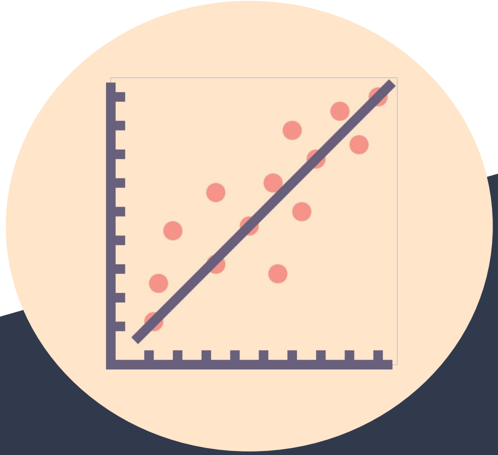
ECE ING4
MACHINE LEARNING

Jeremy Cohen

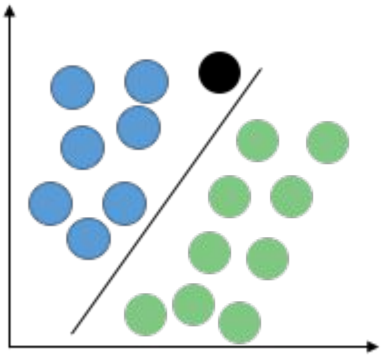


Linear Regression

Week 1 Review

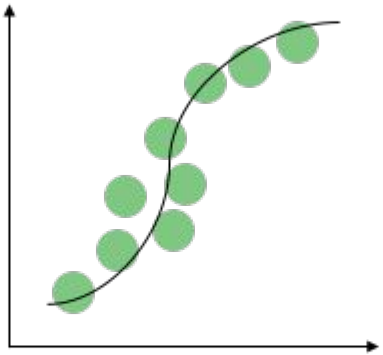
Types of Machine Learning

CLASSIFICATION



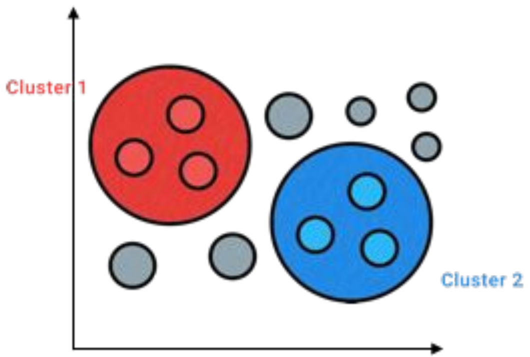
- Supervised
- Output is a discrete number (0,1,2, ...), (SPAM/NOT SPAM), ...

REGRESSION



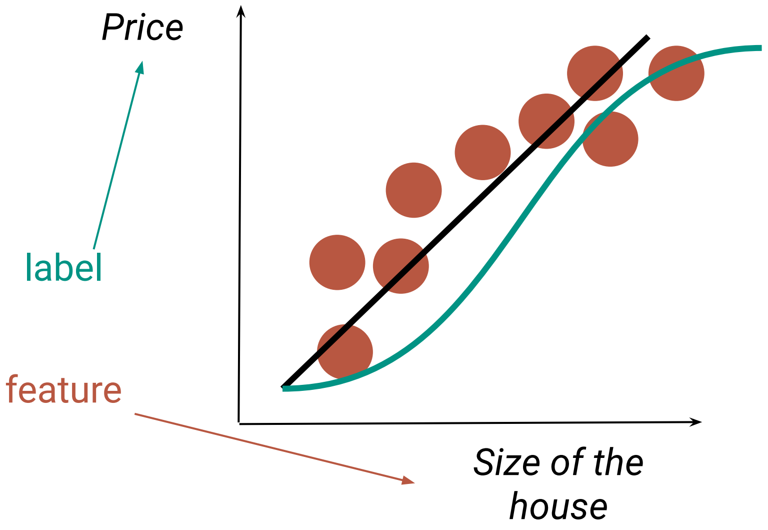
- Supervised
- Output is a continuous number

