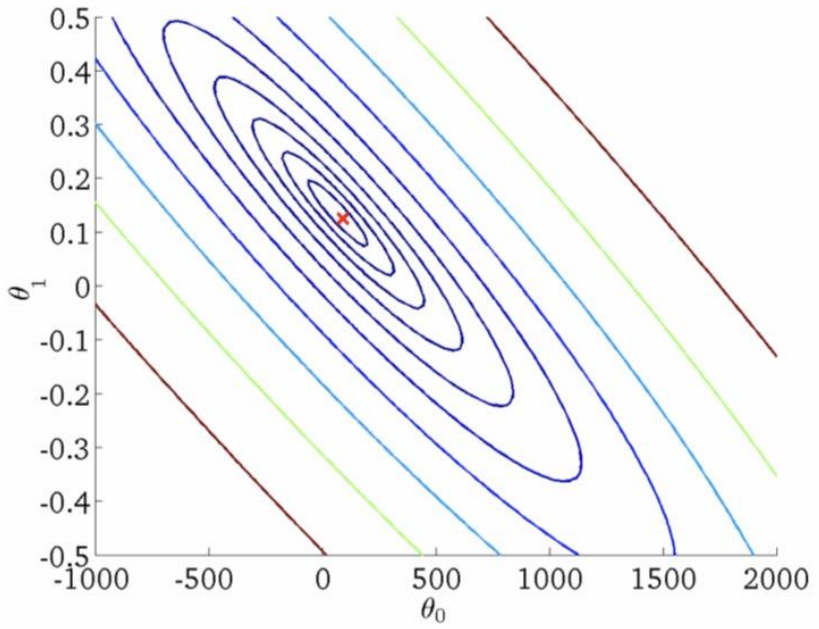


Cost function: Contour curve

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



$J(\theta_0, \theta_1)$



Gradient Descent

Question

- Cost function

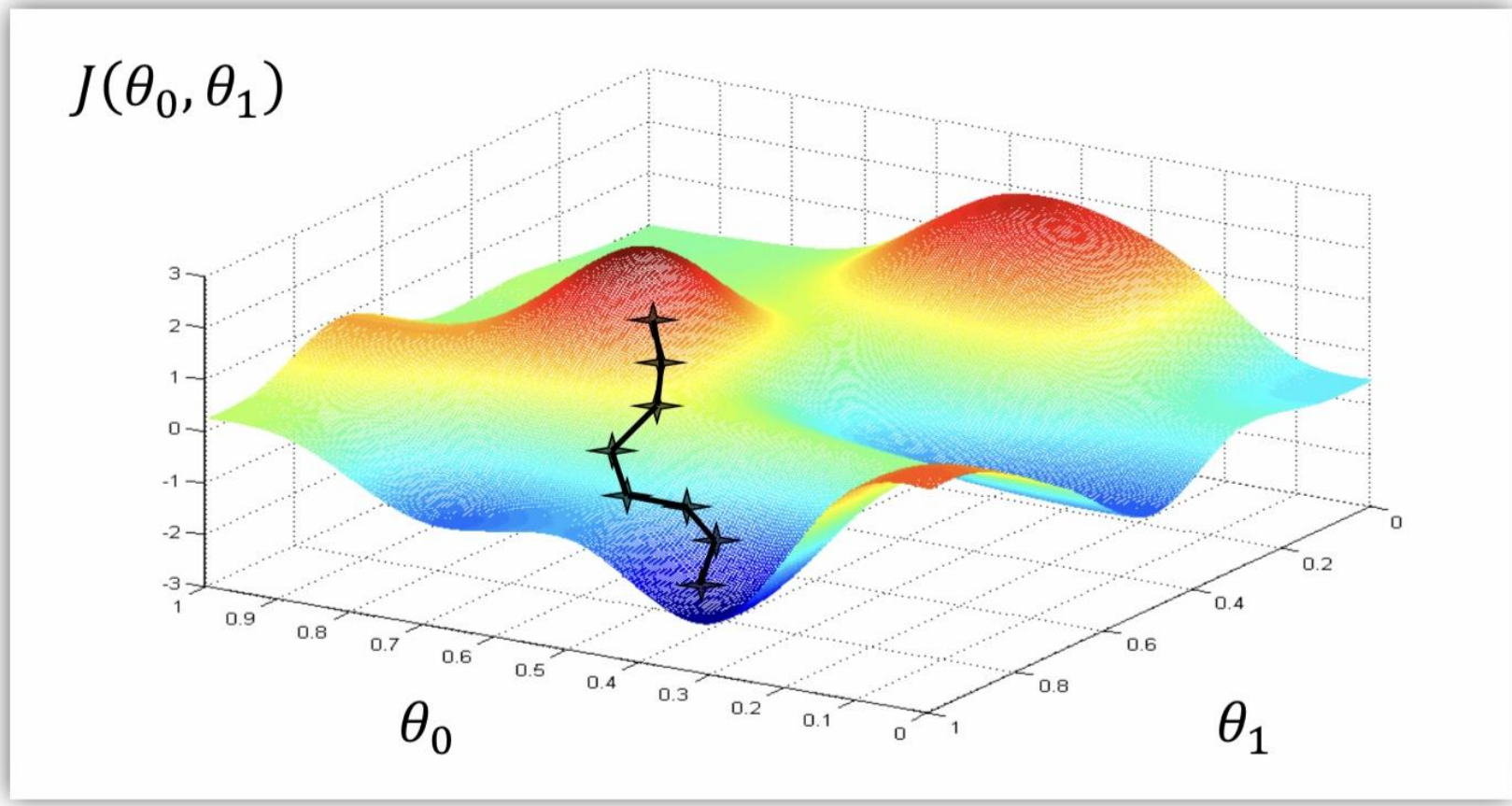
$$J(\theta_0, \theta_1)$$

- Target

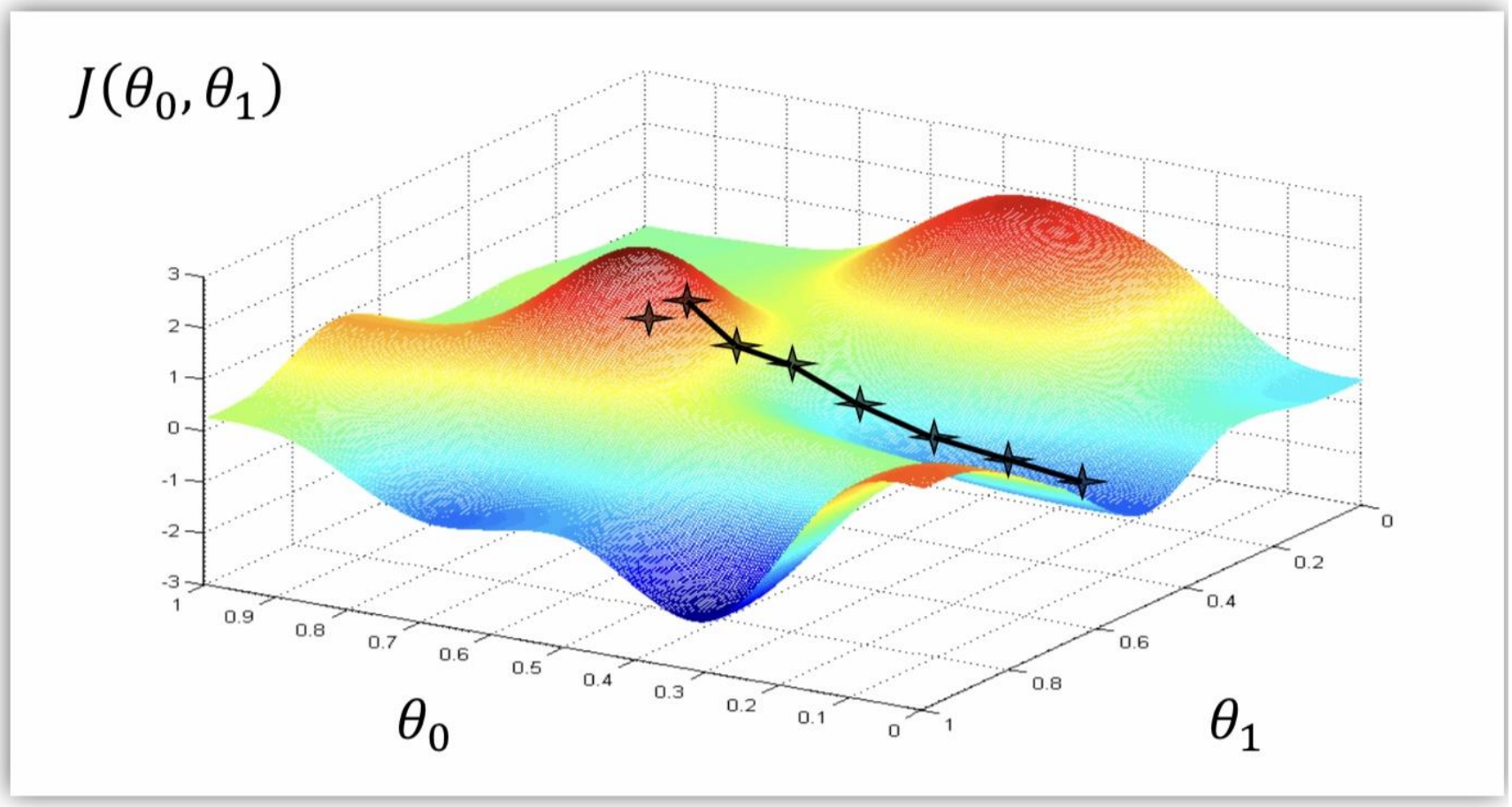
$$\underset{\theta_0, \theta_1}{\text{minimize}} \; J(\theta_0, \theta_1)$$

- General method:
  - Start with a random initial value for parameters θ_0 and θ_1. (for example zero value)
  - Change the value of the parameters in such a way that the value of the cost function $J(\theta_0, \theta_1)$ decreases.
  - Repeat the above operation until we reach a minimum value for the cost function. (convergence)

