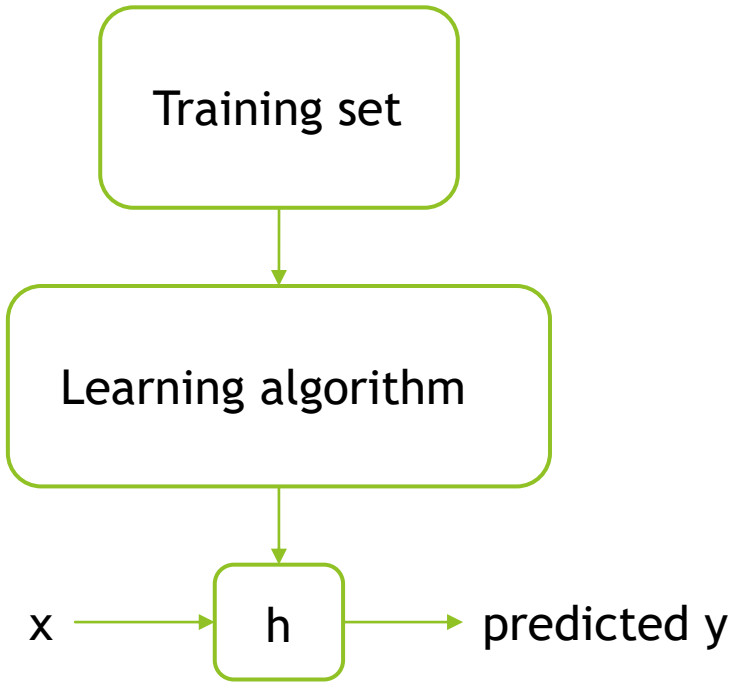
AI in Healthcare

Linear regression

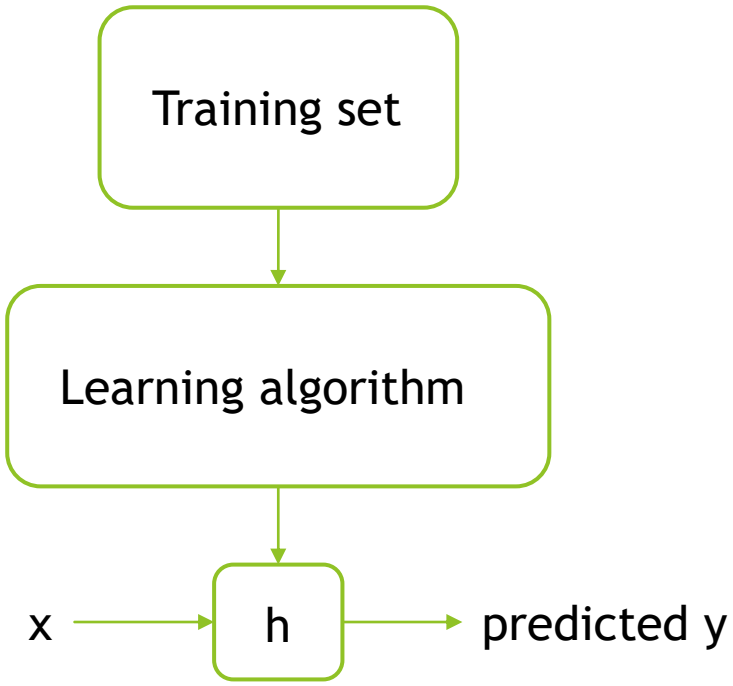
Model Representation

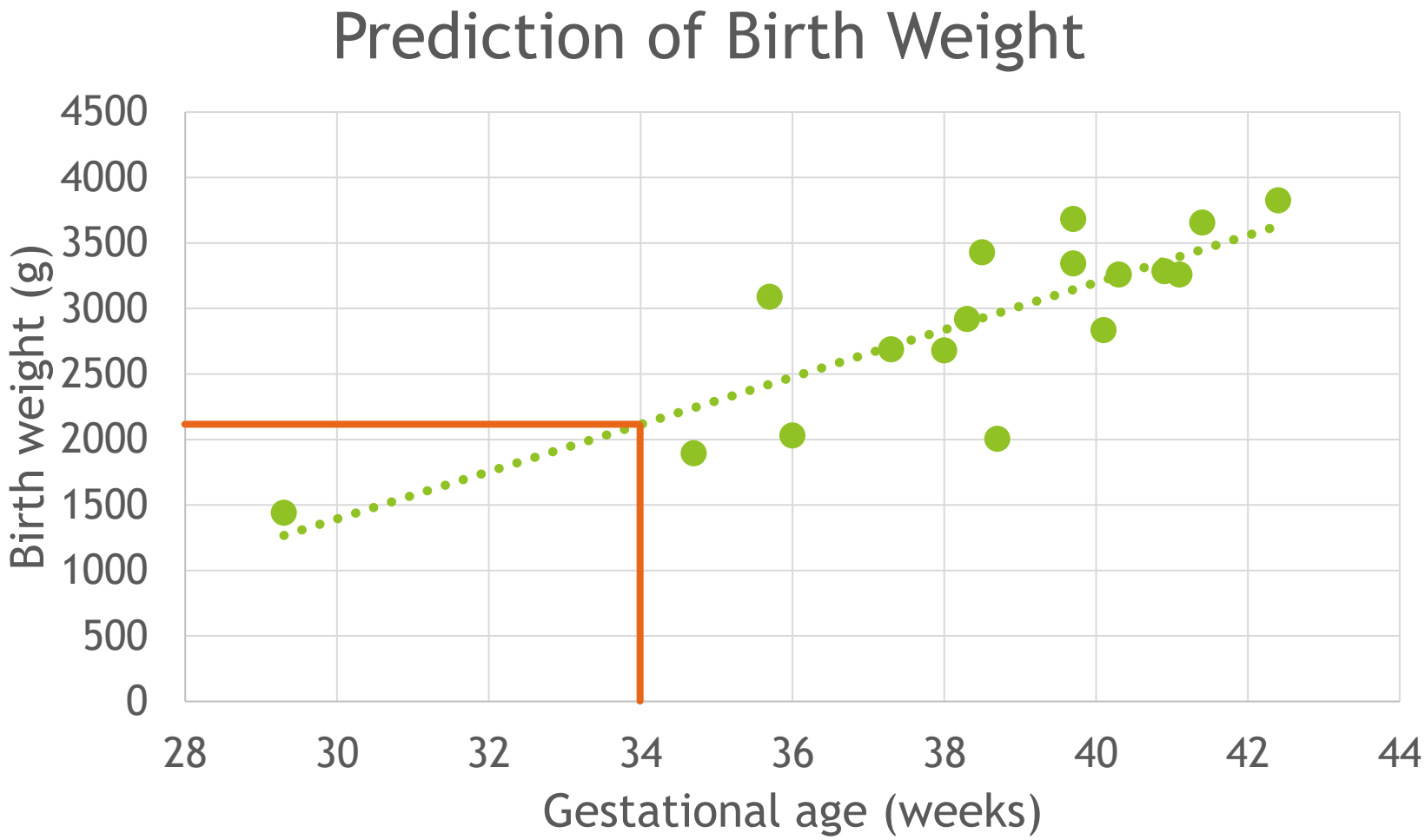


h – hypothesis.