CLUSTERING



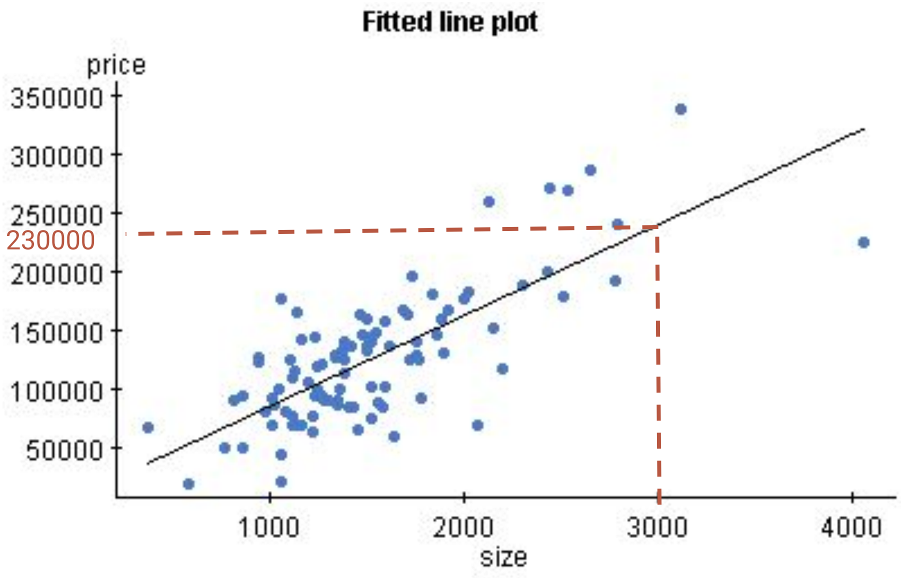
- Unsupervised
- Outputs are clusters

Regression



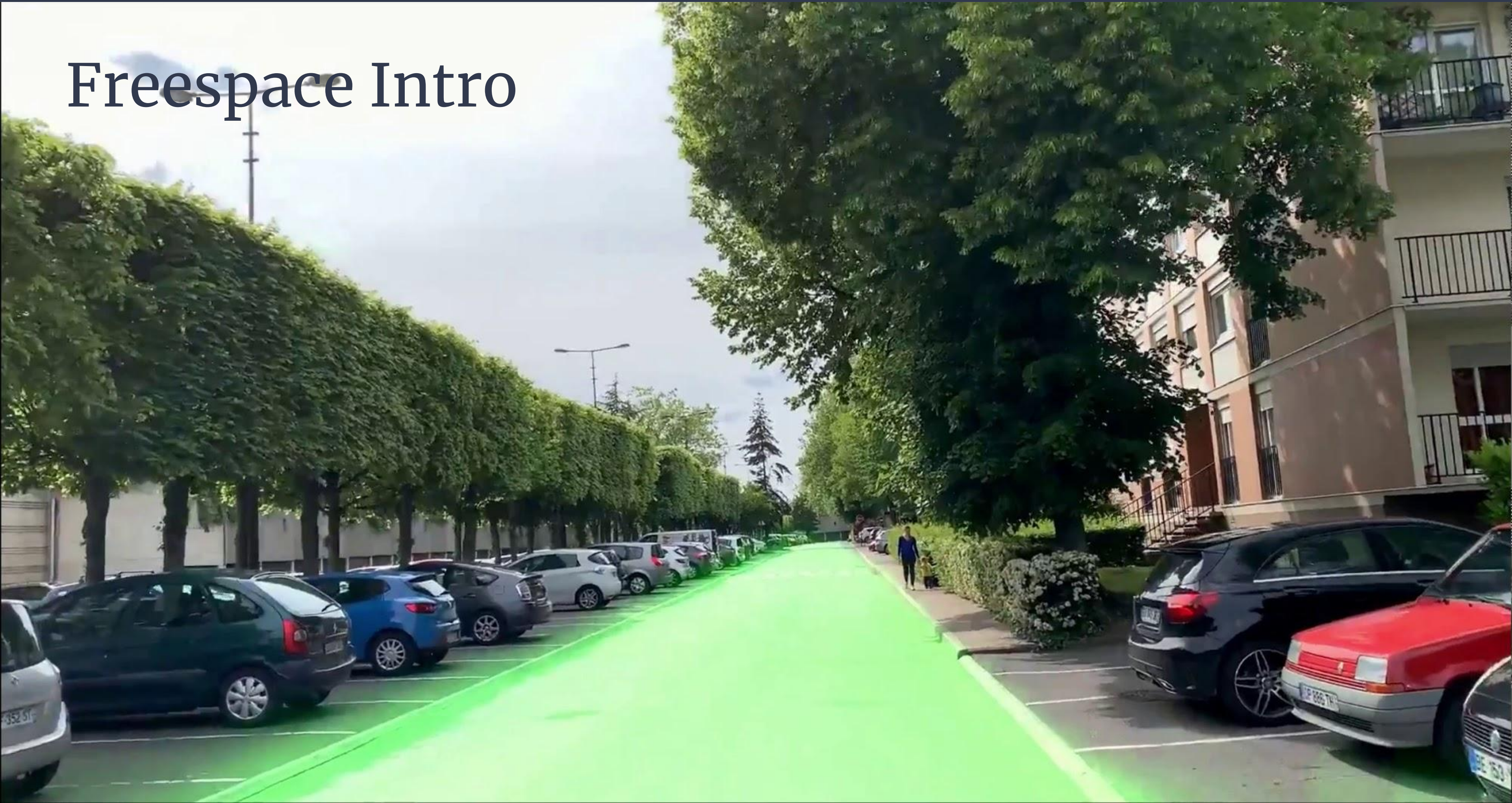
THE GOAL IS TO DETERMINE THE LINE OR CURVE THAT BEST FITS THE DATA

Regression

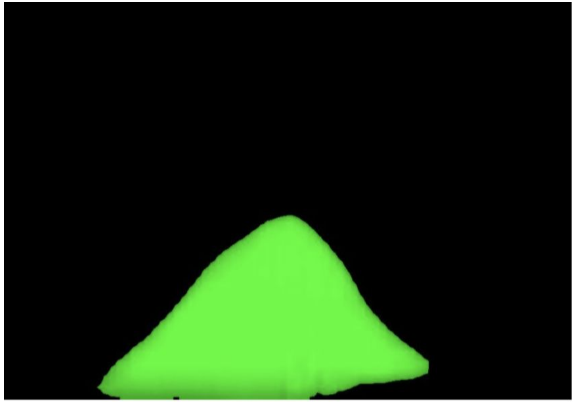


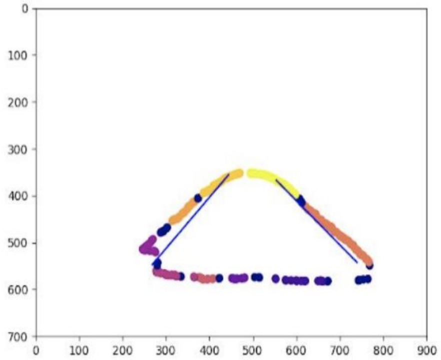
THE GOAL IS TO DETERMINE THE LINE OR CURVE THAT BEST FITS THE DATA

Freespace Intro



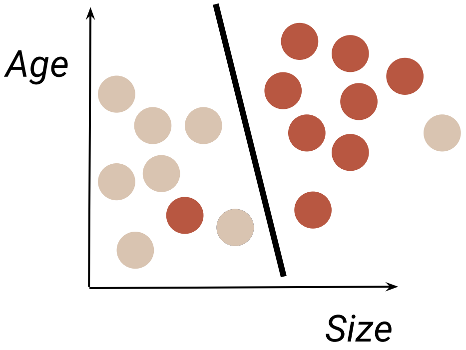
Regression in Practice



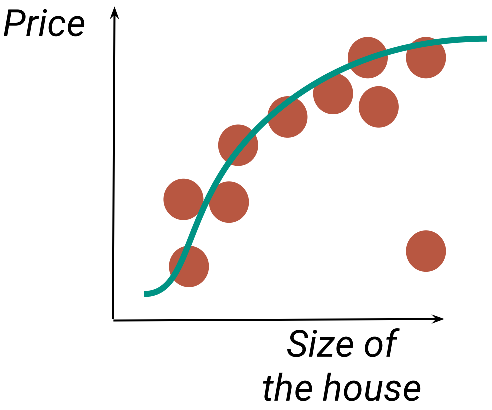


Outliers

CLASSIFICATION

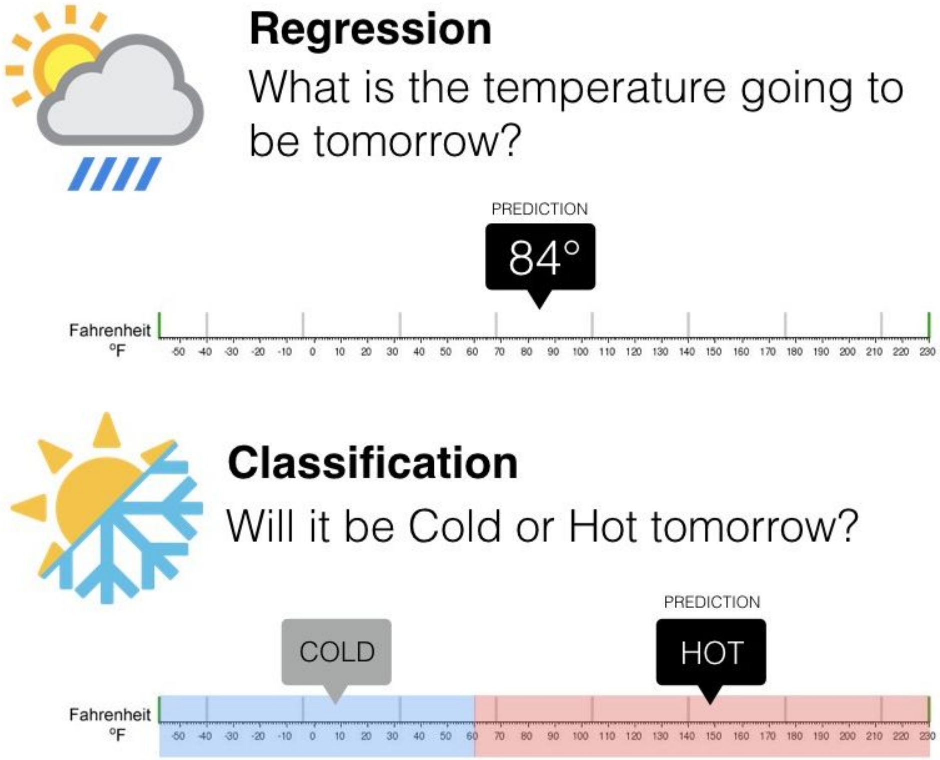


REGRESSION

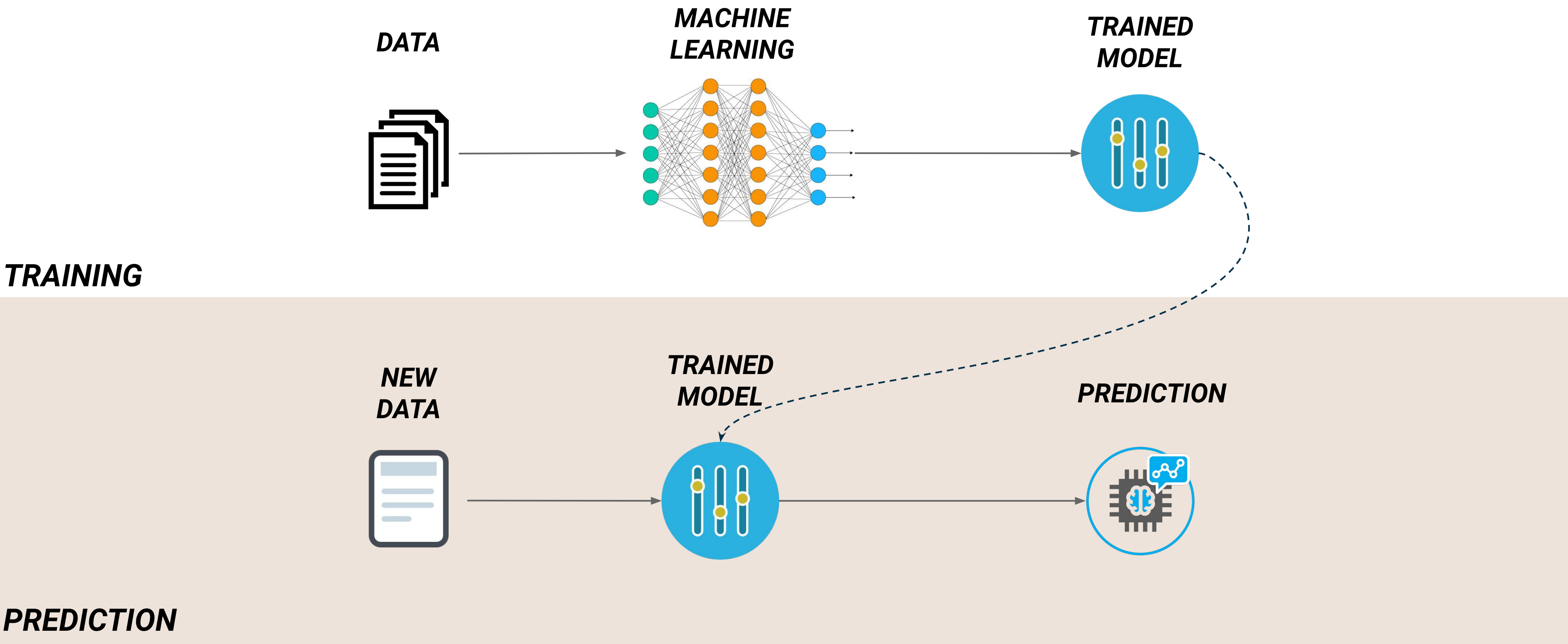


OUTLIERS ARE TO BE REMOVED WHEN TRAINING

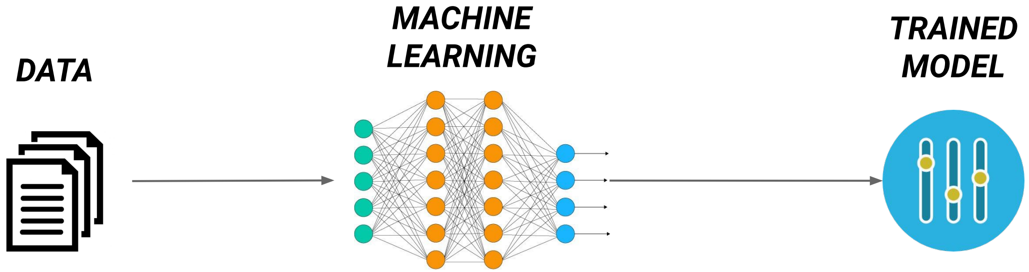
Classification vs Regression

