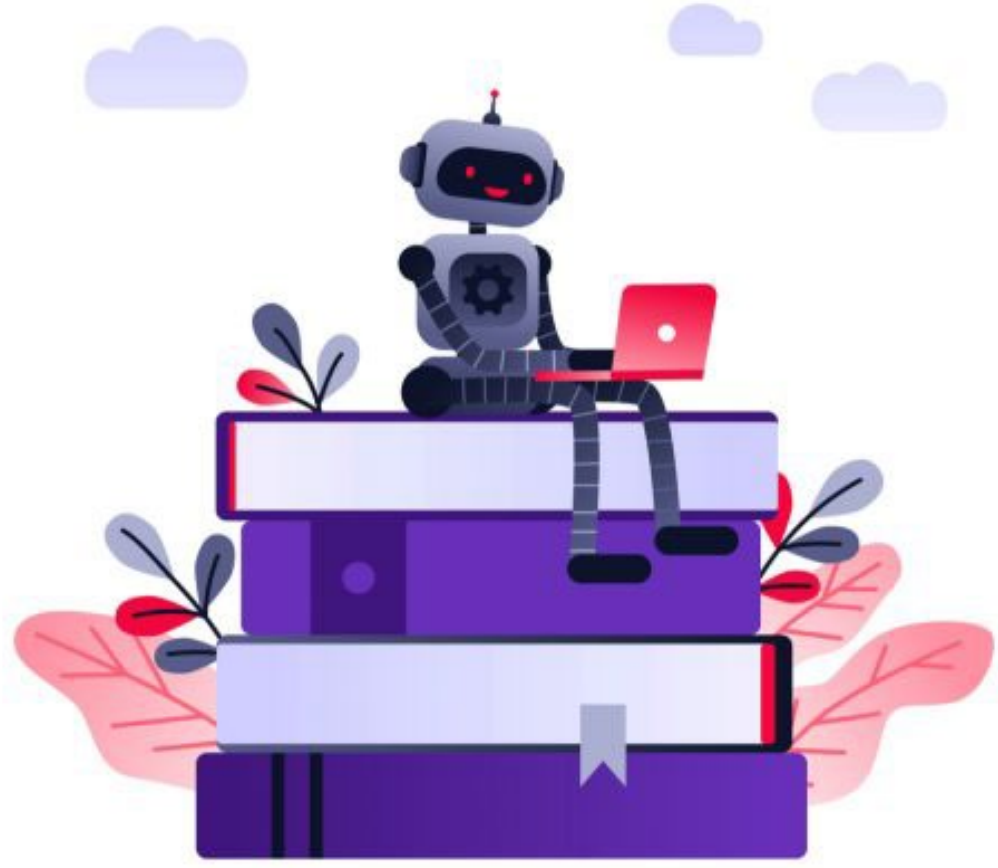
Applied Machine Learning



NUS National University of Singapore | School of Computing

ABOUT THE LECTURER

NUS National University of Singapore | School of Computing

Dr Ai Xin

Dr. Ai Xin is currently a Lecturer with the School of Computing at the National University of Singapore (NUS). She has many years' experience on teaching Artificial Intelligence and Data Science courses, e.g. machine learning, deep learning, data mining and etc.

She graduated from NUS with a PhD degree on Electrical and Computer Engineering. Her research focused on Game Theoretical Modelling, Optimization Methods, Algorithm Design and Wireless Networks.

She worked in BHP Billiton Marketing Asia for eight years and gained a lot of industry experience through different functions, e.g. risk management, supply chain management, sales and marketing planning and etc.

HTTPS://ACE.NUS.EDU.SG/ABOUT-US/INSTRUCTORS/

AGENDA

NUS National University of Singapore | School of Computing

Topic	Time (9am to 5:30pm)
• Introduction • Practical 1 & 2 (Self Study)	Day 1 (AM)
• Clustering: K-Means • Practical 3	Day 1 (AM)
• Classification: K Nearest Neighbors • Practical 4	Day 1 (PM)
• Regression: Linear and Polynomial • Practical 5	Day 1 (PM)
• Support Vector Machine • Practical 6	Day 2 (AM)
• Artificial Neural Networks • Practical 7	Day 2 (AM)
• Deep Learning • Practical 8	Day 2 (PM)
• Assessment: Mini Project (Passing Mark 60%)	Day 2 (PM)

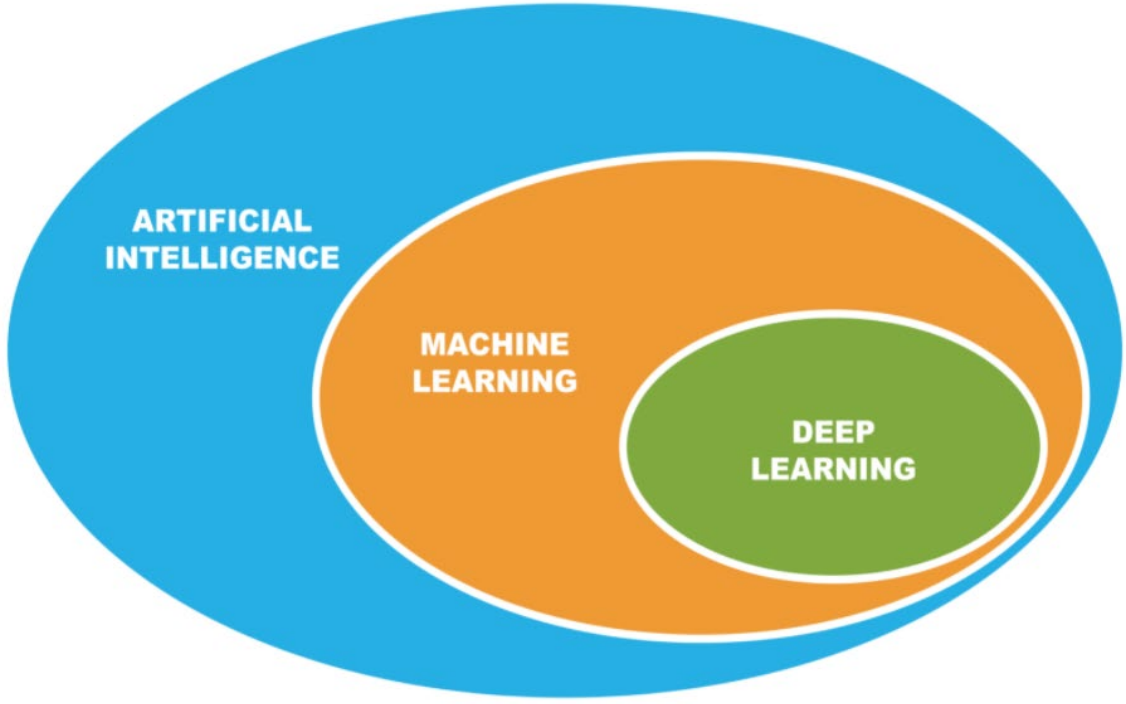
Lunch Breaks (1-2pm); AM Breaks (11-11:15am); PM Breaks (4-4:15pm)

Introduction

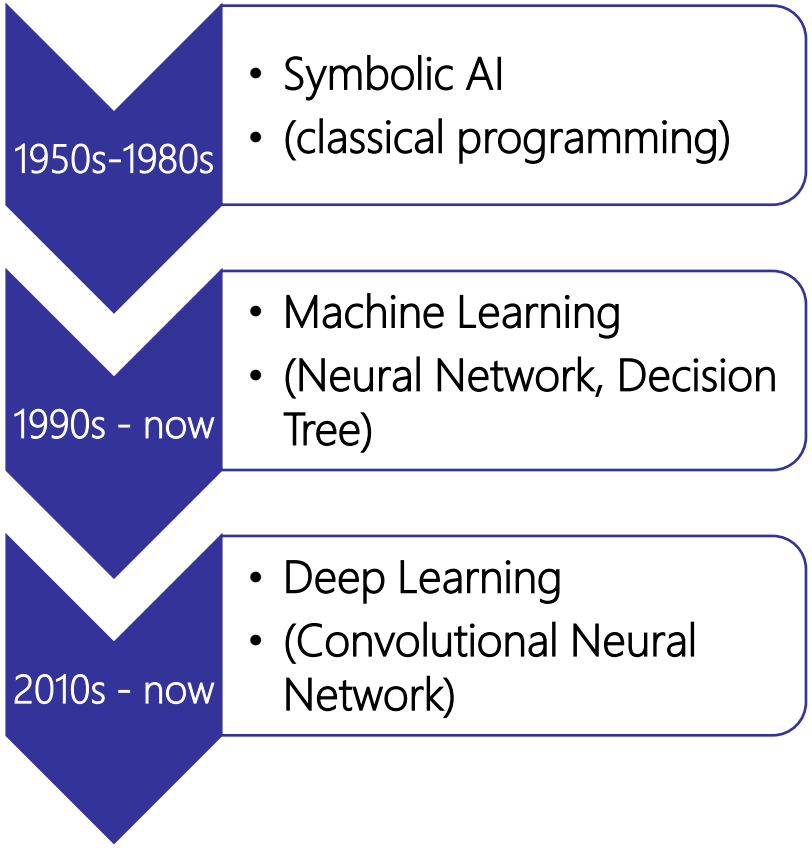
WHAT IS AI?

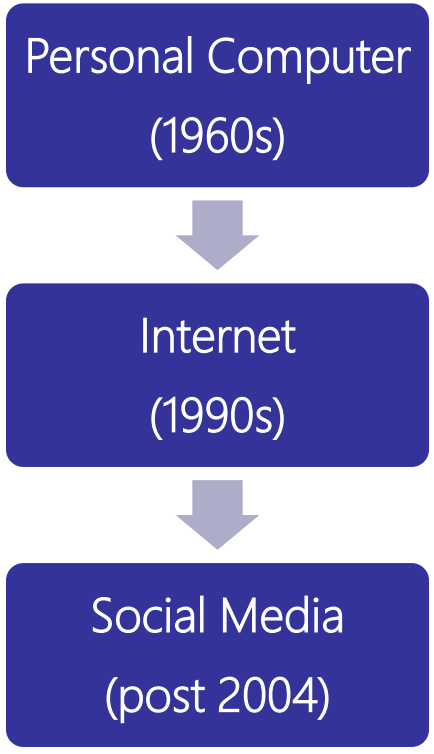
AI is the effort to automate intellectual tasks normally performed by humans

AI is a general field that encompasses machine learning and deep learning, and many other approaches which don't involve any learning.



A bit History of AI





WHAT IS MACHINE LEARNING?

- Machine Learning is the science (and art) of programming computers so they can learn from data.

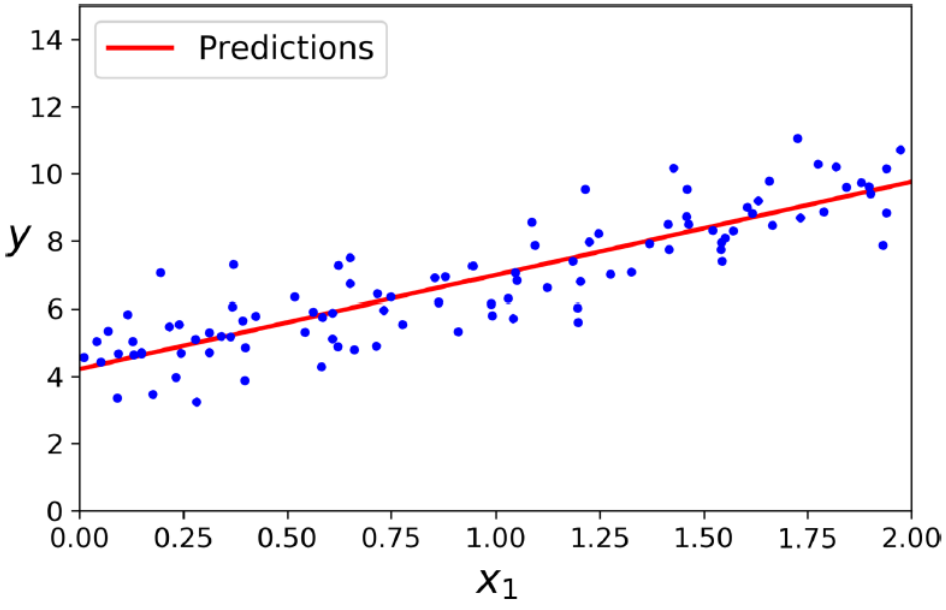
- A slightly more general definition:

[Machine Learning is the] field of study that gives computers the ability to learn without being explicitly programmed.

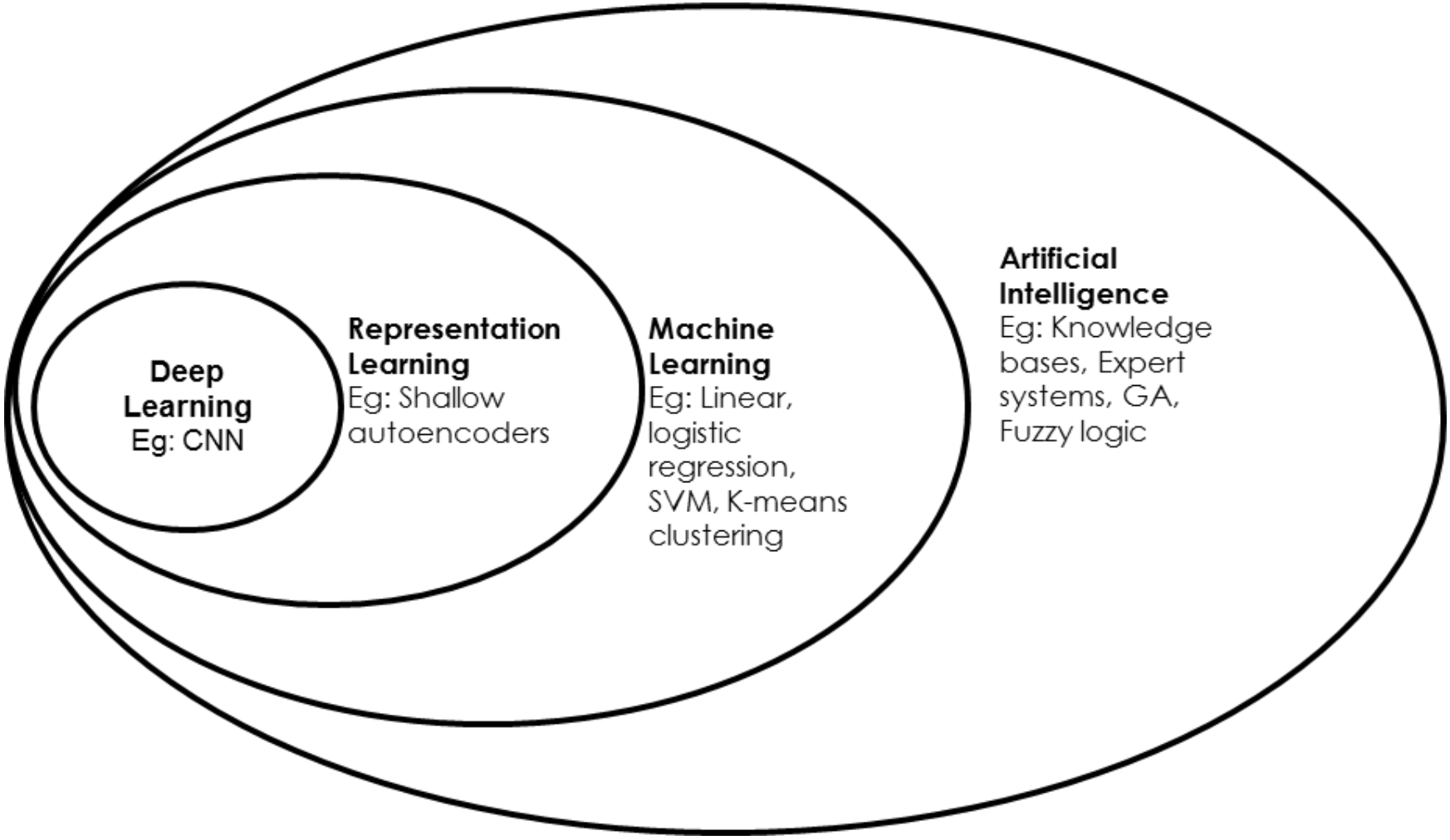
—Arthur Samuel, 1959

Machine Learning Examples

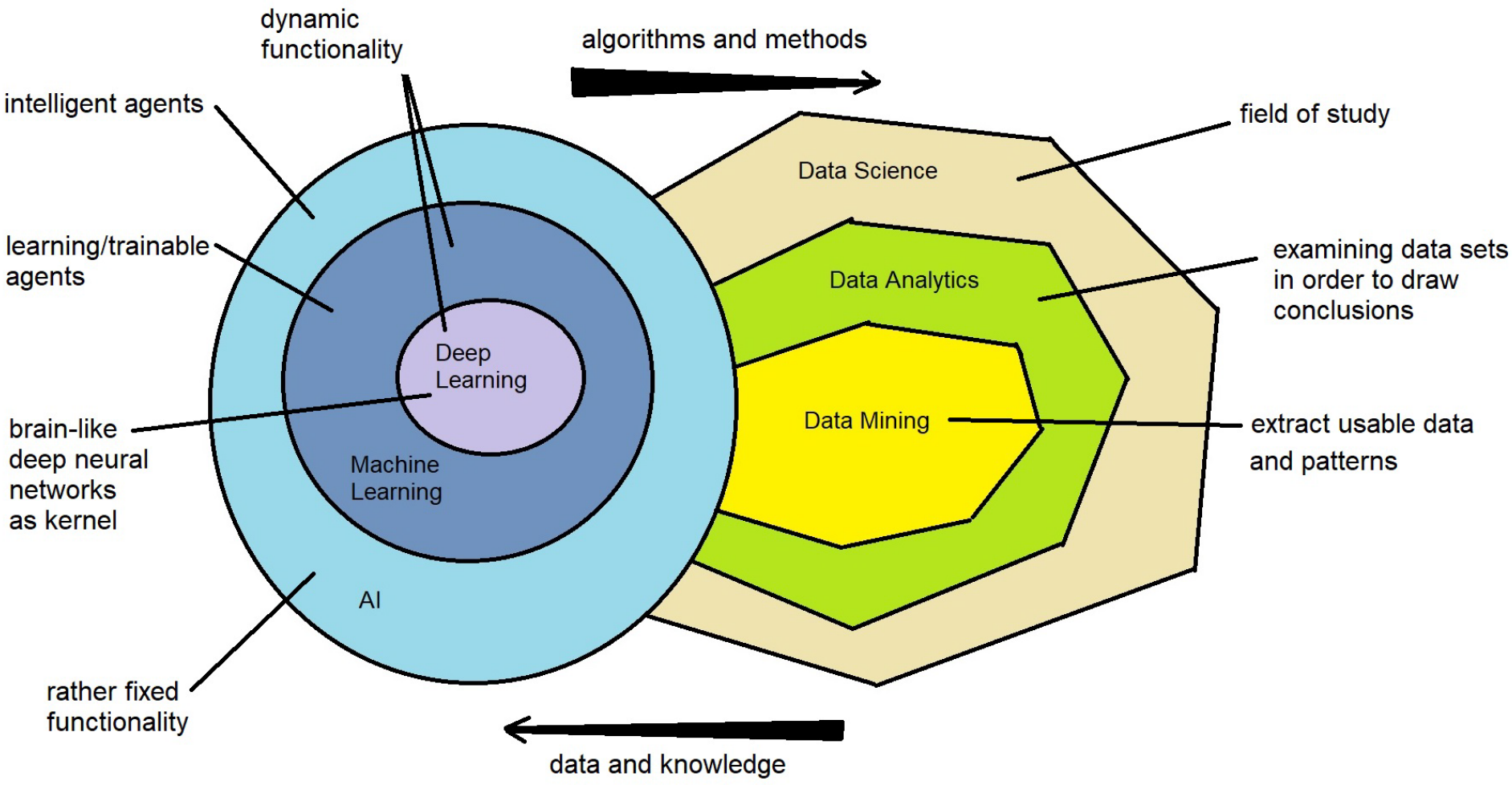
Linear Regression



AI, MACHINE LEARNING, AND DEEP LEARNING



AI, AND DATA SCIENCE



MACHINE LEARNING APPLICATIONS

- Virtual Personal Assistants
- Predictions
- Video Surveillance
- Social Media Services
- Email Spam and Malware Filtering
- Online Customer Support
- Search Engine Result Refining
- Product Recommendations
- Natural Language Processing, Machine Translation, Abstraction
- Driverless Vehicles

PYTHON PROGRAMMING

Python is an interpreted, high-level, general-purpose programming language. Created in 1991.

It supports many OSs, functional, structured, and object oriented programing.

Python 3 was introduced in 2009.

It is the most popular Programming Language for AI, ML, and DA applications.

It is a Free and Open Source Software (FOSS)

WWW.PYTHON.ORG

Python's core philosophy

- Explicit is better than implicit
- Beautiful is better than ugly
- Readability counts.
- Simple is better than complex.
- Complex is better than complicated.

PYTHON PROGRAMMING

Python features are a combination of high level PLs, along with functional PLs (e.g. LISP), and matrix-oriented PLs (e.g. MATLAB).

Personal and organizational investment in learning Python are rational.

A big open source community is supporting that, you wont walk alone…

Many examples, toolboxes, and packages available for free

PACKAGES AND TOOLS WE'RE GOING TO USE

Anaconda

Tensorflow

Keras (in TF)

Pandas

Numpy

Matplotlib

Seaborn

Sklearn

Scikit

Jupyter Notebook

… and a few more

PYTHON PACKAGES

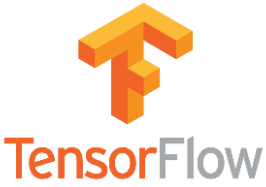
We use **sklearn** package for ML

It is an Open-source ML library for Python. Built on NumPy, SciPy, and Matplotlib. Scikit-learn is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology you might already be familiar with, like NumPy, Pandas, and Matplotlib!

Scikit-learn is a free machine learning library for Python. It features various algorithms like support vector machine, random forests, and k-neighbours, and it also supports Python numerical and scientific libraries like NumPy and SciPy. Then we'll dive into scikit-learn and use preprocessing.

PYTHON PACKAGES

Tensorflow: An end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community. It's very useful for ANN and Deep Learning.



Keras: An open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML.

Jupyter Notebook: an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

PRACTICAL 1: PYTHON BASICS (SELF STUDY)

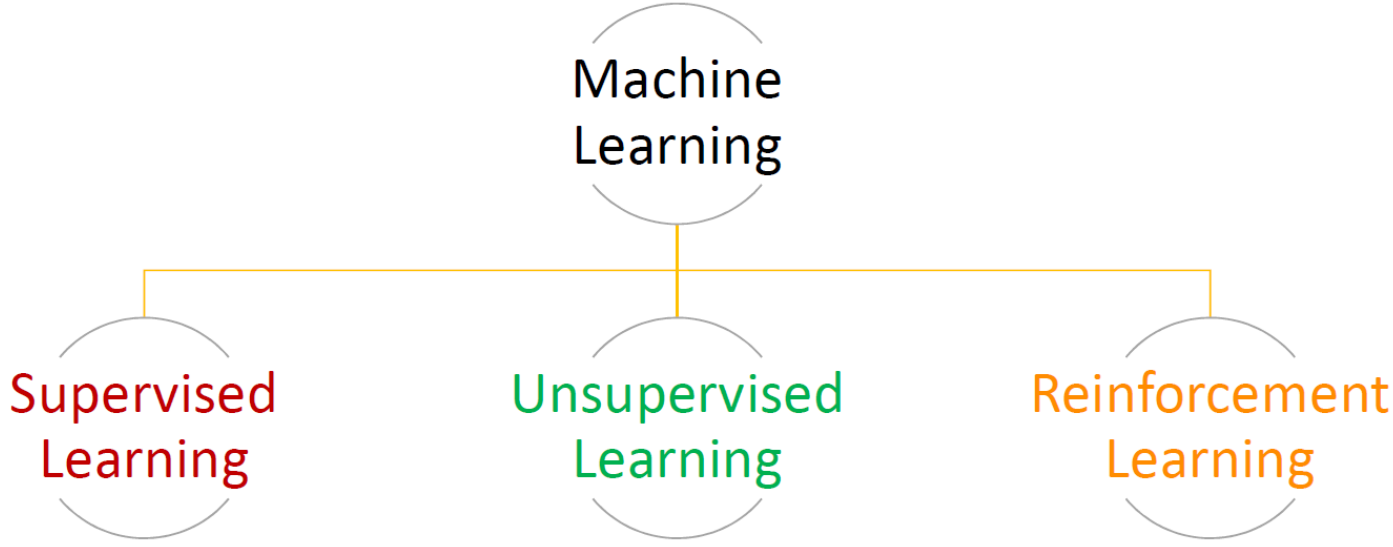
- Data Type
- Control Flow
- Pandas

PRACTICAL 2: DATA VISUALIZATION (SELF STUDY)

- Line Graph
- Pie Chart
- Bar Chart
- Scatter Plot
- Histogram
- Boxplot

Machine Learning Algorithms

Types of Machine Learning



Supervised vs. unsupervised

DATA DEFINITIONS: LABELED AND UNLABELED DATA

No Target (labels) **Variables or Features**

We do not know what type of iris flowers they are.