Gradient Descent: Global Optimum



Gradient Descent: local Optimum



Gradient Descent's algorithm

repeat until convergence {

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

}

Learning rate

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

Correct implementation: Update parameters value simultaneously

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

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

$$\theta_0 := \theta_0 + \Delta\theta_0$$

$$\theta_1 := \theta_1 + \Delta\theta_1$$

Gradient Descent's algorithm

repeat until convergence {

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

}

Learning rate

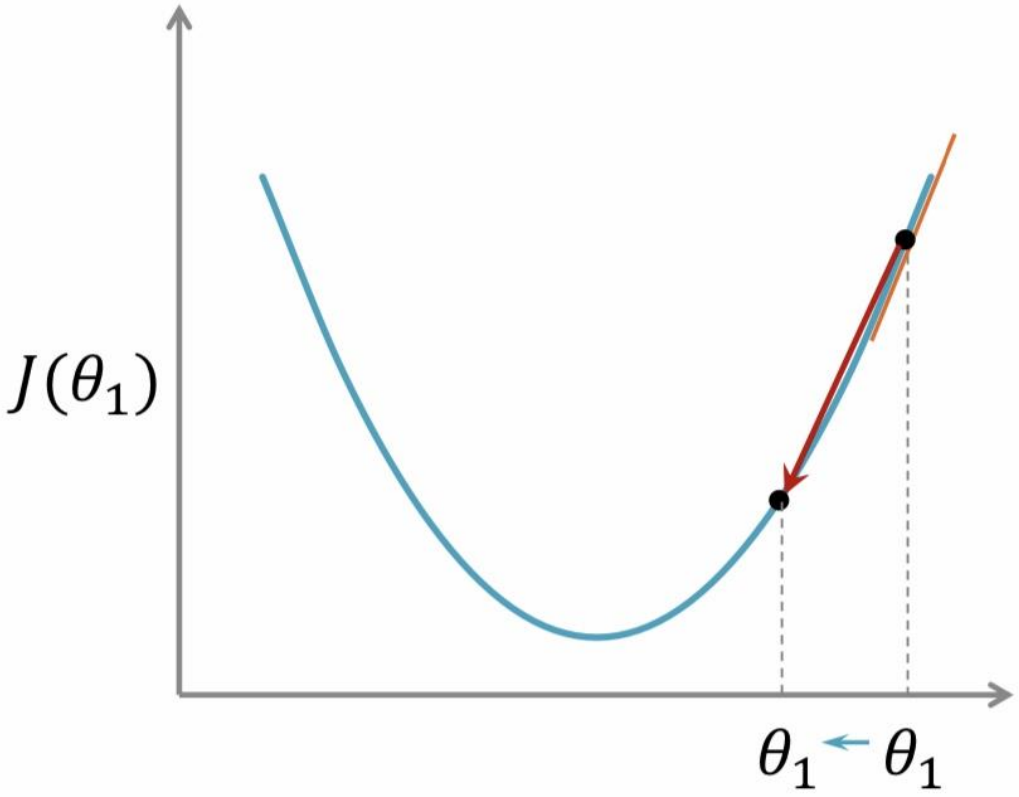
$$h_\theta(x) = \theta_0 + \theta_1 x$$

Incorrect implementation: Update parameters value in order

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

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

Gradient Descent's algorithm: Gradient

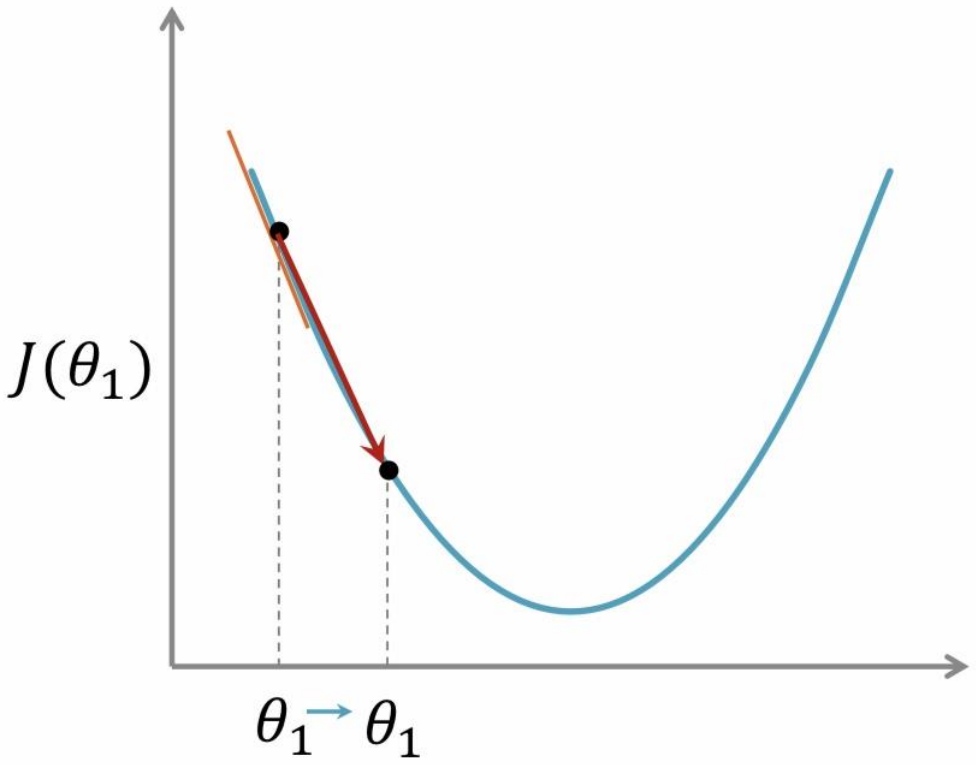


positive slope

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

$$\theta_1 := \theta_1 - \underbrace{\alpha(\geq 0)}_{\geq 0}$$

Gradient Descent's algorithm: Gradient



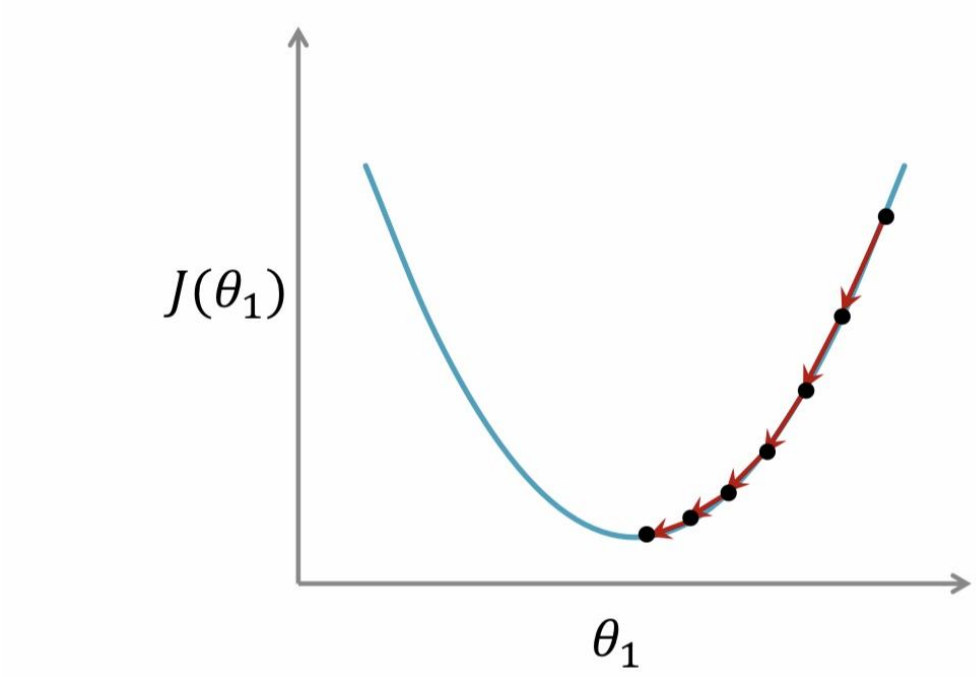
negative slope

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

$$\theta_1 := \theta_1 - \underbrace{\alpha(\leq 0)}_{\leq 0}$$

Gradient Descent's algorithm: Learning Rate

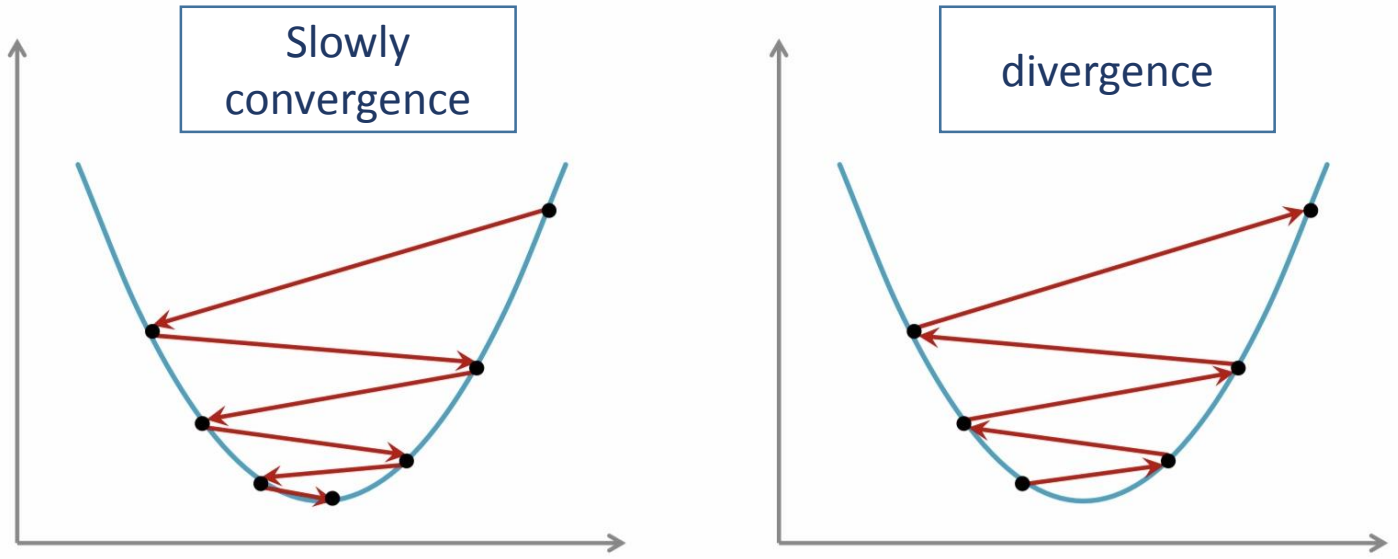
- If the learning rate is too small, the gradient descent will converge slowly.



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

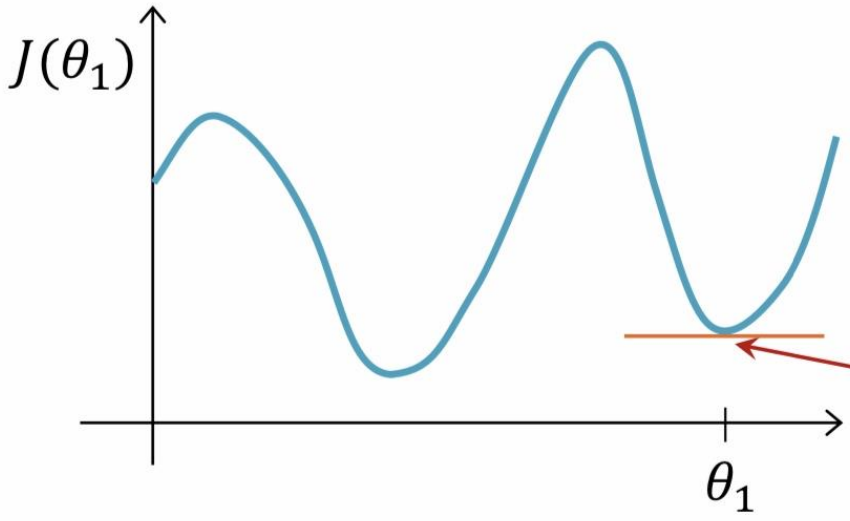
Gradient Descent's algorithm: Learning Rate

- If the learning rate is too large, the gradient descent may converge slowly or even diverge.



Gradient Descent's algorithm: Convergence

- Convergence When the value of parameter θ_1 is in a local minimum.



local minimum

$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} J(\theta_1)}_{\text{zero}}$$

Application of gradient descent in linear regression

Gradient Descent and Linear Regression

repeat until convergence {

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

}

Linear Regression

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

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

Cost Function Gradient

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

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

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

Gradient Descent and Linear Regression

repeat until convergence {

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

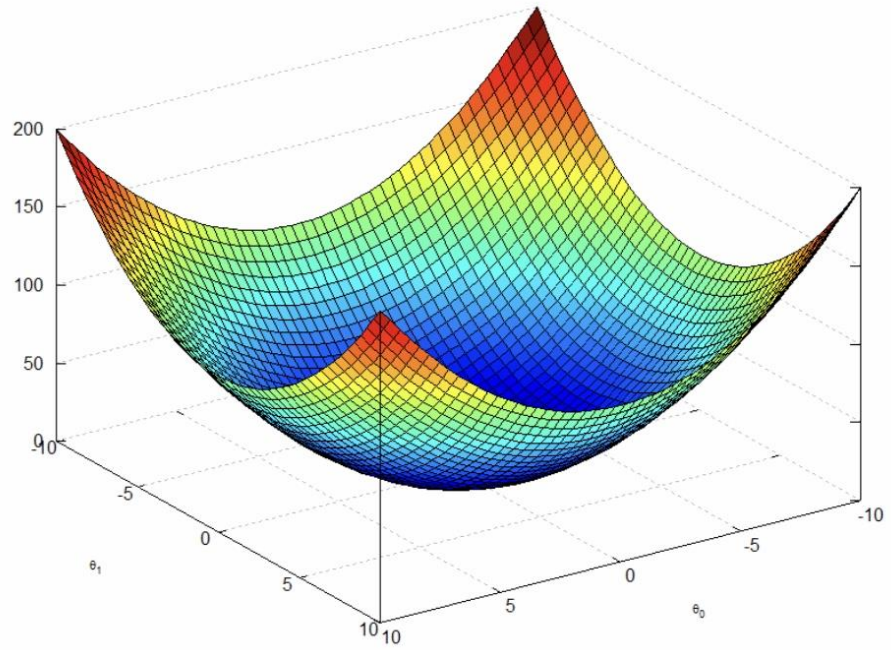
$$\theta_1 := \theta_1 - \alpha \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) \cdot x^{(i)}$$

