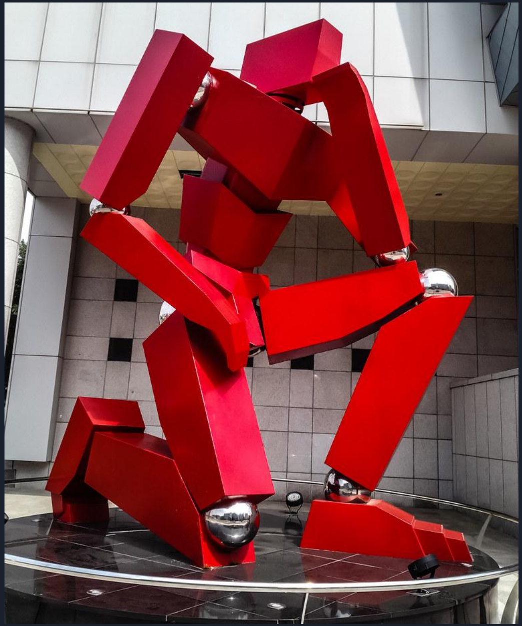
Machine Learning (ML) fundamentals and types of ML algorithms

Ing. Elec. Joaquín Gomez Prats
apolos@gmail.com

Outline:

- Introduction: AI and ML, terms, definitions, classic programming vs ML, Relationships
- ML Algorithms: commonly used Terms. Types of machine learning algorithms, criteria categorisation
- Supervised Learning: key points, examples, uses, common alg.
  - ML process Case: Regression: Fundamentals, Cost Function, Gradient Descent. Steps. Intuition
- Unsupervised Learning: key points, examples, uses, common alg.
  - K-means clustering, description, process
- Fundamentals of Neural Networks: architecture & terminology, description, uses, examples
- Reinforcement Learning: key points, examples, uses, common alg.
  - Case: Q-Learning: Description, process, explore vs exploit.
  - Introduction to Deep Q-learning

Introduction



"Thinking Robot" by purunuri is licensed under CC BY-NC-SA 2.0

Artificial Intelligence

Machine Learning

Neural Networks

Deep Learning

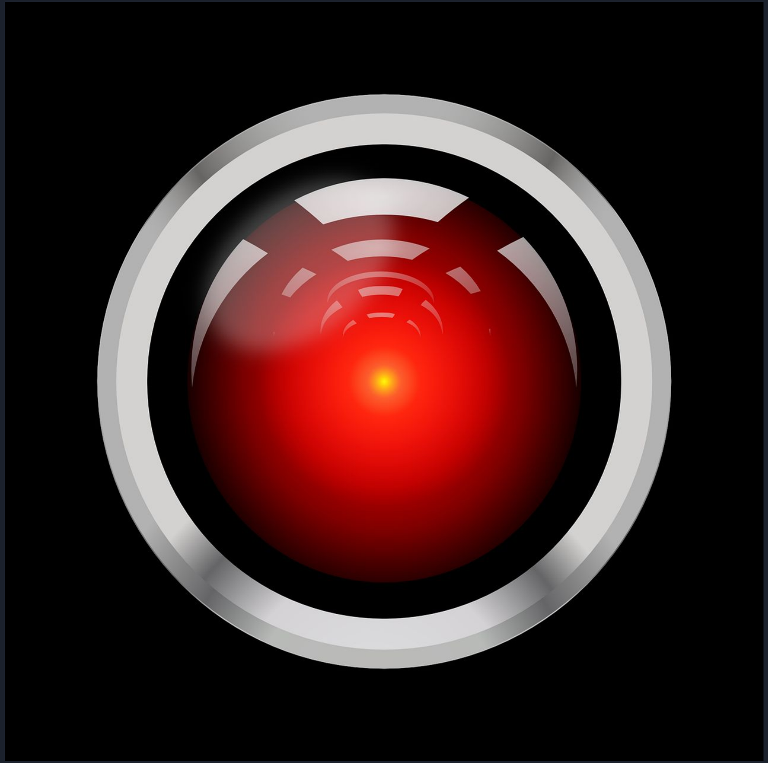
Alpha Zero

OpenAI

Deepmind

Reinforcement Learning

What is AI?



https://commons.wikimedia.org/wiki/File:HAL9000.svg

Artificial Intelligence:

Any code, technique or algorithm that enables machines to mimic, develop or demonstrate human cognition or behaviour such as "learning" and "problem solving"

Machine Learning

Definitions:

“The field of study that gives computers the ability to learn without being explicitly programmed” Arthur Samuel, 1959

“A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E”. Tom M. Mitchell, 1998

Machine Learning

Example: playing checkers.

"This Game is ON" by cogdogblog is licensed under CC BY 2.0

T: task = to play and win checkers

E: experience = the playing of many games

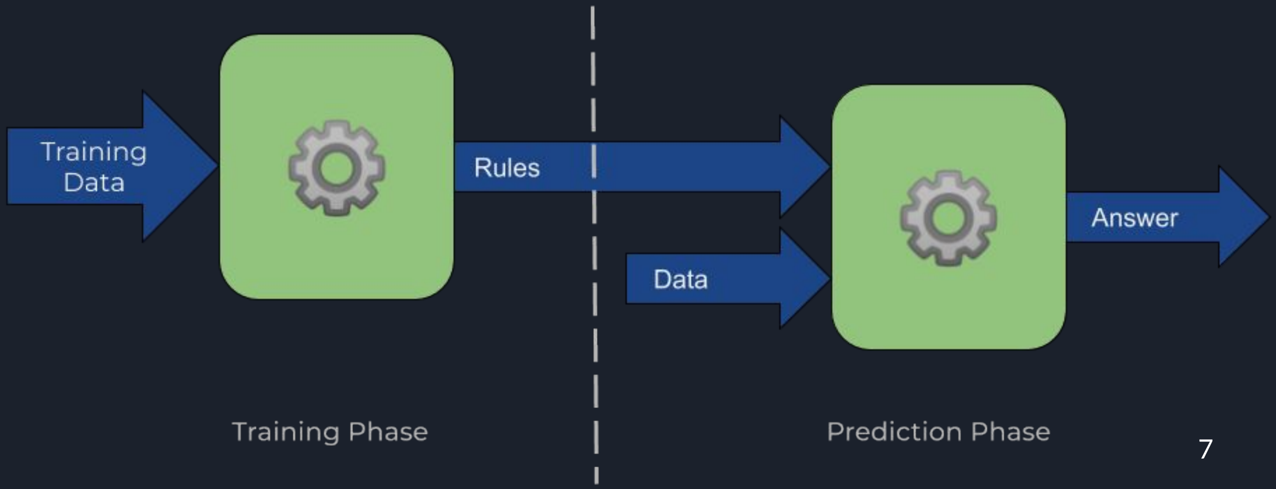
P: Performance measure = statistics of the game results (win, loose, draw)

Machine Learning

Classic Programming

vs

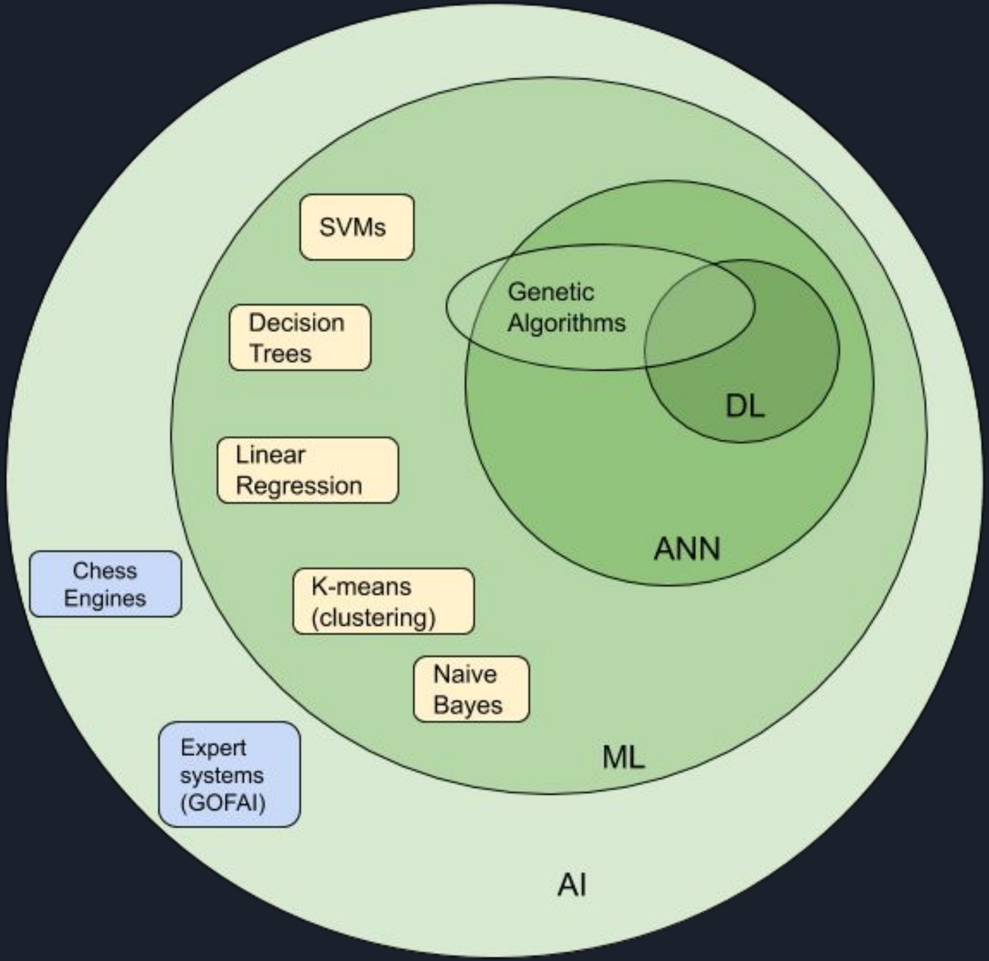
Machine Learning



Machine Learning

source: https://xkcd.com/1838/

Relationships



Commonly ML Terminology

Features: Attributes describing the data. Usually used as inputs. (common feature notation: x’s)

Labelled Data: already classified and identified data (“labeled”). Usually used for training and testing the model. (common label notation: y’s)

Training Set: Data provided to the system to learn. A dataset with features (and if applicable, labeled data).

Instance: One sample in the training dataset. Other names for “instance” are: (data) point, observation (An instance consists of the feature values x’s and, if known, the target outcome y)

Types of Machine Learning Algorithms

Categorization according learning methodology:

ML Algorithms

Supervised Learning	Unsupervised Learning	Reinforcement Learning
“The outcome or output for the given input is known”. The data set is labelled. The “right answers” are given	*“Output is not known. Data is not labelled. It tries to find structure on the data by itself. There is no feedback based on the prediction results*	*The learning algorithm interacts with an environment where it gets trained via a trial and error method The feedback is a reward/punishment signal*

Types of Machine Learning Algorithms

Categorization according output's values or purpose:

(Supervised Learning)

ML Algorithms

- Classification
- Regression

Classification

To predict the belonging to a certain class/category for a given element.