There is no labels.

?

	sepal length	sepal width	petal length	petal width
45	5.100	3.800	1.900	0.400
46	4.800	3.000	1.400	0.300
47	5.100	3.800	1.600	0.200
48	4.600	3.200	1.400	0.200
49	5.300	3.700	1.500	0.200
50	5.000	3.300	1.400	0.200
51	7.000	3.200	4.700	1.400
52	6.400	3.200	4.500	1.500
53	6.900	3.100	4.900	1.500

Target (labels) **Variables or Features**

We collected the data from a known group of iris flower types.

The iris column is the target or labels.

Hence labeled data.

Classification or Prediction techniques are used in these cases.

	iris	sepal length	sepal width	petal length	petal width
43	Iris-setosa	4.400	3.200	1.300	0.200
44	Iris-setosa	5.000	3.500	1.600	0.600
45	Iris-setosa	5.100	3.800	1.900	0.400
46	Iris-setosa	4.800	3.000	1.400	0.300
47	Iris-setosa	5.100	3.800	1.600	0.200
48	Iris-setosa	4.600	3.200	1.400	0.200
49	Iris-setosa	5.300	3.700	1.500	0.200
50	Iris-setosa	5.000	3.300	1.400	0.200
51	Iris-versicolor	7.000	3.200	4.700	1.400
52	Iris-versicolor	6.400	3.200	4.500	1.500
53	Iris-versicolor	6.900	3.100	4.900	1.500
54	Iris-versicolor	5.500	2.300	4.000	1.300
55	Iris-versicolor	6.500	2.800	4.600	1.500

Classification vs. Regression in Supervised Learning

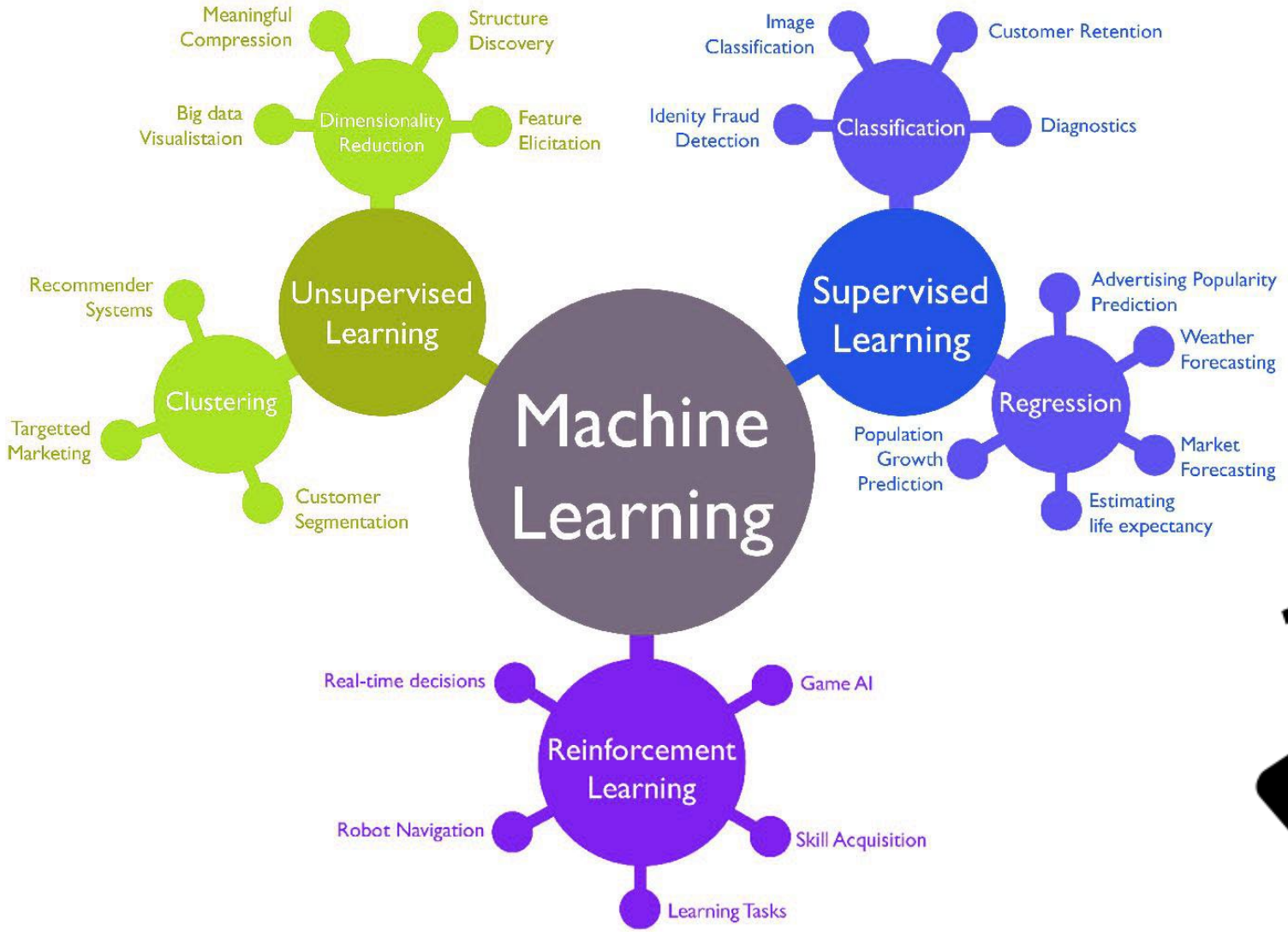
Regression

month	town	flat_type	block	street_name	storey_range	floorAreaSqm	resale_price
2012-09	CHOA CHU KANG	4 ROOM	119	TECK WHYE LANE	04 TO 06	104	**400000**
2013-10	JURONG WEST	3 ROOM	510	JURONG WEST ST 52	13 TO 15	74	**375000**
2013-03	JURONG EAST	5 ROOM	284	TOH GUAN RD	07 TO 09	120	**655000**

Classification

month	town	flat_type	block	street_name	storey_range	floorAreaSqm	resale_price
2012-09	CHOA CHU KANG	4 ROOM	119	TECK WHYE LANE	04 TO 06	104	**Medium**
2013-10	JURONG WEST	3 ROOM	510	JURONG WEST ST 52	13 TO 15	74	**Low**
2013-03	JURONG EAST	5 ROOM	284	TOH GUAN RD	07 TO 09	120	**High**

Machine Learning Approaches



HTTPS://WWW.WORDSTREAM.COM/BLOG/WS/2017/07/28/MACHINE-LEARNING-APPLICATIONS, ABDUL WAHID

ML ALGORITHMS TO REVIEW

Unsupervised Learning

- Clustering: K-Means

Supervised Learning

- Classification: K-Nearest Neighbors
- Regression: Linear and Polynomial
- Support Vector Machines
- Artificial Neural Networks

Deep Learning

K-Means Clustering

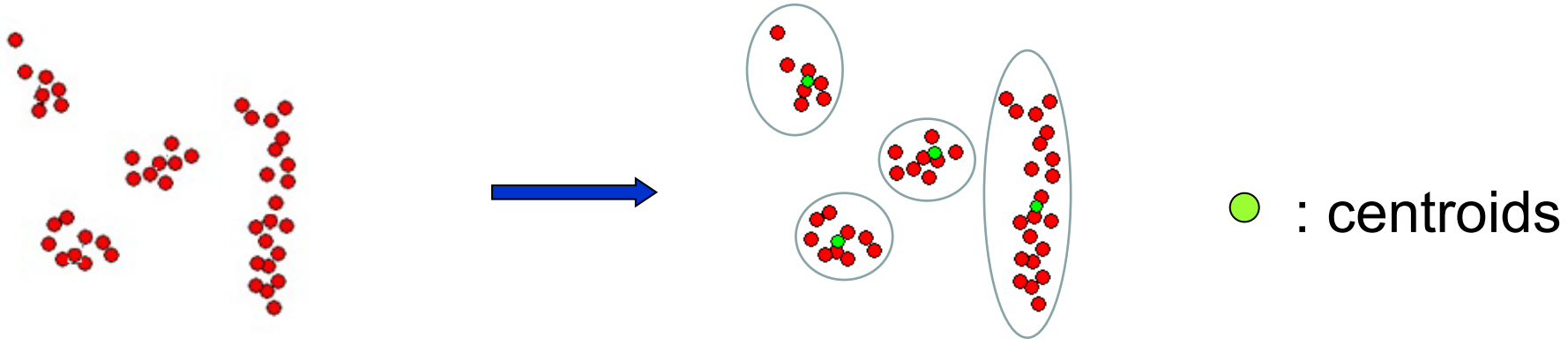
CLUSTERING ANALYSIS

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

Cluster Detection

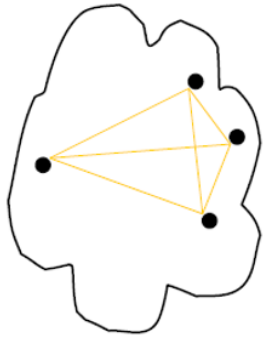
- Cluster: a group of objects known as members
- The centre of a cluster is known as the *centroid*
- Members of a cluster are similar to each other
- Members of different clusters are different
- Clustering is a process of discovering clusters



K-MEANS CLUSTERING

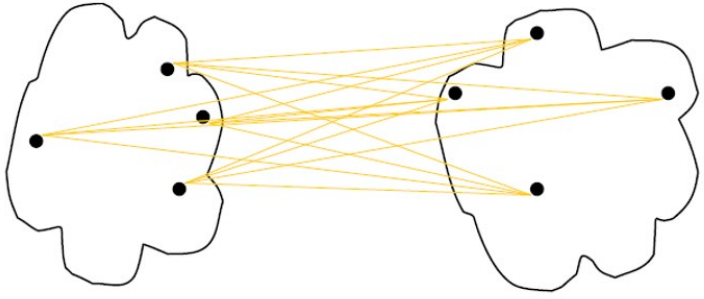
K-means clustering is one of the simplest and popular unsupervised machine learning algorithms.

K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.



It will try to

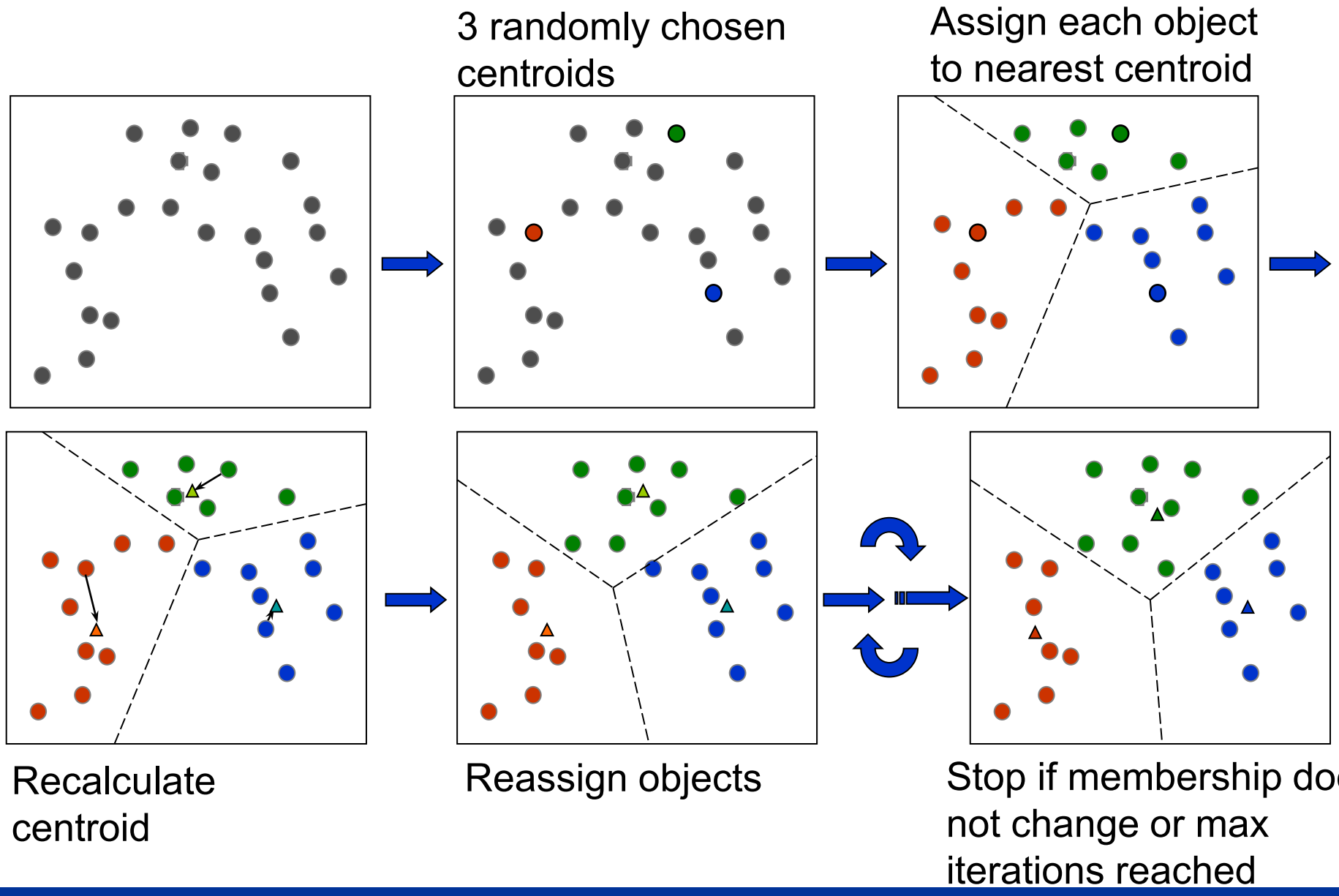
- minimize intra-cluster distances
- maximize inter-cluster distances



K-MEANS ALGORITHM

1. Define the number of clusters (k)
2. Choose k data objects randomly to serve as the initial centroids for the k clusters
3. Assign each data object to the cluster represented by its nearest centroid
4. Find a new centroid for each cluster by calculating the mean/center of its members
5. Undo the memberships of all data objects. Go back to Step 3 and repeat the process until cluster membership no longer changes or a maximum number of iterations is reached.

K-means algorithm (an illustration)



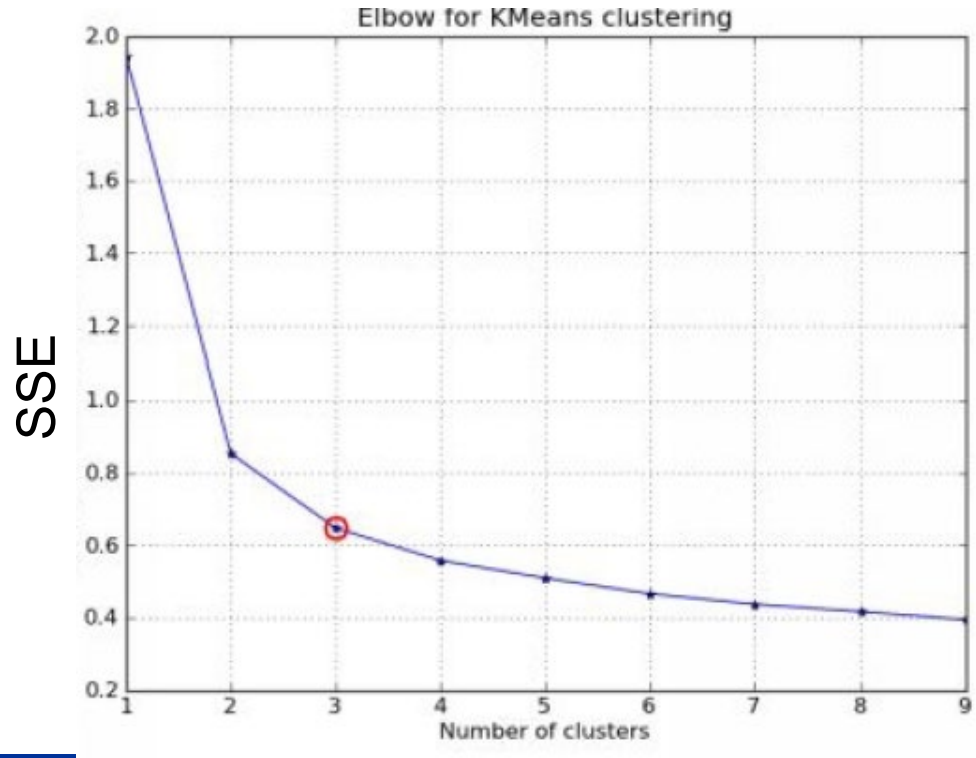
EVALUATE THE MODEL

Sum of Squared Errors (SSE)

SSE measures the distances from each data object to the cluster centroid, take a squared value on the distances and then sum them up together.

Elbow Method:

We can see the larger K the lower SSE, but beyond certain point (e.g. K=3) adding more clusters will not reduce SSE significant, therefore K=3 is the desired K.



K-MEANS STRENGTHS & WEAKNESSES

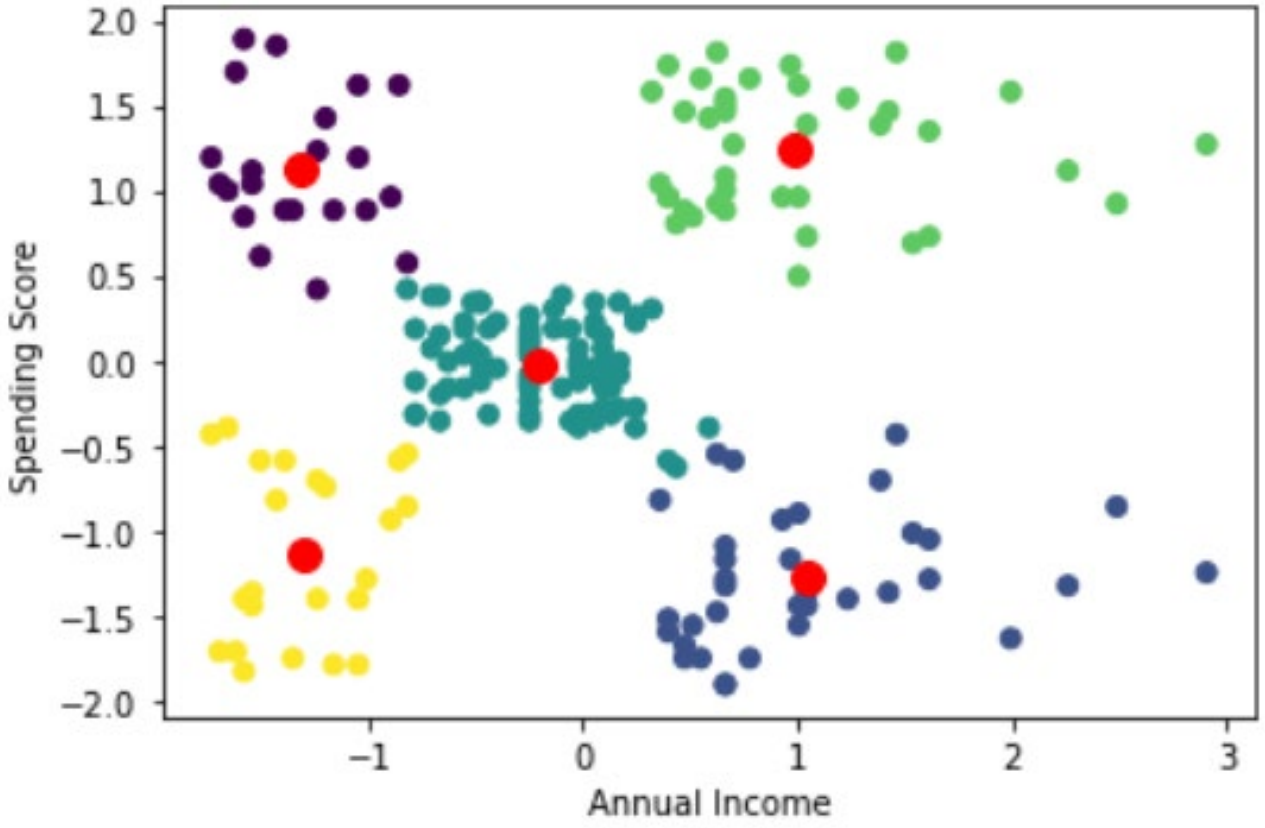
Strengths

- Simple and easy to implement
- Quite efficient

Weaknesses

- Need to specify the value of K, but we may not know what the value should be beforehand
- Sensitive to the choice of initial K centroids: clusters tend to converge to a local optimum solution
- Sensitive to noise