- In linear regression (with one feature x):
  $h_\theta(x) = \theta_0 + \theta_1 x$
- θ_0 and θ_1 - parameters

Exercise


Training set
Learning algorithm
x
h
predicted y


Prediction of Birth Weight
Birth weight (g)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
28
30
32
34
36
38
40
42
44
Gestational age (weeks)

Prediction of Birth Weight

- Training set of birth weights (table)
- m – the number of training examples.
  In the current example m=17
- x_i – i-th input feature.
  For example
  x_1= {34.7, 36, 29.3, 40.1, …, 48.7}
- y - output feature.
  In the current example
  y = {1895, 2030, 1440, 2835, …, 2005}
- $\left(x^{(i)}, y^{(i)}\right)$- i-th training example.
  For example $x^{(2)}$= {36, 341}
  $\left(x^{(4)}, y^{(4)}\right)$ = ({40.1, 371}, 2835)

$x^{(1)}$

	x_1		y
Infant ID	Gestational Age (weeks)	Head circum-ference (mm)	Birth Weight (grams)
1	34.7	336	1895
2	36	341	2030
3	29.3	298	1440
4	40.1	371	2835
…	…	…	…
17	38.7	357	2005

m

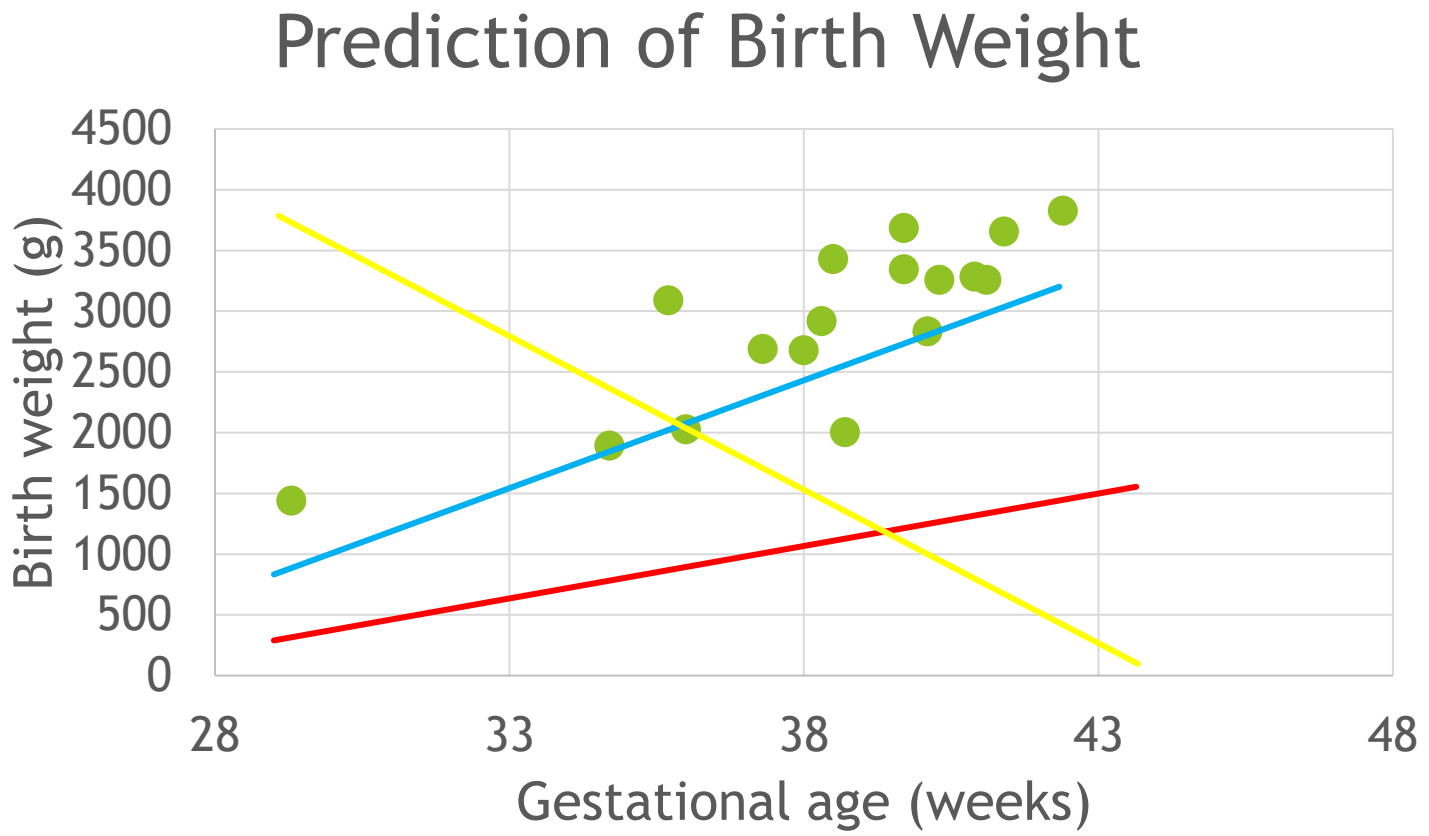
Exercise

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

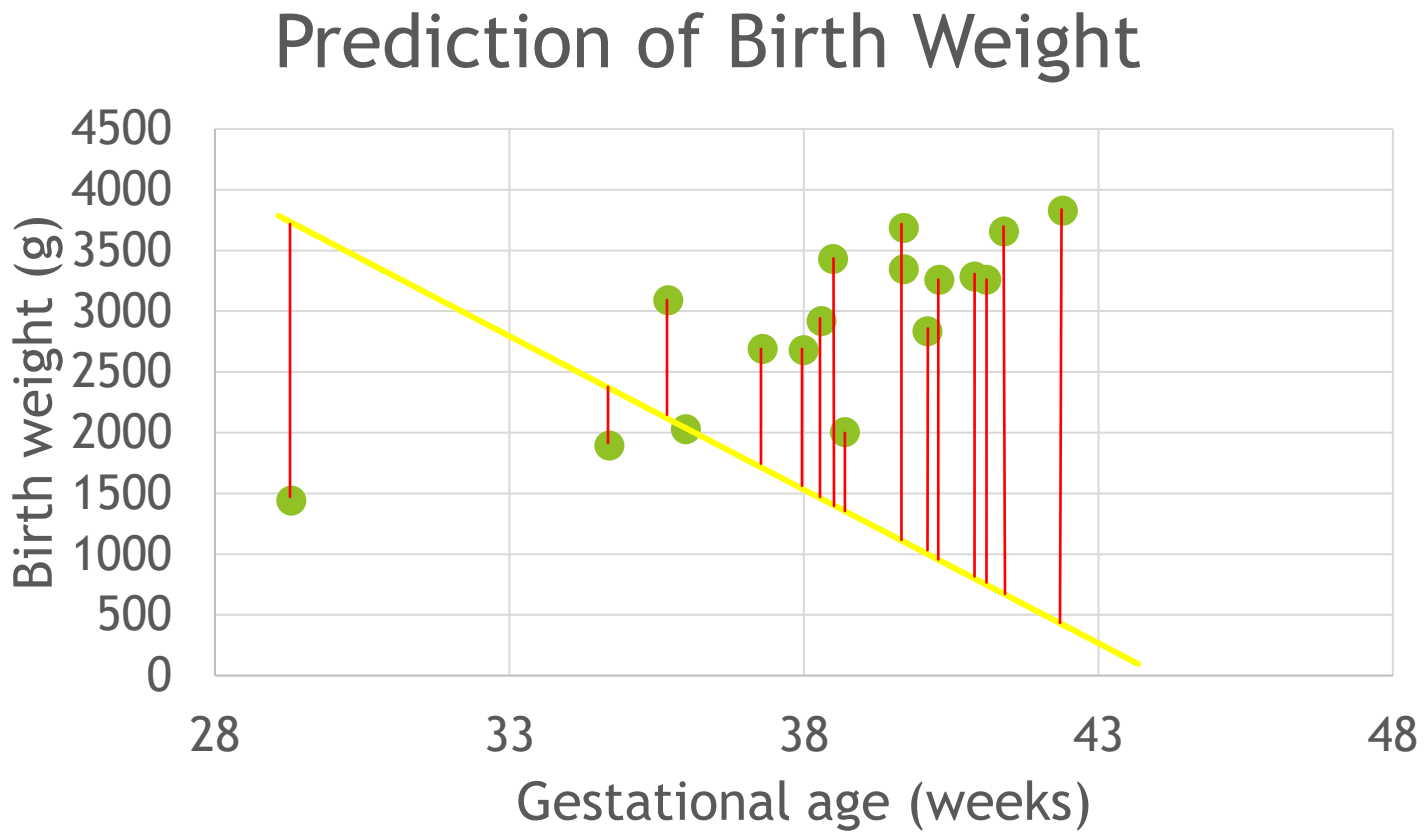
What does the hypothesis function look like, when:

- $\theta_0 = 1;\ \theta_1 = 0$
- $\theta_0 = 2;\ \theta_1 = 1$
- $\theta_0 = 0;\ \theta_1 = 0.5$

