Machine Learning

AIML CLZG565

M3 : Linear Models for Regression

Dr. Sugata Ghosal
sugata.ghosal@pilani.bits-pilani.ac.in

BITS Pilani
Pilani Campus

Linear Regression

Inductive Learning Hypothesis : Interpretation

- Target Concept
- Discrete : f(x) ∈ {Yes, No, Maybe} Classification
- Continuous : f(x) ∈ [20-100] Regression
- Probability Estimation : f(x) ∈ [0-1]

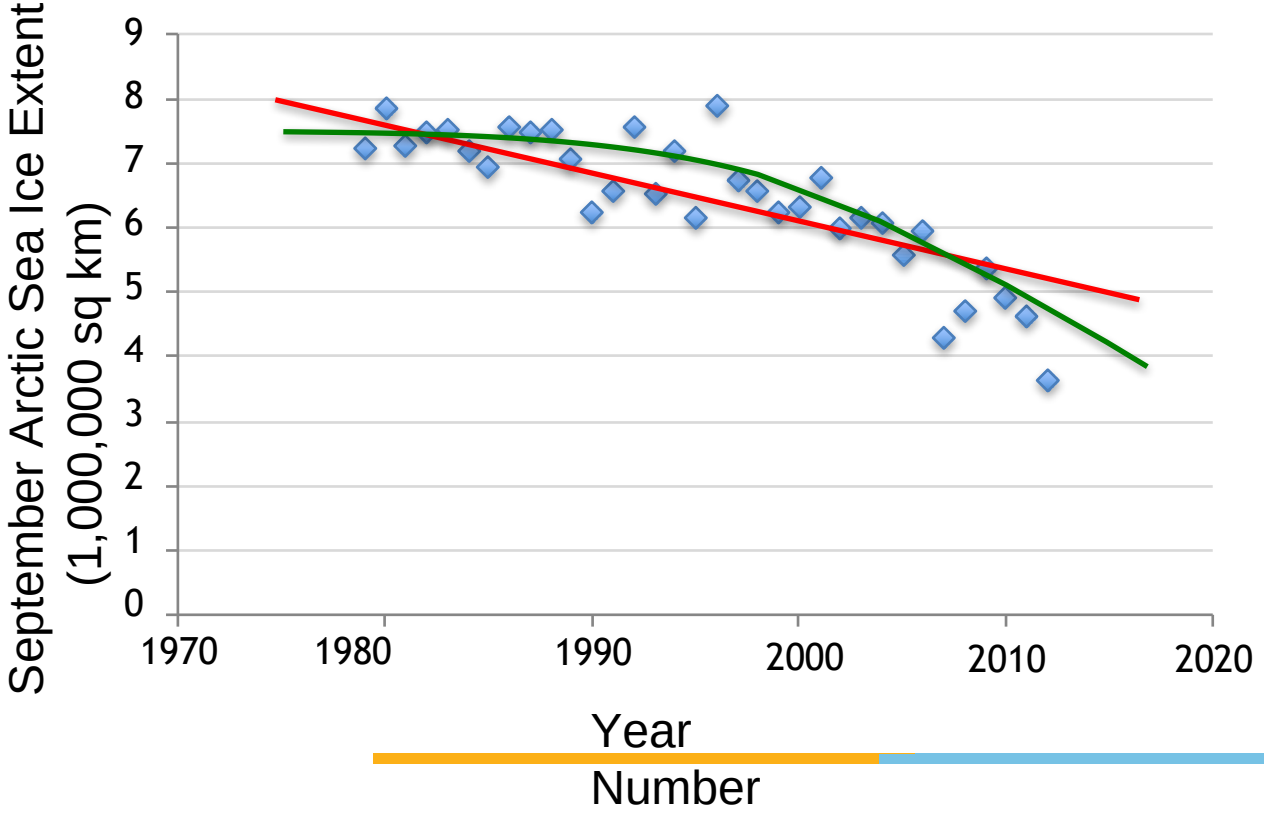
Sky	AirTemp	Altitude	Wind	Water	Forecast	Humidity
Sunny	Warm	Normal	Strong	Warm	Same	60
Sunny	Warm	High	Strong	Warm	Same	75
Rainy	Cold	High	Strong	Warm	Change	70
Sunny	Warm	High	Strong	Cool	Change	45

Formal Representation: Interpretation

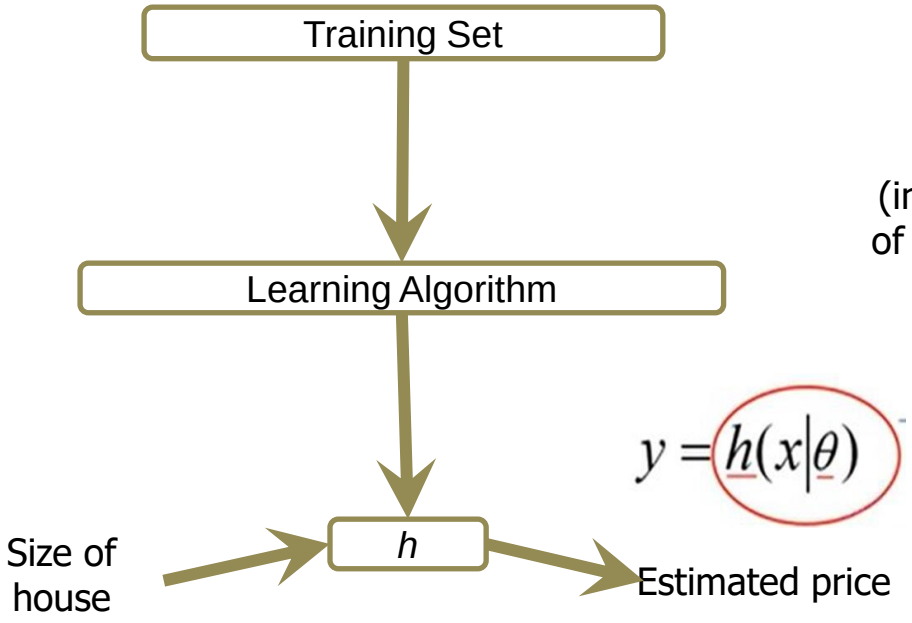
Supervised Learning: Regression

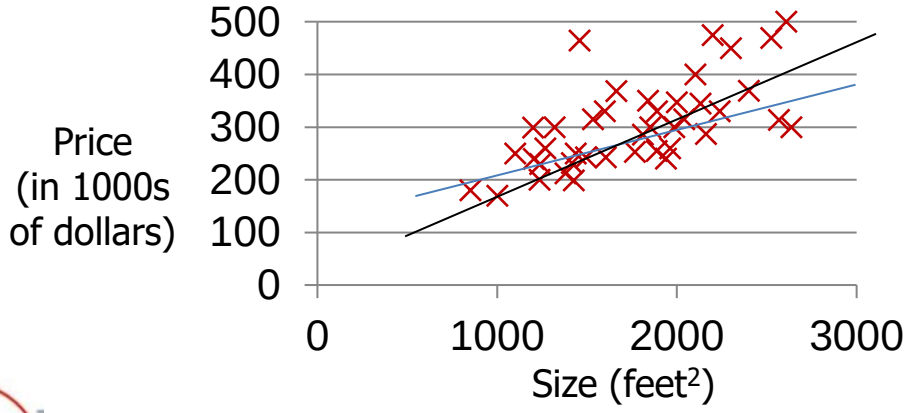
GOAL : Previously unseen records should be assigned a value as accurately as possible.

- Given (x_1, y_1), (x_2, y_2), ..., (x_n, y_n)
- Learn a function $f(x)$ to predict y given x
  - y is real-valued



Machine Learning Process : Interpretation





Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

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

θ_i's : Parameters

h(.) : Model

How to choose θ_i's ?

Types of Regression Models

simple regression model

(Education) x ⟶ y (Income)

multiple regression model

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

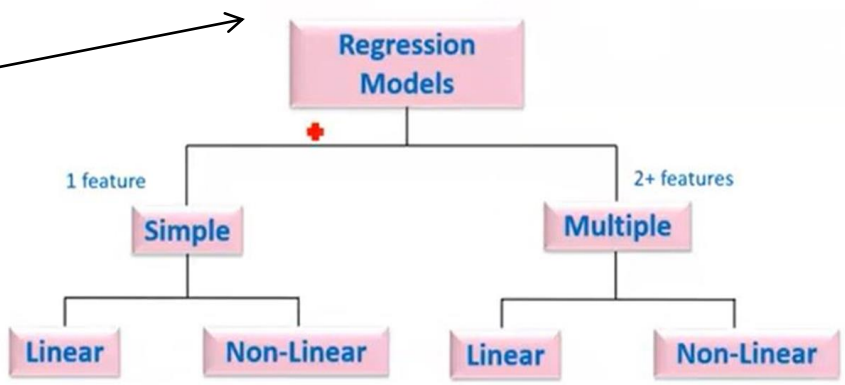
(Education) x_1

(Soft Skills) x_2

(Experience) x_3

(Age) x_4

y (Income)



- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

Assume $x_0 = 1$

Learning the Model Parameters

Closed Form Solution Approach

Gradient Descent Approach

How to choose θ_i's ?

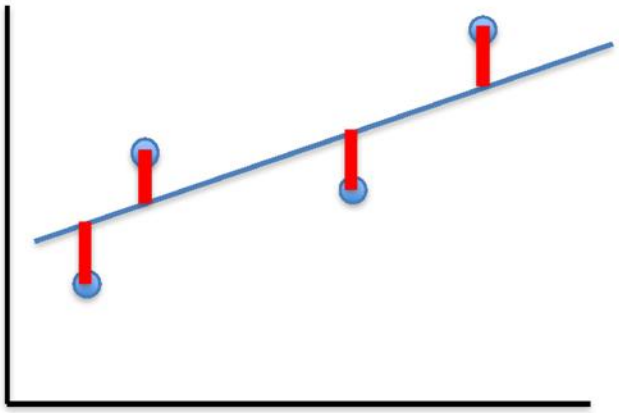
Linear Regression : Least Squares

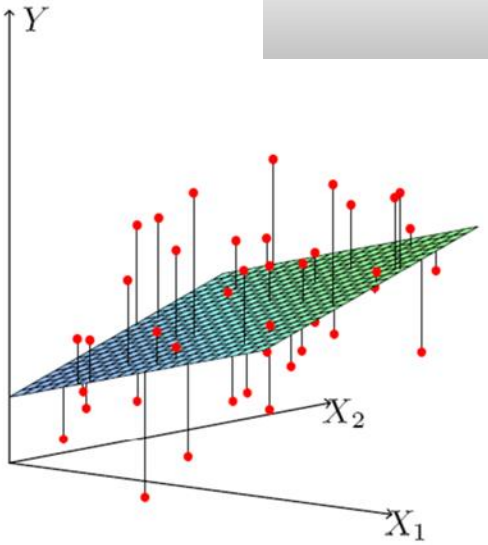
- Cost Function

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

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

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





Intuition Behind Cost Function

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

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

Source Credit : Based on example by Andrew Ng

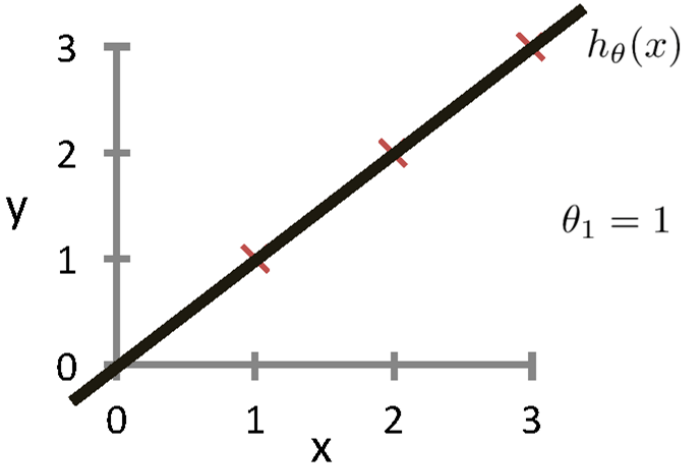
Intuition Behind Cost Function

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

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

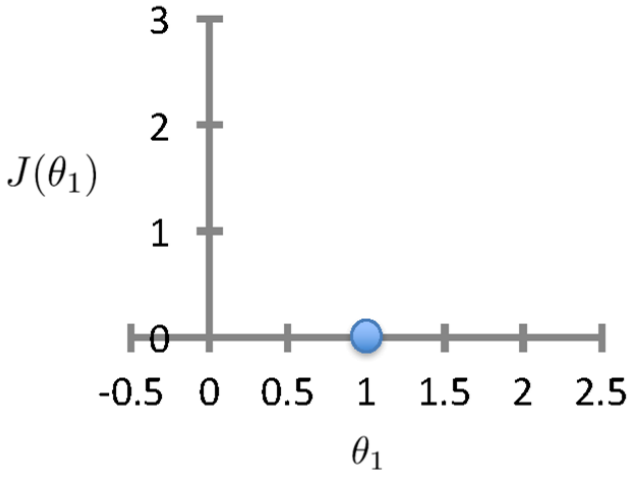
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



Source Credit : Based on example by Andrew Ng

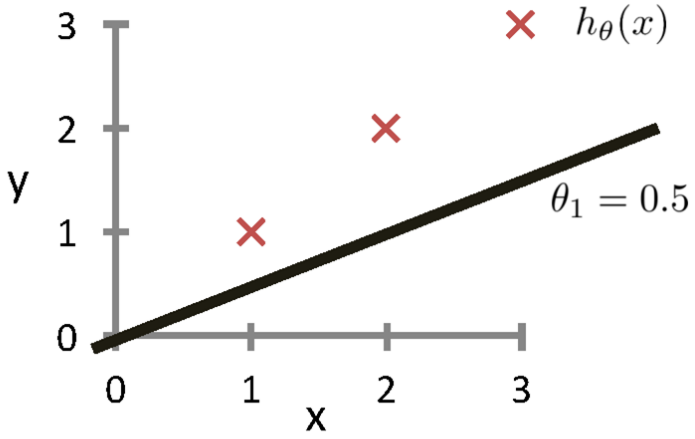
Intuition Behind Cost Function

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

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

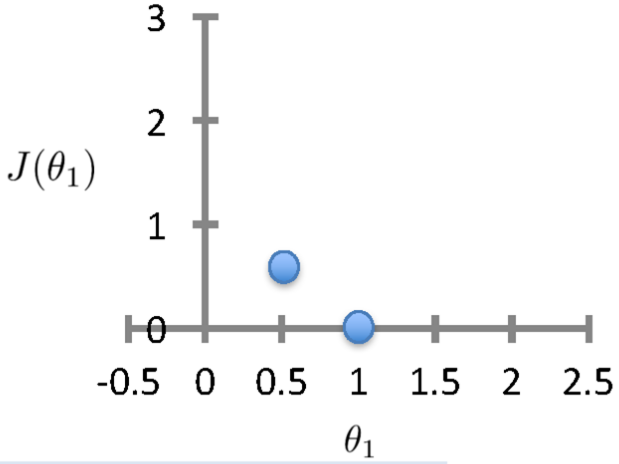
$h_\theta(x)$

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



$J(\theta_1)$

(function of the parameter θ_1)



Based on example by Andrew Ng

$$J([0, 0.5]) = \frac{1}{2 \times 3}\left[(0.5-1)^2 + (1-2)^2 + (1.5-3)^2\right] \approx 0.58$$