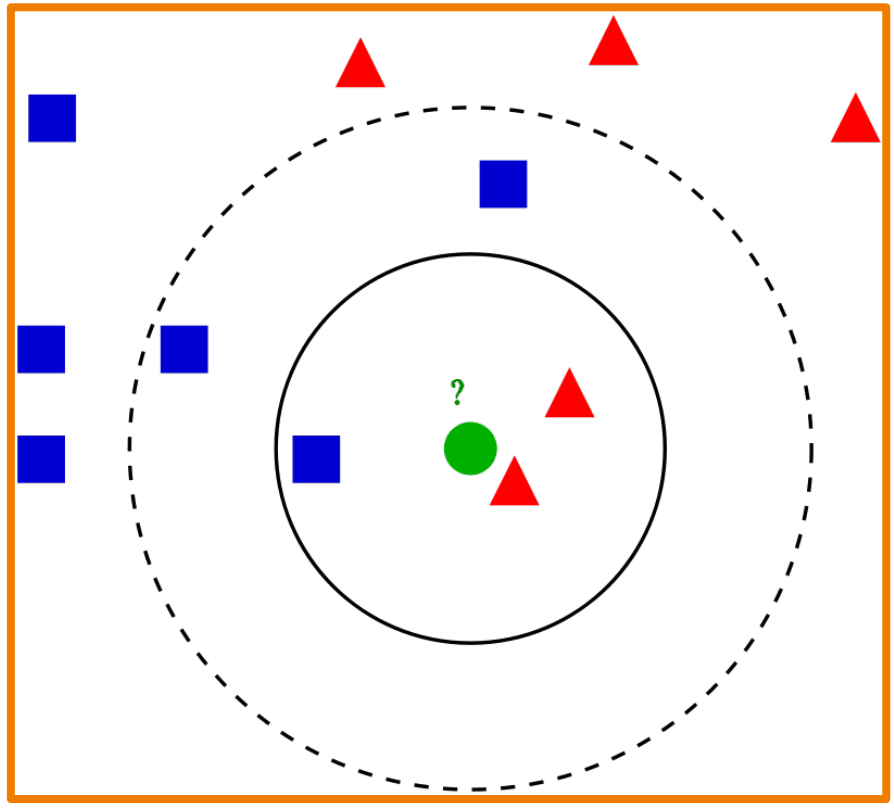
PRACTICAL 3: K-MEANS CLUSTERING



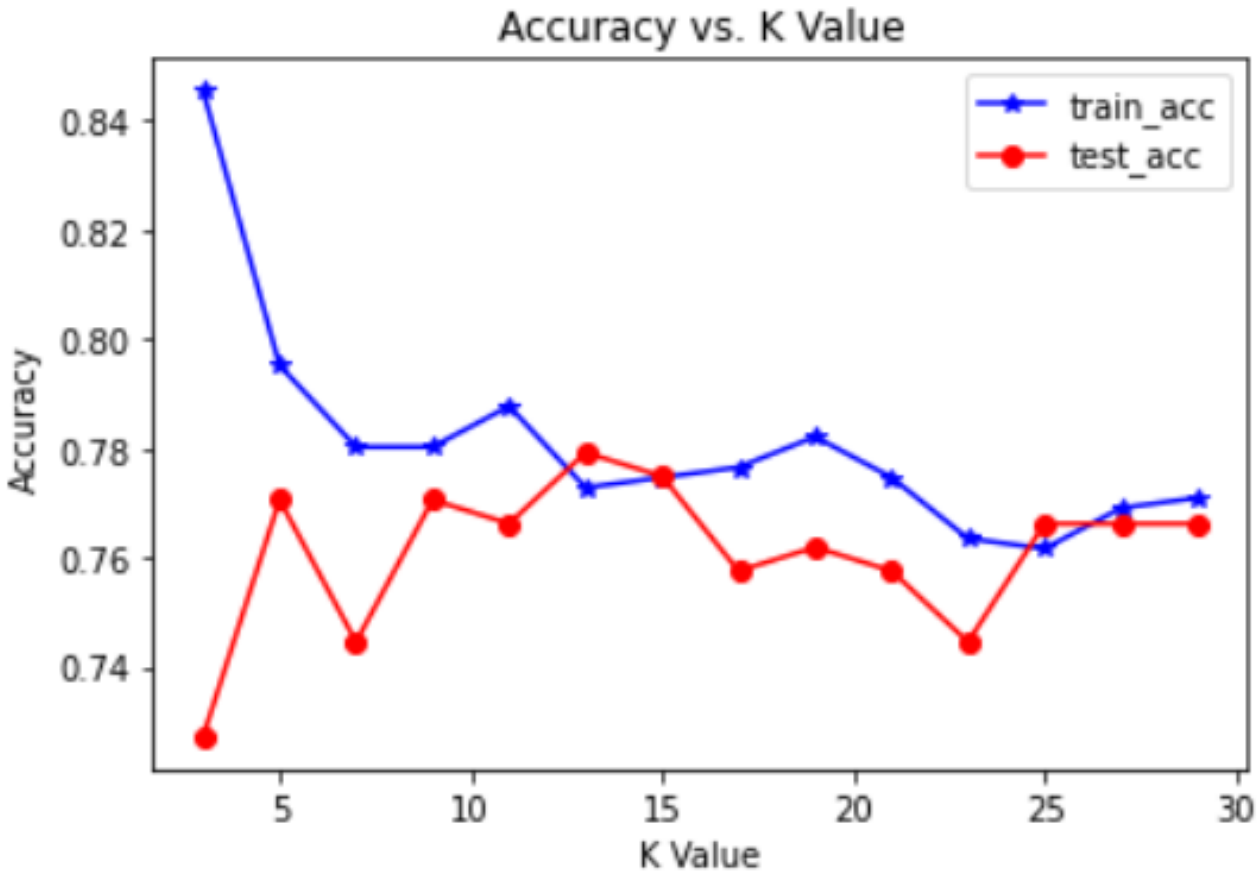
K Nearest Neighbour for Classification

K-NEAREST NEIGHBORS

K-nearest-neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity/distance measure between the new case and k nearest samples/cases. It measures the distances in the feature space.



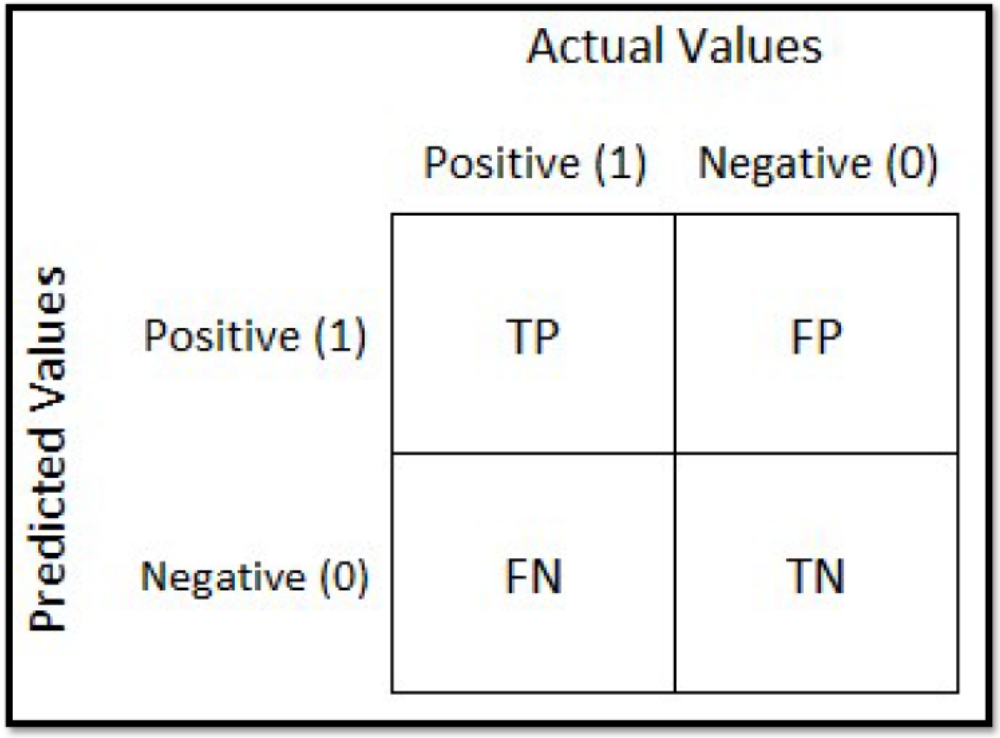
PRACTICAL 4: KNN



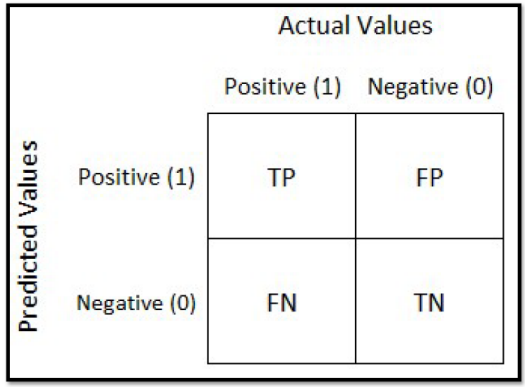
EVALUATE CLASSIFICATION MODEL

Confusion Matrix

- TP: True Positive
- FP: False Positive
- TN: True Negative
- FN: False Negative



EVALUATE CLASSIFICATION MODEL



An Example: Imagine a Machine Learning model is used to classify patients with (P) or without (N) diabetes

- Accuracy = (TP + TN) / (TP + FP + FN +TN)
- Precision = TP / (TP + FP)
  - the ratio of P patients correctly classified by the model out of all P patients classified by the model
- Recall = TP / (TP + FN)
  - the ratio of P patients correctly classified by the model out of all the actual P patients
- F1 score = 2 x (Precision x Recall) / (Precision + Recall)
  - Hamonic mean of precision and recall

Regression

REGRESSION

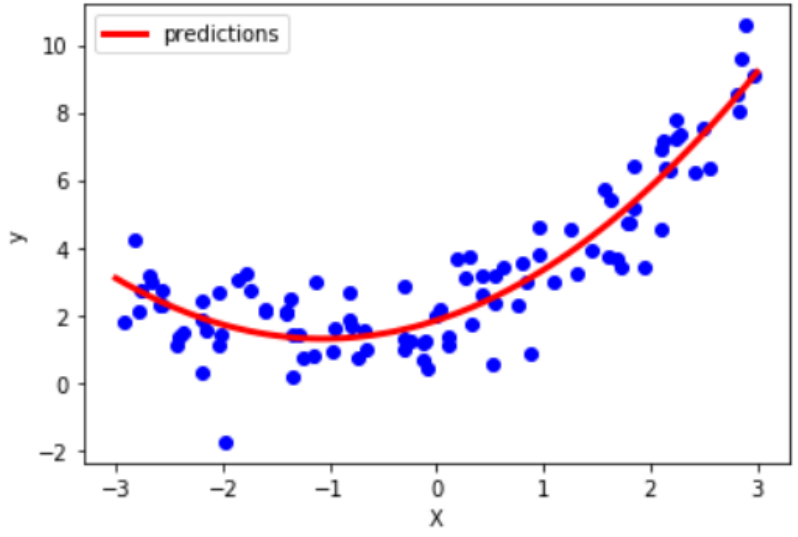
- Regression analysis is a set of statistical methods applied to estimate the relationship among variables.
- Regression is used when the target outputs are continuous, i.e. numerical outputs are expected.
- It is used to determine if a correlation exists between variables.
- It is important to note that correlation does not imply causation.

REGRESSION

Regression analysis is a statistical approach for understanding the relationship between

- input variables (X) and output variable (y)
- independent variables (X) and dependent variable (y)

$$y=f(x)$$



TYPES OF REGRESSION

Simple Linear Regression

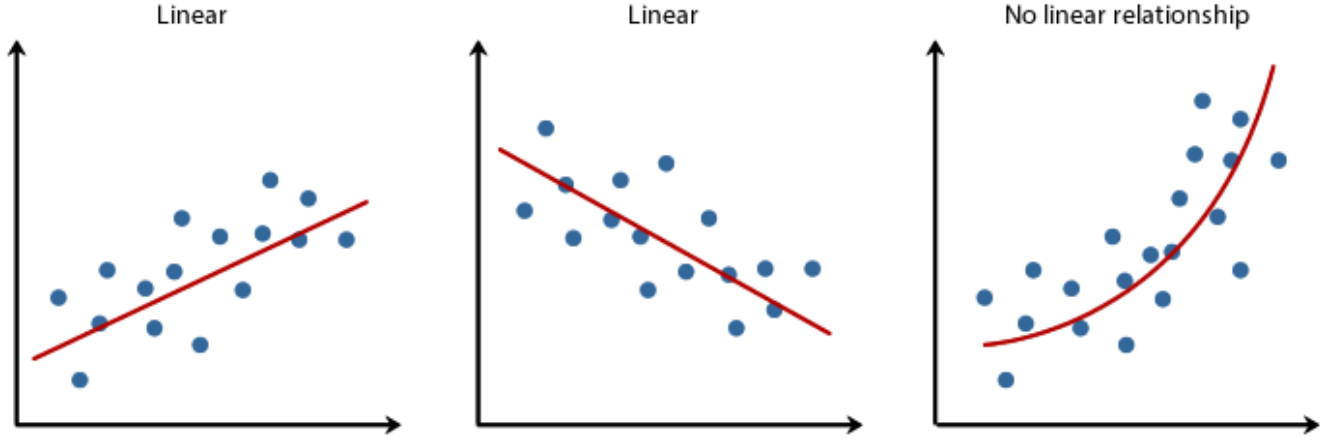
$$y = b_0 + b_1 x_1$$

Multiple Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_n x_n$$

Polynomial Linear Regression

$$y = b_0 + b_1 x_1 + b_2 x_1^2 + \ldots + b_n x_1^n$$



EVALUATE REGRESSION MODEL

NUS National University of Singapore | School of Computing

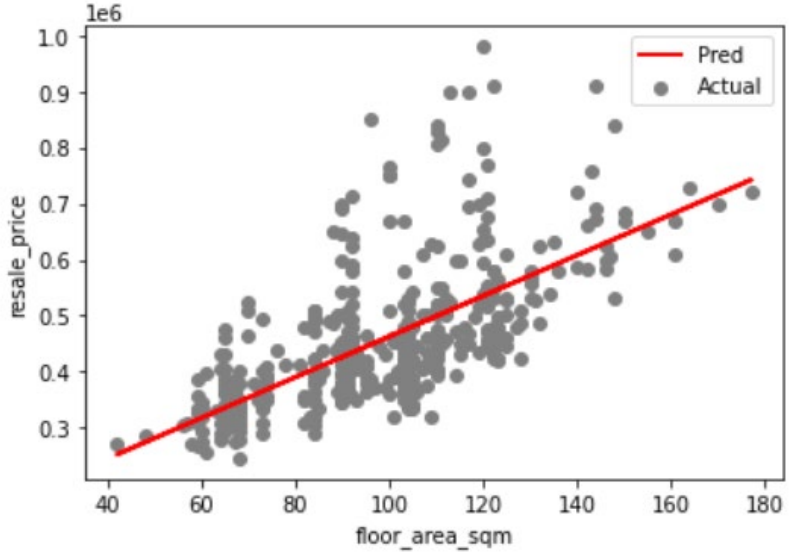
Errors: the lower the better

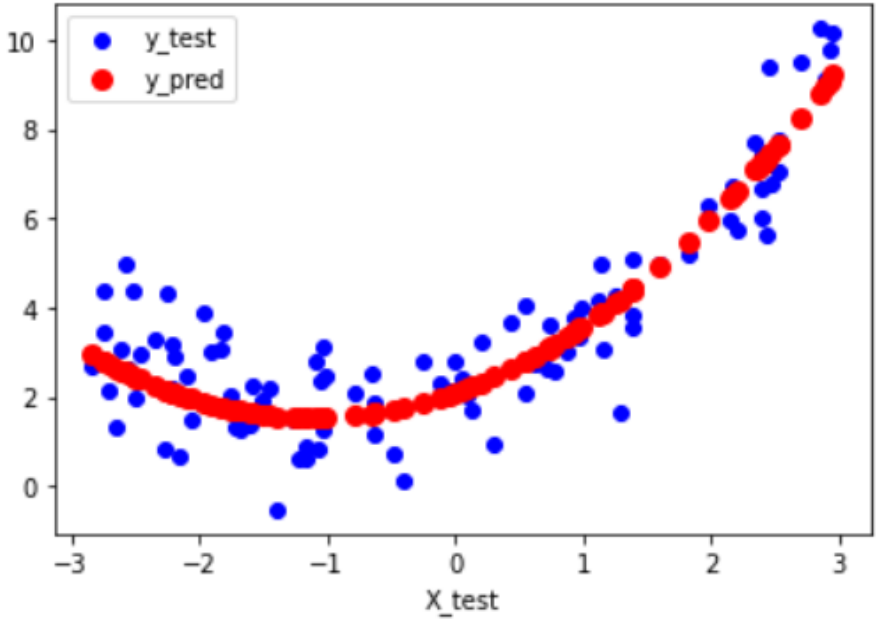
- Mean Squared Error (MSE) = $\Sigma(y_i - y'_i)^2 / n$
- Root Mean Squared Error (RMSE) = $\sqrt{MSE}$
- Mean Absolute Error (MAE) = $\Sigma |y_i - y'_i| / n$

R Squared Value (0 to 1): the higher the better

- the percentage of variance from the samples can be explained through the model

PRACTICAL 5: LINEAR REGRESSION AND POLYNOMIAL REGRESSION





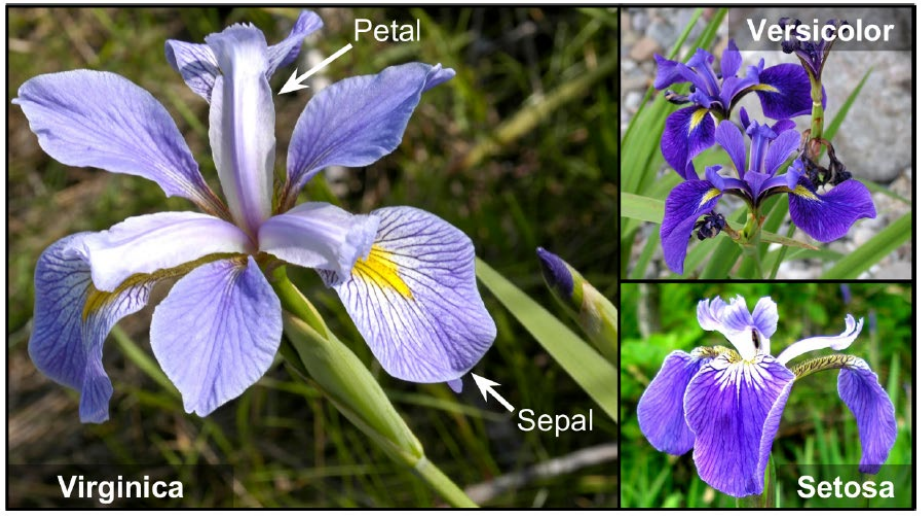
Support Vector Machine (SVM)

SUPPORT VECTOR MACHINES

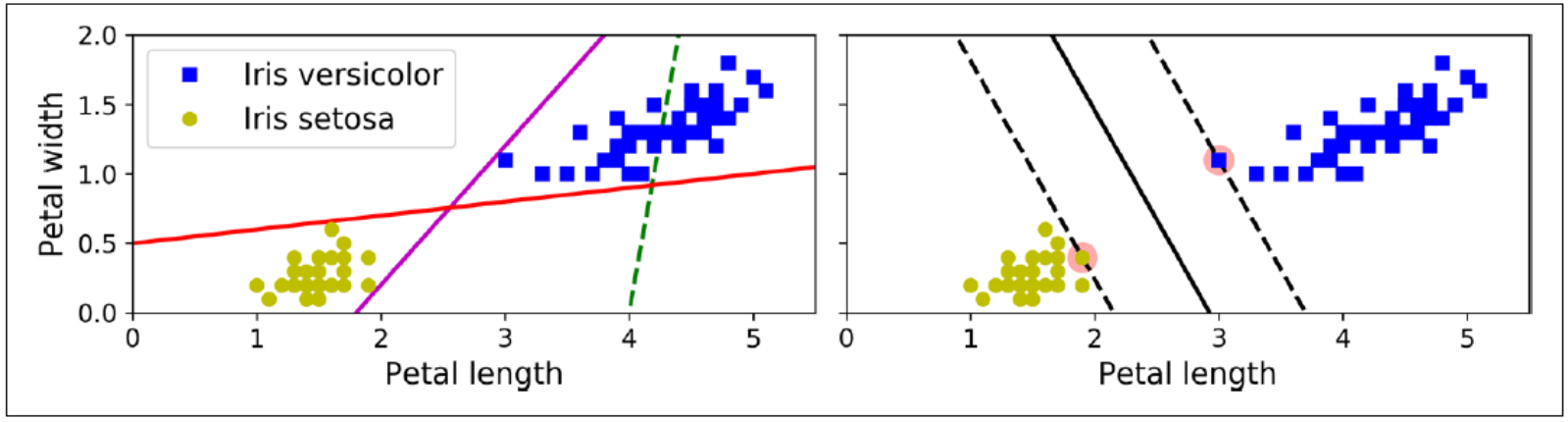
- One of the most popular machine learning model, powerful and versatile

- Capable of performing linear/non-linear, classification/regression tasks but particularly suited for classification tasks

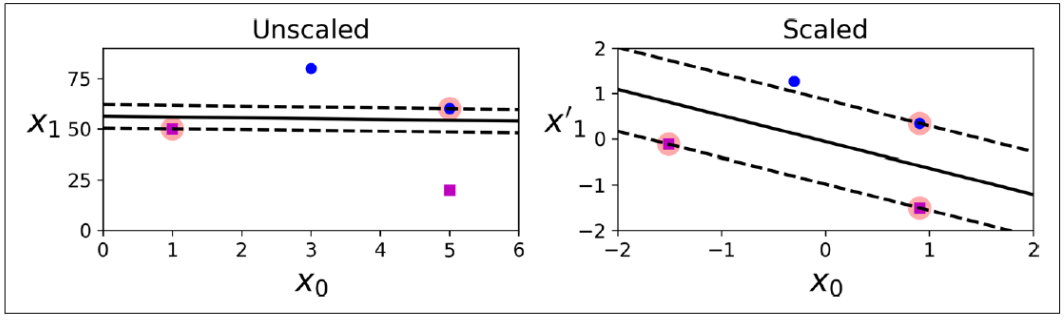
- SVM Classification:
  - Linear SVM Model
  - Kernel Trick
  - Nonlinear SVM Model



Linear SVM Classification

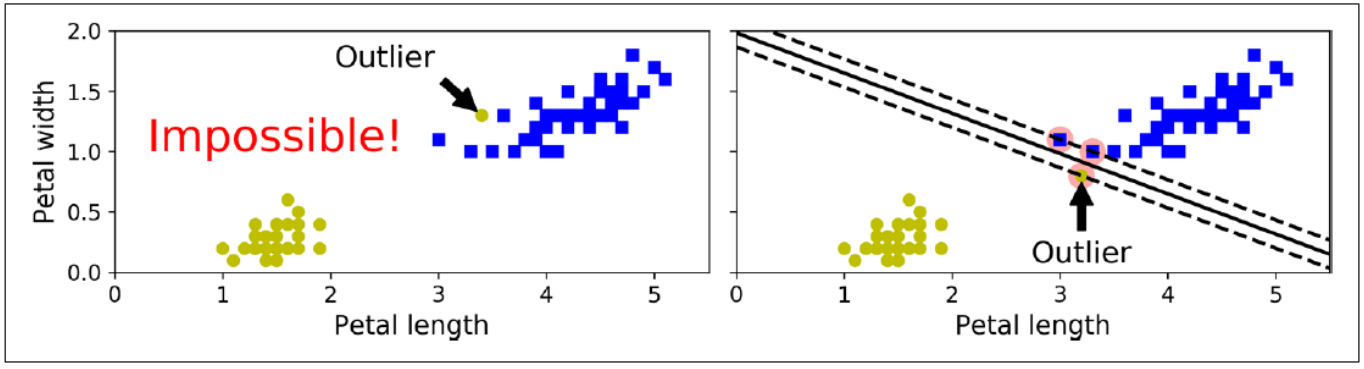


- Fitting the widest possible street between the two classes (large margin classification)
- All instances must be off the street and on the right side (i.e. No Margin Violations)??