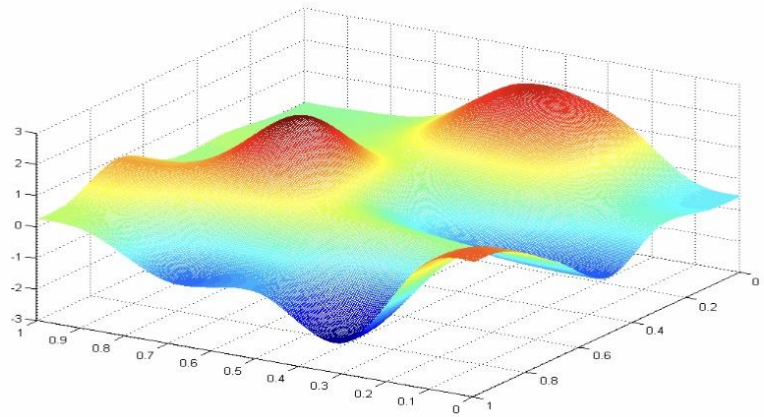
}

Simultaneous update

Gradient Descent and Linear Regression

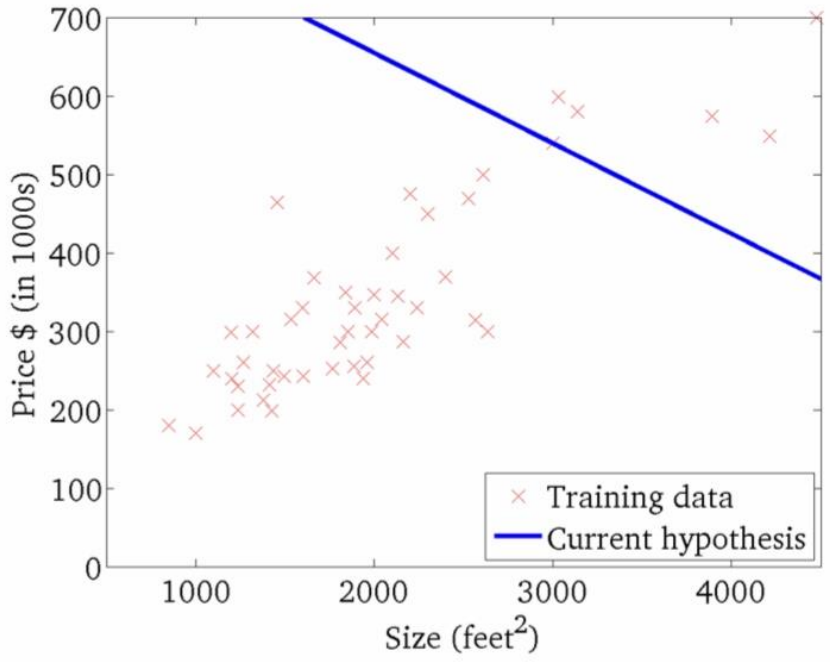
- Note: In linear regression, the cost function is a cognate function, and as a result, the gradient descent necessarily converges to the global optimum in case of convergence.