Ex: Spam / Not-Spam, Dog Breed, object classification

Regression

To predict/estimate continuous values

Ex: Price, temperature, physics magnitudes

Supervised Learning

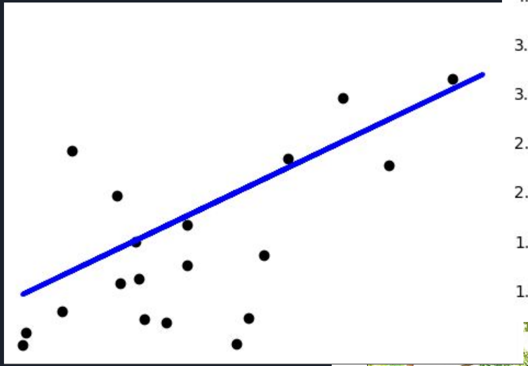
Key Points:

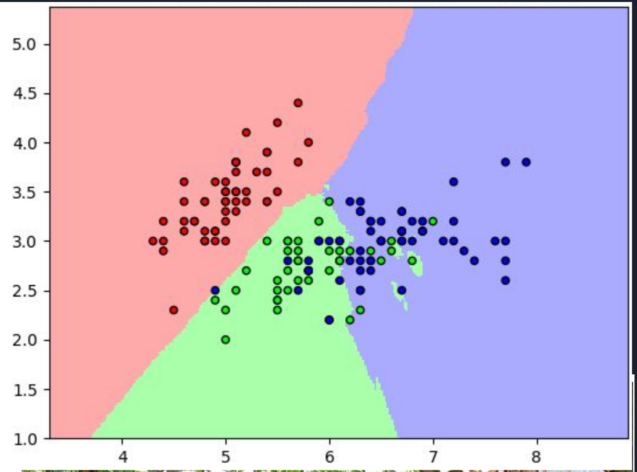
- Labelled data is used for training
- Mainly applied to regression and classification problems

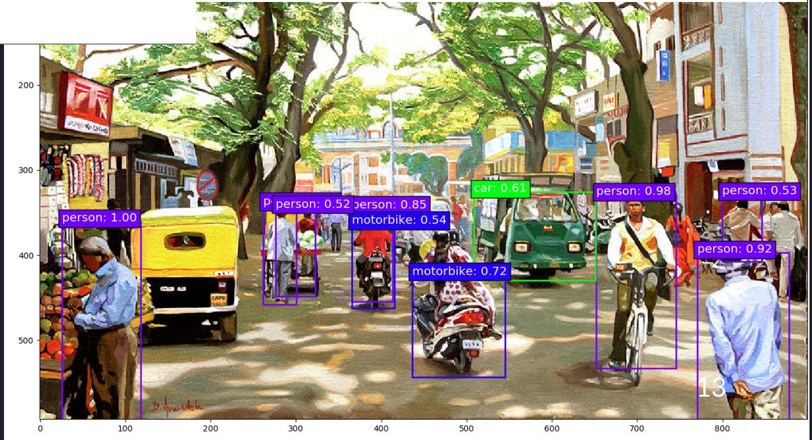
Examples: predict prices (regression), spam/not-spam, (binary classification), classification/object detection on photo (multiclass classification)

Common Algorithms:

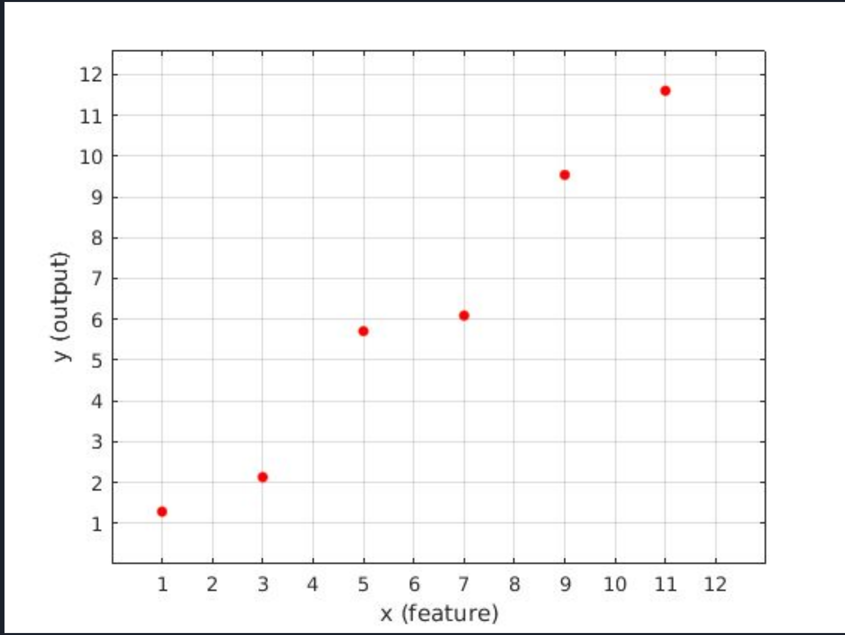
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVM)
- K-Nearest Neighbor
- Decision Trees
- Naive Bayes
- SSD- Single Shot Detector (Neural Networks)







Linear Regression



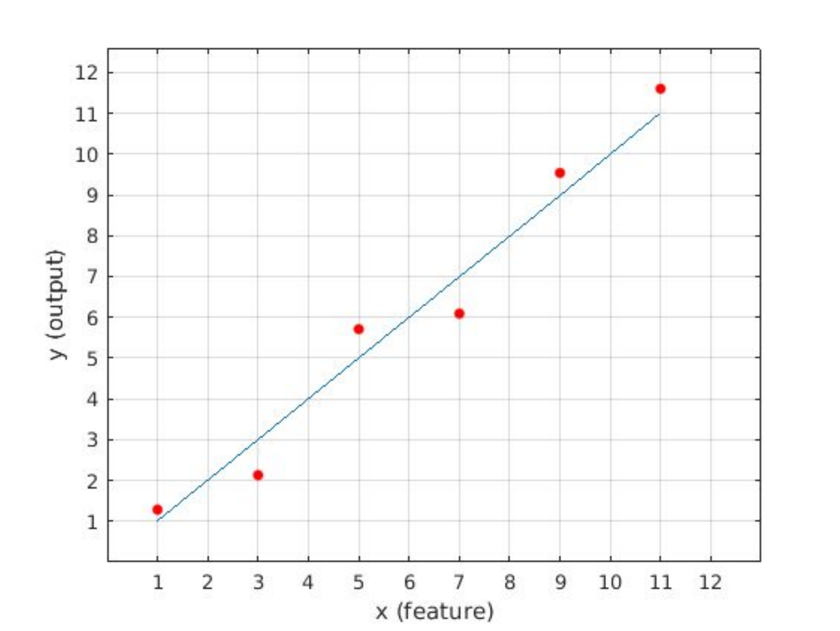
Data Set / Training Set

x	y
1	1.28
3	2.13
5	5.7
7	6.09
9	9.54
11	11.6

m : Number of training examples. E.g. m = 6
$\mathbf{x_i}$'s : "input" variables / features. E.g. $x_i = x_1 = x$
$\mathbf{y_i}$'s : "output" variables / "target" variable / label.
E.g. $y_i = y_1 = y$

(x , y): one single training example
$(x^{(i)}, y^{(i)})$: i^{th} element / training example. i = 1... m
E.g. $(x^{(4)}, y^{(4)}) = (7, 6.09)$

Linear Regression



Our goal is, given a training set, to learn a function $h : X \rightarrow Y$ so that $h(x)$ is an "optimal" predictor for the corresponding value of y.

Where: X: space of input values

Y: space of output values

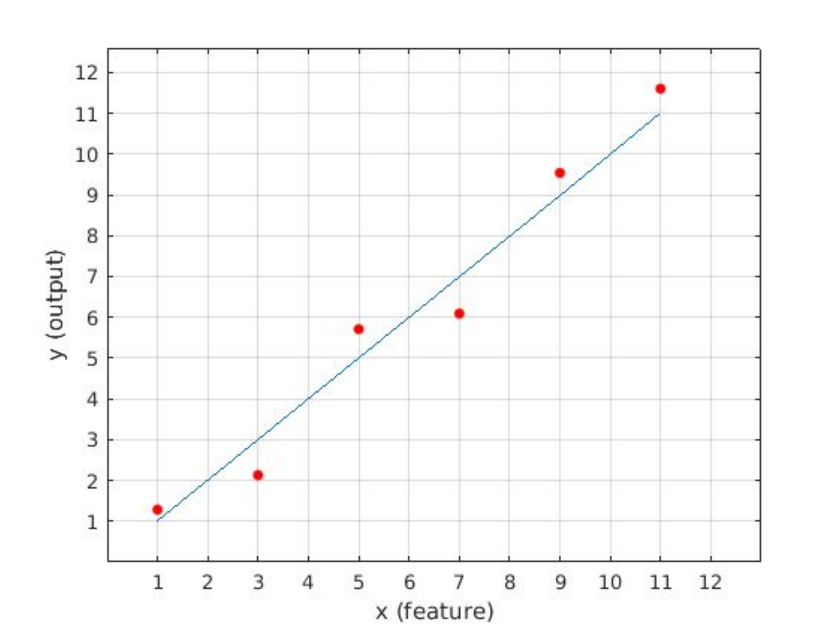
$X = Y = \mathbb{R}$

Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 \cdot x$

θi: Parameters of the model

The idea is to choose the best values of parameters θ_0 , θ_1 (θi's) so that the predicted values of $h_\theta(x)$ are the best possible fit for our training samples $(x^{(i)}, y^{(i)})$. (E.g.. $h_\theta(x)$ is the closest to $y^{(i)}$ for $i = 1...6$)

Linear Regression



Cost Function:

(Loss Function / Error)

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

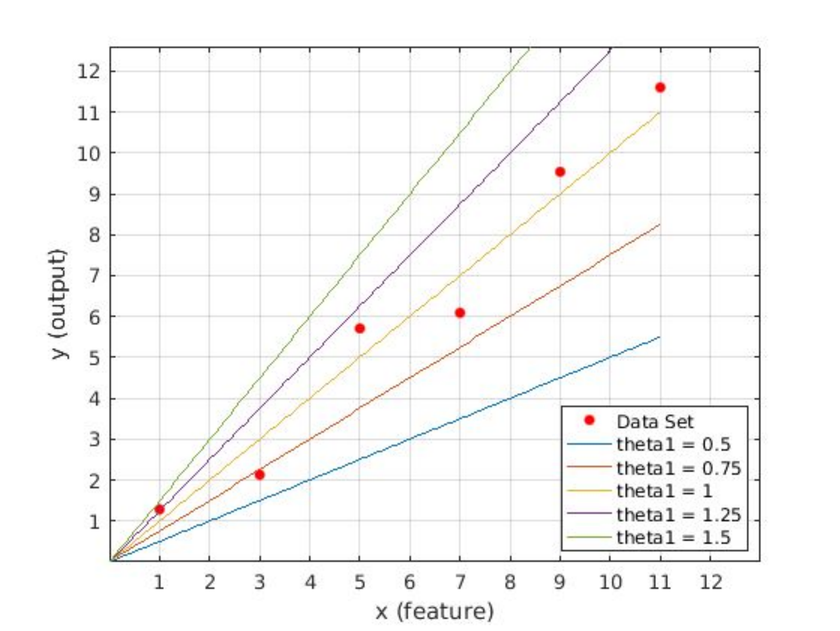
Goal: Minimize $J(\theta_0 , \theta_1)$

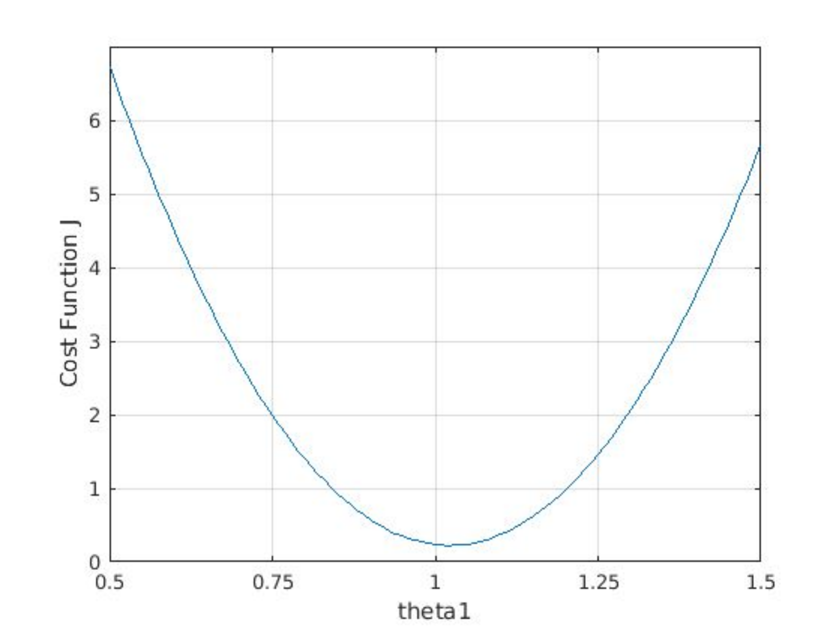
Simplified hypothesis function:

With $\theta_0 = 0$

$h_\theta(x) = \theta_1 \cdot x$ (straight line passing through the origin) => $J = J(\theta_1)$

Linear Regression





Simplified hypothesis function $h_\theta(x) = \theta_1 \cdot x$

$\theta_1 = 0.5, 0.75, 1, 1.25, 1.5$

Cost Function $J(\theta_1)$

Gradient Descent

Goal:

Given a function $J(\theta_0, ..., \theta_j, ..., \theta_k)$

Minimize J for $\theta_0, ..., \theta_j, ..., \theta_k$

Action Outline:

- Start with some values of $\theta_0, ..., \theta_j, ..., \theta_k$
- Keep changing $\theta_0, ..., \theta_j, ..., \theta_k$ to reduce J until we end up at a minimum

Formally:

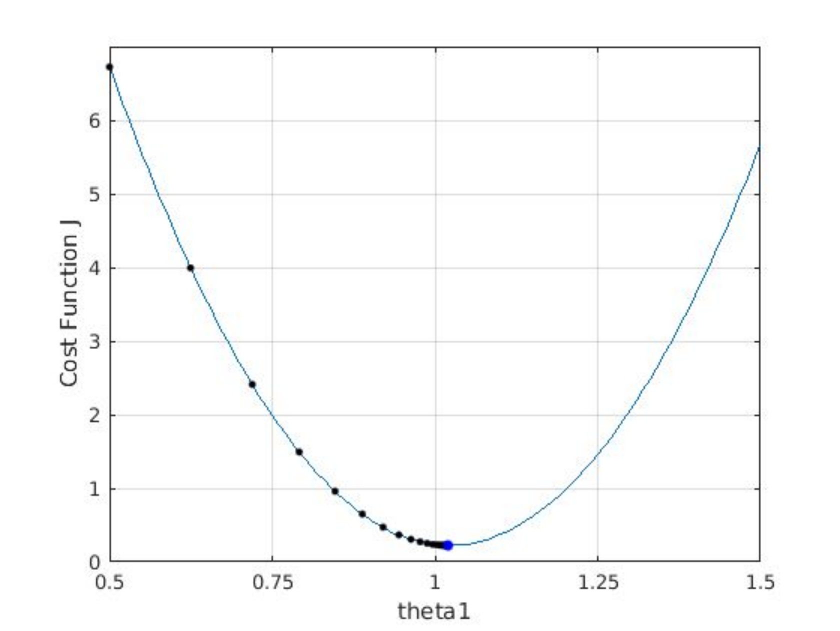
Given $J(\theta_0, ..., \theta_j, ..., \theta_k)$

Repeat until converge:

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

Where: α: learning rate

Gradient Descent



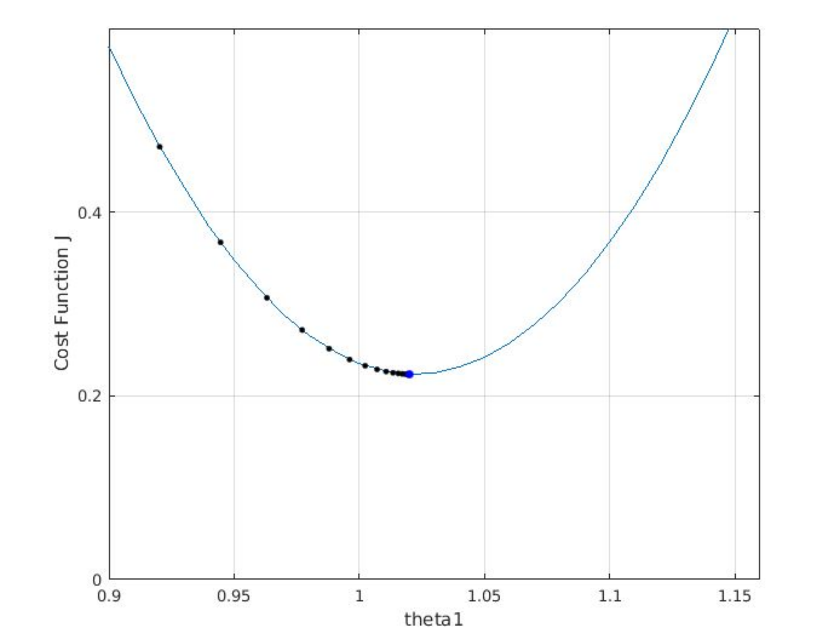
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

Example:

Initial $\theta_1 = 0.5$
Learning rate $\alpha = 0.005$
Iterations = 20

Optimal values found:
$\theta_1 = 1.02$
$J(\theta_1) = 0.22$

Gradient Descent



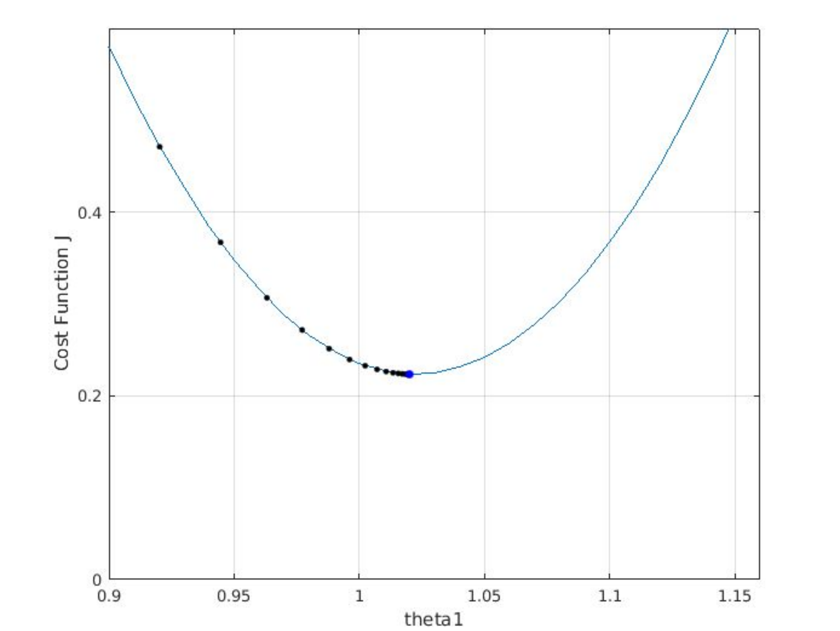
$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1)$$

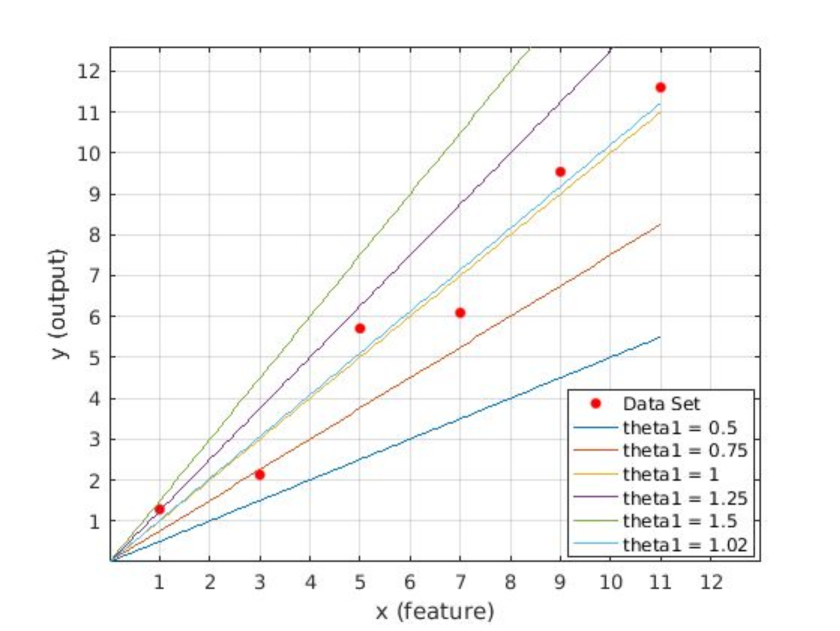
Example:

Initial $\theta_1 = 0.5$
Learning rate $\alpha = 0.005$
Iterations = 20

Optimal values found:
$\theta_1 = 1.02$
$J(\theta_1) = 0.22$

Gradient Descent





$J(\theta_1)$

Initial θ_1 = 0.5

Learning rate α = 0.005

Iterations = 20

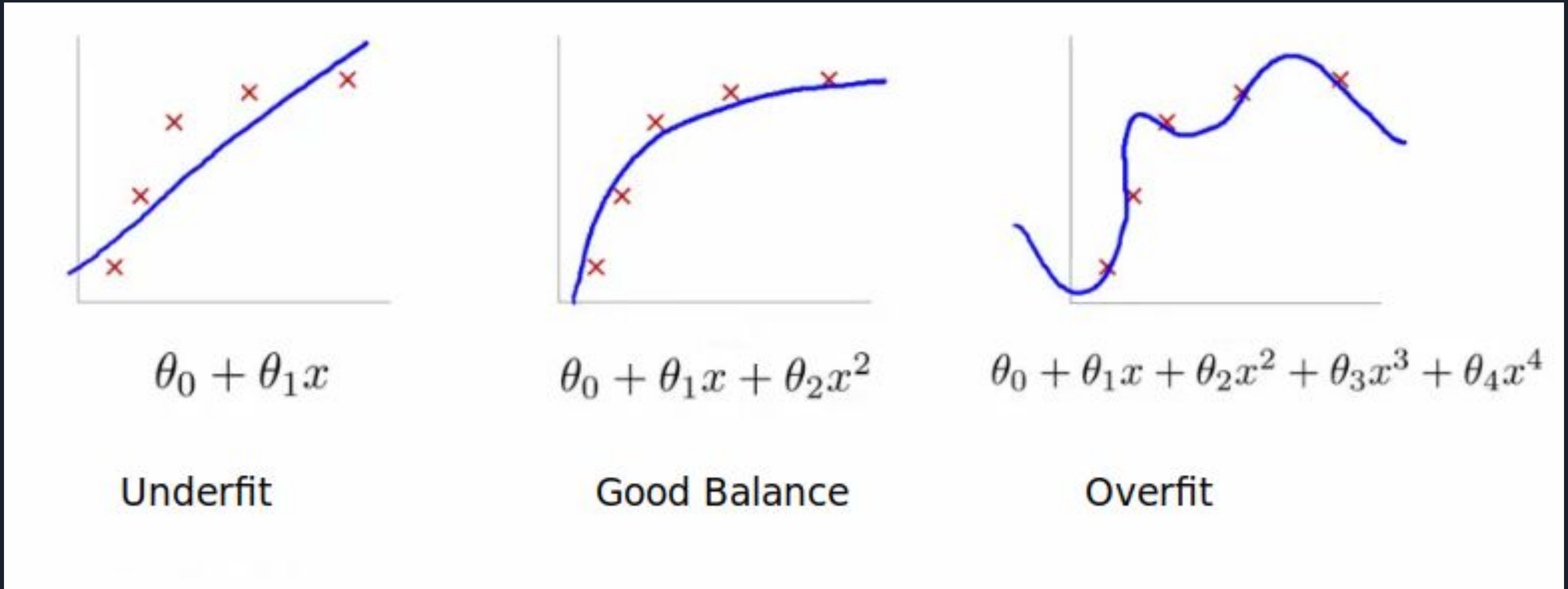
$h_\theta(x) = \theta_1 \cdot x$

Optimal values found:

θ_1 = 1.02

$J(\theta_1)$ = 0.22

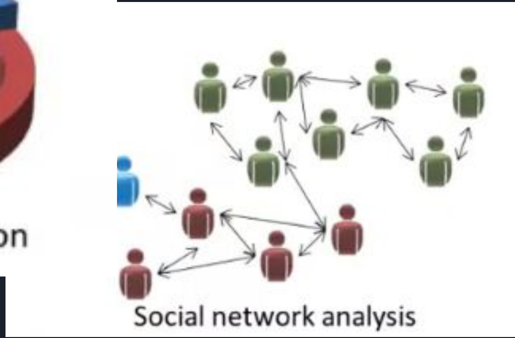
Underfitting & Overfitting



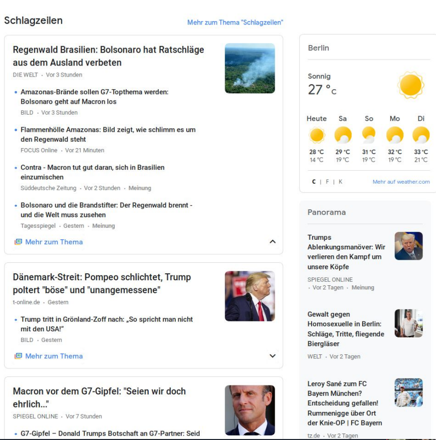
Unsupervised Learning

Key Points:

- The outcome or output for the given inputs is unknown. Data is not labelled
- Mostly used to find patterns or relationships among the given data



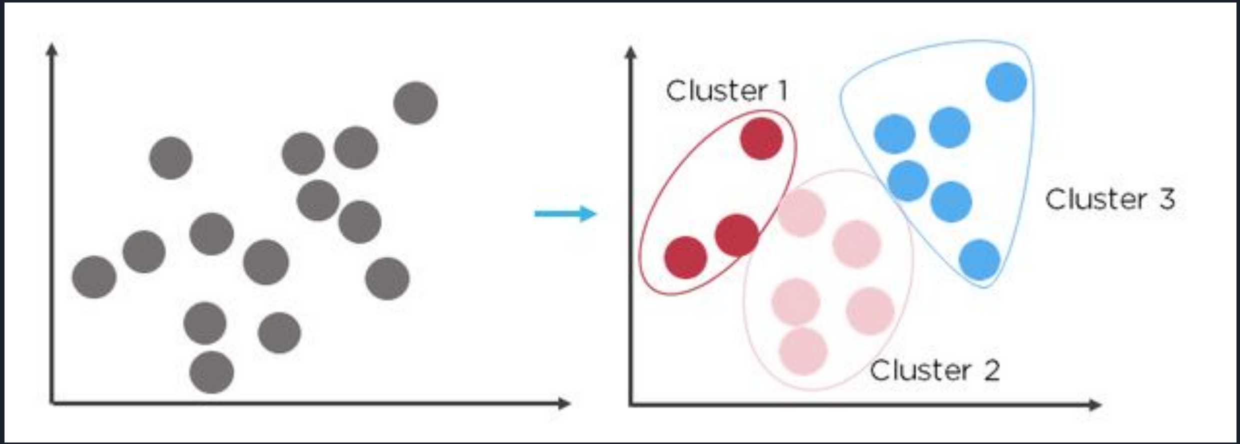
Examples:
Google news: group news of the same kind/topic. Market segmentation.
Human Genome: automatically group genes based on different features
Cocktail party problem/algorithm



Common Algorithms:

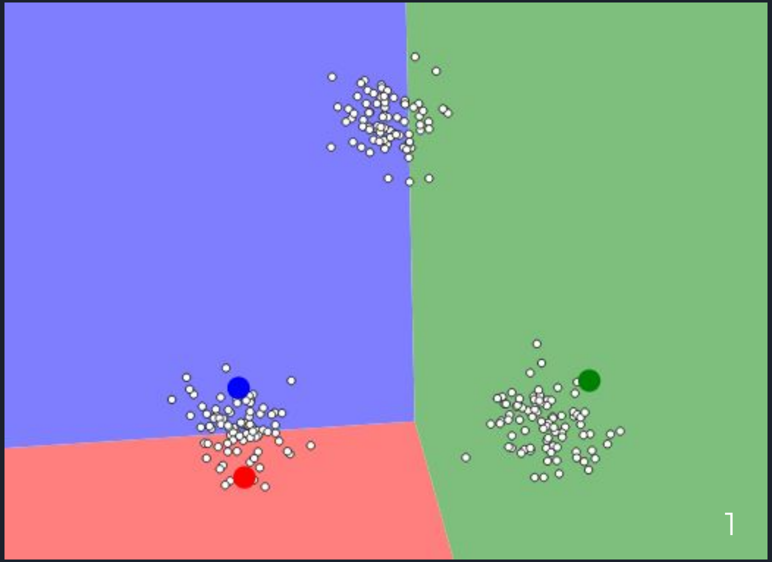
- K-means (clustering)
- Hierarchical Clustering
- Principal component Analysis (PCA)
- Singular value decomposition (SVD) (Dimensionality Reduction)

Clustering: K-Means



k-means Clustering divides data into multiple clusters for analyzation. For each cluster, centroids are calculated. Data is then assigned to a cluster according to its relation (normally "distance measure") to each centroid, grouping similar data together

Clustering: K-Means



Steps to k-means clustering:

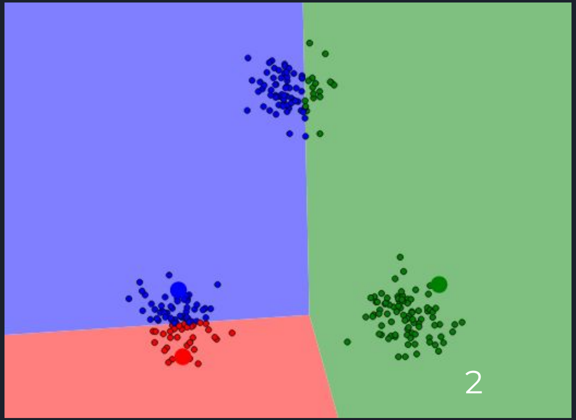
1. Define k centroids. Initialize these at random (there are also better algorithms for initializing the centroids that end up converging more effectively).

2. Determine the distances of each data point to the centroids and assign each point to the closest cluster centroid based (upon minimum distance, often Euclidean distance)

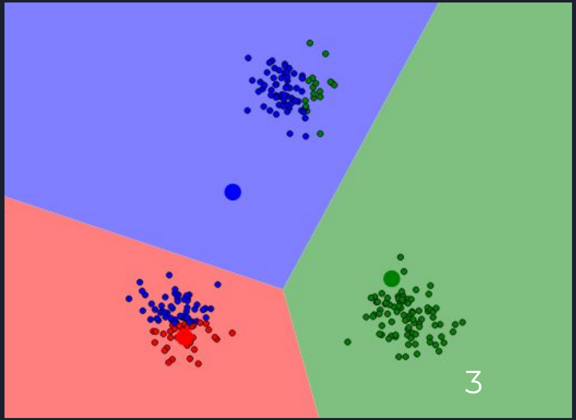
3. Calculate cluster centroids again

Repeat steps 2 and 3 until optimum where there is no switching of data points from one cluster to another (or the centroid "movement" remains below a certain threshold)