Sensitive to Feature Scales

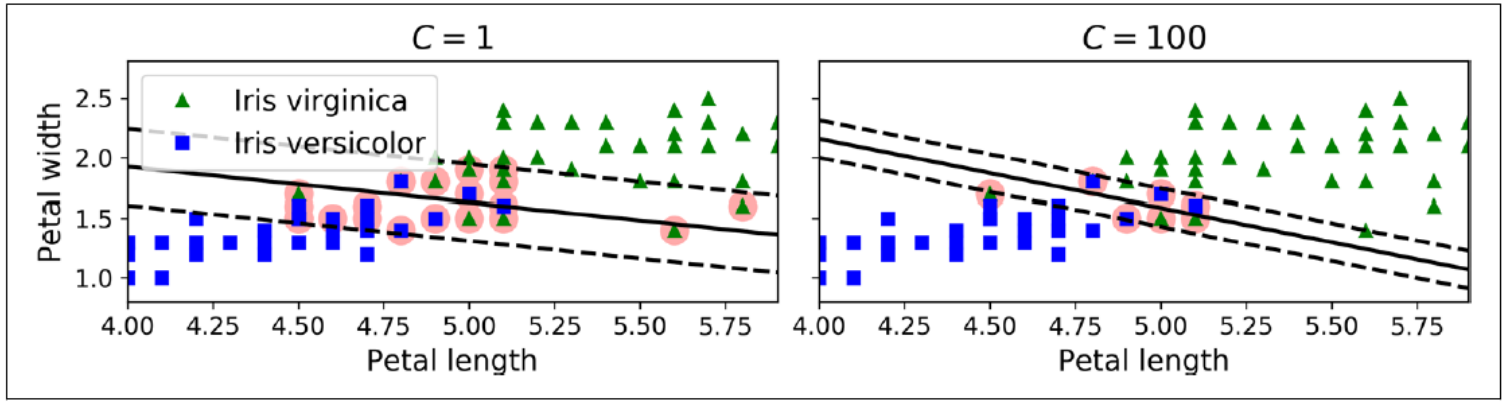
Use C to control the margin violations



Hard Margin Classification

- no margin violations
- sensitive to outliers

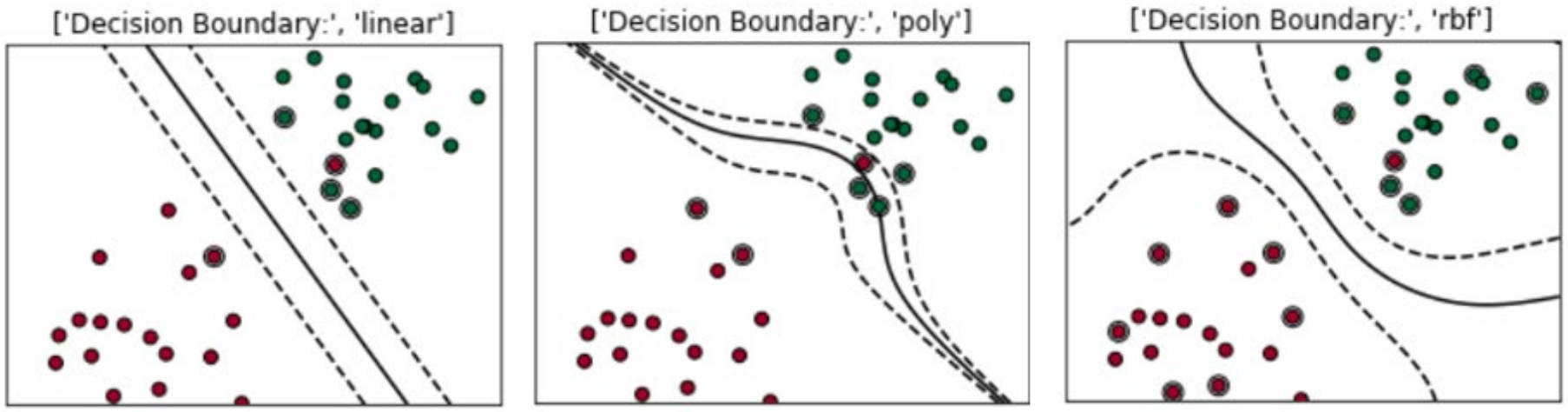
Soft Margin Classification



Large Margin Violations (wide street)

Fewer Margin Violations (narrow street)

SVM KERNEL TRICK



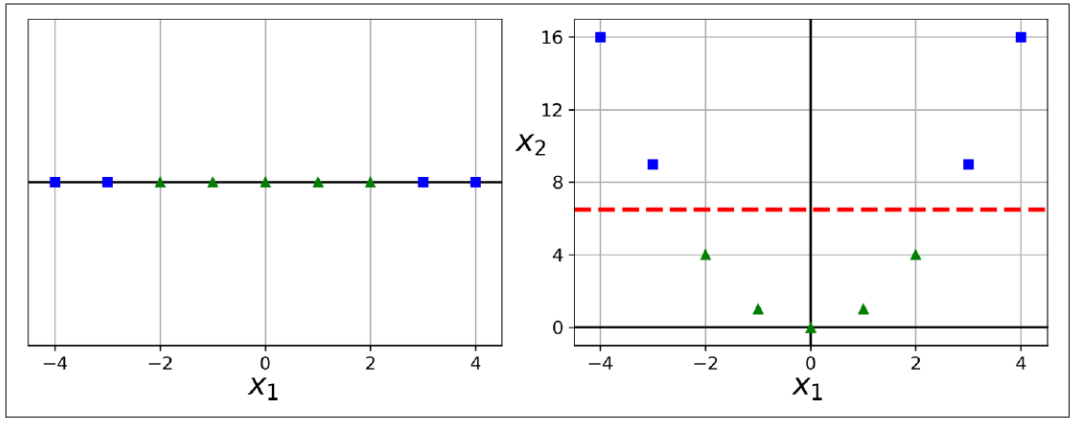
- We can choose ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’, ‘precomputed’ or a callable as our kernel/transformation.
- RBF = Redial Basis Function

HTTPS://TOWARDSDATASCIENCE.COM/SUPPORT-VECTOR-MACHINE-SIMPLY-EXPLAINED-FEE28EBA5496

Kernel Tricks

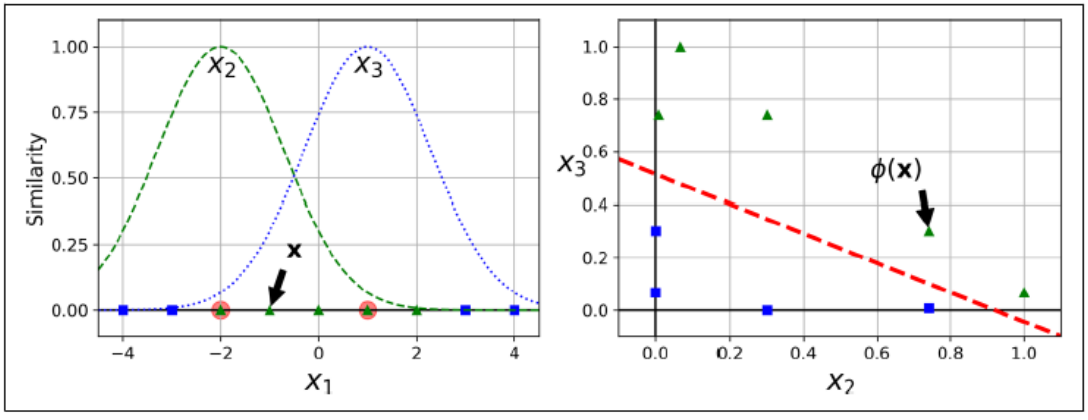
Feature Transformation using Polynomial Function

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_2 x^2 + a_1 x + a_0$$



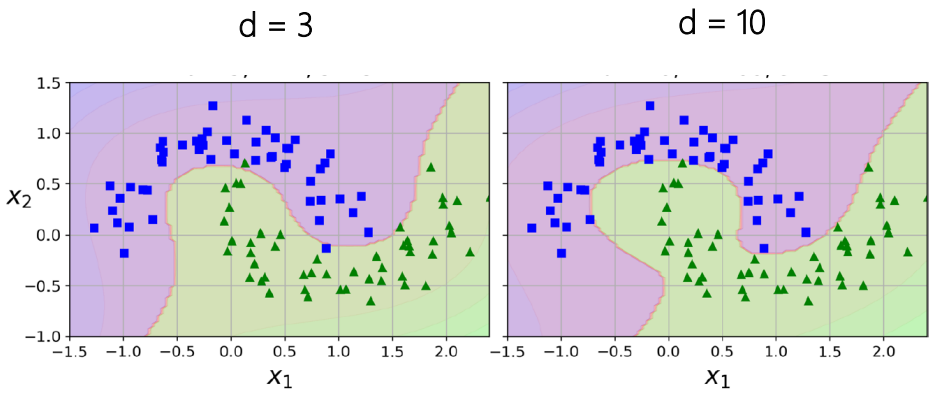
Feature Transformation using Gaussian Radial Basis Function (RBF)

$$\phi_\gamma(\mathbf{x}, \ell) = \exp\left(-\gamma \| \mathbf{x} - \ell \|^2\right)$$

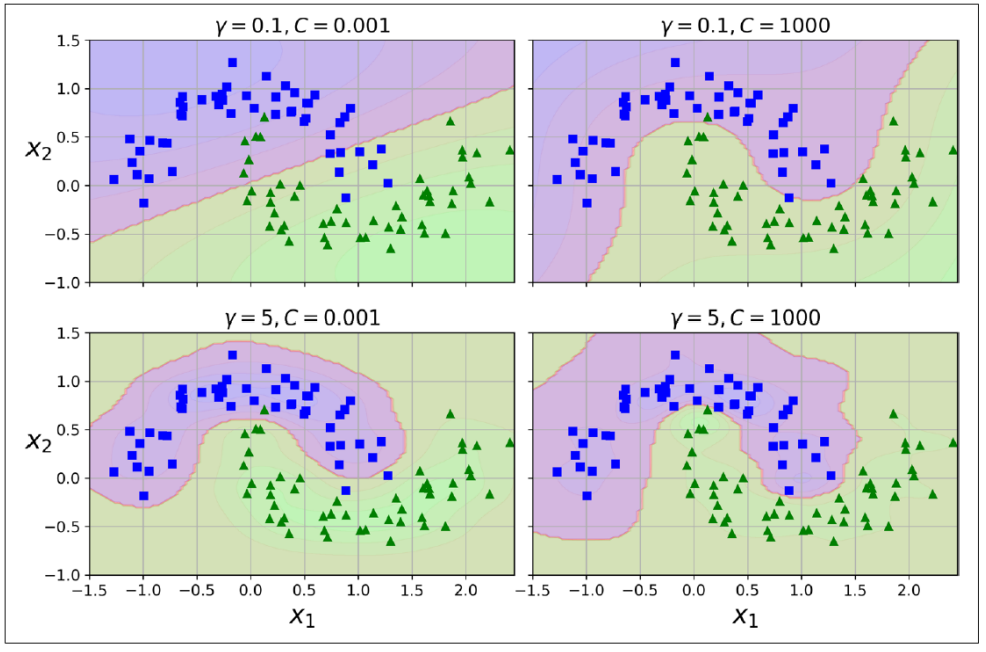


Nonlinear SVM Classification

Polynomial Kernel

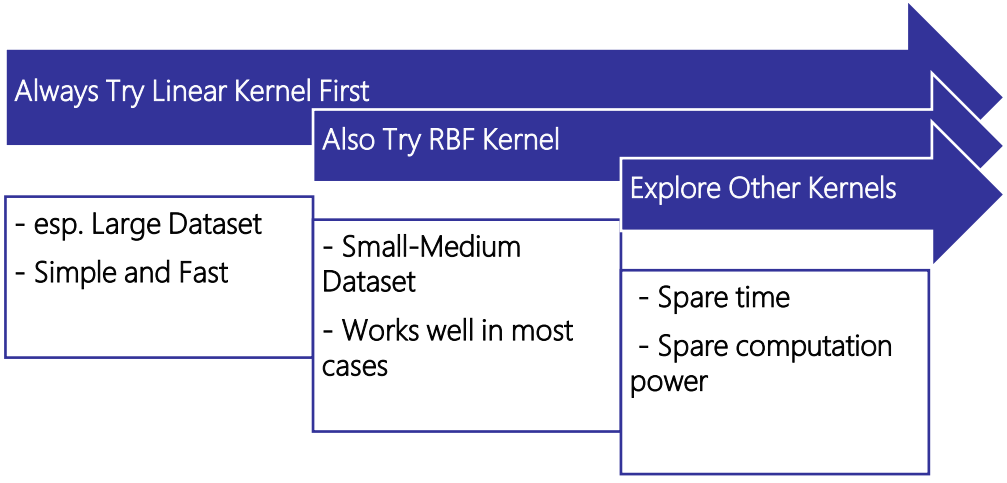


RBF Kernel

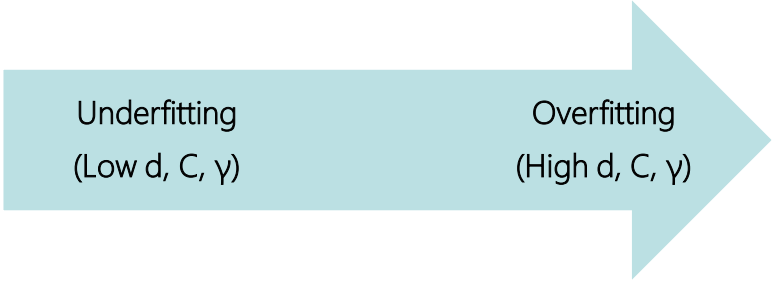


Tuning SVM Models

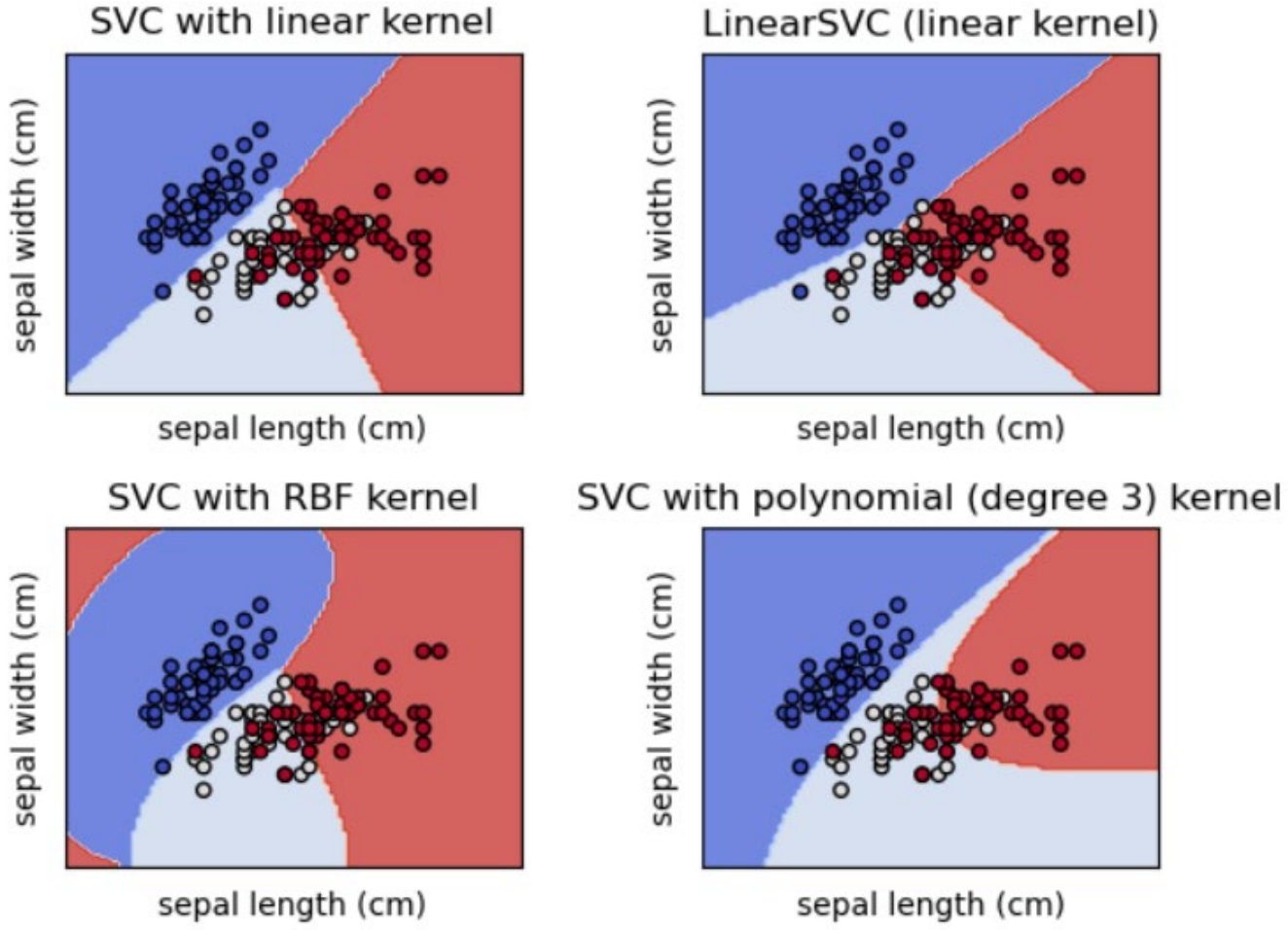
Choose from different kernels



Tuning Hyperparameters



PRACTICAL 6: SVM



Artificial Neural Networks (ANN)

ARTIFICIAL NEURAL NETWORKS

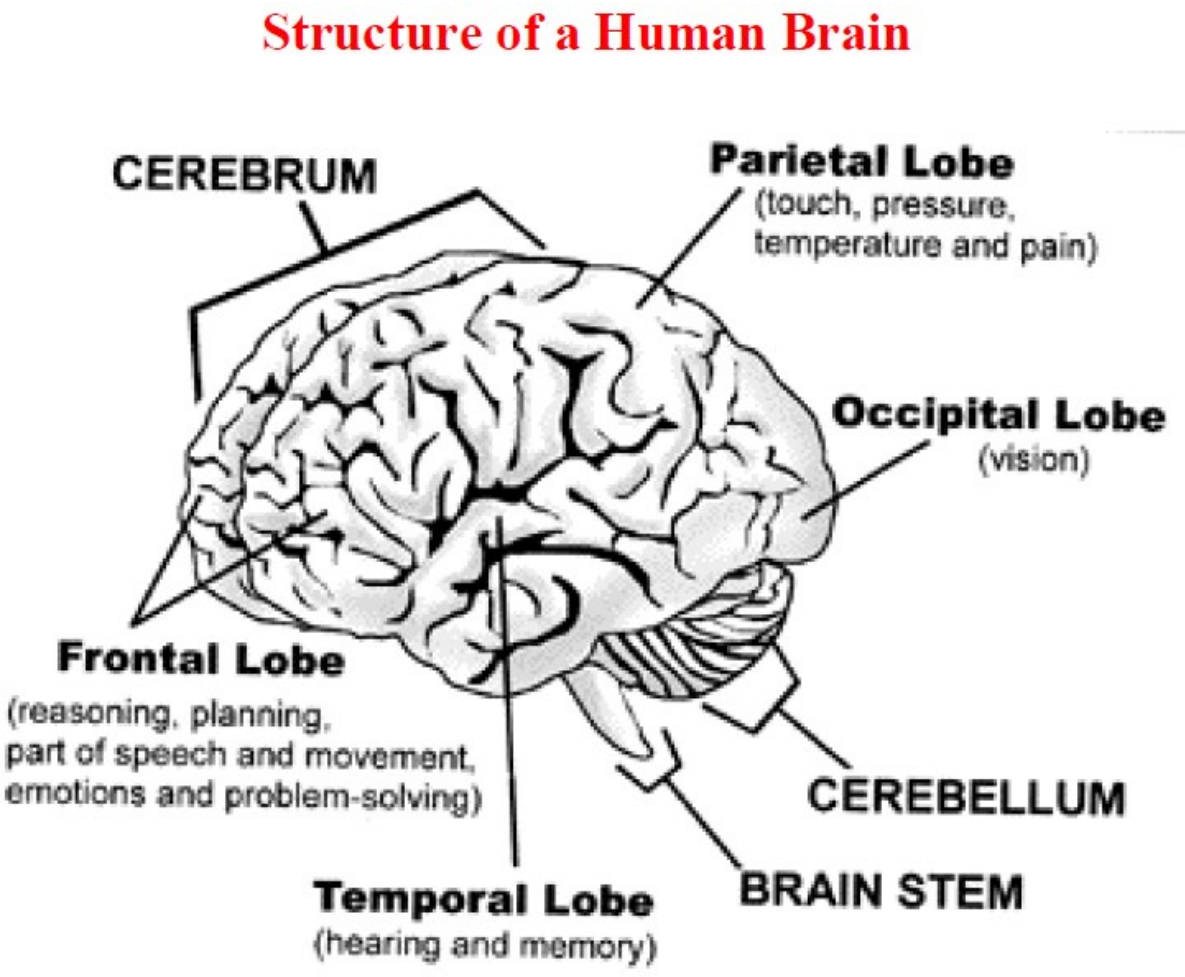
Neural Networks (NNs) are networks of neurons, as found in real (i.e. biological) brains.

Artificial Neurons are crude approximations of the neurons found in brains. They may be physical devices, or purely mathematical/ software constructs.

Artificial Neural Networks (ANNs) are networks of Artificial Neurons, and hence constitute crude approximations to parts of real brains.

From a practical point of view, an ANN is just a parallel computational system consisting of many simple processing elements connected together in a specific way in order to perform a particular task.

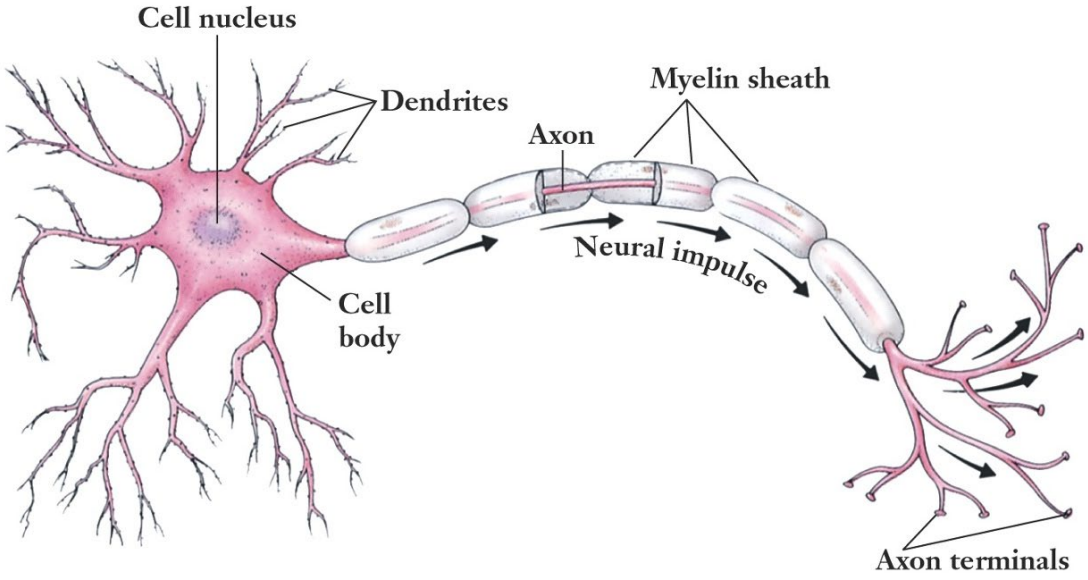
ARTIFICIAL NEURAL NETWORKS

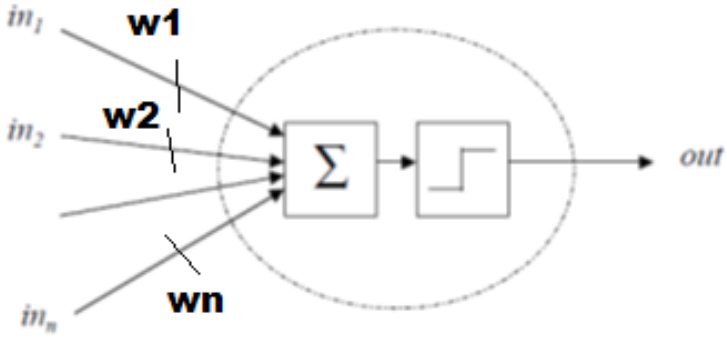


human brain, With 100 billion cells, each with 1,000 to 10,000 synapses, the neocortex makes roughly 500 trillion connections and contains 300 million feet of wiring packed with other tissue into a one-and-a-half-quart volume in the brain.

WWW.NYTIMES.COM

ARTIFICIAL NEURAL NETWORKS





A biological neuron, and its model. Weighted summation of inputs and an activation function.

HTTPS://WWW.THINGLINK.COM/SCENE/

ARTIFICIAL NEURAL NETWORKS, WHY?

They are extremely powerful computational devices (Turing Equivalent Universal Computers).

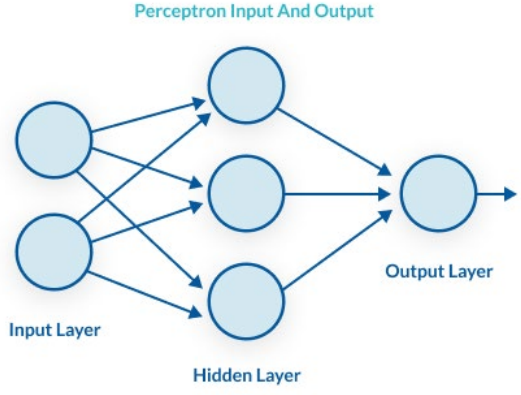
Massive parallelism makes them very efficient.

