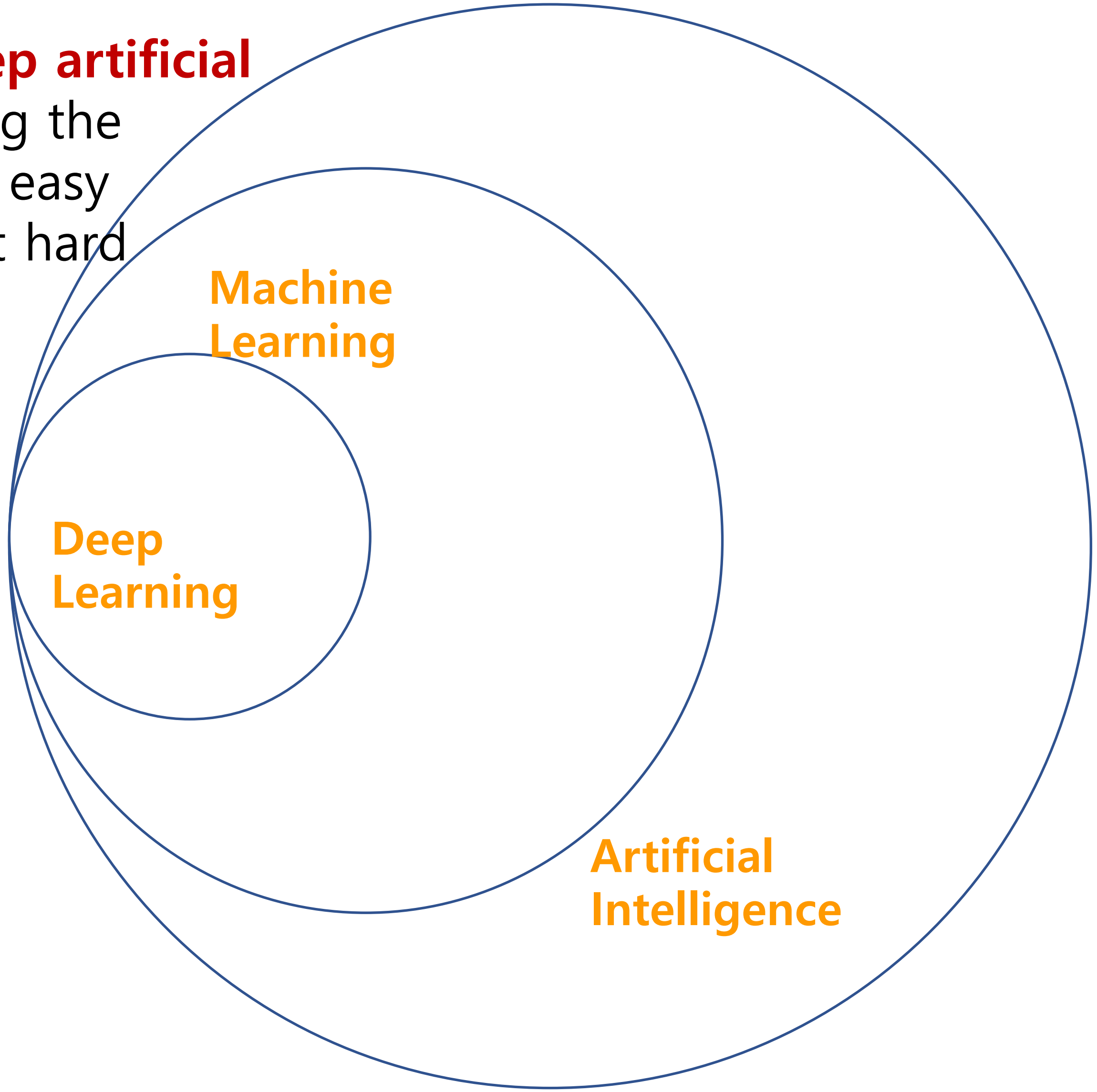
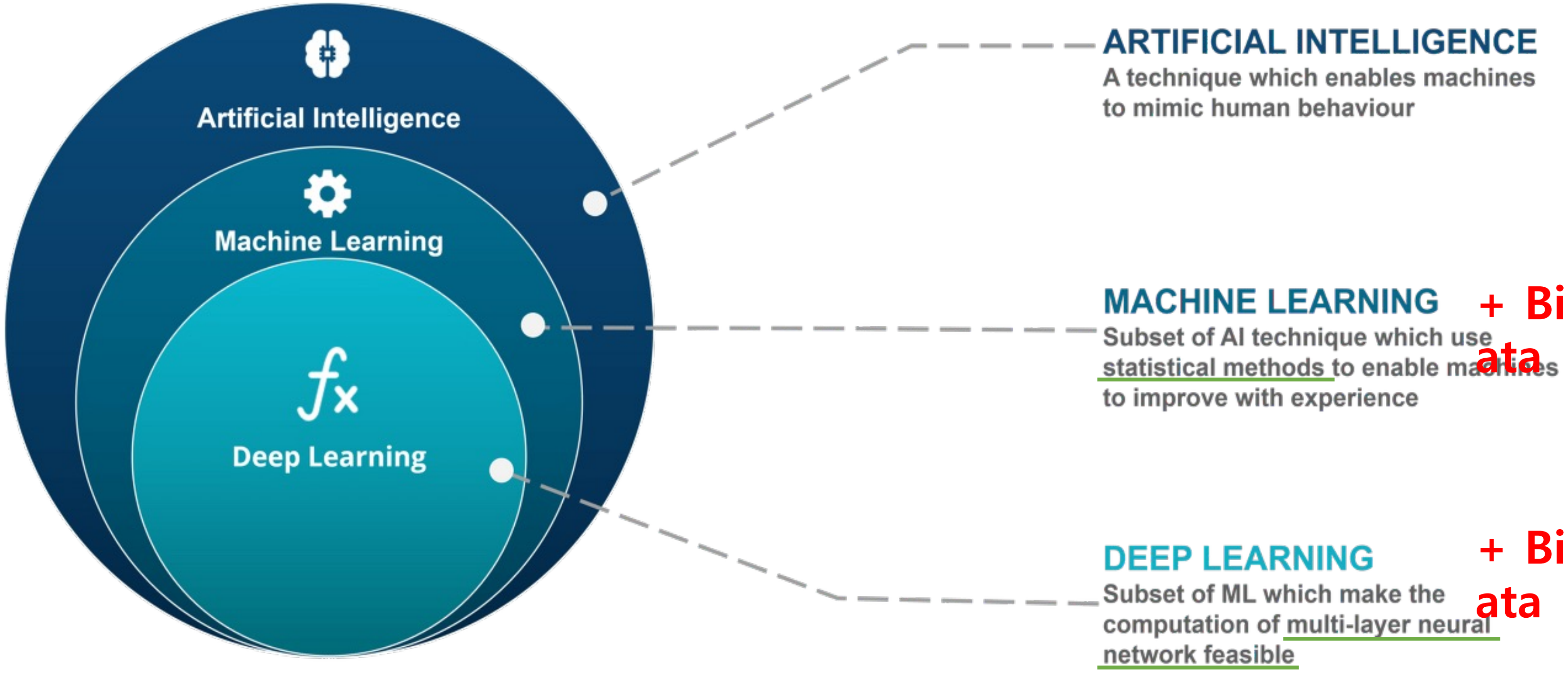
Data-driven machine learning