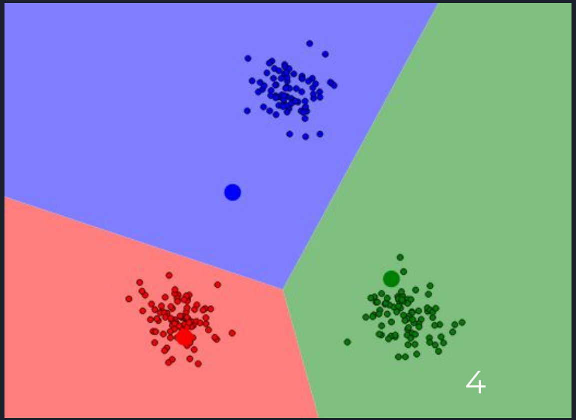
Clustering: K-Means



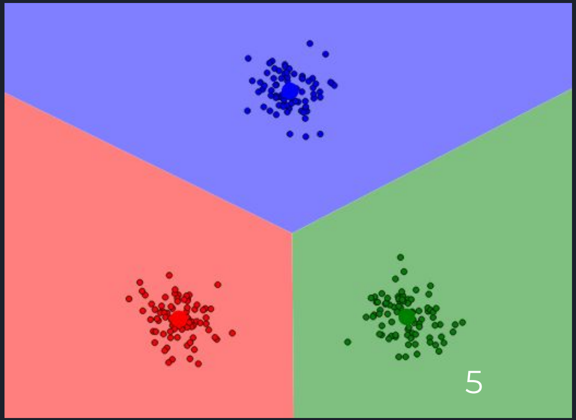
Step 2



Step 3



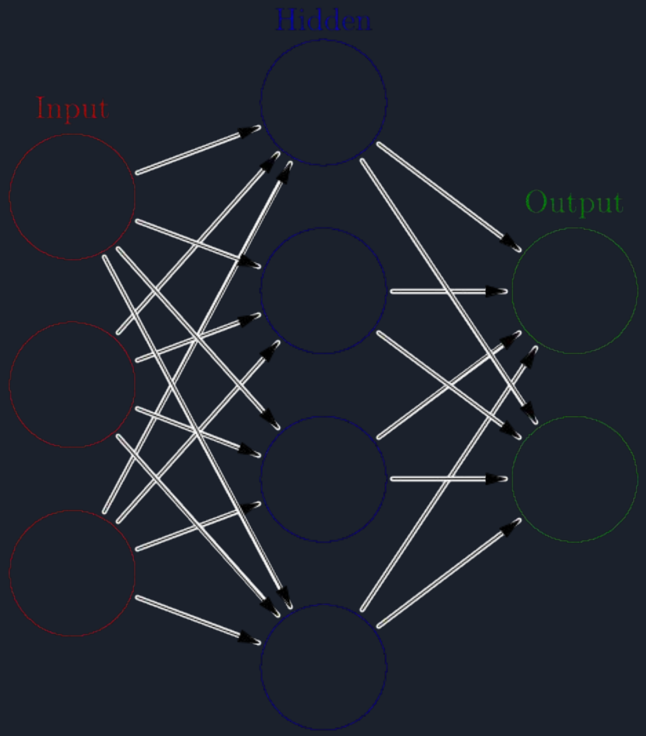
Repeat Step 2



Repeat Step 3

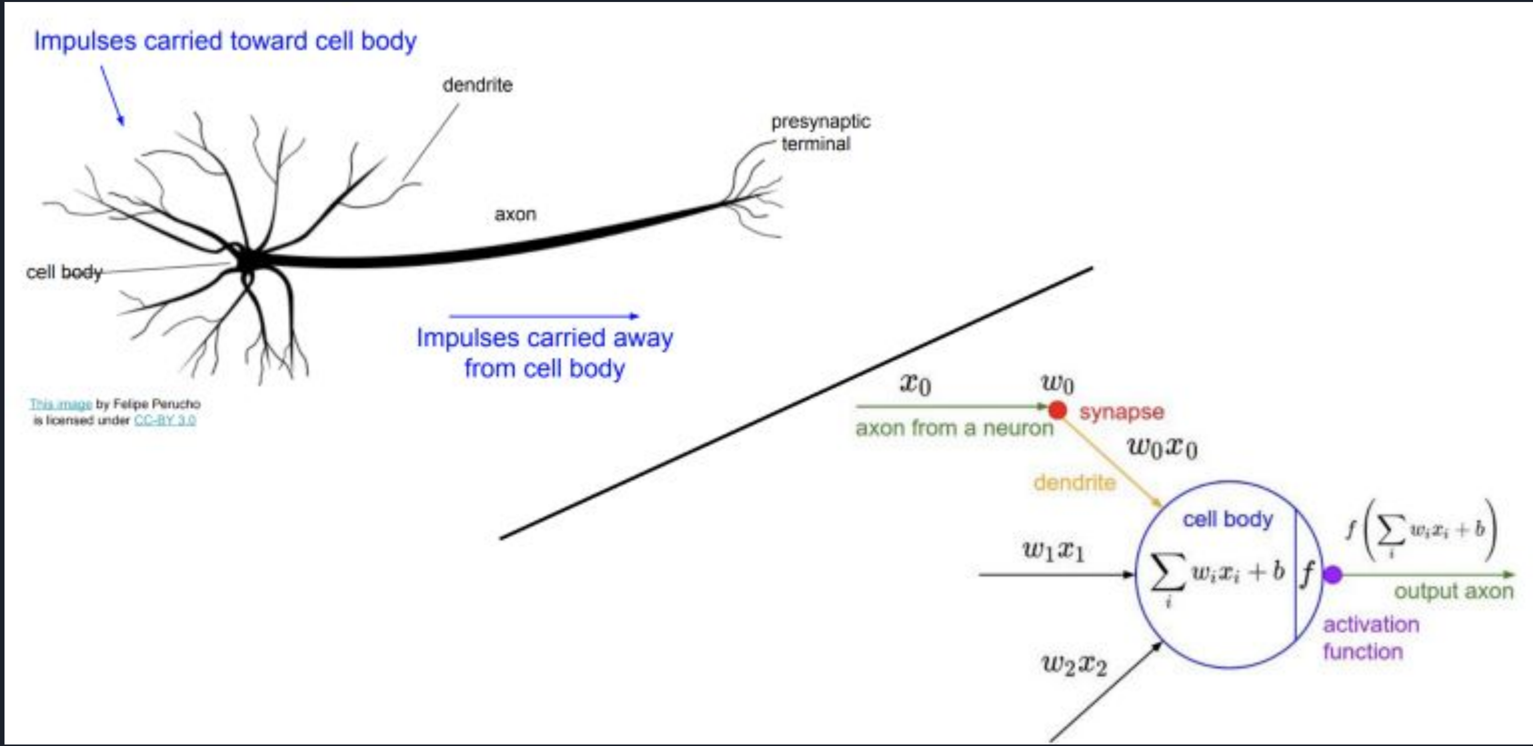
Artificial Neural Networks

Fundamentals



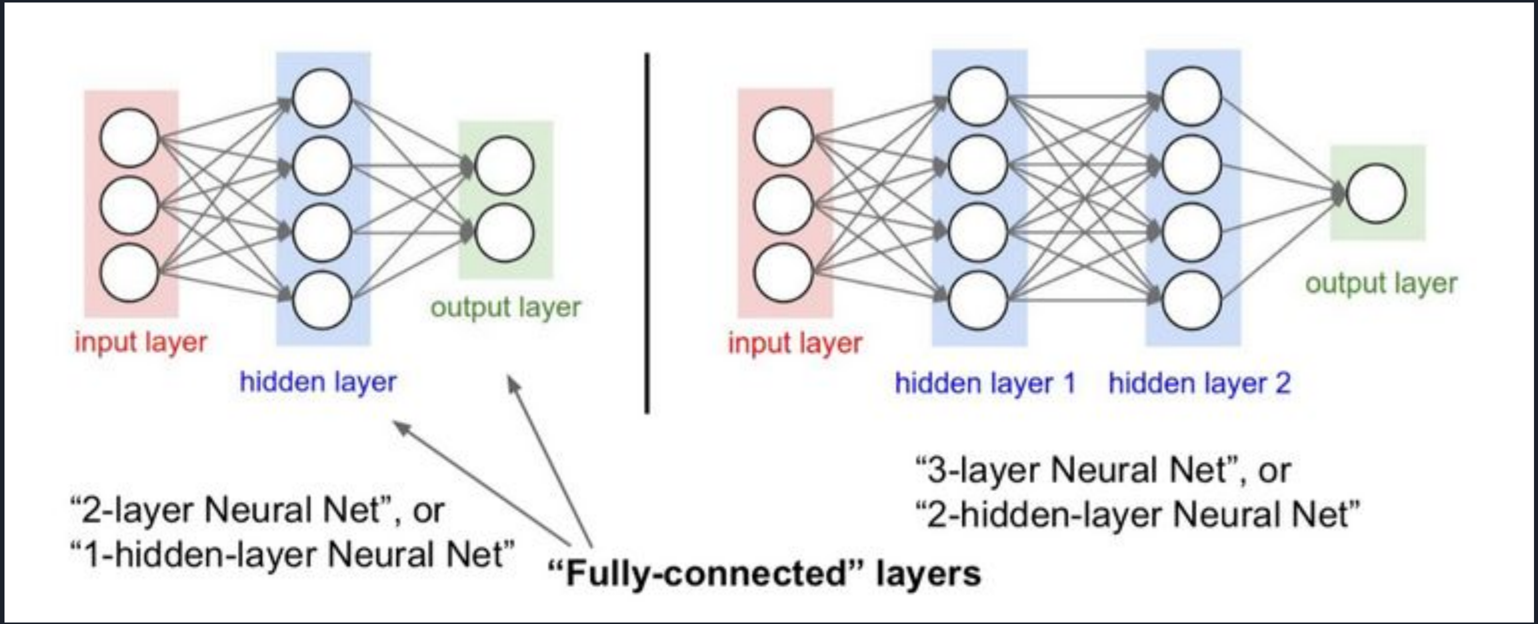
Artificial Neural Networks

Biological Inspiration:

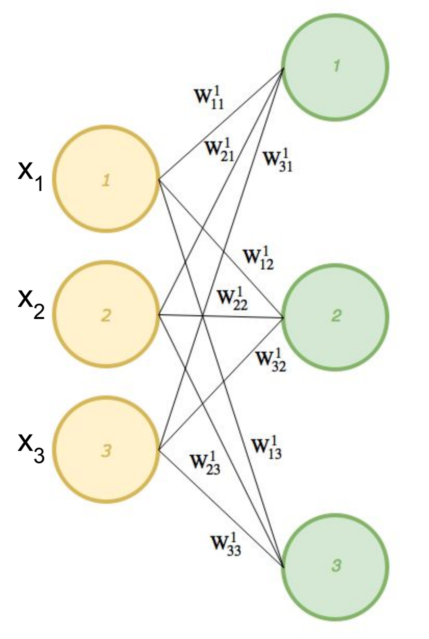


Artificial Neural Networks

Architecture & Terminology



Artificial Neural Networks



Weighted Sum z_1^1:

$$z_1^1 = x_1 \cdot w_{11}^1 + x_2 \cdot w_{21}^1 + x_3 \cdot w_{31}^1 = \sum_{i=1}^{3} x_i \cdot w_{i1}^1$$

Vector - Matrix Representation:

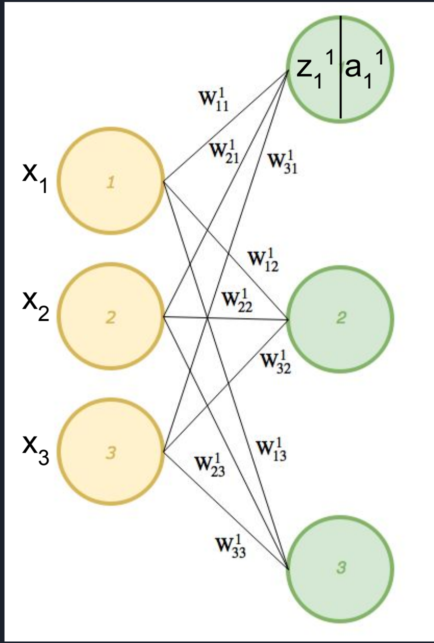
$$X = [x_1 \ x_2 \ x_2]$$

$$W^1 = \begin{bmatrix} W_{11}^1 & W_{12}^1 & W_{13}^1 \\ W_{21}^1 & W_{22}^1 & W_{23}^1 \\ W_{31}^1 & W_{32}^1 & W_{33}^1 \end{bmatrix}$$