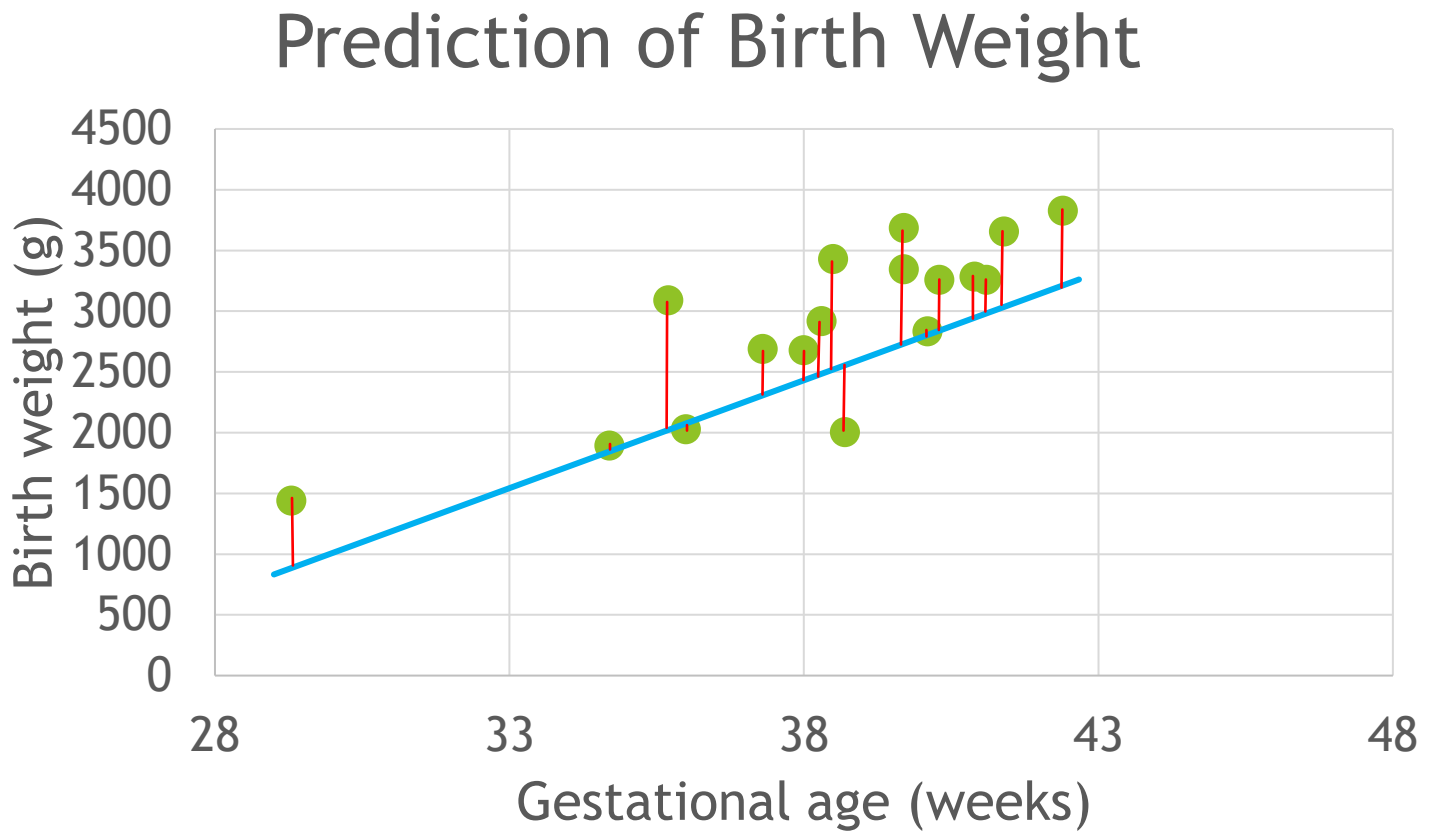
Exercise


Prediction of Birth Weight
Birth weight (g)
4500
4000
3500
3000
2500
2000
1500
1000
500
0
28
33
38
43
48
Gestational age (weeks)

Cost Function

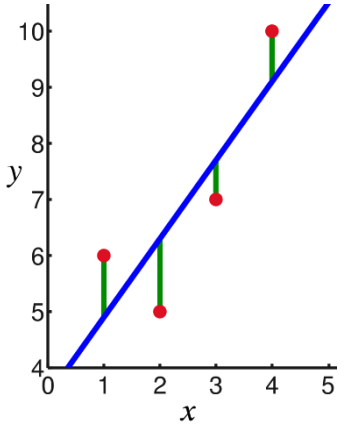


Cost Function



$$minimize\ h_\theta(x) - y$$

Cost Function



- Cost function (square error cost function):

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

- The aim is to minimize the cost function:

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