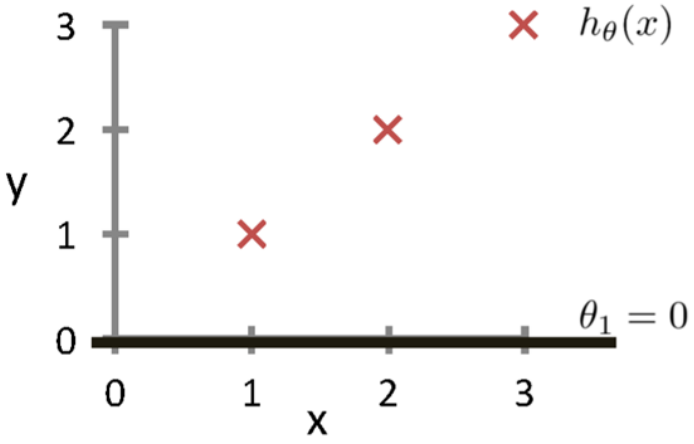
Intuition Behind Cost Function

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

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

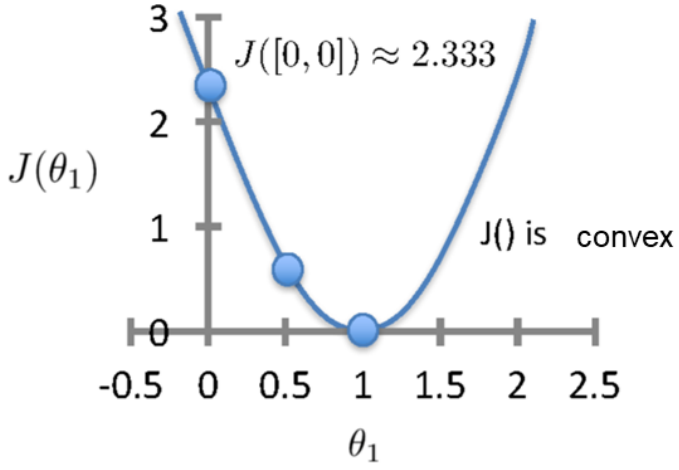
$h_\theta(x)$

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



$J(\theta_1)$

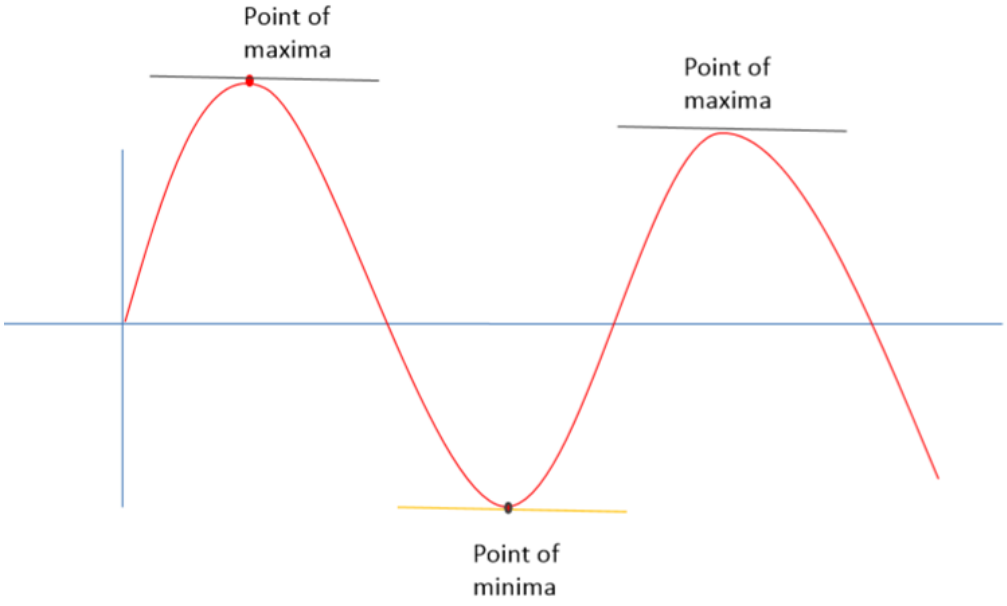
(function of the parameter θ_1)



Source Credit : Based on example by Andrew Ng

Maxima and Minima

- For maxima and minima $m=dy/dx=\tan 0^0=0$
- $dy/dx = 0$ means tangent is parralel to X –axis.



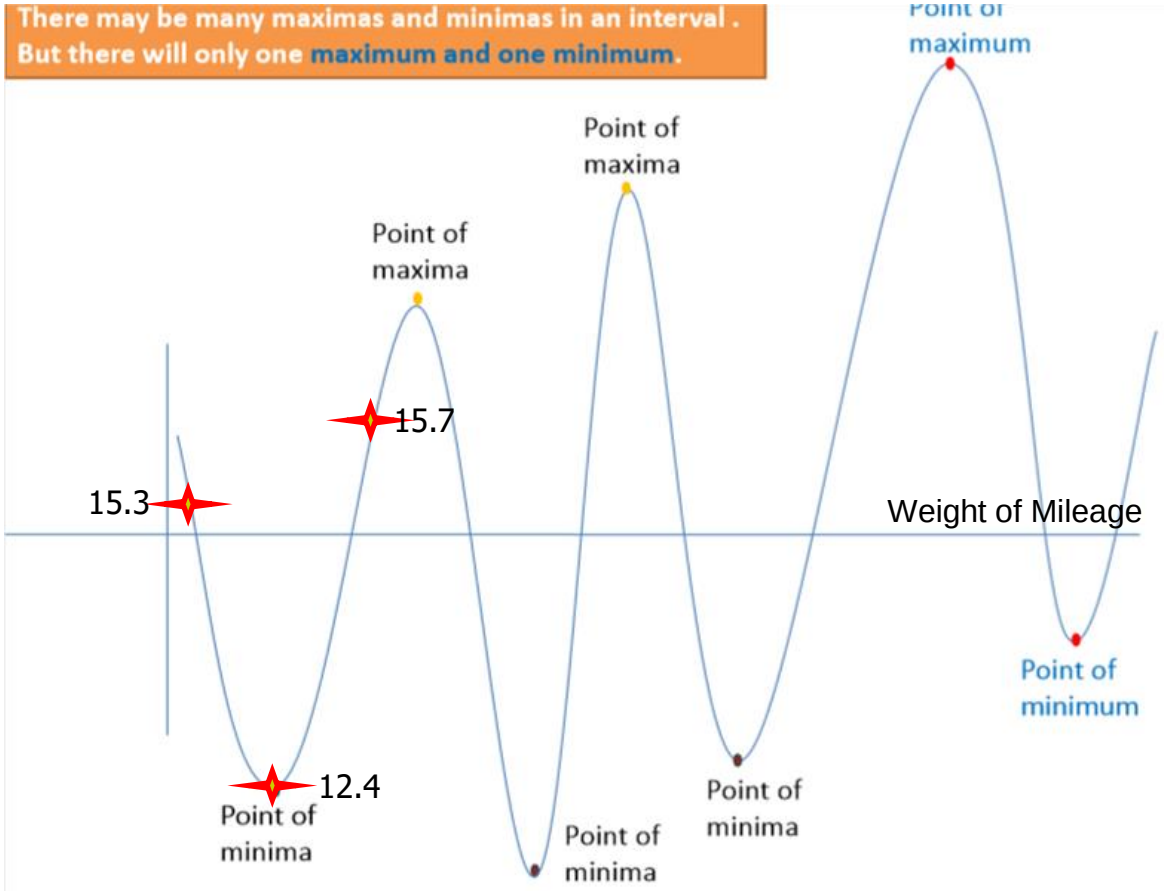
Notion of Maxima and Minima of a function

In Machine Learning

CarPrice = 1 + 0.05Mileage

CarPrice = 1 + 0.25Mileage

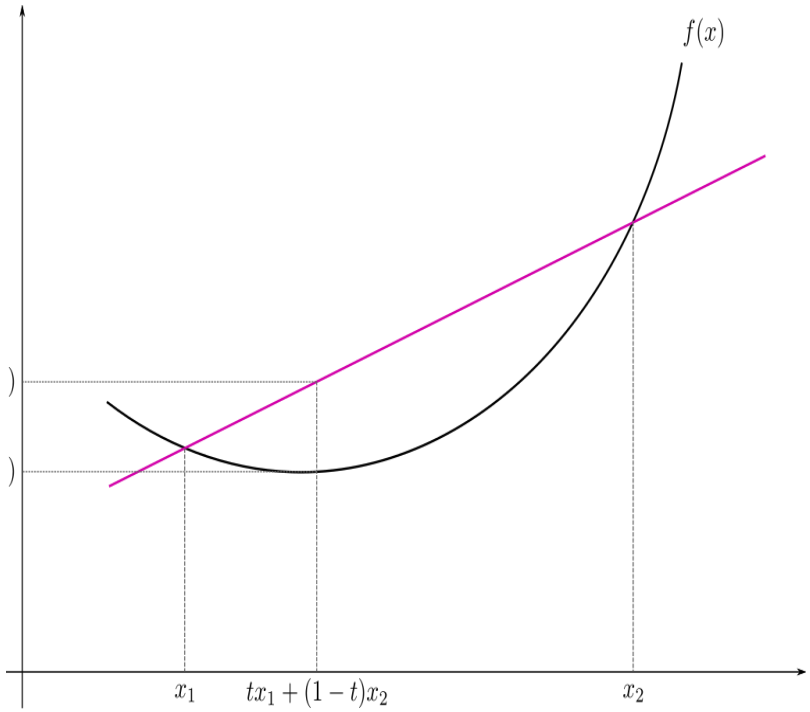
CarPrice = 1 + 0.75Mileage

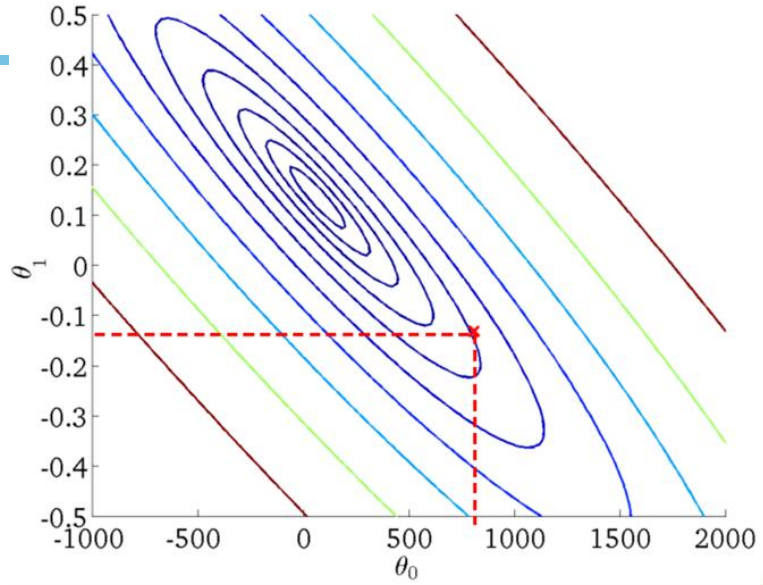


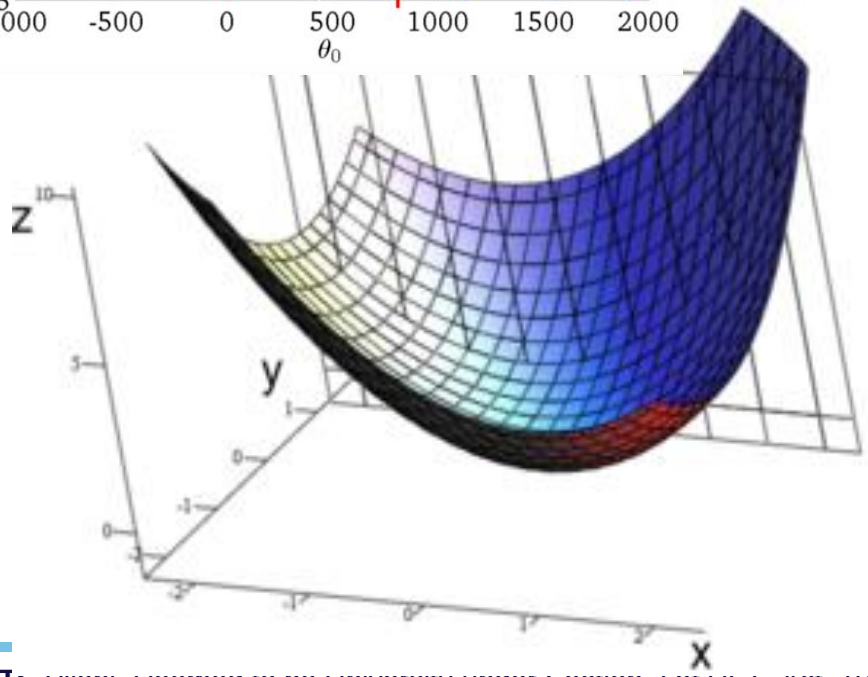
Mileage (in kmpl)	*Car Price (in cr)*
9.8	10.48
9.12	1.75
9.5	6.95
10	2.51

Sum of absolute Error / Training Error

Convex Function : Multivariate







Real-valued function defined on an *n*-dimensional interval is called **convex** if the line segment between any two points on the graph of the function lies above or on the graph

Closed Form Solution

Vectorization

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

- Benefits of vectorization
  - More compact equations
  - Faster code (using optimized matrix libraries)
- Consider our model:

$$h(\boldsymbol{x}) = \sum_{j=0}^{d} \theta_j x_j$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

- Can write the model in vectorized form as $h(\boldsymbol{x}) = \boldsymbol{\theta}^\intercal \boldsymbol{x}$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad x^T = \begin{bmatrix} 1 & No.of.Years\ Experience \end{bmatrix}$$

Vectorization

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

- Consider our model for n instances:

$$h\left(\boldsymbol{x}^{(i)}\right) = \sum_{j=0}^{d} \theta_j x_j^{(i)}$$

- Let

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$\mathbb{R}^{(d+1)\times 1}$ $\mathbb{R}^{n\times(d+1)}$

- Can write the model in vectorized form as $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{X}\boldsymbol{\theta}$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

Vectorization

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

- For the linear regression cost function:

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

$$= \frac{1}{2n}\sum_{i=1}^{n}\left(\boldsymbol{\theta}^{\intercal}\boldsymbol{x}^{(i)} - y^{(i)}\right)^2$$

$$= \frac{1}{2n}\underbrace{(\boldsymbol{X\theta} - \boldsymbol{y})^{\intercal}}_{\mathbb{R}^{1\times n}}\underbrace{(\boldsymbol{X\theta} - \boldsymbol{y})}_{\mathbb{R}^{n\times 1}}$$

$\mathbb{R}^{n\times(d+1)}$, $\mathbb{R}^{(d+1)\times 1}$

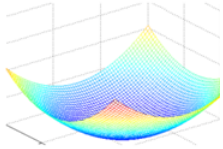
Let:

$$\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

Vectorization

- Instead of using GD, solve for optimal θ analytically
  - Notice that the solution is when $\frac{\partial}{\partial \theta} J(\theta) = 0$



- Derivation:

$$\mathcal{J}(\theta) = \frac{1}{2n} (X\theta - y)^T (X\theta - y)$$

1 x 1

$$\propto \theta^T X^T X\theta - \boxed{y^T X\theta} - \boxed{\theta^T X^T y} + y^T y$$

$$\propto \theta^T X^T X\theta - 2\theta^T X^T y + y^T y$$

Take derivative and set equal to 0, then solve for θ:

$$\frac{\partial}{\partial \theta} (\theta^T X^T X\theta - 2\theta^T X^T y + \cancel{y^T y}) = 0$$

$$(X^T X)\theta - X^T y = 0$$

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

Closed Form Solution: $\theta = (X^T X)^{-1} X^T y$

Vectorization

- Can obtain θ by simply plugging X and y into

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

$$X = \begin{bmatrix} 1 & x_1^{(1)} & \dots & x_d^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(i)} & \dots & x_d^{(i)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \dots & x_d^{(n)} \end{bmatrix} \qquad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(n)} \end{bmatrix}$$