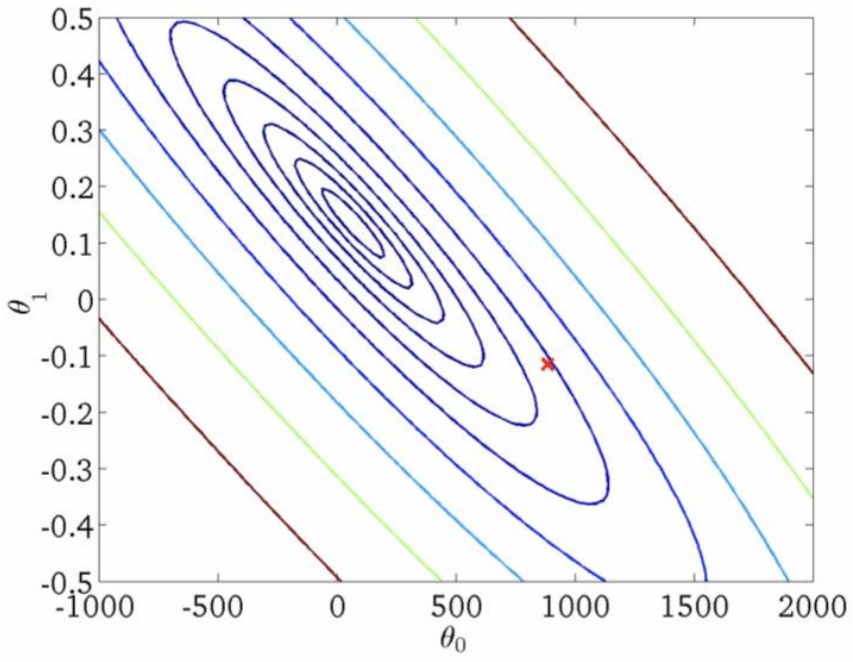


Gradient Descent

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

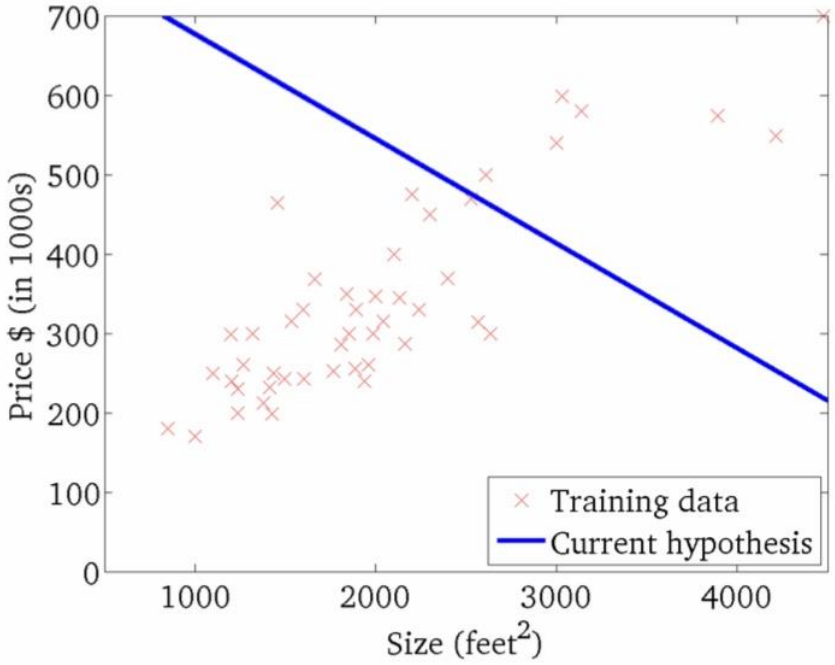


$$J(\theta_0, \theta_1)$$

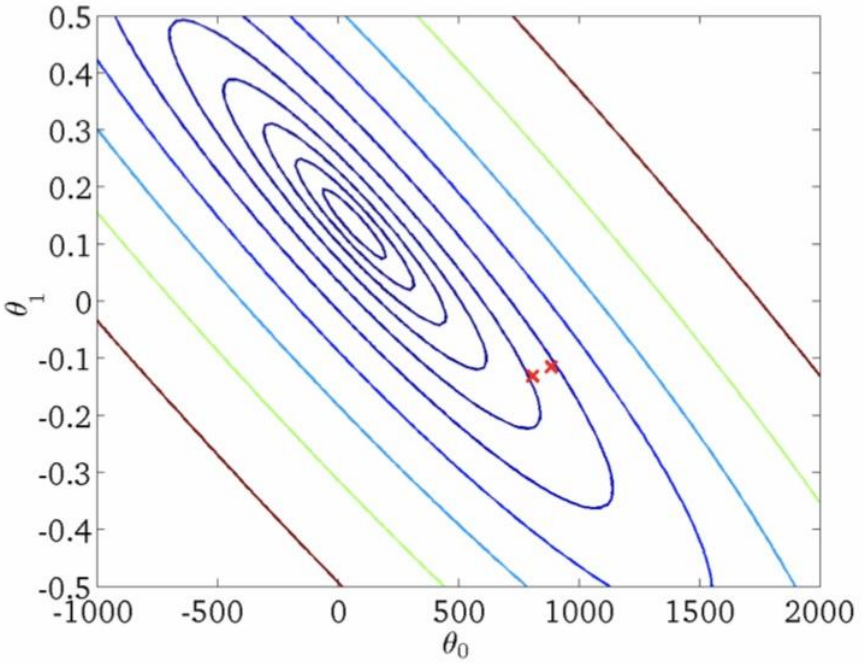


Gradient Descent

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

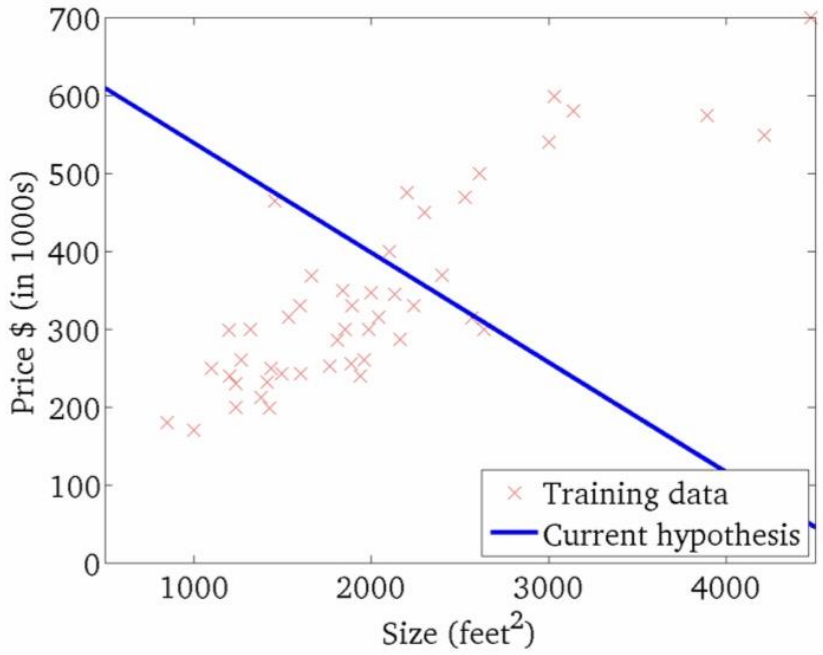


$$J(\theta_0, \theta_1)$$

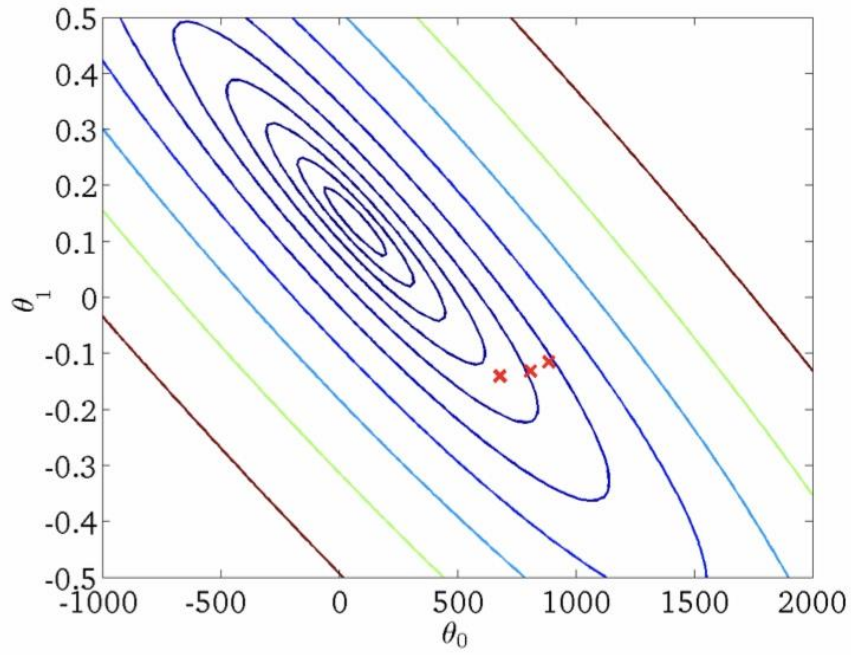


Gradient Descent

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

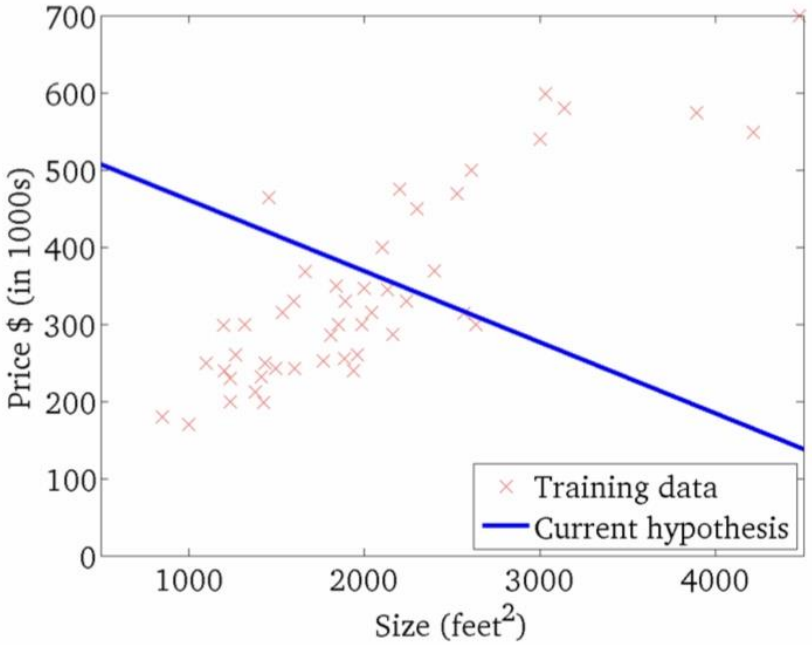


$$J(\theta_0, \theta_1)$$

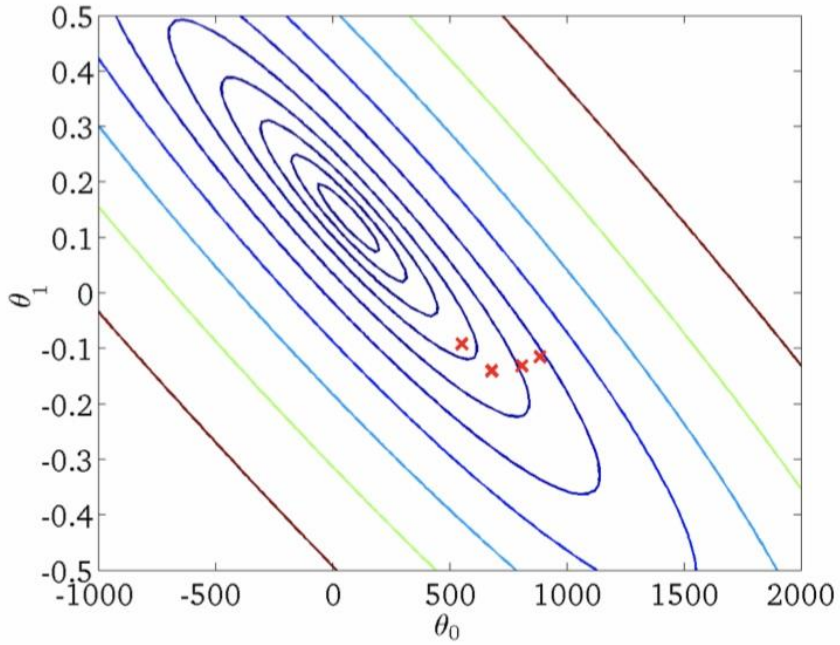


Gradient Descent

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

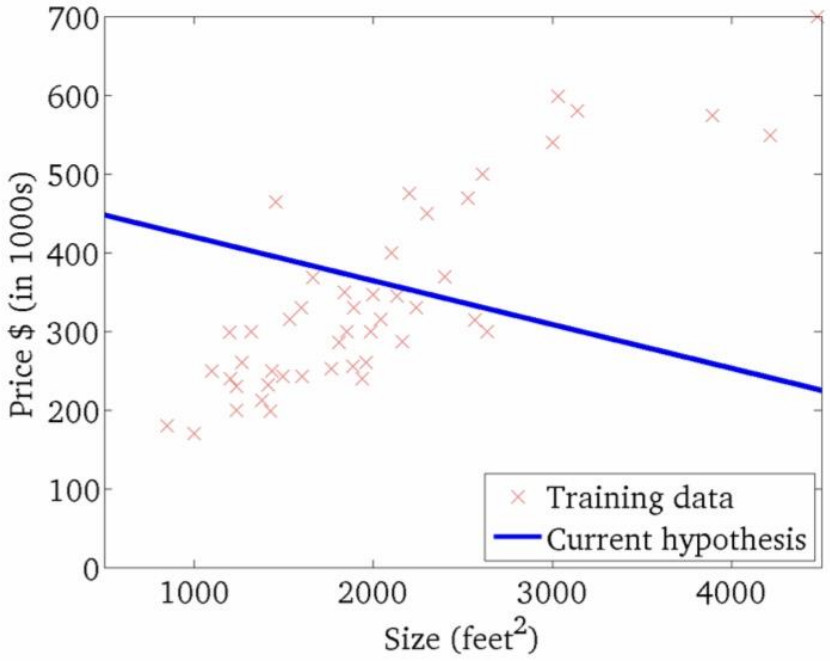


$$J(\theta_0, \theta_1)$$

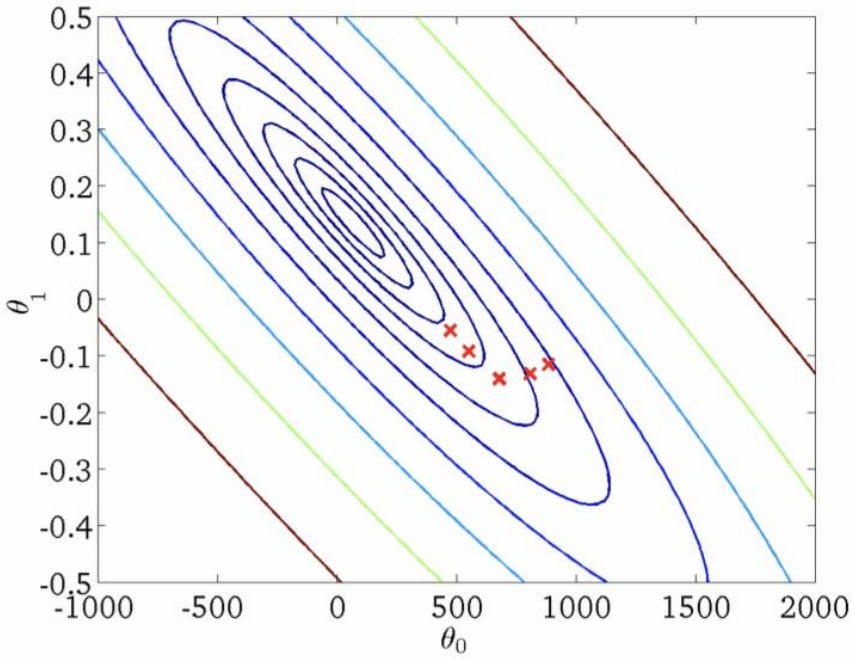


Gradient Descent

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

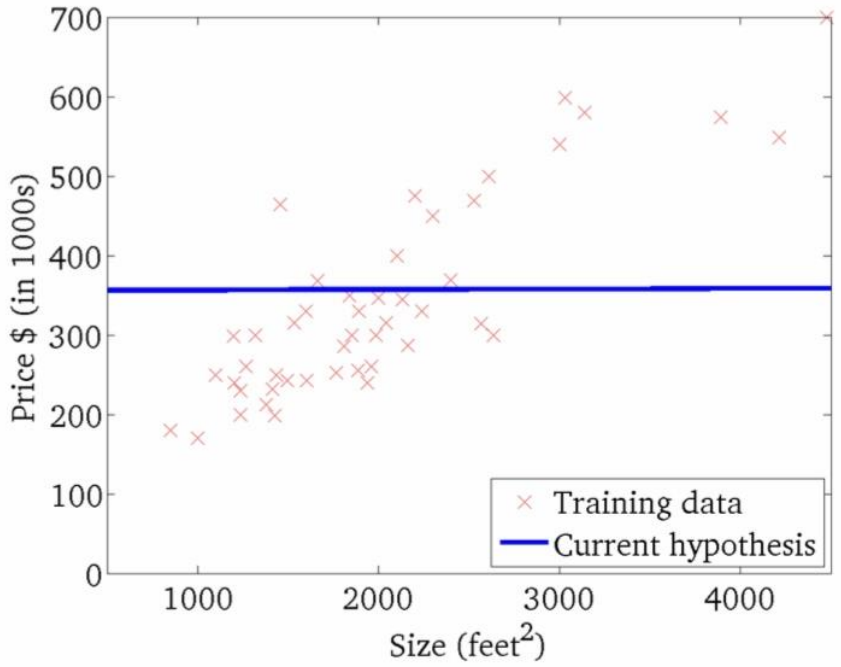