They can learn and generalize from training data – so there is no need for a subtle design.

They are particularly fault tolerant, and noise tolerant

In principle, they can do anything a symbolic/logic system can do, and more.

They are very good in dealing with semi structured, and unstructured data (text, image, …)



HTTPS://WWW.THINGLINK.COM/SCENE/

ARTIFICIAL NEURAL NETWORKS, HISTORY

NUS National University of Singapore | School of Computing

Date	Topic
1943	McCulloch and Pitts proposed the McCulloch-Pitts neuron model
1949	Hebb published his book The Organization of Behavior, in which the Hebbian learning rule was proposed.
1958	Rosenblatt introduced the simple single layer networks now called Perceptrons.
1969	Minsky and Papert's book Perceptrons demonstrated the limitation of single layer perceptrons, and almost the whole field went into hibernation.
1982	Hopfield published a series of papers on Hopfield networks
1986	The Back-Propagation learning algorithm for Multi-Layer Perceptrons was rediscovered and the whole field took off again. A big breakthrough
1987	Convolutional neural networks were presented at the Neural Information Processing Workshop
1990s	The sub-field of Radial Basis Function Networks was developed.
1997	Long short term memory, LSTM
2000s	The power of Ensembles of Neural Networks and Support Vector Machines becomes apparent.
2006	Deep Neural Networks, Deep Learning, A big breakthrough
2014	Generative Adversarial Network (GAN)

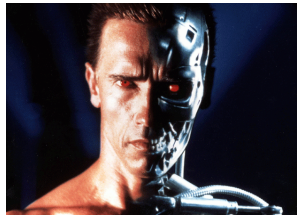
ANN APPLICATIONS

Brain modelling

Models of human development – help children with developmental problems

Simulations of adult performance – aid our understanding of how the brain works

Neuropsychological models – suggest remedial actions for brain damaged patients



Real world applications

Financial modelling – predicting stocks, shares, currency exchange rates

Other time series prediction – climate, weather, airline marketing tactician

Computer games – intelligent agents, backgammon, first person shooters

Control systems – autonomous adaptable robots, microwave controllers

Pattern recognition – speech recognition, hand-writing recognition, sonar signals

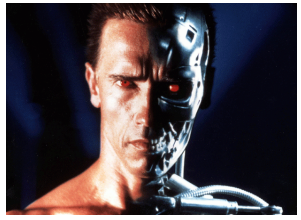
Data analysis – data compression, data mining, PCA, GTM

Noise reduction – function approximation, ECG noise reduction

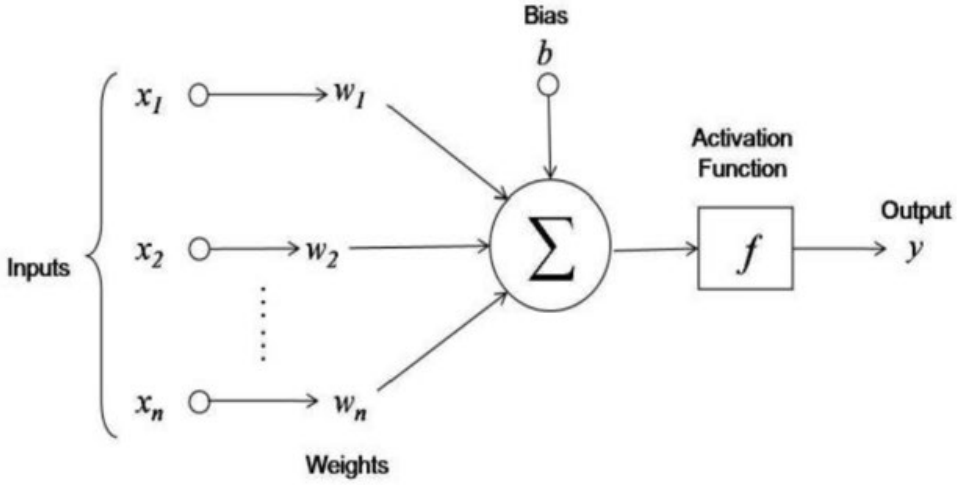
Bioinformatics – protein secondary structure, DNA sequencing

ANN APPLICATIONS, BUT GENERALLY, THEY ARE …

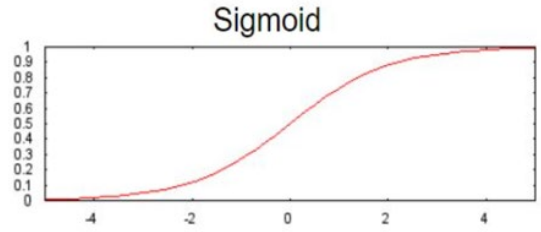
Function Estimation	Classification	Mapping	Decision Making
• Continuous variables • Prediction	• Discrete variables • Prediction • Novelty detection • Games	• Translation • Image restoration • Clustering • Scene analysis	• Translation • Robotics • Driverless Vehicles • Decision Theaters • Prescription



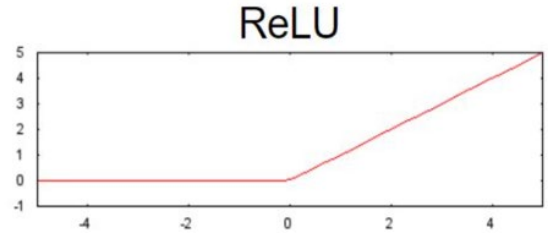
WHAT DOES A NEURON DO?



Activation Functions



$$F(x) = \frac{1}{1+e^{-x}}$$



$$F(x) = max(0, x)$$

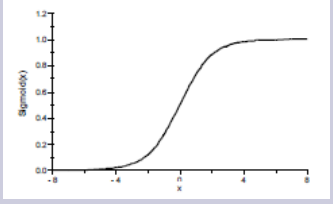
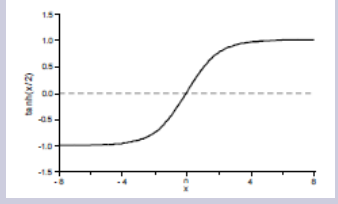
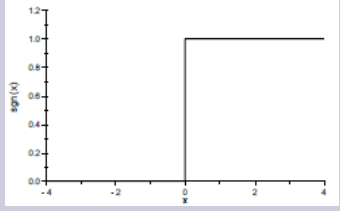
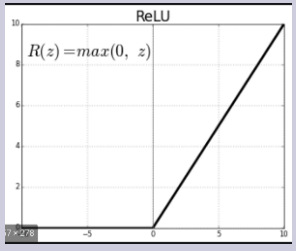
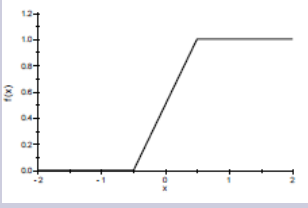
Step 1: sum of weights

$$w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Step 2: apply activation function

$$y = f(w_1x_1 + w_2x_2 + \cdots + w_nx_n + b)$$

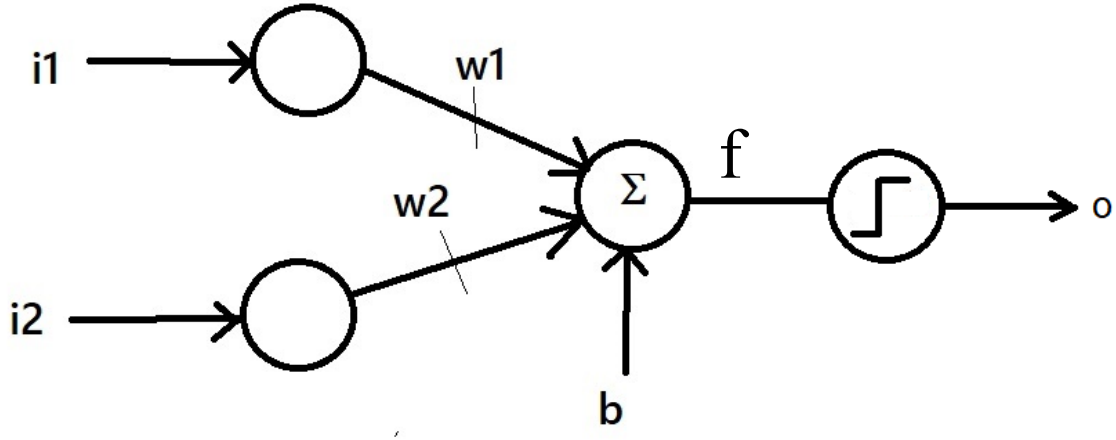
ACTIVATION FUNCTIONS

1	2	3	4	5
Sigmoid Function	**Hyperbolic Tangent**	**Step Function**	**ReLU Function**	**Piecewise Linear Func**
$\text{Sigmoid}(x) = \frac{1}{1+e^{-x}}$	$\tanh\left(\frac{x}{2}\right) = \frac{1-e^{-x}}{1+e^{-x}}$	$\text{sgn}(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$	$f(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$	$f(x) = \begin{cases} 1 & \text{if } x \geq 0.5 \\ x + 0.5 & \text{if } -0.5 \leq x \leq 0.5 \\ 0 & \text{if } x \leq 0.5 \end{cases}$
				

6 $\text{Sign Function},\ f(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$

1,2: invertible, differentiable;
1,4: popular;
4,5: partially differentiable,

PERCEPTRON/ SINGLE NEURON (AND/OR)

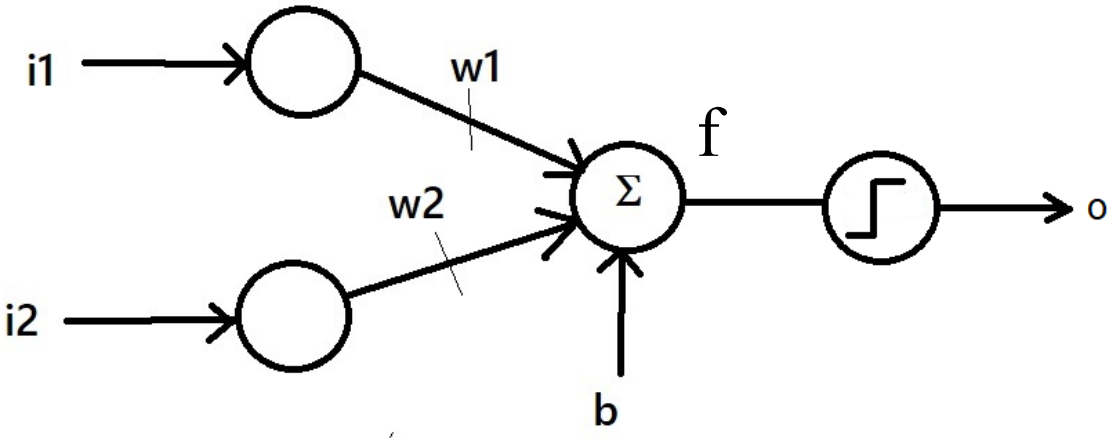


$$o = step(f) = \begin{cases} 1 & f \geq 0 \\ o & f < 0 \end{cases}$$

i1	i2	O=i1∨i2 (OR)	O=i1∧i2 (AND)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	1	1

$$or \begin{cases} w1 = 1 \\ w2 = 1 \\ b = -0.5 \end{cases} \qquad and \begin{cases} w1 = 1 \\ w2 = 1 \\ b = -1.5 \end{cases}$$

PERCEPTRON/ SINGLE NEURON (XOR)



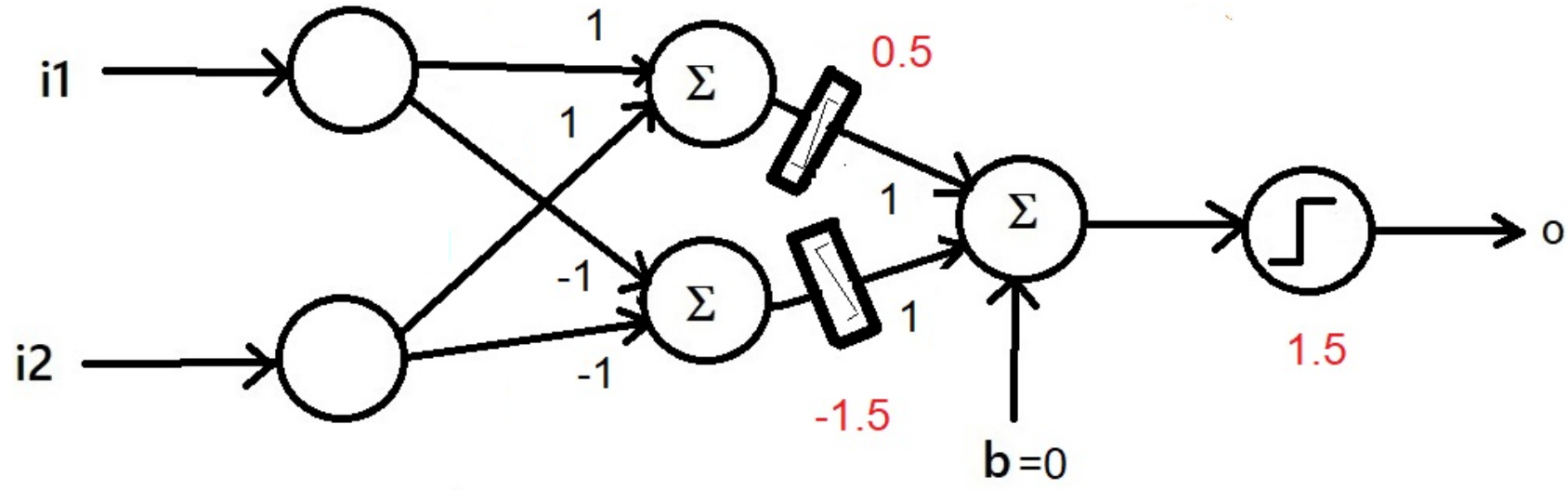
$$o = step(f) = \begin{cases} 1 & f \geq 0 \\ o & f < 0 \end{cases}$$

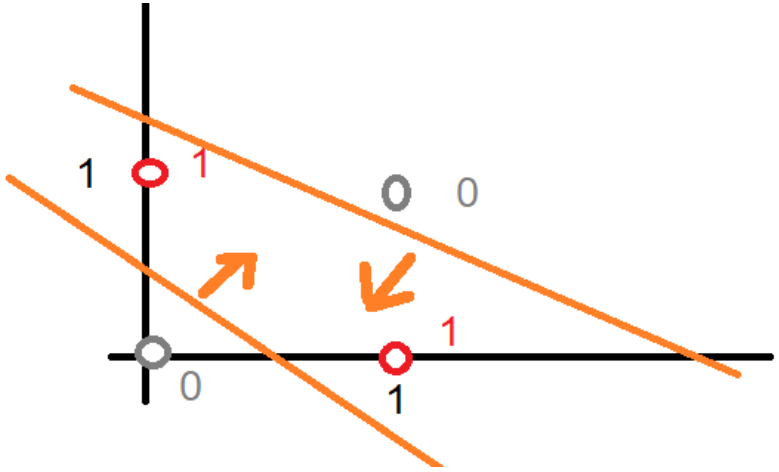
NO Solution !!!

To deal with such problems, a Perceptron needs a multi-neuron hidden layer with non-linear activation function.

i1	i2	O=i1 xor i2
0	0	0
0	1	1
1	0	1
1	1	0

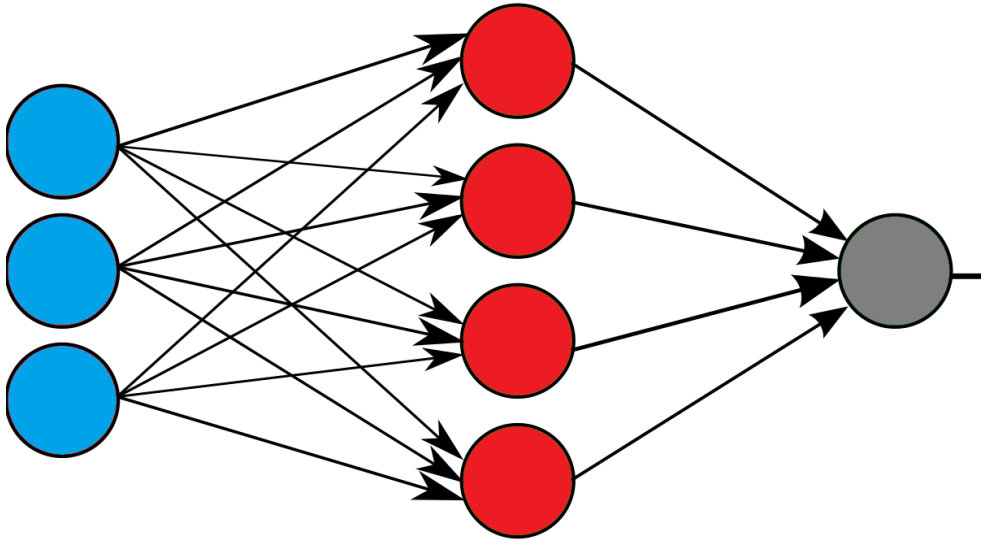
MULTI LAYERED PERCEPTRON (MLP) TO SOLVE XOR PROBLEM



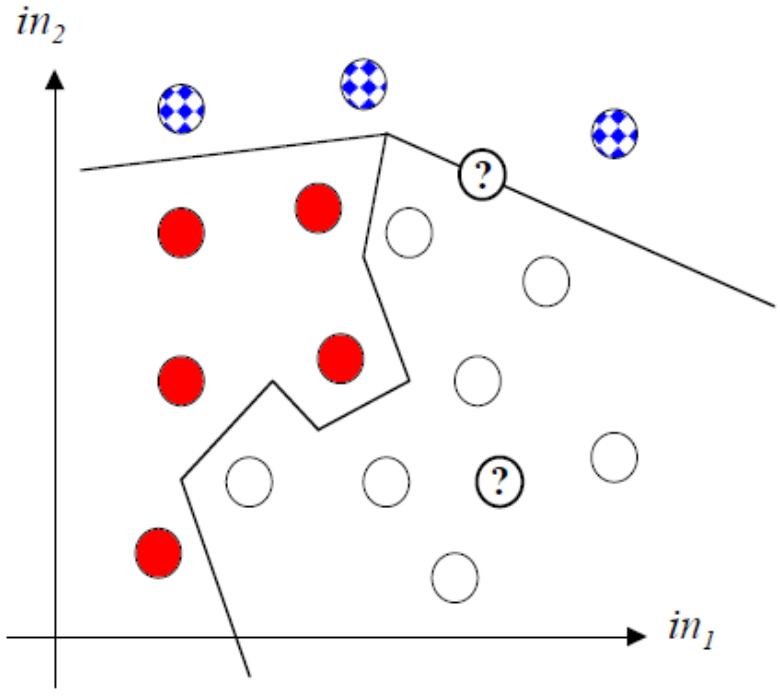


i1	i2	O=i1 xor i2
0	0	0
0	1	1
1	0	1
1	1	0

MULTILAYER PERCEPTRON (MLP)

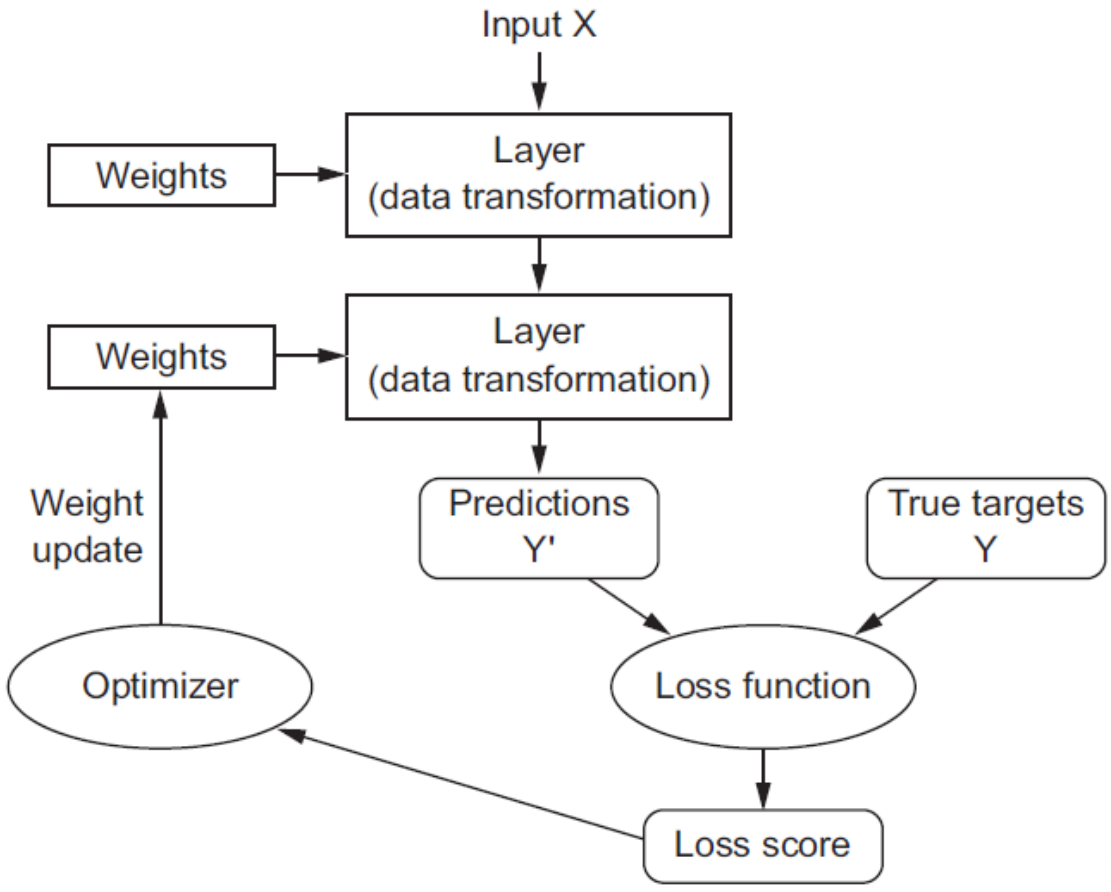


Input hidden layers output



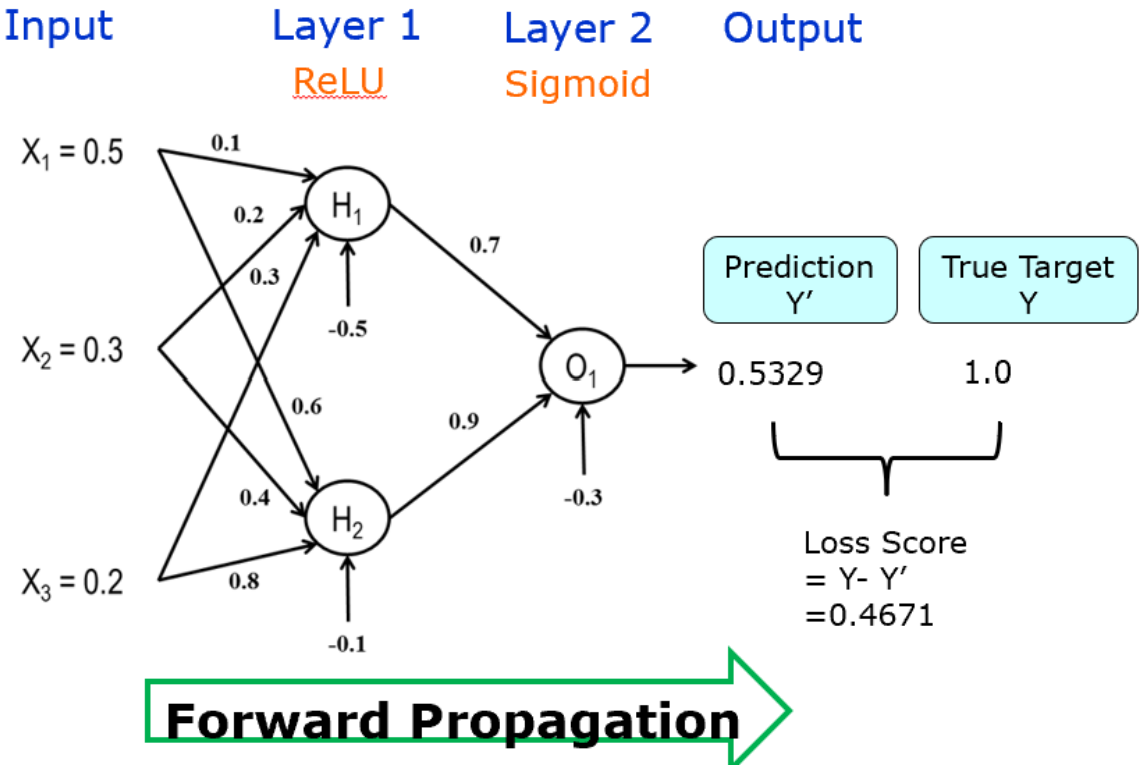
- In a classification problem, a MLP, after a proper training, can classify the samples using line pieces/segments boundary.
- Number of line pieces depends on the number of hidden neurons, training samples, training epochs, and training algorithm.