AI, Machine Learning, Deep Learning

Deep Learning uses a **deep artificial neural network** for solving the recognition tasks that are easy for people to perform but hard for people to describe formally



AI, Machine Learning, Deep Learning



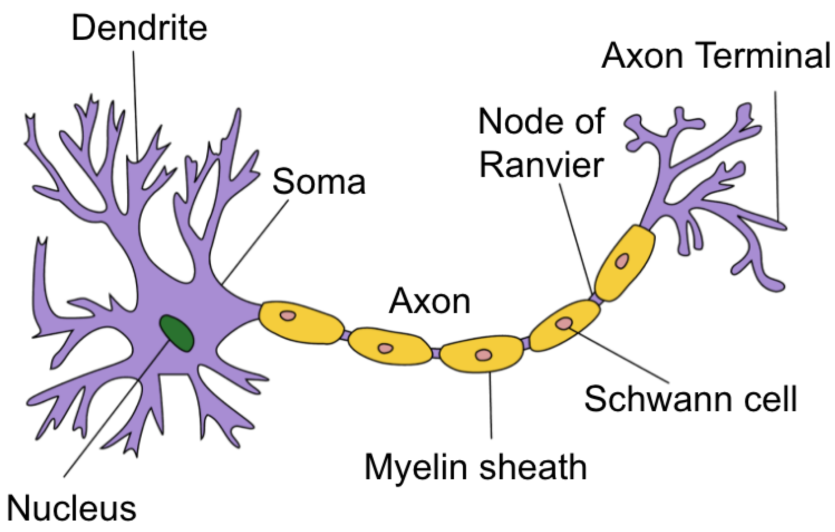
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour

MACHINE LEARNING + Big data
Subset of AI technique which use statistical methods to enable machines to improve with experience

DEEP LEARNING + Big data
Subset of ML which make the computation of multi-layer neural network feasible

Neuron

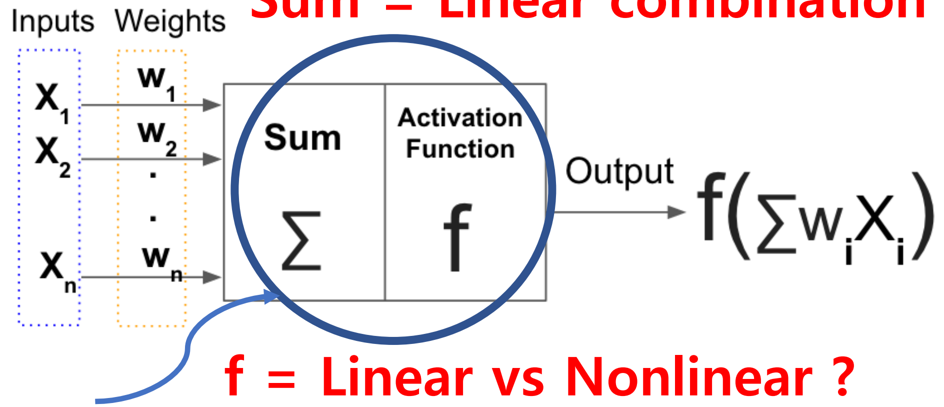
Input: Electric signals



Structure of a typical neuron
(source: Wikipedia)

Input: Data

Sum = Linear combination



f = Linear vs Nonlinear ?

neuron

Structure of artificial neuron

https://en.wikipedia.org/wiki/Neuron (Check fig. for neurons)

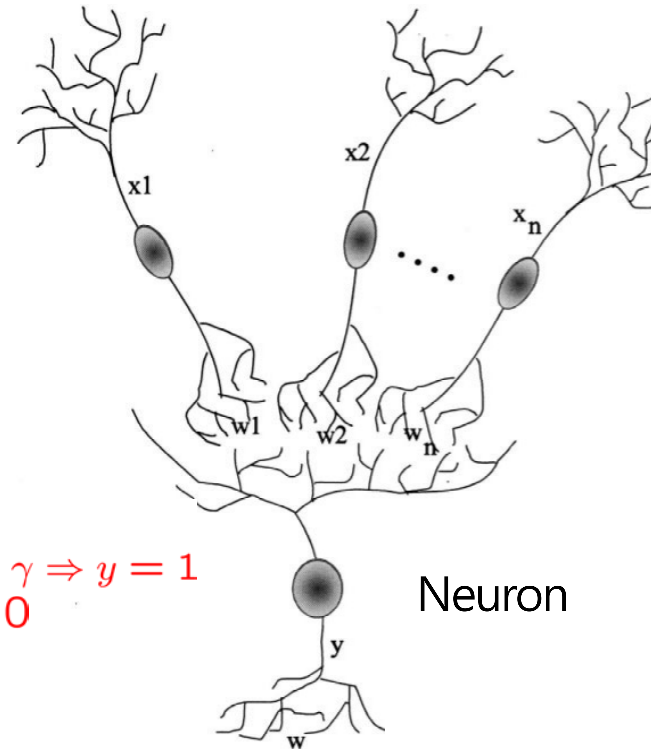
http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

Artificial Neural Network

Electric potential at neuron y:

$$w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n > \gamma \Rightarrow y = 1$$

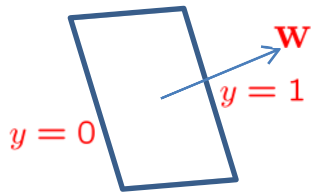
$$\sum_i w_i x_i - \gamma > 0$$



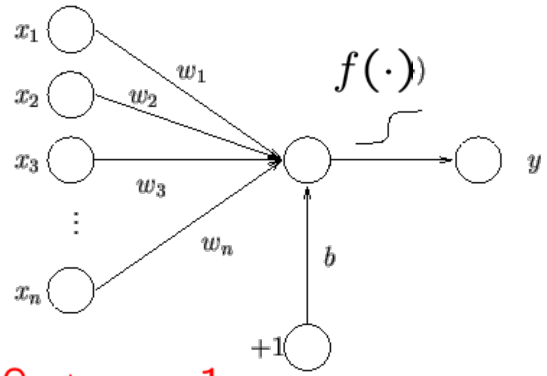
Neuron

Decision boundary:

$$\sum_i w_i x_i + b = 0$$

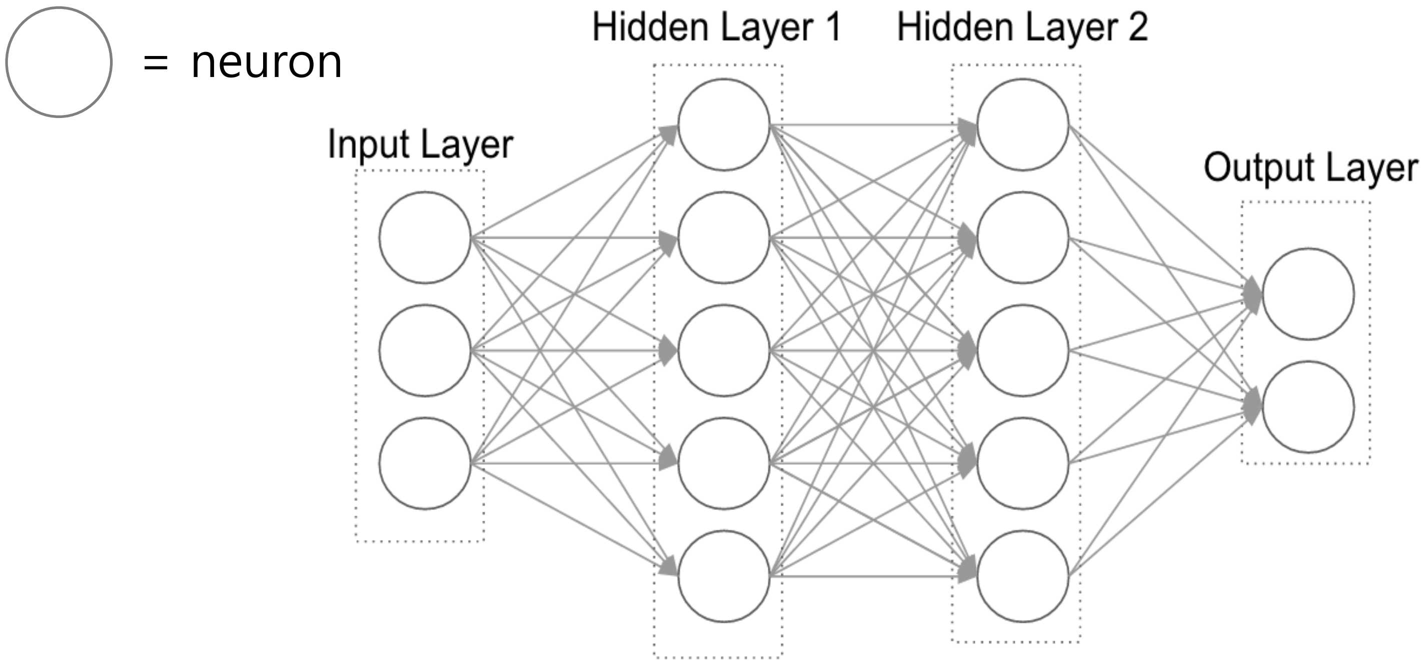


$$y = f(\sum_i w_i x_i + b)$$



$$\sum_i w_i x_i + b > 0 \Rightarrow y = 1$$

Neural Networks



Feedforward neural network with 2 hidden layers

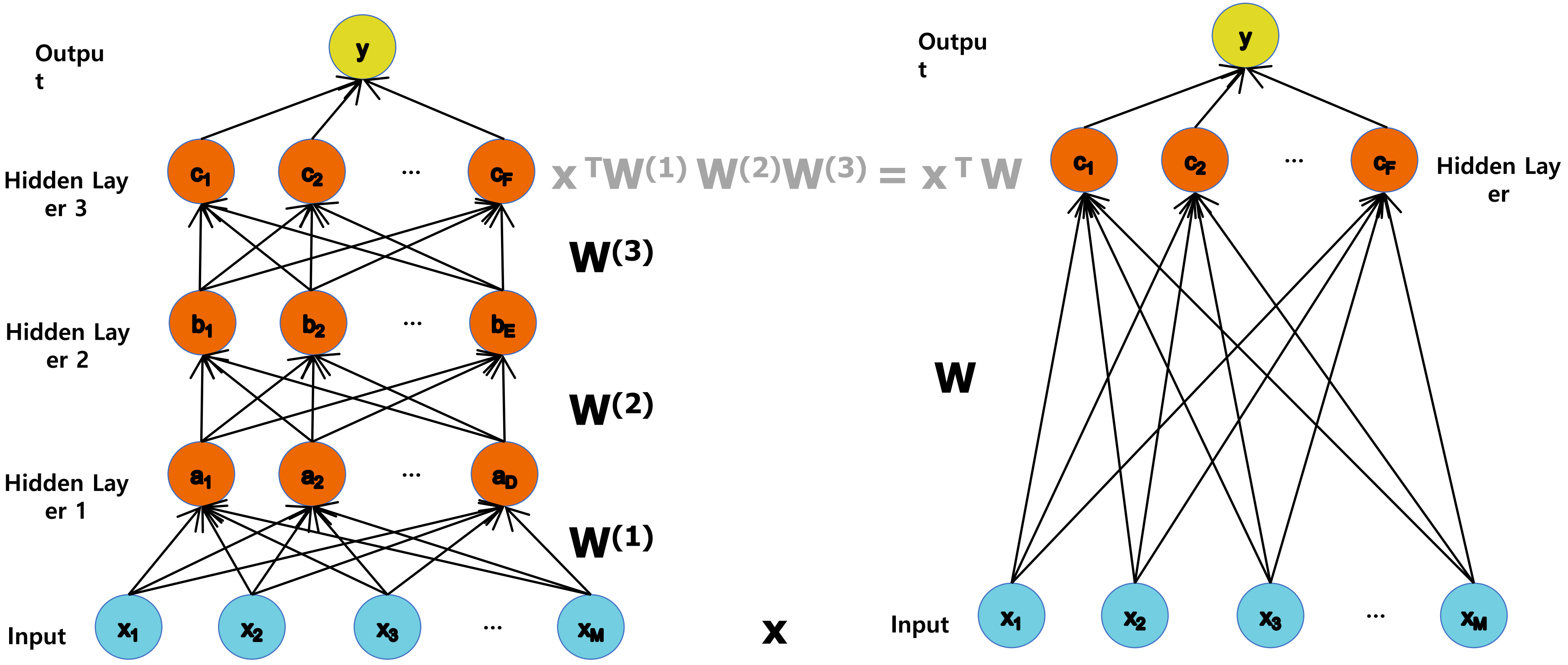
http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/

input dimension = # input neurons (3 dim in the fig.)
output dimension = # output neurons (2 dim in the fig.)

- Each layer outputs activation functions composed with the linear combinations of its inputs.
- The neurons are all parallel in the same layers.

