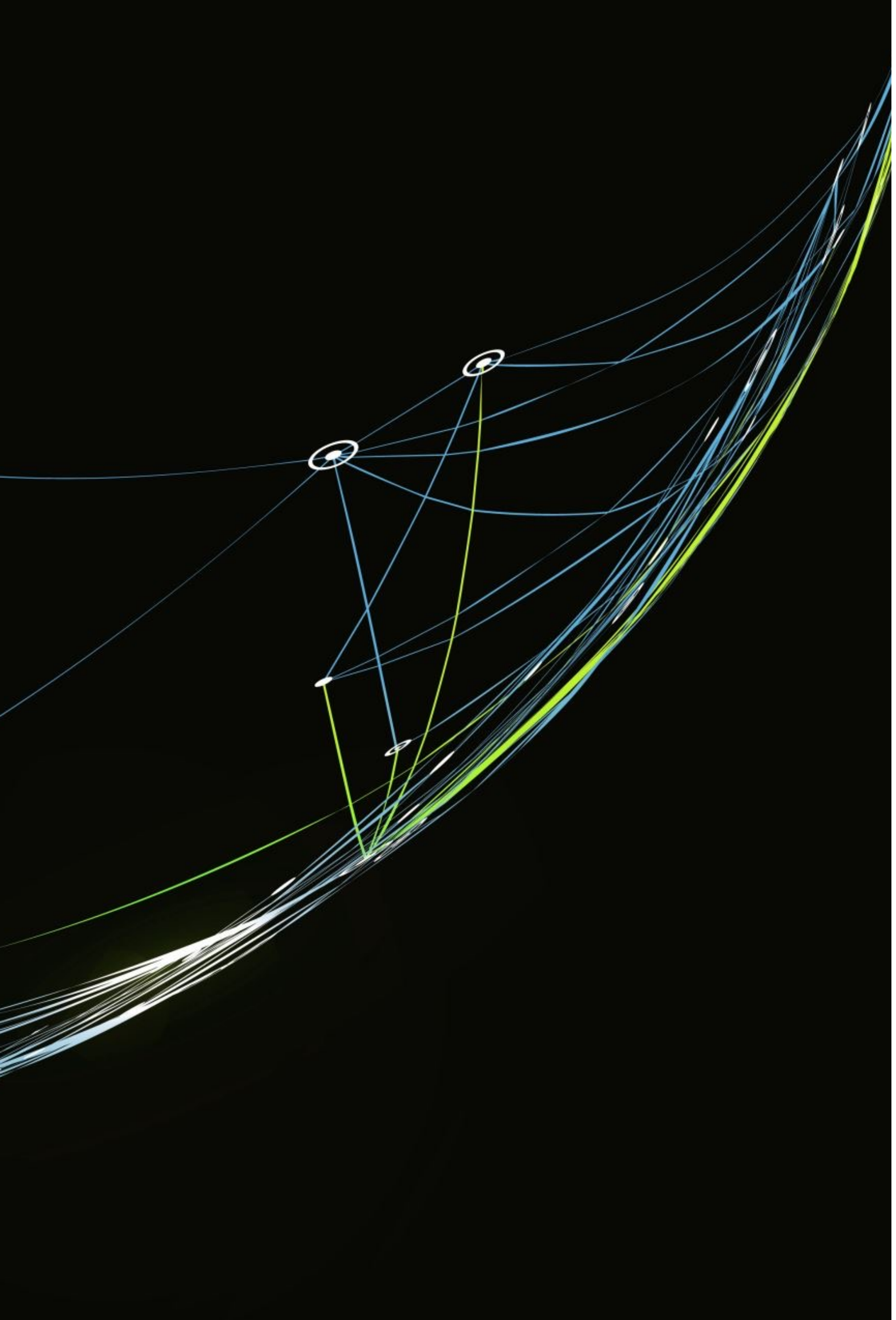


Modelos de Regresión

REGRESIÓN LINEAL

TC3006C

Regression

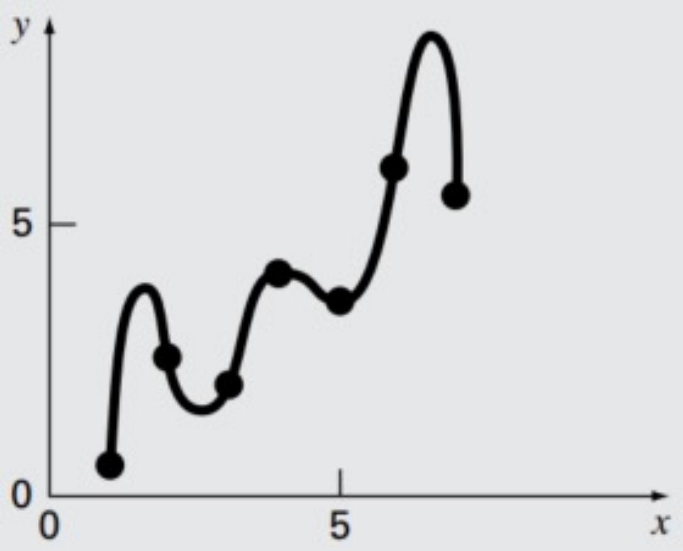
Allows data analysis

Able to predict an estimate of a data point that was not analyzed

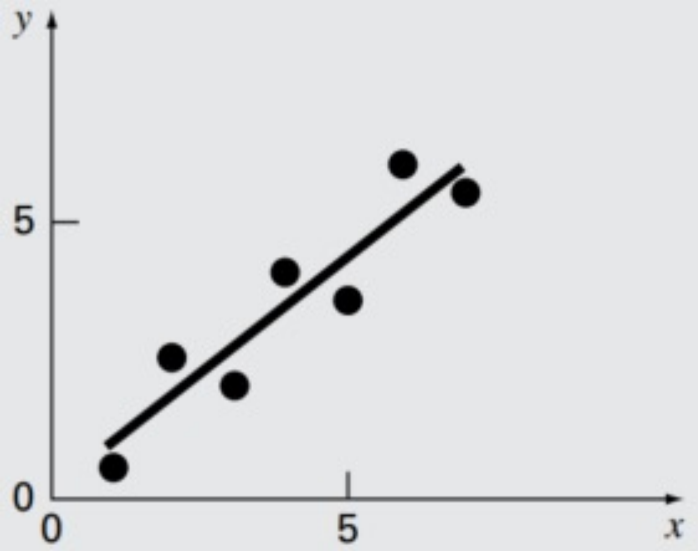
Generates a model that describes data behavior

Can be used when data has errors

Not to be confused with interpolation



Interpolation



Regression

Linear regression

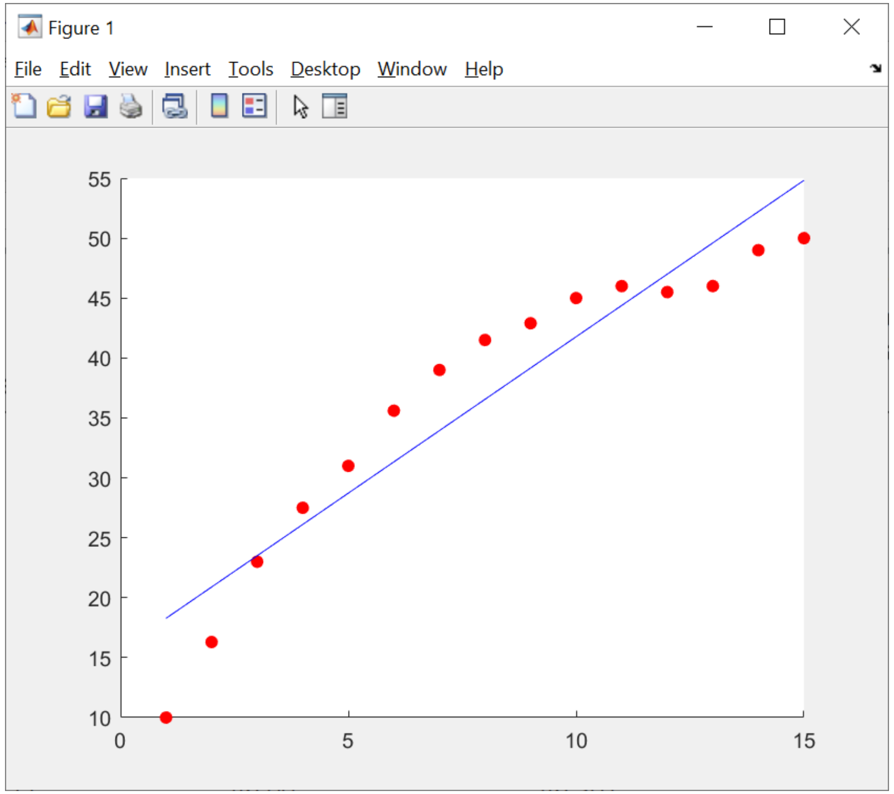
Fits a line to a set of observed data points

