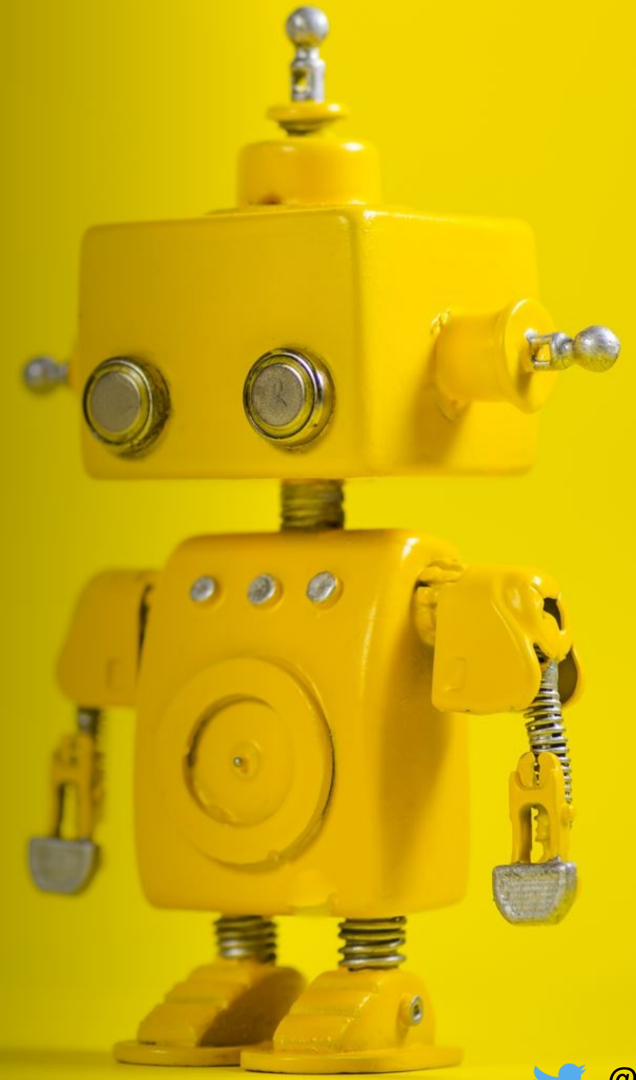
FruitPunch AI Bootcamp

Session 1– Linear Regression

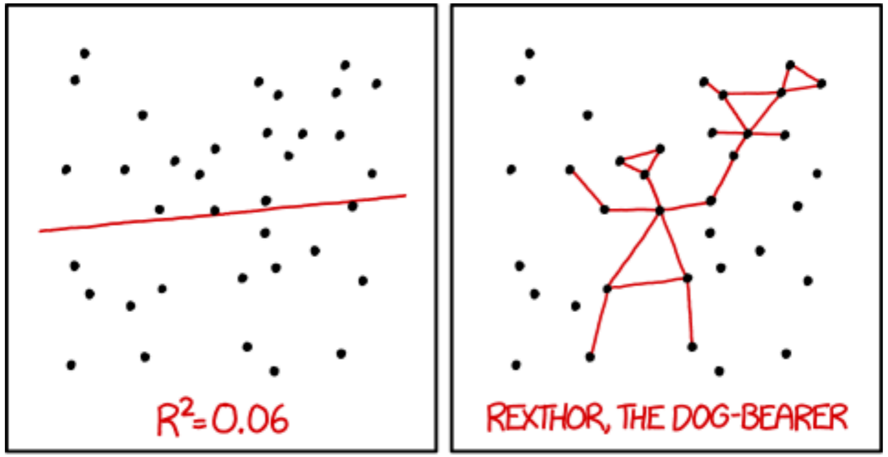
15.11.2020

M. Alican Noyan

FruitPunch AI & Ipsumio



@malicannoyan



What is Machine Learning?

Challenge 1
Linear regression

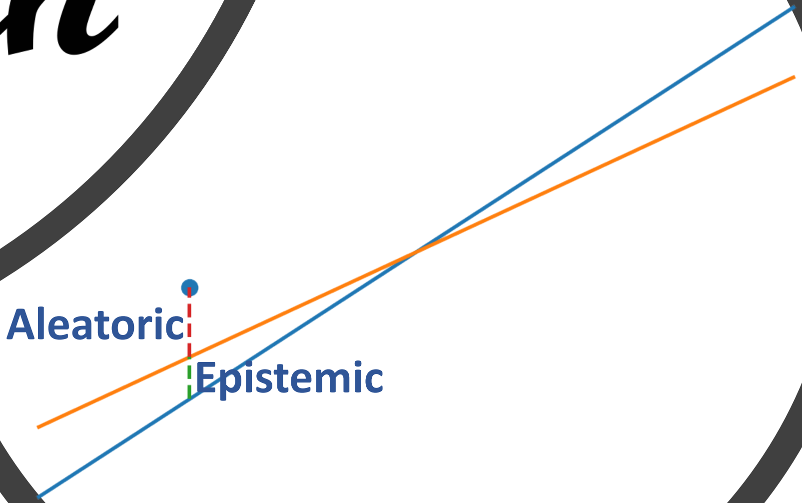
45 min

3 hours

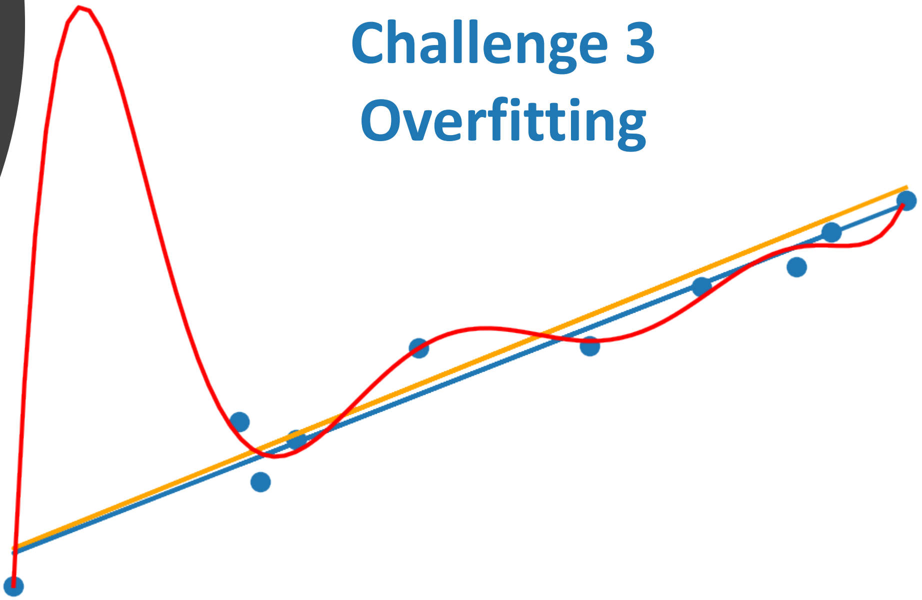
Challenge 4
Implement Linear regression

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

Challenge 2



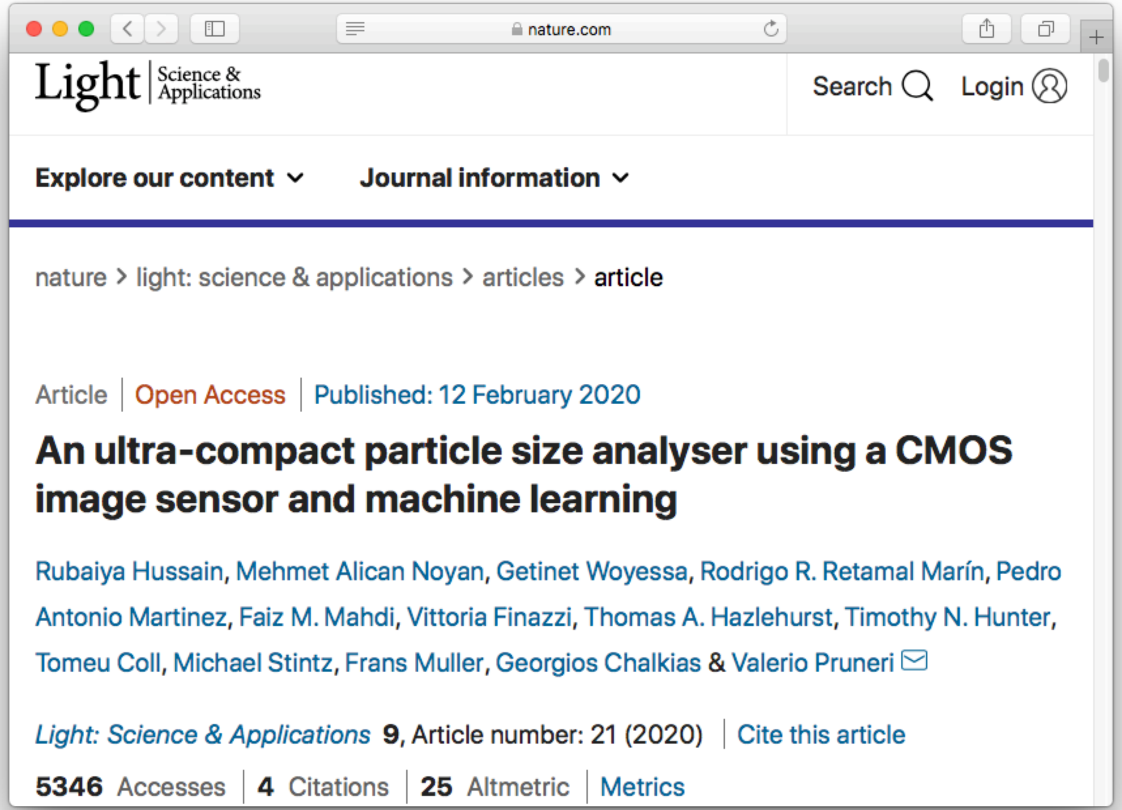
Challenge 3
Overfitting



Any ML experience?	50/50
Any ML course?	None/All of them
Stats?	Low/High
Software?	Low/High

Mehmet Alican Noyan
B.Sc. Metallurgy and Materials Engineering (ODTÜ - Ankara)
M.Sc. Materials Science and Nanotechnology (Bilkent - Ankara)
Ph.D. Photonics (ICFO - Barcelona)

Ipsumio
www.ipsumio.com



nature.com

Light | Science & Applications

Search Login

Explore our content Journal information

nature > light: science & applications > articles > article

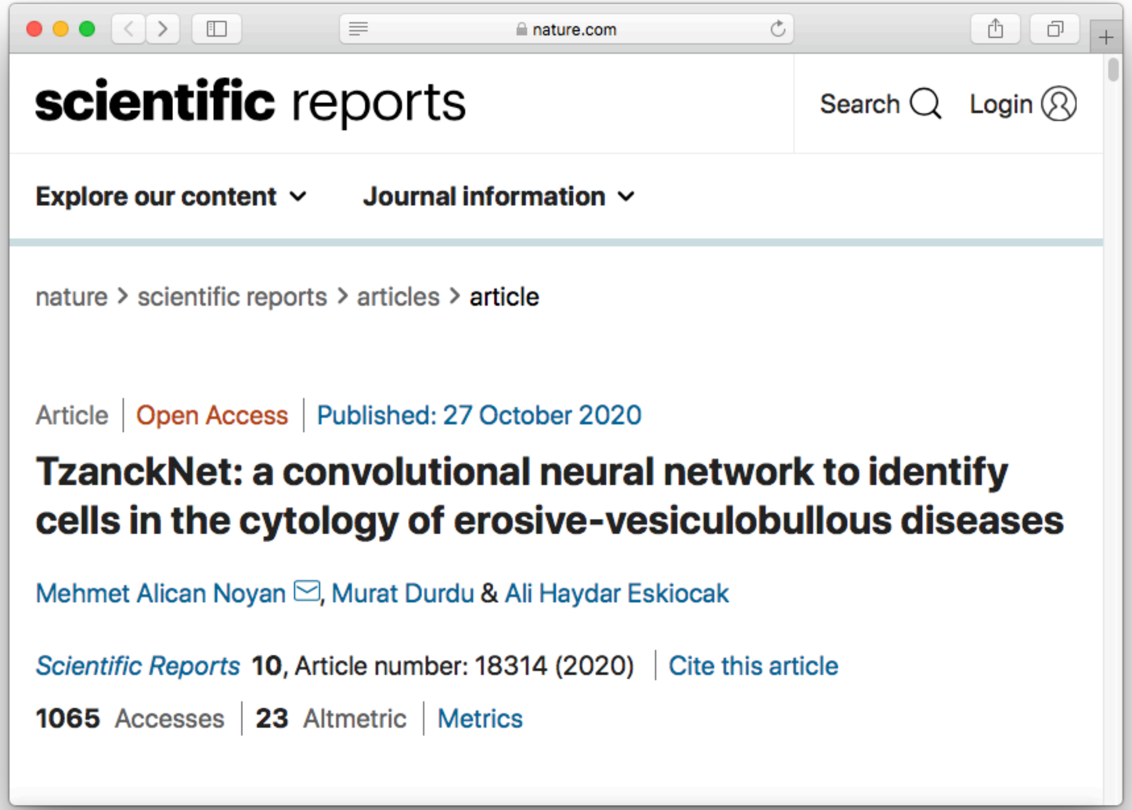
Article | Open Access | Published: 12 February 2020