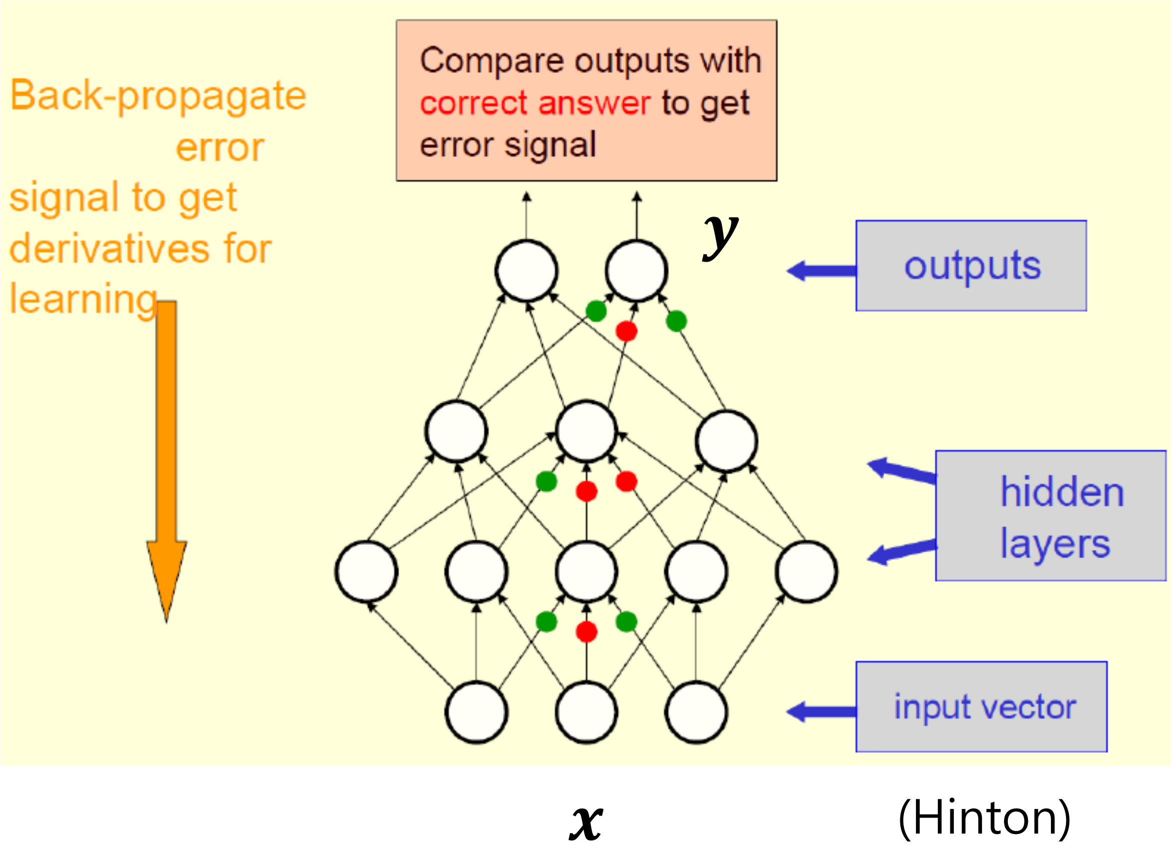
Linear hidden layers can be reduced to a single layer

Example: In case of the activation function f = Id (identity), all hidden layers can be reduced to one hidden linear layer. Then the data will be underfitted. To better fit, nonlinear activation function f will be needed.



$$x^T W^{(1)} W^{(2)} W^{(3)} = x^T W$$

Deep Learning via Deep Neural Network (DNN)



Back-propagate error signal to get derivatives for learning

Compare outputs with correct answer to get error signal

outputs

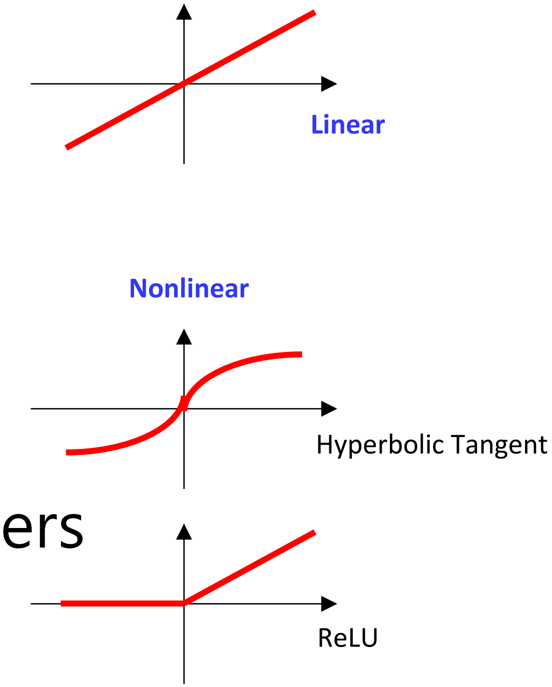
hidden layers

input vector

(Hinton)

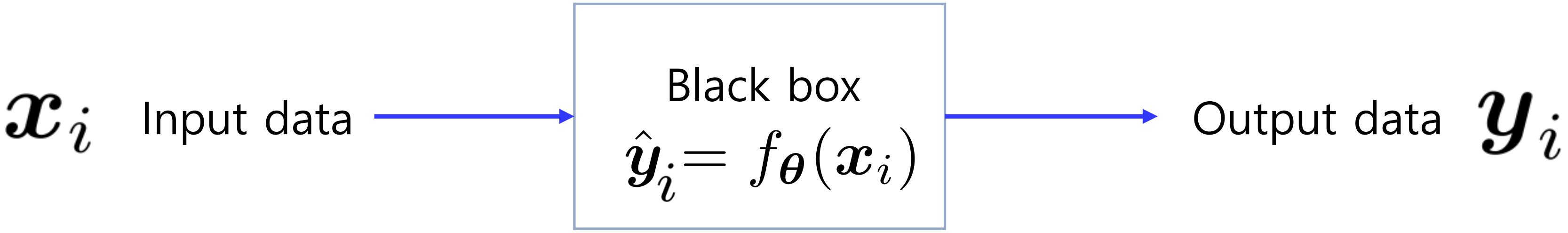
Back propagation performs Gradient descent search over a vector space

Example of Activation functions



Deep network if the number of hidden layers is greater than or equal to **2**.

Learning via Neural Networks = Find $f_{\boldsymbol{\theta}}$



$\boldsymbol{x}_i$ Input data

Black box

$\hat{\boldsymbol{y}}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$

Output data $\boldsymbol{y}_i$

1. *Given training data,* $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{N}$
2. *Choose Output function,* $\hat{\boldsymbol{y}}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$
3. *Choose Objective (Loss) function,* $\ell(\hat{\boldsymbol{y}}_i, \boldsymbol{y}_i) \in \mathbb{R}$
4. *Find the minimizer such that*