Tries to find out the best linear relationship between the input and output.

Linear equation

$$y = mx + b$$

This equation works for any number of input variables



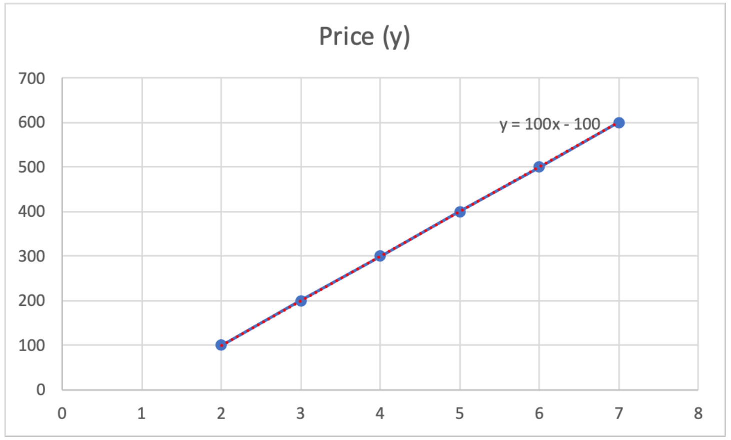
Linear regression…

CPUs (x)	Price (y)
2	100
3	200
4	300
5	400
6	500
7	600

Try to find the function (***y = mx+b***) that describes the relation between cpu and price

Hypothesis function

- Estimated function describing data
- “Guessed” function



What is the price of 8 CPUs?