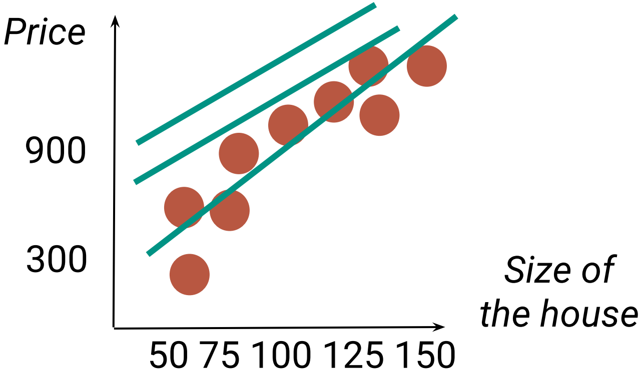


Machine Learning Process

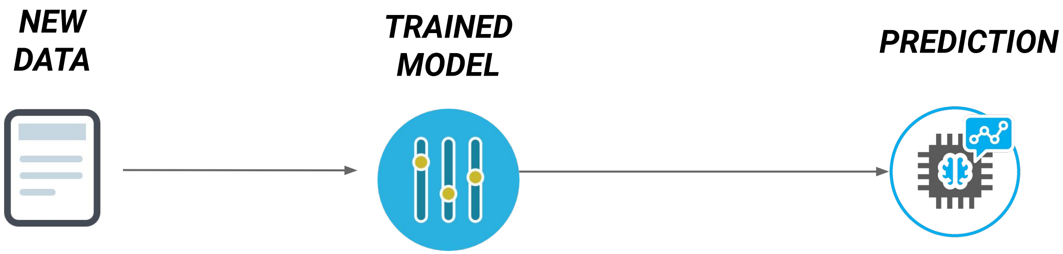


Training Linear Regression



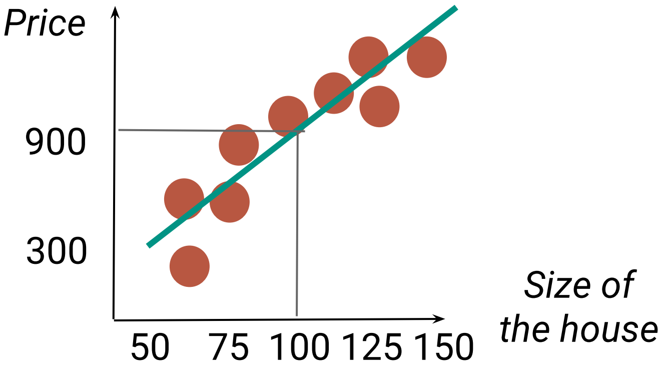


Testing Linear Regression



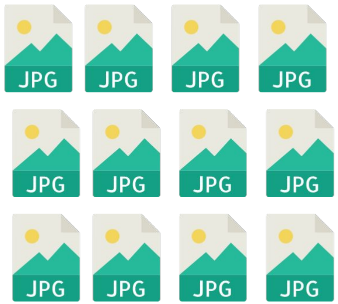
I want to buy a house of 100 square meters.
How much will it cost?

915 k€



3 Datasets

Training Set



~70% of the dataset

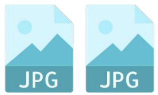
Used to **train the model**

Validation Set

~20% of the dataset

Used to **test** the model and **generalize better to new data**

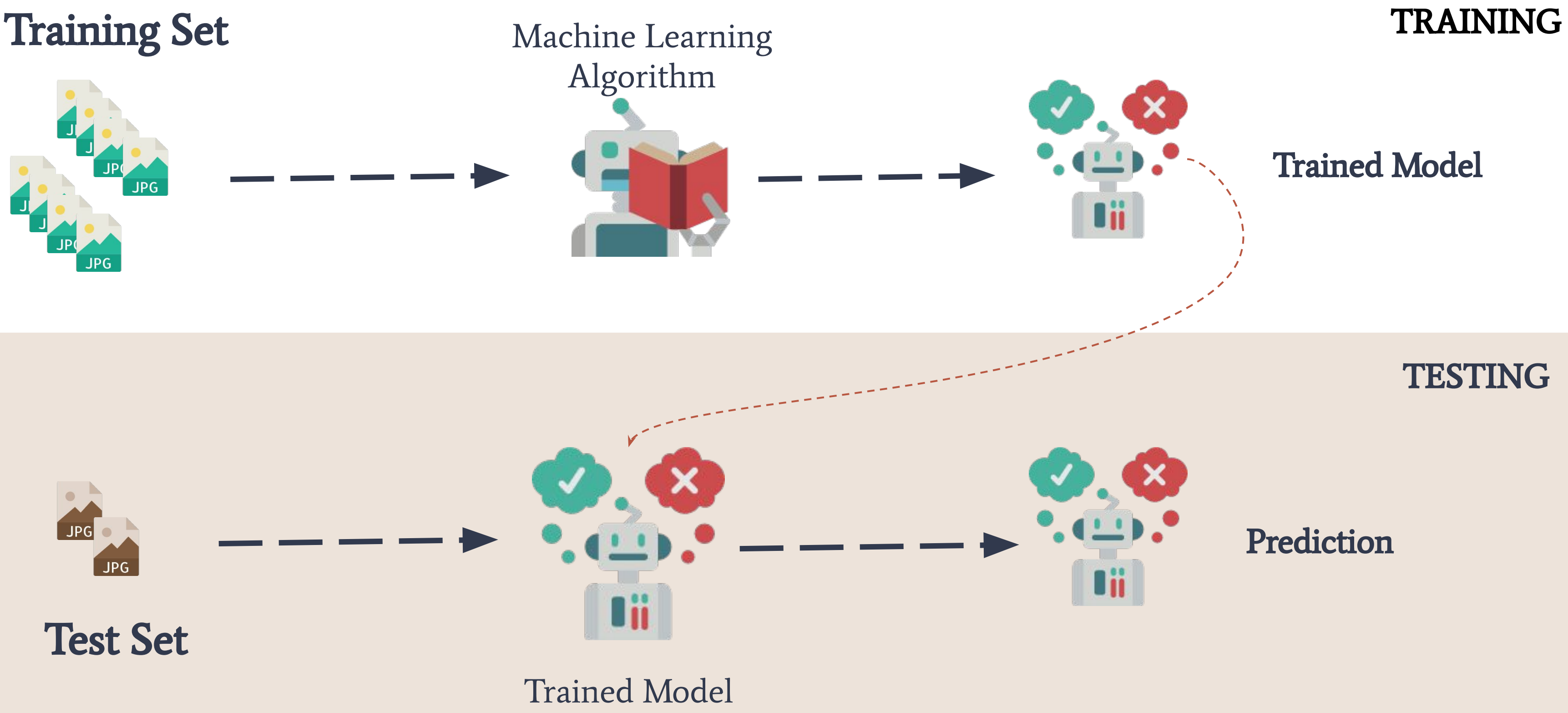
Test Set



~10% of the dataset

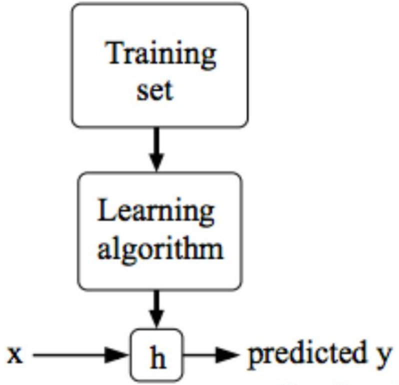
Used **only once** when both accuracies are good and ready for real-world