- If $X^T X$ is not invertible (i.e., singular), may need to:
  - Use pseudo-inverse instead of the inverse
    - In python, `numpy.linalg.pinv(a)`
  - Remove redundant (not linearly independent) features
  - Remove extra features to ensure that $d \leq n$

Fit the Linear Regression Model :

Using Closed Form

X No.of.Years of Experience (in Years)	Y Salary Of the Employee (in Lakhs)
1	2
2	3
3	4
4	5
5	6

$$\left(X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * X \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \\ 1 & 4 \\ 1 & 5 \end{bmatrix} \right) - 1 * X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\left(\begin{bmatrix} 5 & 15 \\ 15 & 55 \end{bmatrix} \right) - 1 * X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\left(\begin{bmatrix} 1.1 & -0.3 \\ -0.3 & 0.1 \end{bmatrix} \right) * X^T \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

$$\begin{bmatrix} 0.8 & 0.5 & 0.2 & -0.1 & -0.4 \\ -0.2 & -0.1 & 0 & 0.1 & -0.2 \end{bmatrix} * y \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

Can obtain θ by simply plugging X and y into

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

$$\theta^T = \begin{bmatrix} \theta_0 & \theta_1 \end{bmatrix} \quad X = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{bmatrix} \quad y = \begin{bmatrix} 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

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

Gradient Descent Approach

BITS Pilani

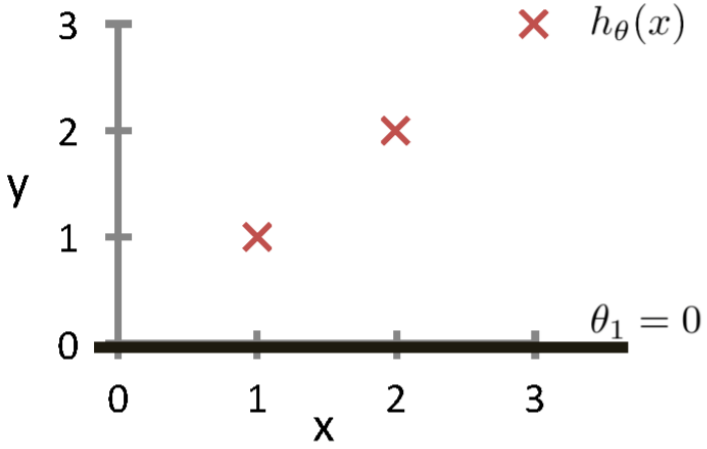
Intuition Behind Cost Function

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

For insight on J(), let's assume $x \in \mathbb{R}$ so $\boldsymbol{\theta} = [\theta_0, \theta_1]$

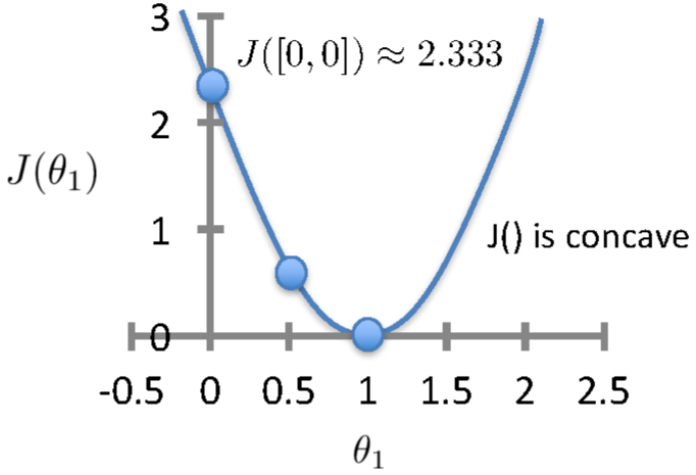
$h_\theta(x)$

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



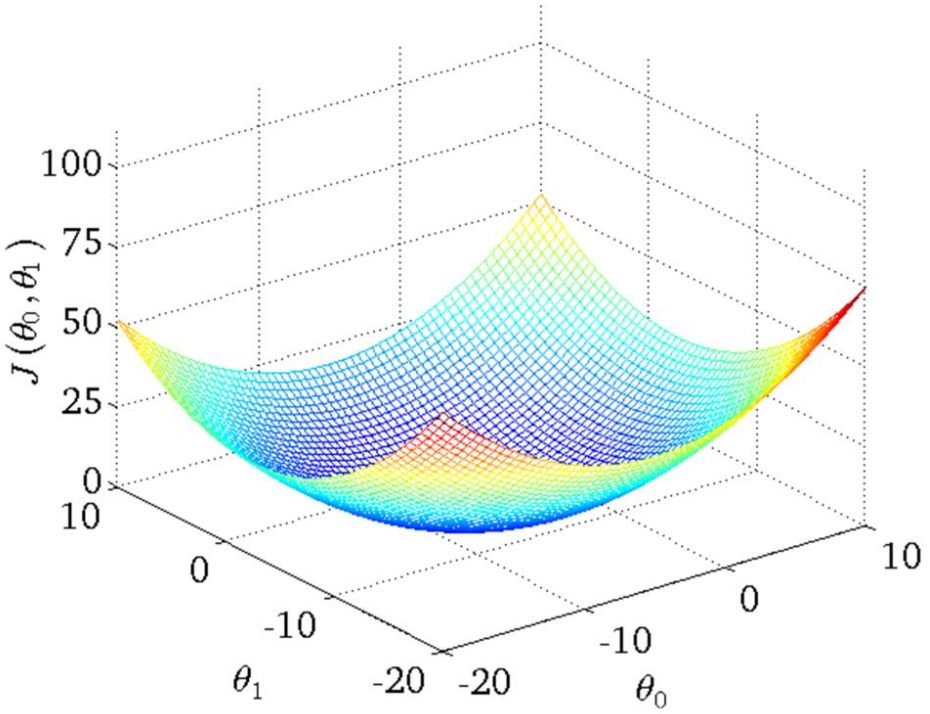
$J(\theta_1)$

(function of the parameter θ_1)



Source Credit : Based on example by Andrew Ng

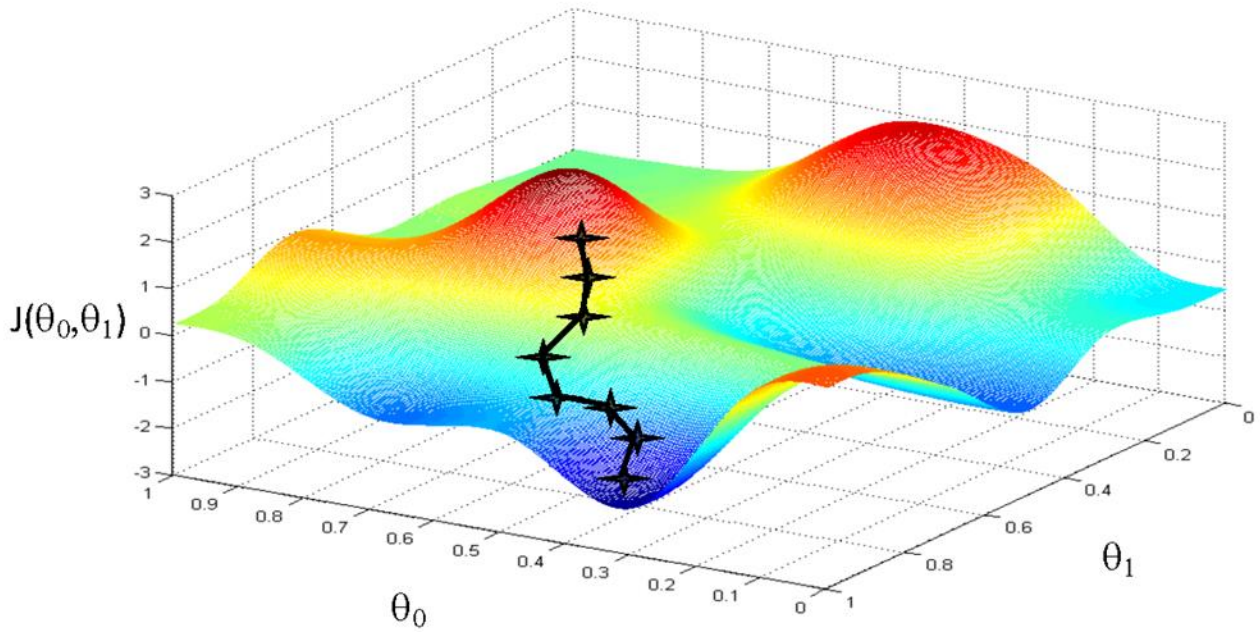
Intuition Behind Cost Function



Source Credit : Based on example by Andrew Ng

Basic Search Procedure

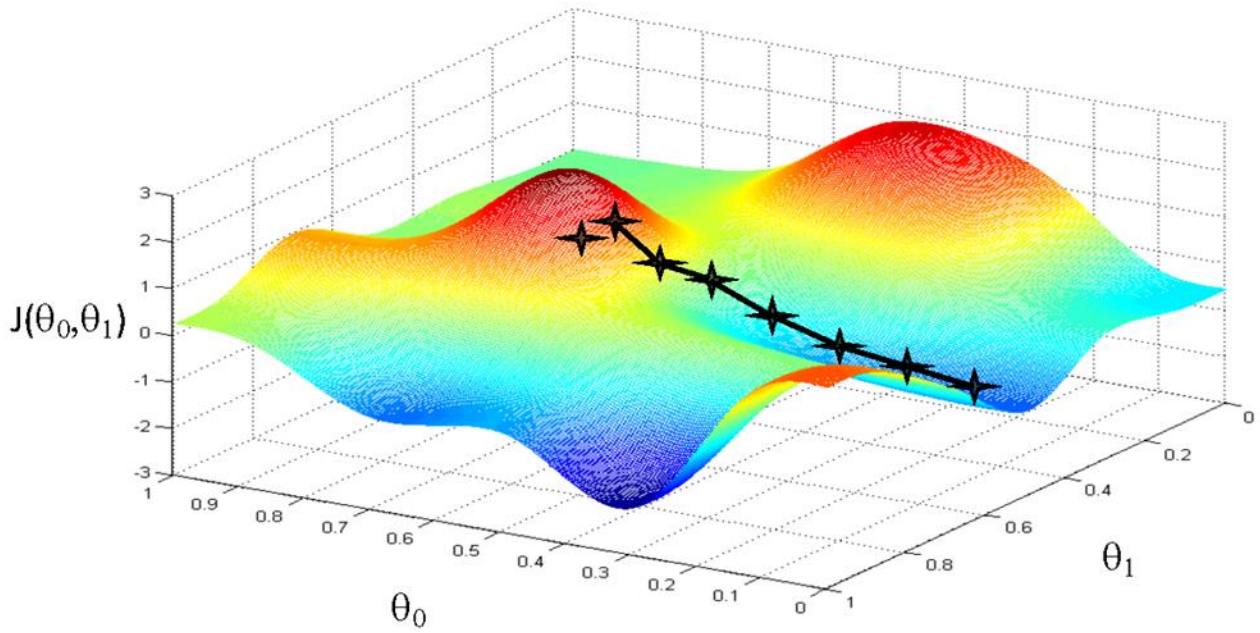
- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Source Credit : Figure by Andrew Ng

Basic Search Procedure

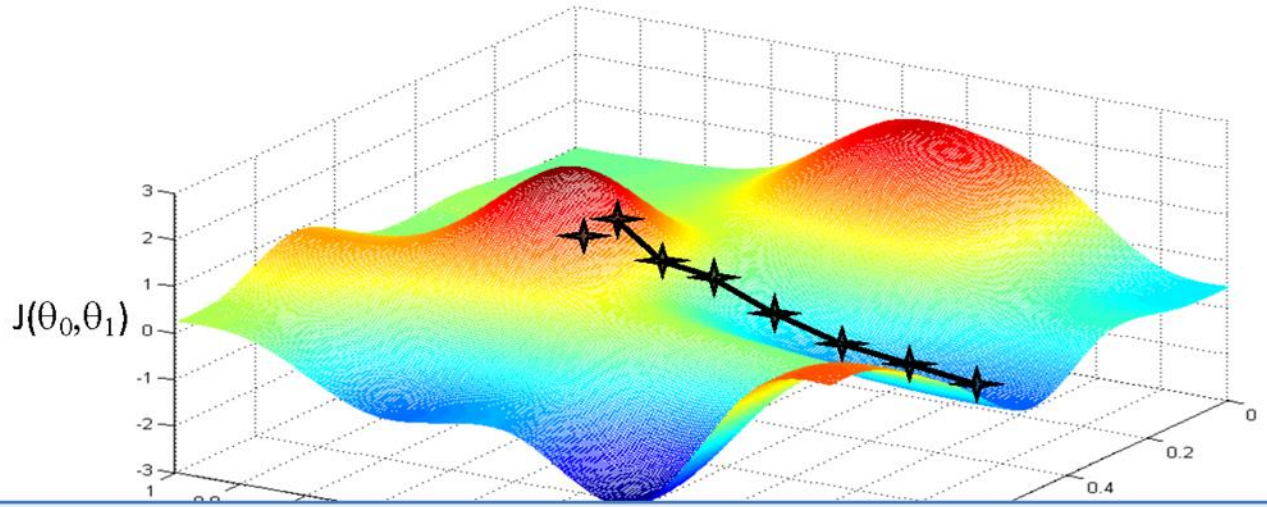
- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Source Credit : Figure by Andrew Ng

Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
  - Choose a new value for θ to reduce $J(\theta)$



Since the least squares objective function is convex (concave), we don't need to worry about local minima

Source Credit : Figure by Andrew Ng

Gradient Descent

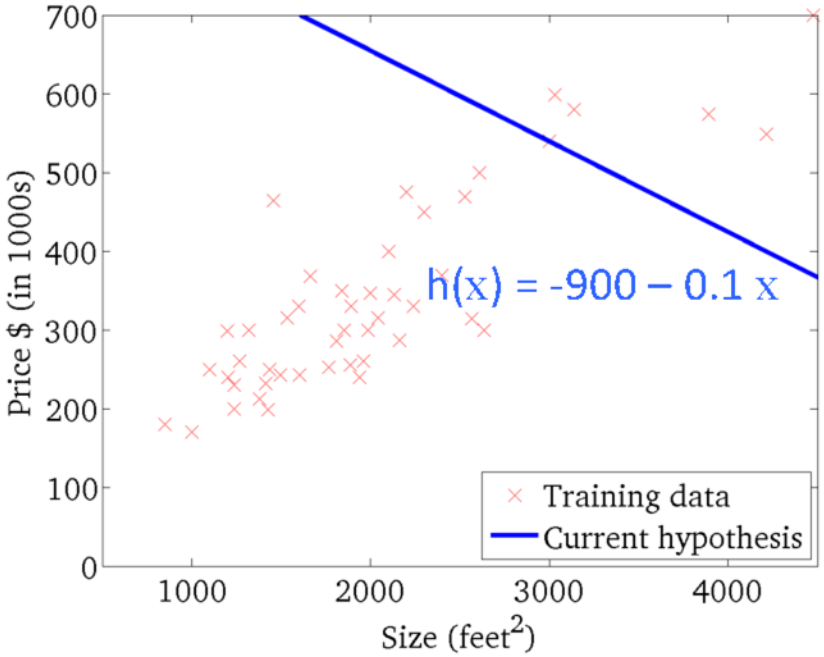
Size in feet² (x)	Price ($) in 1000's (y)
2104	460
1416	232
1534	315
852	178
…	…