Exercise

x	y
1	1
2	2
3	3

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

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

- Plot $J(\theta_1)$ when $\theta_0 = 0$
- Plot $h_\theta(x)$ for each of the θ_1 used
- What is the minimum $J(\theta_1)$?

Cost function

In case both of the values θ_0 and θ_1 are used, the cost function $J(\theta_0, \theta_1)$ is a 3D plot

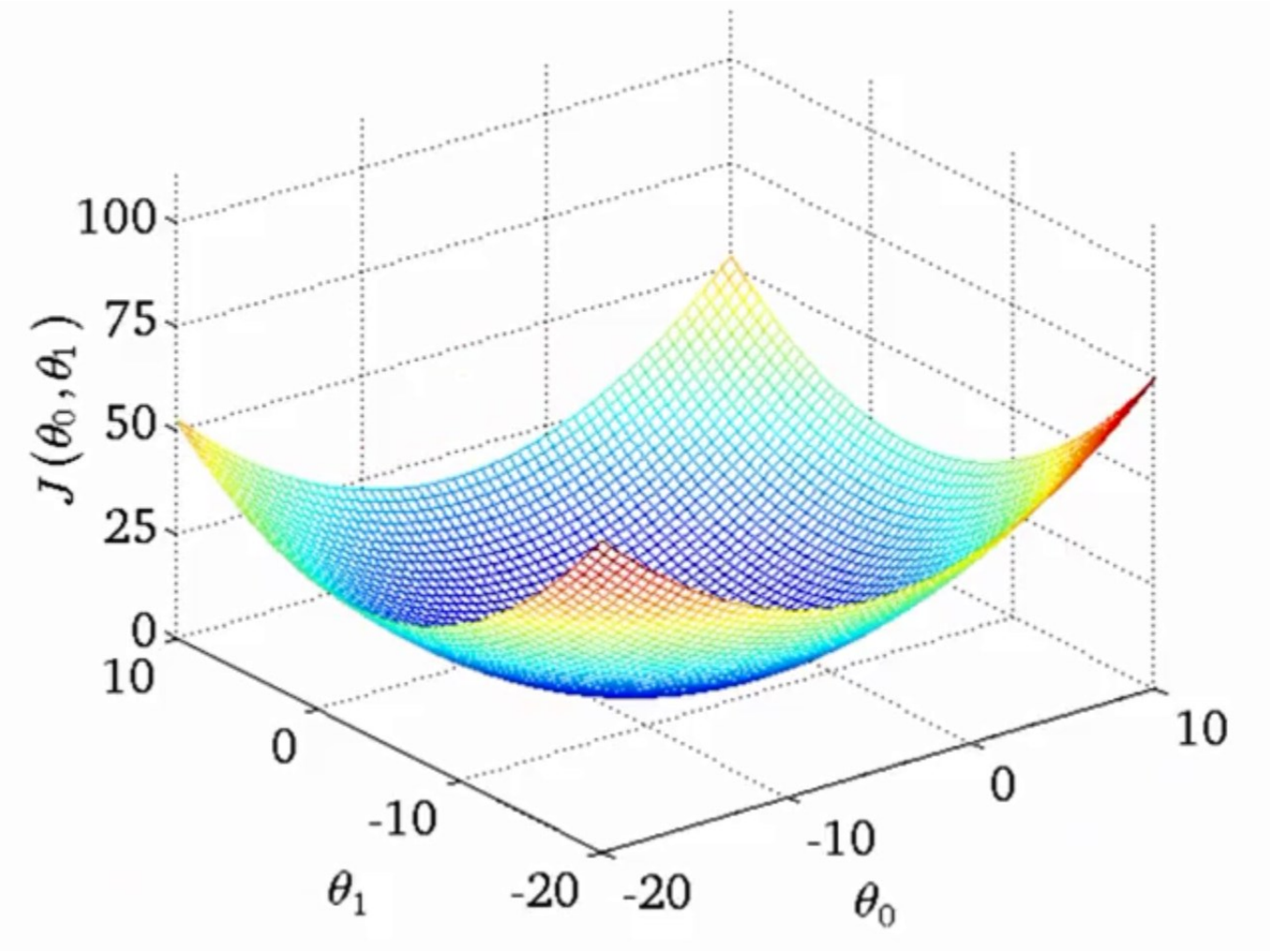
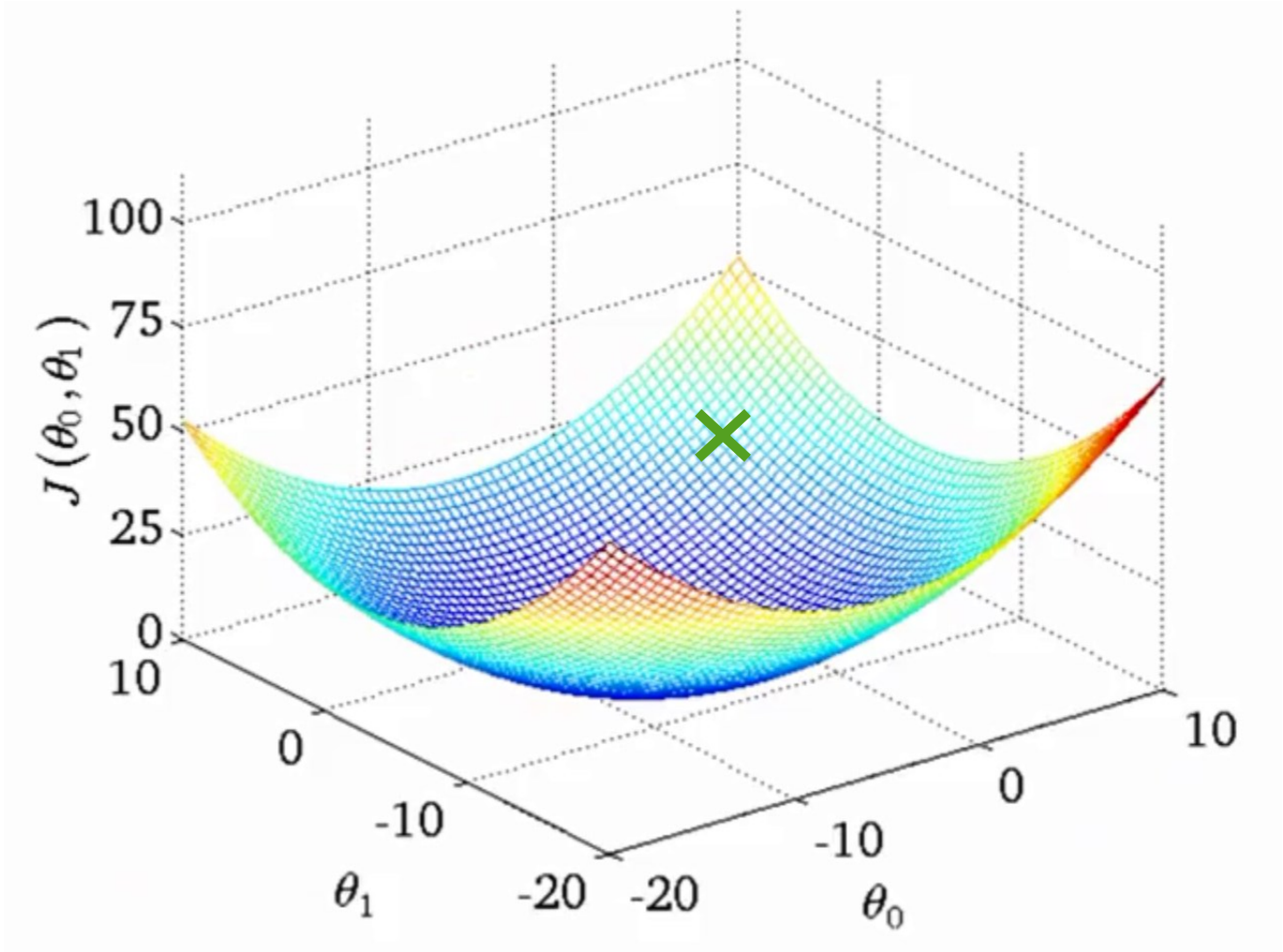