Train Test Process



Model

House characteristics (size, neighborhood, …): Feature X
House Price : Label Y



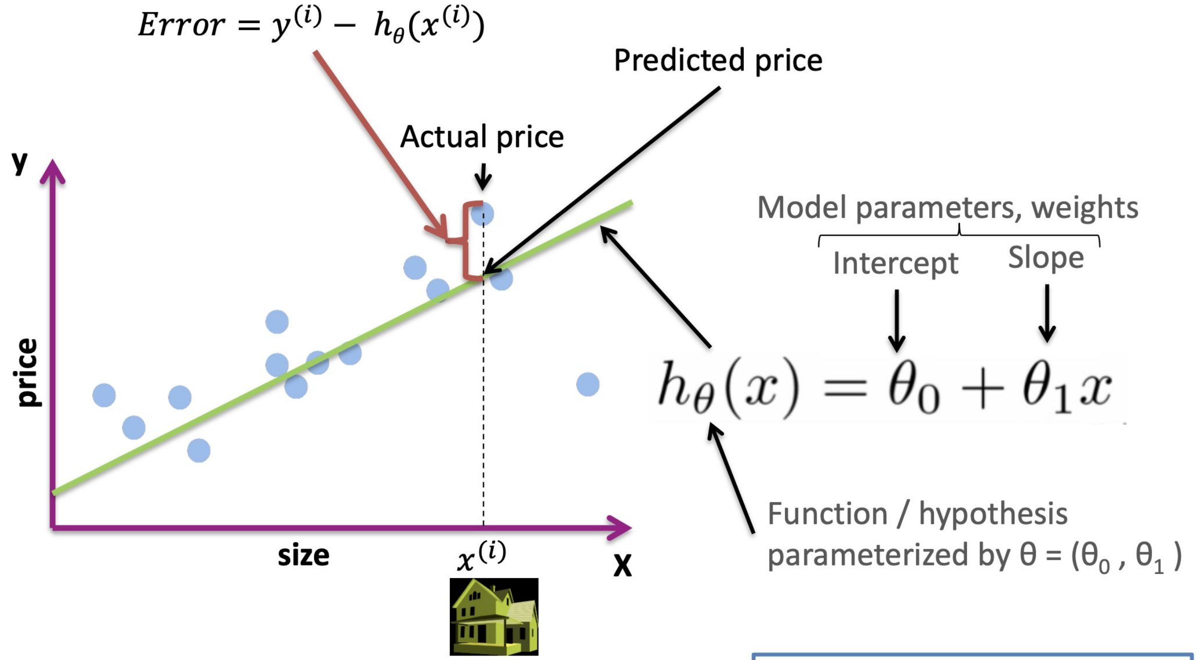
House characteristics

Predicted House price

h: X → Y

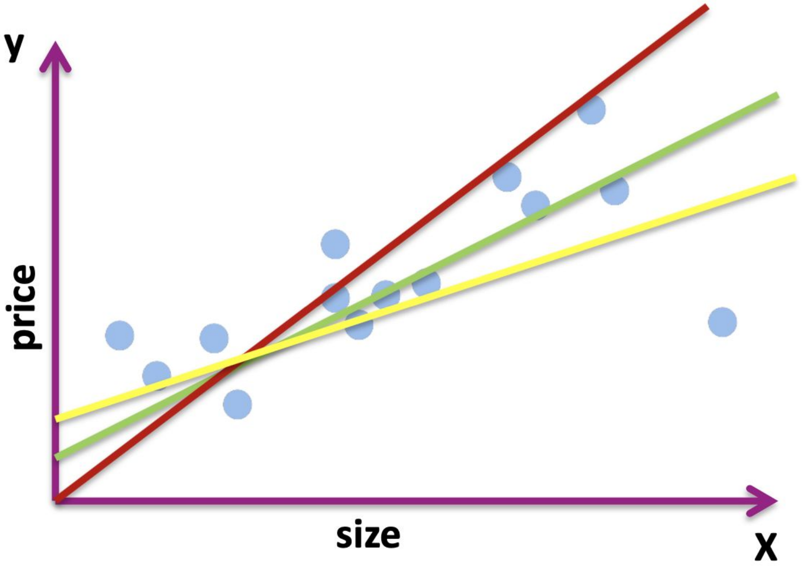
Hypothesis or function that takes as input the house's characteristics to estimate its price.

Simple Linear Regression



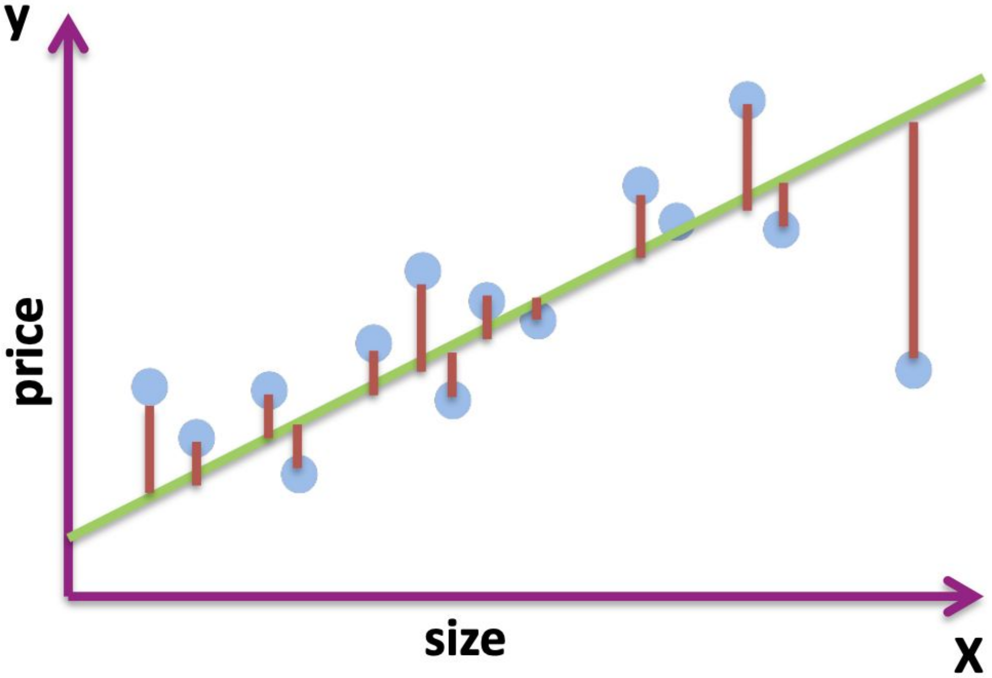
$$y = \theta_0 + \theta_1 x + \varepsilon$$

The Best Fit



Each line has different parameters $\theta = (\theta_0 , \theta_1)$
➔ different errors