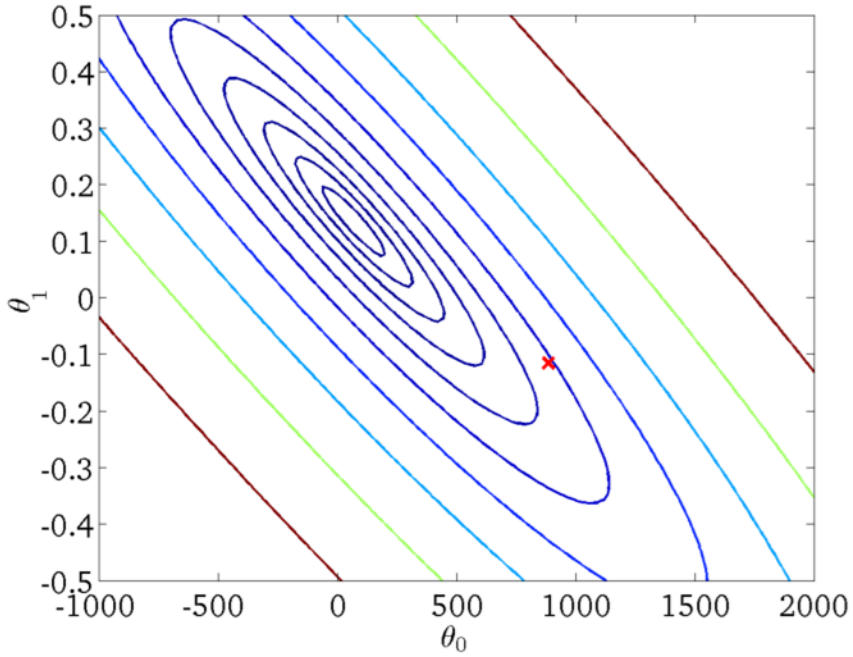
$h_\theta(x)$

(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)



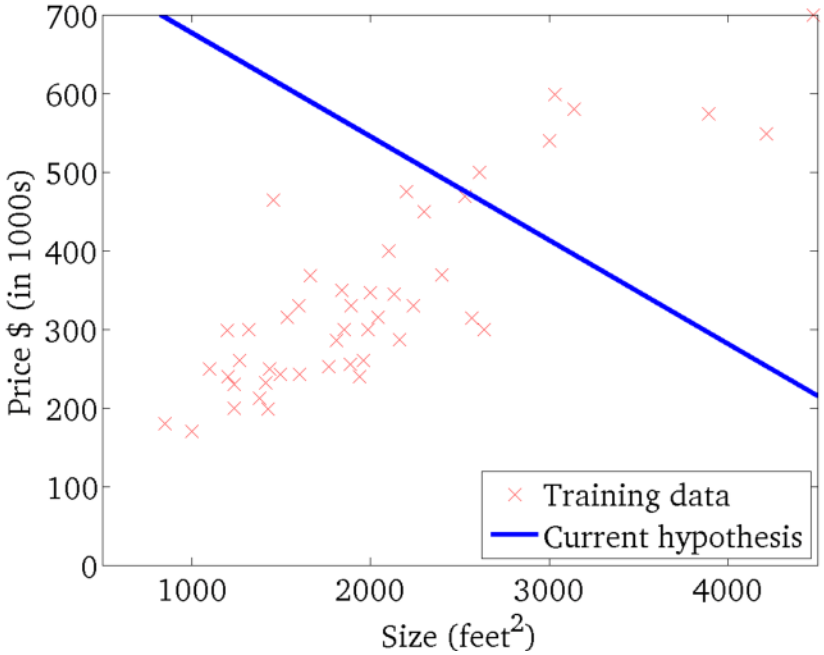


Source Credit : Slide by Andrew Ng

Gradient Descent

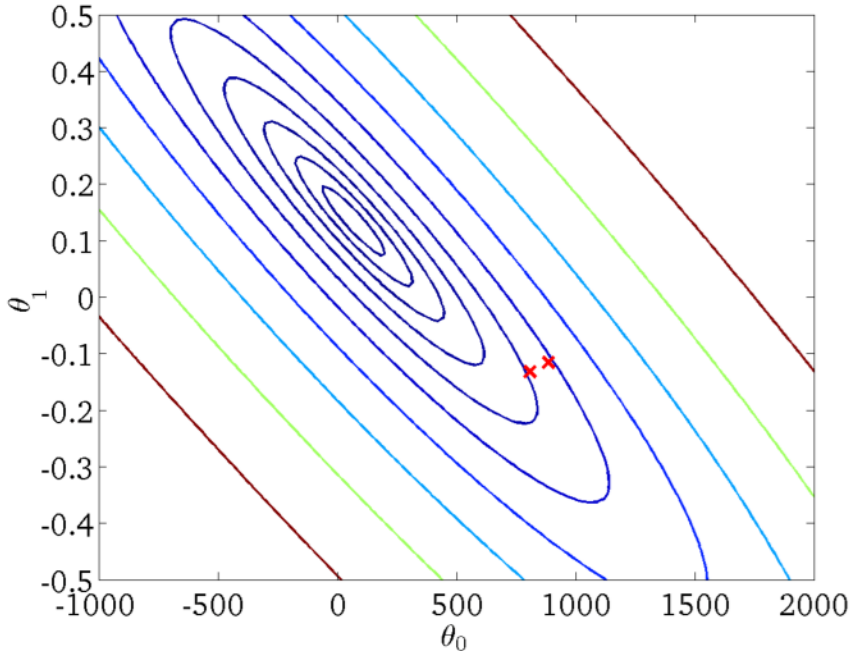
$h_\theta(x)$

(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)

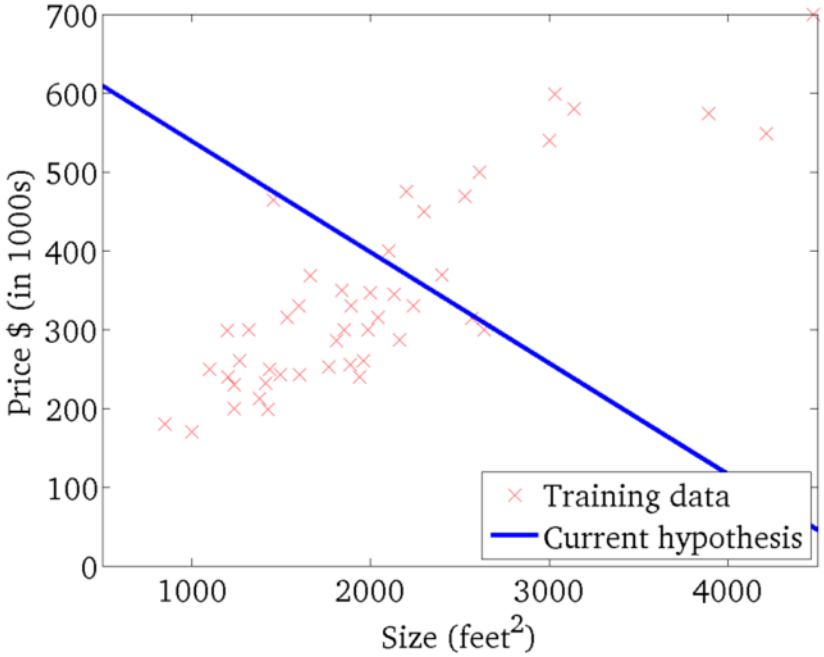


Source Credit : Slide by Andrew Ng

Gradient Descent

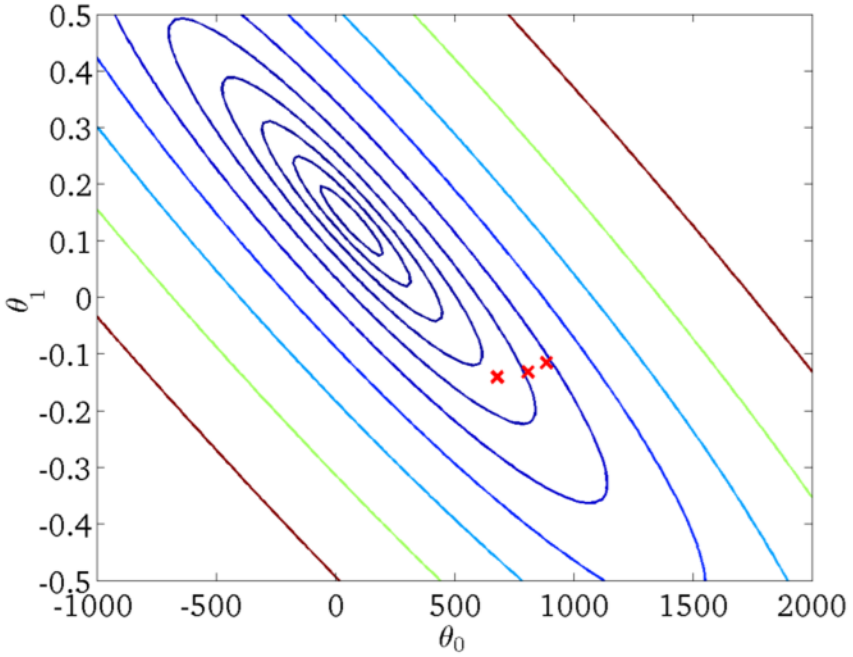
$h_\theta(x)$

(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)

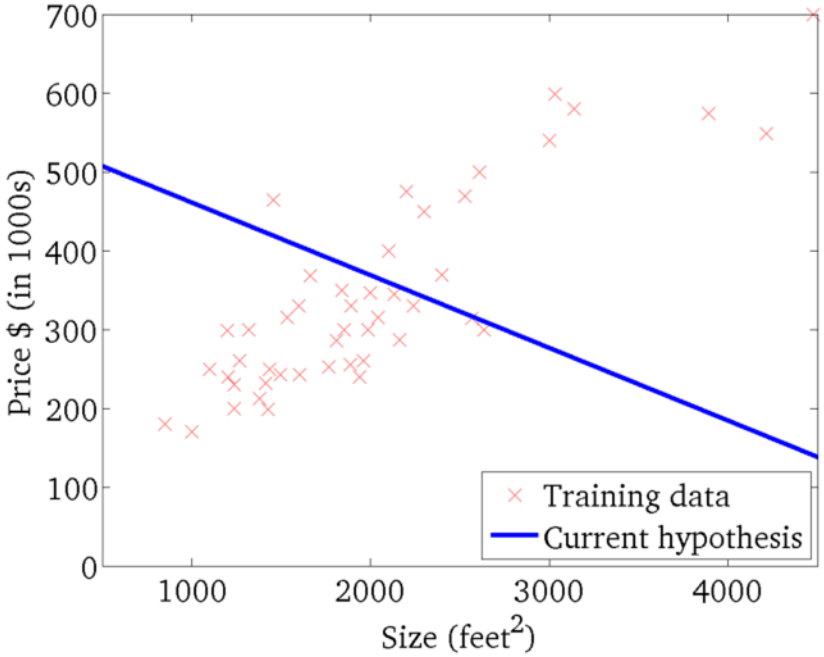


Source Credit : Slide by Andrew Ng

Gradient Descent

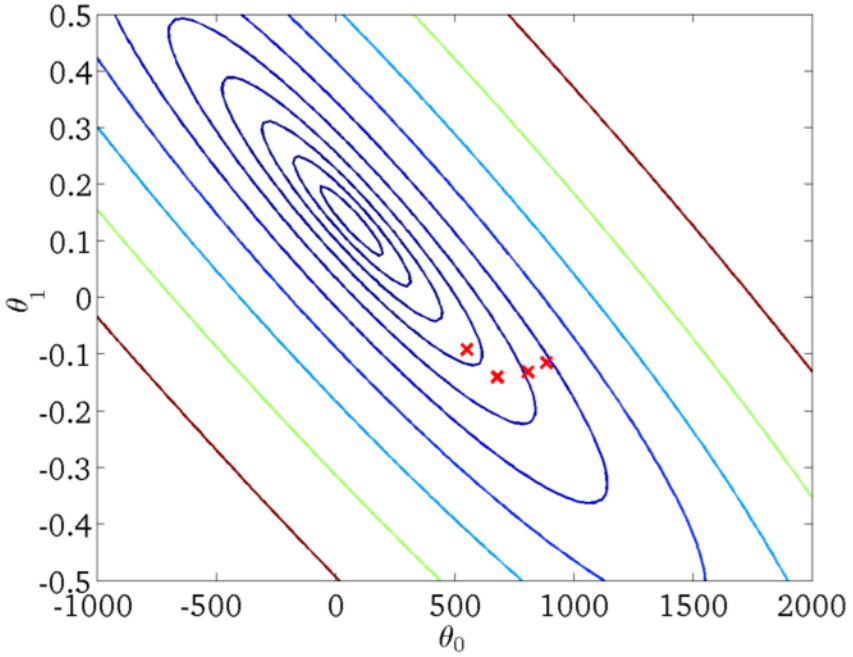
$h_\theta(x)$

(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)

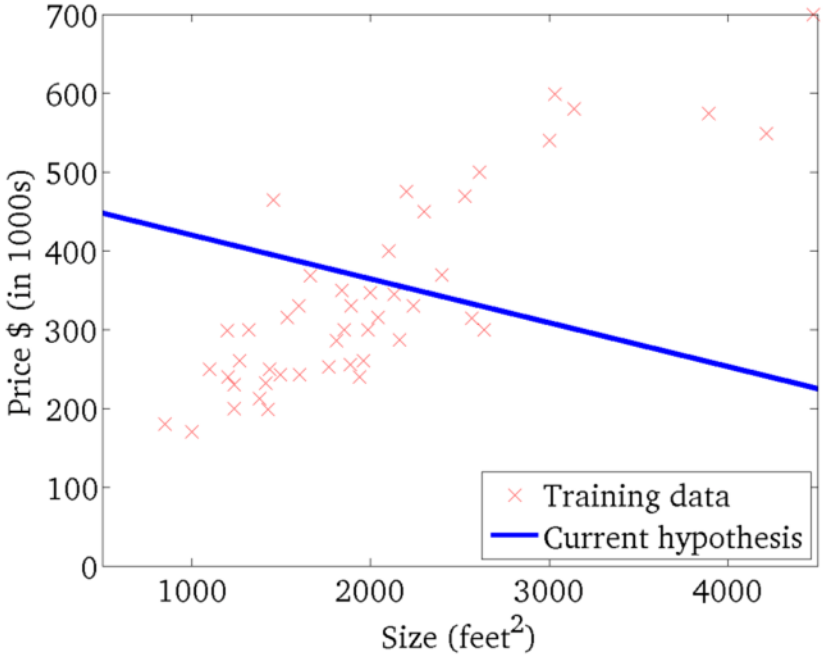


Source Credit : Slide by Andrew Ng

Gradient Descent

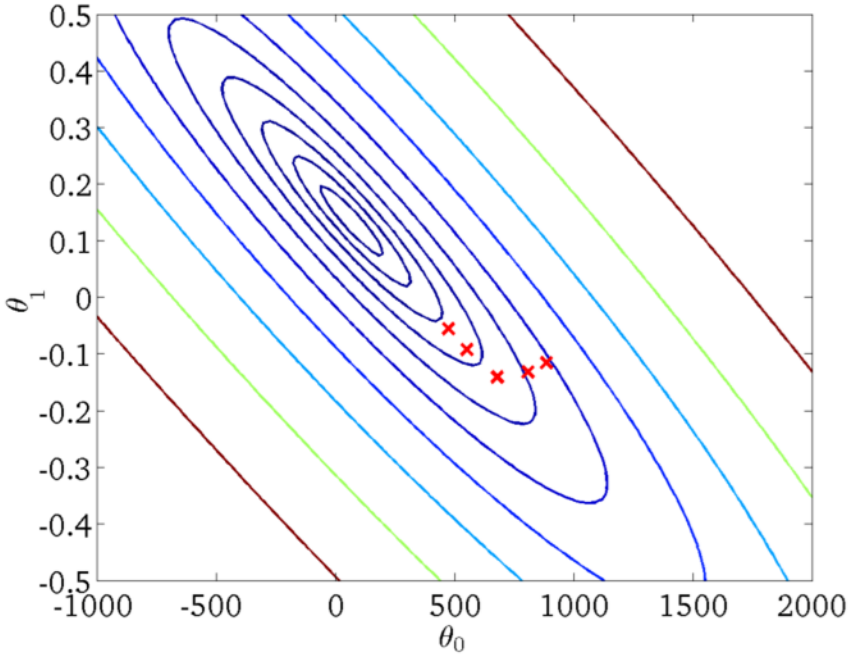
$h_\theta(x)$

(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)