$$J(\theta_0, \theta_1)$$

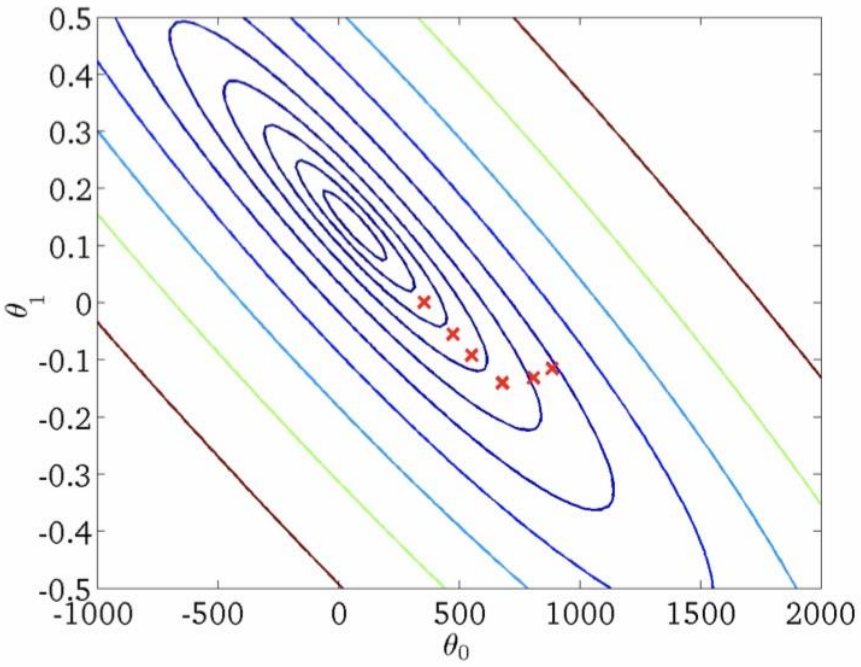


Gradient Descent

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

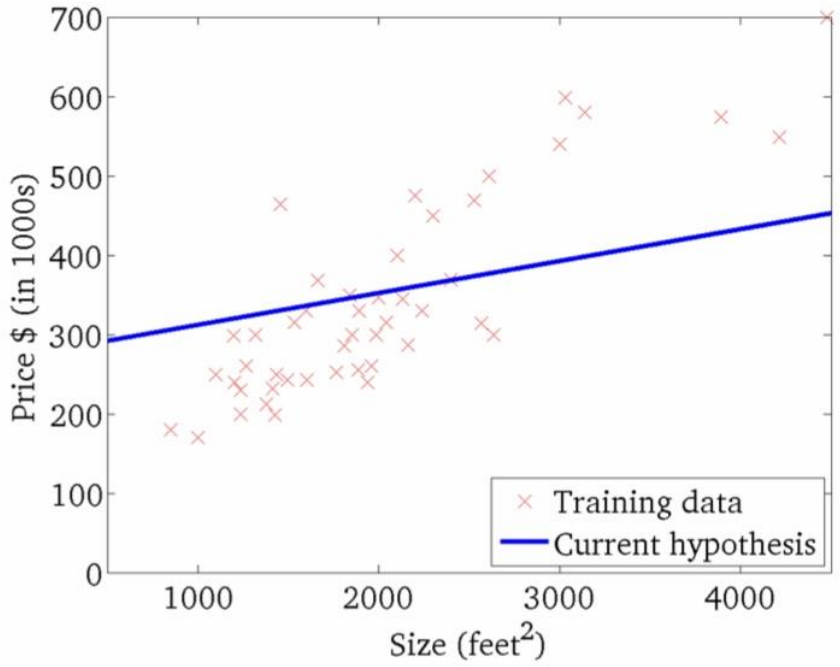


$$J(\theta_0, \theta_1)$$

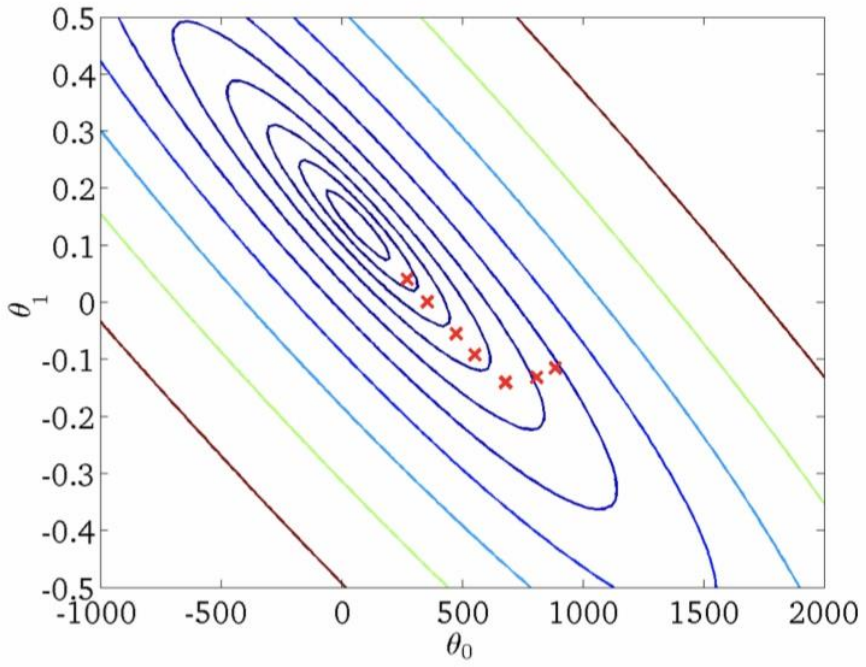


Gradient Descent

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

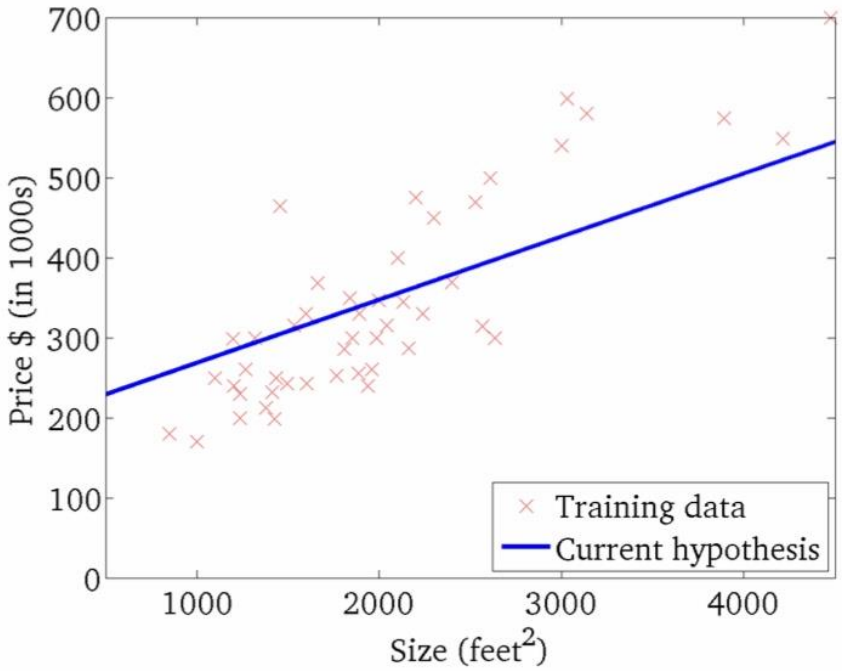


$$J(\theta_0, \theta_1)$$

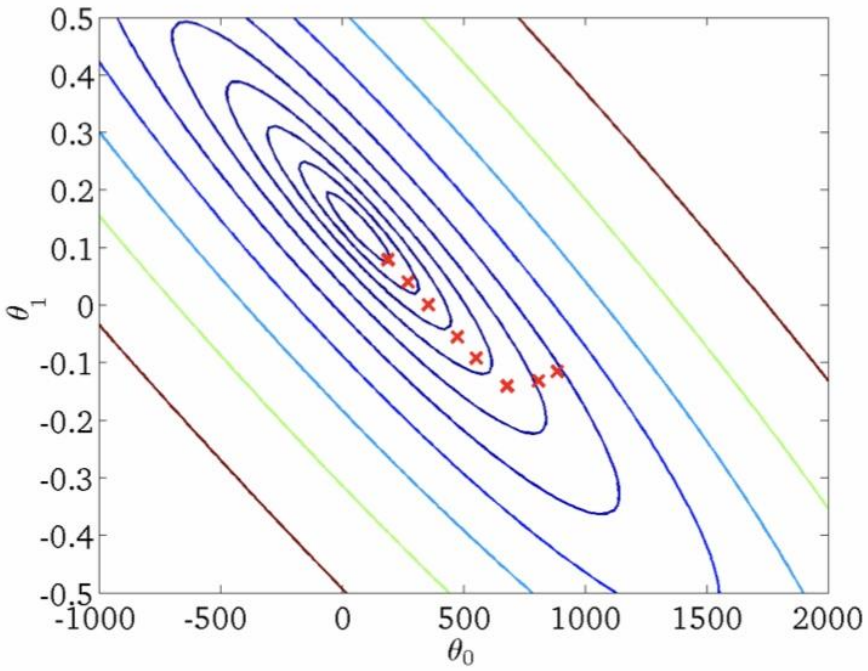


Gradient Descent

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

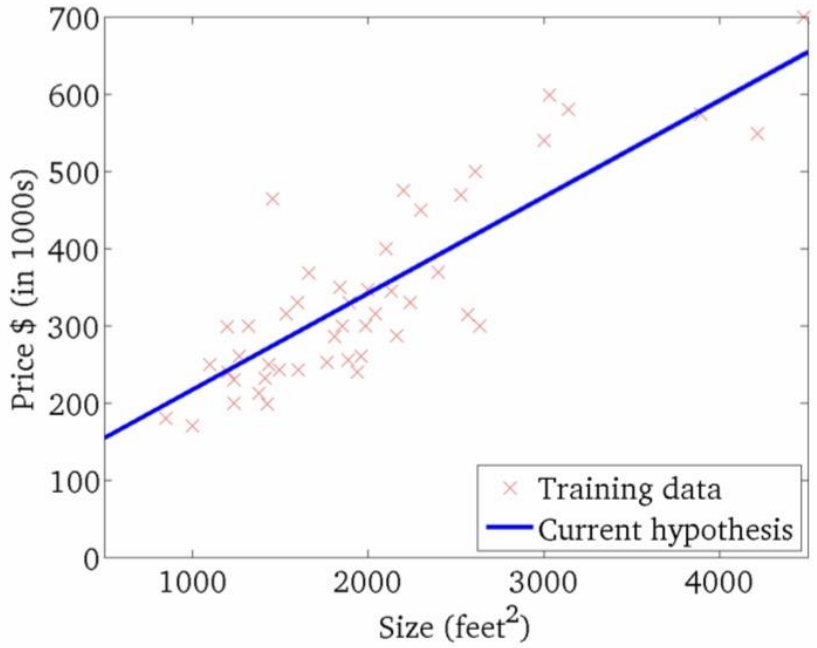


$$J(\theta_0, \theta_1)$$

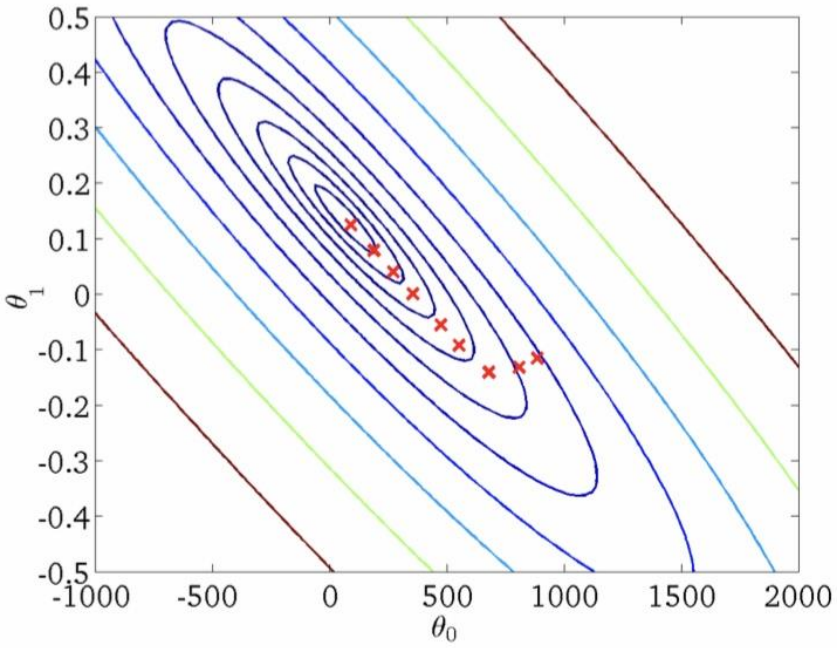


Gradient Descent

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



$$J(\theta_0, \theta_1)$$



Batch Gradient Descent

- Batch gradient descent. In each iteration of the algorithm, all the training samples are used to update the parameter values.