TRAINING


Input X
Weights
Layer
(data transformation)
Weights
Layer
(data transformation)
Weight
update
Predictions
Y'
True targets
Y
Optimizer
Loss function
Loss score

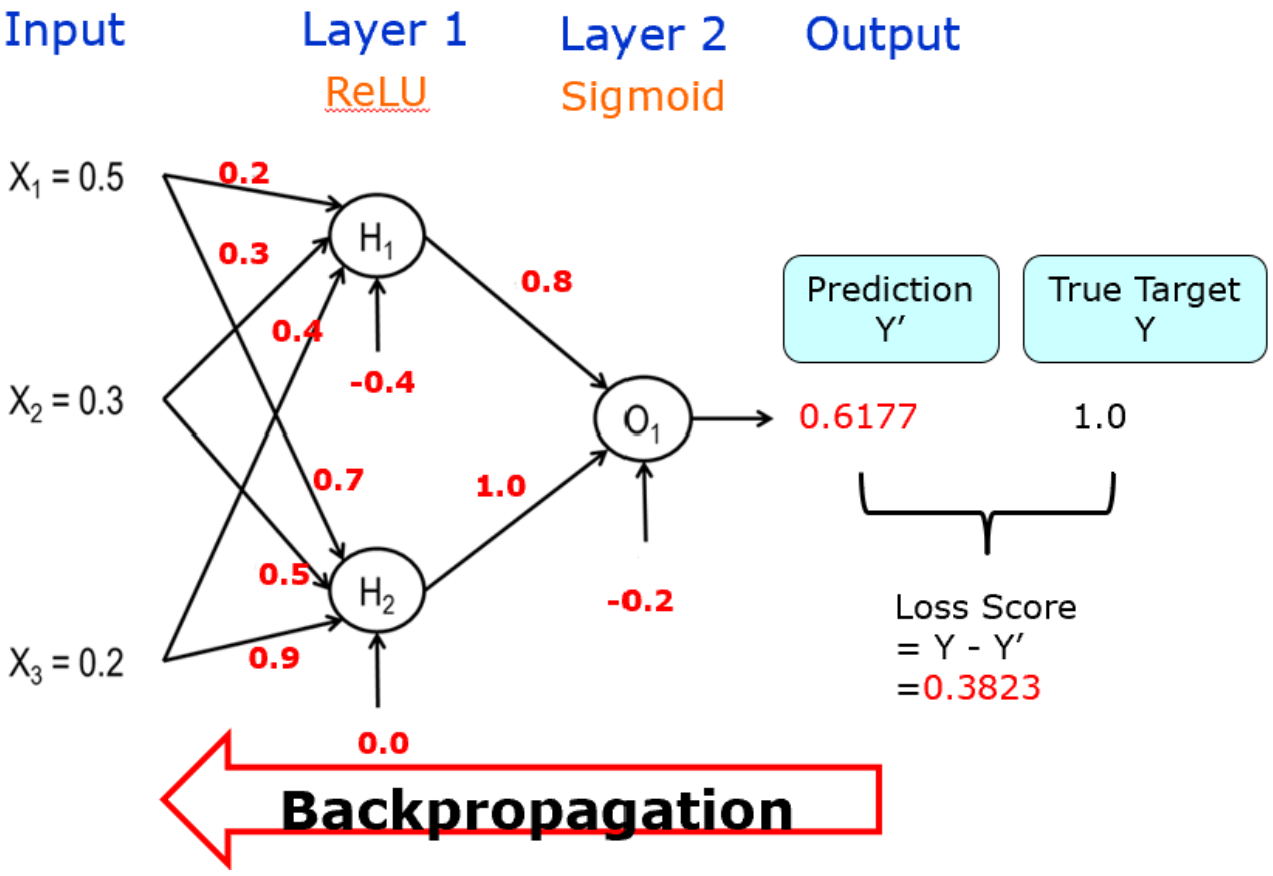
A Simple MLP

Forward Propagation



A Simple MLP

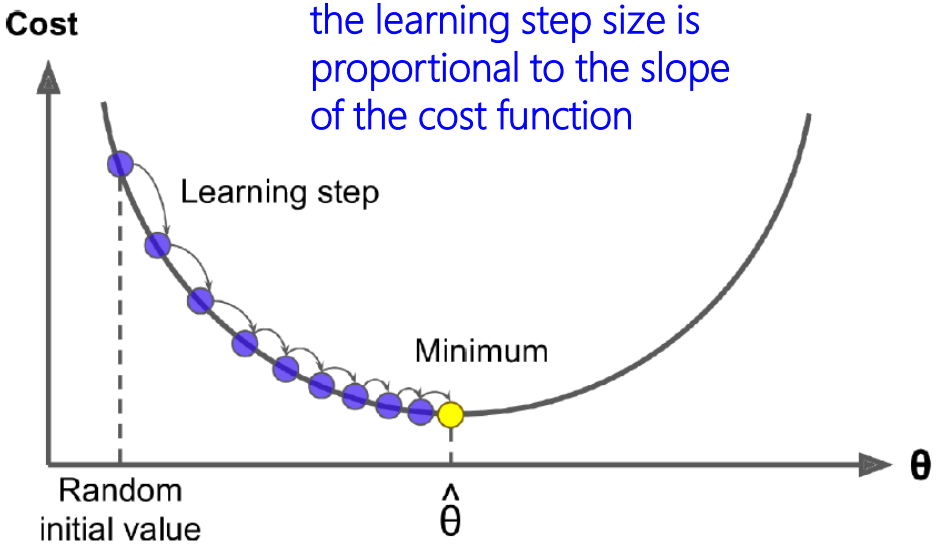
Backward Propagation

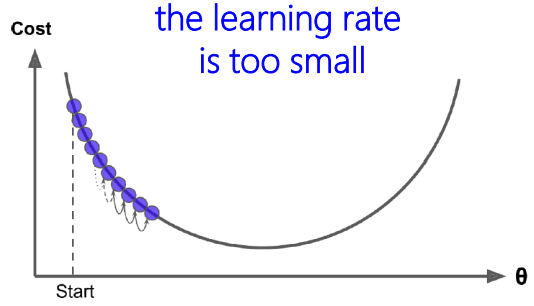


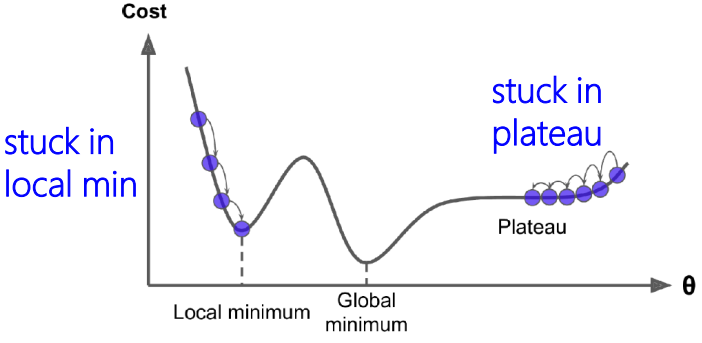
Backpropagation: Computation of the gradient values of a neural network

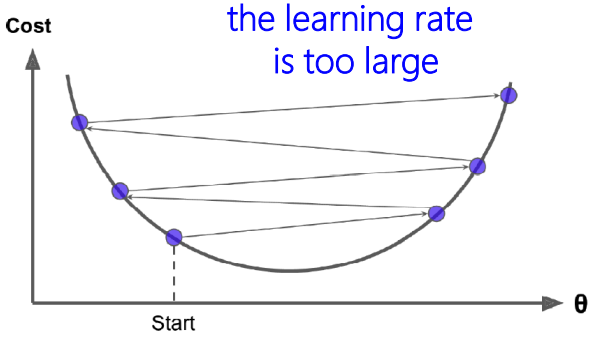
Gradient Descent

The learning rate cannot be too small or too large









Stochastic Gradient Descent

Start with Random Weights (W)

For Each Epoch / Iteration

- Select a batch of training samples X and target Y
- Run the model on X to obtain prediction Y'
- Compute the loss L(Y', Y)
- Compute the gradient of

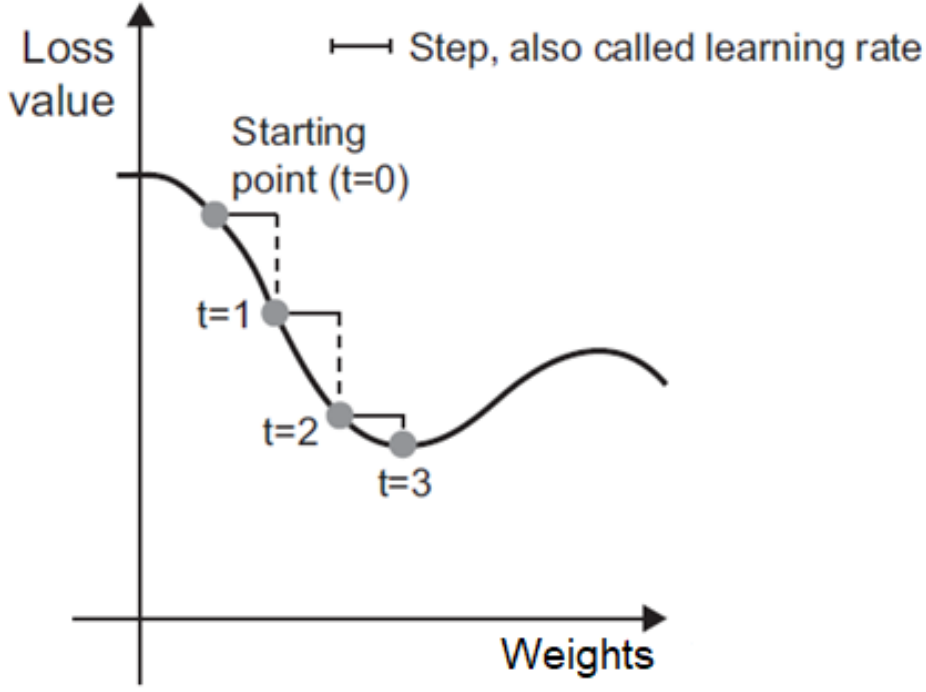
L(Y', Y) wrt Weights (W)

- Move the weights a little

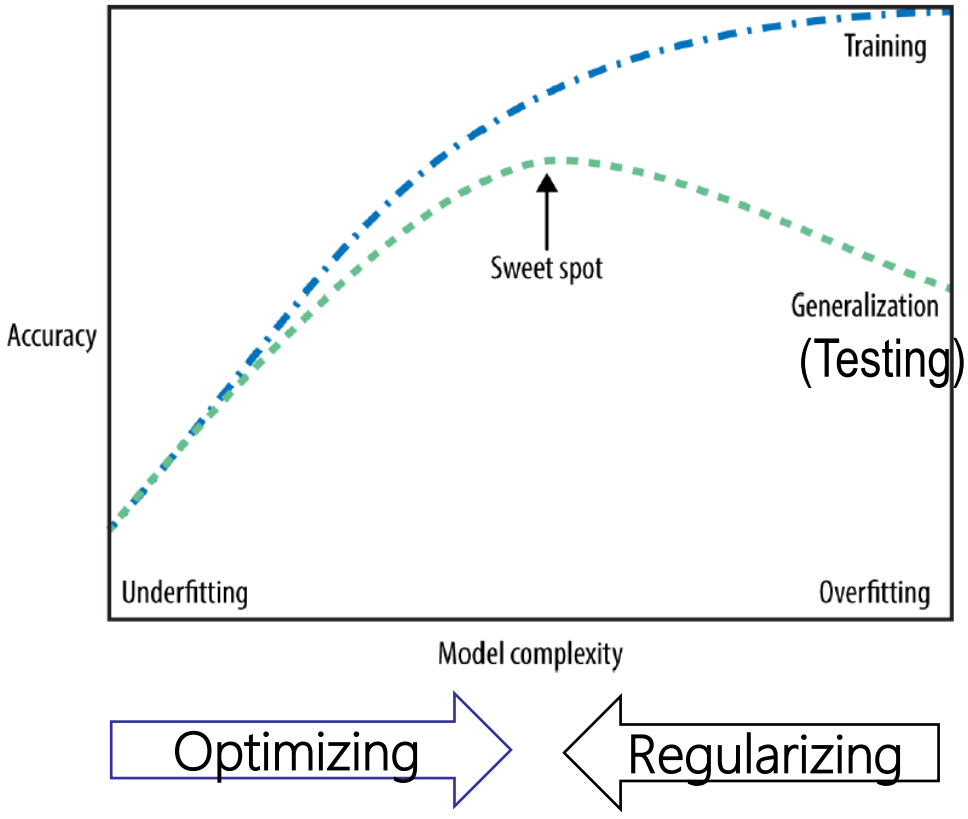
W = W- step *gradient

- Go to the next batch of training

samples until finish all training samples



Generalization, Overfitting and Underfitting

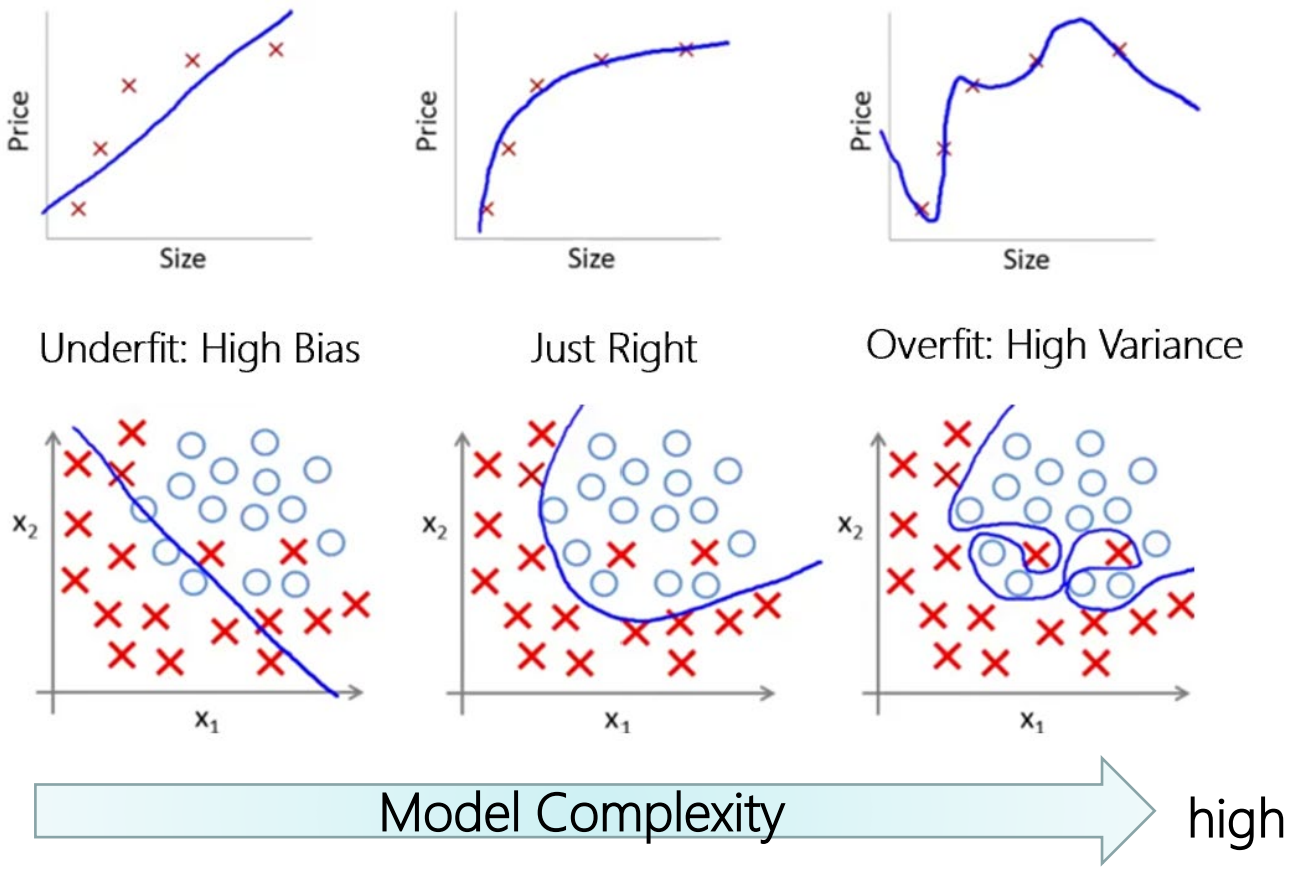


Balancing Optimization and Generalization

Tradeoff of Model Complexity against Training and Testing accuracy

The Bias-Variance Tradeoff

- Model Error
  - Bias
  - Variance
  - Noise



PREVENTING UNDER-FITTING AND OVER-FITTING

To prevent under-fitting we need to make sure that:

1. The network has enough hidden units to represent the required mappings.
2. We train the network for long enough so that the error/cost function is sufficiently minimized.

PREVENTING UNDER-FITTING AND OVER-FITTING

NUS National University of Singapore | School of Computing

To prevent over-fitting we can:

1. Stop the training early – before it has had time to learn the training data too well.
2. Restrict the number of adjustable parameters the network has, e.g. by reducing the number of hidden units, or layers.
3. Add some form of regularization term to the error function to encourage smoother network mappings.
4. Add noise to the training patterns to smear out the data points.

The universal workflow of machine learning

1. Defining the problem and assembling a dataset
2. Choosing a measure of success
3. Deciding on an evaluation protocol
4. Preparing your data
5. Developing a model that does better than a baseline
6. Scaling up: train the model until it overfits
7. Regularizing your model and tuning your hyperparameters

Developing Models in Keras

1. Define your training data
   - input tensors and target tensors
2. Define a network of layers (or model)
   - maps your inputs to your targets
3. Configure the learning process
   - loss function, optimizer and metrics to monitor
4. Iterate on your training data
   - calling the fit() method of your Model

Two ways to define a model

- Sequential
  - Only for linear stacks of layers
  - Most common network architecture

```
model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(784,)))
model.add(layers.Dense(10, activation='softmax'))
```

- Functional API
  - Can build arbitrary architectures

```
input_tensor = layers.Input(shape=(784,))
x = layers.Dense(32, activation='relu')(input_tensor)
output_tensor = layers.Dense(10, activation='softmax')(x)

model = models.Model(inputs=input_tensor, outputs=output_tensor)
```

A Sequential Model

```
# Define the ANN model
model = Sequential()
model.add(Dense(10, activation='relu', input_shape = (8,)))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

- Number of Layers and Number of Neurons
  - The more layers and more neurons ➔ model overfitting
  - The less layers and less neurons ➔ model underfitting

- Activation Function
  - ReLu is the most popular one
  - The last Layer depends on the type of problem
    - Classification
      - Binary ➔ ‘Sigmoid’
      - Multiple ➔ ‘Softmax’
    - Regression ➔ None

Compilation Step: configure learning process

```
from tensorflow.keras import optimizers
model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=0.01),
              metrics=['accuracy'])
```

- Optimizer
  - SGD, RMSprop, Adam, and etc.
- Loss Function
  - mean_squared_error, mean_absolute_error (“regression”)
  - categorical_crossentropy, binary_crossentropy (“classification”)
- Metrics
  - mae (“regression”)
  - accuracy (“classification”)

```
network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Last-layer and Loss configuration

Problem Type	Last-layer Activation	Loss Function
Binary Classification	Sigmoid	binary_crossentropy
Multiple Classification	Softmax	categorical_crossentropy
Regression	None	mse

Fit Step: learning through training data

- Training data: Numpy arrays
  - input_tensor & target_tensor
- Epochs
  - Number of iterations
    - More epochs/iterations ➔ overfitting
    - Less epochs/iterations ➔ underfitting
- Batch Size
  - Each iteration, fit() iterates over all the samples batch by batch.
  - In a run of each batch:
    - Forward Pass -> Loss-> Gradient Descent -> Backward Pass -> Update Weights

```
model.fit(input_tensor, target_tensor, batch_size=128, epochs=10)
```

ANN HYPERPARAMETERS

Number of layers

Number of neurons

Activation function

Number of epochs

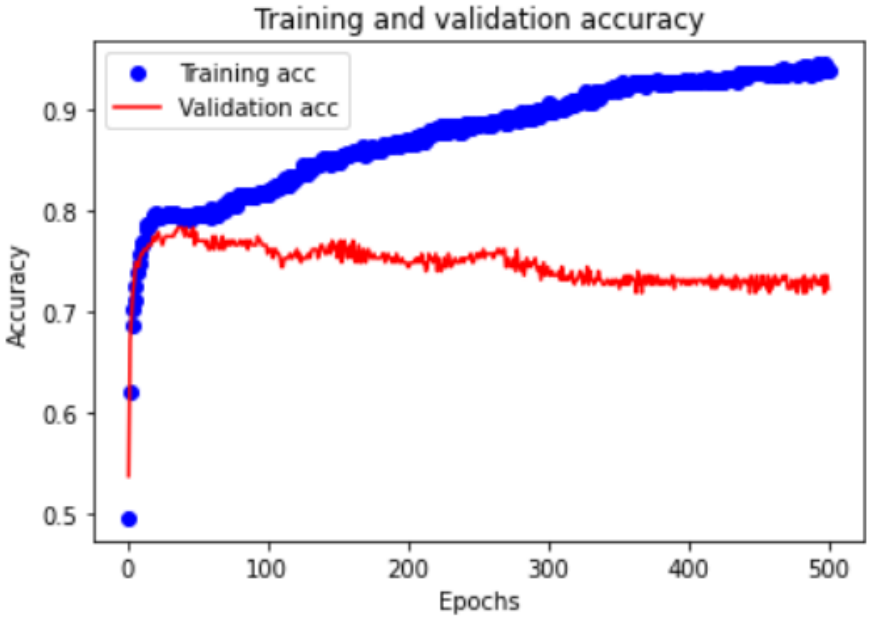
Batch Size

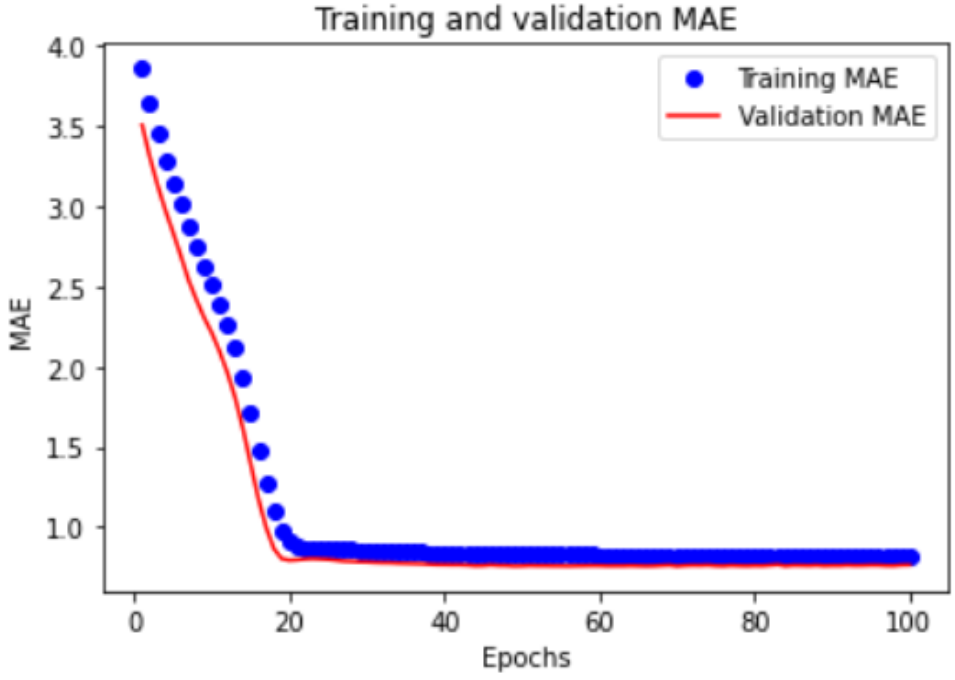
Training algorithm (Optimizer) and Learning rate

Regularization

……..

