Machine Learning Fundamentals – DTSC102

Lecture 5
Linear Regression

Course Instructor: Dr.-Ing. Maggie Mashaly
maggie.ezzat@guc.edu.eg
C3.220

Contents

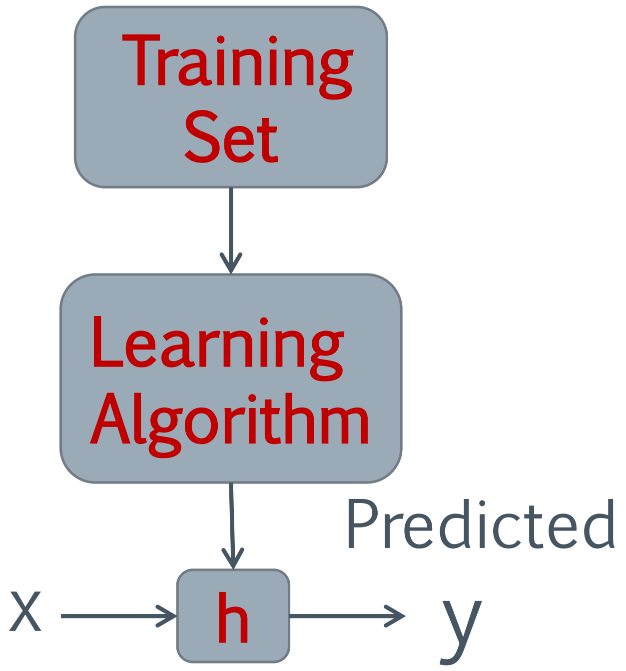
- Supervised Learning
- Linear Regression
- Method of Gradient Descent
- Parameter Learning

Supervised Learning

- Learning a function *h: x →y* such that *h(x)* is a good predictor for the corresponding value of *y*
- *h* is called the hypothesis
- How do we represent *h* ??

In its simplest form:

$$h(x) = \theta_0 + \theta_1 x$$



where h(x) is some straight-line function of x, and h(x) is a continuous output

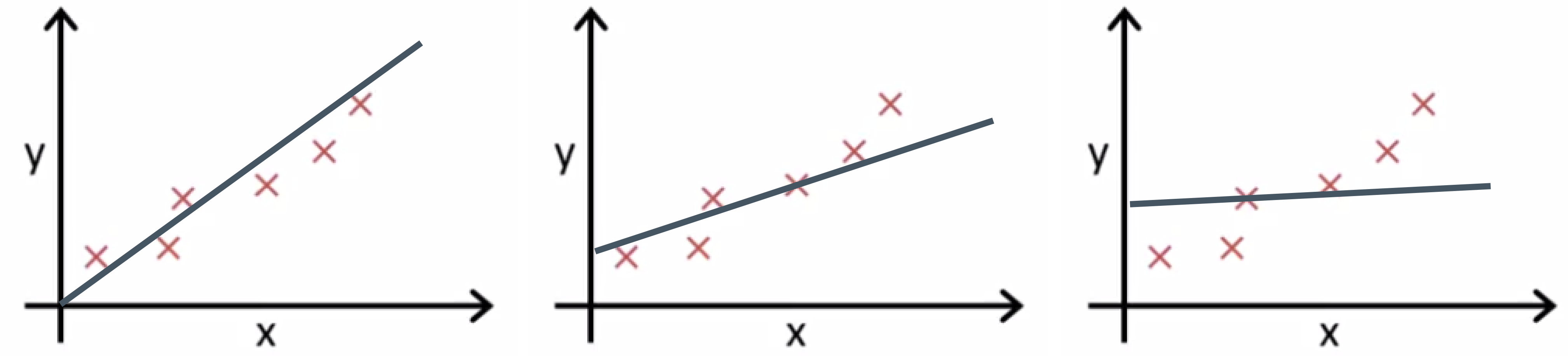
- That is Linear Regression with One Variable

Supervised Learning

Linear Regression with One Variable

$$h(x) = \theta_0 + \theta_1 x$$

Where θ_0, θ_1 are the function parameters



- How to choose $\theta_i' s$??

Dr.- Ing. Maggie Mashaly

Supervised Learning

Linear Regression with One Variable

$$h(x) = \theta_0 + \theta_1 x$$

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

- i.e.: Minimize the difference between $h(x_i)$ and y_i

Looks like an optimization problem…

- Objective: Minimizing cost function
- Cost function: average difference between $h(x_i)$ and y_i
- Optimize with respect to: θ_0, θ_1

Dr.- Ing. Maggie Mashaly

Linear Regression with One Variable

Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Notes:

- $J(\theta_0, \theta_1)$ is defined as a “Mean Square Error Function”; most commonly used for regression problems
- Division by sample size m to get the mean error
- Function is halved as a convenience for the computation of the gradient descent, as the derivative term of the square function will cancel out the (1/2) term.

Optimization Problem

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

Linear Regression with One Variable

BUT Mind the difference between $h_\theta(x)$ and $\mathrm{J}(\theta_0, \theta_1)$.....

- Consider a simple example where $\theta_0 = 0$
  - Hypothesis: $h_\theta(x) = \theta_1 x$
  - Parameter: θ_1
  - Cost Function: $\mathrm{J}(\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2$
  - Goal: Minimize $\mathrm{J}(\theta_1)$

$h_\theta(x)$
(for fixed θ_1, this is a function of x)

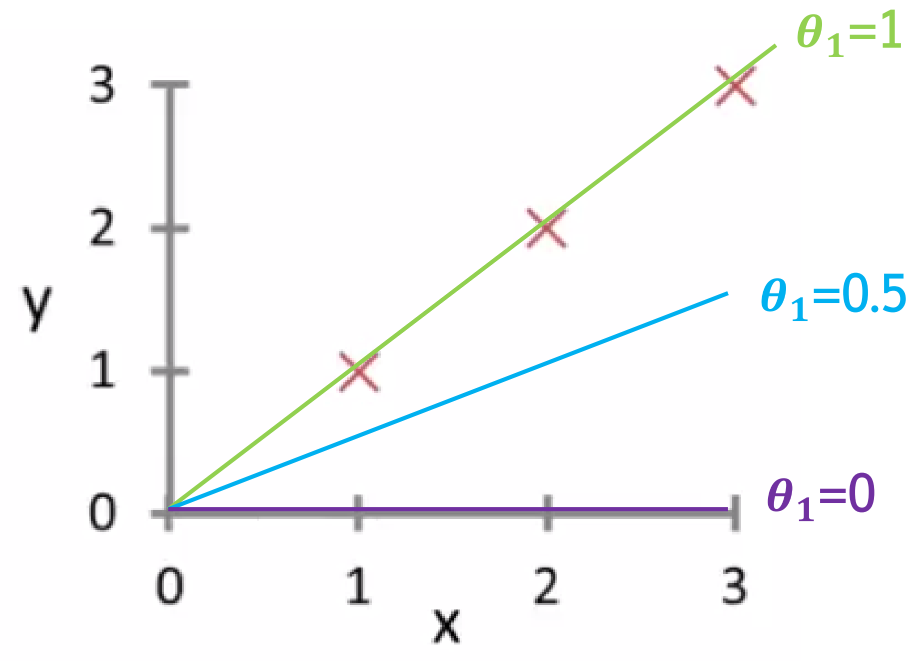
$\mathrm{J}(\theta_1)$
(function of the parameter θ_1)

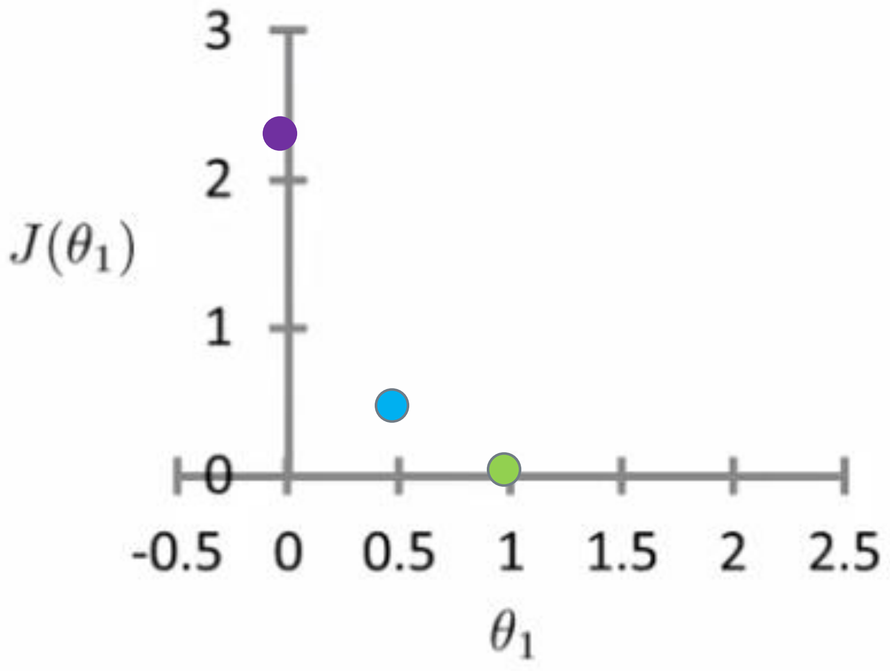
Dr.- Ing. Maggie Mashaly

Linear Regression with One Variable

$h_\theta(x)$
(for fixed θ_1, this is a function of x)