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

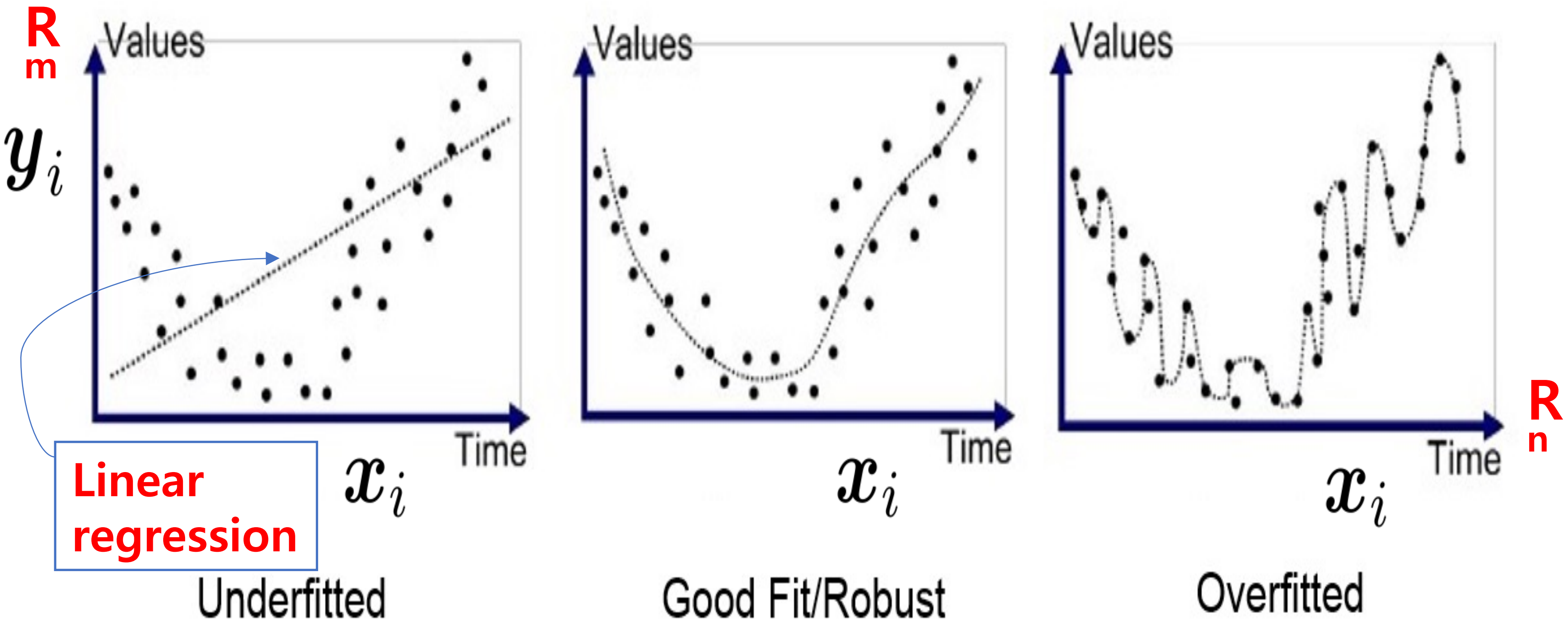
5. *Train with Stochastic Gradient Descent*

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

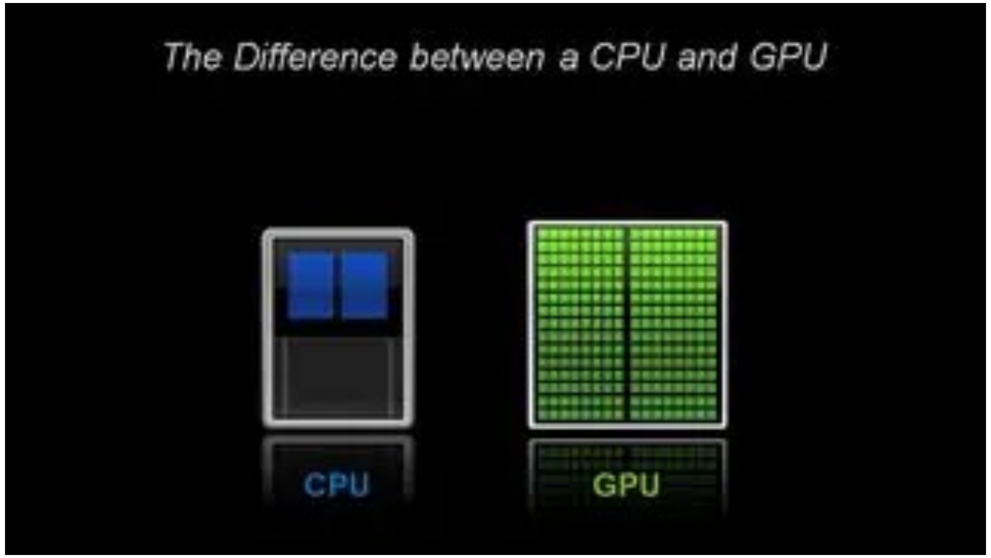
Underfit, Overfit

Learning function from Data = Finding $\hat{\boldsymbol{y}}_i = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$

(training dataset: input=x_i, output=y_i)



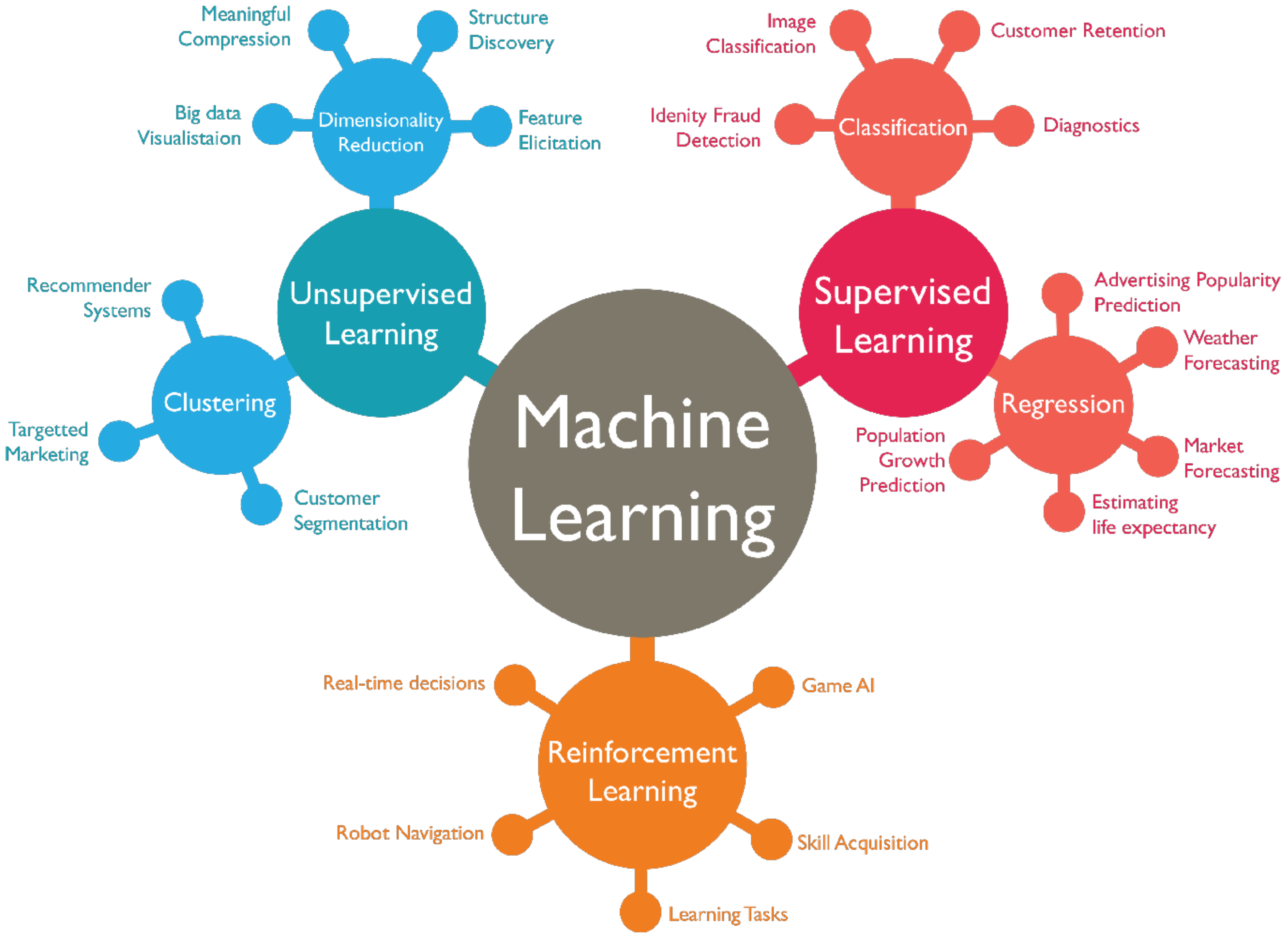
CPU vs GPU



https://blogs.nvidia.com/blog/2009/12/16/whats-the-difference-between-a-cpu-and-a-gpu/

CPU	GPU
Central Processing Unit	Graphics Processing Unit
Several cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a handful of operations at once	Can do thousands of operations at once

Types of Machine Learning



https://medium.com/intro-to-artificial-intelligence/reinforce-a-policy-gradient-based-reinforcement-learning-algorithm-84bde440c816

Types of Machine Learning (I)

Supervised learning: The training data you feed to the algorithm includes the desired solutions, called labels.