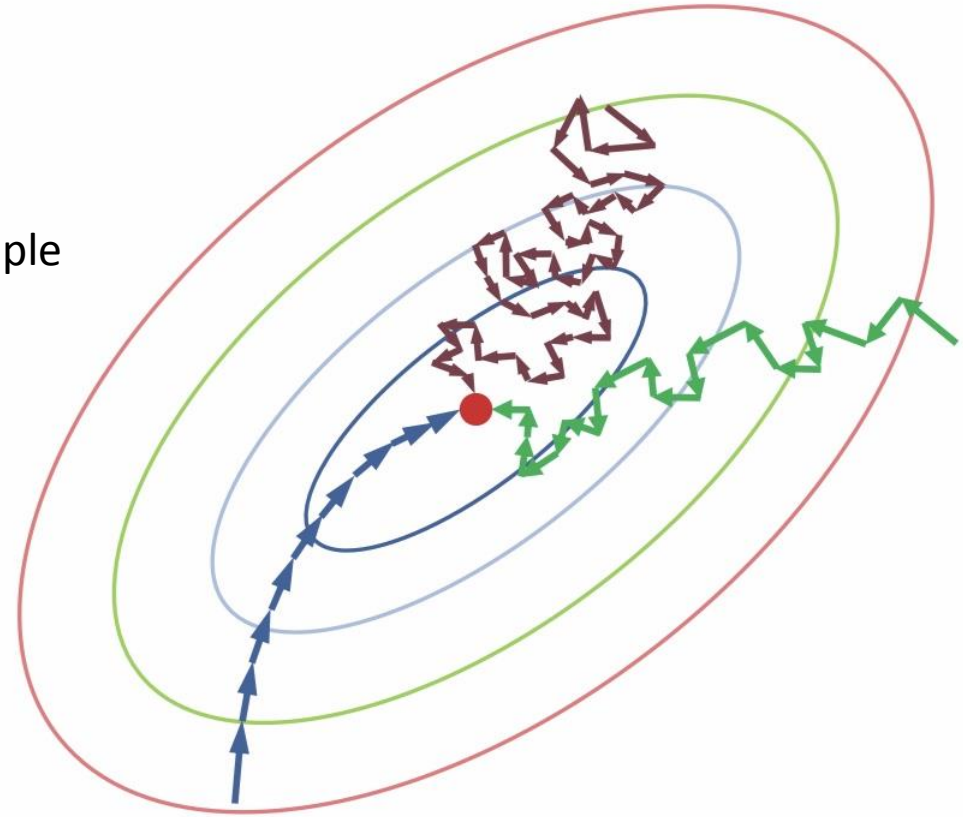
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad (\text{for } j = 0 \text{ and } j = 1)$$

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

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

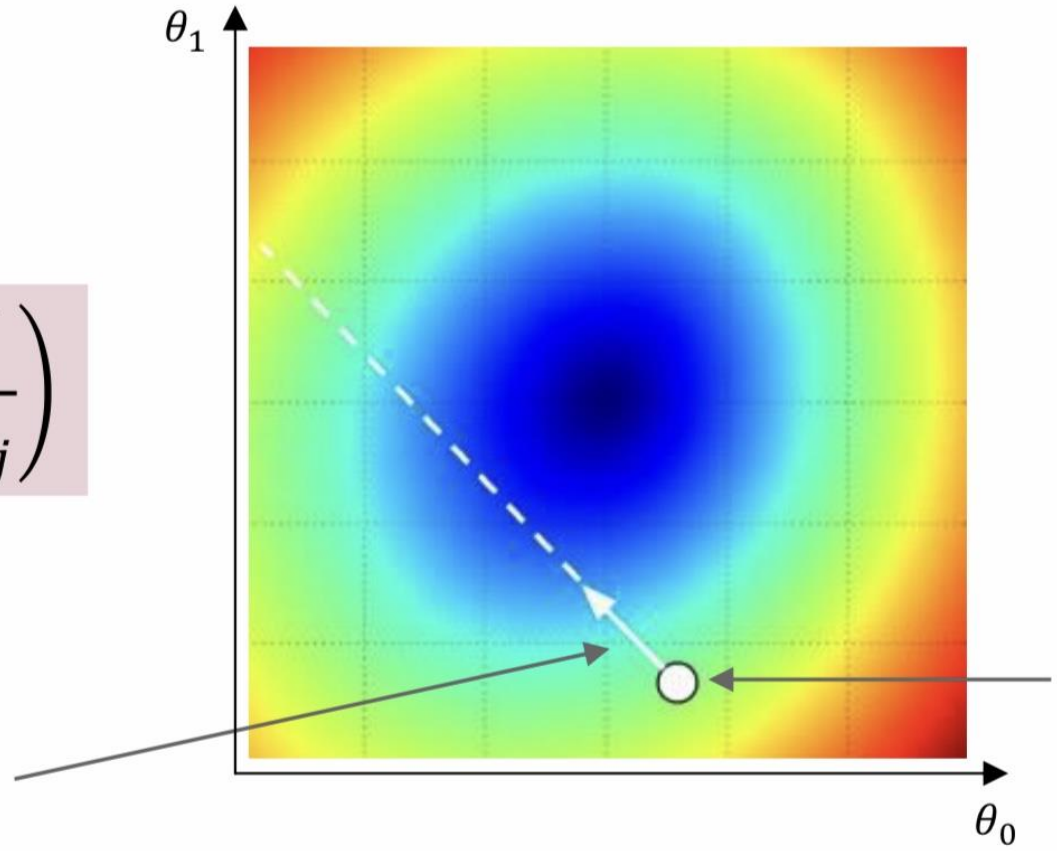
Types of Gradient Descent

----- Gradient descent with full batches
Update in each iteration using all samples
----- Random gradient descent (online)
Update in each iteration using a random sample
----- A descending gradient with small batches
Update in each iteration using a random subset of samples



Reminder: Gradient Descent Algorithm

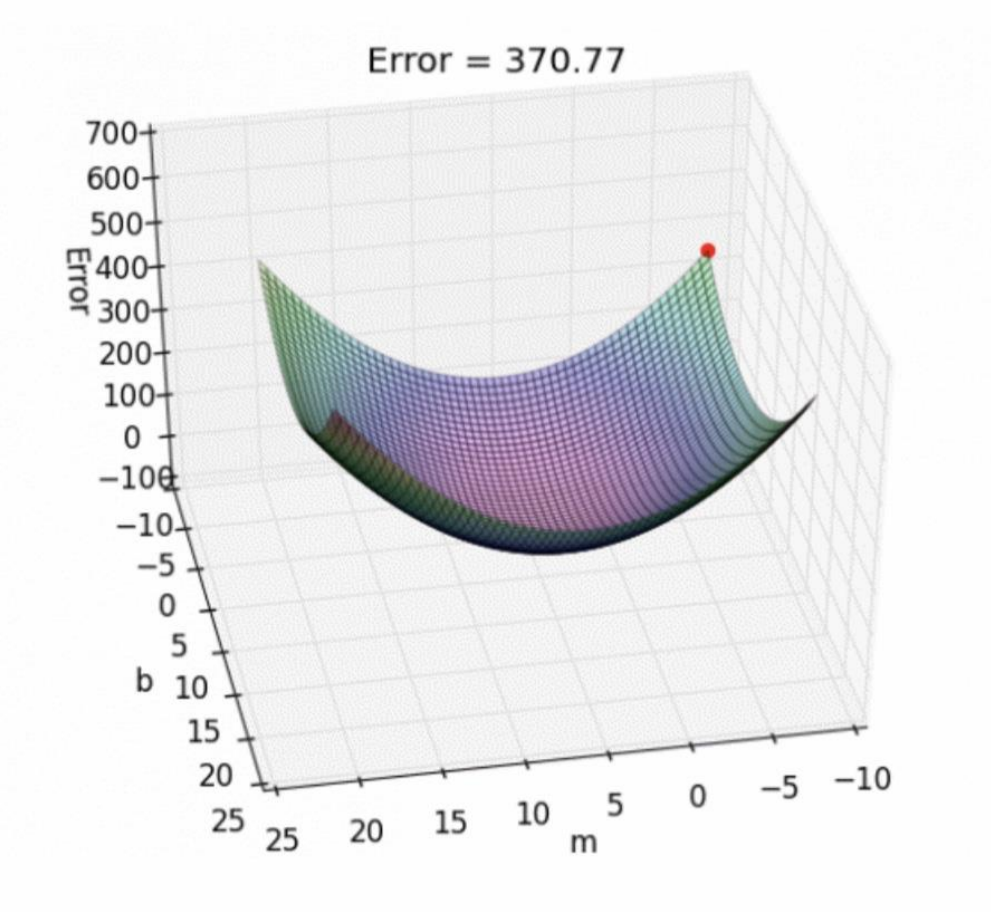
$$\theta_j = \theta_j - \alpha \left(\frac{\partial J}{\partial \theta_j} \right)$$

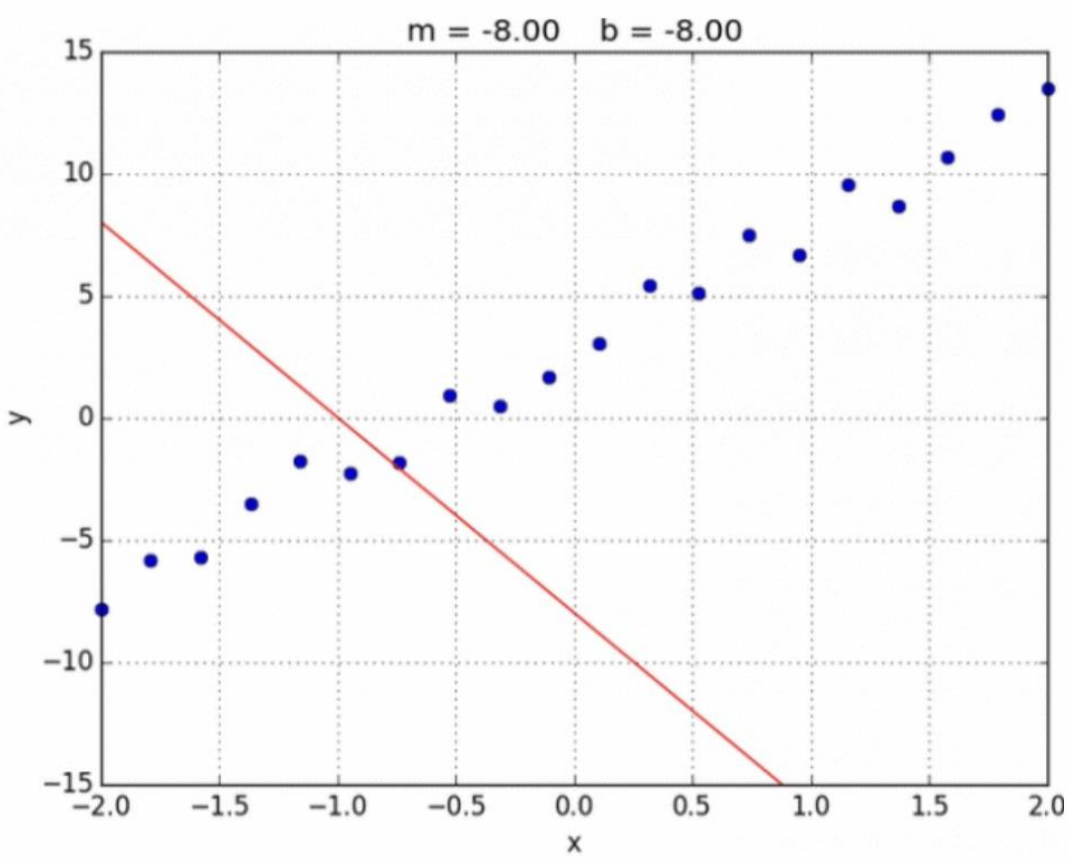


Initial value of θ

against the direction of the gradient

Reminder: Gradient Descent Algorithm





Multivariate linear regression

Univariate linear regression (one feature)

Linear regression with one feature.

Price (1000 dollars) y	Meterage (square foot) x
460	2104
232	1416
315	1534
178	852
…	…

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

Multivariate linear regression (more than one feature)

- Linear regression with more than one feature.

Price (1000 dollars) y	House age (year) x_4	Number of levels x_3	Number of bedrooms x_2	Meterage (square foot) x_1
460	45	1	5	2104
232	40	2	3	1416
315	30	2	3	1534
178	36	1	2	852
…	…	….	…	….

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

Multivariate linear regression (more than one feature)

Price (1000 dollars) y	House age (year) x_4	Number of levels x_3	Number of bedrooms x_2	Meterage (square foot) x_1
460	45	1	5	2104
232	40	2	3	1416
315	30	2	3	1534
178	36	1	2	852
…	…	….	…	….