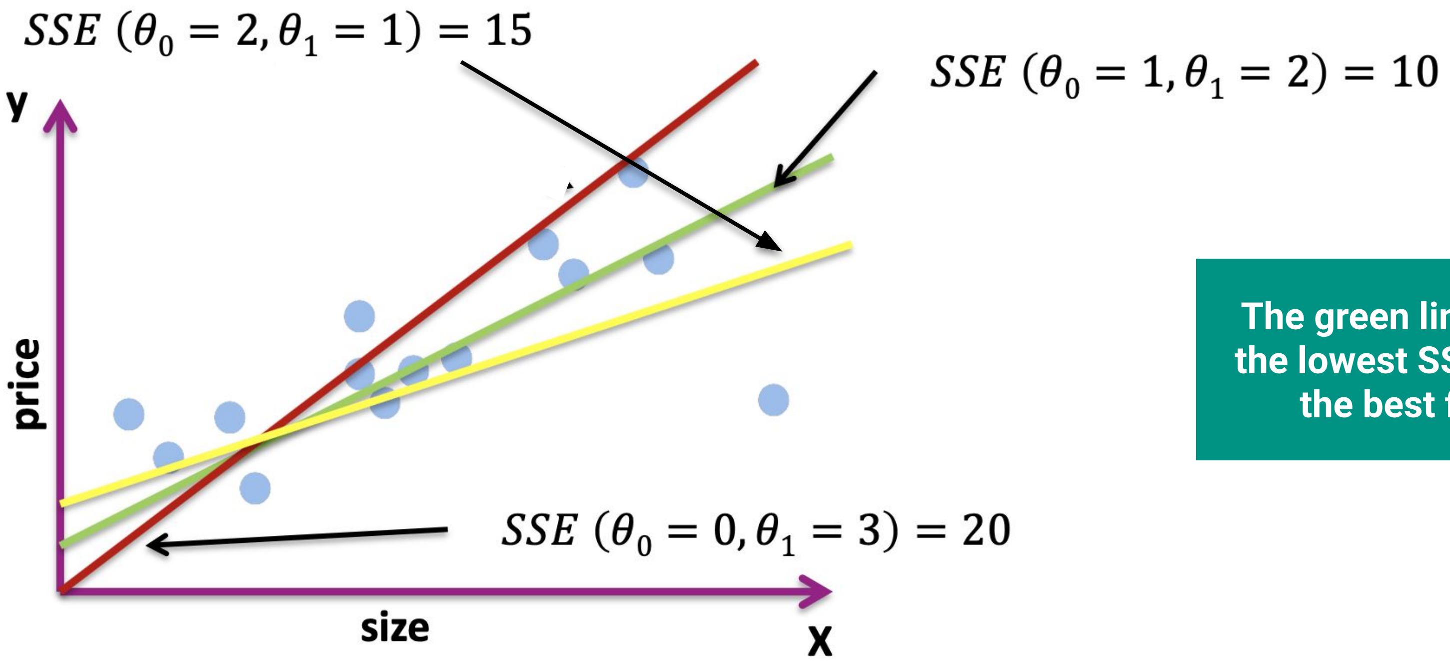
Sum of Squared Errors (SSE)



$$SSE\ (\theta_0, \theta_1) =$$

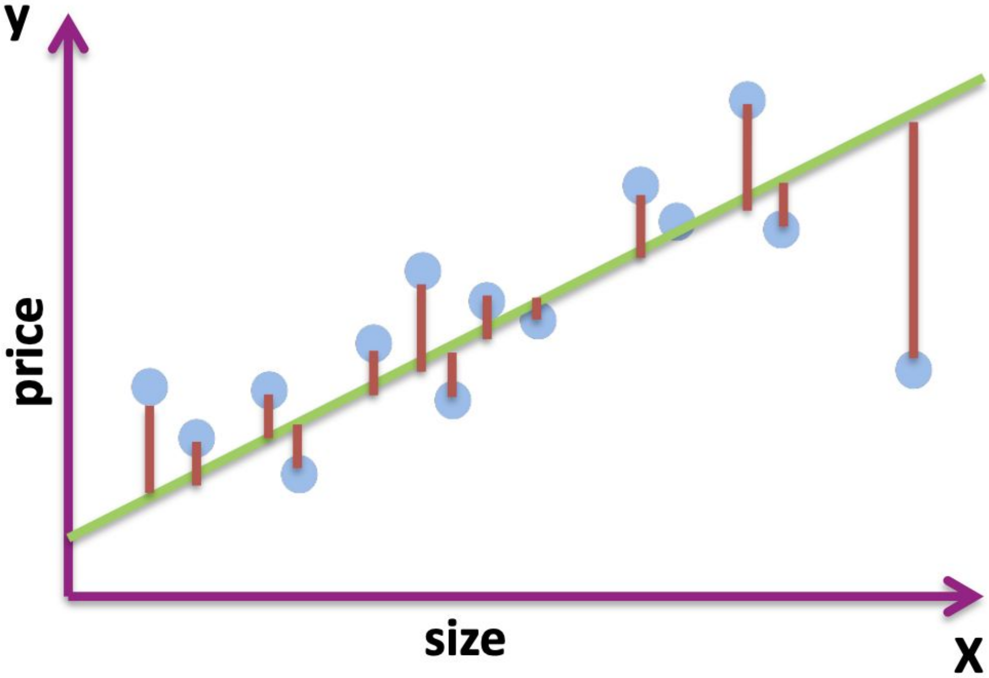
$$(\ y^{(1)} - (\theta_0 + \theta_1 * x^{(1)})\ )^2 +$$
$$(\ y^{(2)} - (\theta_0 + \theta_1 * x^{(2)})\ )^2 +$$
.... +
$$(\ y^{(m)} - (\theta_0 + \theta_1 * x^{(m)})\ )^2$$

The Best Fit



The green line has the lowest SSE and the best fit!

Mean Squared Error



$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

The Best Fit

The best hypothesis $h_\theta(x)$ is the one that minimizes the cost function

Sum over all data points

Actual

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

Predicted

1/m - means we determine the average

1/2m - simplifies the maths

The Best Fit

The learning algorithm should find $\theta^* = (\theta_0^*, \theta_1^*)$ that minimizes this cost

Several algorithms:

- Gradient descent
- Ordinary least square (OLS): Used in the linear regression in python (sklearn)

Gradient Descent

Notation - Simple Linear Regression

X Size (m^2)	Y Price (1000$)
20	300
37	540
88	986
...	...

m = 47

m: Number of examples

x(i): Input of the i-th training example

X(2) = [37]

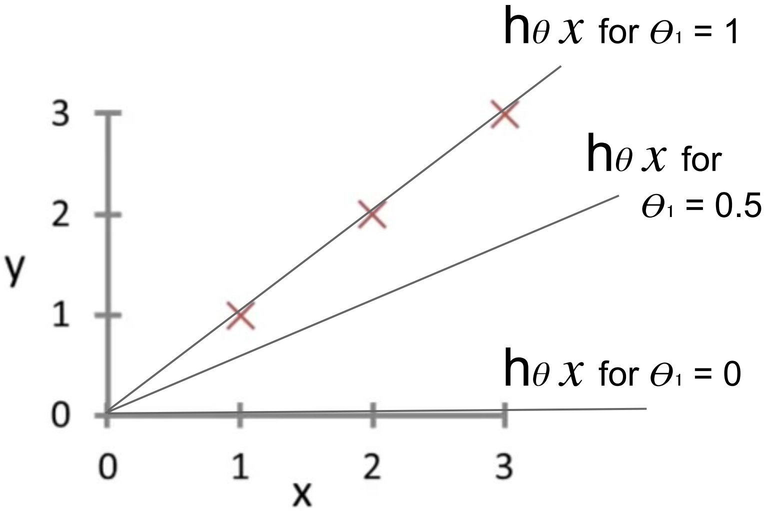
Parameter influence

Hypothesis : $\Theta_0 = 0$

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

Parameter influence

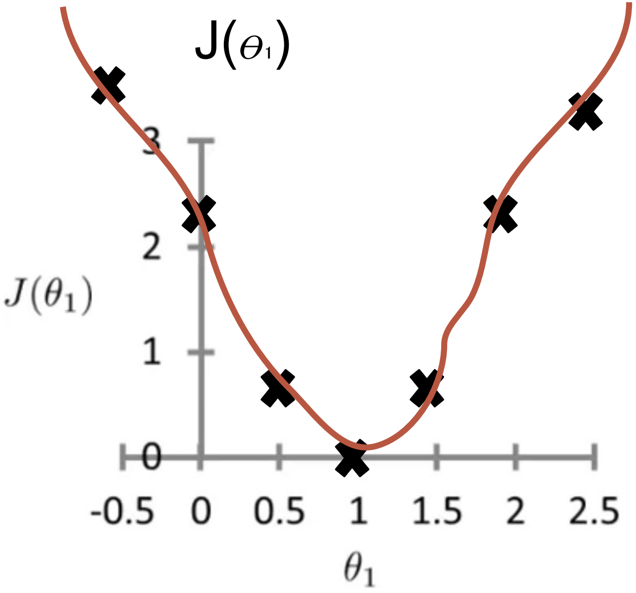
$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2.$$



$J(1) = 1/2m\ (0^2+0^2+0^2) = 0$

$J(0.5) = 1/2m\ ((0.5-1)^2+(1-2)^2+(1.5-3)^2) \sim 0.6$

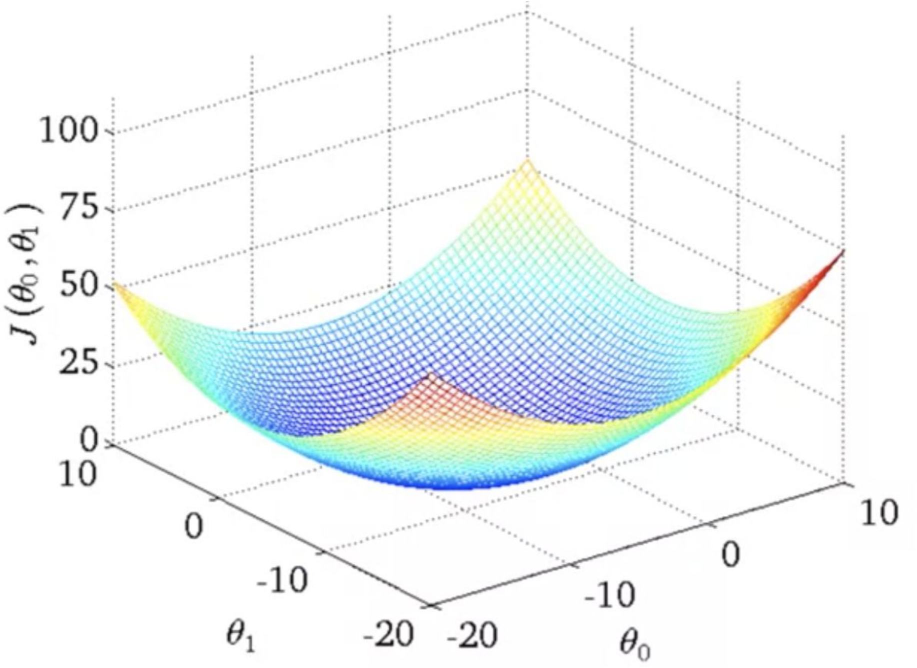
$J(0) = 14/6 \sim 2.3$



We want to minimize J(θ).