Regression: Predict a target numeric value, such as the price of a car, given a
set of features (mileage, age, brand, etc.), or estimate the model parameters
(Linear models for regression: LS, ML and MAP approach, SVM, Decision Trees, Neural Networks)

Classification: Classify a given data. For example, a spam filter is trained with many
example emails (training set) along with their class (spam or ham; labels), and it
must learn how to classify new emails
(k-Nearest Neighbor, Logistic regression, SVM, Decision trees, Neural networks)

Regression

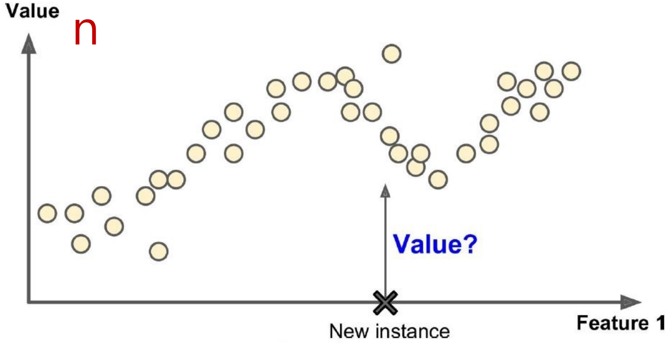


Figure 1-6. Regression

Classification

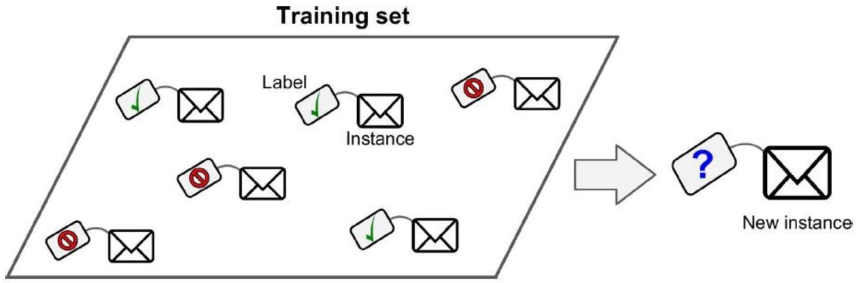
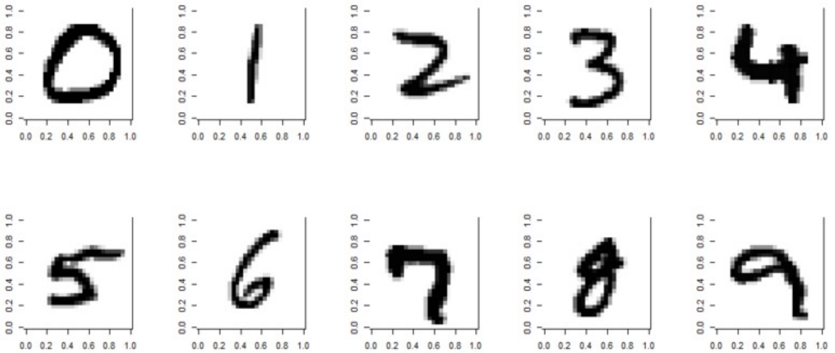


Figure 1-5. A labeled training set for supervised learning (e.g., spam classification)

Supervised Learning

Example: Hand-written digits (MNIST)



vectorization

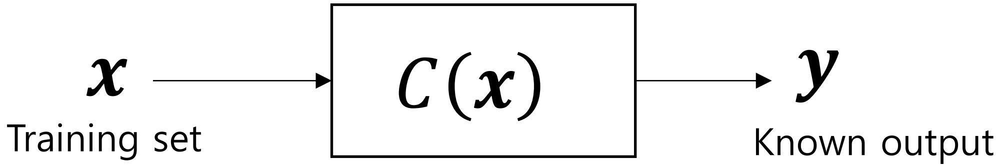
$\Rightarrow$

$$S_x = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \dots, \boldsymbol{x}_9\}$$

Feature vectors

- Machine Learning (Supervised)

$C(\boldsymbol{x})$ is called a **classifier**



- Design $C(\boldsymbol{x})$ so that the output becomes the same with the known output corresponding to each training (feature) vector

(Supervised Learning with **labeled** training vectors)

Types of Machine Learning (II)

Unsupervised learning: The training data is unlabeled. The system tries to learn without a teacher

Clustering: Detect groups of data with similar characteristics
(k-Means, Expectation maximization (EM), Hierarchical cluster analysis)

Dimensionality reduction: Simplify the data without loosing much information
(Principal component analysis (PCA))

Reinforcement learning: The learning system (agent) observes the environment, selects and performs actions, and get rewards/penalties in return. Then it learns by itself what is the best strategy (policy), to get the most reward over time

Example: DeepMind’s AlphaGo, and algorithms for robot walking

Clustering

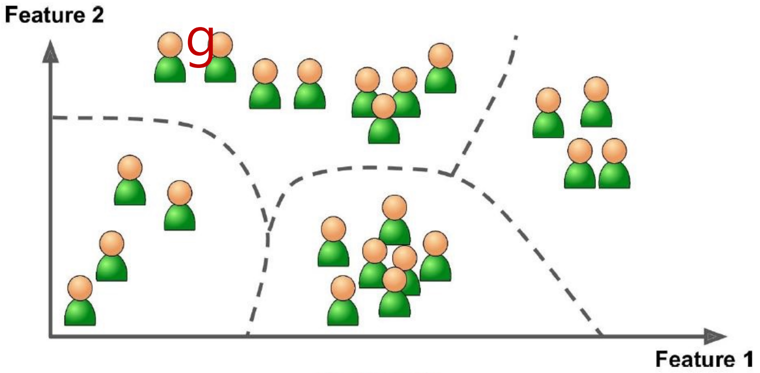
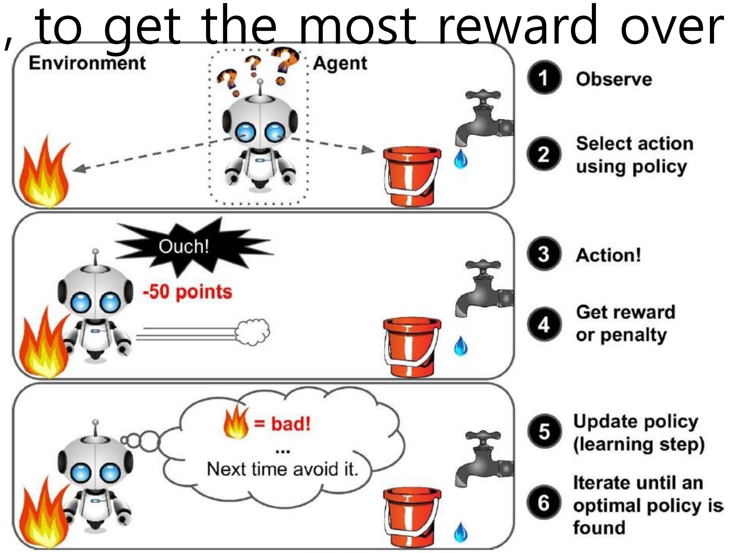