The number of training examples

m

n

Symbols:
n: number of features
$x^{(i)}$: inputs in i th training example
$x^{(i)}_j$: the value of j th feature in i th training example

$$x^{(2)} = \begin{bmatrix} 1416 \\ 3 \\ 2 \\ 40 \end{bmatrix} \leftarrow x_1^{(2)} \quad \leftarrow x_3^{(2)}$$

Hypothesis

Univariate linear regression

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

Multivariate linear regression

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

For simplicity, we define $x_0 = 1$

$$x = \begin{bmatrix} x_0 = 1 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix} \quad \Rightarrow \quad h_\theta(x) = \theta^T x$$

Gradient Descent in Multivariate Linear Regression

Gradient Descent

hypothesis

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

parameters

$$\theta = (\theta_0, \theta_1, \ldots, \theta_n)$$

cost dependant

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

repeat until convergence {

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

}

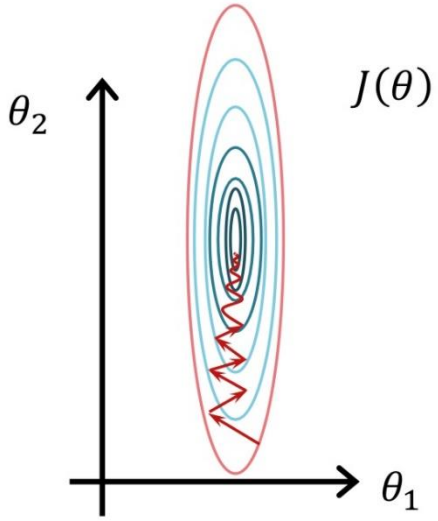
$$\frac{\partial}{\partial \theta_j} J(\theta) = \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) \cdot x_j^{(i)}$$

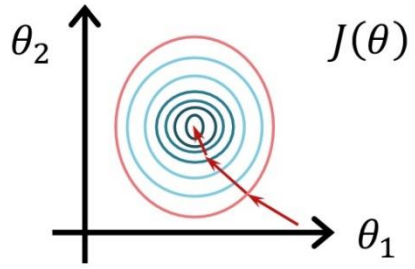
Practical tricks in multivariate regression

- Scaling features
- Determine the learning rate

Scaling features (normalization)

- Idea: Ensuring that feature values are on the same scale.
- Objective: increasing the speed of convergence in decreasing gradient.
- Example:
  - x_1: house size (0 to 2000)
  - x_2: Number of bedrooms (1 to 5)





Scaling features

- Scaling: The value of each attribute is in a small coefficient from -1 to +1.

Example:

$$x_1 = \frac{size - 1000}{2000} \qquad -0.5 \leq x_1 \leq 0.5$$

$$x_2 = \frac{\#\ bedrooms - 2}{5} \qquad -0.5 \leq x_2 \leq 0.5$$

Average normalization:

$$x_j = \frac{x_j - \mu_j}{\sigma_j}$$

μ_j → mean

σ_j → Standard deviation

Gradient Descent

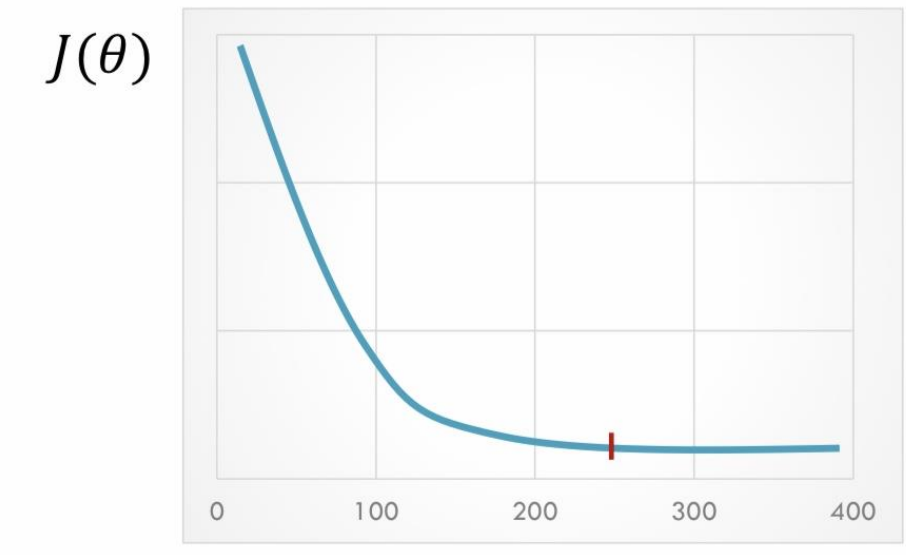
- Gradient Descent

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

- Question: How can you make sure that the decreasing gradient works correctly?
- Question: What is the right value for the learning rate?

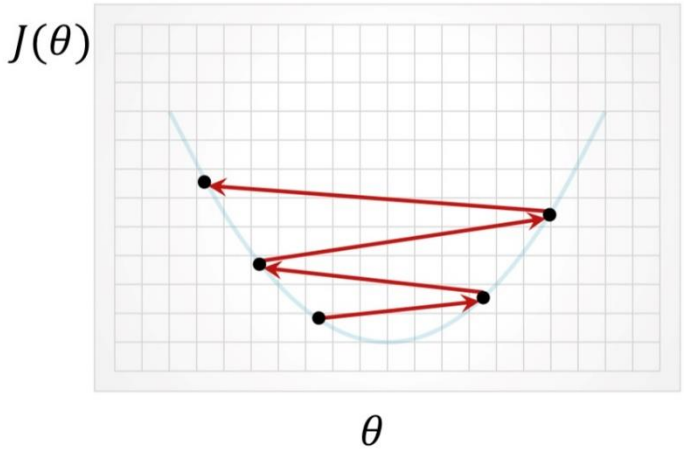
Correct Performance for Gradient Descent

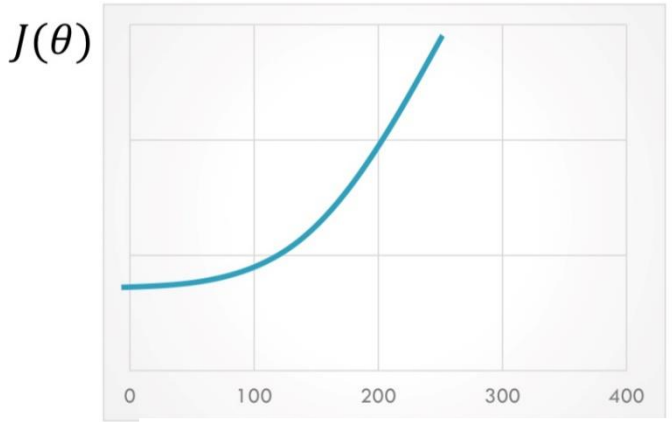
- Convergence test:
- If the value of J(θ) changes by less than 10^{-3} in one iteration, convergence has occurred.



The Signs of Incorrect Performance of Gradient Descent

- Lack of convergence:
- Solution: Use smaller values for the learning rate, but if the learning rate is too small the convergence will be very slow.





Summary

- Learning rate
  - Very small: Very slow convergence
  - Very large: slow convergence or no convergence
- Choice of learning rate
  - In order to choose a suitable value for the learning rate, try the following values:

...,0.001,0.003, 0.01, 0.03, 1.0, ...

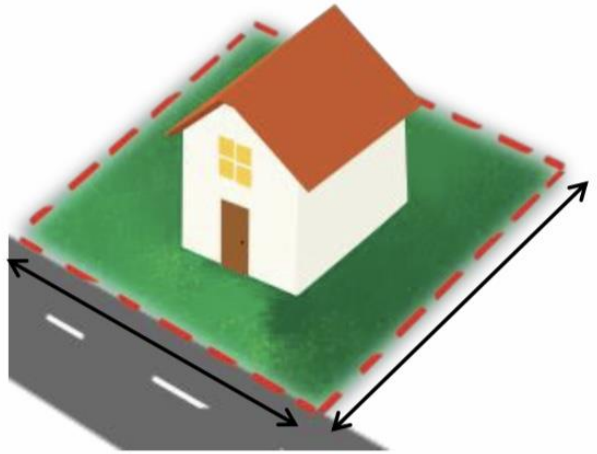
Polynomial regression

Pricing a House: Feature Selection

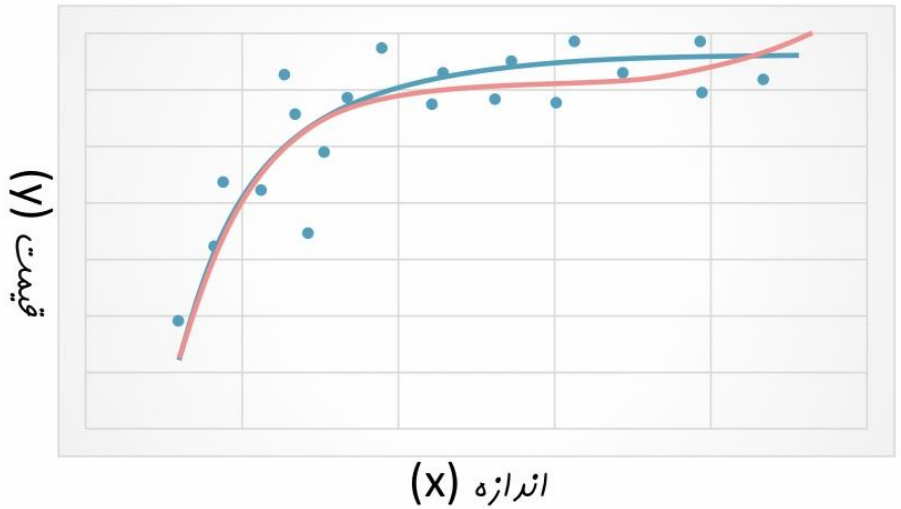
$$h_\theta(x) = \theta_0 + \theta_1 \times \underbrace{(\text{frontage})}_{x_1} + \theta_2 \times \underbrace{(\text{depth})}_{x_2}$$

$$Area = frontage \times depth$$

$$h_\theta(x) = \theta_0 + \theta_1 \times (\text{Area})$$



Polynomial regression



Polynomial regression

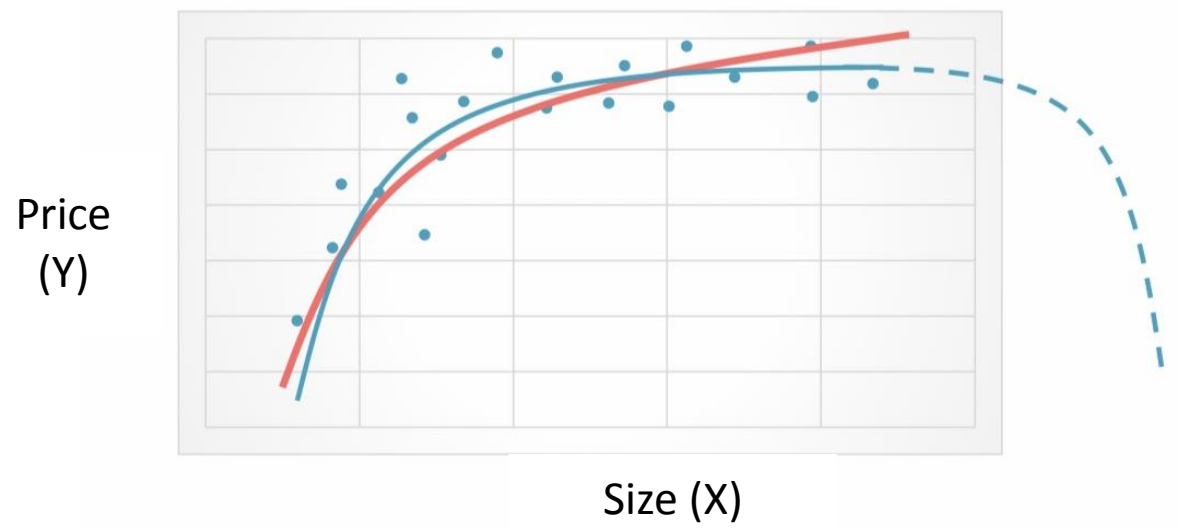
$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

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

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

$$\begin{aligned} x_1 &= (Size)^1 & 1 \le x_1 \le 10^3 \\ x_2 &= (Size)^2 & 1 \le x_2 \le 10^6 \\ x_3 &= (Size)^3 & 1 \le x_3 \le 10^9 \end{aligned}$$