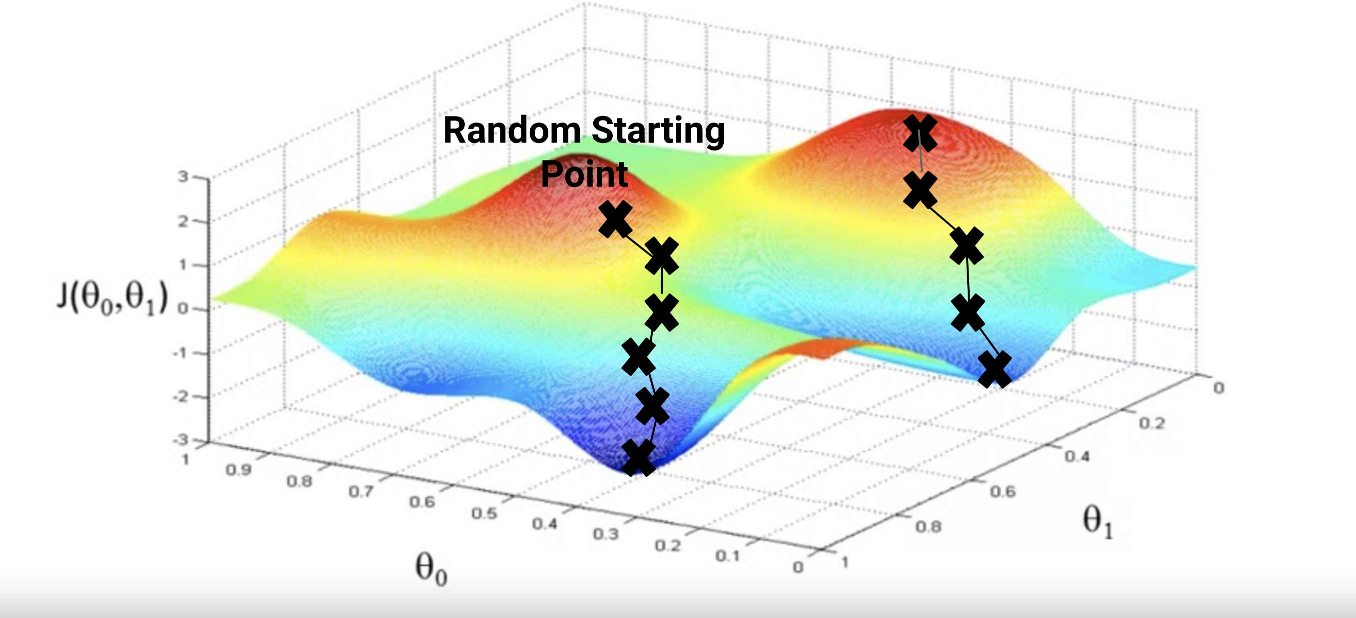
Parameter influence

$\Theta_0 \neq 0$



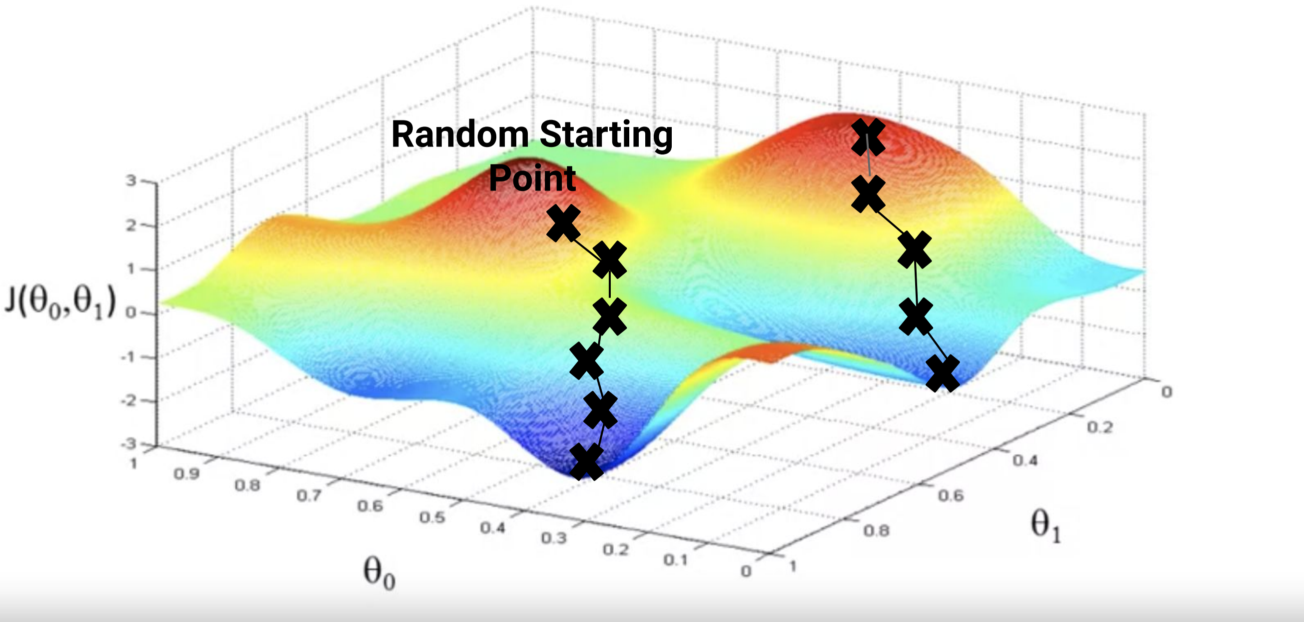
Gradient Descent

- Start with random θ_0 and θ_1
- Change θ_0 and θ_1 to reduce $J(\theta_0, \theta_1)$ until we find a minimum