An ultra-compact particle size analyser using a CMOS image sensor and machine learning

Rubaiya Hussain, Mehmet Alican Noyan, Getinet Woyessa, Rodrigo R. Retamal Marín, Pedro Antonio Martinez, Faiz M. Mahdi, Vittoria Finazzi, Thomas A. Hazlehurst, Timothy N. Hunter, Tomeu Coll, Michael Stintz, Frans Muller, Georgios Chalkias & Valerio Pruneri

Light: Science & Applications **9**, Article number: 21 (2020) | Cite this article

5346 Accesses | **4** Citations | **25** Altmetric | Metrics



nature.com

scientific reports

Search Login

Explore our content Journal information

nature > scientific reports > articles > article

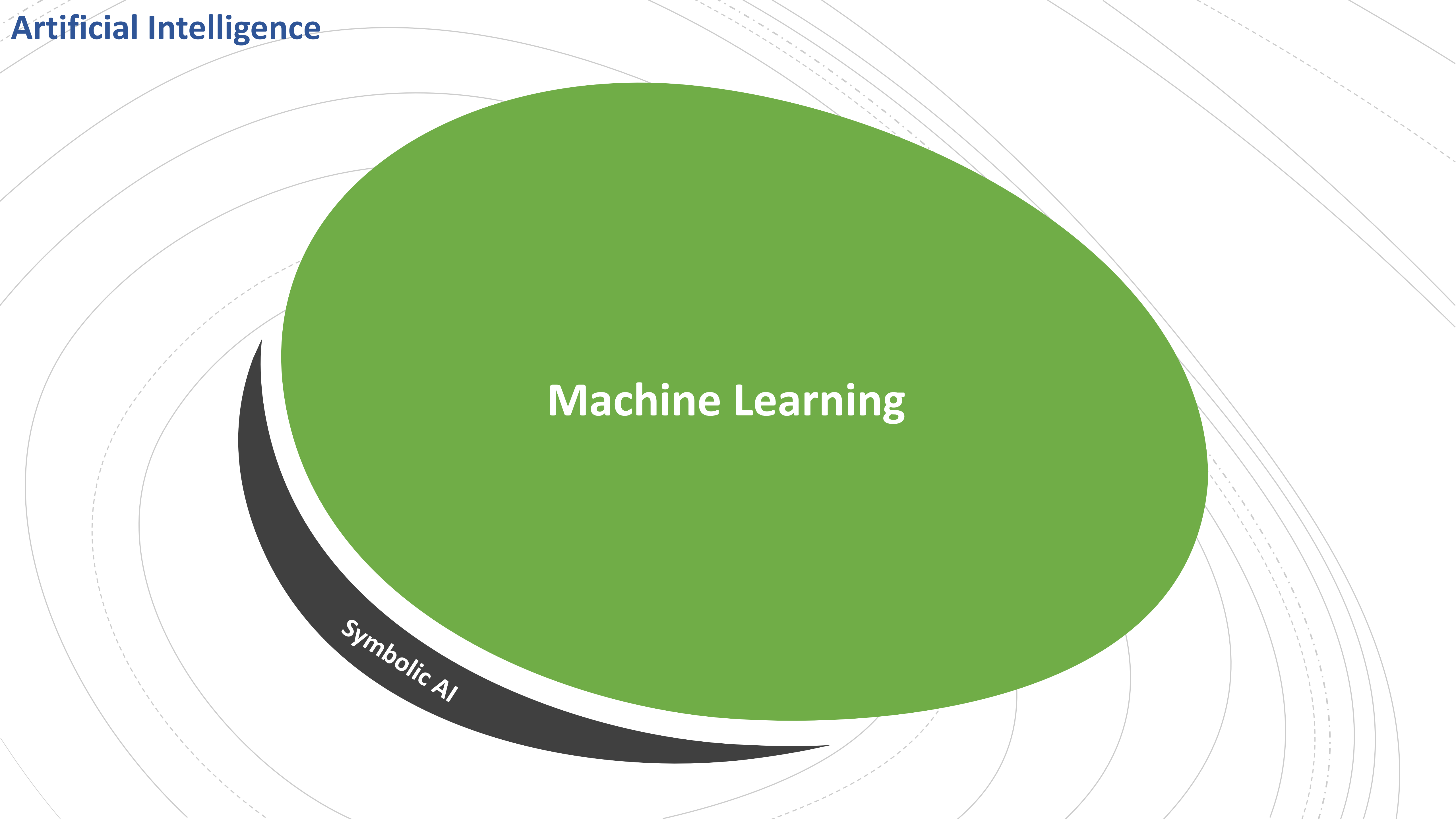
Article | Open Access | Published: 27 October 2020

TzanckNet: a convolutional neural network to identify cells in the cytology of erosive-vesiculobullous diseases

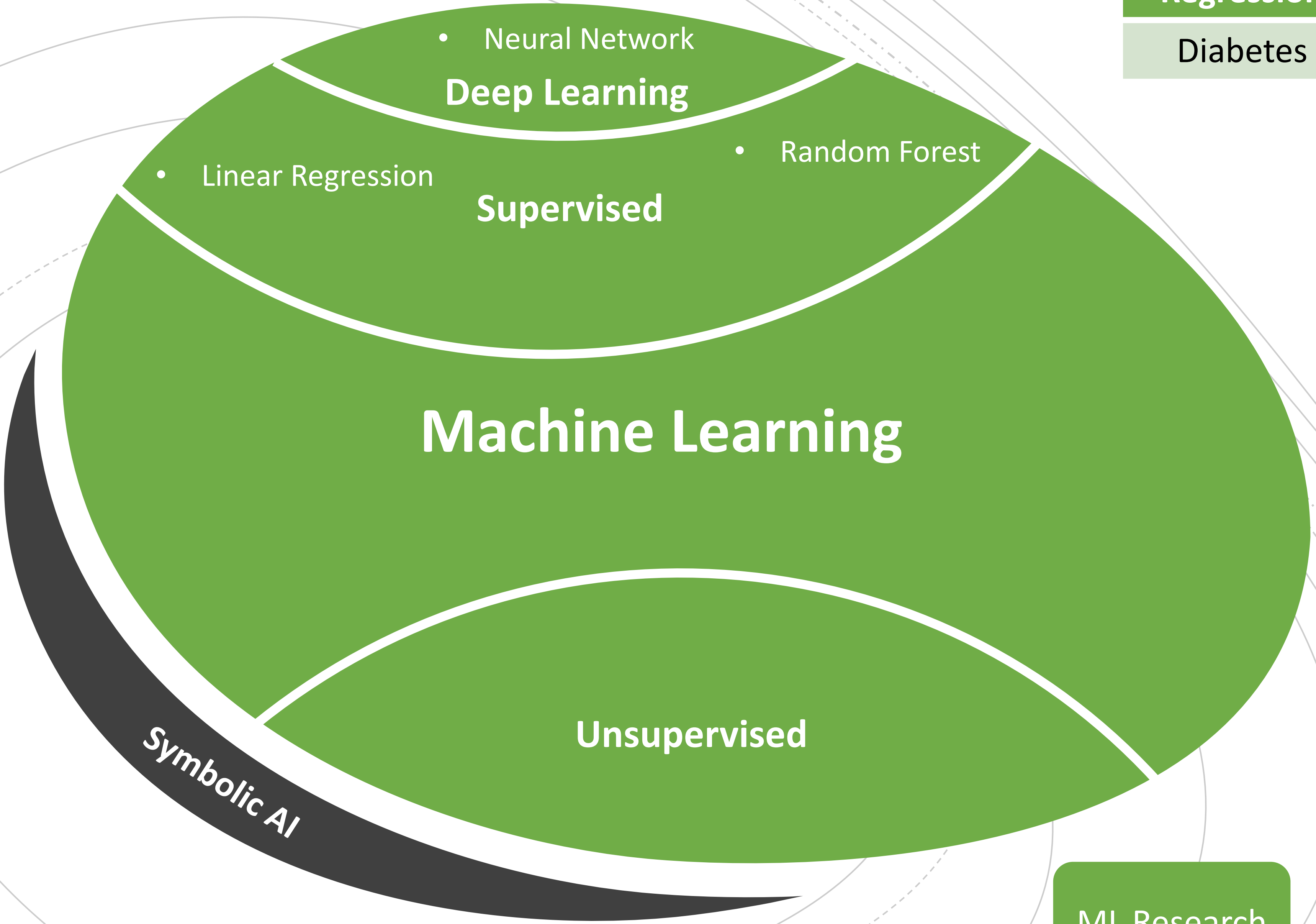
Mehmet Alican Noyan, Murat Durdu & Ali Haydar Eskiocak