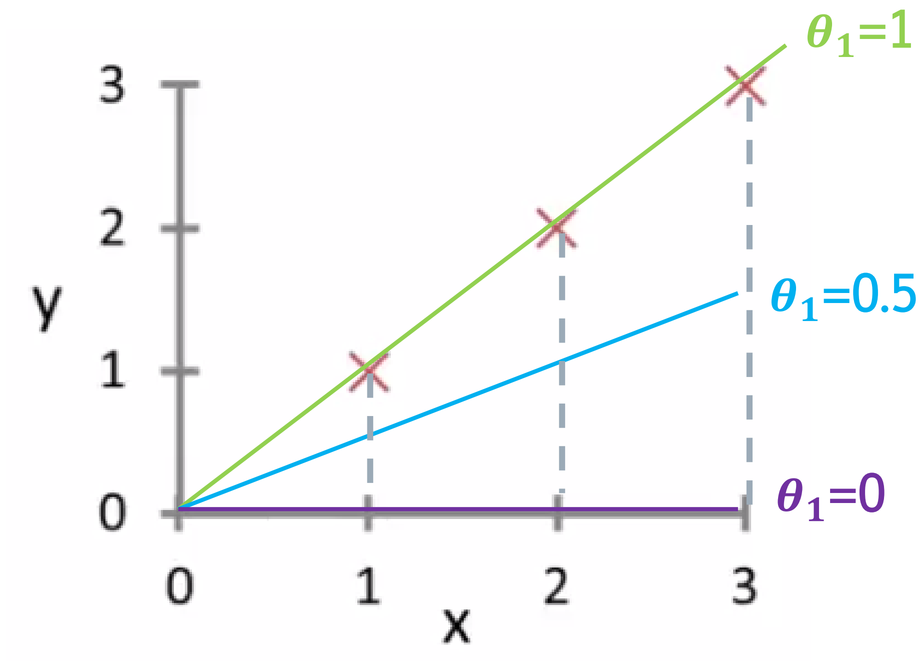
$J(\theta_1)$
(function of the parameter θ_1)



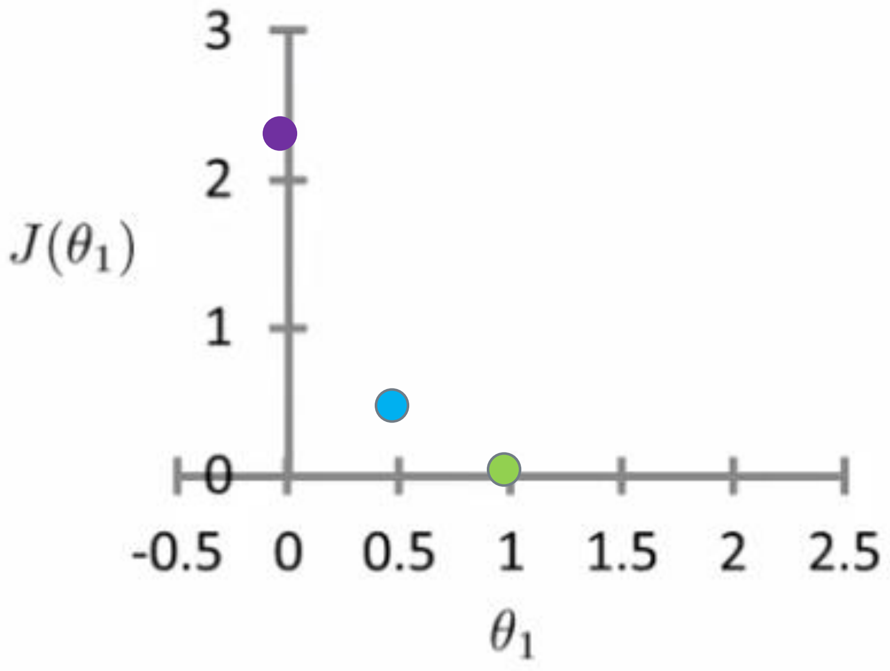


Linear Regression with One Variable

$h_\theta(x)$
(for fixed θ_1, this is a function of x)



$\mathrm{J}(\theta_1)$
(function of the parameter θ_1)



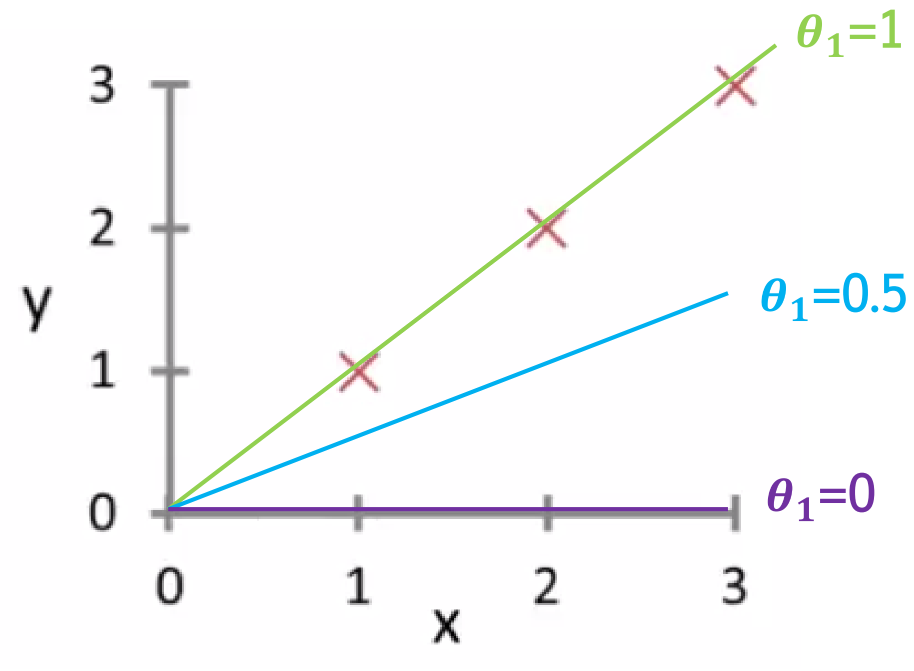
Linear Regression with One Variable

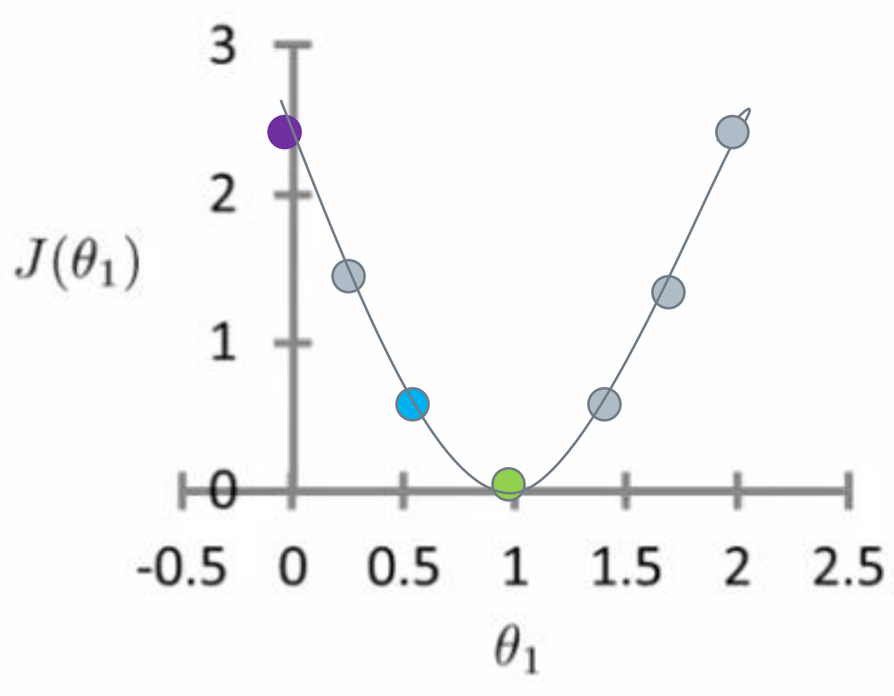
$h_\theta(x)$

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

$J(\theta_1)$

(function of the parameter θ_1)