Image from Andrew Ng Coursera course „Machine Learning“

Gradient Descent

- Gradient descent is used to minimize the cost function
- To do that:
  1. Start with some θ_0 and θ_1
  2. Change θ_0 and θ_1 to reduce $J(\theta_0, \theta_1)$ until we end up at the minimum $J(\theta_0, \theta_1)$

Gradient Descent



Gradient

- Gradient gives the direction of the greatest increase of the function f:

$$grad(f) = \frac{\partial f}{\partial x}\vec{i} + \frac{\partial f}{\partial y}\vec{j} + \frac{\partial f}{\partial z}\vec{k}$$

- i,j and k are the unit vectors in the direction of x, y and z coordinates, respectively
- For example, when $f(x, y, z) = 5x^3 + 2y - z$, then

$$grad(f) = 15x^2\vec{i} + 2\vec{j} - 1\vec{k}$$

Gradient Descent

Repeat until convergence:

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

for $j = 0$ and $j = 1$

- $:=$ means assignment, $a := a - 1$ means subtract one from a
- α – learning rate – how big step we take when updating θ_j
- θ_0 and θ_1 are updated simultaneously (without changing $J(\theta_0, \theta_1)$ in between):

$$\text{var1} = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

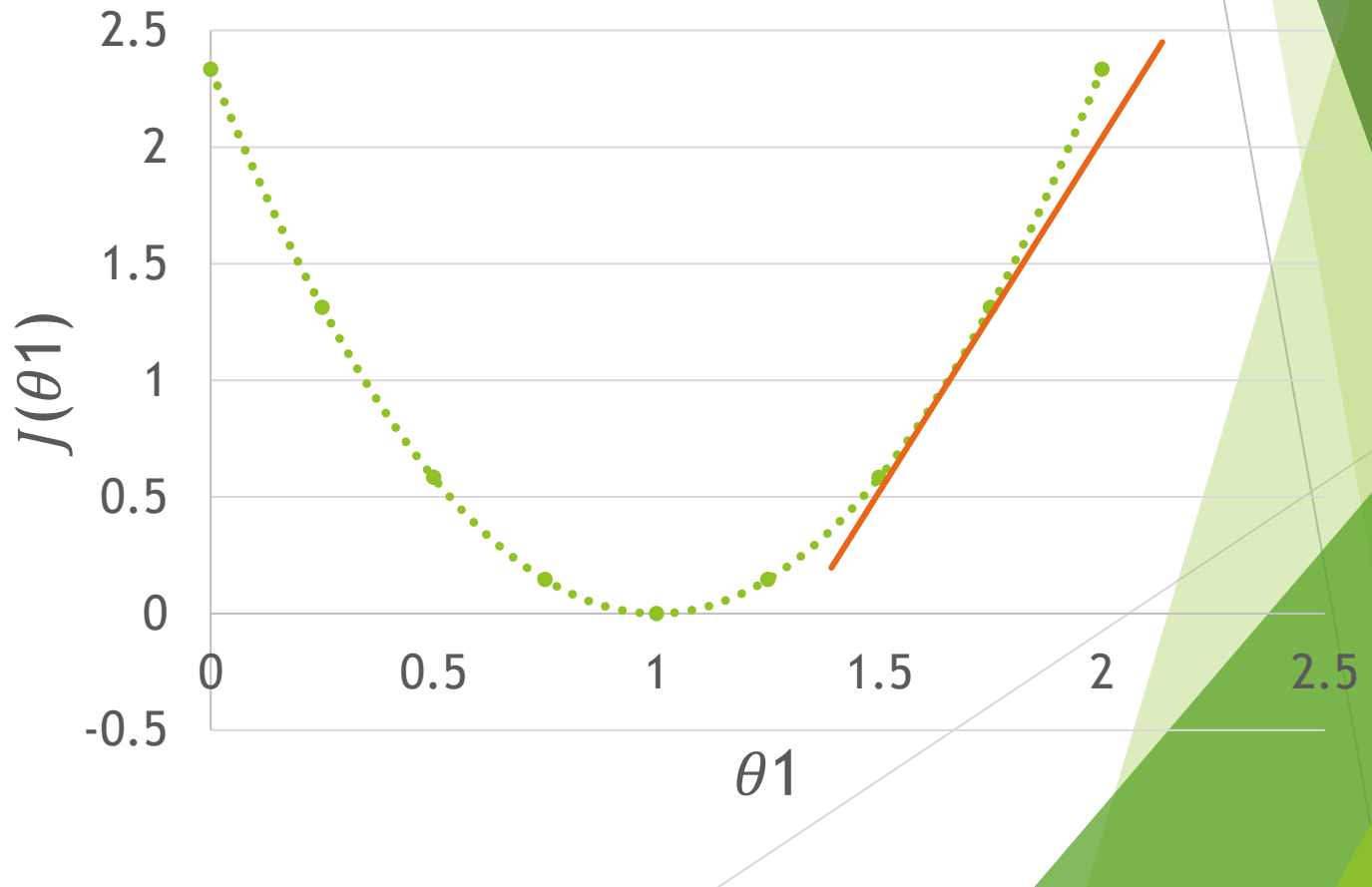
$$\text{var2} = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 = var1$$

$$\theta_1 = var2$$

Gradient Descent

- $\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$
- $\frac{\partial}{\partial \theta_1} J(\theta_1)$ is the slope of the line



Exercise

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

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

Find

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

Gradient Descent

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

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

Gradient Descent: learning rate α

- If α is too small, gradient descent will be slow
- If α is too large, gradient descent may fail to converge, or may even diverge