Scientific Reports **10**, Article number: 18314 (2020) | Cite this article

1065 Accesses | **23** Altmetric | Metrics


Artificial Intelligence
Machine Learning
Symbolic AI

Artificial Intelligence



Regression	Classification
Diabetes	Digits

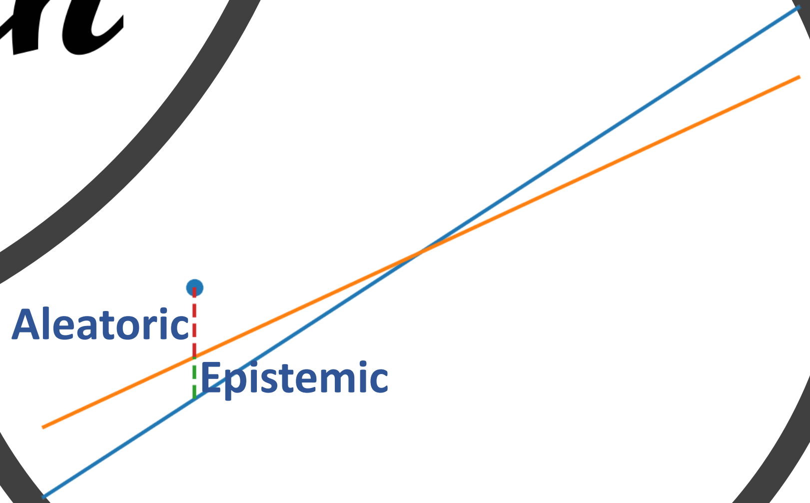
ML Research

Applied ML

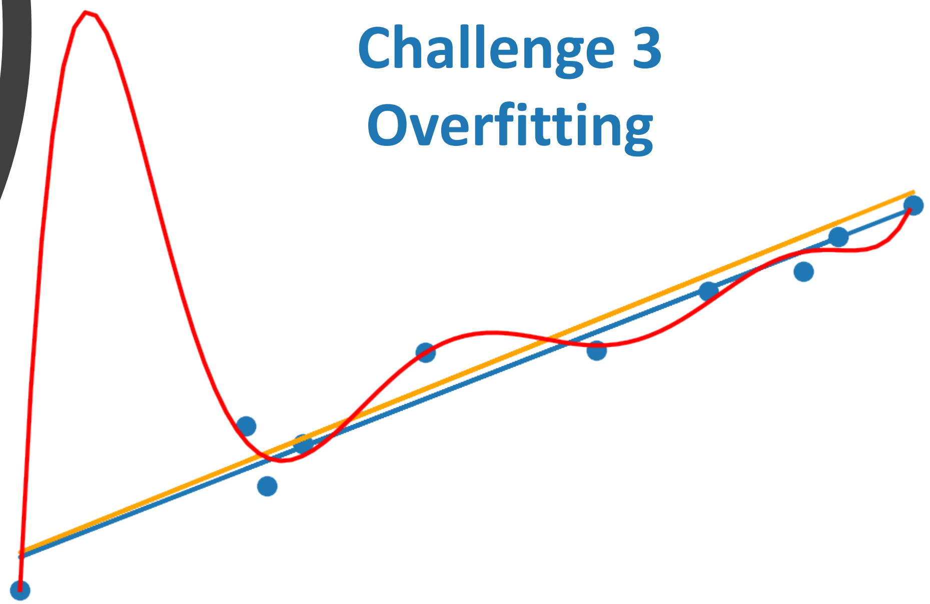
Data = Signal + Noise

Challenge 1
Linear regression

Challenge 2



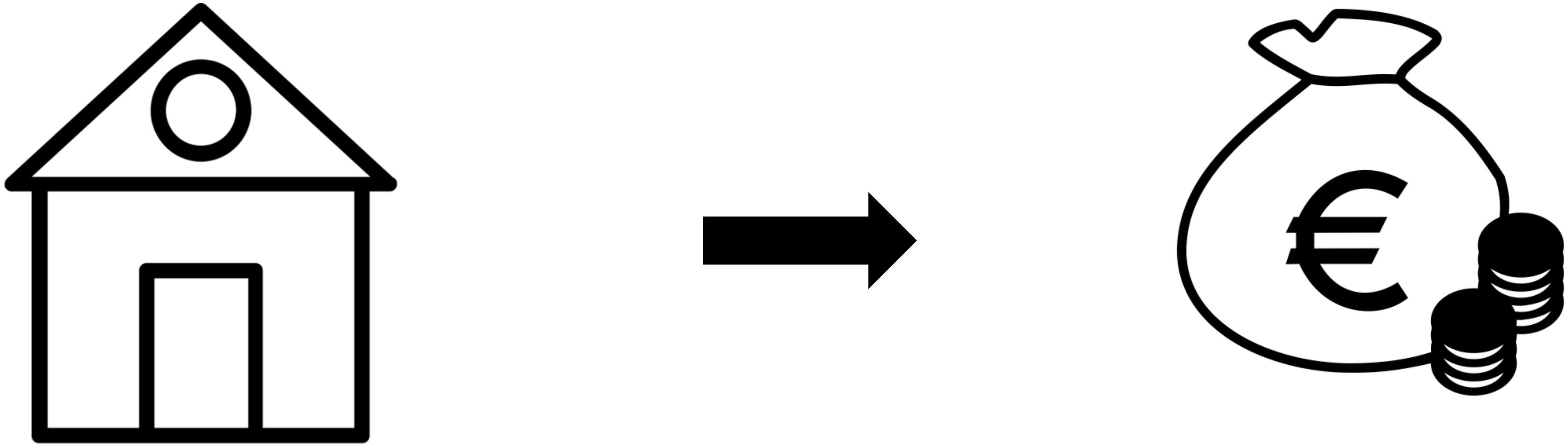
Challenge 3
Overfitting



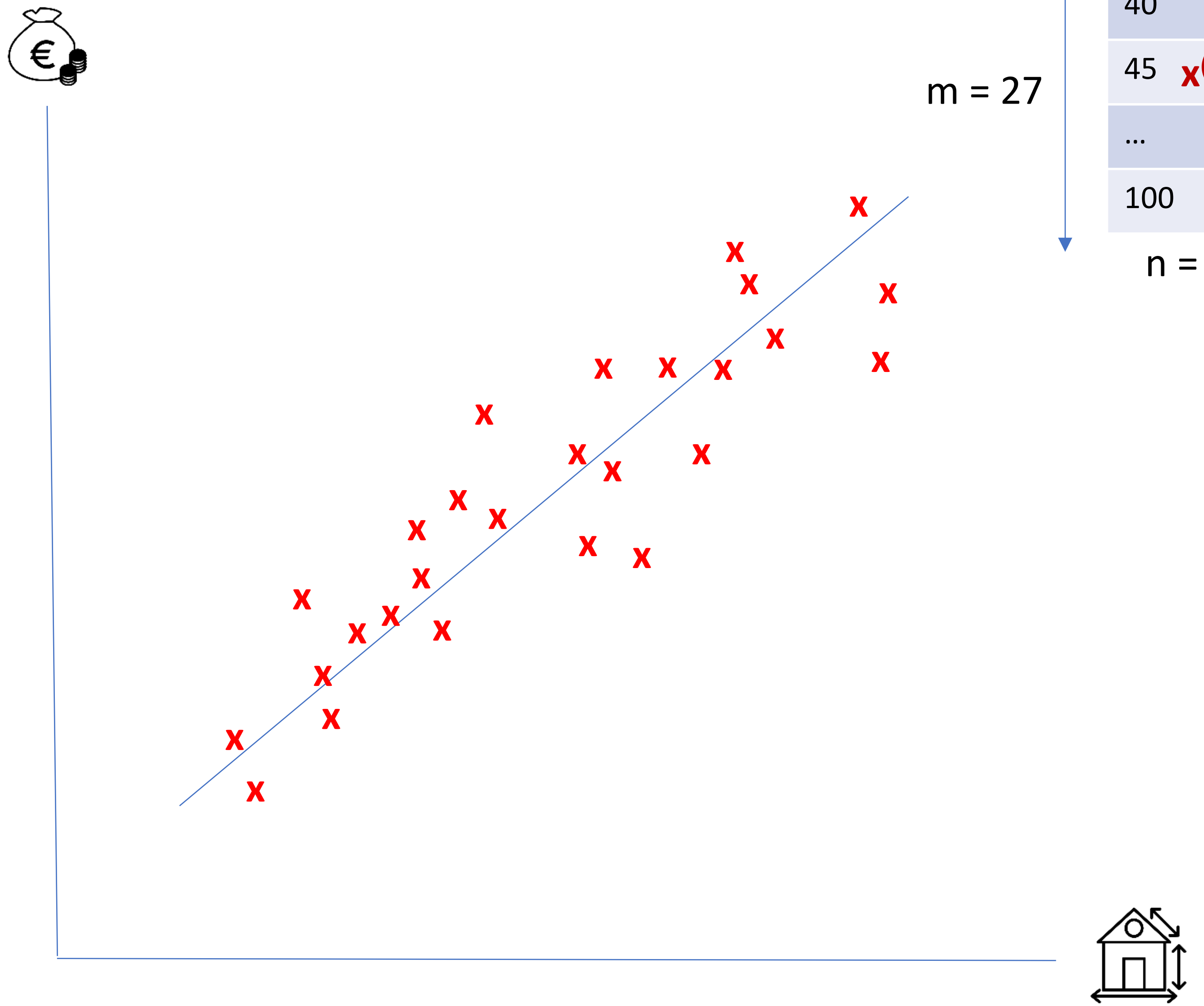
Challenge 4
Implement Linear regression

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

House price prediction



House price prediction



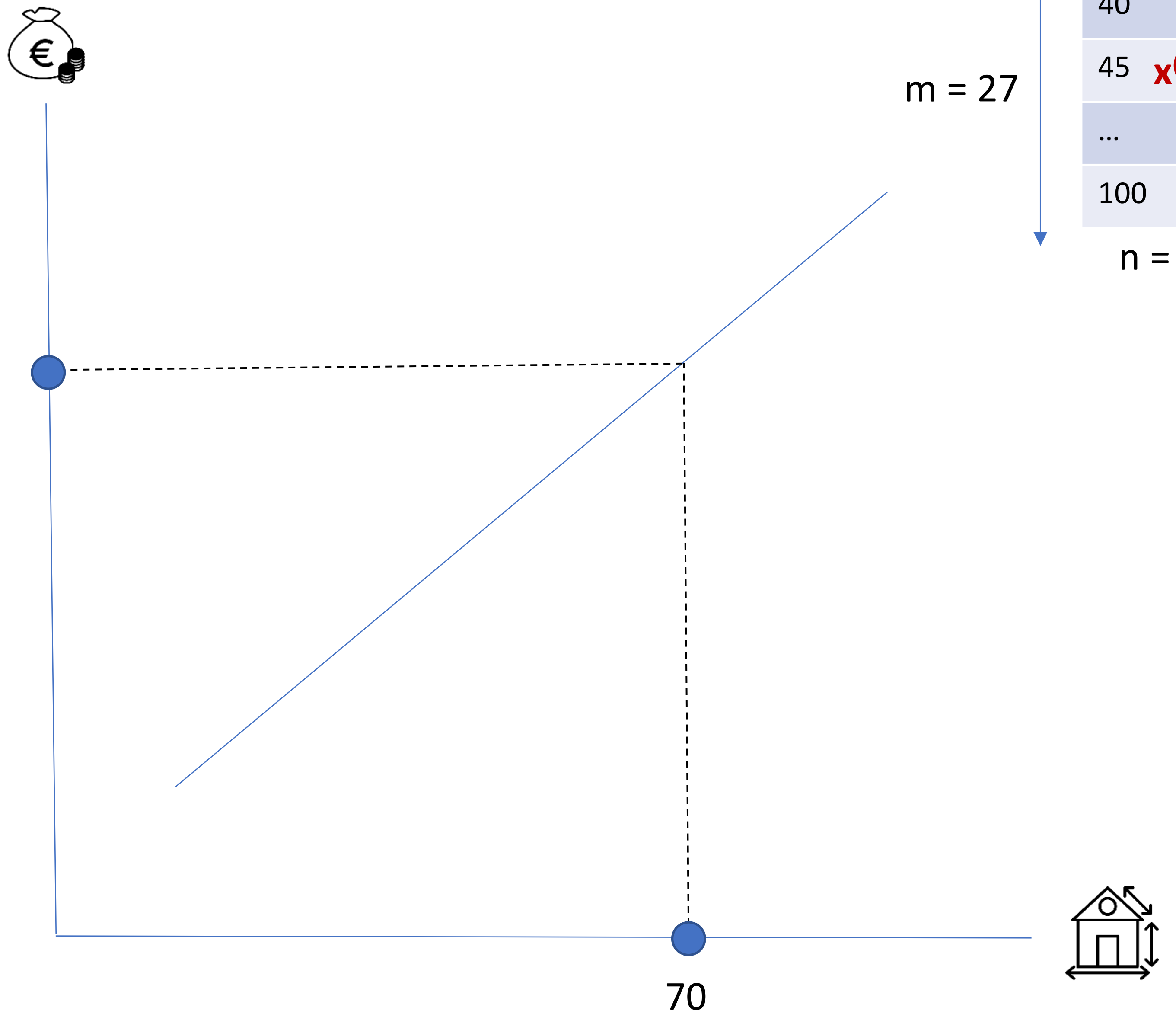
m = 27

Size (m^2)	Price (€)
40	150,000
45 $x^{(2)}$	170,000 $y^{(2)}$
…	
100	310,000

n = 1

$(x^{(i)}, y^{(i)})$

House price prediction



m = 27

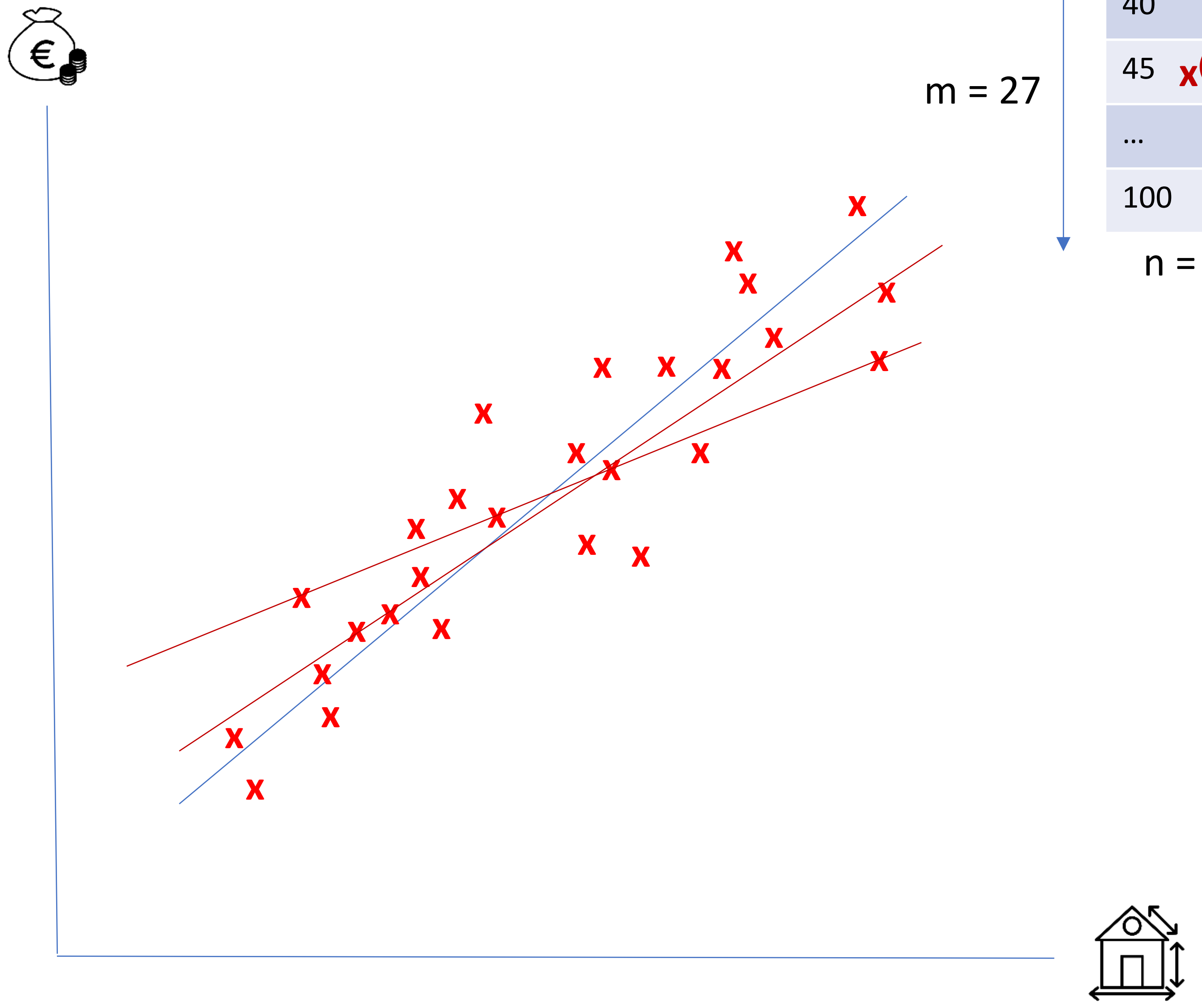
Size (m^2)	Price (€)
40	150,000
45 $\mathbf{x^{(2)}}$	170,000 $\mathbf{y^{(2)}}$
…	
100	310,000