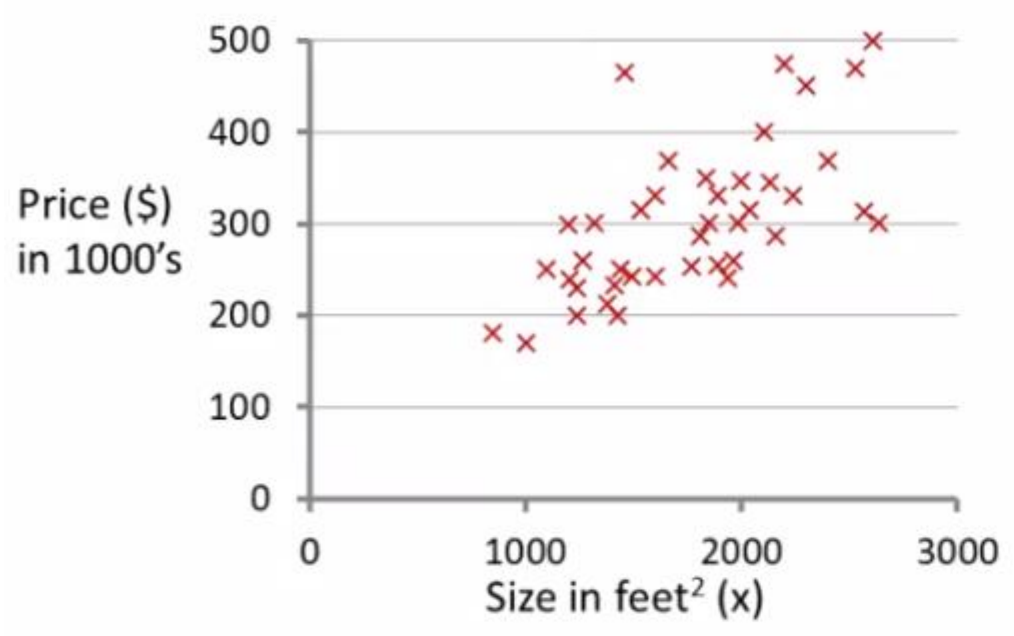


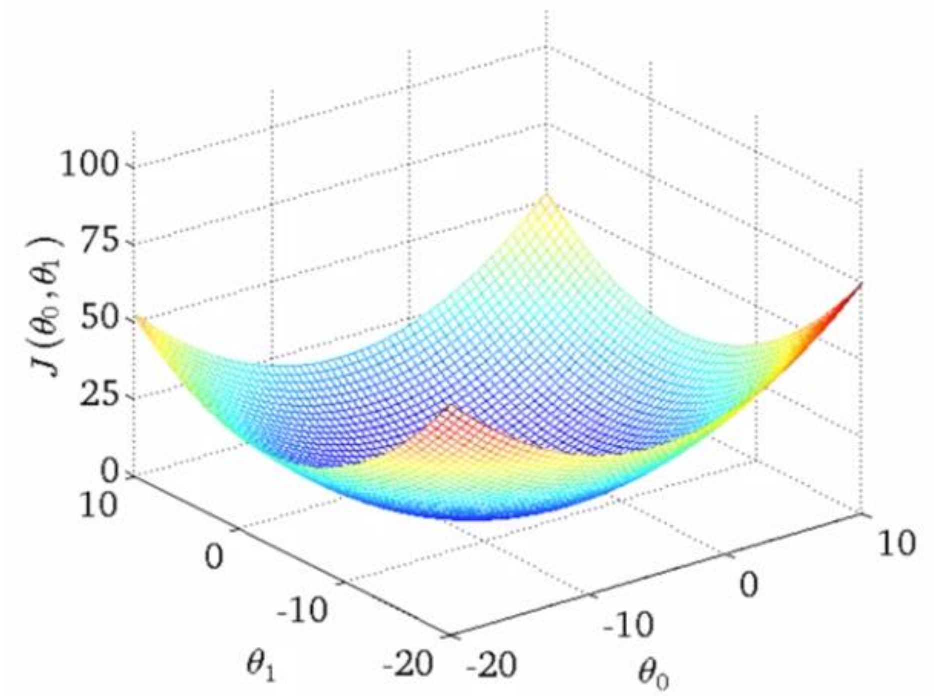
Linear Regression with One Variable

Now consider an example where $\theta_0, \theta_1 \neq 0$

$h_\theta(x)$
(for fixed θ_0, θ_1,
this is a function of x)

$\mathrm{J}(\theta_0, \theta_1)$
(function of the
parameters θ_0, θ_1)

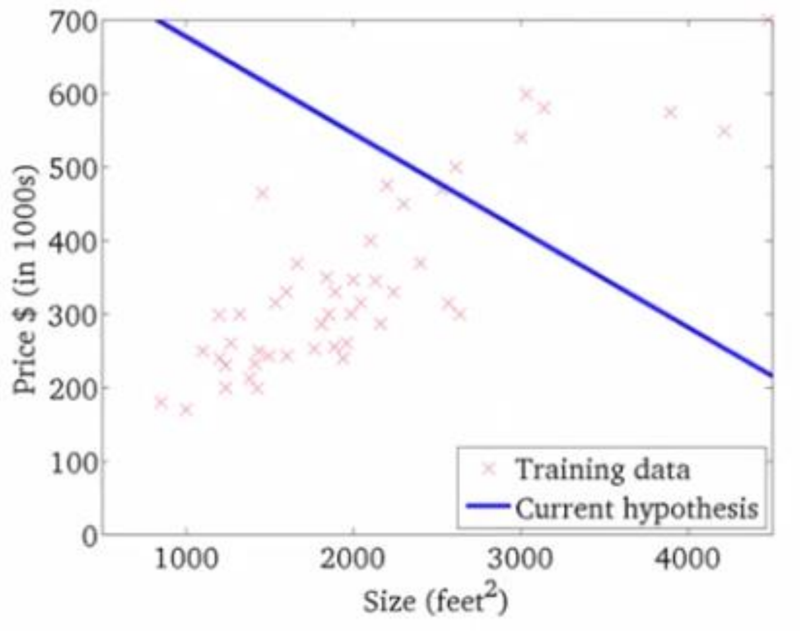




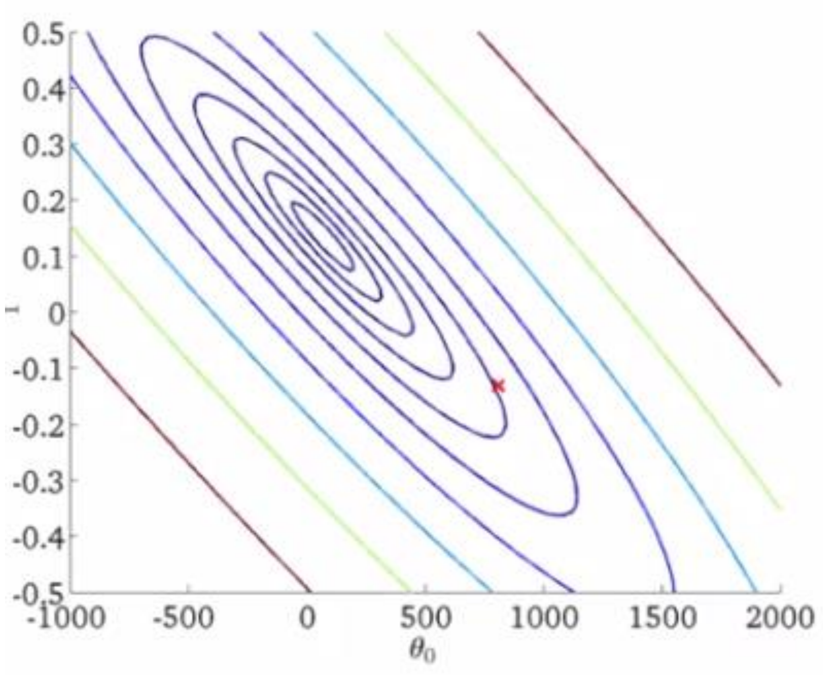
Linear Regression with One Variable

Now consider an example where $\theta_0, \theta_1 \neq 0$

$h_\theta(x)$
(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)



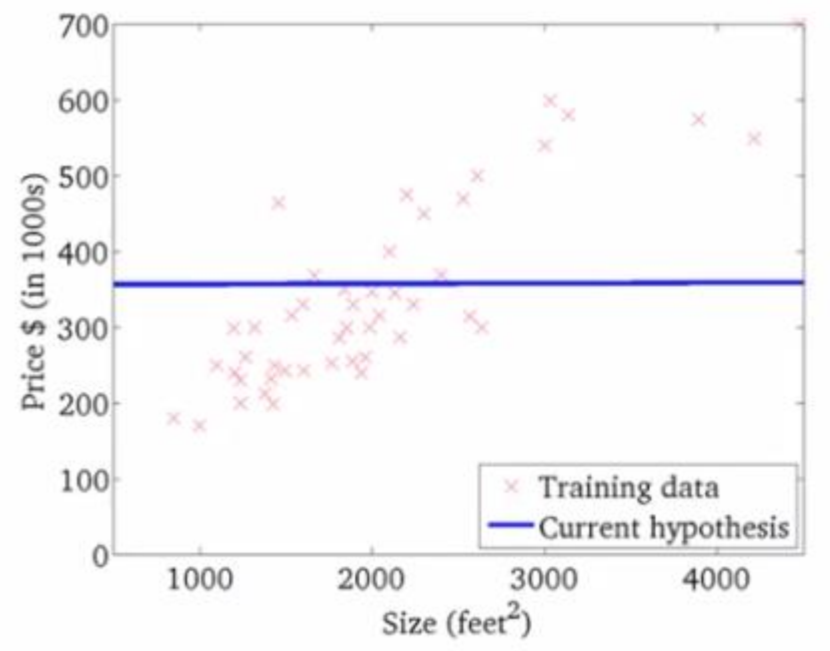
- Each ellipse in the contour plot shows sets of values for θ_0, θ_1that take on the same value for $J(\theta_0, \theta_1)$