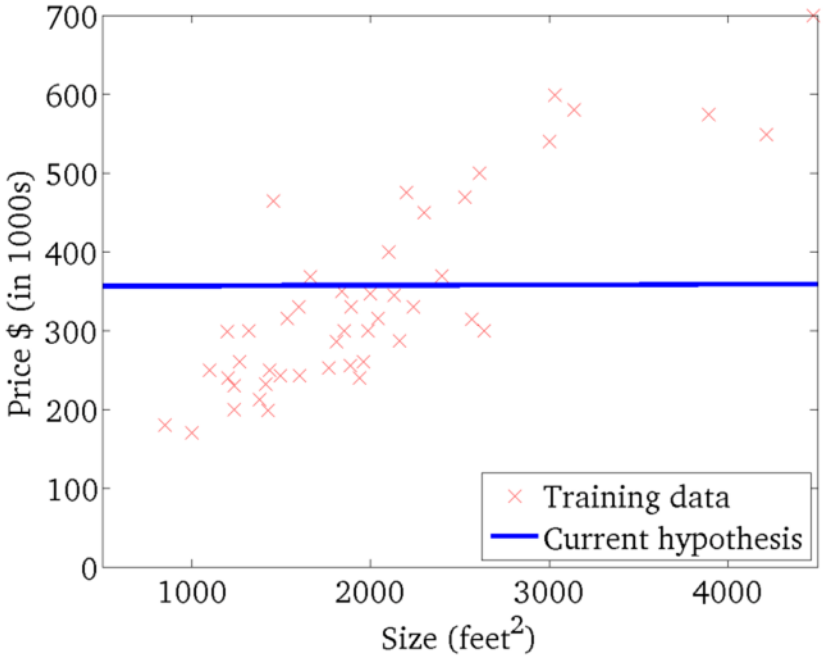


Source Credit : Slide by Andrew Ng

Gradient Descent

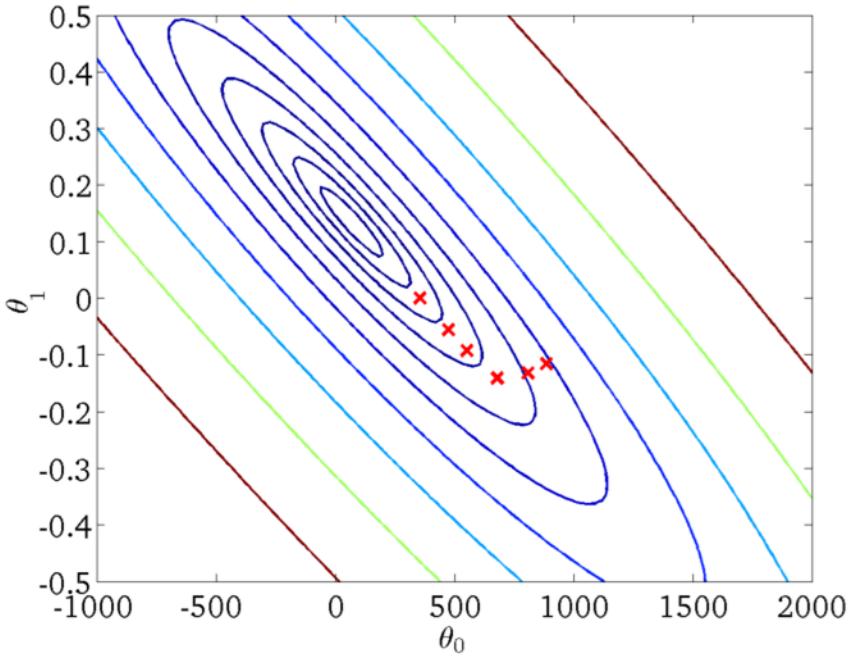
$h_\theta(x)$

(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)

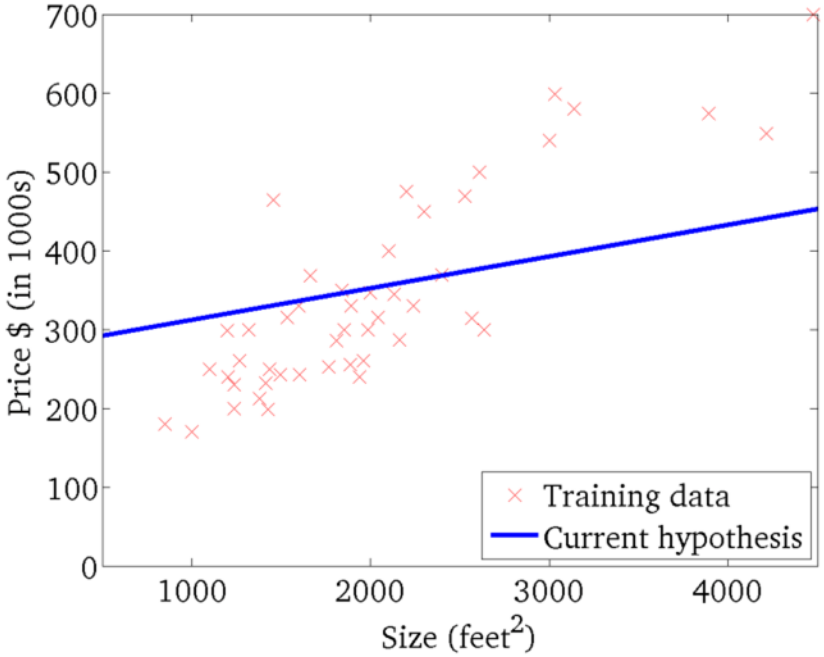


Source Credit : Slide by Andrew Ng

Gradient Descent

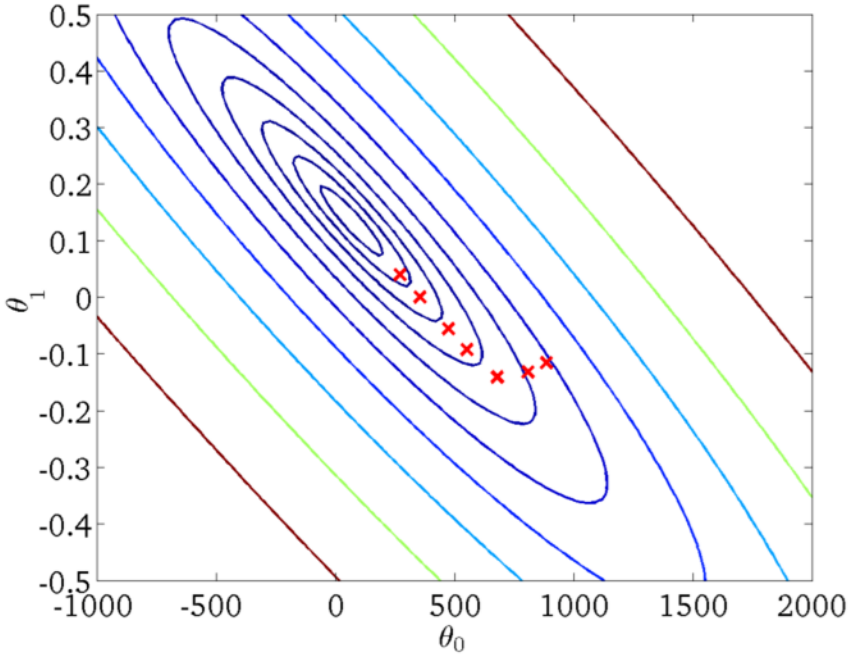
$h_\theta(x)$

(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)

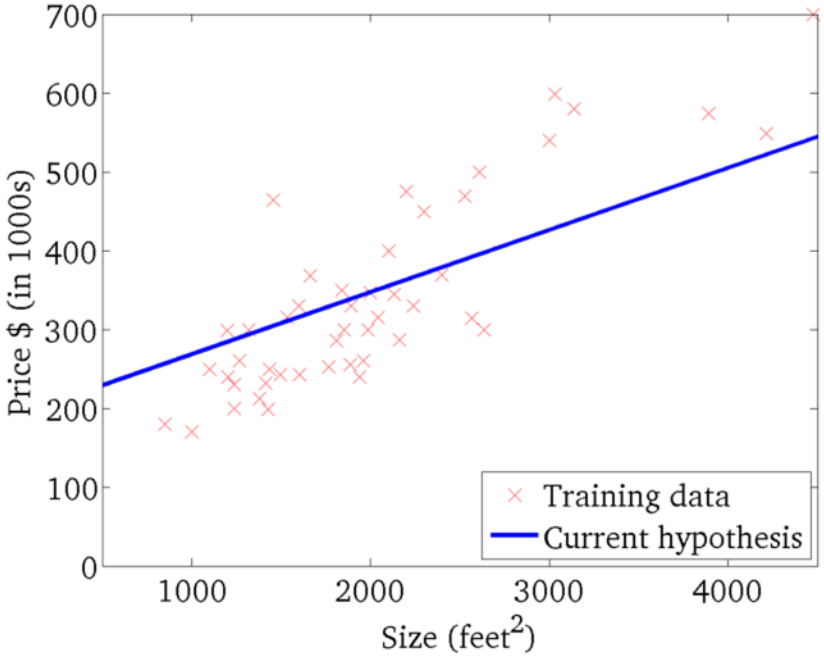


Source Credit : Slide by Andrew Ng

Gradient Descent

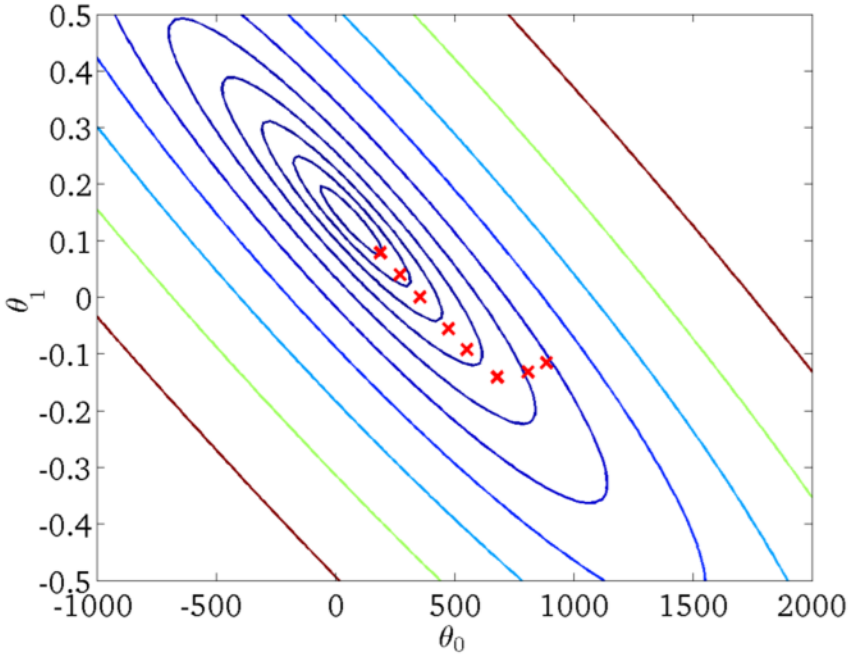
$h_\theta(x)$

(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)

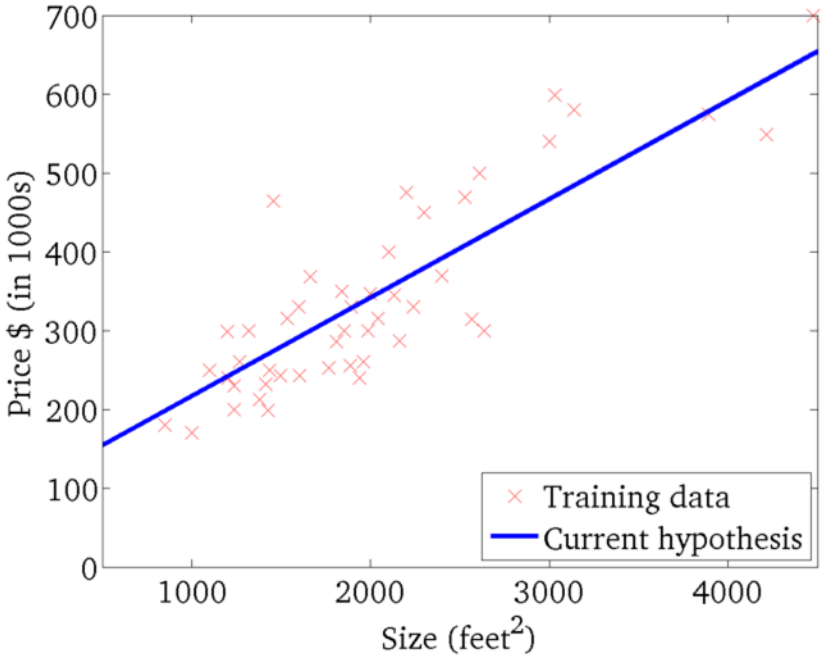


Source Credit : Slide by Andrew Ng

Gradient Descent

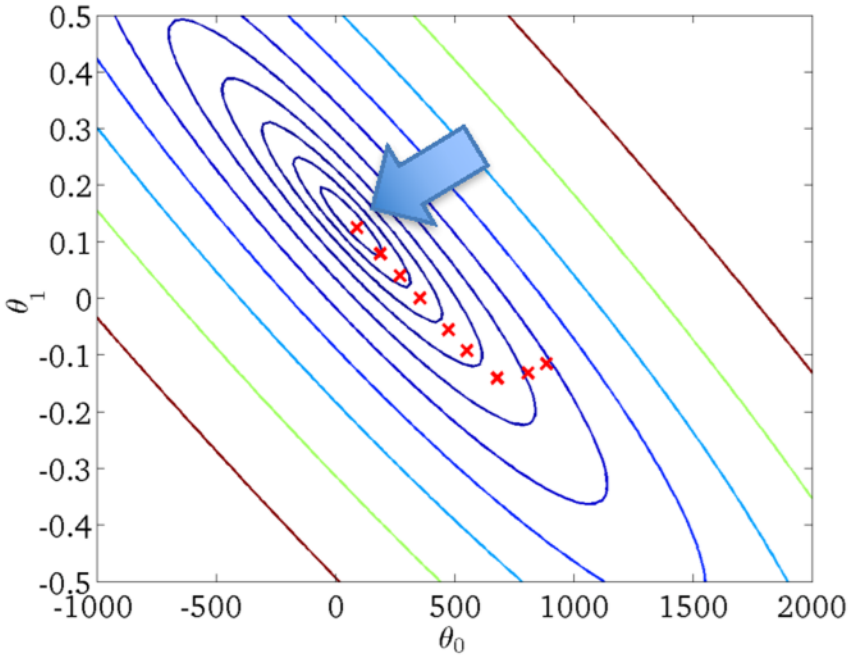
$h_\theta(x)$

(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)