n = 1

$(x^{(i)}, y^{(i)})$

$$y = \theta_0 + \theta_1 \times x$$

House price prediction



m = 27

Size (m²)	Price (€)
40	150,000
45 $x^{(2)}$	170,000 $y^{(2)}$
…	
100	310,000

n = 1

$(x^{(i)}, y^{(i)})$

$y = \theta_0 + \theta_1 \times x$

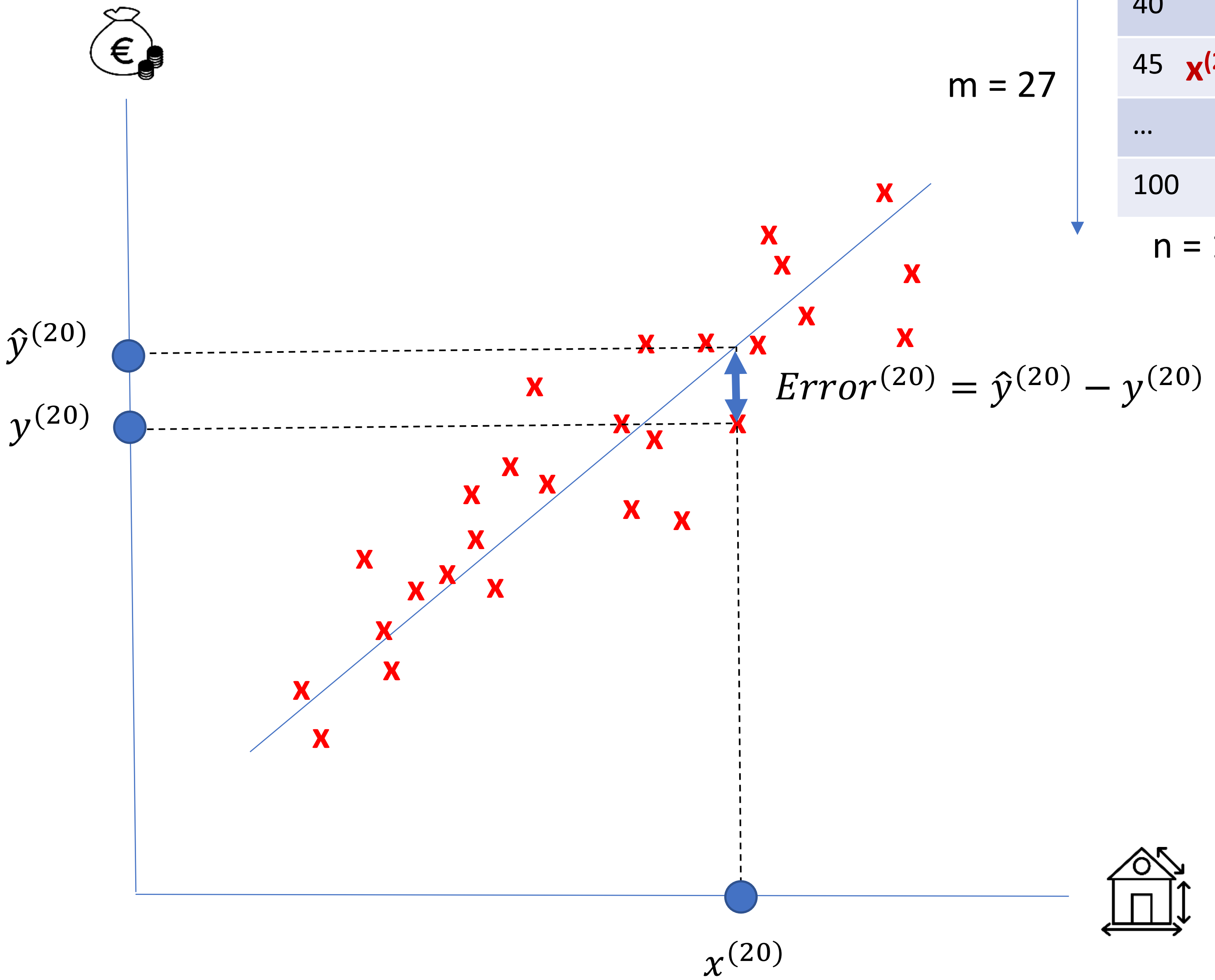
House price prediction

Size (m^2)	Price (€)
40	150,000
45 $x^{(2)}$	170,000 $y^{(2)}$
…	
100	310,000

m = 27

n = 1

$(x^{(i)}, y^{(i)})$



$y = \theta_0 + \theta_1 \times x$

$$\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})$$

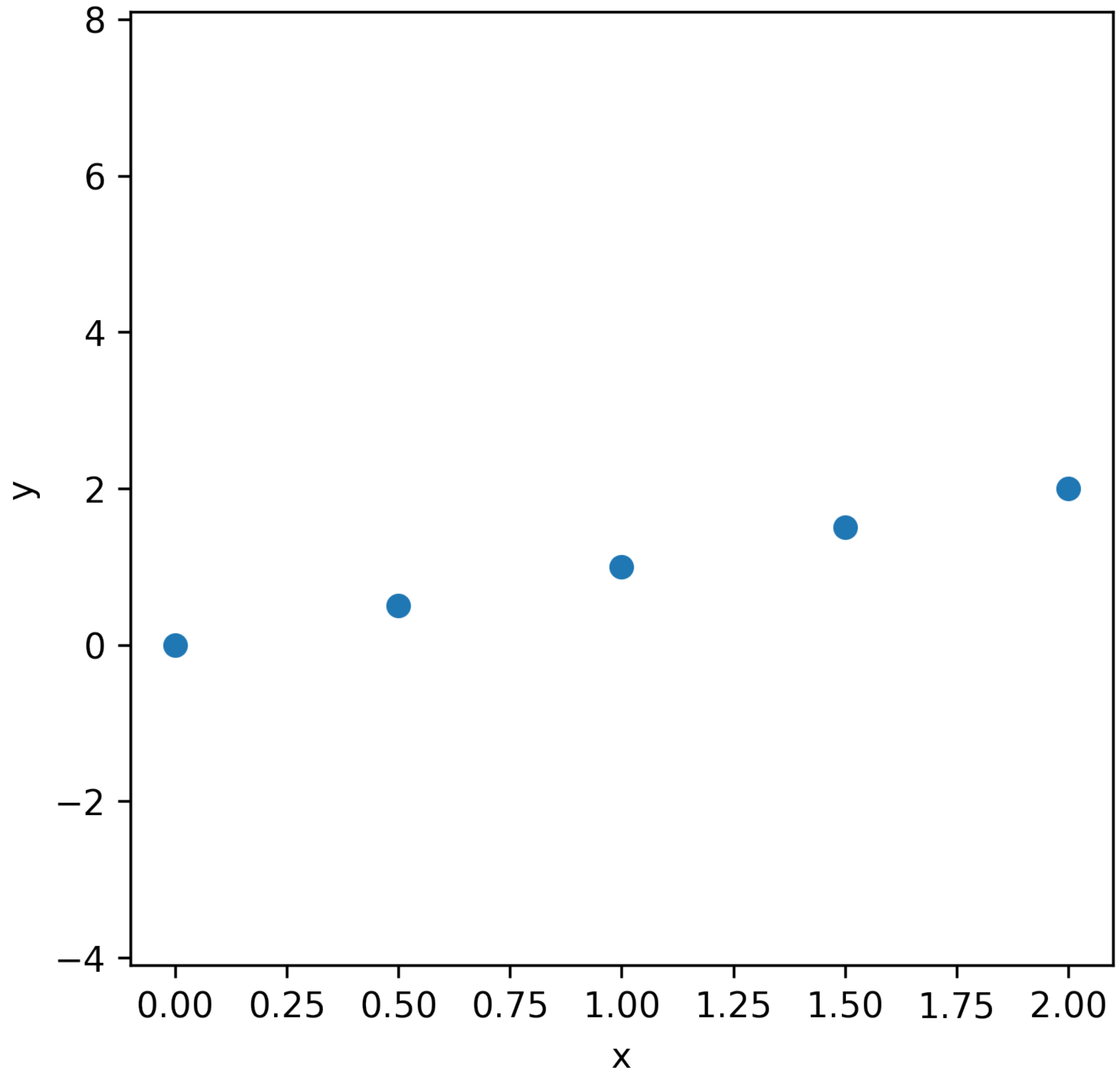
$$\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

$$\frac{1}{m}\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

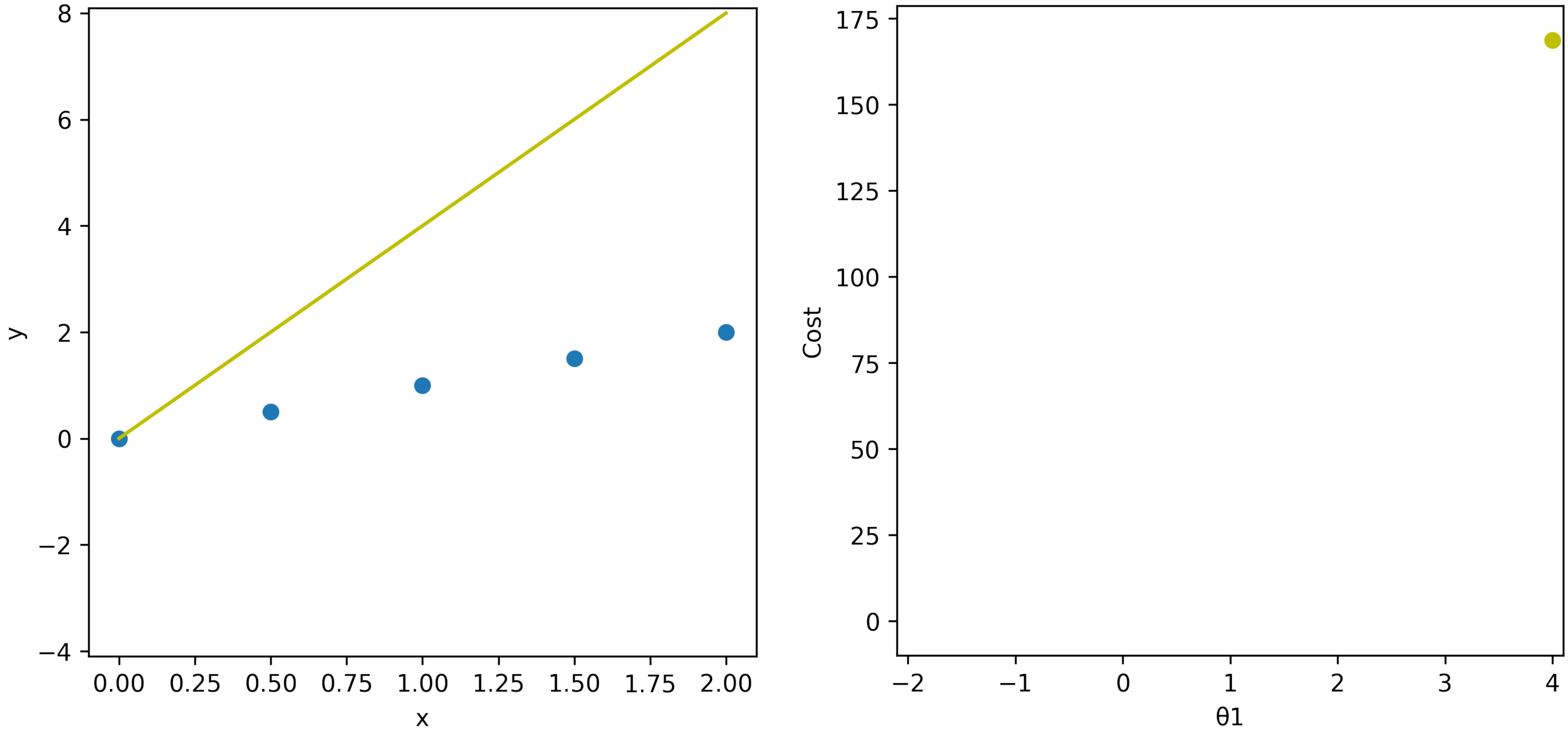
$$\frac{1}{2m}\sum_{i=1}^{m} (\hat{y}^{(i)} - y^{(i)})^2$$