Dr.- Ing. Maggie Mashaly

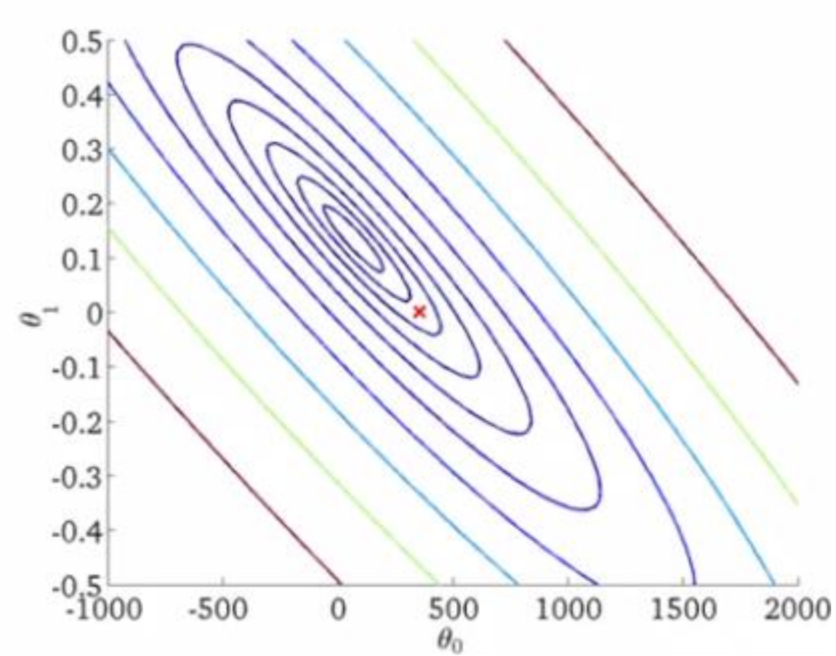
Linear Regression with One Variable

Now consider an example where $\theta_0, \theta_1 \neq 0$

$h_\theta(x)$
(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)



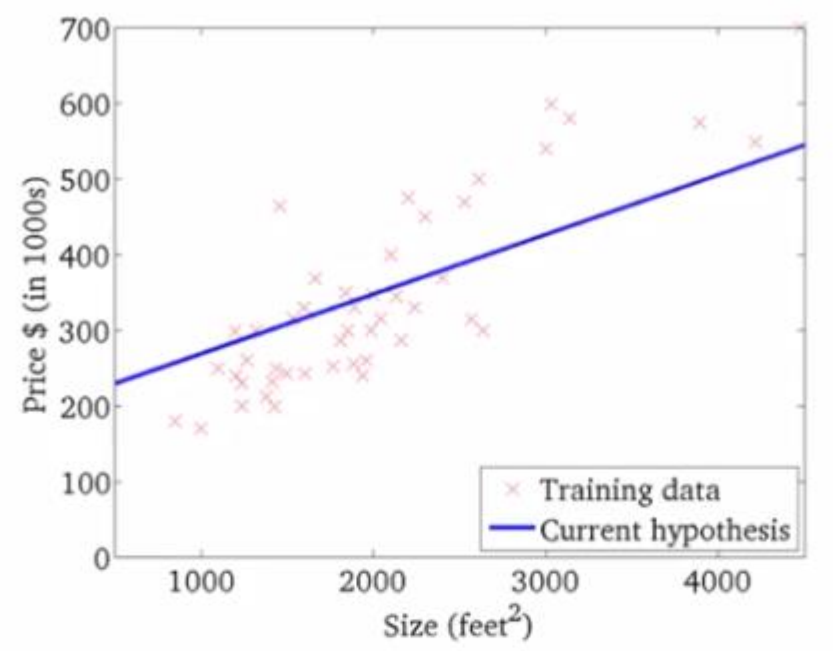
- Each ellipse in the contour plot shows sets of values for θ_0, θ_1 that take on the same value for $J(\theta_0, \theta_1)$

Dr.- Ing. Maggie Mashaly

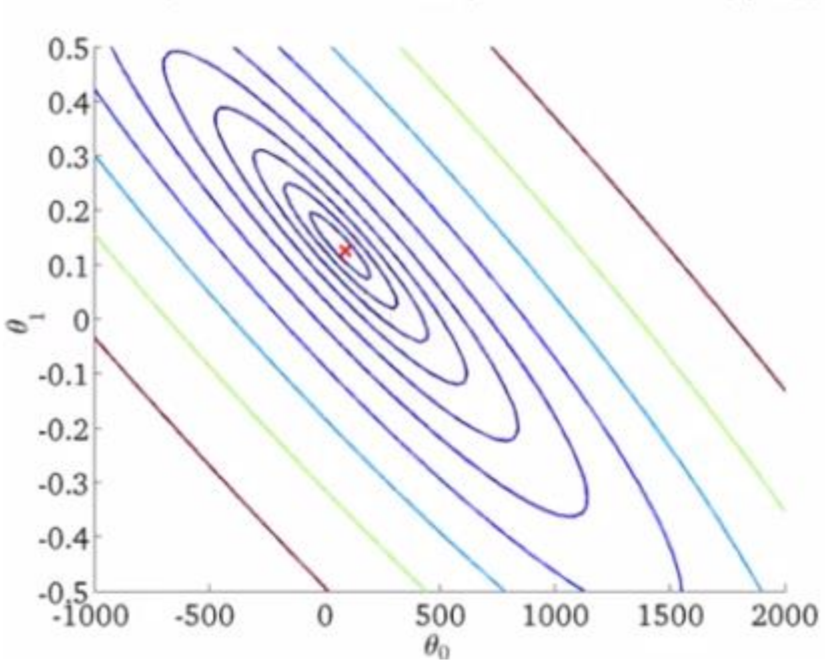
Linear Regression with One Variable

Now consider an example where $\theta_0, \theta_1 \neq 0$

$h_\theta(x)$
(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)



- Each ellipse in the contour plot shows sets of values for θ_0, θ_1that take on the same value for $J(\theta_0, \theta_1)$

Dr.- Ing. Maggie Mashaly

Parameter Learning

So how can we find values for θ_0, θ_1 to minimize the cost function $J(\theta_0, \theta_1)$??

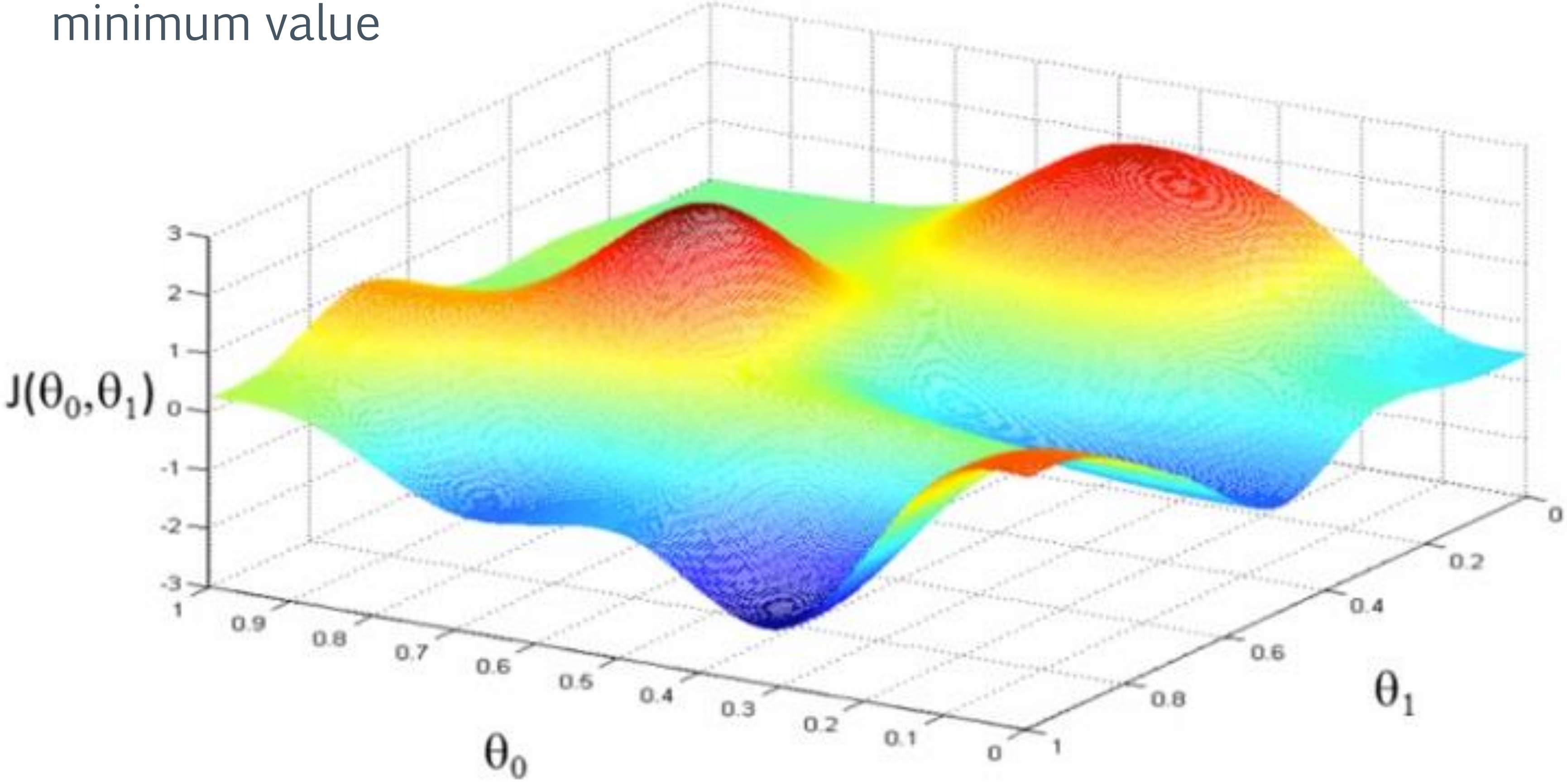
- Using the method of **Gradient Descent**
  - Simplest algorithm for unconstrained programming
  - Also known as Steepest Descent