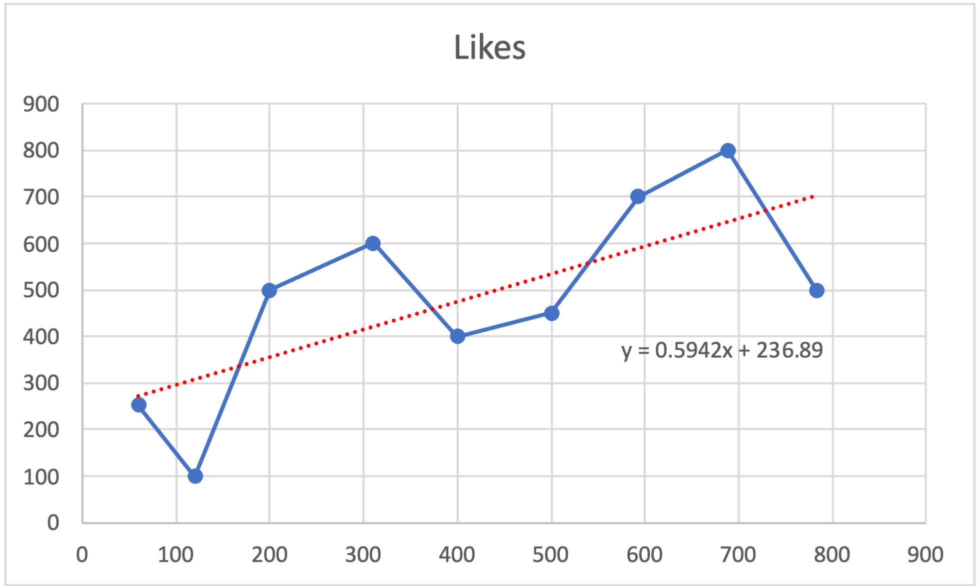
Linear regression...

CPUs (x)	Price (y)
2	80
3	200
4	250
5	400
6	700
7	800
8	???

How about with this more realistic data?

Linear regression…

Friends	Likes
60	253
120	100
200	500
310	600
400	400
500	450
593	700
688	800
783	500



How to automatically fit a line?