Figure 1-8. Clustering



Environment
Agent
Ouch!
-50 points
= bad!
...
Next time avoid it.
1 Observe
2 Select action using policy
3 Action!
4 Get reward or penalty
5 Update policy (learning step)
6 Iterate until an optimal policy is found

Figure 1-12. Reinforcement Learning

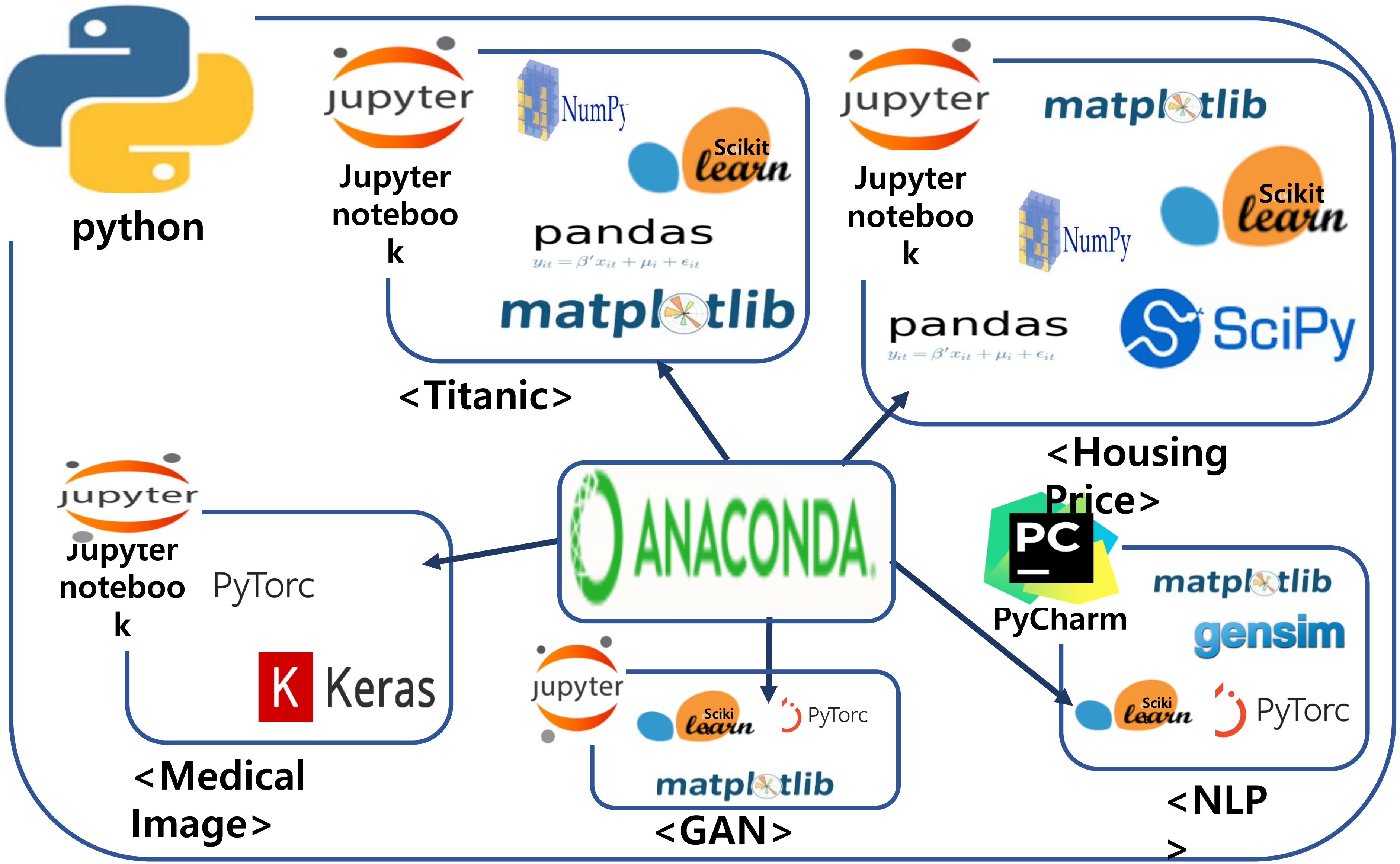
SW Tools for Machine Learning (I)

Language	**python**	• The basic programming development language • Intuitive, friendly and easy for use
Development Program	**Jupyter notebook**	• Can be executed in block unit • Internet connection required
	PyCharm	• File structure views and quick jumping between files • Python development for Google app engine
Package Management	**Anaconda**	• Aims to simplify package installation and management • Manages package groups in an independent environment - Preventing collisions due to package-specific compatibility issues

SW Tools for Machine Learning (II)

Package (Library)	**Pytorch**	PyTorc	• Developed for deep learning neural networks • Developed by Facebook
	Keras	K Keras	• Designed to enable fast experimentation with deep neural networks • Focuses on being user-friendly, modular, and extensible.
	Numpy	NumPy	• Specialized for matrix operations.
	Pandas	pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$	• Specialized for data processing • e.g., merging, deleting, reshaping
	Scikit-Learn	Scikit learn	• Includes various machine learning algorithms • e.g., regression, clustering
	matplotlib	matplotlib	• Specialized for graph plotting
	gensim	gensim	• Robust open-source for vector space modeling • Uses Numpy, SciPy and optionally Cython for performance