Gradient Descent for Linear Regression

- Linear regression model with one feature:

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

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

- Gradient descent algorithm:

Repeat until convergence:

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

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

Multivariate Linear Regression

- Multiple features (variables)

- n – number of features (n=4)
- $x^{(i)}$ - input (features) of i-th training example
- $x_j^{(i)}$ - value of feature j in i-th training example ($x_4^{(3)} = 30$)

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

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

Multivariate Linear Regression: Hypothesis

- Hypothesis:

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

- For matrix calculations $x_0 = 1$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}; \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$$

- Therefore

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

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

Exercise

Make sure that

$$\theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

if

$$x = \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}; \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \dots \\ \theta_n \end{bmatrix}$$

Multivariate Linear Regression

- Hypothesis:

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

- Cost function:

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

- Gradient descent algorithm:

Repeat until convergence:

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

$$j = 0, \ldots, n$$

Feature Scaling

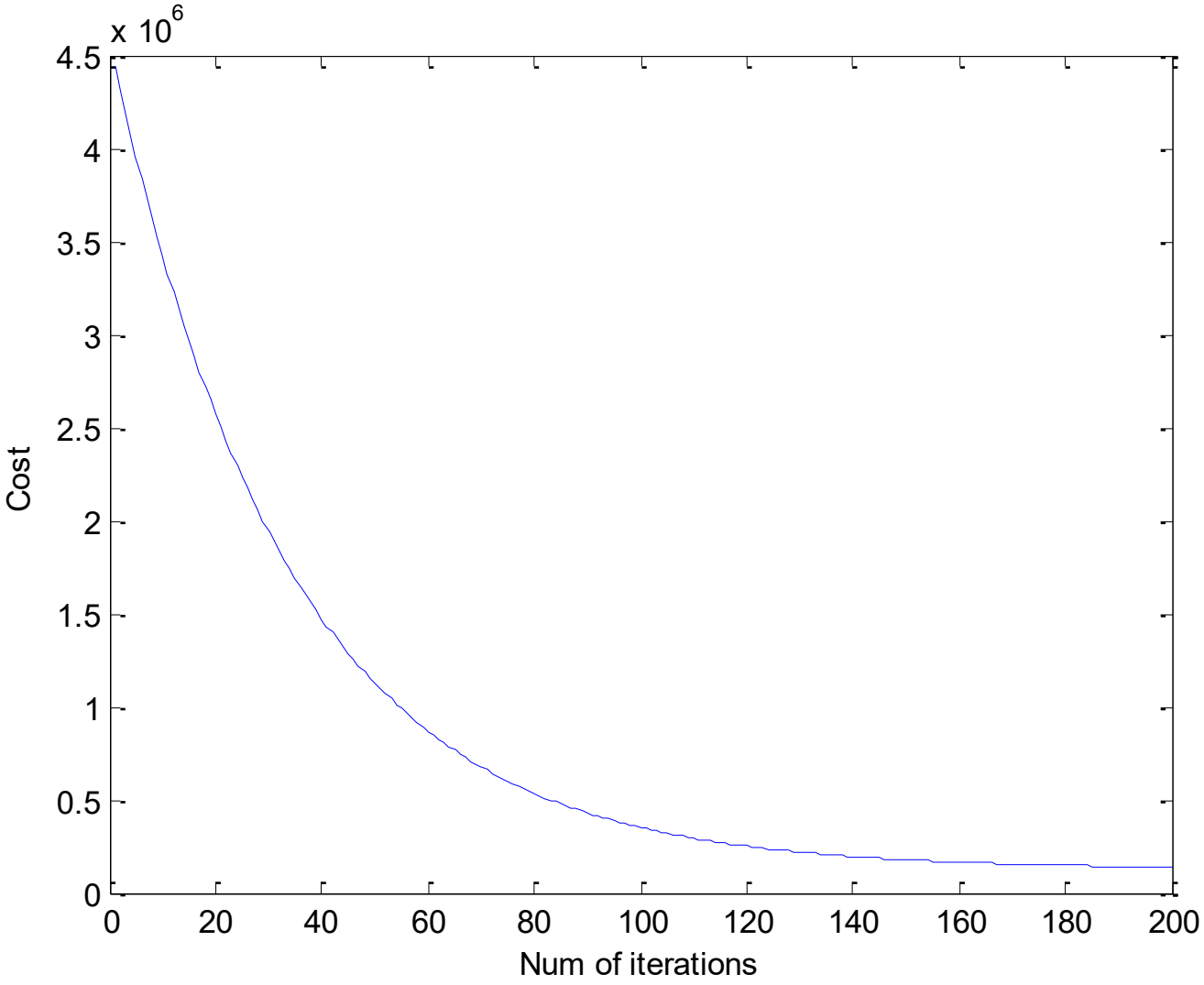
- If features have different scales, it is important to normalize these features
- **Mean normalization** – features have approximately zero mean