- How does Gradient Descent work?
  - 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 value

Parameter Learning

Gradient Descent

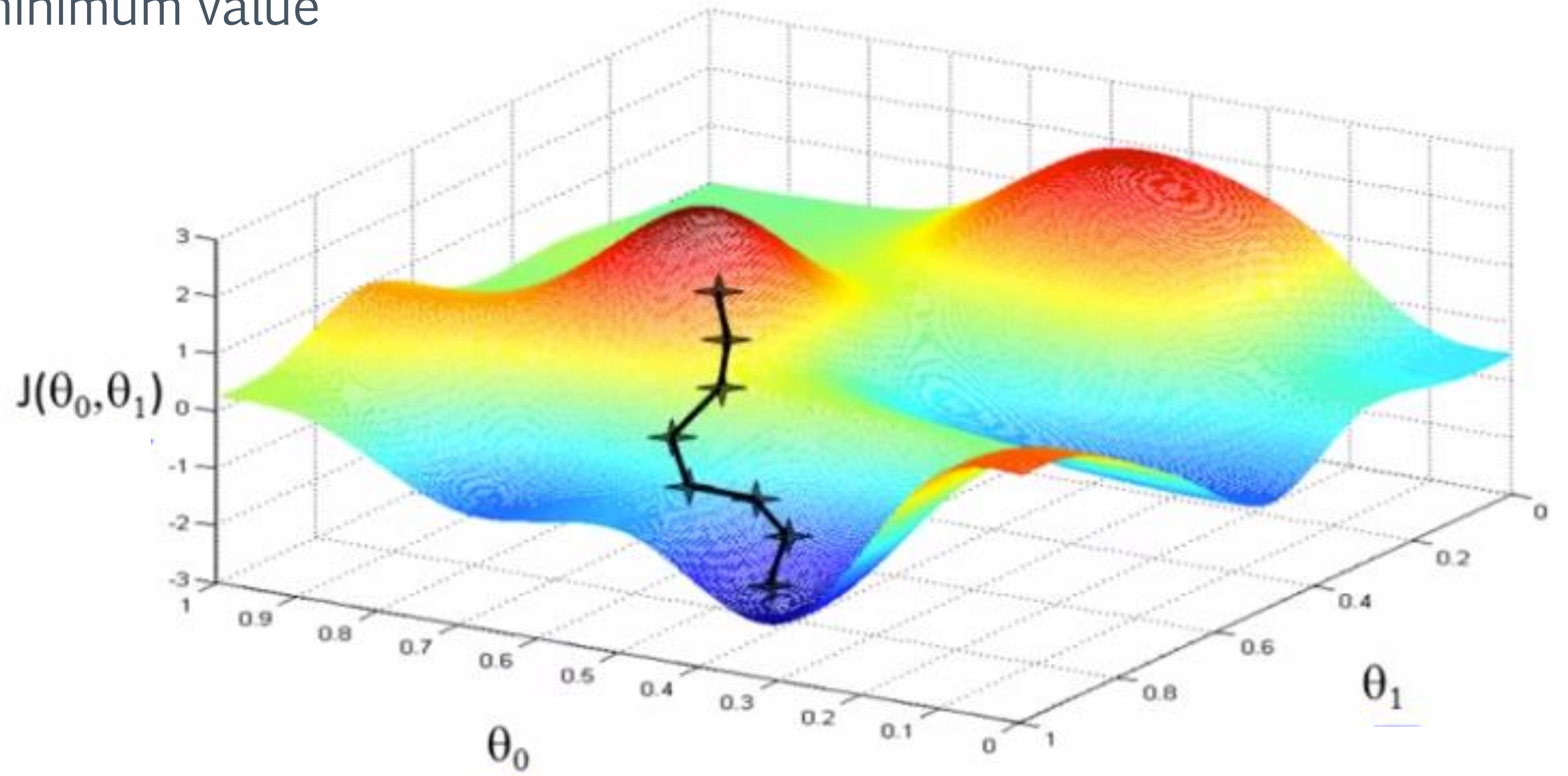
- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum value



Parameter Learning

Gradient Descent

- 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 global minimum value

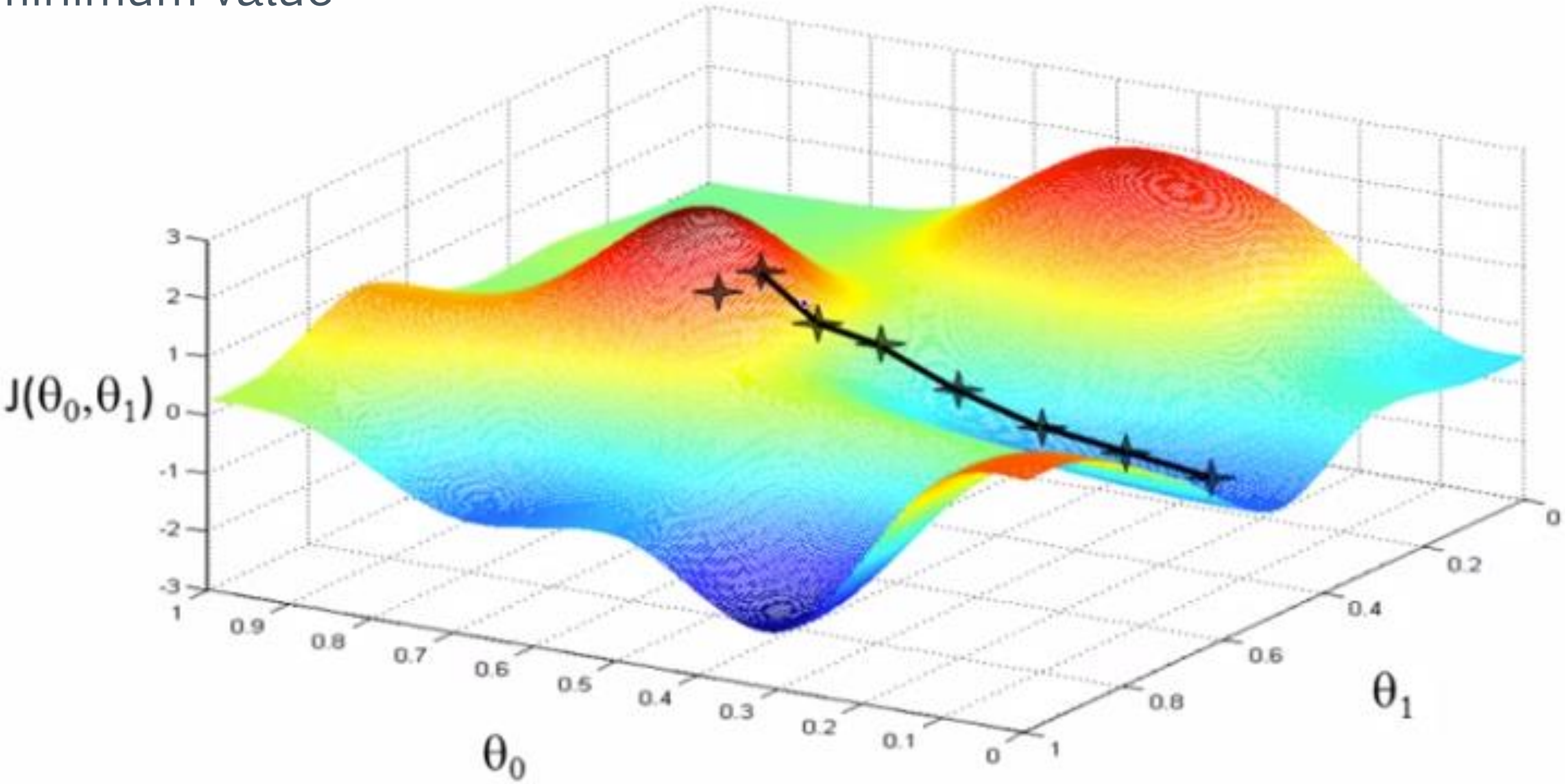


Dr.- Ing. Maggie Mashaly

Parameter Learning

Gradient Descent

- Start with some θ_0, θ_1(Your beginning value makes a big difference!)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a global minimum value



Dr.- Ing. Maggie Mashaly

Parameter Learning

Gradient Descent Algorithm

Repeat until convergence

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

where:

- α is the learning rate which controls the sizes of steps we take; i.e.: large α implies aggressive gradient descent with large steps and small α implies taking small steps
- $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ is the rate of change of $J(\theta_0, \theta_1)$ with respect to θ_j
- θ_0, θ_1 must be updated simultaneously