$Z = X \cdot W^1$, where Z is a 3x1 vector in this case

Artificial Neural Networks

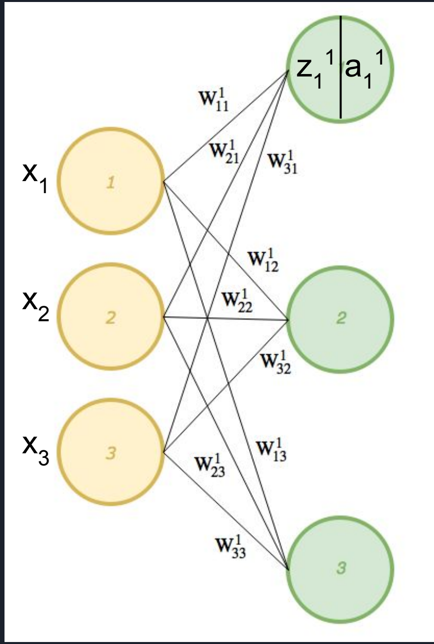
Activation Function



$$a_1^1 = f(z_1^1) = f(\sum_{i=1}^{3} x_i \cdot w_{i1}^1)$$

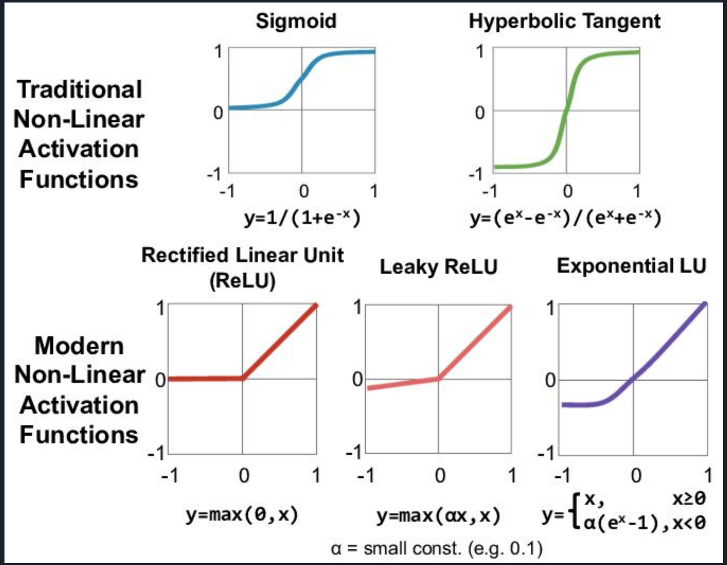
Artificial Neural Networks

Activation Function

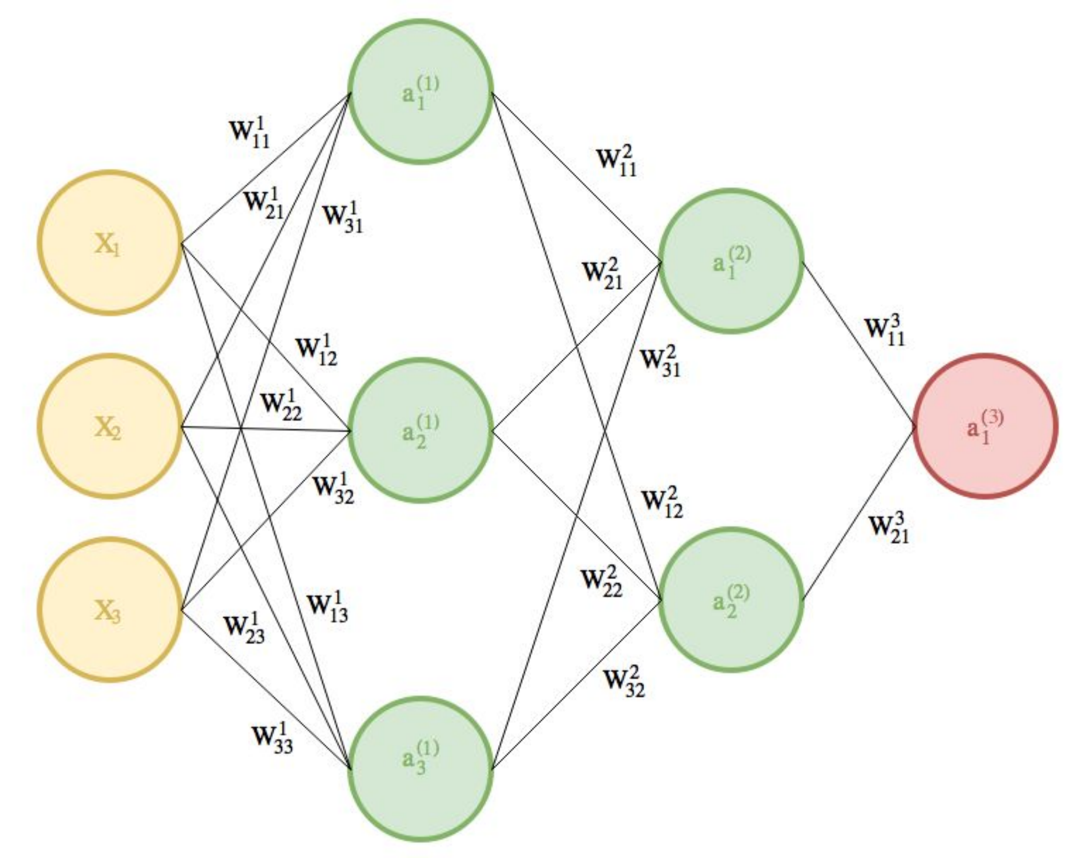


$$a_1^1 = f(z_1^1) = f(\sum_{i=1}^{3} x_i \cdot w_{i1}^1)$$

Some Activation Functions:



Artificial Neural Networks



Concise Matrix Representation

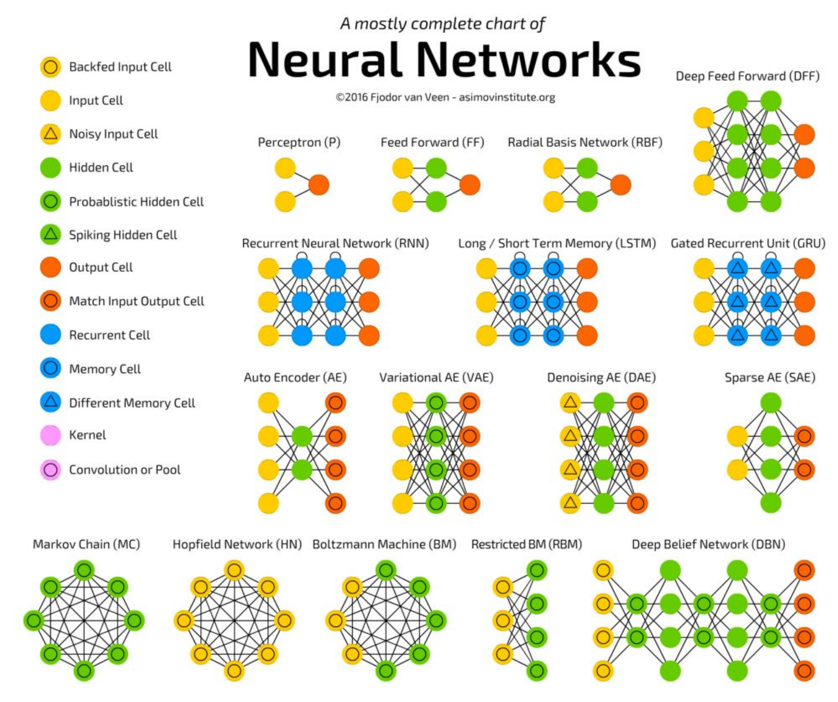
$$a^{(L)} = f\,(a^{(L-1)} \cdot W^{(L)})$$

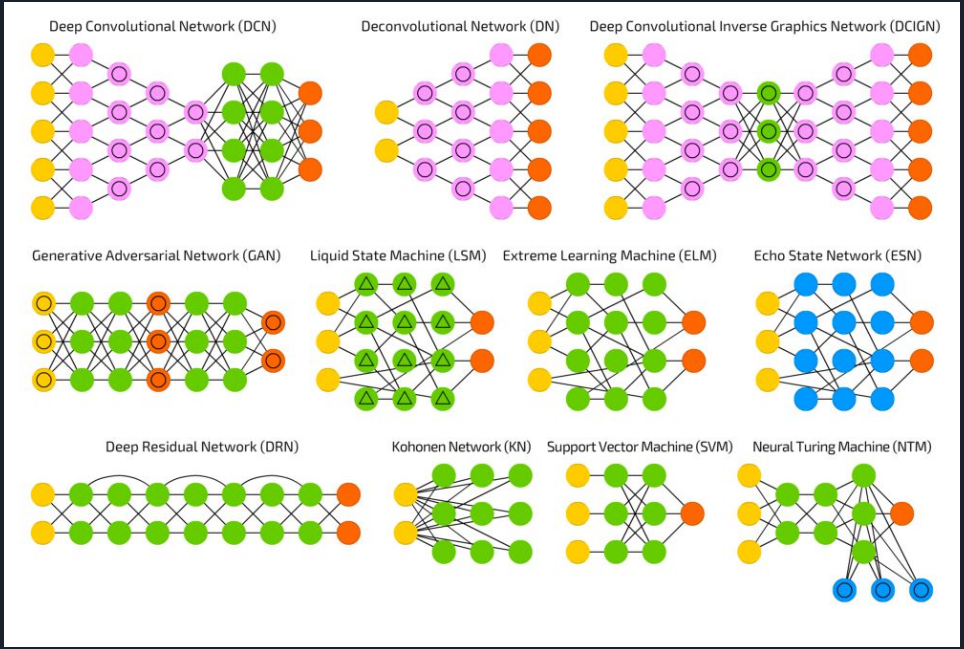
L: layer number

$$a^{(0)} = X$$

Artificial Neural Networks

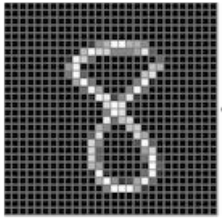
Topologies





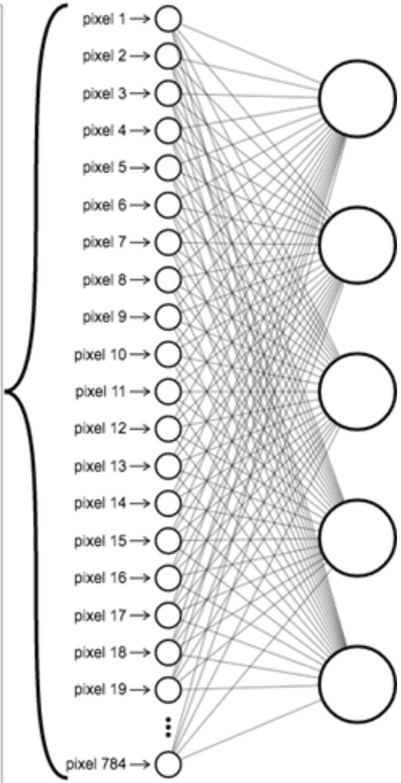
Artificial Neural Networks

Example: MNIST database (Input)



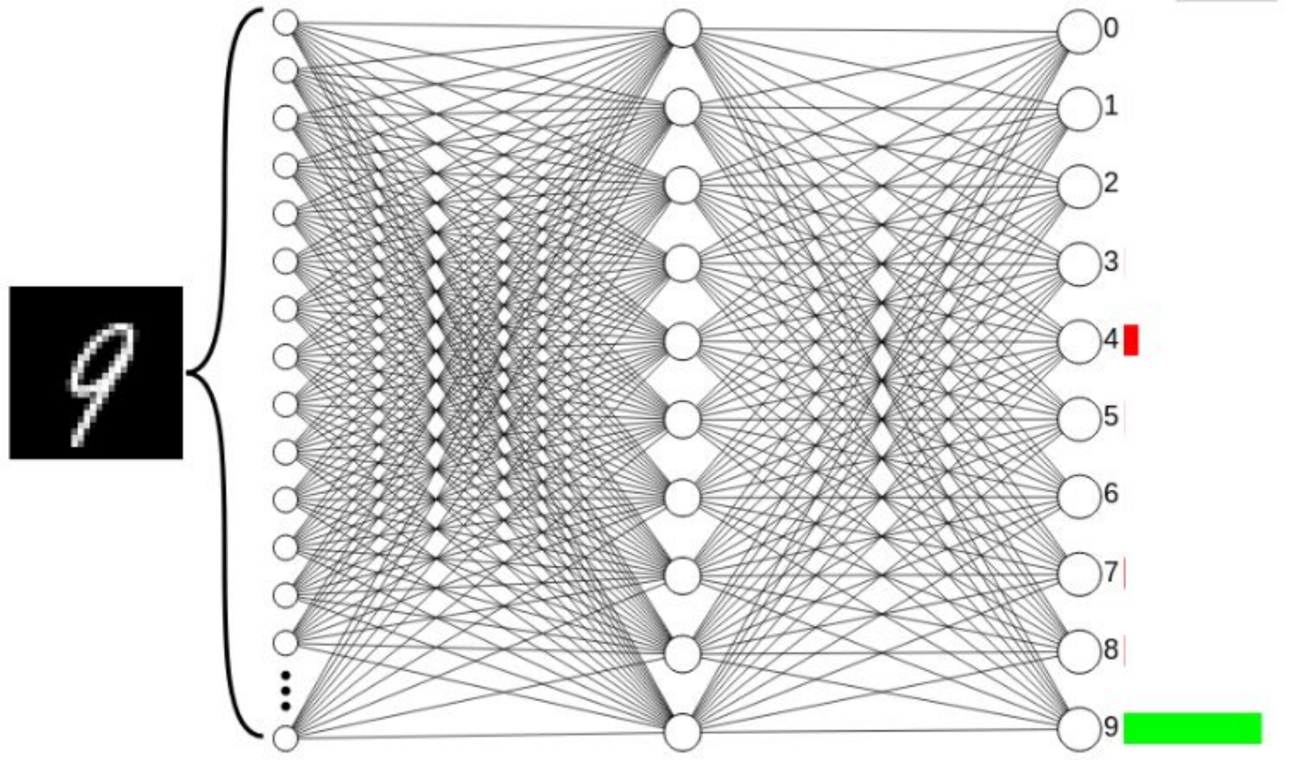
28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