PRACTICAL 7: ANN



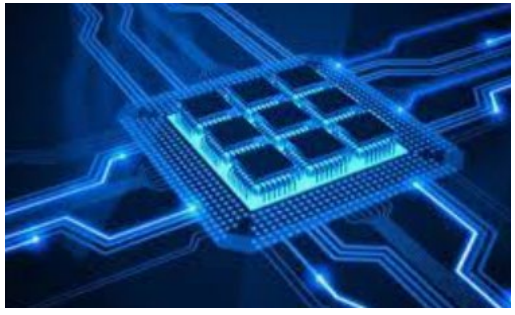


Deep Learning

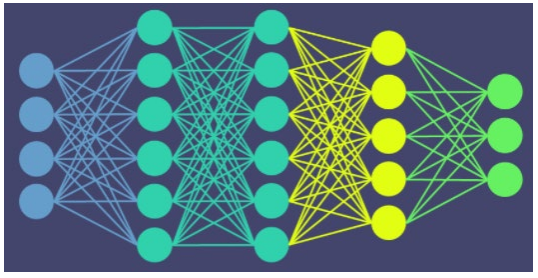
Neural Networks with Many Layers

Why Now?

Big Data



Computing Power



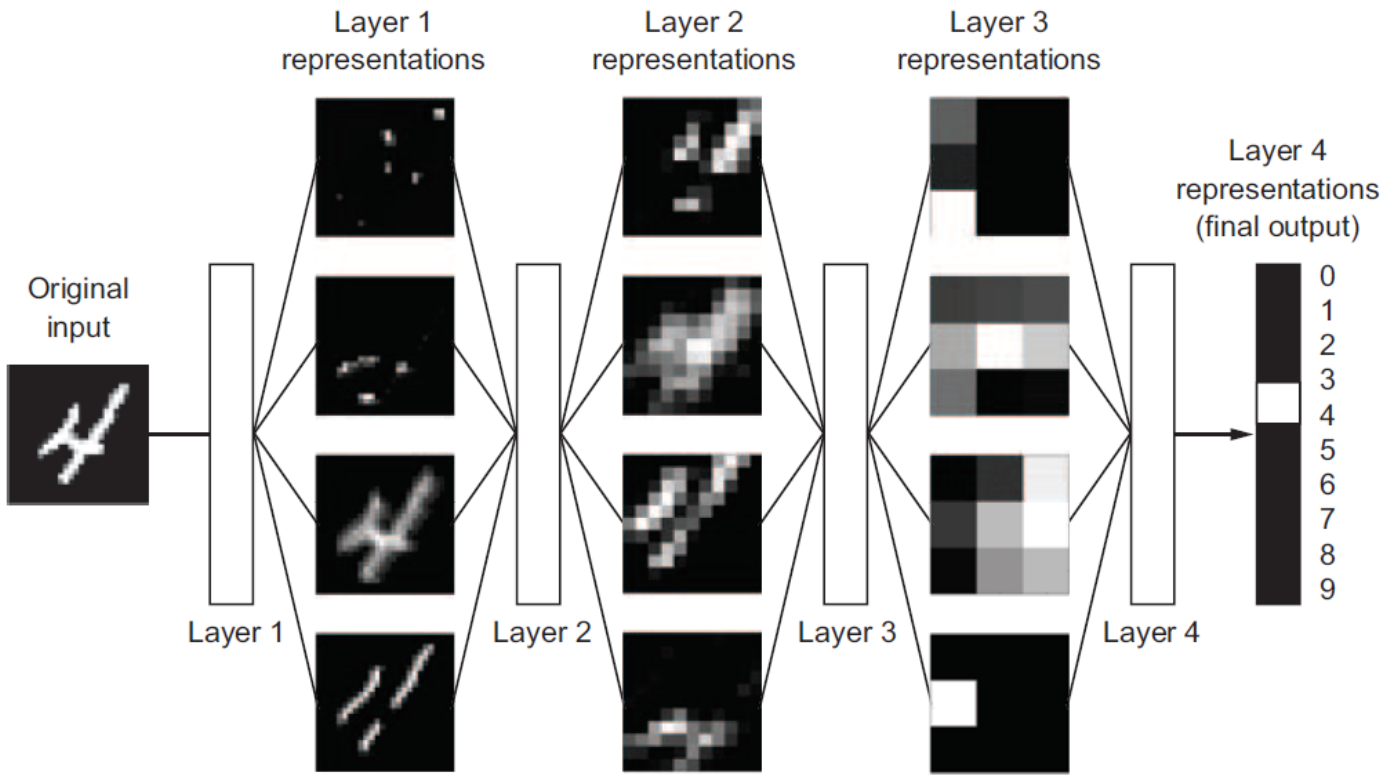
Algorithms

Data is Critical To Breakthroughs in AI

Year	Breakthroughs in AI	Datasets (First Available)	Algorithms (First Proposed)
1994	Human-level read-speech recognition	Spoken Wall Street Journal articles and other texts (1991)	Hidden Markov Model (1984)
1997	IBM Deep Blue defeated Garry Kasparov	700,000 Grandmaster chess games, aka "The Extended Book" (1991)	Negascout planning algorithm (1983)
2005	Google's Arabic- and Chinese-to-English translation	1.8 trillion tokens from Google Web and News pages (collected in 2005)	Statistical machine translation algorithm (1988)
2011	IBM Watson became the world Jeopardy! champion	8.6 million documents from Wikipedia, Wiktionary, Wikiquote, and Project Gutenberg (updated in 2010)	Mixture-of-Experts algorithm (1991)
2014	Google's GoogleNet object classification at near-human performance	ImageNet corpus of 1.5 million labeled images and 1,000 object categories (2010)	Convolutional neural network algorithm (1989)
2015	Google's Deepmind achieved human parity in playing 29 Atari games by learning general control from video	Arcade Learning Environment dataset of over 50 Atari games (2013)	Q-learning algorithm (1992)
Average No. of Years to Breakthrough:		**3 years**	**18 years**

Source: Train AI 2017, https://www.crowdflower.com/train-ai/

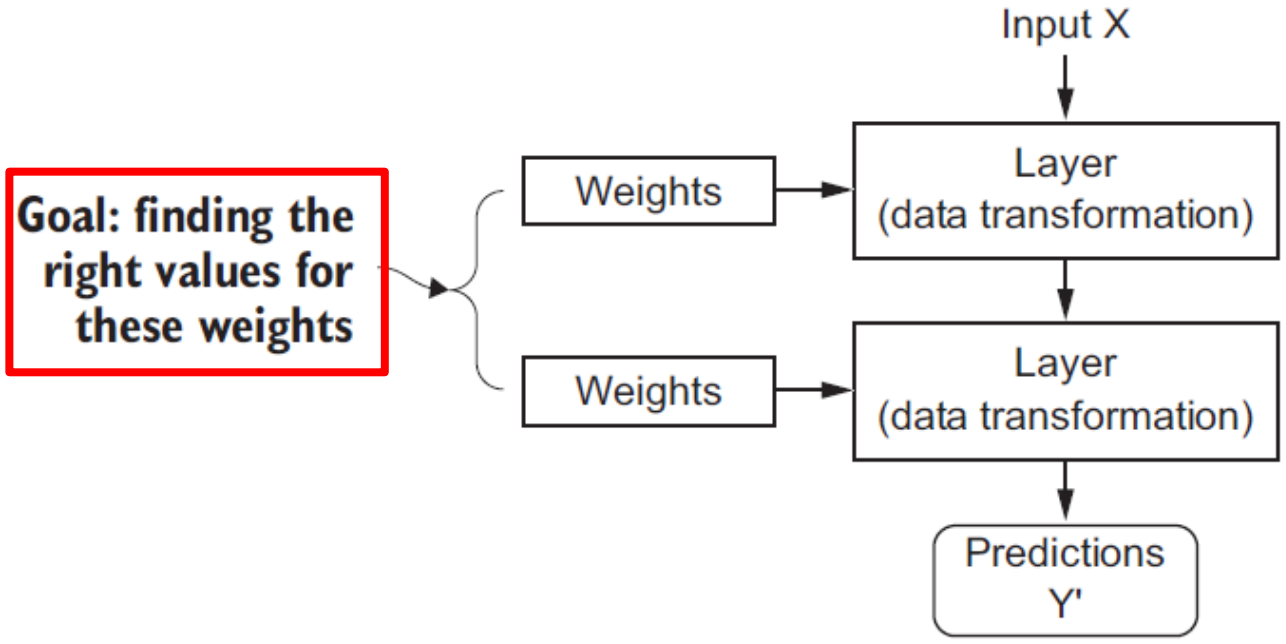
A MNIST Example



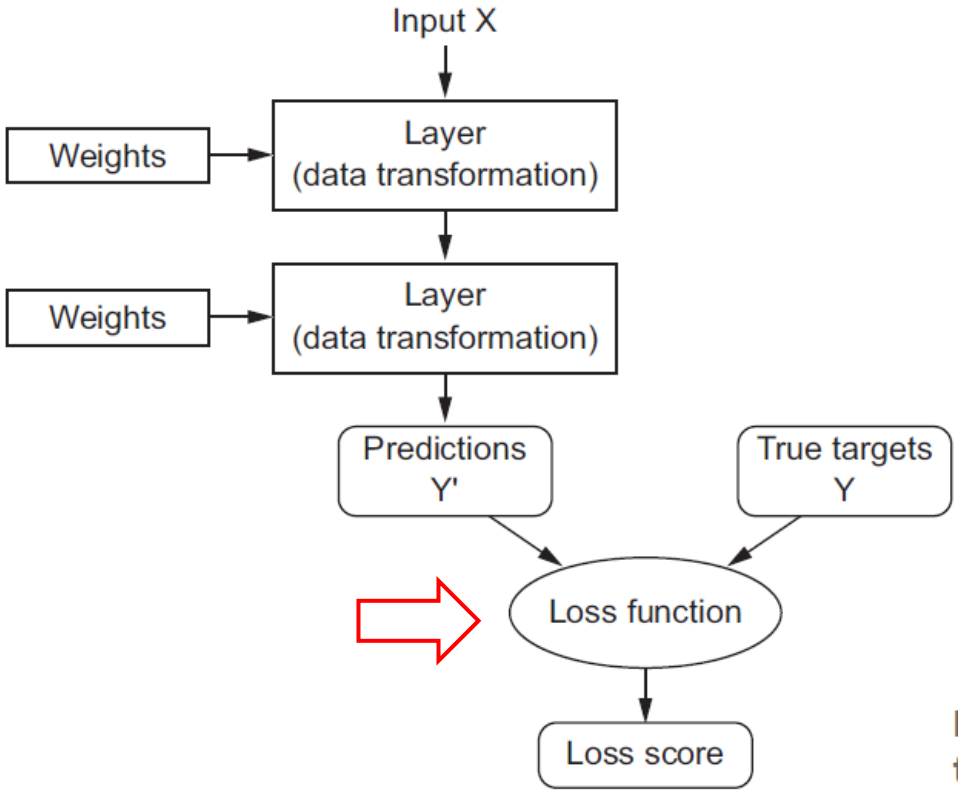
DL Basic Components (e.g. MNIST)

- Inputs
  - Input Data points (e.g. digits images)
- Labels
  - Examples of the expected output (e.g. digit 0 to 9)
- Loss Function: A way to measure whether the algorithm is doing a good job
  - The measurement is used as a feedback signal to adjust the way the algorithm works. This adjustment step is what we call learning.

A DL model is parameterized by its weights

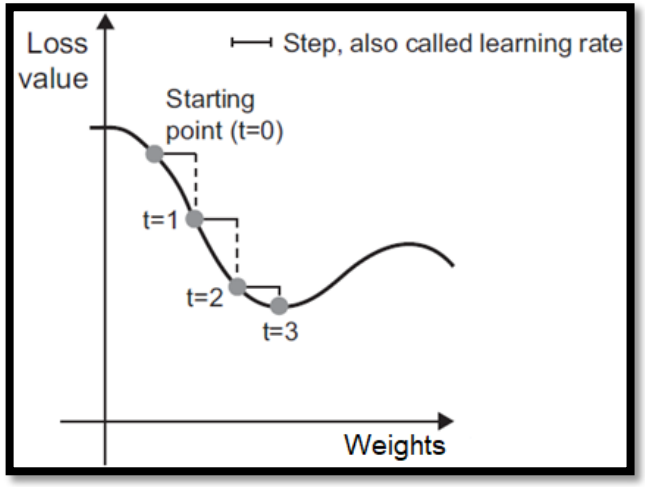


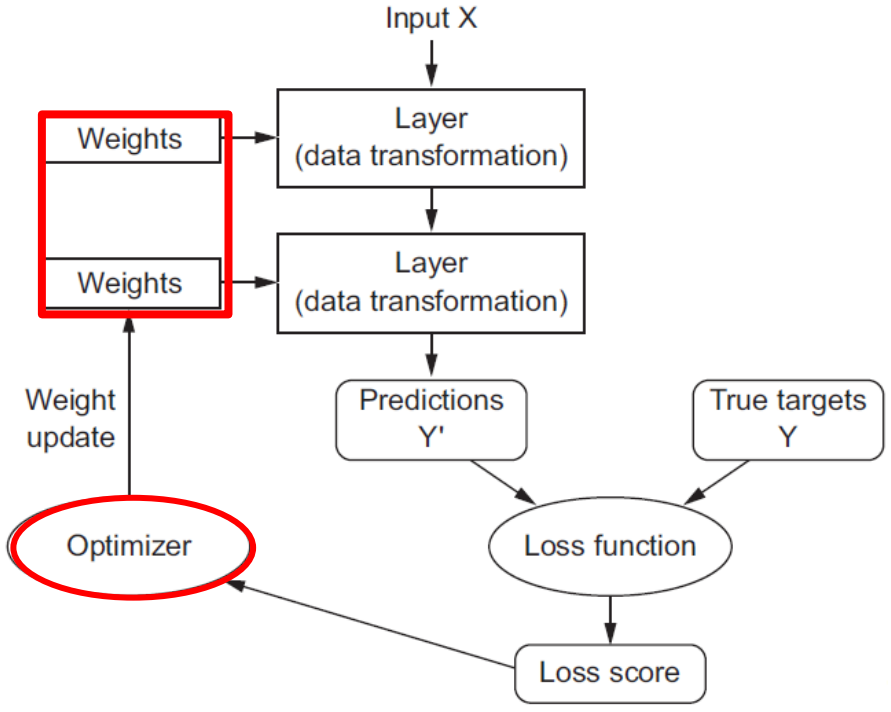
A loss function measures the quality of the network's output



The loss score is used as a feedback signal to adjust the weights through Optimizers

Optimizer is using gradient descent algorithms to update the weights and minimize the loss

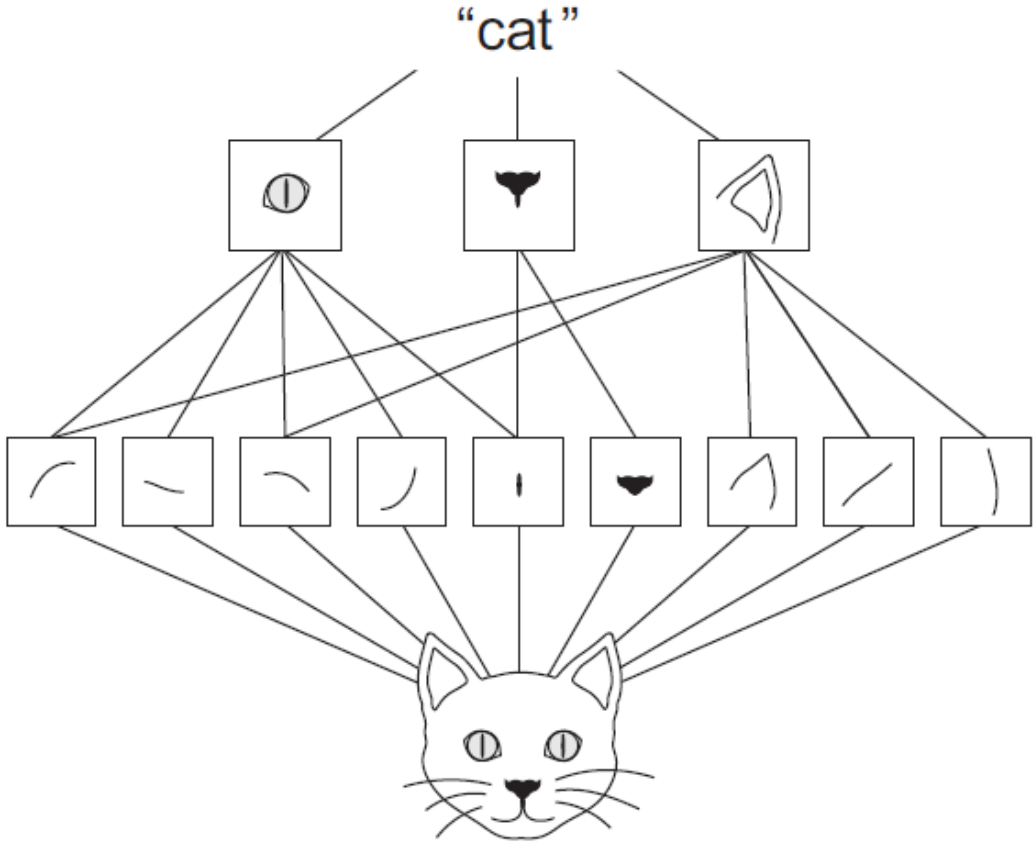




Different Types of Layers

- Dense Layer: Fully Connected Layers
  - Prediction and Classification using structured datasets
- Convolutional Layer: Using filters to learn local patterns
  - Image Classification, Face Recognition and etc.
- Recurrent Layer: make prediction based on the past sequences
  - Sentiment Analysis, Language translation and etc.
- Attention / Transformer Layer: assign weights to different words, indicating the importance & relevance among different words
  - Natural Language Processing (NLP)

Convolutional Layer

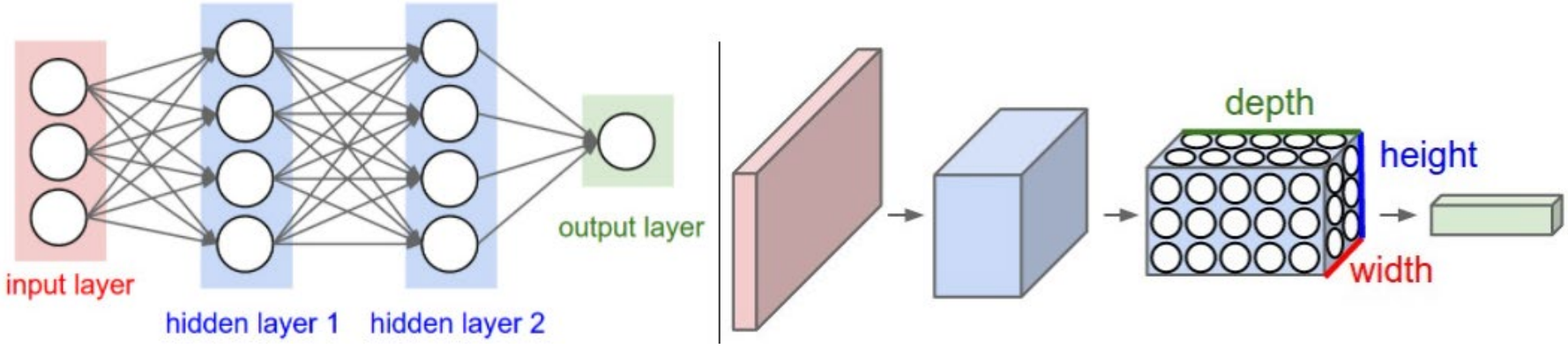


PRACTICAL 8: DL

We are going to use ANN and CNN to solve an image classification problem.

The problem we are trying to solve here is to classify grayscale images of handwritten digits (28 pixels by 28 pixels), into their 10 categories (0 to 9). The dataset we will use is the MNIST dataset, a classic dataset in the machine learning community, which has been around for almost as long as the field itself and has been very intensively studied. It's a set of 60,000 training images, plus 10,000 test images, assembled by the National Institute of Standards and Technology (the NIST in MNIST) in the 1980s.