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

Minimize a simplified J

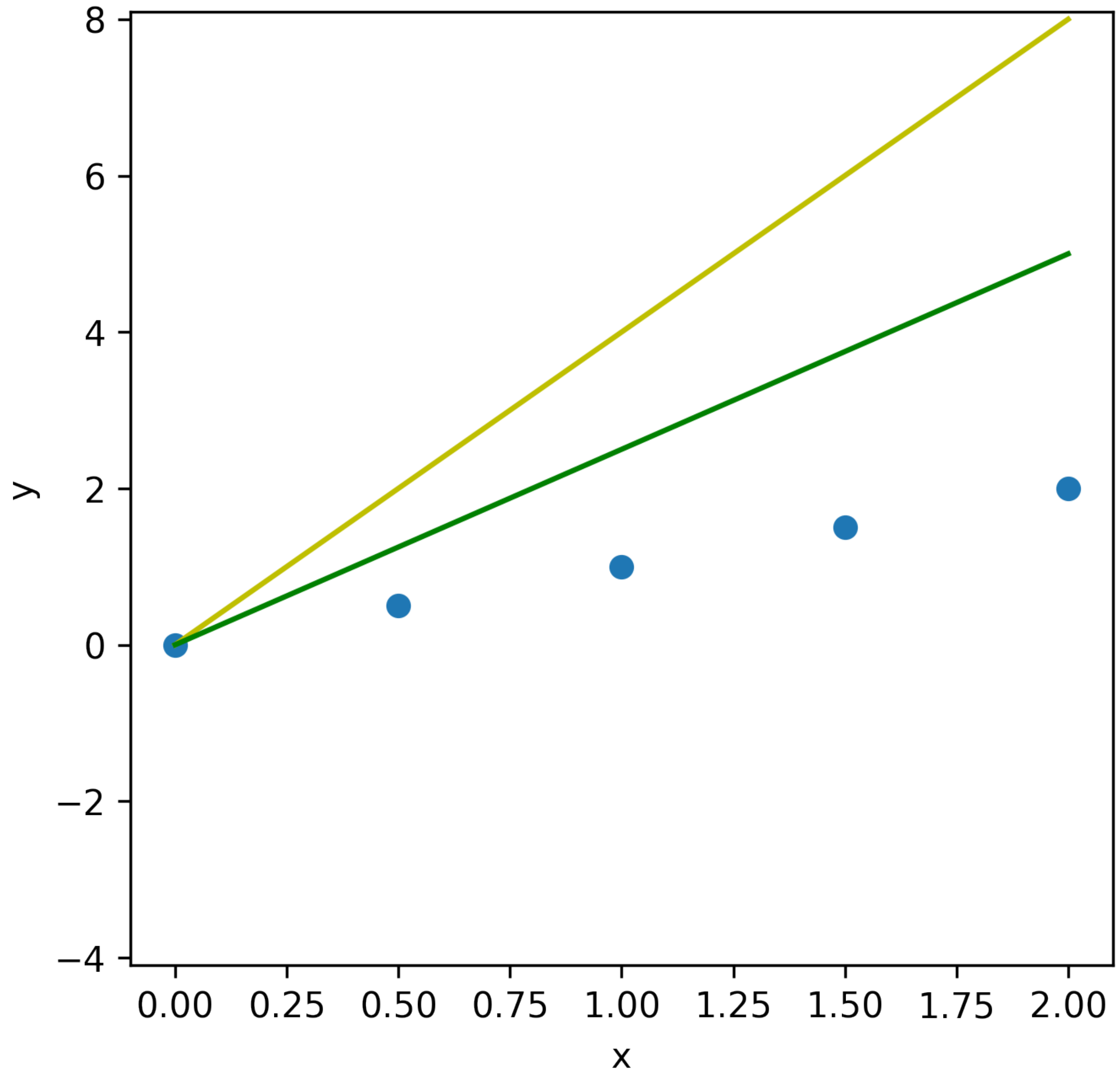


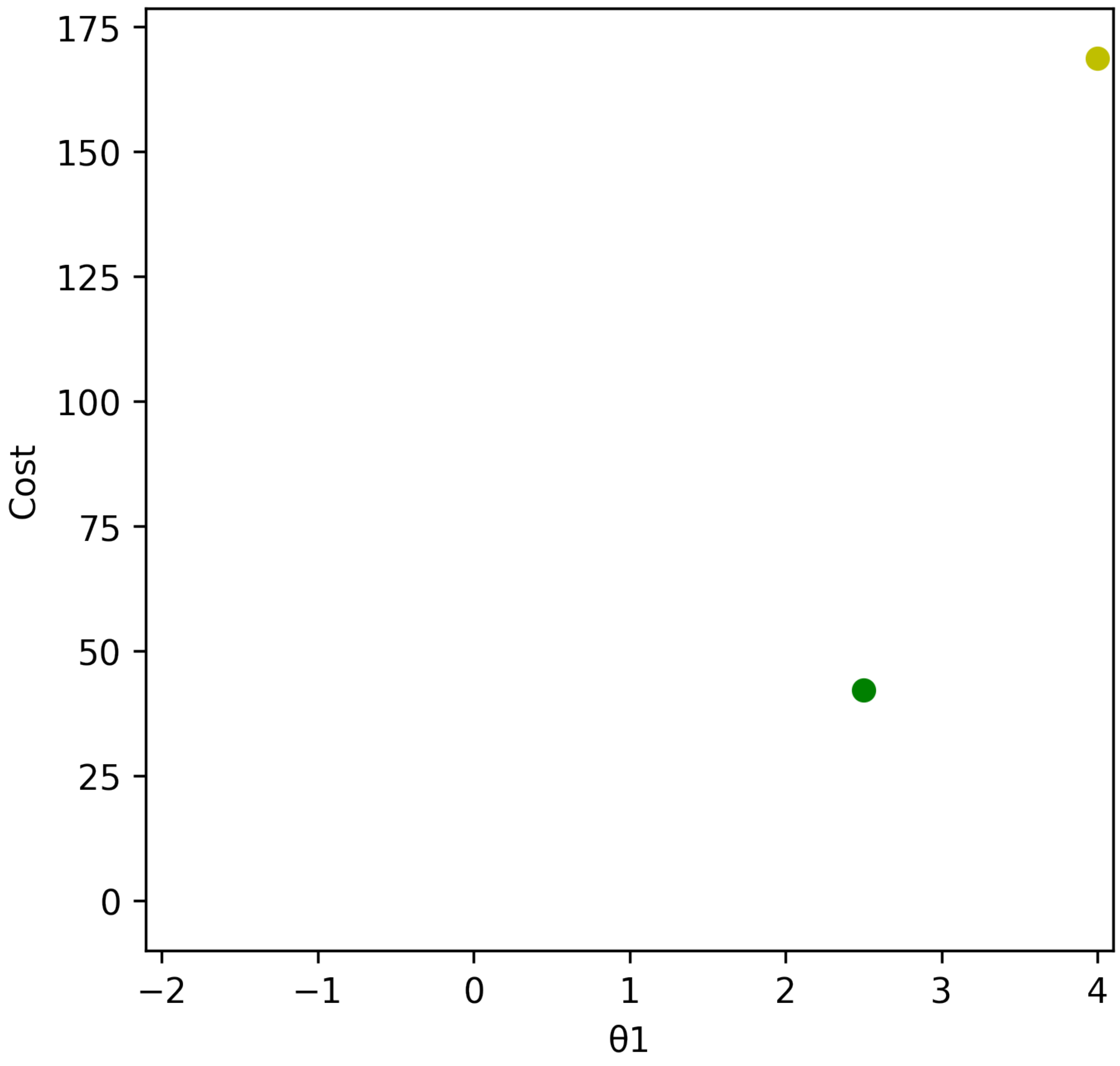
Minimize a simplified J



$$J(0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (0 + \theta_1 x^{(i)} - y^{(i)})^2$$