Feature Selection

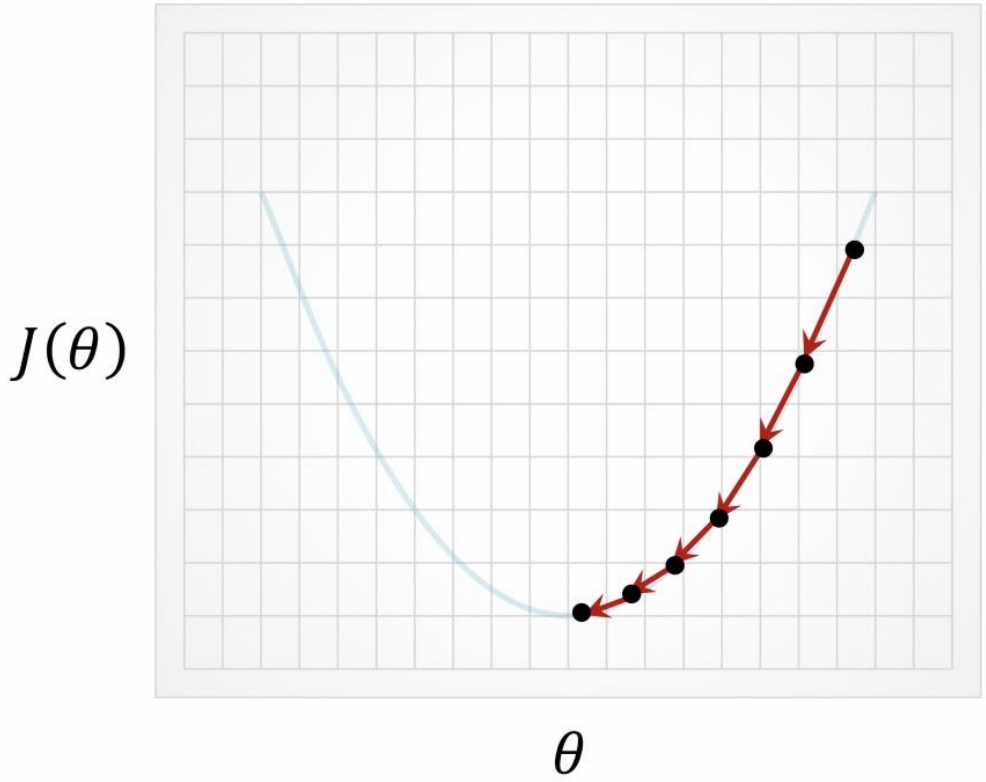


$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2$$

$$h_\theta(x) = \theta_0 + \theta_1(size) + \theta_2\left(\sqrt{size}\right)$$

Linear Regression: Normal Equation

Gradient Descent and Normal Equation



Gradient Descent

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

Updating rule

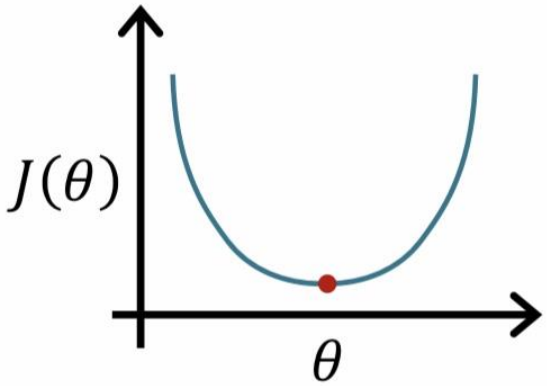
Normal equation: an analytical method to determine the value of parameters.

Normal Equation

Normal Equation

$\theta \in \mathbb{R}$: $J(\theta) = a\theta^2 + b\theta + c$

$\frac{d}{d\theta} J(\theta) \stackrel{\text{def}}{=} 0$



$\theta \in \mathbb{R}^{n+1}$: $J(\theta_0, \theta_1, \ldots, \theta_n) = \frac{1}{2} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$

$\frac{\partial}{\partial \theta_j} J(\theta) \stackrel{\text{def}}{=} 0$ $\quad (j = 0,1,2, \ldots, n)$

Normal Equation : example (m = 4)

x_0	Size (feet^2) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$X\theta = y$$

Normal Equation : example (m = 5)

x_0	Size (feet²) x_1	Number of bedrooms x_2	Number of floors x_3	Age of home (years) x_4	Price ($1000) y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178
1	3000	4	1	38	540

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \\ 1 & 3000 & 4 & 1 & 38 \end{bmatrix} \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \\ 540 \end{bmatrix}$$

$$X\theta = y$$

Normal Equation: General State

m training examples, n features

$$X\theta = y$$

$$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1} \qquad X = \begin{bmatrix} \_\_\_\_\_ (x^{(1)})^T \_\_\_\_\_ \\ \_\_\_\_\_ (x^{(2)})^T \_\_\_\_\_ \\ \_\_\_\_\_ (x^{(3)})^T \_\_\_\_\_ \\ \vdots \\ \_\_\_\_\_ (x^{(m)})^T \_\_\_\_\_ \end{bmatrix} \in \mathbb{R}^{m \times (n+1)} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m$$

Normal Equation

Solving linear equations

$$X\theta = y$$

$$X^T X\theta = X^T y$$

← Normal Equation

$$\theta = \underbrace{(X^T X)^{-1} X^T}_{X^+} y$$

Python:

```
theta = pinv(X.T @ X) @ X.T @ y
```

Gradient Descent and Normal Equation

Decreasing gradient:

- Need to choose alpha
- Need many repetitions
- It works well even for very large values of n.

$n \geq 10000$

Normal equation:

- No need to choose alpha
- No need to repeat
- It is slow for large n due to the need to calculate the inverse of the X^TX matrix.

$n < 10000$

Normal equation and irreversibility of the X^TX matrix

Normal equation

$$\theta = (X^T X)^{-1} X^T y$$

Question: But what if X^TX is not invertible?

Python:

```
theta = pinv(X.T @ X) @ X.T @ y
```

pseudo reverse

Irreversibility of the X^TX matrix reasons

- Redundancy of features. (Linear dependence)

$$x_1 = size\ (feet^2)$$

$$x_2 = size\ (m^2)$$

$$x_1 = (3.28)^2\, x_2$$

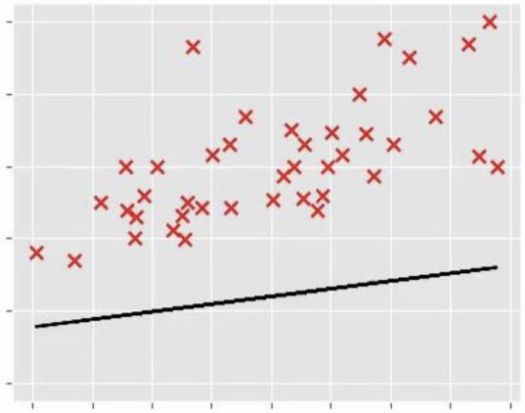
- Huge number of features. ($n \geq m$)
  - solution. Removing some features using regularization (continued)

Regression with Local Weighting

Parametric and non-parametric learning methods

- Parametric learning methods:
  - There is a fixed set of parameters.
  - We don't need a training set to predict new data.
  - Example: regression, logistic regression, neural networks

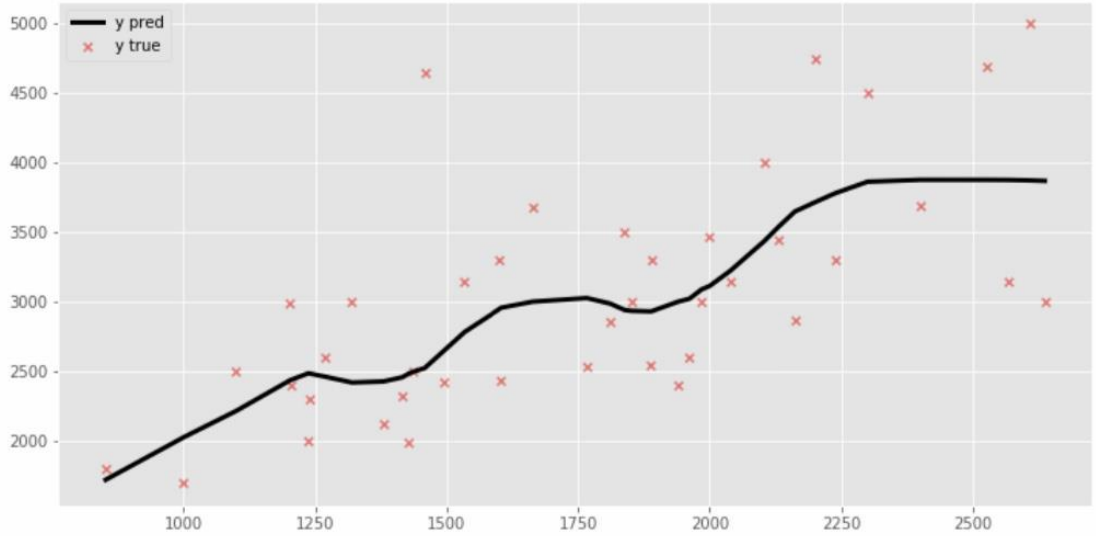
$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$$



$$h_\theta(x^{(new)}) = \theta^T x^{(new)}$$

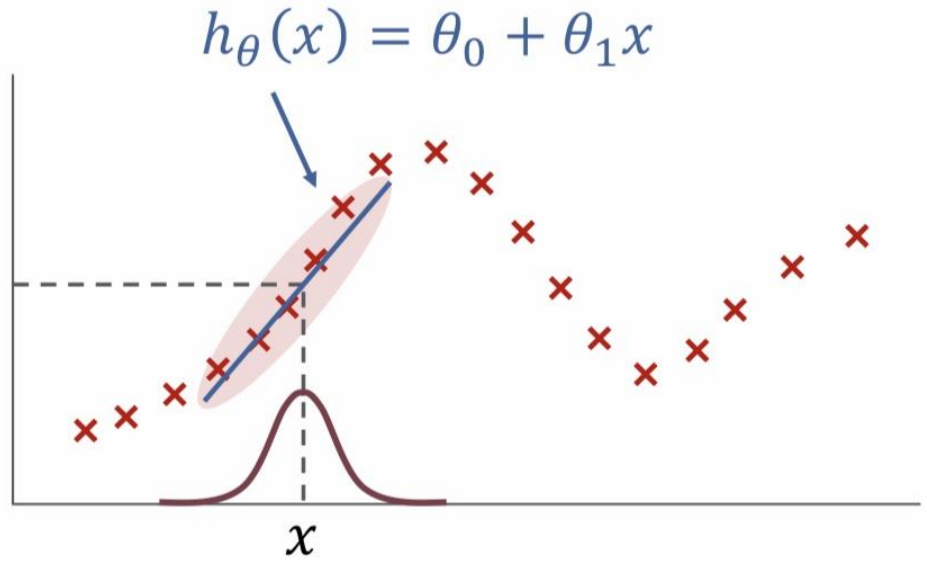
Parametric and non-parametric learning methods

- Non-parametric learning methods:
  - The number of parameters increases (linearly) with the size of the training set.
  - In order to predict new data, we need the entire training set.
  - Example: regression with local weighting (next slide)



Regression with Local Weighting

Idea: Giving more importance to closer data



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

$$w^{(i)} = \exp\left(-\frac{\left(x^{(i)} - x\right)^2}{2\tau^2}\right)$$

Band width

Regression: probabilistic interpretation

Regression: probabilistic interpretation

model

error

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

$$\epsilon^{(i)} \sim N(0, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

- Considering the influence of other factors
  - such as unintended features
- Considering the impact of noise

The errors are independent and follow a uniform Gaussian distribution.

$$y^{(i)} \sim N\left(\theta^T x^{(i)}, \sigma^2\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)$$

Maximum Likelihood Estimation

- Likelihood function: probability of seeing training data as a function of theta parameters

$$L(\theta) = p(Y|X;\theta) = \prod_{i=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)$$

The logarithm of the Likelihood Function:

$$\begin{aligned} l(\theta) = \ln L(\theta) &= \ln \prod_{i=1}^{m} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right) \\ &= \sum_{i=1}^{m} \ln\left[\frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{\left(y^{(i)} - \theta^T x^{(i)}\right)^2}{2\sigma^2}\right)\right] \\ &= m \ln \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2 \end{aligned}$$

Maximum Likelihood Estimation

- Maximum Likelihood Estimation
- Find a value for the theta parameter such that L(θ) is maximized.

$$\theta_{MLE} = \arg\max_{\theta} l(\theta)$$

$$= \arg\max_{\theta} \; m \ln \frac{1}{\sigma\sqrt{2\pi}} - \frac{1}{2\sigma^2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

$$= \arg\max_{\theta} \; -\frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

$$= \arg\min_{\theta} \; \frac{1}{2} \sum_{i=1}^{m} \left(y^{(i)} - \theta^T x^{(i)}\right)^2$$

Cost function