Parameter Learning

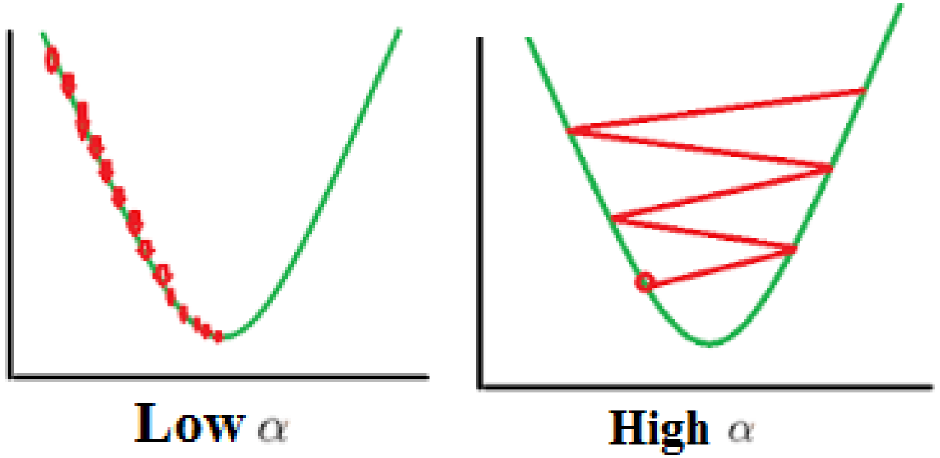
Gradient Descent Algorithm

Repeat until convergence

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

How to select α:

- 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



Dr.- Ing. Maggie Mashaly

GUC German University in Cairo

Parameter Learning

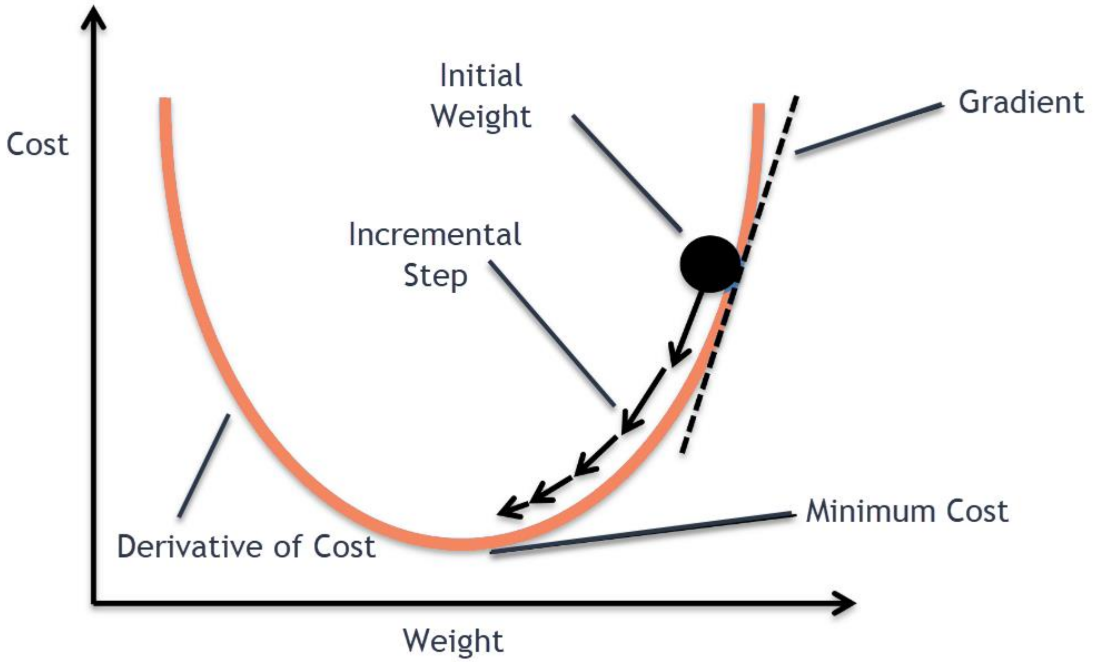
Gradient Descent Algorithm

Repeat until convergence

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

How to select α:

- And anyway, as we approach a local minimum, the gradient term $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ will be come smaller, and gradient descent will automatically take smaller steps. So no need to decrease α over time



Dr.- Ing. Maggie Mashaly

Parameter Learning

Gradient Descent Algorithm

Repeat until convergence

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

Important to know:

- This gradient descent on cost function $J(\theta_0, \theta_1)$ is called Batch Gradient Descent as it looks at every example in the entire training set on every step
- J is a convex quadratic function
- While gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local, optima; thus gradient descent always converges (assuming the learning rate α is not too large) to the global minimum.

Gradient Descent for Linear Regression

Linear Regression Model

$$h(x) = \theta_0 + \theta_1 x$$

$$\mathrm{J}(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

Gradient Descent Algorithm

Repeat until convergence

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

for j=0 & j=1

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

For j=0 : $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x - y_i)$

For j=1 : $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x - y_i) \,.\, x_i$

Gradient Descent for Linear Regression

Algorithm

Repeat until convergence:

$$\{\theta_0 = \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x - y_i) \ ,$$

$$\theta_1 = \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (\theta_0 + \theta_1 x - y_i) \, . \, x_i\}$$

And don't forget to update the parameters simultaneously!

Dr.- Ing. Maggie Mashaly

Gradient Descent – Fine Tuning

Now that you understand the algorithm, lets turn into fine tuning details!

1. Scaling Features

It is found that during gradient descent if the features are on the **same scale** then the algorithm converges faster than when the features are not appropriately scaled in the same range.

Solution : Get every feature approximately in the range between $-1 \le x \le 1$

➢ Normalization: Divide each feature by max of the feature column

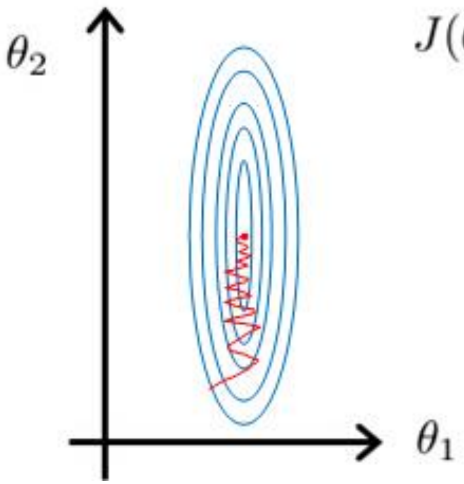
e.g.:

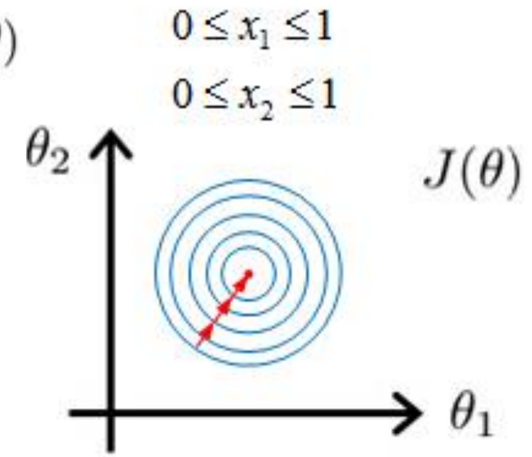
$x_1 = size\ (0 - 2000\ feet^2)$

$x_2 = number\ of\ bedrooms\ (1 - 5)$

$x_1 = size/2000$

$x_2 = number\ of\ bedrooms/5$





Dr.- Ing. Maggie Mashaly

Gradient Descent – Fine Tuning

1. Scaling Features

Another Solution:

- **Mean Normalization:** Replace a feature x_i with $x_i - \mu_i$ so that the approximate mean of the features is 0 which is then normalized. (not applied to x_0)

$$x_i = \frac{x_i - \mu_i}{S_i}$$

e.g.:

$x_1 = size\ (0 - 2000\ feet^2)$

$x_2 = number\ of\ bedrooms\ (1 - 5)$

⇓

$x_1 = (size - 1000)/2000$

$x_2 = (number\ of\ bedrooms - 2)/5$

where:

x_i is the feature value

μ_i is the mean

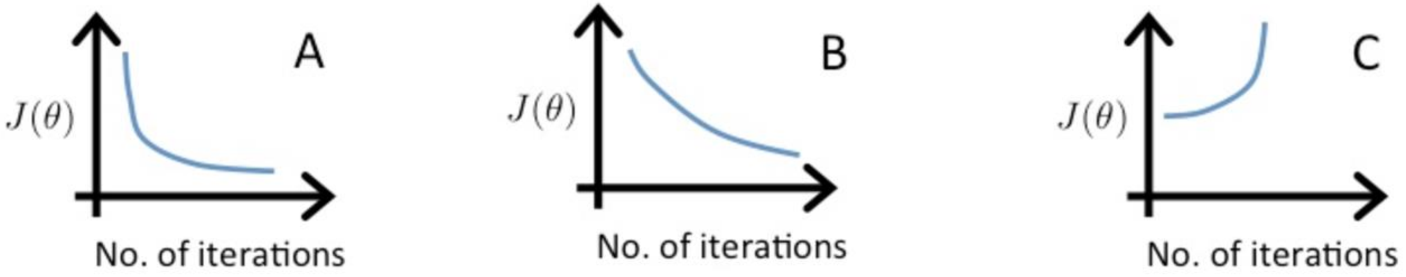
S_i is the standard deviation or the range (min-max)