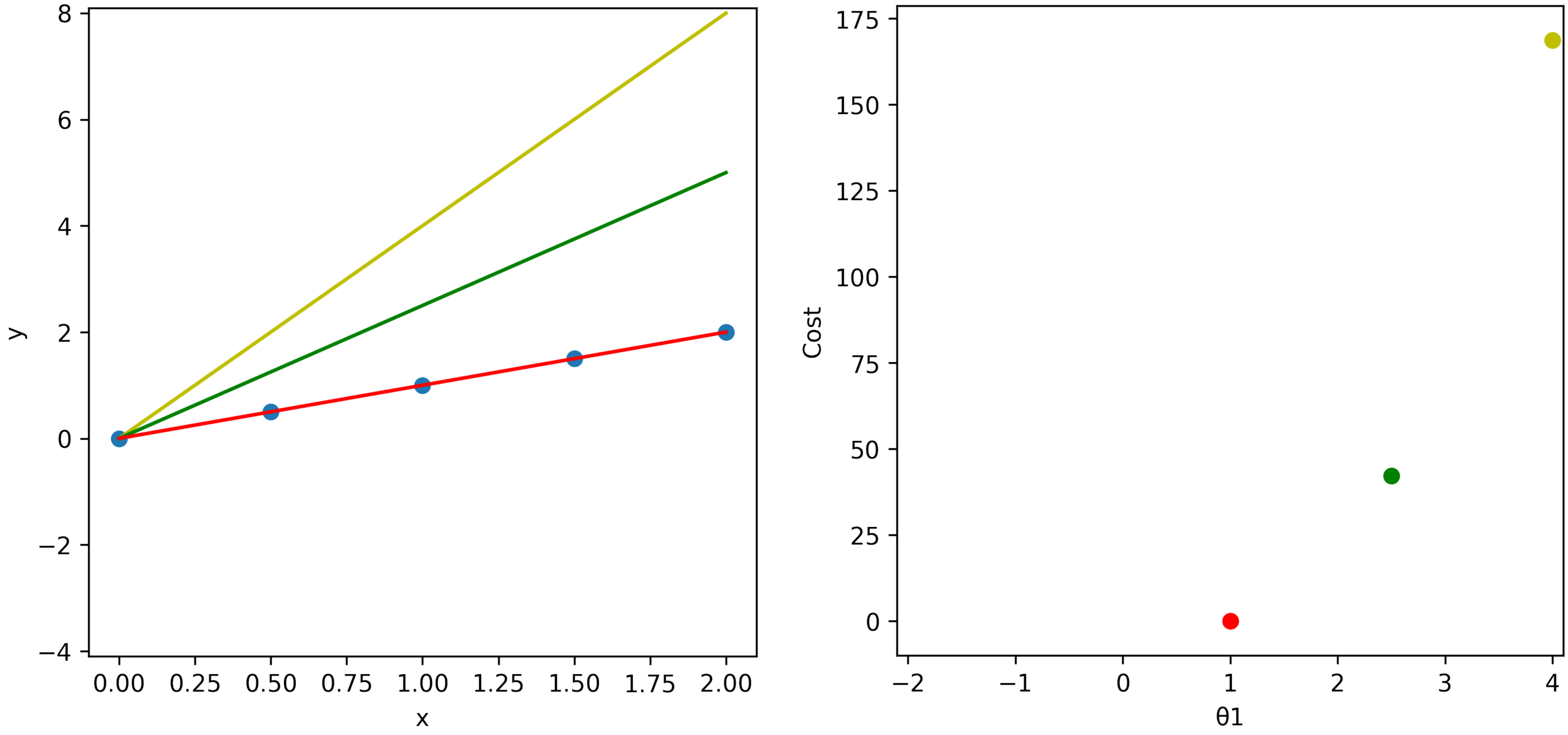
Minimize a simplified J





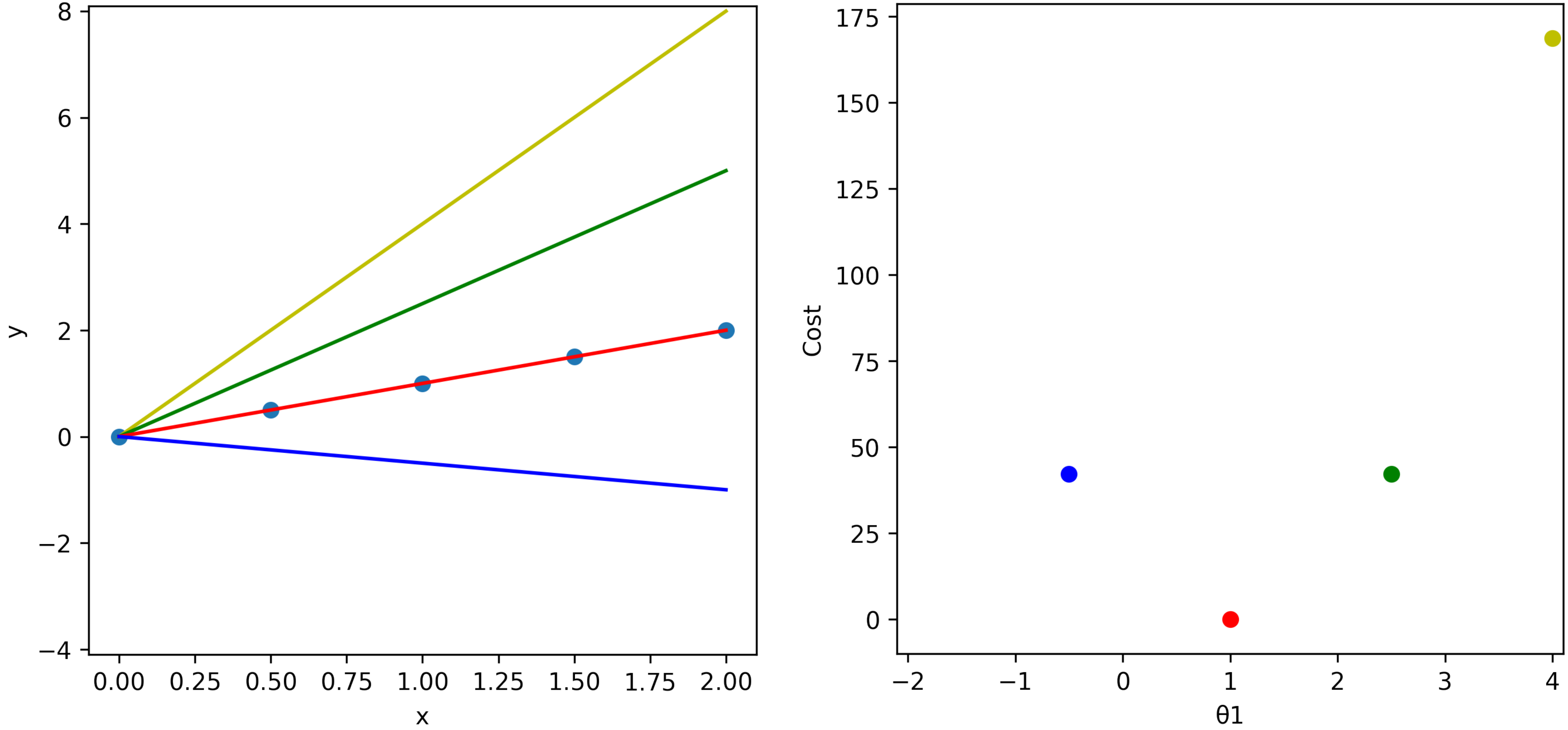
$$J(0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (0 + \theta_1 x^{(i)} - y^{(i)})^2$$

Minimize a simplified J



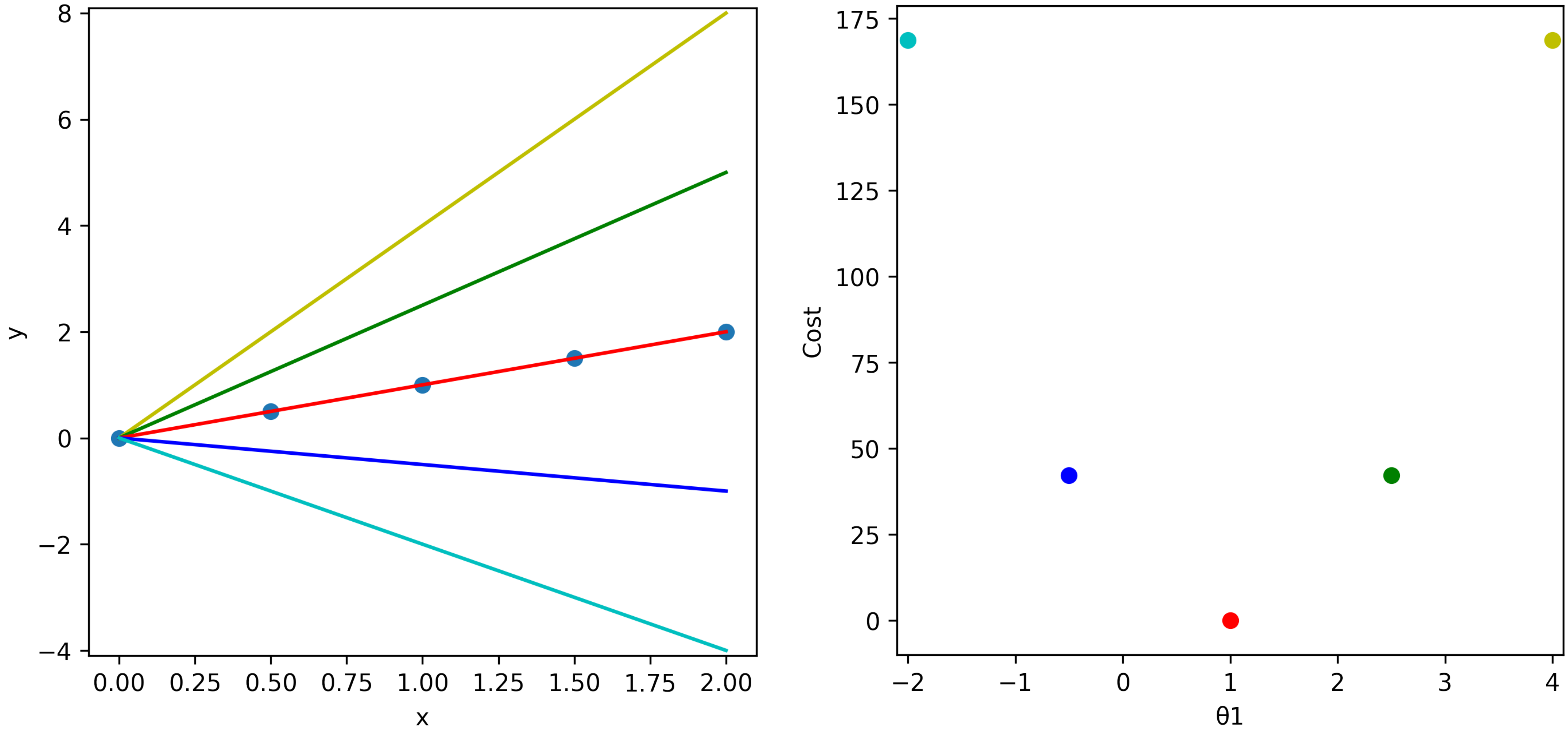
$$J(0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(0+\theta_1 x^{(i)} - y^{(i)})^2$$

Minimize a simplified J



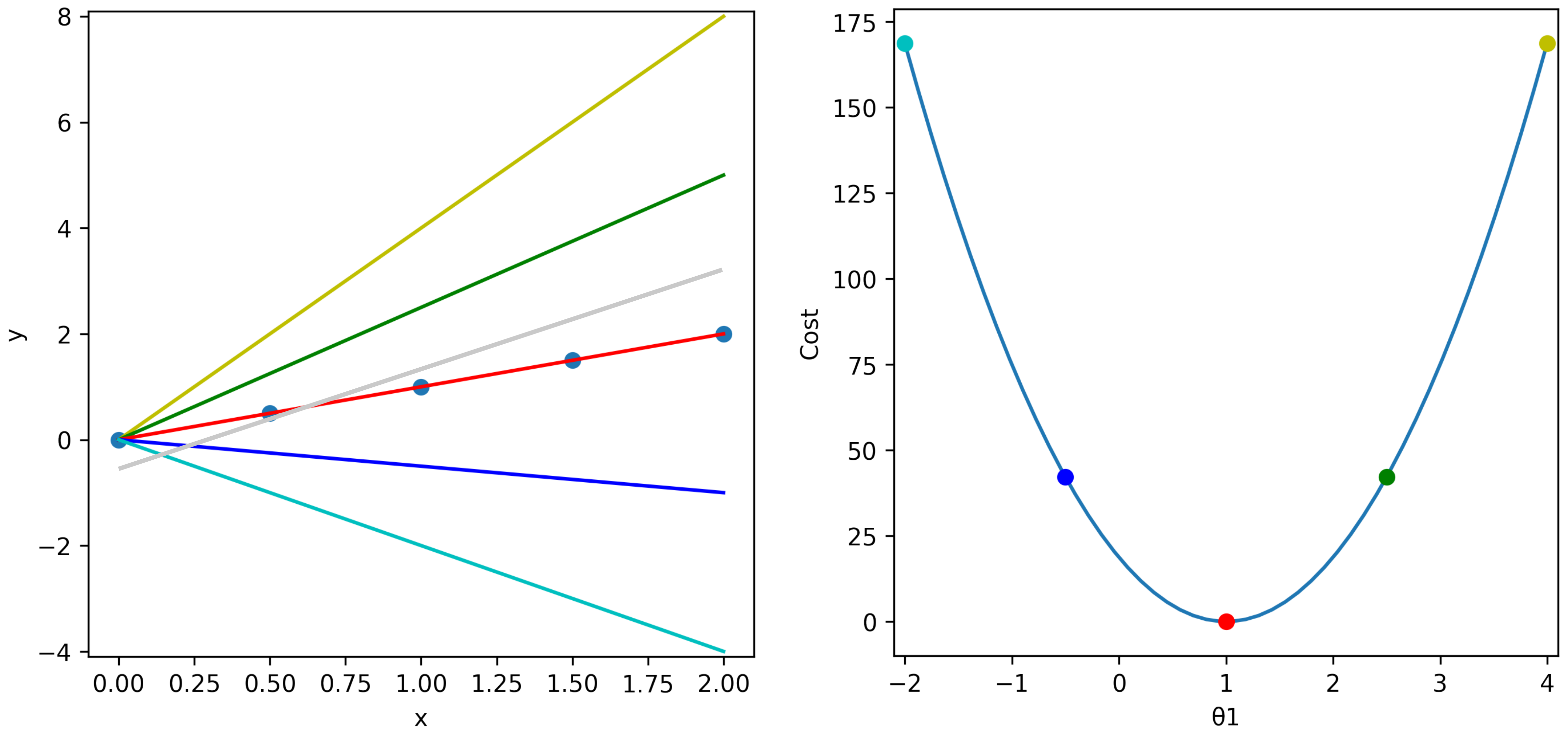
$$J(0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(0 + \theta_1 x^{(i)} - y^{(i)})^2$$