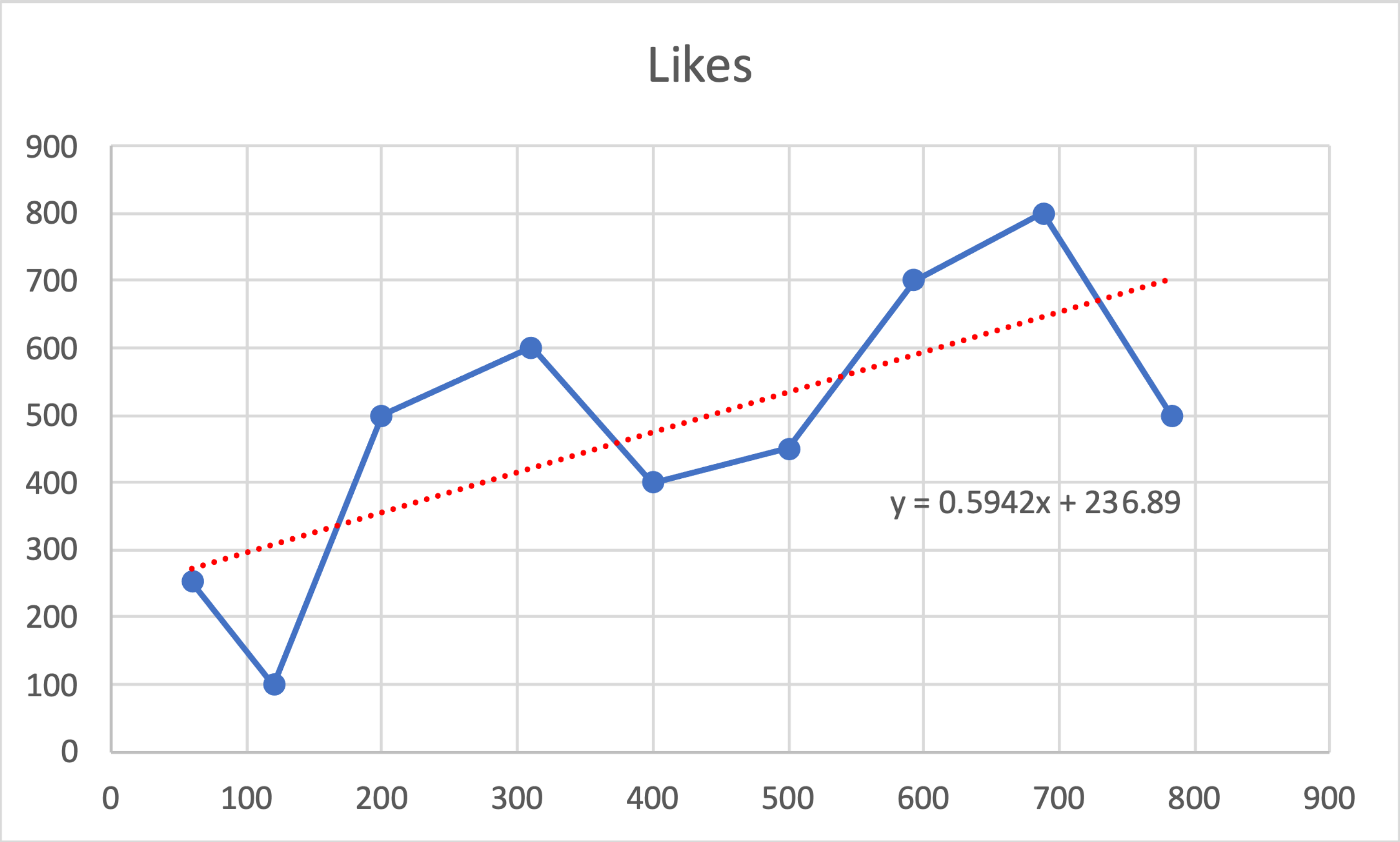
Cost/loss function

Cost/loss function

Method for evaluating how well your algorithm models your dataset

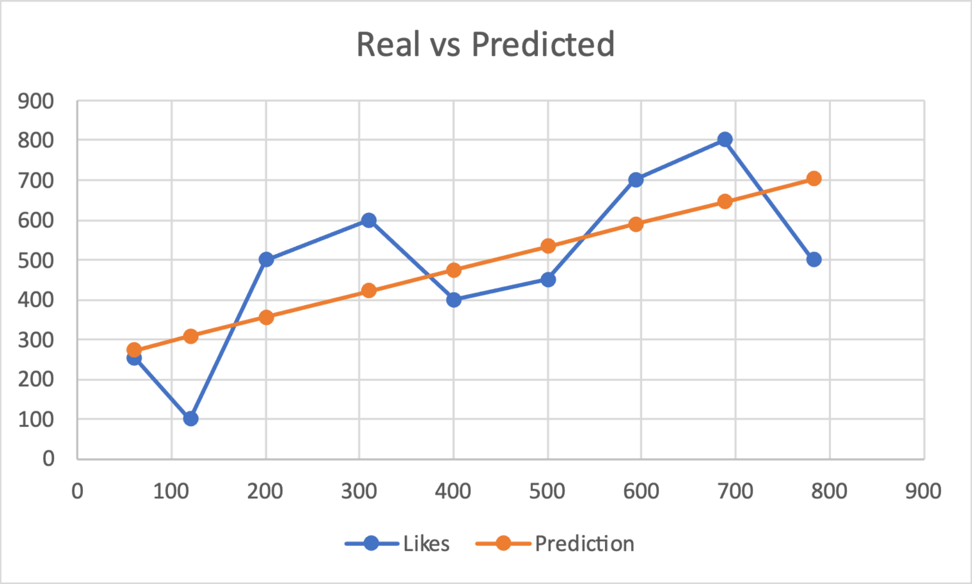
- Good predictions will result in a low value (near 0)
- Bad predictions will result in a high value

How do we get the error?



Cost/loss...

Friends	Likes	Prediction	Error
60	253	272.542	-19.542
120	100	308.194	-208.194
200	500	355.73	144.27
310	600	421.092	178.908
400	400	474.57	-74.57
500	450	533.99	-83.99
593	700	589.2506	110.7494
688	800	645.6996	154.3004
783	500	702.1486	-202.1486



Negative and positive error?

Cost/loss…

Friends	**Likes**	**Prediction**	**Error**	**Absolute Error**	**Squared Error**
60.00	253.00	272.54	- 19.54	19.54	381.89
120.00	100.00	308.19	- 208.19	208.19	43,344.74
200.00	500.00	355.73	144.27	144.27	20,813.83
310.00	600.00	421.09	178.91	178.91	32,008.07
400.00	400.00	474.57	- 74.57	74.57	5,560.68
500.00	450.00	533.99	- 83.99	83.99	7,054.32
593.00	700.00	589.25	110.75	110.75	12,265.43
688.00	800.00	645.70	154.30	154.30	23,808.61
783.00	500.00	702.15	- 202.15	202.15	40,864.06
		Sum	-0.22	1,176.67	186,101.64
				MSE	**20,677.96**

Mean Squared Error / L2 Loss

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

1. Compare predicted value with real value
2. Compensate negative sign
3. Sum error of every data point
4. Get average error

Some cost/loss functions

Mean Absolute Error (MAE) / L1 Loss

$$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right|$$

Root Mean Squared Error (RMSE)

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Mean Percent Absolute Error (MAPE)

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right|$$

Sum of Squared Errors (SSE)

$$SSE = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

D	E	F
SP	DAX	FTS
0.00468	0.002193	0.003
.007787	0.008455	0.012
0.03047	-0.01783	-0.02
.003391	-0.01173	-0.00
0.02153	-0.01987	-0.01
0.02282	-0.01353	-0.00
.001757	-0.01767	-0.00
0.03403	-0.04738	-0.05
.001328	-0.01955	-0.01

Class exercise

Write down the initial hypothesis with random parameters for a linear regression model for the following dataset

- You wish to predict **USD Based ISE**

- https://archive.ics.uci.edu/ml/datasets/ISTANBUL+STOCK+EXCHANGE#

Optimization (Learning)