Dr.- Ing. Maggie Mashaly

Gradient Descent – Fine Tuning

2. Learning Rate α

In order to tell if gradient descent is running well, plot $J(\theta)$ vs number of iterations



- In graph A: α seems to be appropriate as $J(\theta)$ converges soon
- In graph B: $J(\theta)$ is taking too long to converge, thus α must be increased
- In graph C: $J(\theta)$ is increasing, thus α must be decreased

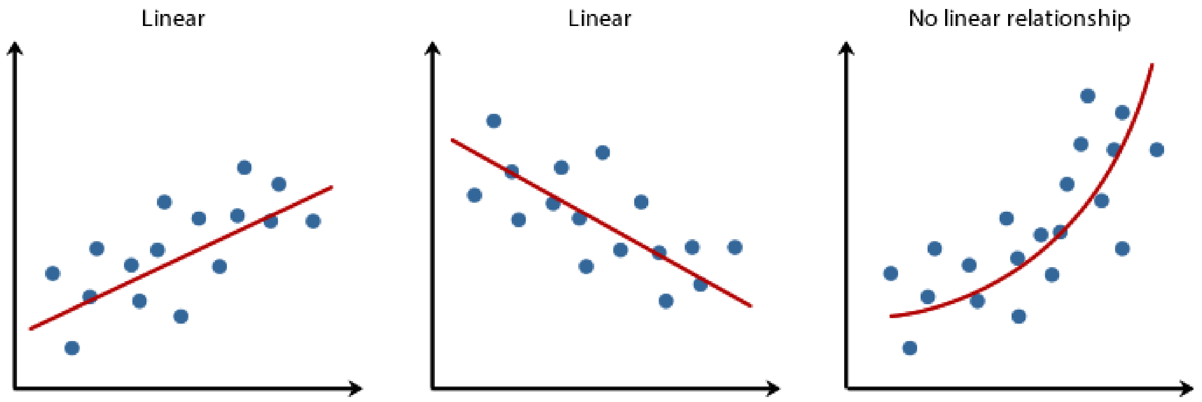
➢ While programming, we typically declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration

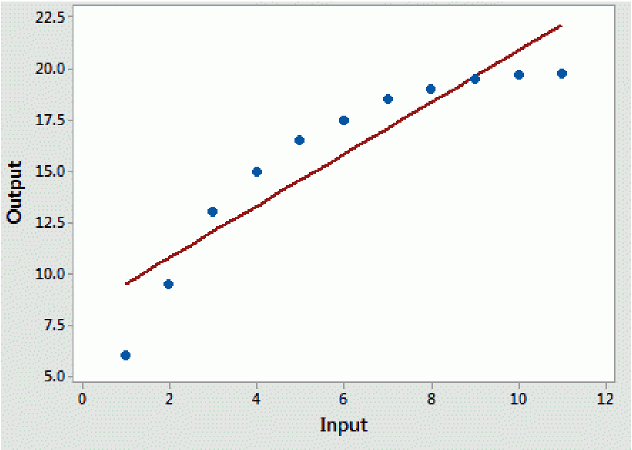
➢ Typical values to try for α are: 0.001, 0.003, 0.01, 0.03, 0.1, 0.3, 1,…

Dr.- Ing. Maggie Mashaly

Linear hypothesis : Not always a good option!

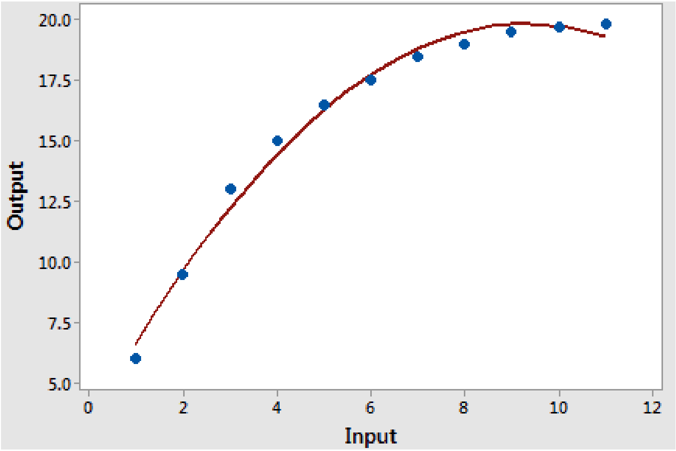
e.g.:





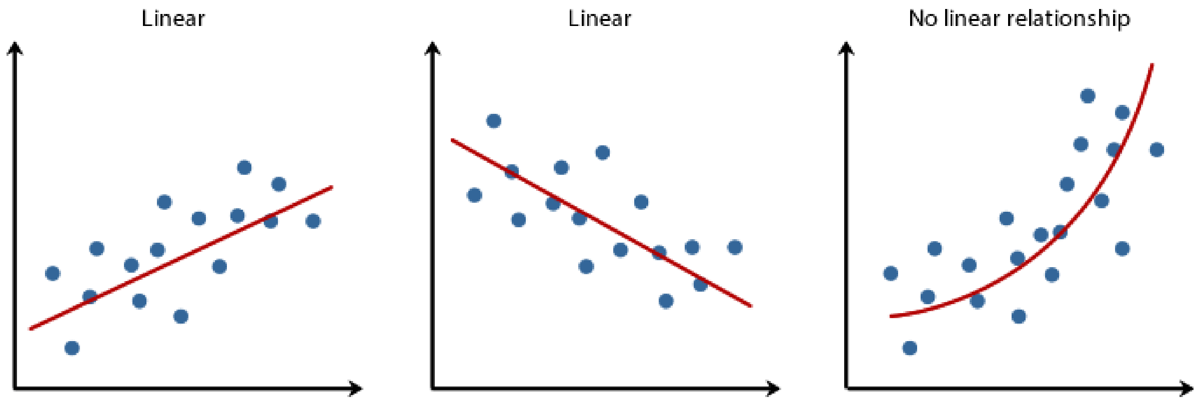
A straight line is not a good representation for the data

Try a quadratic function

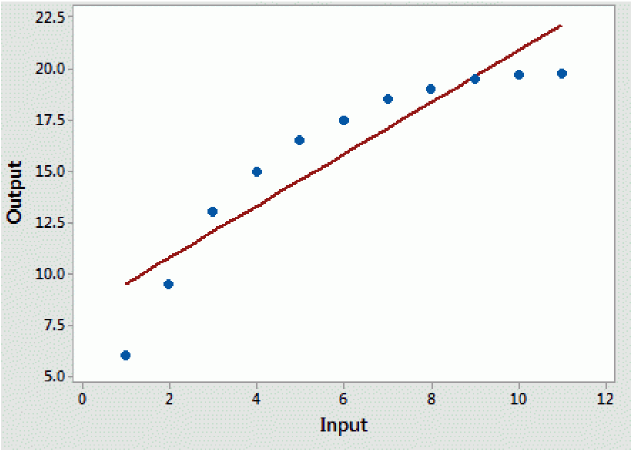


$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

Linear hypothesis : Not always a good option!

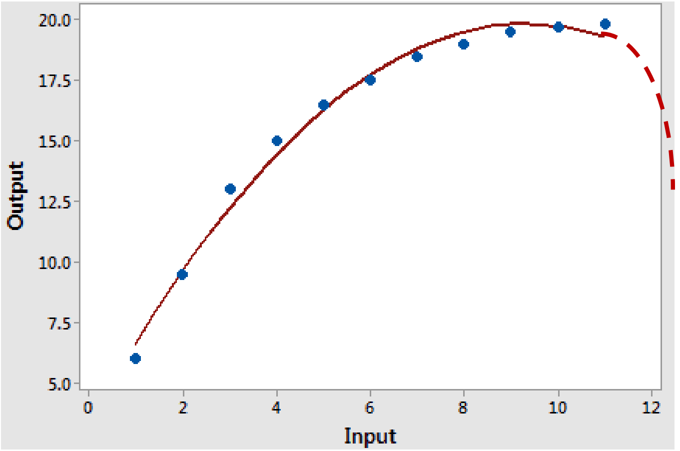


e.g.:



A straight line is not a good representation for the data

Try a quadratic function



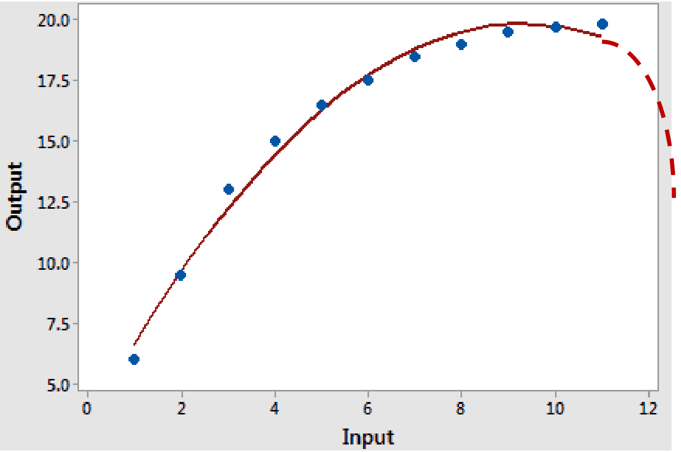
$$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$$