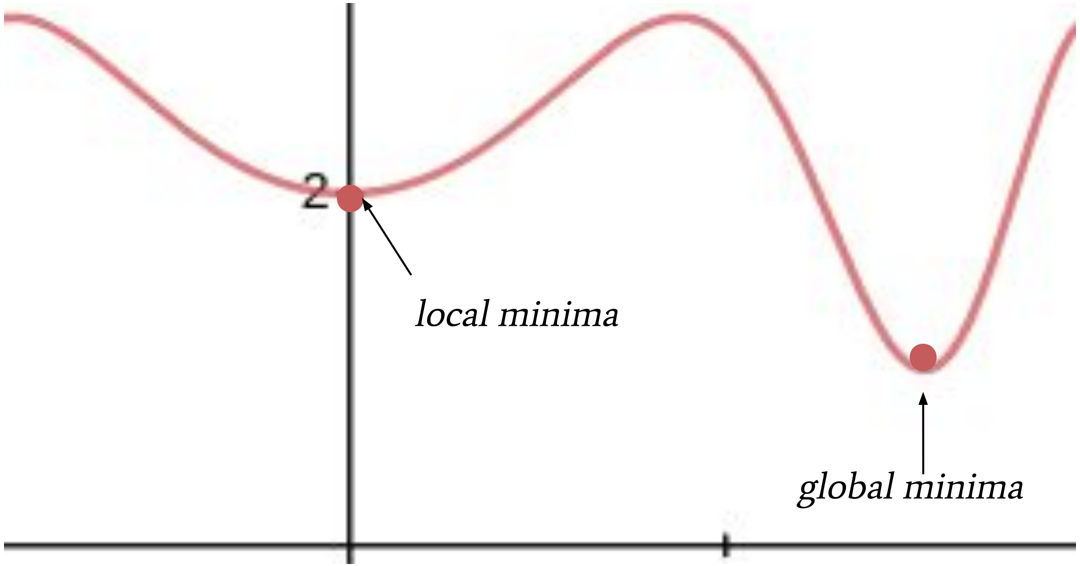


Gradient Descent

For different initializations, we might have different results



Local-Global minima

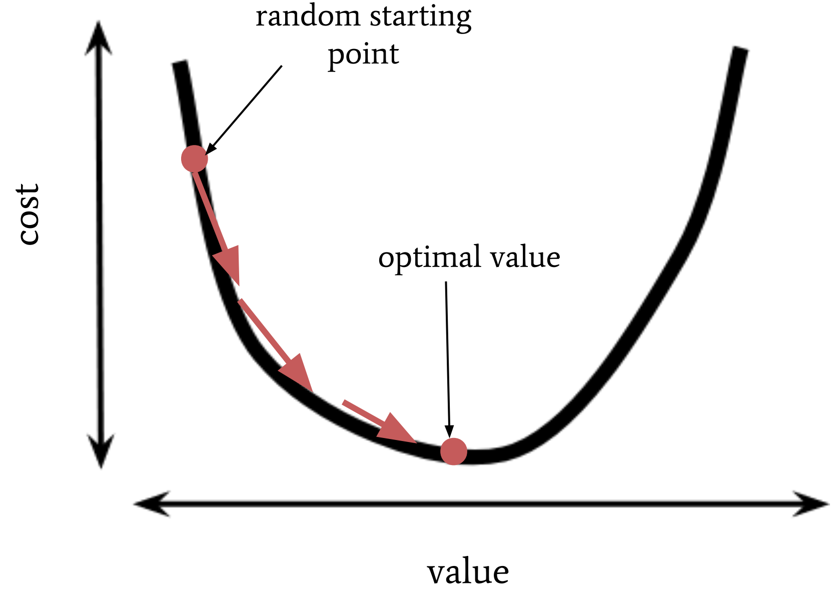


Gradient Descent

Repeat until convergence :

$$\underbrace{\boxed{\theta_j :=}}_{\text{assignment}}\overbrace{\boxed{\theta_j}}^{\text{current value}} - \underbrace{\boxed{\alpha}}_{\text{learning rate}}\underbrace{\boxed{\frac{\partial}{\partial\theta_j}J(\theta_0,\theta_1)}}_{\text{derivative}} \qquad (\text{for } j = 0 \text{ and } j = 1)$$

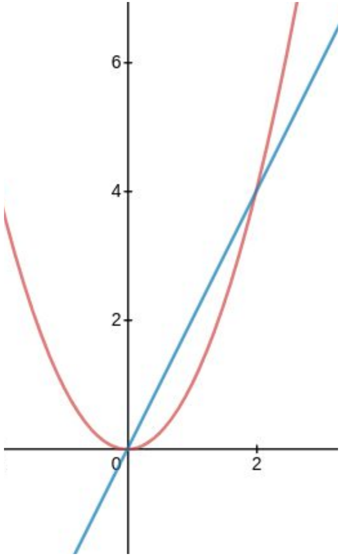
Gradient Descent



gradient / rate of change / slope / derivative.

exemple : $f'(x^2) = 2x$

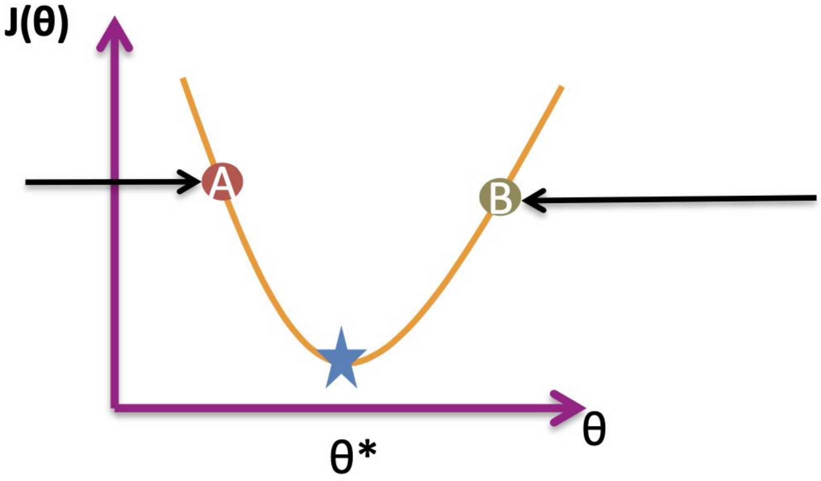
gradient for $x = 2$ is 4



Gradient Descent

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

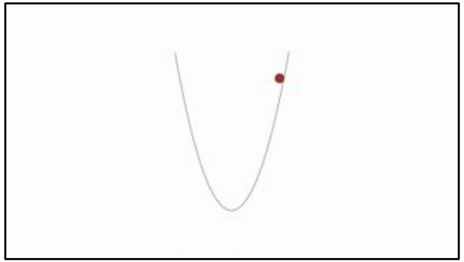
In this case, the derivative (gradient) $\partial J(\theta) / \partial\theta < 0$.
$\theta^A - \alpha^* \partial J(\theta) / \partial\theta > \theta^A$
θ^A is moving to the right
θ is increasing



In this case, the derivative (gradient) $\partial J(\theta) / \partial\theta > 0$.
$\theta^B - \alpha^* \partial J(\theta) / \partial\theta < \theta^B$
θ^B is moving to the left
θ is decreasing

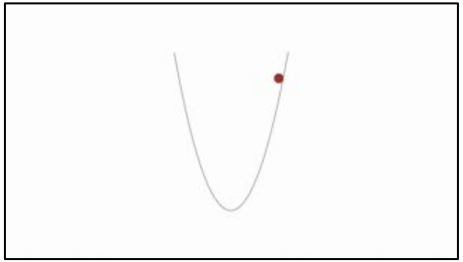
Learning Rate

GOOD LEARNING RATE



- Not too low
- Allows the network to find the right parameters

BAD LEARNING RATE



- Too high
- Learns fast but quickly overshoots and ends up increasing the error

Gradient Descent Recap

Gradient Descent

(n=1):

Repeat {

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial\theta_0}J(\theta)}$$

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

(simultaneously update θ_0, θ_1)

}

Multivariate Linear Regression

Notation – Multivariate Linear Regression

n=4

Size X_1	Number of Bedrooms X_2	Floor X_3	Age (years) X_4	Price (1000$) Y
20	0	3	30	300
37	1	$\boxed{0}$	45	540
88	3	4	60	986
...				...

m = 47

$$X^{(2)} = \begin{pmatrix} 37 \\ 1 \\ 0 \\ 45 \end{pmatrix}$$

$$X_3^{(2)} = 0$$

m: Number of examples

n: Number of features

$x^{(i)}$: Input (features) of the i-th training example

$x_j^{(i)}$: Value of feature j for the i-th training example

Multivariate Linear Regression

In simple linear regression, $h_\theta(x) = \theta_0 + \theta_1 x$

In multiple linear regression, $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

For convenience of notation, define $x_0 = 1$ $(x_0^{(i)} = 1)$

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

Multivariate Linear Regression

Rewrite in matrix notation

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

$$\begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix} \quad \theta, x \in R^{n+1} \quad \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix}$$

$$h_\theta(x) = \sum_{i=1}^{n} \theta_i x_i \quad = \theta^T x$$

For correct multiplication, we need Θ^T

$\Theta^T = [\Theta_0 \quad \Theta_1 \quad \dots \quad \Theta n]$

$$X = \begin{pmatrix} X_0 \\ X_1 \\ \dots \\ Xn \end{pmatrix} \begin{matrix} \Rightarrow 1 \\ \\ \\ \\ \end{matrix}$$

Multivariate Linear Regression

Cost function:

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

Gradient Descent

Simple Linear Regression

Repeat $\{$

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

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

(simultaneously update θ_0, θ_1) $\}$

Multivariate Linear Regression

Repeat $\{$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

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

$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_1^{(i)}$$

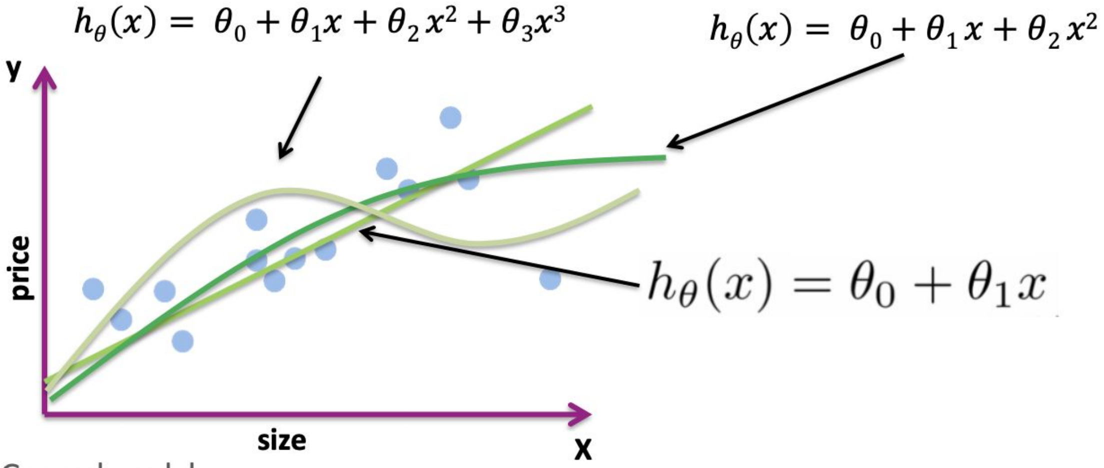
$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_2^{(i)}$$

$\ldots$

Polynomial Regression

Polynomial Regression

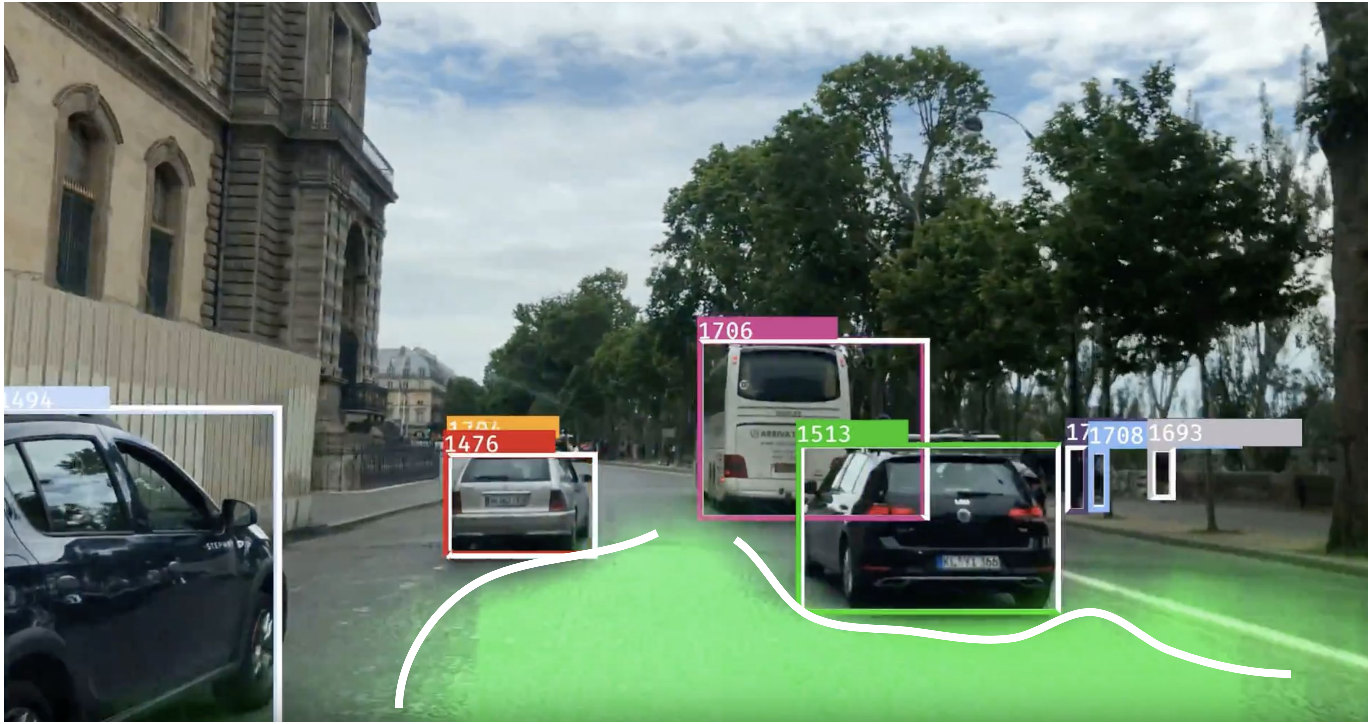
Polynomial regression is a particular case of multiple regression where the features are powers of one single feature x



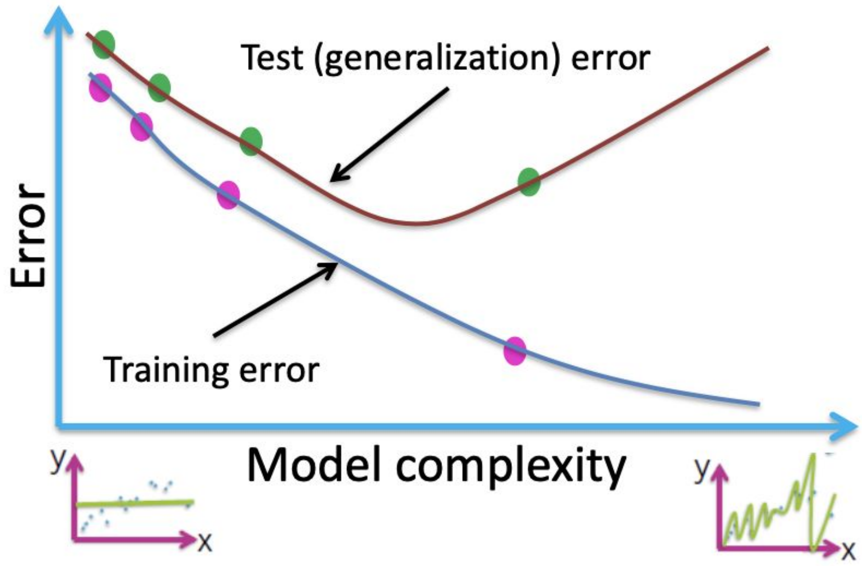
General model:

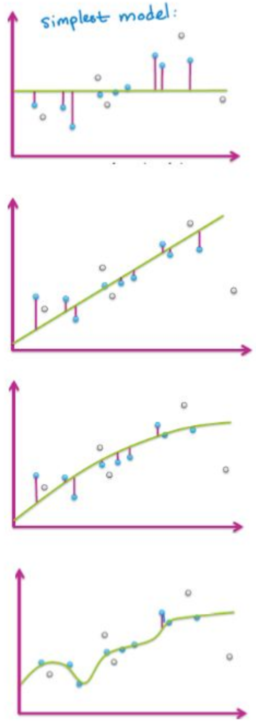
$$y = \theta_0 + \theta_1 x + \theta_2 x^2 \theta_1 x + \cdots + \theta_p x^p + \varepsilon$$

Polynomial Regression in Practice

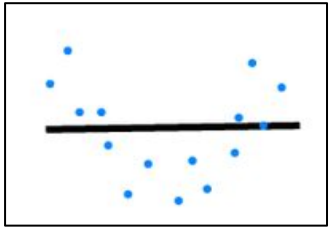


Performance

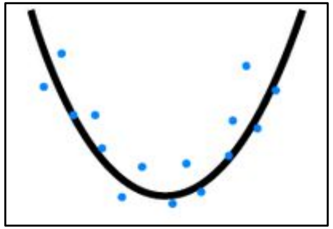




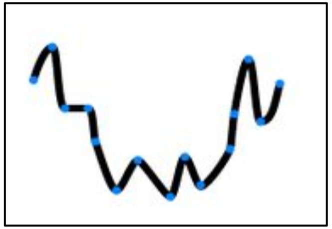
Performance



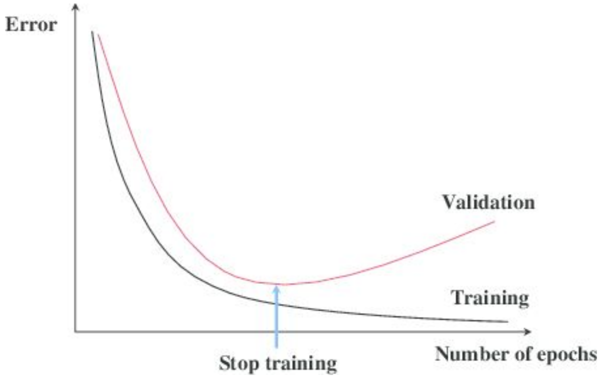
UNDERFITTING



JUST RIGHT



OVERFITTING



Thank

You

jeremycohen.podia.com

https://www.linkedin.com/in/jeremycohen2626/