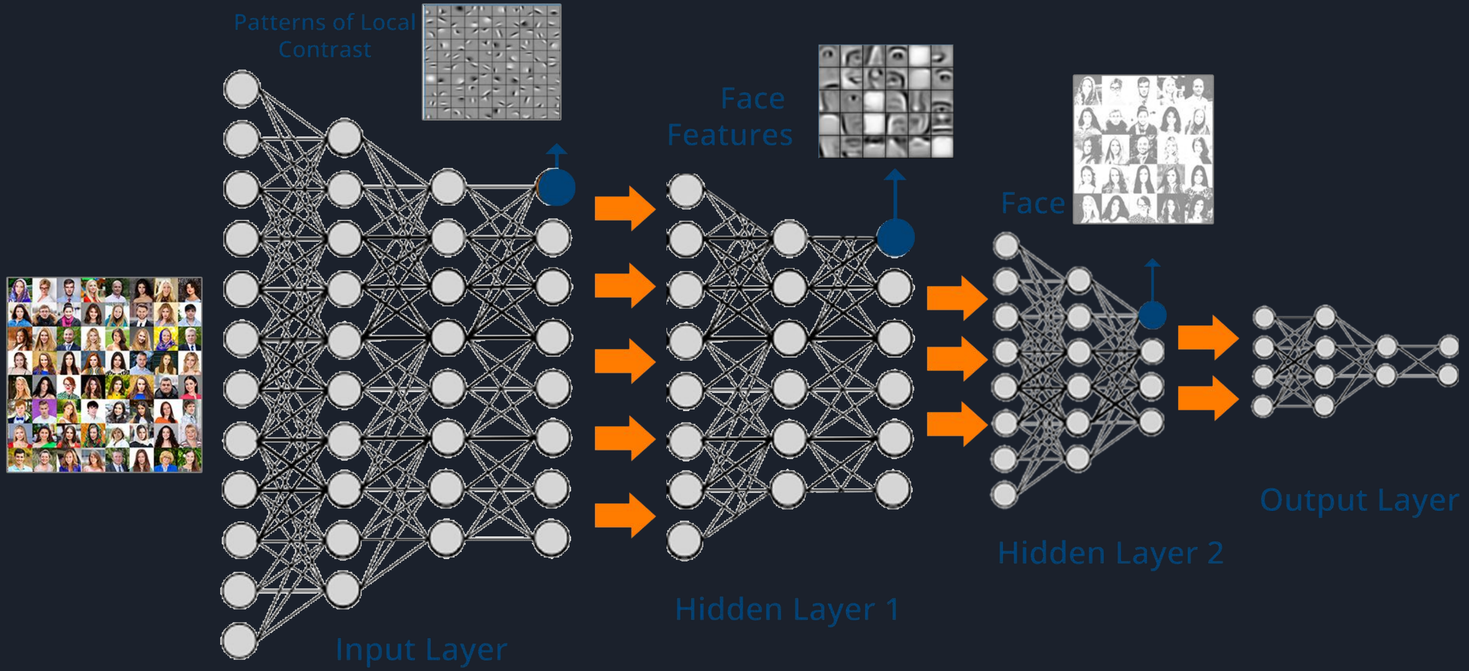
Artificial Neural Networks

Example: MNIST database



Artificial Neural Networks

Progressive feature detection



Artificial Neural Networks

Example: MobileNetV2

Neural Network (CNN): MobileNetV2

Input: 224 x 224 x 3 (150 528)

Layers: 157

Total parameters: 3 538 984

Labels (categories): 1000

('n02093754', 'Border_terrier', 0.97215736)
('n02095570', 'Lakeland_terrier', 0.0019579164)
('n02094114', 'Norfolk_terrier', 0.0008627547)
('n02096051', 'Airedale', 0.000678198)
('n02094258', 'Norwich_terrier', 0.0004774261)

Reinforcement Learning

Key Points:

- The learning algorithm, or ”“**Agent**” interacts with an environment and learns from experience
- The Agent collects training information with the goal of maximizing a reward (or “minimizing the punishment”).

Examples:

- Games: AlphaZero (Go, Chess), Dota II (OpenAI)
- Robotic hands
- Self-driving Cars

Common Algorithms:

- Q-Learning: involves predicting the value of Q (quality) given other values in a matrix
- DQ-Learning. The Q function is predicted with a Neural Network
- Temporal Difference (TD)

Reinforcement Learning

Terminology

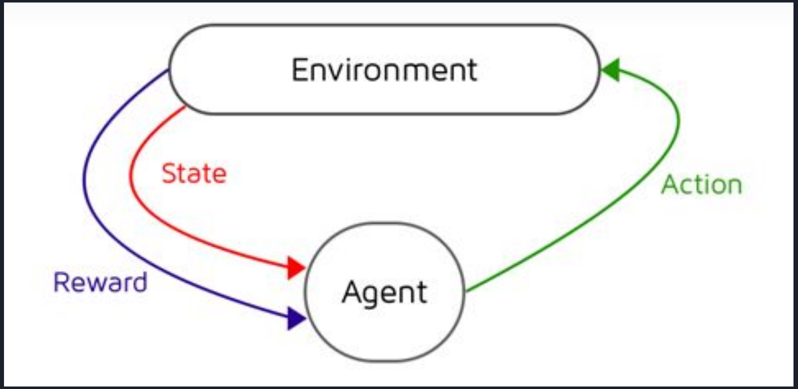
- **Environment**: World/Frame in which the Agent interacts/operates

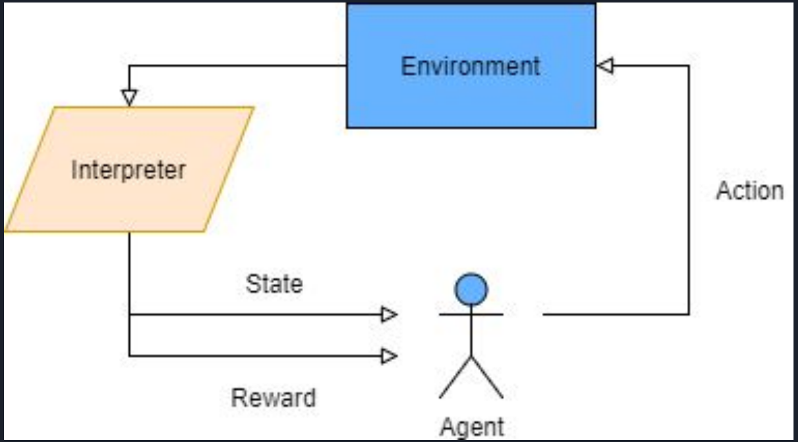
- **Agent**: The explorer, learner and decision-maker in the reinforcement learning scenario

- **Reward**: Feedback signal from the environment (or from the “interpreter”). Used to train the Agent

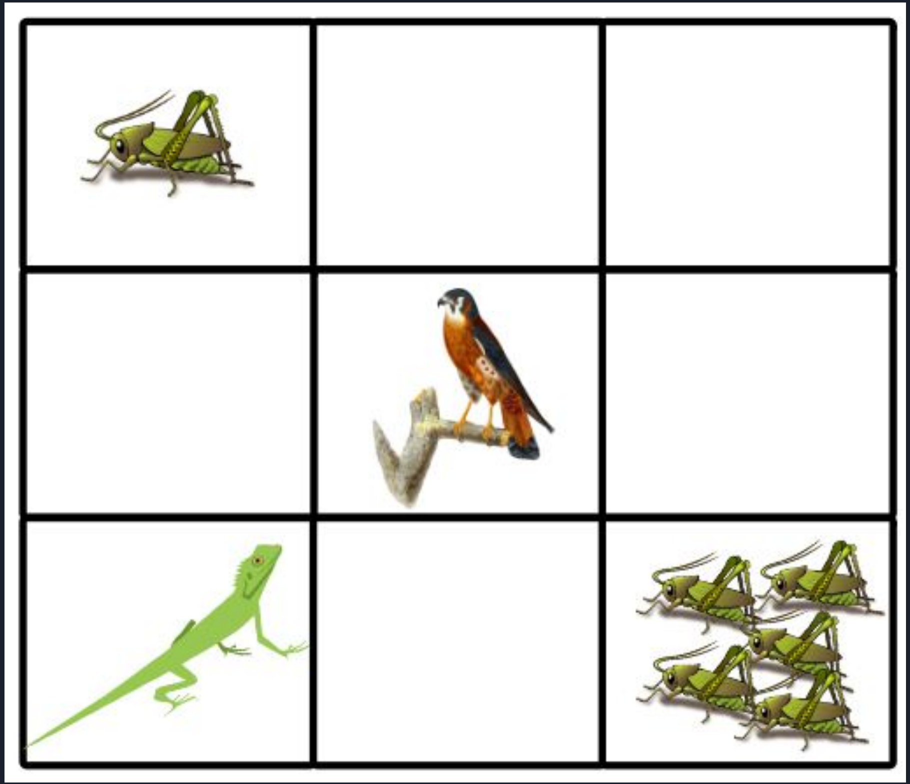
- **State**: Current situation of the agent (based on its observations)