Source Credit : Slide by Andrew Ng

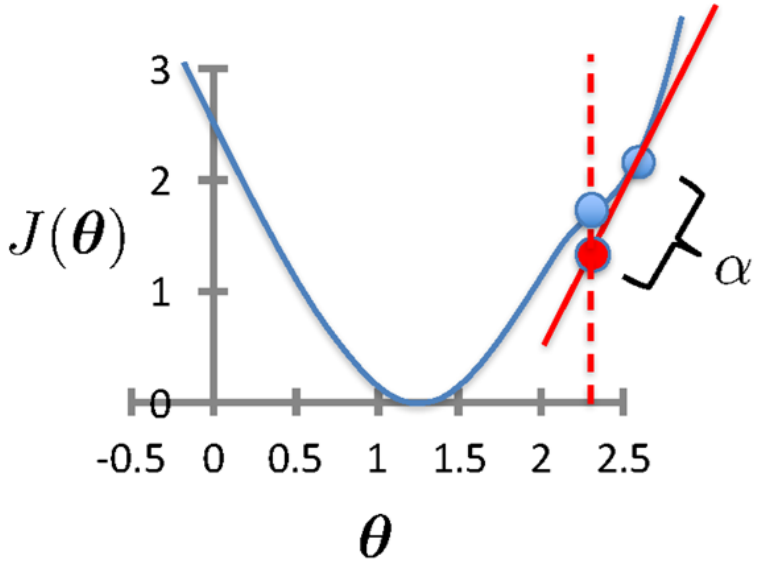
Gradient Descent

- Initialize θ
- Repeat until convergence

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

simultaneous update
for j = 0 ... d

learning rate (small)
e.g., α = 0.05



Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for j = 0 ... d

For Linear Regression:

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right)^2$$

$$= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)^2$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right)$$

$$= \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{k=0}^{d} \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)}$$

Gradient Descent for Linear Regression

- Initialize θ
- Repeat until convergence

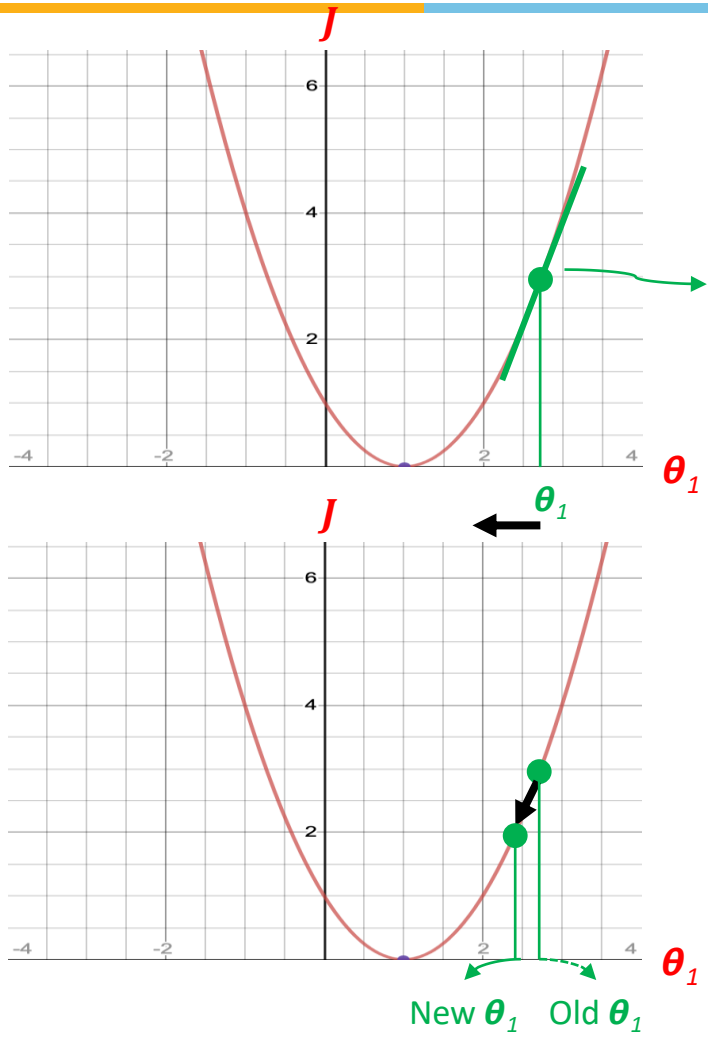
$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)}$$

simultaneous update for j = 0 ... d

- To achieve simultaneous update
  - At the start of each GD iteration, compute $h_{\boldsymbol{\theta}} \left(\boldsymbol{x}^{(i)} \right)$
  - Use this stored value in the update step loop
- Assume convergence when $\| \boldsymbol{\theta}_{new} - \boldsymbol{\theta}_{old} \|_2 < \epsilon$

L_2 norm: $\| \boldsymbol{v} \|_2 = \sqrt{\sum_i v_i^2} = \sqrt{v_1^2 + v_2^2 + \ldots + v_{|v|}^2}$

Guarantee of Convergence



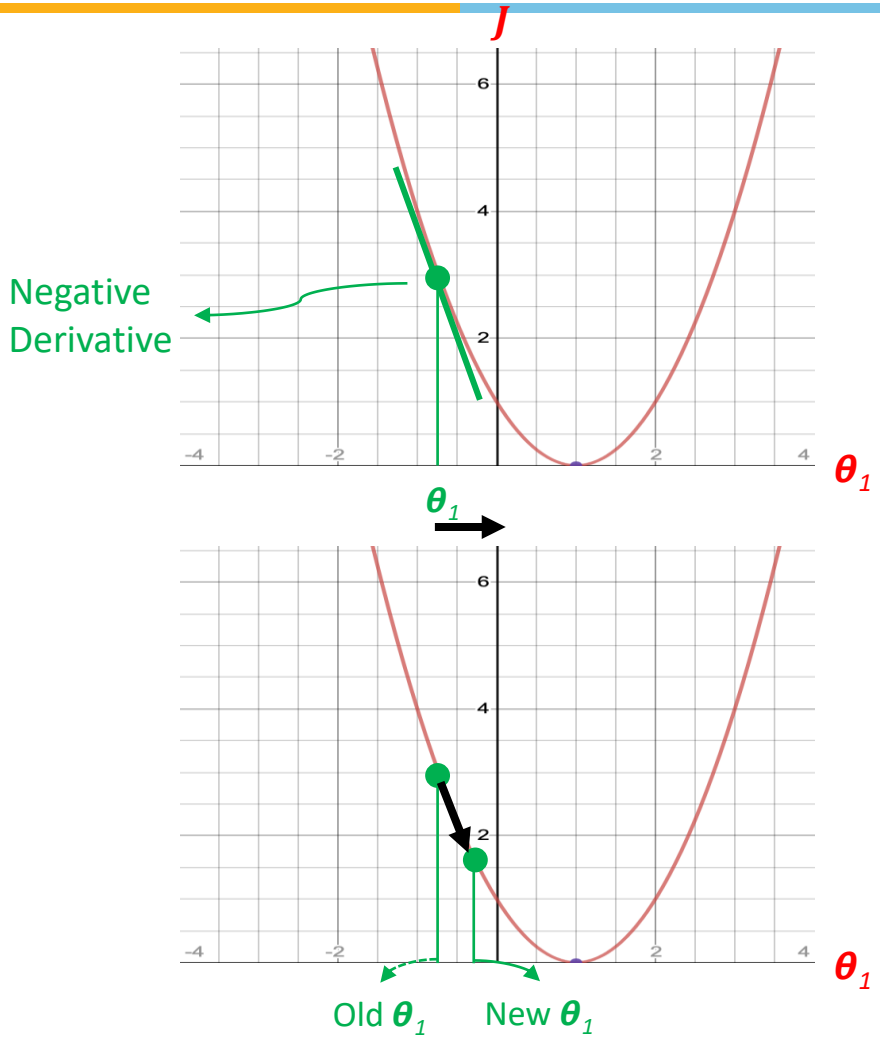
$$\theta_1 = \theta_1 - \alpha \frac{\boldsymbol{d}\, \boldsymbol{J(\theta_1)}}{\boldsymbol{d}\, \theta_j}$$

$$= \theta_1 - \alpha\, (Positive\ Number)$$

Decrease θ_1 by a certain value

Source Credit: Prof. Mohammad Hammoud

Guarantee of Convergence



Source Credit: Prof. Mohammad Hammoud

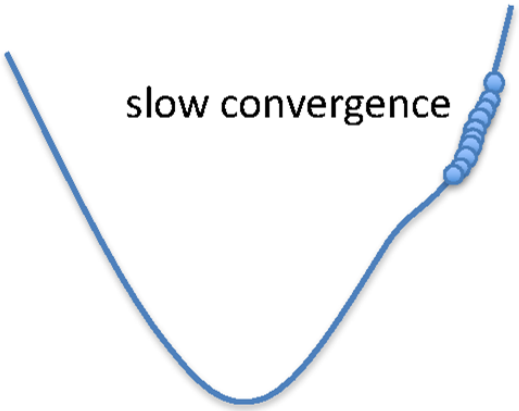
$$\theta_1 = \theta_1 - \alpha \frac{\boldsymbol{d}\, \boldsymbol{J(\theta_1)}}{\boldsymbol{d}\, \theta_j}$$