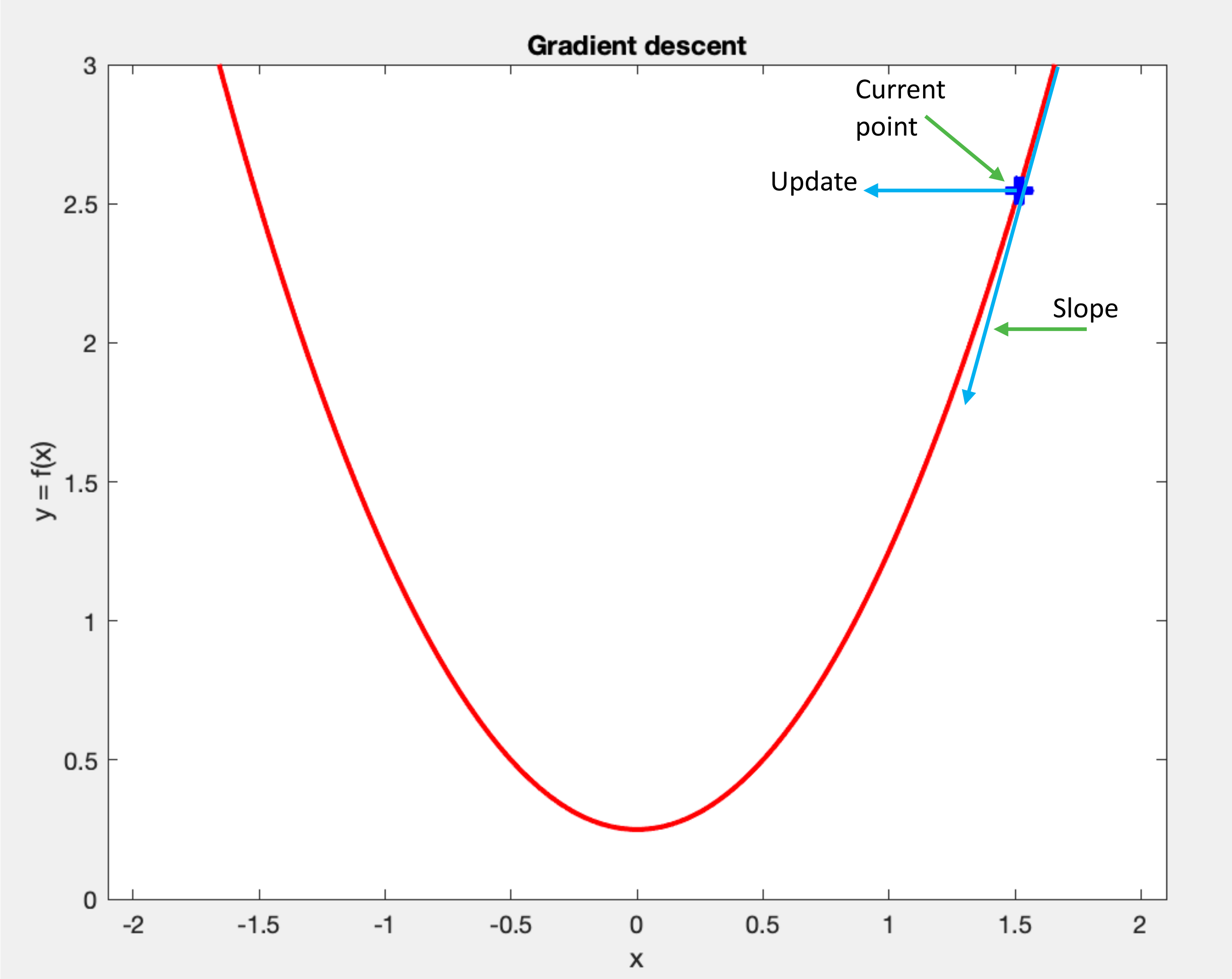
We need:

1. A model to predict new values

2. A cost/loss function

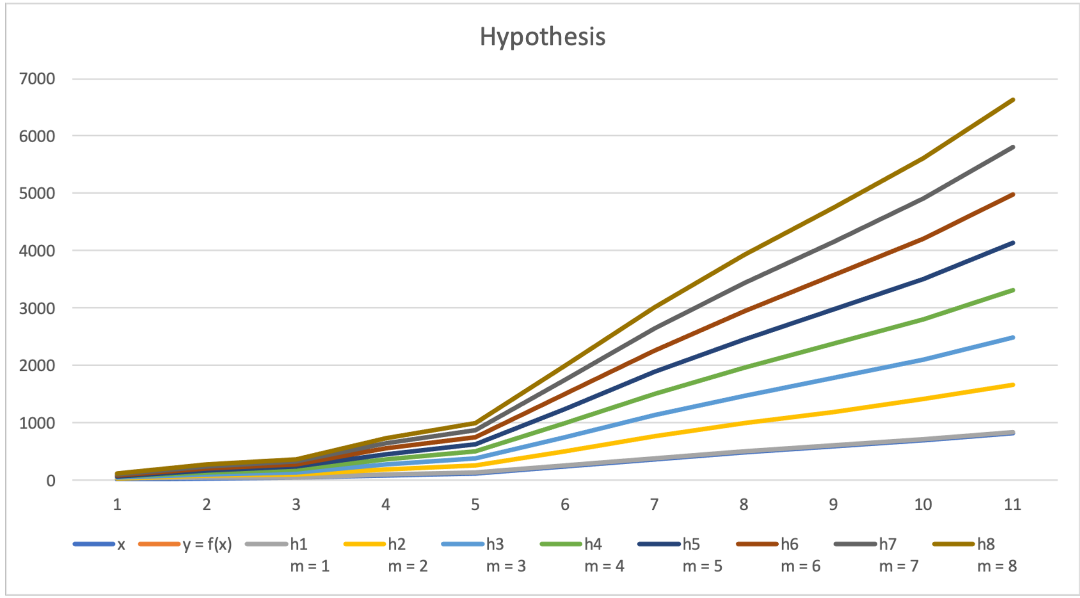
Machine learning uses iterative algorithms (trial and error) to find a solution to an optimization problem