A regular 3-layer Neural Network vs. A ConvNet



source: http://cs231n.github.io/convolutional-networks/

The MNIST classification problem

A Regular Neural Network

```
network = models.Sequential()
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
network.summary()
```

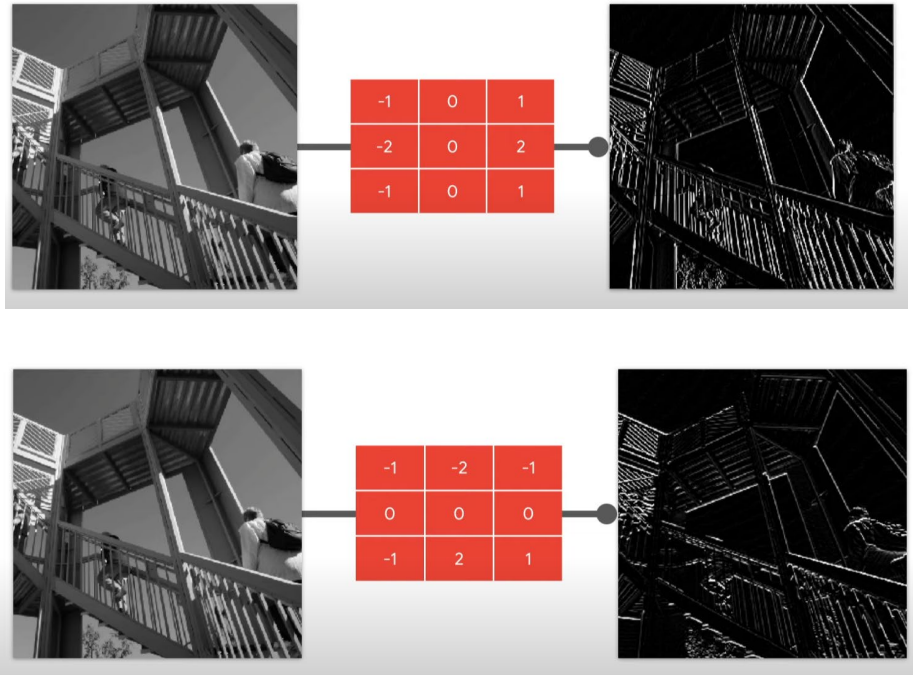
Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	401920
dense_1 (Dense)	(None, 10)	5130

ConvNets

```
model = models.Sequential()
model.add(layers.Conv2D(32, kernel_size = (3, 3), activation='relu', input_shape=(28, 28, 1)))
model.add(layers.MaxPooling2D(pool_size = (2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 26, 26, 32)	320
maxpooling2d_1 (MaxPooling2D)	(None, 13, 13, 32)	0
conv2d_2 (Conv2D)	(None, 11, 11, 64)	18496
maxpooling2d_2 (MaxPooling2D)	(None, 5, 5, 64)	0
conv2d_3 (Conv2D)	(None, 3, 3, 64)	36928
flatten_1 (Flatten)	(None, 576)	0
dense_1 (Dense)	(None, 64)	36928
dense_2 (Dense)	(None, 10)	650

Conv2D Layer (through filters)



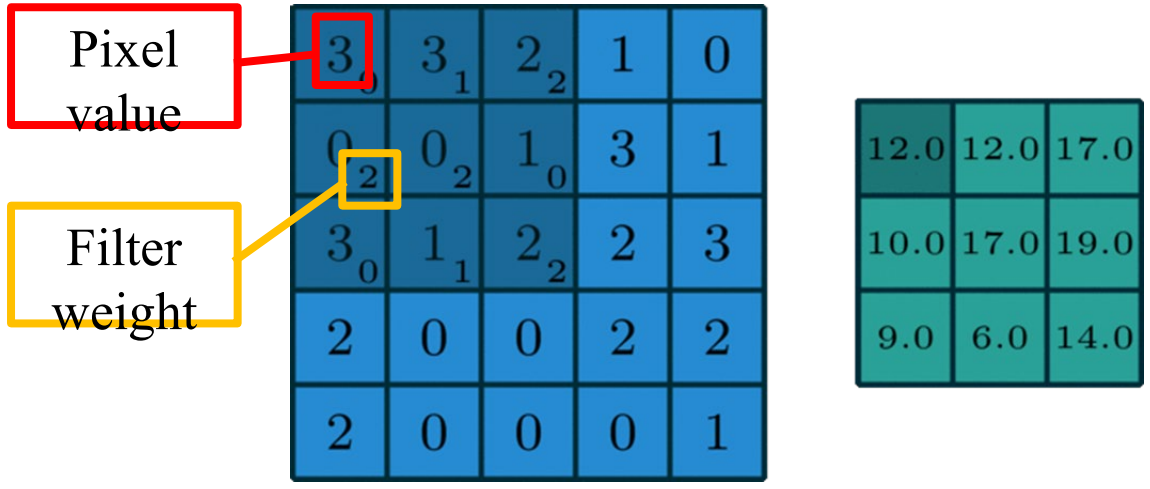
MaxPooling

source: https://youtu.be/x_VrgWTKkiM

A Simple Example of 2D convolution

Input Image: (5, 5, 1) ➔ Output (3, 3, 1)

One Filter: (3, 3)

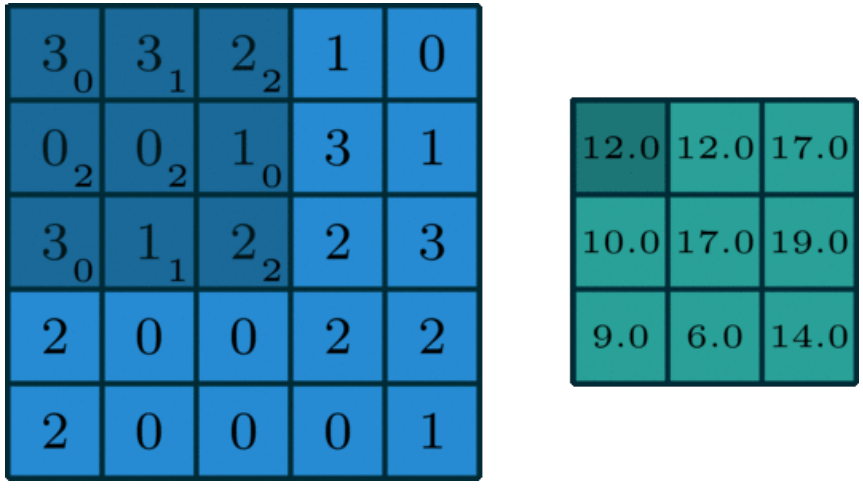


Source:
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

A Simple Example of 2D convolution

Input Image: (5, 5, 1) ➔ Output (3, 3, 1)

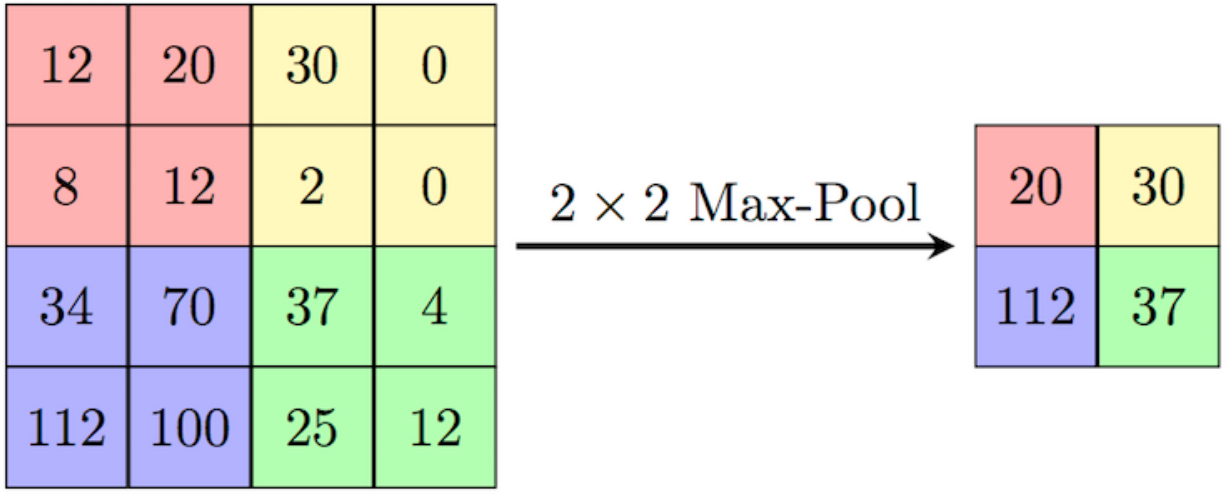
One Filter: (3, 3)



Source:
https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

Max Pooling Operation

Down sample the tensors by taking the max value in a window (e.g. 2x2)



Source:
https://computersciencewiki.org/index.php/Max-pooling_/_Pooling

Assessment: Mini Project

ASSESSMENT: MINI PROJECT

- Datasets: bank.csv
- Tasks: Build machine learning model to predict the target variable “deposit”.
- You can use any ML model we learned in this course, e.g. KNN, SVM, ANN.
- Tune the Model hyperparameters to improve the model performance.
- You need to build at least one model but feel free to build a few models and compare their performance.
- Report your best training and testing results

CONCLUSION

These days, machine learning techniques are being widely used to solve real-world problems by storing, manipulating, extracting and retrieving data from large sources.

Supervised machine learning and unsupervised machine learning both will play an essential role in the future world of technology.

Programme Evaluation

THAT'S ALL …

Thank You!

Any Question?

USEFUL REFERENCES

1. Artificial Intelligence, A Modern Approach, Stewart Rusell and Peter Norvig, Pearson Pub, 3rd ed. 2009.
2. Artificial Intelligence For Dummies, John Paul Mueller, Luca Massaron, Wiley, 1st ed., 2018.
3. Deep Learning For Dummies, John Paul Mueller, Luca Massaron, Wiley, 1st ed., 2019.
4. Aurélien Géron (2019). Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow, 2nd edition.
5. François Chollet (2018). Deep Learning with Python.

Appendix

Anaconda Environment Setup

ANACONDA

Anaconda: A free and open-source distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment.

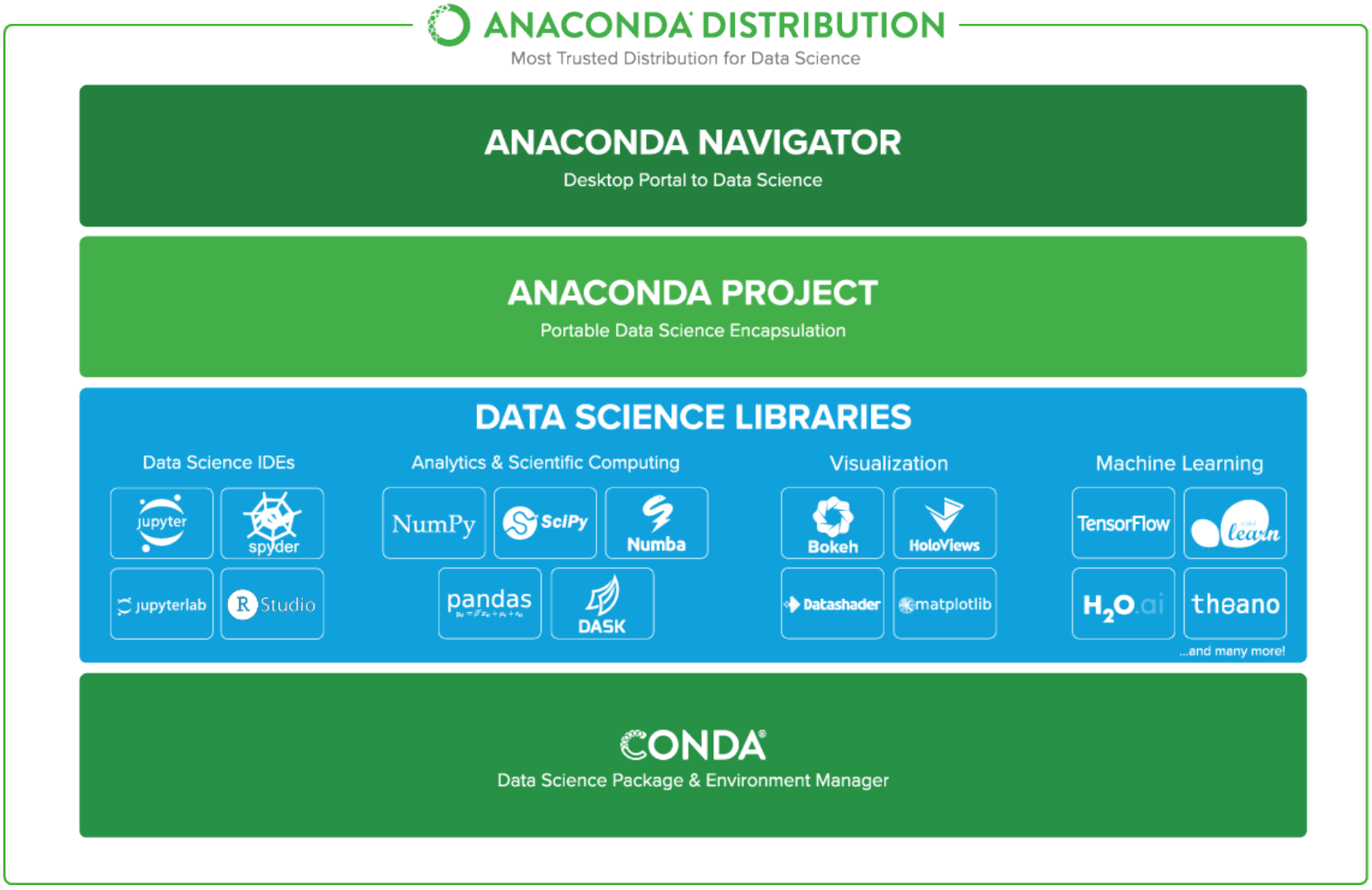
Developers describe Anaconda as "The Enterprise Data Science Platform for Data Scientists, IT Professionals and Business Leaders".

Anaconda functions as a wrapper or a Swiss army knife, but a good one.

It is an extra shell or a package for ML and DA.

HTTPS://STACKOVERFLOW.COM/

ANACONDA

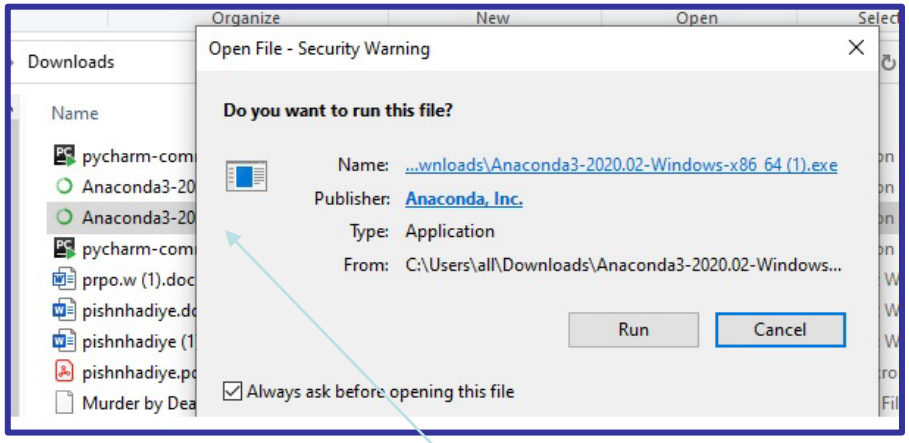


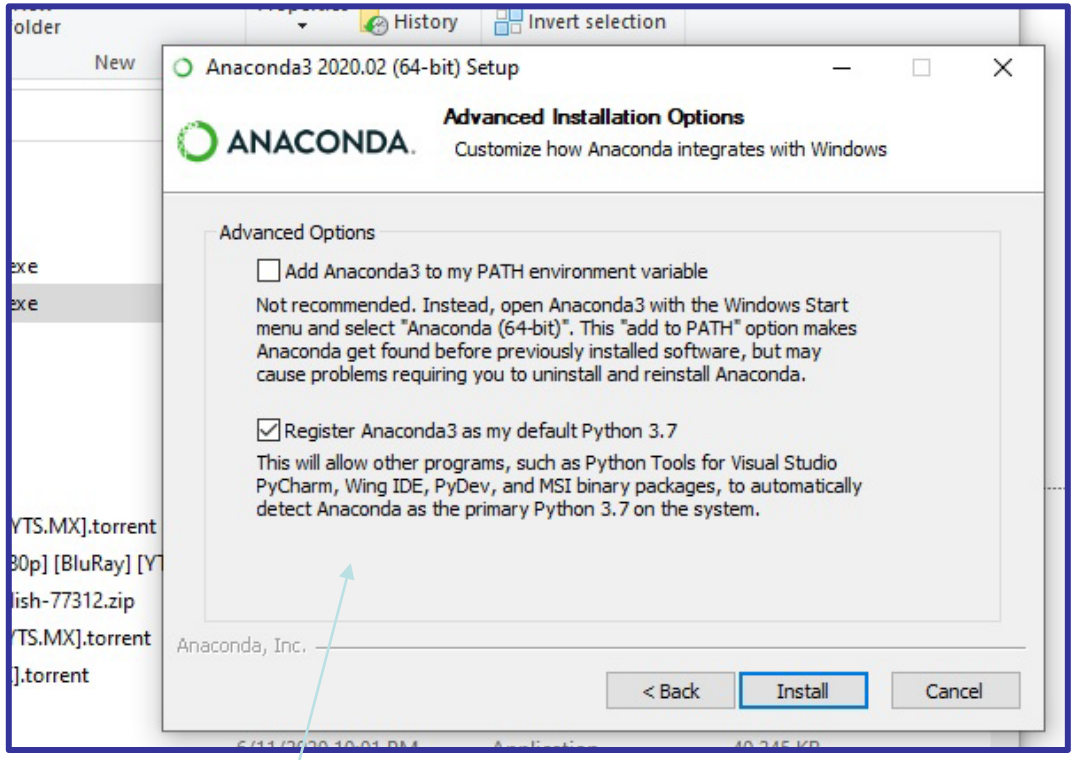
HTTPS://STACKOVERFLOW.COM/

PROCEDURE

1. Install Anaconda
    1. Go to: Anaconda → Products → individual, https://www.anaconda.com/products/individual
    2. Find a proper latest version for your OS and mode (64bits, in some rare cases 32bits)
    3. What is it? Well, to me Anaconda is something between an OS and a toolbox, necessary however for many Python programs and apps. Should we get back to older ICT terminology, it would be defined as a shell or a distribution. It will support many AI and Data Science programs written in Python or R. it is a FOSS.
    4. In July 2020, the advisable version is for Pyhton3.7

ANACONDA INSTALLATION



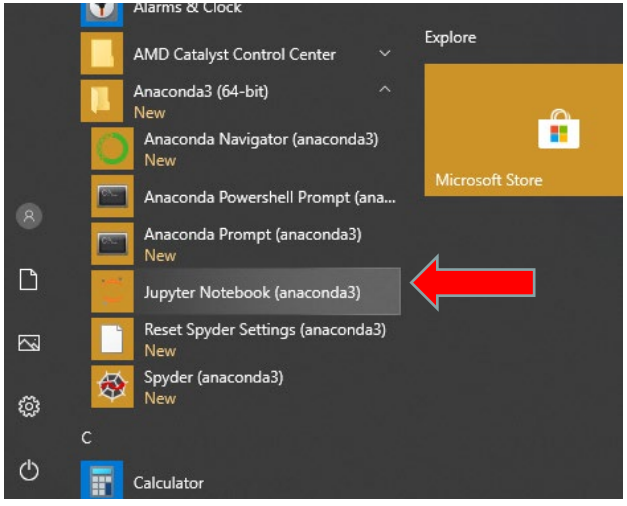


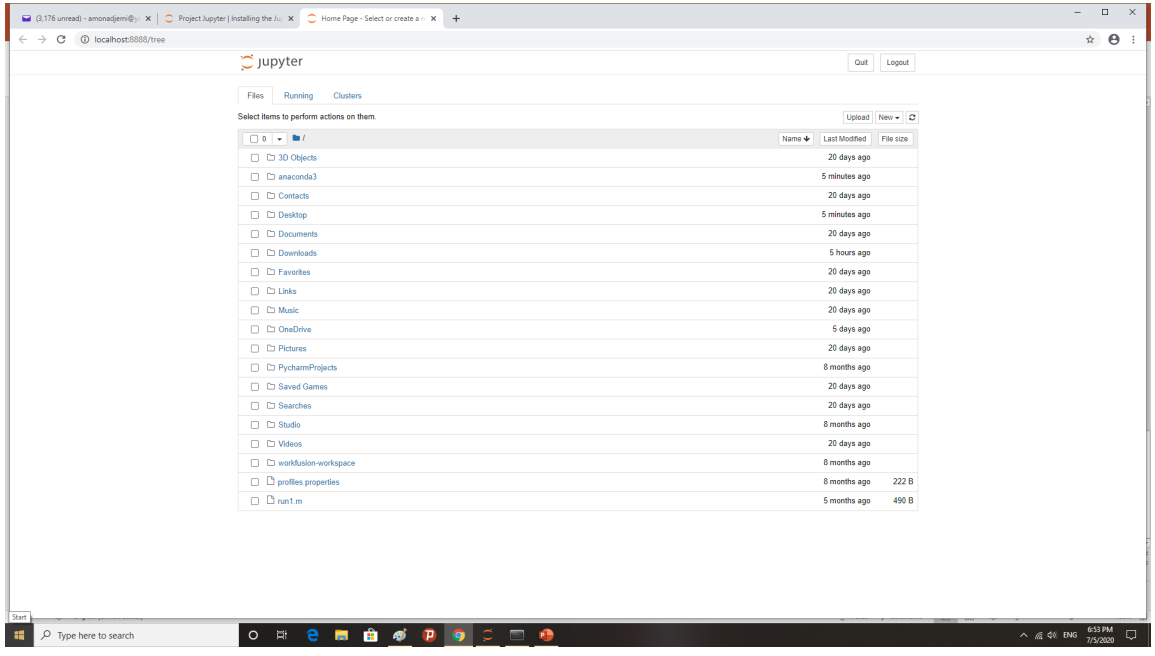
1- Run the installer file, 2- Keep it like this

JUPYTER INSTALLATION

Typically, when you install Anaconda, Jupyter notebook would be installed automatically.

Run it, and you can have the IDE interface in you web browser.

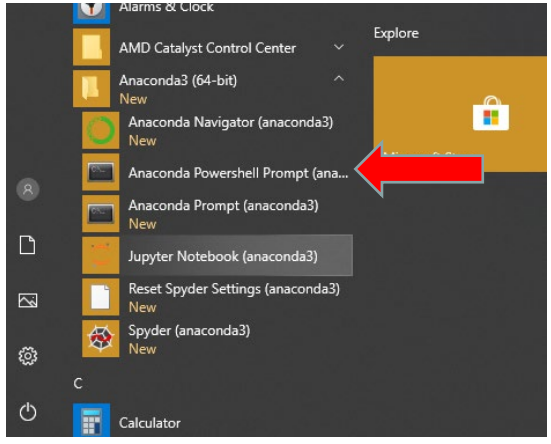


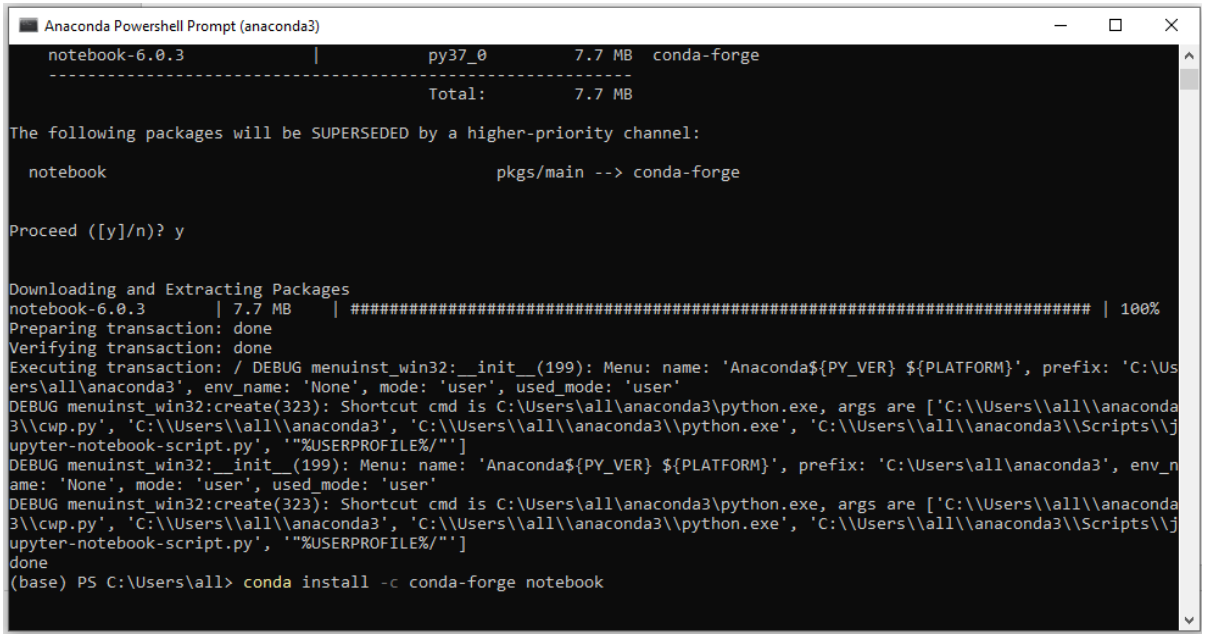


JUPYTER INSTALLATION

Or, run the Anaconda Powershell, and type this two commands:

- >>conda install -c conda-forge jupyterlab
- >>conda install -c conda-forge notebook





PYTHON PACKAGES INSTALLATION

REMEMBER: All through all setup steps, you must be connected to the Internet.

You may need several Python packages. Here you are some examples:

```
>>conda install -c conda-forge  tensorflow
>>conda install -c conda-forge   keras
```

To install Tensorflow and Keras packages/toolboxes/libraries.

Alternatively, you may use pip command:

```
pip  install  tensorflow
```

To check which packages have been already installed in your Python/Anaconda system, use:

```
>>conda list
```

PYTHON PACKAGES INSTALLATION

To install packages from your local hard drive:

conda install --offline package path and name

For example:

conda install --offline "C:\Users\sleam\Downloads\SimpSOM-1.3.4.tar.gz"

To install packages from a web source:

conda install -c URL

For example:

conda install -c https://conda.binstar.org/pymc pymc

Instead, you may try:

```
pip  install  package_name      #(from anaconda powershell prompt)
!pip  install  package_name     #(from jupyter notebook)
```