$$= \theta_1 - \alpha\, (Negative\ Number)$$

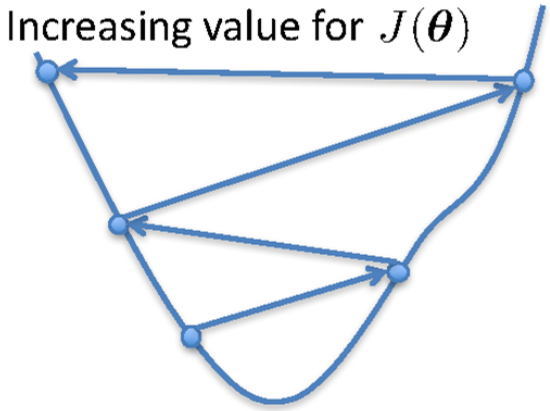
Increase θ_1 by a certain value

Choosing Learning Rate

α too small



α too large



- May overshoot the minimum
- May fail to converge
- May even diverge

To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

Gradient Descent algorithm : Effect of Feature Scaling

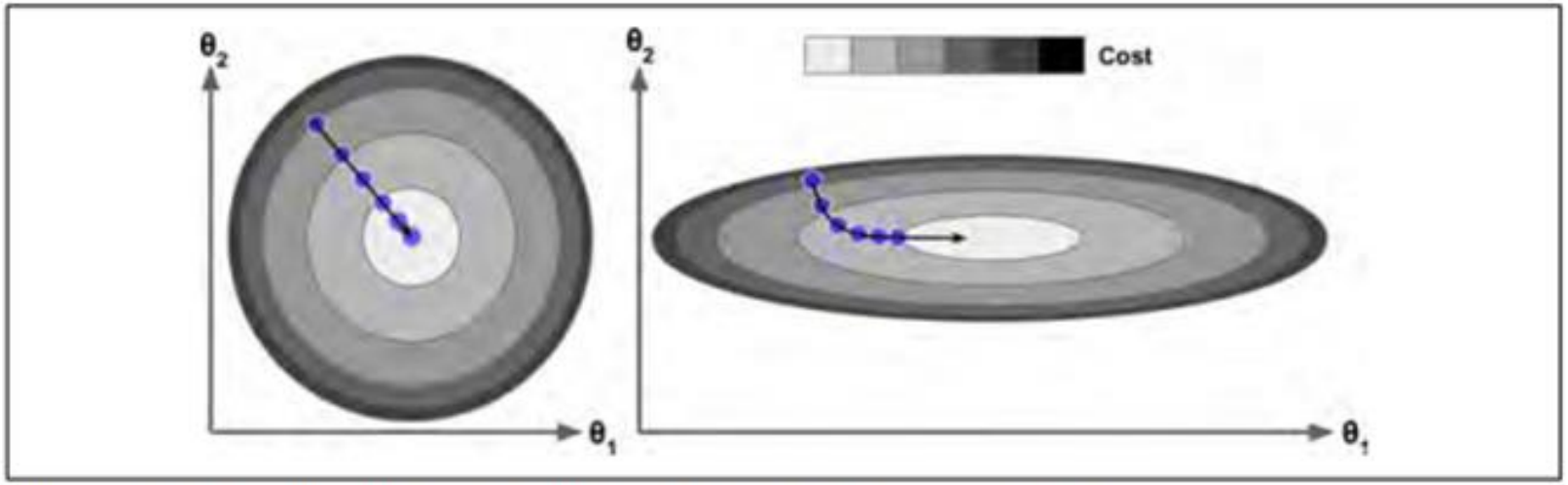


Figure 4-7. Gradient Descent with and without feature scaling

Gradient Descent algorithm : Effect of Learning Rate

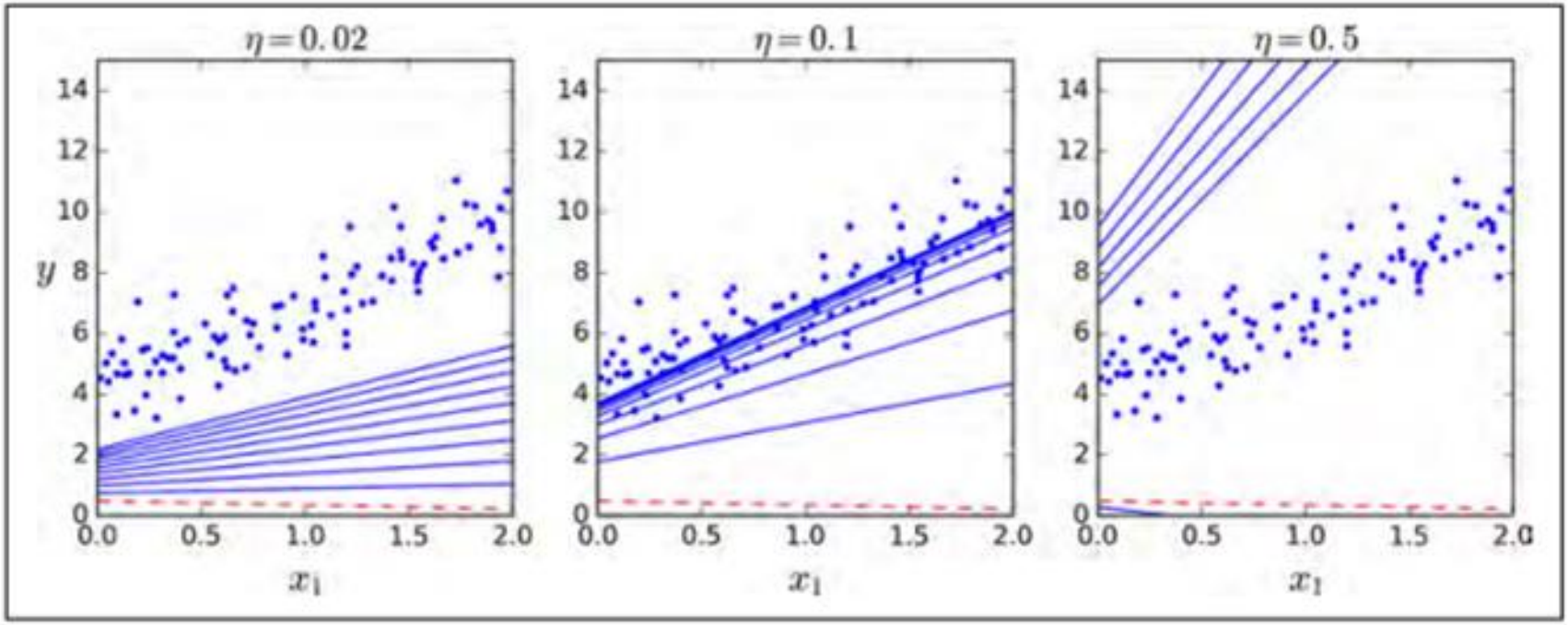
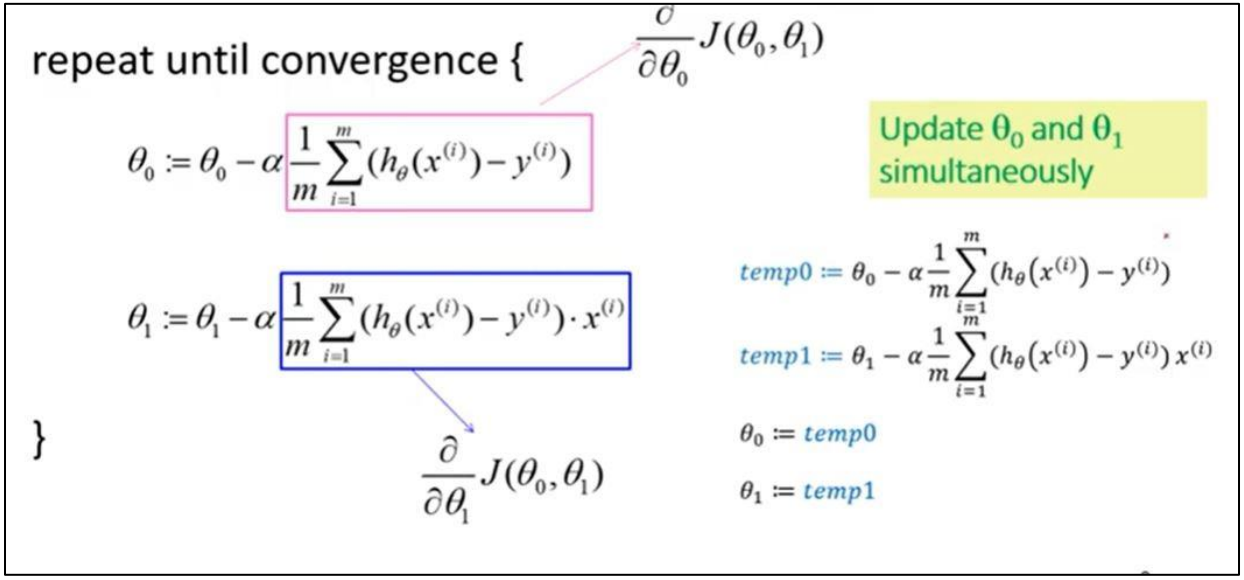


Figure 4-8. Gradient Descent with various learning rates

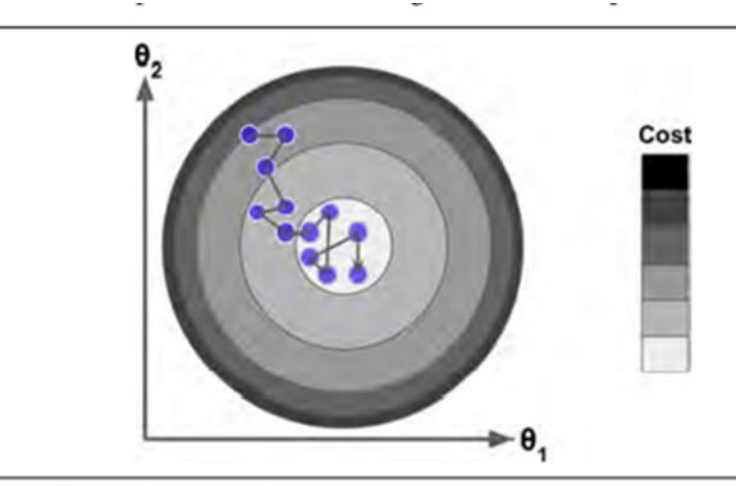
Gradient Descent: Variants

- **Batch** gradient descent refers to calculating the derivative from all training data before calculating an update.
- **Minibatch** refers to calculating derivative of mini groups of training data before calculating an update.
- **Stochastic** gradient descent refers to calculating the derivative from each training data instance and calculating the update immediately

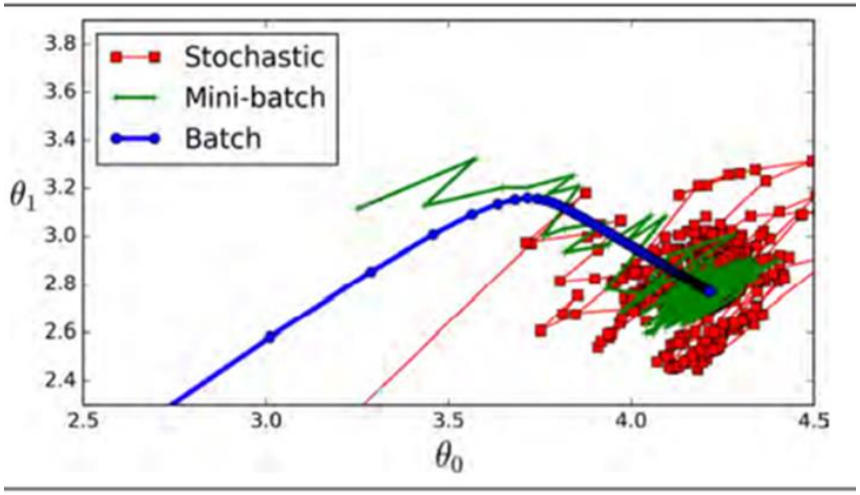


Gradient Descent: Variants

- **Batch** gradient descent refers to calculating the derivative from all training data before calculating an update.
- **Minibatch** refers to calculating derivative of mini groups of training data before calculating an update.
- **Stochastic** gradient descent refers to calculating the derivative from each training data instance and calculating the update immediately



Stochastic Gradient Descent



1. Gradient Descent paths in parameter space

Fit a linear Regression Line :

Gradient Descent

innovate achieve lead

Steps :

(Assuming : 'n' no. of instances and two predictors {x1, x2} and linear regression)

1. Identification of the equations y = w0+w1X1+W2X2
2. Cost function & derivative
    1. W0` = w0 – 1/n * learning rate * (sum (w0+w1X1+W2X2 - y))
    2. W1` = w1 – 1/n * learning rate * (sum (w0+w1X1+W2X2 - y) * x1)
    3. W2` = w2 – 1/n * learning rate * (sum (w0+w1X1+W2X2 - y)*x2)
3. Apply the equations

Fit a linear Regression Line :

Gradient Descent

Fit a linear regression. Show only the first iteration of Gradient descent algorithm using learning rate of **0.02** for the following data , if the Relative Risk of Coronary Heart Disease is believed to be only linearly dependent on BMI as well as Diastolic Pressure. Assume the intercept of the regression model as **5** and the slope of independent variables as **-0.03 (negative)**.

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

Steps :

1. Identification of the equations y = w0+w1X1+W2X2
2. Cost function & derivative
    1. W0` = w0 – 1/3 * learning rate * (sum (w0+w1X1+W2X2 - y))
    2. W1` = w1 – 1/3 * learning rate * (sum (w0+w1X1+W2X2 - y) * x1)
    3. W2` = w2 – 1/3 * learning rate * (sum (w0+w1X1+W2X2 - y)*x2)
3. Apply the equations

Fit a linear Regression Line :

Gradient Descent

Fit a linear regression. Show only the first iteration of Gradient descent algorithm using learning rate of **0.02** for the following data , if the Relative Risk of Coronary Heart Disease is believed to be only linearly dependent on BMI as well as Diastolic Pressure. Assume the intercept of the regression model as **5** and the slope of independent variables as **-0.03 (negative)**.

Patient	Systolic Pressure mm Hg	Diastolic Pressure mm Hg	BMI	Waist Circumference Threshold cm	RR-CHD (Relative Risk of Coronary Heart Disease)
1	140	80	35	100	1.81
2	120	80	25	80	1.22
3	130	100	30	60	1.71

Steps :

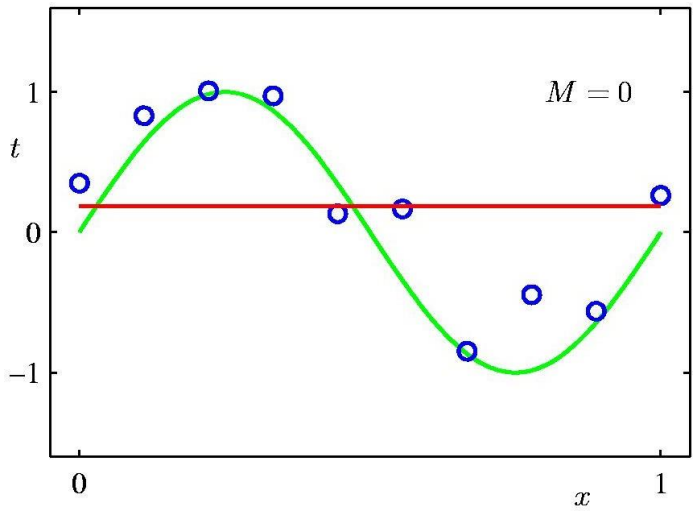
1. Identification of the equations: RR-CHD = 5 – 0.03 * BMI – 0.03 * DiastolicPressure
2. Cost function & derivative
   1. W0` = w0 – 1/3 * 0.02 * (sum (5-0.03BMI-0.03DiastolicPresure - RRCHD))
      = 5 – 1/3 * 0.02 * (sum (5-0.03BMI-0.03DiastolicPresure - RRCHD))
   2. W1` = w1 – 1/3 * 0.02 * (sum (5-0.03BMI-0.03DiastolicPresure - RRCHD) * BMI)
      = -0.03 – 1/3 * 0.02 * (sum (5-0.03BMI-0.03DiastolicPresure - RRCHD) * BMI)
   3. W2` = w2 – 1/3 * 0.02 * (sum (5-0.03BMI-0.03DiastolicPresure - RRCHD) * DiastolicPresure)
      = -0.03 – 1/3 * 0.02 * (sum (5-0.03BMI-0.03DiastolicPresure - RRCHD) * DiastolicPresure)
3. Apply the equations : Answer at the end of first iteration:
   W0 = 5.0016 , W1 = 0.0476, w2= 0.179

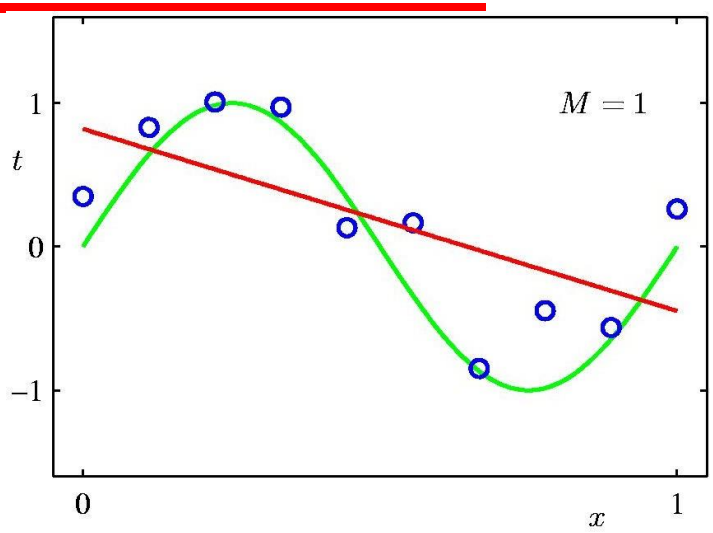
Closed Form Solution Vs. Gradient Descent

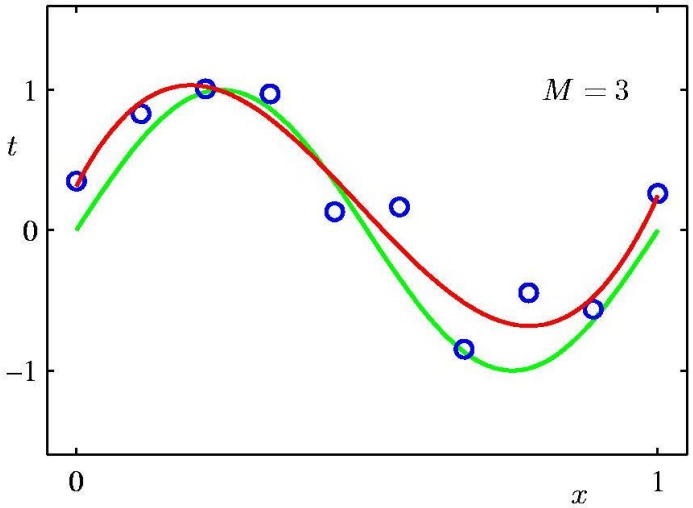
Gradient Descent	Closed Form Solution
• Requires multiple iterations • Need to choose α • Works well when n is large • Can support incremental learning	• Non-iterative • No need for α • Slow if n is large – Computing $(X^T X)^{-1}$ is roughly $O(n^3)$

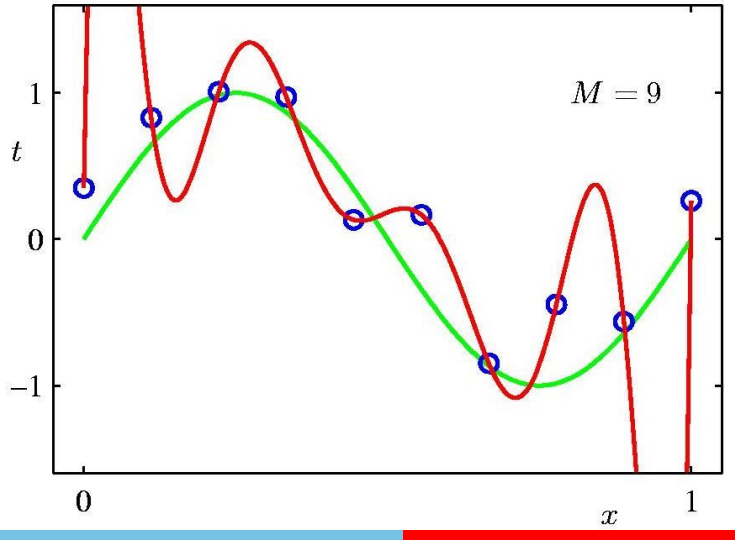
Linear Basis Models

Polynomial Regression









Linear Basis Function Models

- The inputs **X** for linear regression can be:
  - Original quantitative inputs
  - Transformation of quantitative inputs
    - e.g. log, exp, square root, square, etc.
  - Polynomial transformation
    - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
  - Basis expansions
  - Dummy coding of categorical inputs
  - Interactions between variables
    - example: $x_3 = x_1 \cdot x_2$

This allows use of linear regression techniques to fit non-linear datasets.