But!
Quadratic function eventually comes down, which is not case with our data…

Dr.- Ing. Maggie Mashaly

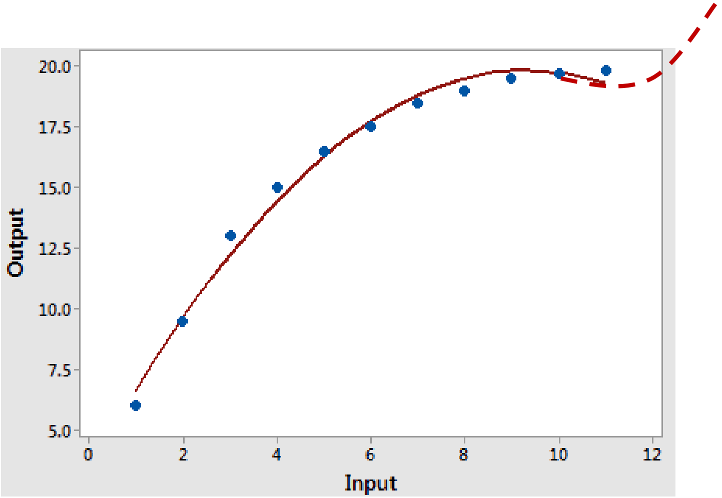
Linear hypothesis : Not always a good option!

e.g.:



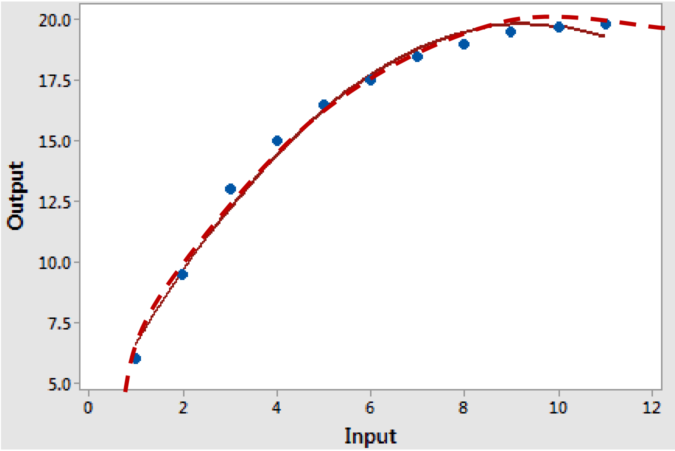
$h(x) = \theta_0 + \theta_1 x + \theta_2 x^2$

Try a cubic function



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

Or a square root function



$h(x) = \theta_0 + \theta_1 x + \theta_2 \sqrt{x}$

The Hypothesis is not a linear function but the regression is still linear because regression is interested in values of θ which is still linear

Dr.- Ing. Maggie Mashaly

Polynomial hypothesis

Considering the choice of a cubic function

- Hypothesis Function:

$h(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

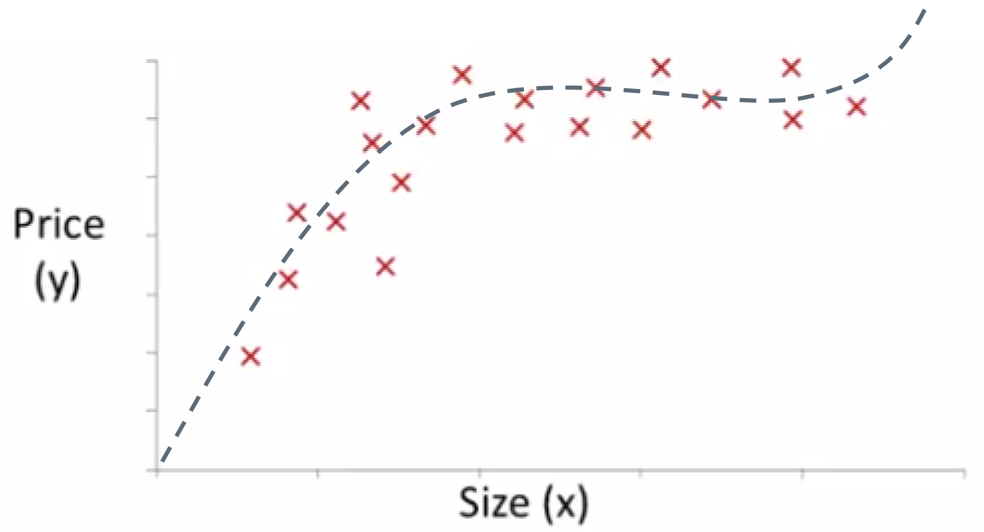
$h(x) = \theta_0 + \theta_1(size) + \theta_2(size)^2 + \theta_3(size)^3$

- Features:

$x_1 = (size)$

$x_2 = (size)^2$

$x_3 = (size)^3$



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

- Important to consider in Polynomial hypothesis: Feature Scaling

➢ If $(size)$=1:1000, then $(size)^2$=1:1000000 and $(size)^3$=1:1000000000

- Then use Gradient Descent to find parameters θ that minimizes J(θ)

….. but is Gradient Descent the only method to solve Regression problems??

Alternative to Gradient Descent

Recall from Calculus how to minimize any function with respect to its parameters?

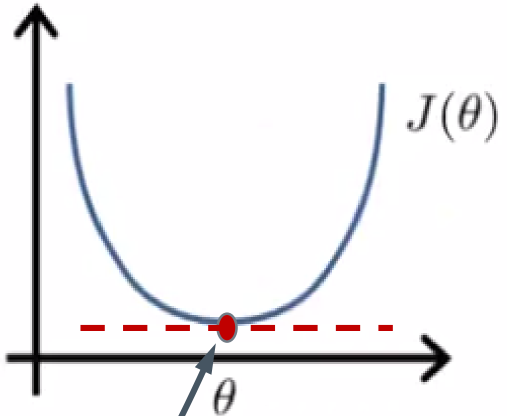
1. Differentiate the function with respect to each parameter
2. Equate derivative to zero to find each parameter's value

- This approach is called: **Normal Equation**
  - A method to solve for $\boldsymbol{\theta}$ analytically
  - Finds optimum parameters without iteration
  - For cost function

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

1. Set $\frac{\partial}{\partial \theta_j} J(\theta) = 0$, for every j
2. Solve for $\theta_0, \theta_1, .., \theta_m$ (m+1 equations)

Or better, solve it using Matrices…



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

Condition for Minimum because the function is convex

Normal Equation

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

$$= \frac{\partial}{\partial\theta_j}\frac{1}{2m}\sum_{i=1}^{m}\left(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)} - y^{(i)}\right)^2$$

$$= \frac{1}{m}\sum_{i=1}^{m}\left(\theta_0 x_0^{(i)} + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)} - y^{(i)}\right)x_j^{(i)}$$

$$= \frac{1}{m}\sum_{i=1}^{m}\Big(\theta_0 \underbrace{x_0^{(i)} x_j^{(i)}}_{\substack{R_{0,j} \\ \text{Correlation}}}\Big) + \frac{1}{m}\sum_{i=1}^{m}\Big(\theta_1 \underbrace{x_1^{(i)} x_j^{(i)}}_{\substack{R_{1,j} \\ \text{Correlation}}}\Big) + \cdots \frac{1}{m}\sum_{i=1}^{m}\Big(\theta_m \underbrace{x_m^{(i)} x_j^{(i)}}_{\substack{R_{m,j} \\ \text{Correlation}}}\Big) - \frac{1}{m}\sum_{i=1}^{m}\Big(\underbrace{y^{(i)} x_j^{(i)}}_{\substack{R_{y,j} \\ \text{Correlation}}}\Big)$$

$$\underbrace{(\boldsymbol{X}^T\boldsymbol{X})}_{R_{xx}}\boldsymbol{\theta} - \underbrace{\boldsymbol{X}^T\boldsymbol{Y}}_{R_{xy}} = \boldsymbol{0} \quad \Rightarrow \quad \boldsymbol{\theta} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$$

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 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

- Add a column for extra feature x_0 with value 1
- Construct Matrix X including all features for training data and Vector Y for price values

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix}, Y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}, \textit{Normal Equation:} \boldsymbol{\theta = (X^T X)^{-1} X^T Y}$$

Dr.- Ing. Maggie Mashaly

Gradient Descent vs. Normal Equation

Assuming m training examples and n features

Gradient Descent	Normal Equation
Need to choose α	No need to choose α
Needs many iterations	Doesn't need to iterate
Needs Feature Scaling	No need for Feature Scaling
Complexity: $O(kn^2)$	Complexity: $O(n^3)$ (due to the inverse operator)
Works well even if n is large i.e.: suitable for $n \geq 10^6$	Slow if n is large

➢ For Normal Equation, $\boldsymbol{X^T X}$ may not always be invertible due to..

1. Redundant Features (linearly dependent) → Eliminate them
2. Too many Features → Delete some features / use Regularization

Dr.- Ing. Maggie Mashaly