- **Action**: What the agent can do in the environment





Reinforcement Learning

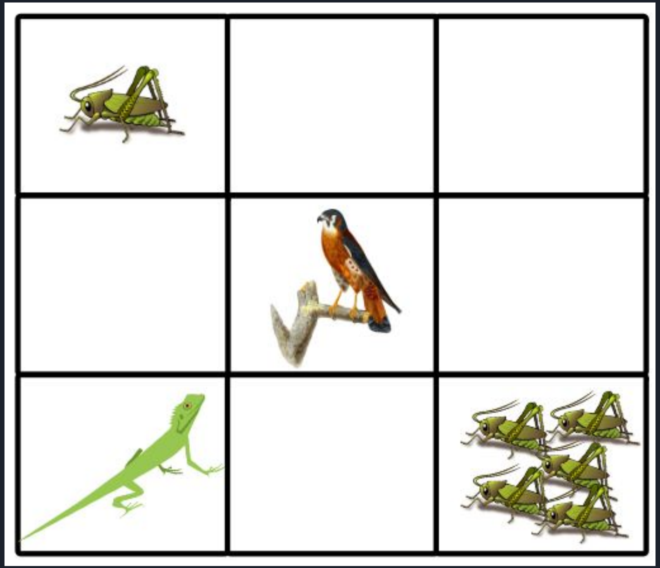


Agent: the lizard

Actions: moving left, right, up, or down

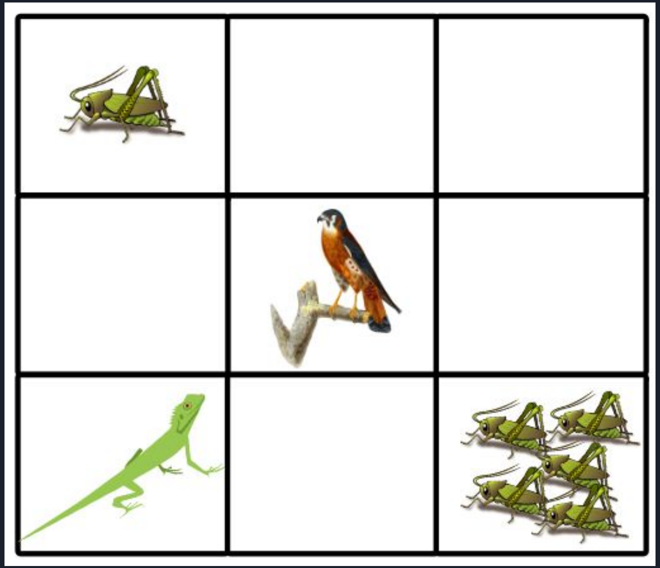
Situation	Reward
One cricket	+1
Empty	0
Hits Wall	-1
Cricket Group	+10 Game Over
Bird	-10 Game Over

Reinforcement Learning



Q Table		Actions (a)			
		Left	Right	Up	Down
s	1. 1 Cricket				
	2. Empty (1)				
	3. Empty (2)				
	4. Empty (3)				
	5. Bird				
	6. Empty (4)				
	7. Empty (5)				
	8. Empty (6)				
	9. 5 Crickets				

Reinforcement Learning



Q Table		Actions (a)			
		Left	Right	Up	Down
s	1. 1 Cricket				
	2. Empty (1)				
	3. Empty (2)				
	4. Empty (3)				
	5. Bird				
	6. Empty (4)				
	7. Empty (5)				
	8. Empty (6)				
	9. 5 Crickets				

$$q(s,a) \;=\; r\,(s,a) \;+\; \gamma \cdot max_{a'}\,(q(s',a'))$$

γ : discount factor

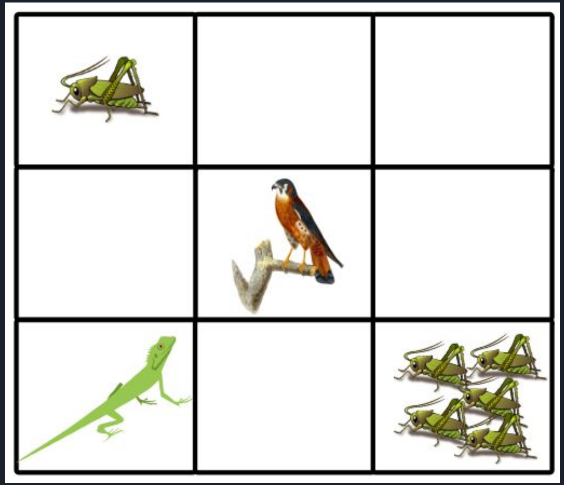
Iterating, q(s,a) is update as follows:

$$q(s,a) \;\leftarrow (1-\alpha) \;\cdot q(s,a) \;+ \alpha\,[\; r\,(s,a) \;+\; \gamma \cdot max_{a'}\,(q(s',a'))\;]$$

α: Learning Rate (0...1) E.g.: 0.1
γ : discount factor (0...1), E.g.: 0.9

Reinforcement Learning

Exploration vs Exploitation



The epsilon (ϵ) - greedy strategy

To get a balance between exploitation and exploration, in the iterative process we use what is called an epsilon - greedy strategy.
We define an exploration rate ϵ that we initially set to 1.
This exploration rate is the probability that our agent will explore the environment rather than exploit it. (ϵ=1 => 100%).

As the agent learns more about the environment, at the start of each new episode, ϵ will decay by some rate, i.e. explore less and exploit more.
The agent will become “greedy” in terms of exploiting the environment once it has had the opportunity to explore and learn more about it.

Reinforcement Learning

The Q-learning Algorithm:

Initialize Q(s,a) for all (s,a)

Repeat for each **episode:**

    Initialize state s (and Environment)

    Repeat for each **step** on the **episode**:

        Given state s, choose action a, using policy derived from Q(s,a) and ϵ - greedy strategy

        Take action a, get new state s' and reward r

        $q(s,a) := (1-\alpha) \cdot q(s,a) + \alpha \cdot [\, r(s,a) + \gamma \cdot \max_{a'}(\, q(s',a')\,)\,]$

        s : = s'

        do until s is terminal

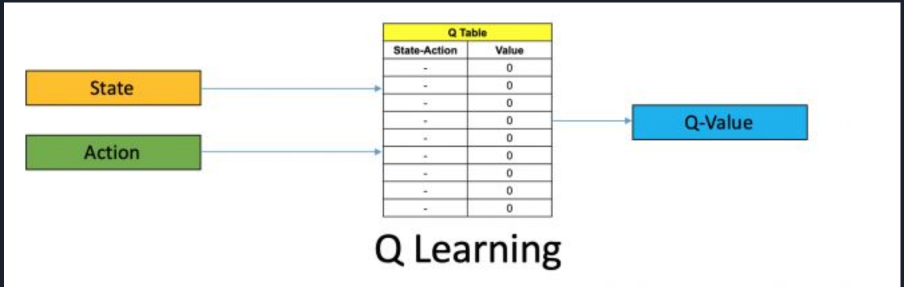
    update ϵ

Reinforcement Learning

Deep Q-learning introduction

(Tabular) Q-learning limitation:

- Does not escalate well. States- space increases very rapidly with every added input.
  E.g. Imagine a video game were each state in the environment would be represented by a set of pixels, (or the game of Go)



Reinforcement Learning

Deep Q-learning introduction

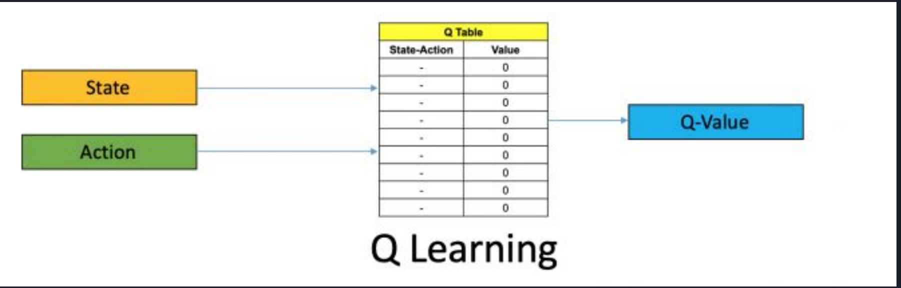
Proposal:

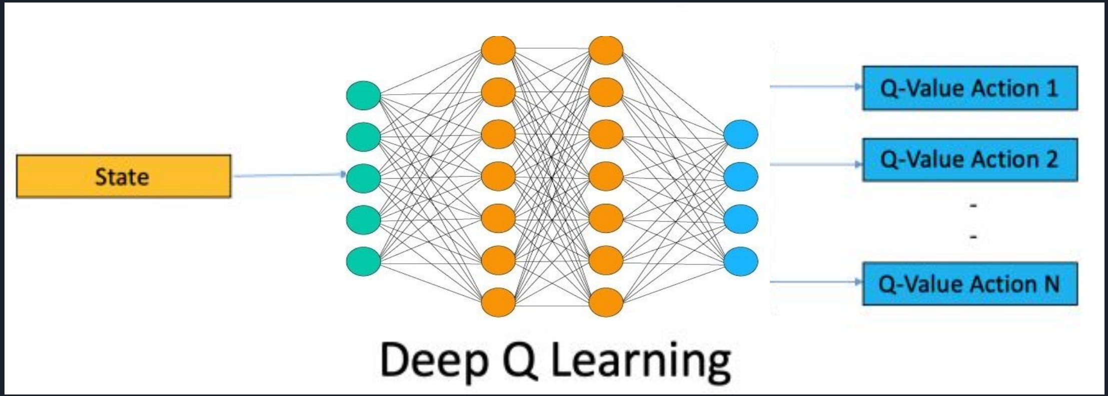
- Use instead a function approximator to estimate the optimal Q-function

- Model to approximate the Q-Function:

  Deep Neural Network

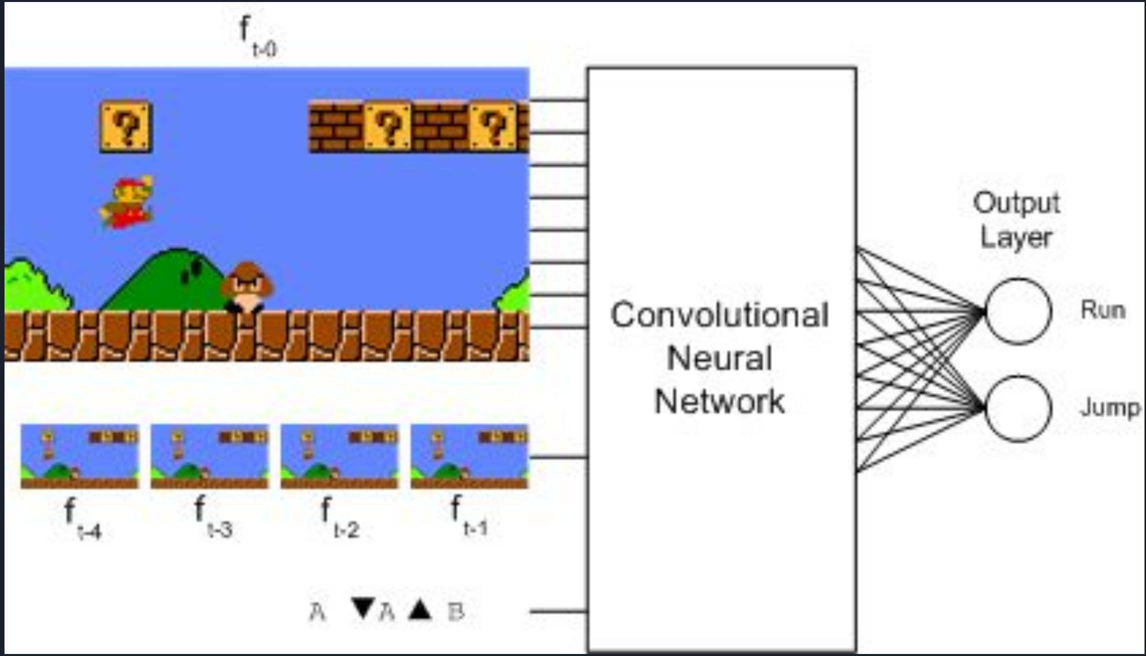
  For this case called:
  Deep Q-Network





Reinforcement Learning

Deep Q-learning: example



References and citation

https://scikit-learn.org/stable/

https://deepmind.com

https://openai.com/

Thomas M. Mitchell. *Machine Learning*. McGraw-Hill, 1997

Goodfellow, Bengio, Courville. *Deep Learning*. MIT Press, 2016

https://www.deeplearningbook.org/

Sutton, Barto. *Reinforcement Learning: An Introduction*.MIT Press, 2012

https://machinelearningmastery.com/

https://www.xkcd.com/

https://www.naftaliharris.com/blog/visualizing-k-means-clustering/

“Thank you very much”

apolos@gmail.com