X No.of.Years of Experience (in Years)	X^2	Y Salary Of the Employee (in Lakhs)
1	1	2
2	4	3
3	9	4
4	16	5
5	25	6

X1 = Graduate	X2 = PostGraduate	X3 = Others	Y Salary Of the Employee
0	0	1	2
1	0	0	3
0	0	1	4
0	1	0	5
1	0	0	6

Linear Basis Function Models

X No.of.Years of Experience (in Years)	X^2	Y Salary Of the Employee (in Lakhs)
1	1	2
2	4	3
3	9	4
4	16	5
5	25	6

Example: an M-th order polynomial function of one dimensional feature x:

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j x^j$$

where x^j = j-th power of x

$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x})$$

$\Phi_j(x)$ are known as *basis functions*. Typically, $\Phi_0(x) = 1$, so that w_0 acts as a bias.

In the simplest case, we use linear basis functions : $\Phi_d(x) = x_d$.

They are called linear models because this function is linear in **w**.

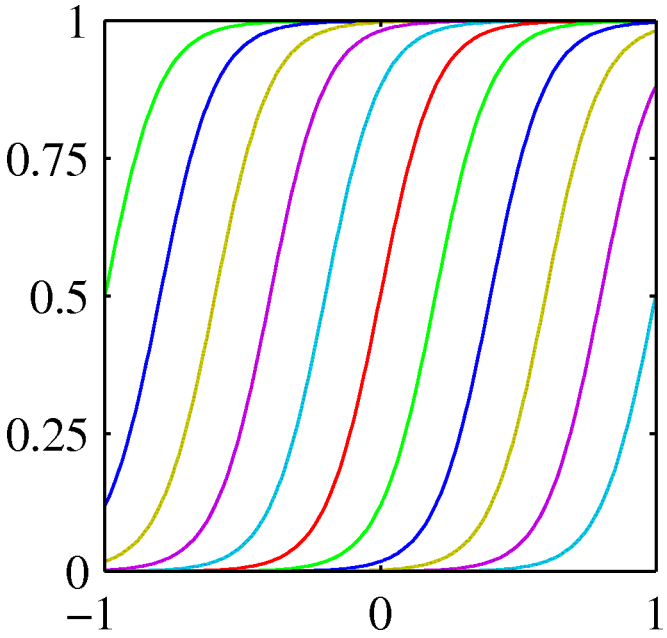
Linear Basis Function Models - Examples

Sigmoidal basis functions:

where $$\phi_j(x) = \sigma\left(\frac{x - \mu_j}{s}\right)$$

$$\sigma(a) = \frac{1}{1 + \exp(-a)}.$$

Also these are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (slope).

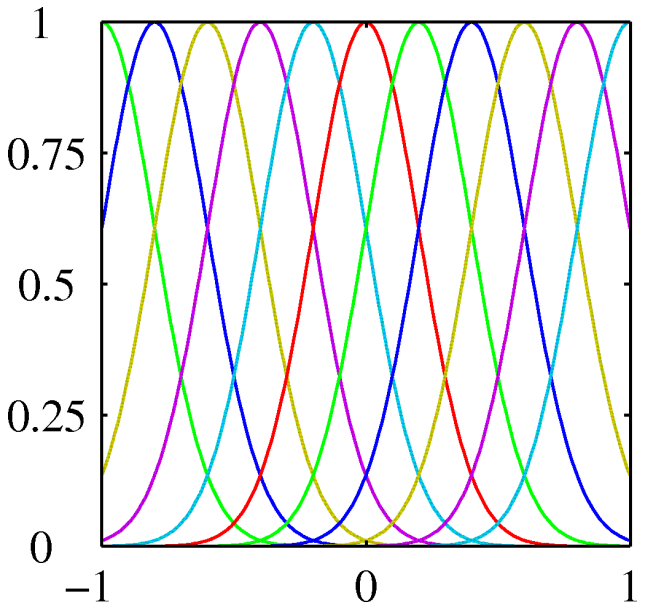


Linear Basis Function Models - Examples

Gaussian basis functions:

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).

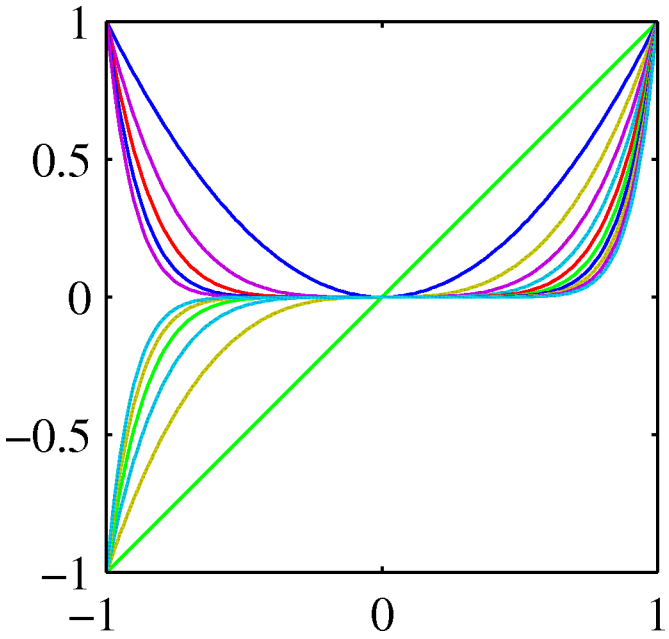


Linear Basis Function Models - Examples

Polynomial basis functions:

$$\phi_j(x) = x^j.$$

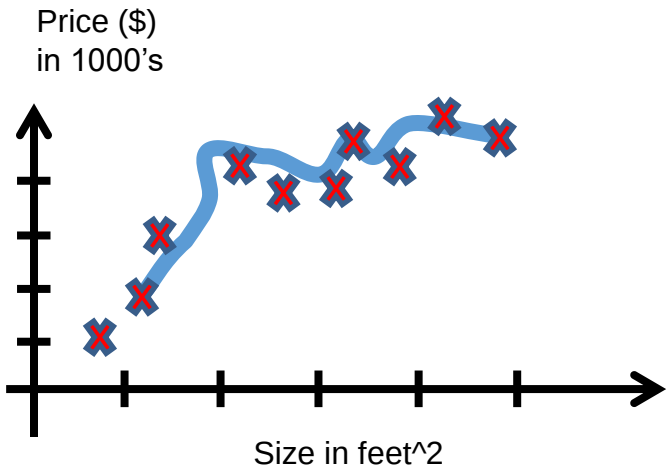
These are global; a small change in *x* affect all basis functions.



Notion of Bias - Variance

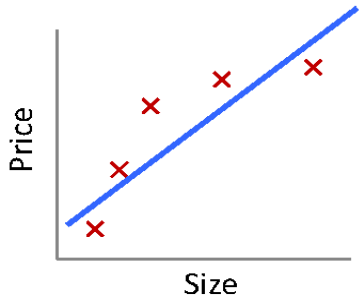
Addressing overfitting

- $x_1 =$ 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}



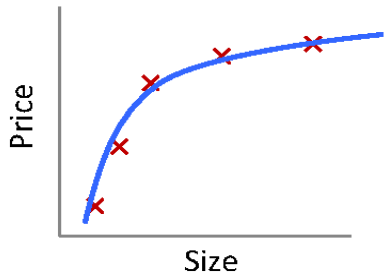
Slide credit: Andrew Ng

Quality of Fit



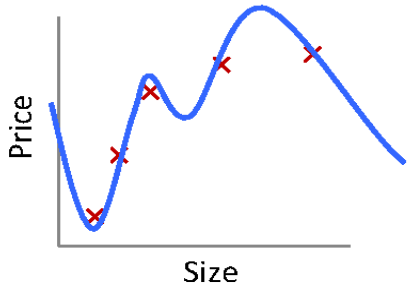
$\theta_0 + \theta_1 x$

Underfitting
(high bias)



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

Correct fit



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

Overfitting
(high variance)

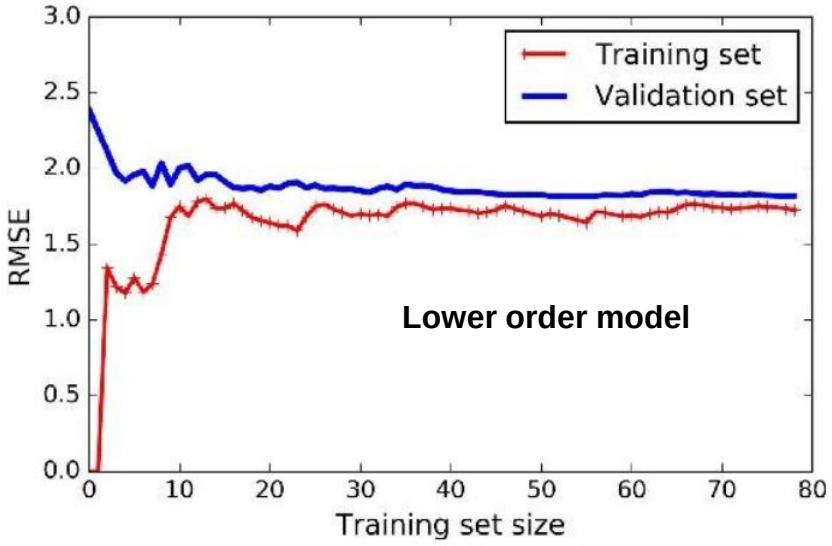
Overfitting:

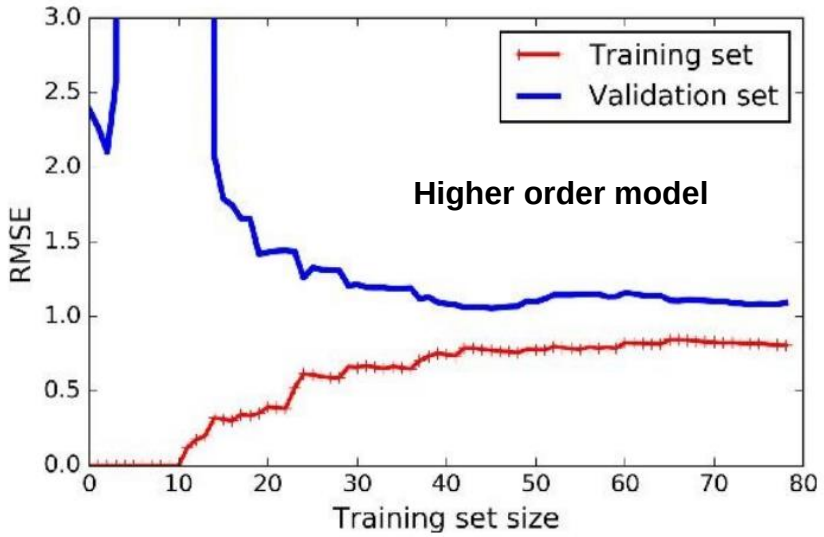
- The learned hypothesis may fit the training set very well ($J(\boldsymbol{\theta}) \approx 0$)
- ...but fails to generalize to new examples

Based on example by Andrew Ng

Effect of Training Size on Over fitting

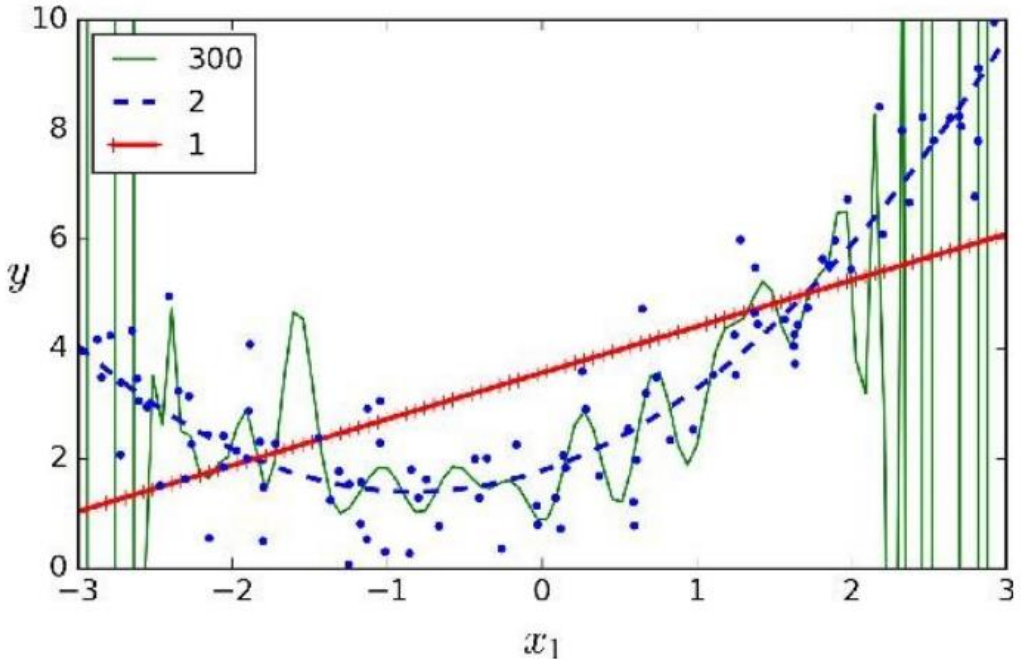
- Size of training dataset needs to be large to prevent when higher order model is used.





Polynomial Fitting can lead to Over fitting

- Underlying target function is quadratic
- Linear model results in under fitting with large bias
- Polynomial of order 300 results in a large variance



Regularization

Regularization

- A method for automatically controlling the complexity of the learned hypothesis
- **Idea**: penalize for large values of θ_j
  - Can incorporate into the cost function
  - Works well when we have a lot of features, each that contributes a bit to predicting the label
- Can also address overfitting by eliminating features (either manually or via model selection)

Regularization

- Linear regression objective function

$$J(\boldsymbol{\theta}) = \underbrace{\frac{1}{2n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)^2}_{\text{model fit to data}} + \underbrace{\frac{\lambda}{2}\sum_{j=1}^{d}\theta_j^2}_{\text{regularization}}$$

  - λ is the regularization parameter ($\lambda \geq 0$)
  - No regularization on θ_0!

Understanding Regularization

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

- Note that $\sum_{j=1}^{d}\theta_j^2 = \|\boldsymbol{\theta}_{1:d}\|_2^2$
  - This is the magnitude of the feature coefficient vector!

- We can also think of this as:

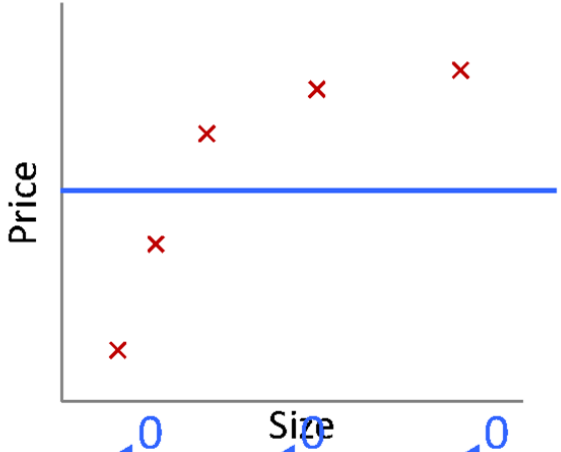
$$\sum_{j=1}^{d}(\theta_j - 0)^2 = \|\boldsymbol{\theta}_{1:d} - \vec{\mathbf{0}}\|_2^2$$

  - L_2 regularization pulls coefficients toward 0

Understanding Regularization

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

- What happens if we set λ to be huge (e.g., 10^{10})?



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

0 0 0 0

Based on example by Andrew Ng

Regularized Linear Regression

- Cost Function

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

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

- Gradient update:

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

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

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

$$\theta_j \leftarrow \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)} - \lambda\theta_j$$

- We can rewrite the gradient step as:

$$\theta_j \leftarrow \theta_j\left(1 - \alpha\lambda\right) - \alpha\frac{1}{n}\sum_{i=1}^{n}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)x_j^{(i)}$$

Thank you !