Minimize a simplified J



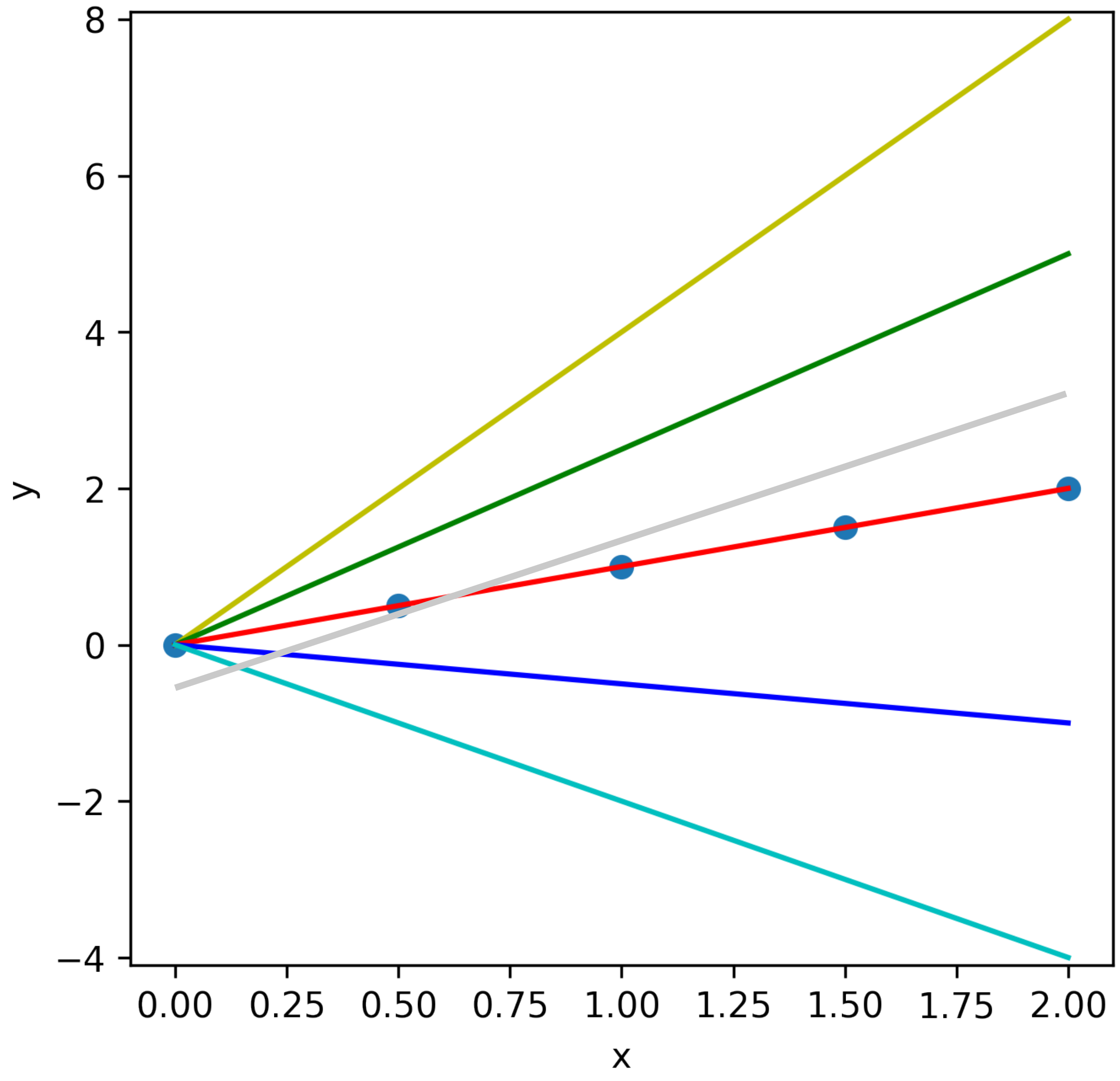
$$J(0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(0+\theta_1 x^{(i)} - y^{(i)})^2$$

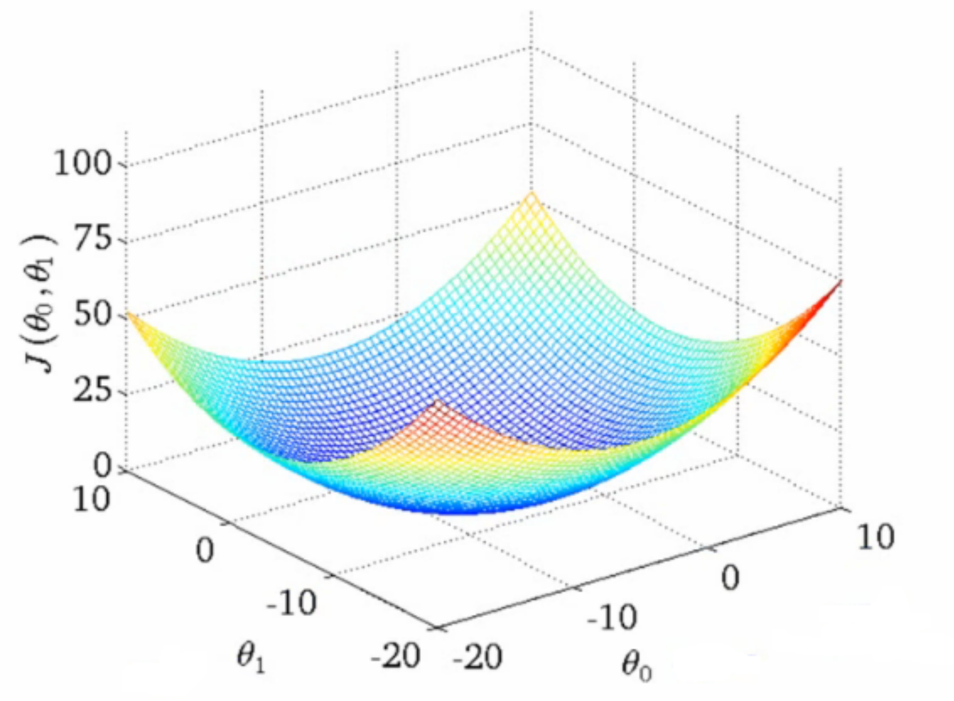
Minimize a simplified J



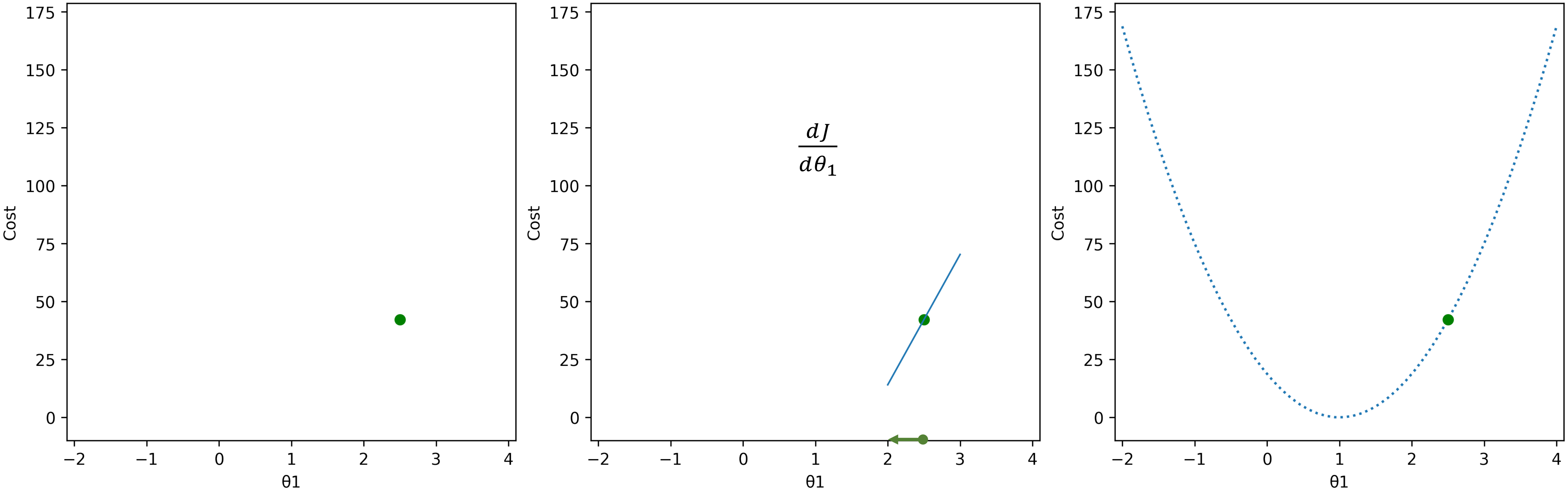
$$J(0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(0 + \theta_1 x^{(i)} - y^{(i)})^2$$