$$x_1 \leftarrow \frac{x_1 - \mu_1}{s_1}$$

  - μ_1 – mean value
  - s_1 – max minus min or standard deviation

Checking gradient descent

- Aim: find $\min J(\theta)$
- Plot the cost funtion $J(\theta)$ for each iteration
- $J(\theta)$ should decrease after every iteration (if it does not, decrease learning rate α)
- The number of iterations before the gradient descent converges may vary significantly (depends on α)

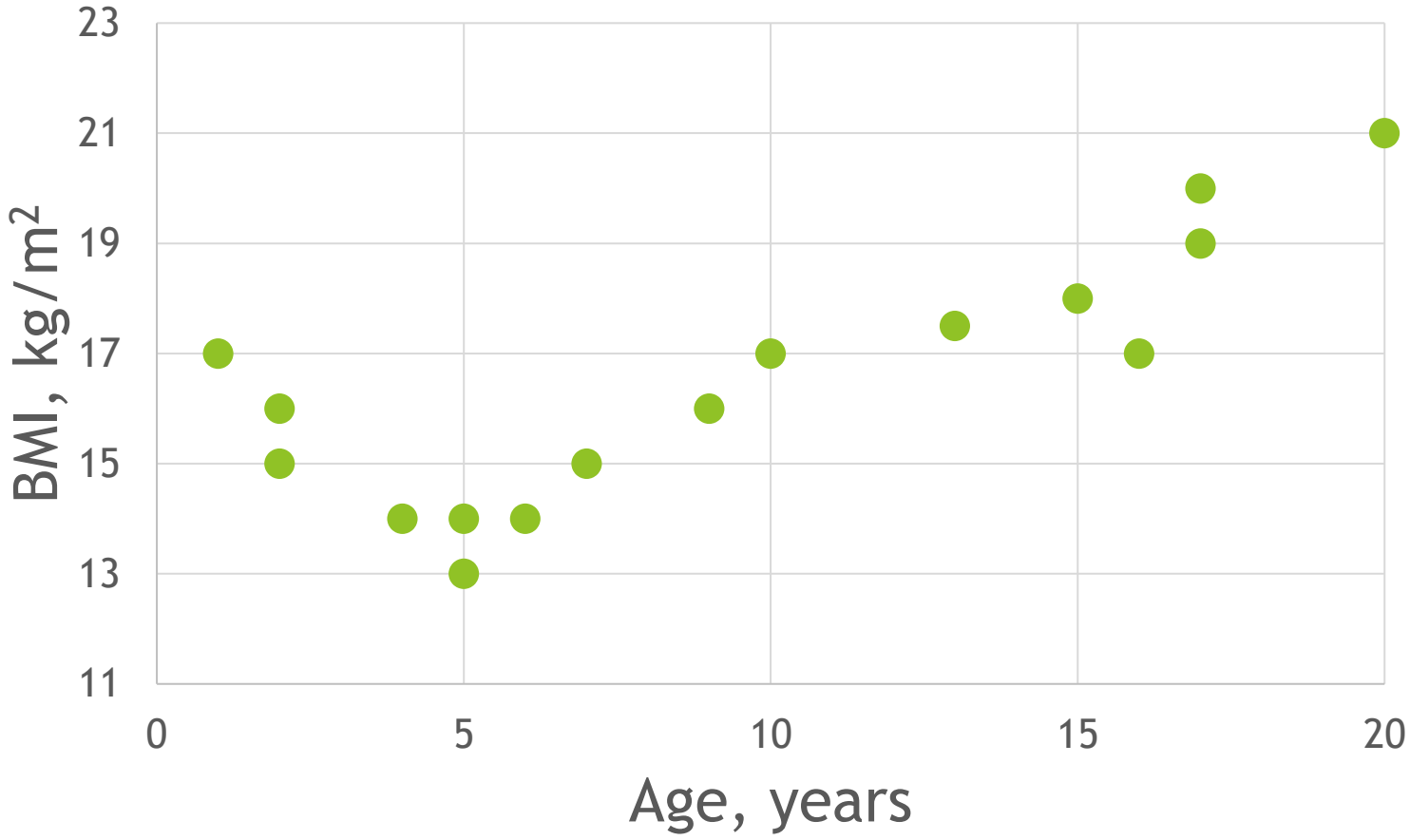


Learning rate

- Learning rate α should be small enough for $J(\theta)$ to decrease on every iteration
- If α is too small, gradient descent can be too slow to converge
- Recommended values:

$$\{\dots, 1, 0.3, 0.1, 0.03, 0.01, 0.003, 0.001, \dots\}$$

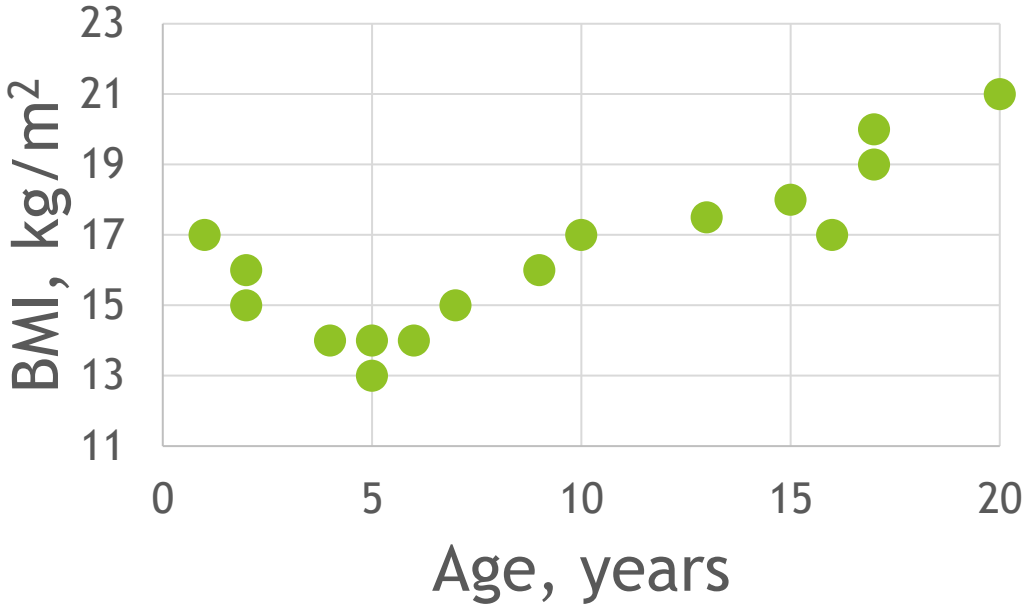
Polynomial regression



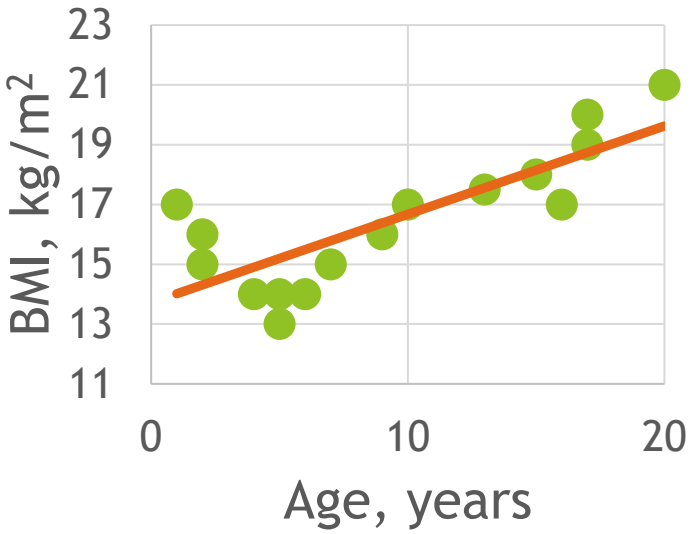
Polynomial regression

- Modify current features. For example use:
  - $h_\theta(x) = \theta_0 + \theta_1(age) + \theta_2(age)^2 + \theta_3(age)^3$

    $(x_1 = (age),\ x_2 = (age)^2,\ x_3 = (age)^3)$
- Feature scaling may be important!

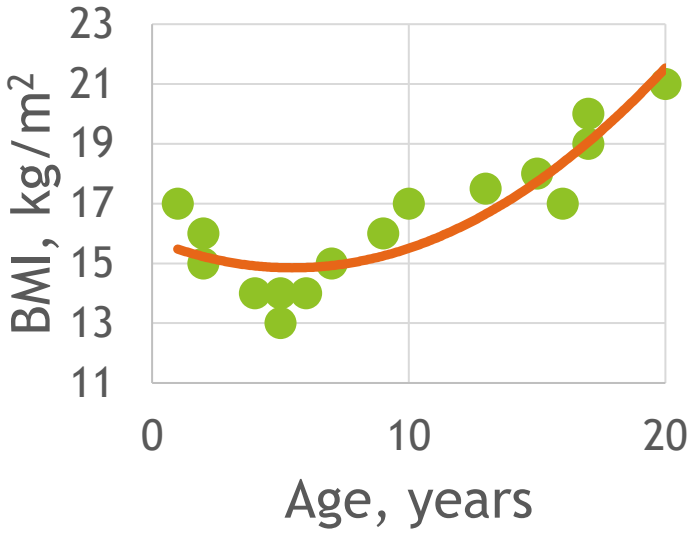


The Problem of Overfitting



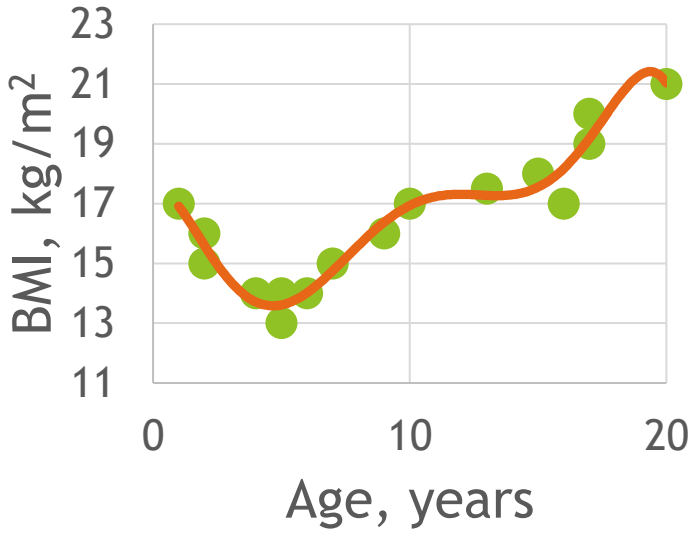
$\theta_0 + \theta_1 x$

Underfitting



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

Just right



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

Overfitting

Addressing overfitting

1. Reduce the number of features
   - Manually select features
   - Automatically select features
2. Regularization

Regularization

- Adding regularization term decreases parameters θ_j:

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

- λ – regularization parameter
- Small values for parameters θ_j
  - Simpler hypothesis
  - Less prone to overfitting

Regularized Linear Regression

Linear regression gradient descent with regularization:

Repeat until convergence:

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

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

$$j = 1, \ldots, n$$

Normal equation

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

- For smaller number of features (<1000)
- For example when

x_0	x_1	x_2	x_3	y
1	5	1020	25	457
1	3	1637	45	376
1	8	2011	27	467

Then

$$X = \begin{bmatrix} 1 & 5 & 1020 & 25 \\ 1 & 3 & 1637 & 45 \\ 1 & 8 & 2011 & 27 \end{bmatrix}, y = \begin{bmatrix} 457 \\ 376 \\ 467 \end{bmatrix}$$