Some problems may be solved mathematically, but this solution is usually very expensive in terms of memory or time

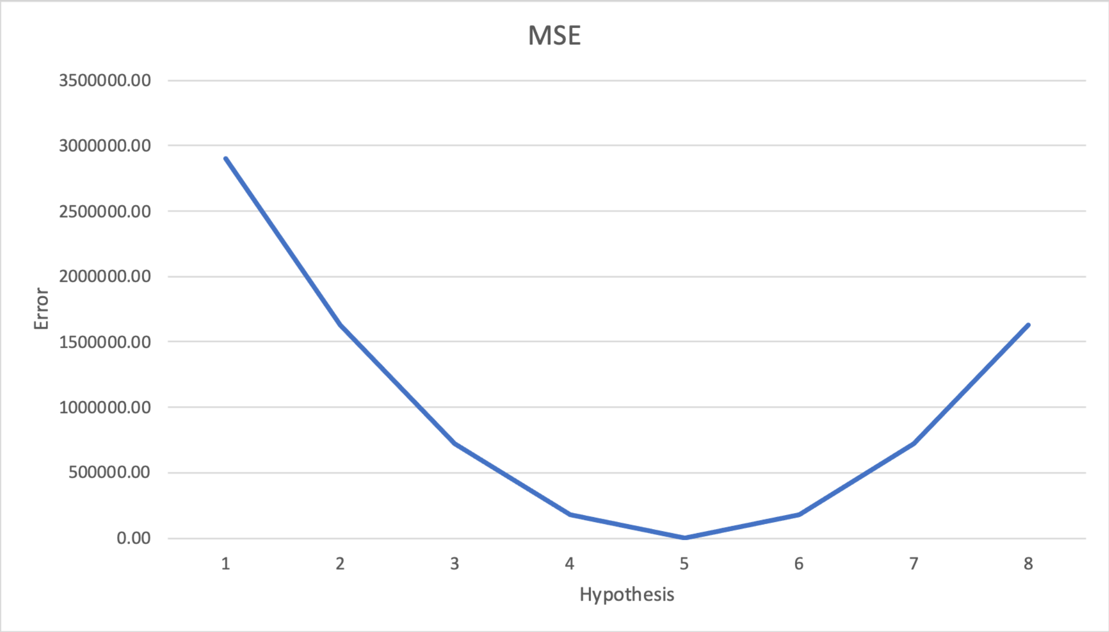


Gradient Descent

		y = x +13	y = 2x +13	y = 3x +13	y = 4x +13	y = 5x +13	y = 6x +13	y = 7x +13	y = 8x +13
x	y = f(x)	h1 m = 1	h2 m = 2	h3 m = 3	h4 m = 4	h5 m = 5	h6 m = 6	h7 m = 7	h8 m = 8
13	78	26	39	52	65	78	91	104	117
33	178	46	79	112	145	178	211	244	277
45	238	58	103	148	193	238	283	328	373
90	463	103	193	283	373	463	553	643	733
124	633	137	261	385	509	633	757	881	1005
248	1253	261	509	757	1005	1253	1501	1749	1997
376	1893	389	765	1141	1517	1893	2269	2645	3021
489	2458	502	991	1480	1969	2458	2947	3436	3925
593	2978	606	1199	1792	2385	2978	3571	4164	4757
700	3513	713	1413	2113	2813	3513	4213	4913	5613
827	4148	840	1667	2494	3321	4148	4975	5802	6629



Hypothesis



	e1	e2	e3	e4	e5	e6	e7	e8
	2704	1521	676	169	0	169	676	1521
	17424	9801	4356	1089	0	1089	4356	9801
	32400	18225	8100	2025	0	2025	8100	18225
	129600	72900	32400	8100	0	8100	32400	72900
	246016	138384	61504	15376	0	15376	61504	138384
	984064	553536	246016	61504	0	61504	246016	553536
	2262016	1272384	565504	141376	0	141376	565504	1272384
	3825936	2152089	956484	239121	0	239121	956484	2152089
	5626384	3164841	1406596	351649	0	351649	1406596	3164841
	7840000	4410000	1960000	490000	0	490000	1960000	4410000
	10942864	6155361	2735716	683929	0	683929	2735716	6155361
MSE	**2900855.27**	**1631731.09**	**725213.82**	**181303.45**	**0.00**	**181303.45**	**725213.82**	**1631731.09**