Minimize a simplified J



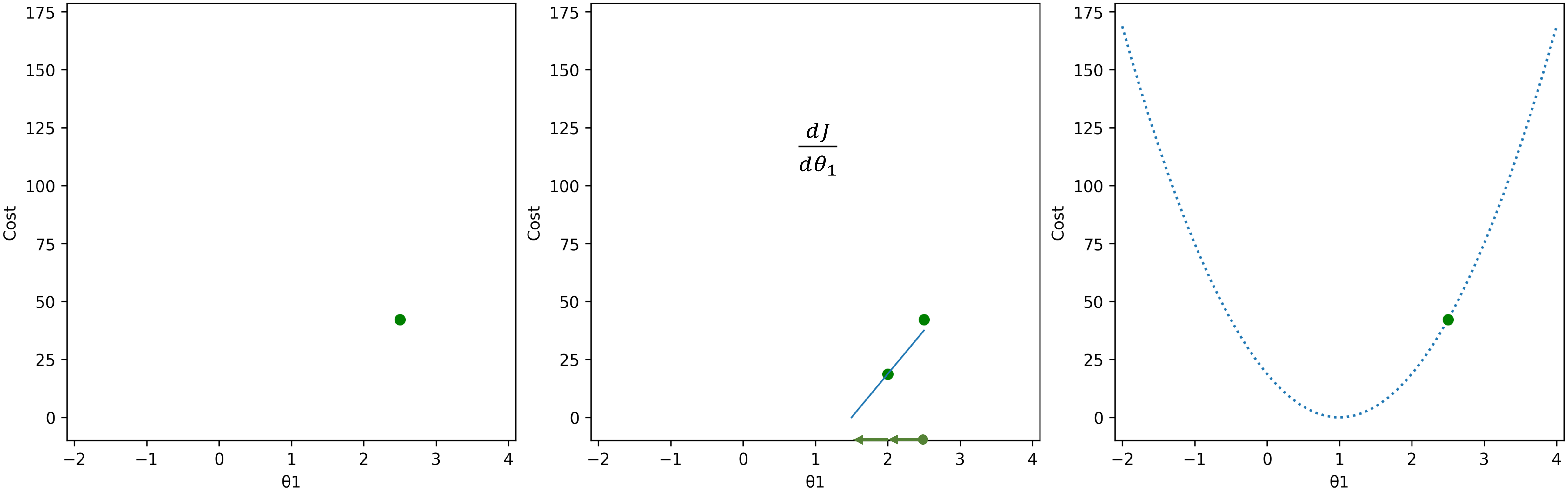


Gradient Descent



$$2 = 2.5 - 0.5$$

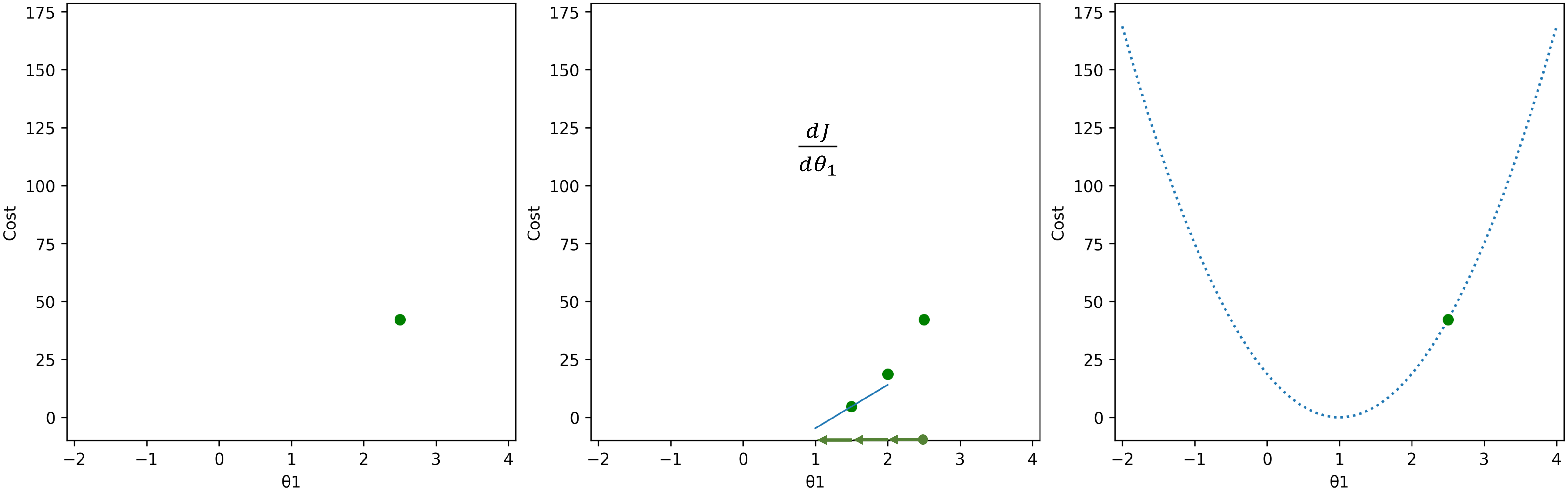
Gradient Descent



$$2 = 2.5 - 0.5$$

$$1.5 = 2 - 0.5$$

Gradient Descent

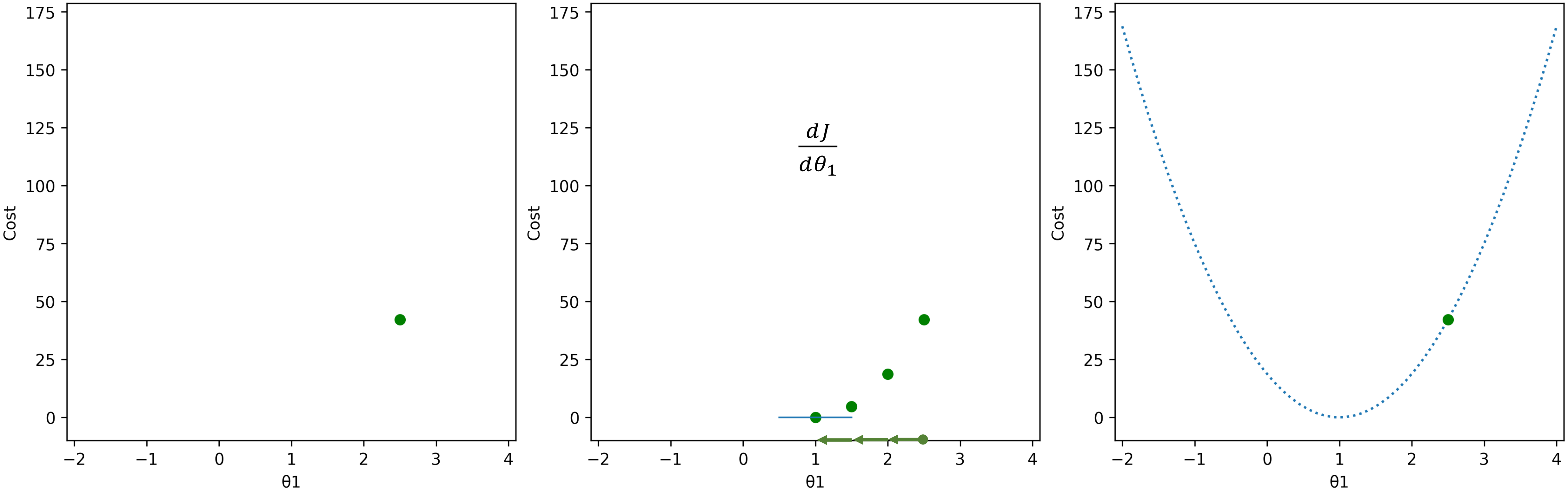


$$2 = 2.5 - 0.5$$

$$1.5 = 2 - 0.5$$

$$1 = 1.5 - 0.5$$

Gradient Descent



Repeat until slope = 0

$$\theta_1 = \theta_1 - 0.5$$

$$2 = 2.5 - 0.5$$

$$1.5 = 2 - 0.5$$

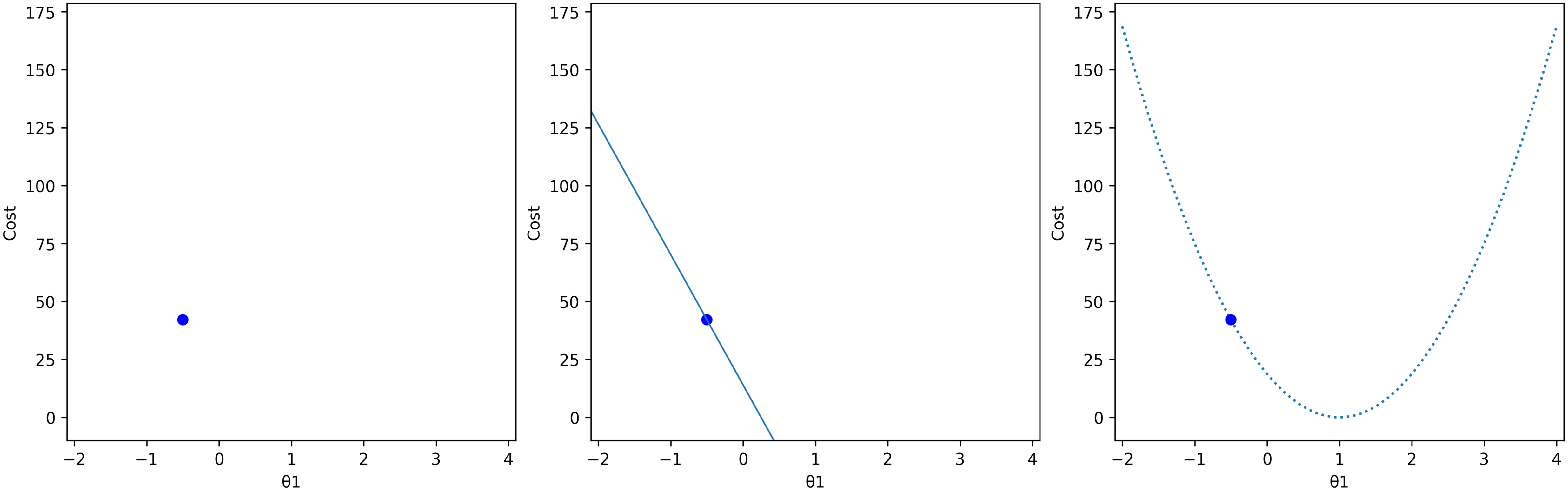
$$1 = 1.5 - 0.5$$

Repeat until convergence

$$\theta_1 = \theta_1 - \alpha \frac{dJ}{d\theta_1}$$

Learning rate (lr)

Gradient Descent



Repeat until convergence

$$\theta_1 = \theta_1 - \alpha \frac{dJ}{d\theta_1}$$

Gradient Descent

$$\theta_0 = \theta_0 - \alpha \frac{\partial J}{\partial \theta_0}$$

$$\theta_1 = \theta_1 - \alpha \frac{\partial J}{\partial \theta_1}$$

$$\theta_2 = \theta_2 - \alpha \frac{\partial J}{\partial \theta_2}$$

Repeat until convergence

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$ j = 0,1,2…

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

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

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

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

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

Dot product

x	y ?
-1	-3
0	-1
1	1
3	5
4	7

$$\theta_0 + \theta_1 * x = y$$

$$\theta_0 = -1$$
$$\theta_1 = 2$$

$$X \cdot \theta = y$$

```
X
```

```
array([[ 1., -1.],
       [ 1.,  0.],
       [ 1.,  1.],
       [ 1.,  3.],
       [ 1.,  4.]])
```

```
theta
```

```
array([[-1],
       [ 2]])
```

```
np.dot(X, theta)
```

```
array([[-3.],
       [-1.],
       [ 1.],
       [ 5.],
       [ 7.]])
```

```
X.shape
```

```
(5, 2)
```

```
theta.shape
```

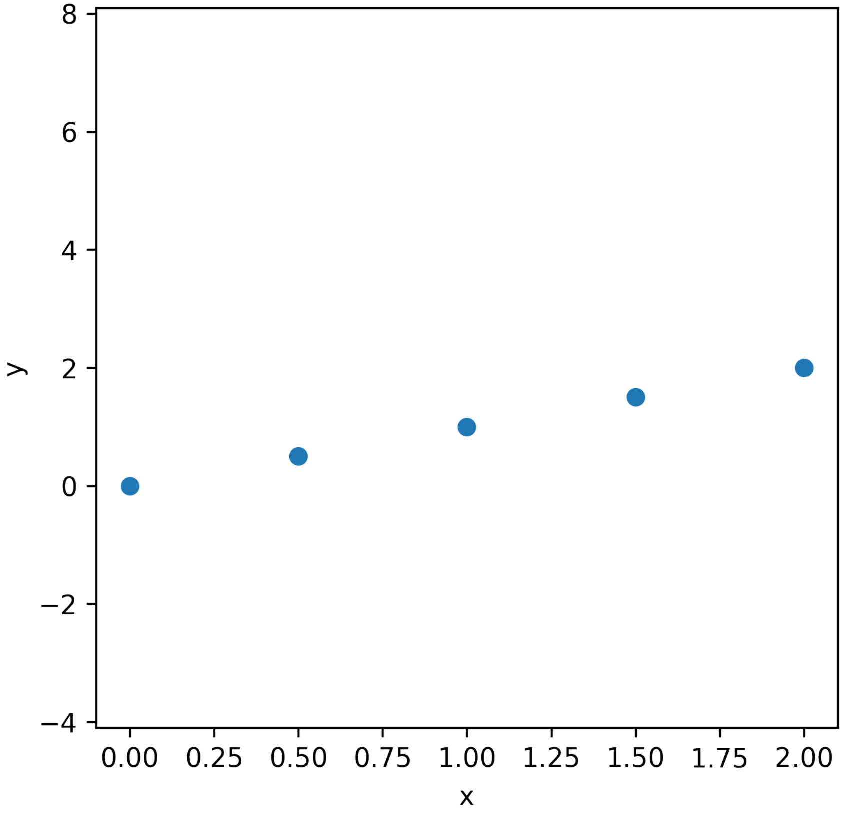
```
(2, 1)
```

```
theta.shape
```

```
(2,)
```

np.reshape()
np.newaxis

The Challenge




```
def cost_function(X, y, theta):
```

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

```
def GD_one_step(X, y, theta, lr):
```

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

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

```
def GD(X, y, lr, epoch):
```

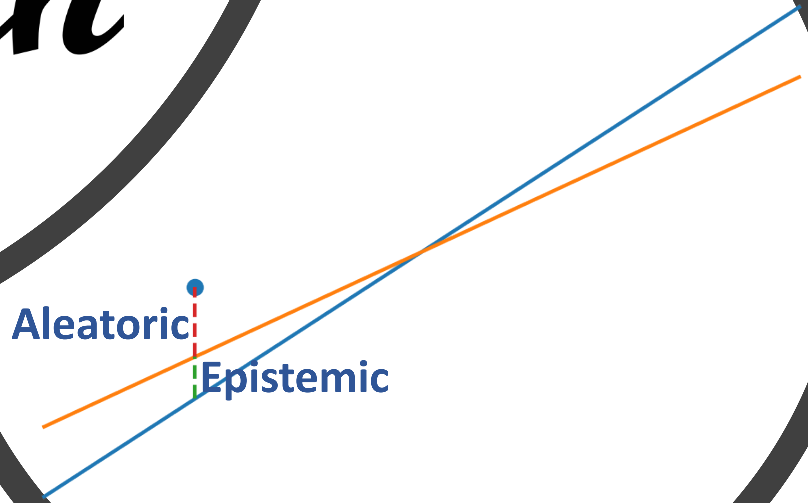
Repeat until convergence

Challenge 1
Linear regression

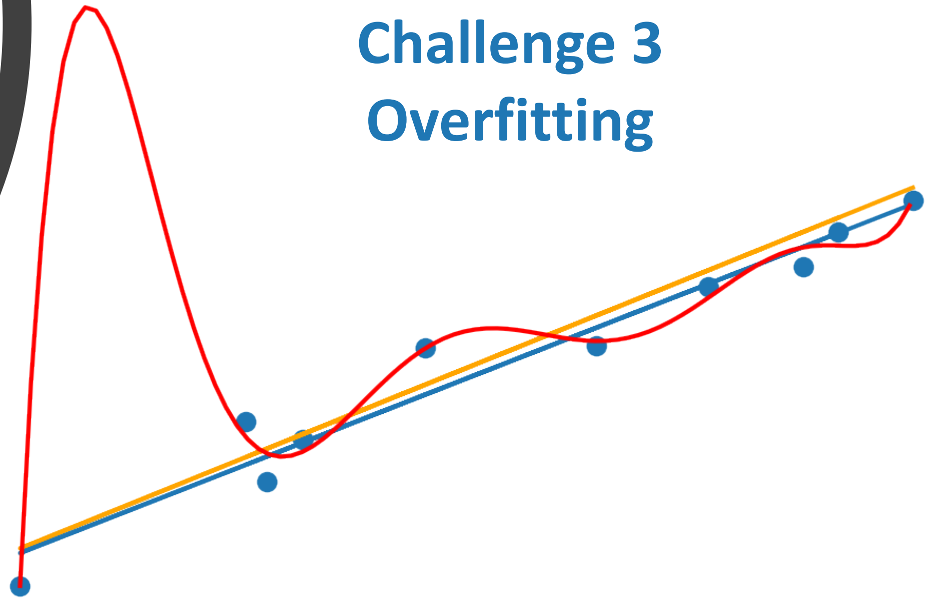
Challenge 4
Implement Linear regression

$$\theta_j = \theta_j - \alpha \frac{\partial J}{\partial \theta_j}$$

Challenge 2



Challenge 3
Overfitting



Good luck!

@malicannoyan