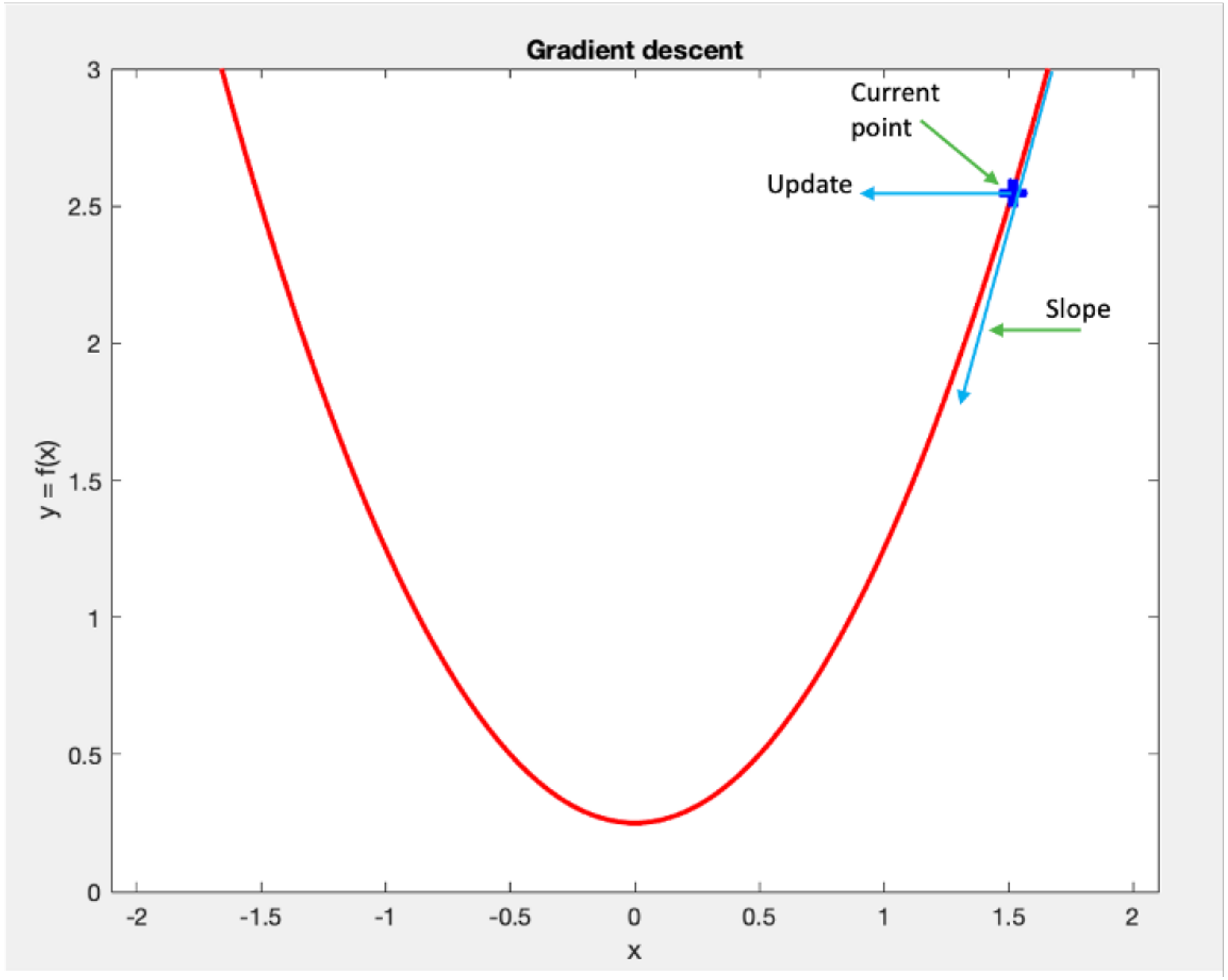
Mean Squared Errors

Gradient descent

Way to minimize an objective function

The function is parameterized by a model's parameters

The parameters are updated in the opposite direction of the gradient of the objective function



Gradient descent

Cost Function

$$J(\Theta_0, \Theta_1) = \frac{1}{2m}\sum_{i=1}^{m}[\underbrace{h_\Theta(x_i)}_{\text{Predicted Value}} - \underbrace{y_i}_{\text{True Value}}]^2$$

Gradient Descent

$$\boxed{\Theta_j = \Theta_j - \underbrace{\alpha}_{\text{Learning Rate}}\frac{\partial}{\partial\Theta_j}J(\Theta_0, \Theta_1)}$$

Now,

$$\frac{\partial}{\partial\Theta}J_\Theta = \frac{\partial}{\partial\Theta}\frac{1}{2m}\sum_{i=1}^{m}[h_\Theta(x_i) - y]^2$$

$$= \frac{1}{m}\sum_{i=1}^{m}(h_\Theta(x_i) - y)\frac{\partial}{\partial\Theta_j}(\Theta x_i - y)$$

$$= \frac{1}{m}(h_\Theta(x_i) - y)x_i$$

Therefore,

$$\boxed{\Theta_j := \Theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}[(h_\Theta(x_i) - y)x_i]}$$

https://www.geeksforgeeks.org/gradient-descent-in-linear-regression/

Procedure:

1. Calculate slope at current point
   - For one parameter we use a derivative
   - For multiple parameters we use the gradient
2. Move in the direction of negative gradient with a step size α (learning rate)
3. Update parameter
4. Repeat until converged

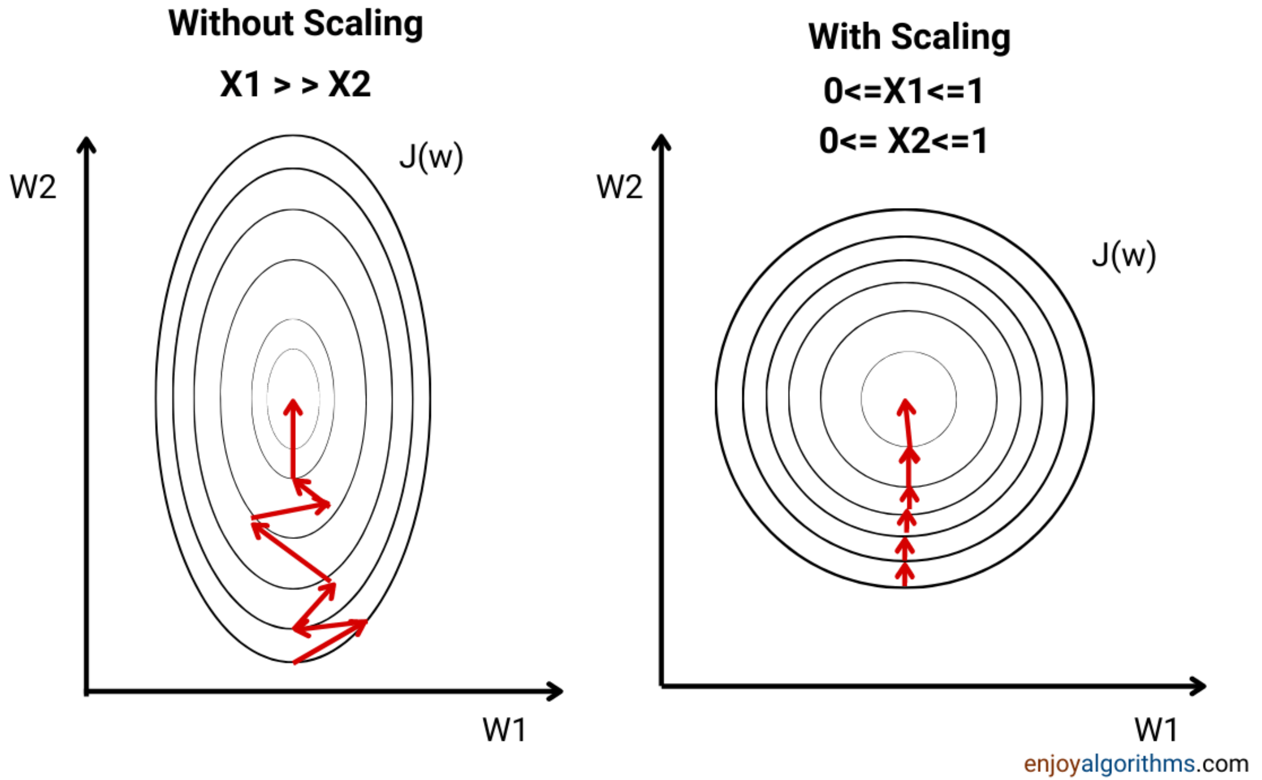
Feature Scaling

ML models expect ranges of features to be on the same scale to decide their importance without any bias

- Number of bedrooms (0 - 10)
- Price (0 - 1,000,000)

Scaling helps:

- Gradient descent flow smoothly
- Eliminates magnitude bias
- Useful with distance-based and gradient descent based algorithms



Feature Scaling…

NORMALIZATION

Scaling all features into the same range

Min-Max normalization

$$x_i' = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

range is [0,1]

STANDARDIZATION

Scaling technique where the values are centered around a mean with a unit standard deviation

Mean (μ) = 0 and Standard deviation (σ) = 1 (Gaussian distribution)

$$x_i' = \frac{x_i - \mu}{\sigma}$$

Class exercise

Make two runs of gradient descent for linear regression, using the following data set to fit the model

Pre(x)	Post(y)
65	74
45	70
79	100
24	67

Code in Python

- linear_reg_gd.py

Main types of gradient descent

Batch gradient descent
- Uses entire dataset per update
- Accurate
- Usually very slow
- Not useful if dataset does not fit in RAM
- Cannot be used online

Mini-batch gradient descent
- Uses a small batch of training examples
- Sizes usually between 50 and 256
- Fast
- Converges gradually
- Can be used online

Stochastic Gradient Descent (SGD)
- Performs a parameter update per training example
- Converges but can move randomly for some time
- Much faster than normal Gradient Descent
- Can be used online

Adam (Adaptive Moment Estimation)
- Keeps exponentially decaying average of past squared gradients
- Keeps an exponentially decaying average of past gradients, similar to momentum
- Like a heavy ball with friction that prefers flat minima in the error surface