SW Tools for Machine Learning Projects



python

Jupyter noteboo k

<Titanic>

Jupyter noteboo k

<Housing Price>

Jupyter noteboo k

PyTorc

<Medical Image>

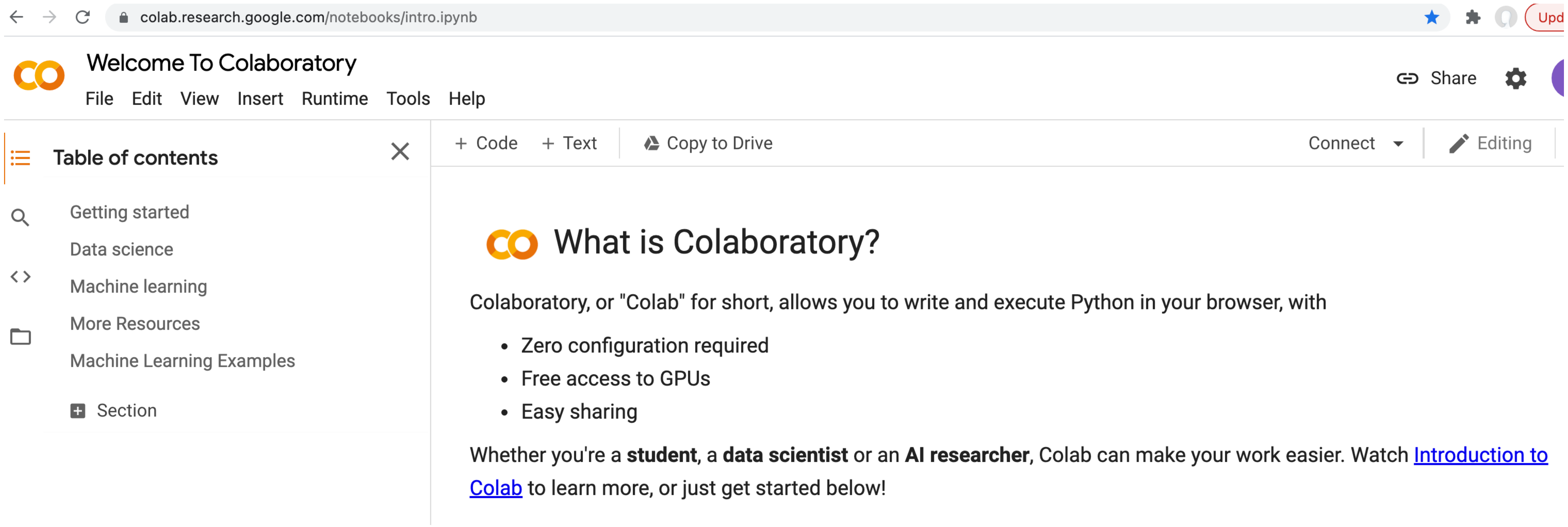
<GAN>

PyCharm

<NLP >

Google colab

https://colab.research.google.com/



What is Colaboratory?

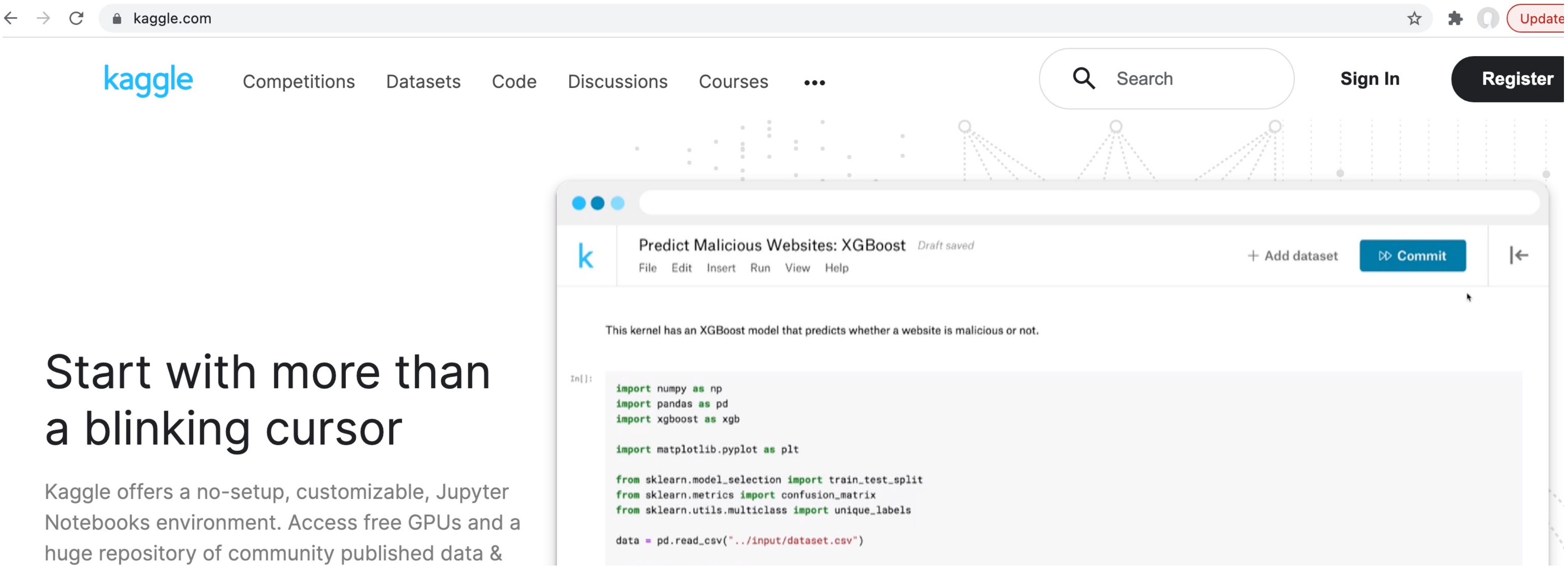
Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser, with

- Zero configuration required
- Free access to GPUs
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier. Watch Introduction to Colab to learn more, or just get started below!

Kaggle

https://www.kaggle.com/



Predict Malicious Websites: XGBoost *Draft saved*

Start with more than a blinking cursor

Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access free GPUs and a huge repository of community published data &

This kernel has an XGBoost model that predicts whether a website is malicious or not.

```
In[]:
import numpy as np
import pandas as pd
import xgboost as xgb

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.utils.multiclass import unique_labels

data = pd.read_csv("../input/dataset.csv")
```

Software Tools for Machine Learning

1. Python Guide
   1. Jump to Python (Korean) https://wikidocs.net/book/1
   2. A Byte of Python (English) https://python.swaroopch.com/modules.html

2. Pytorch Guide
   1. Pytorch website (English) https://pytorch.org/tutorials/

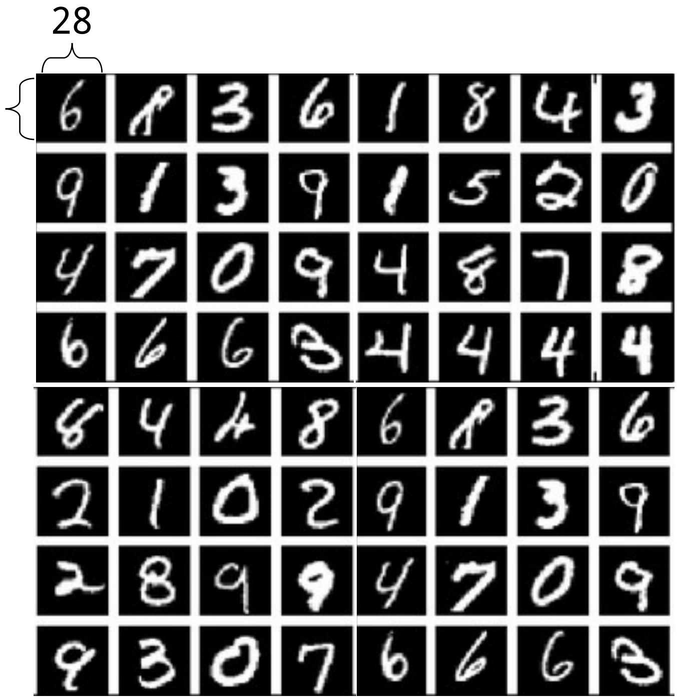
3. Numpy Guide
   1. Scipy website (English) https://docs.scipy.org/doc/numpy-1.14.0/search.html

4. Matplotlib Guide
   1. Matplotlib website (English) https://matplotlib.org/

Dataset

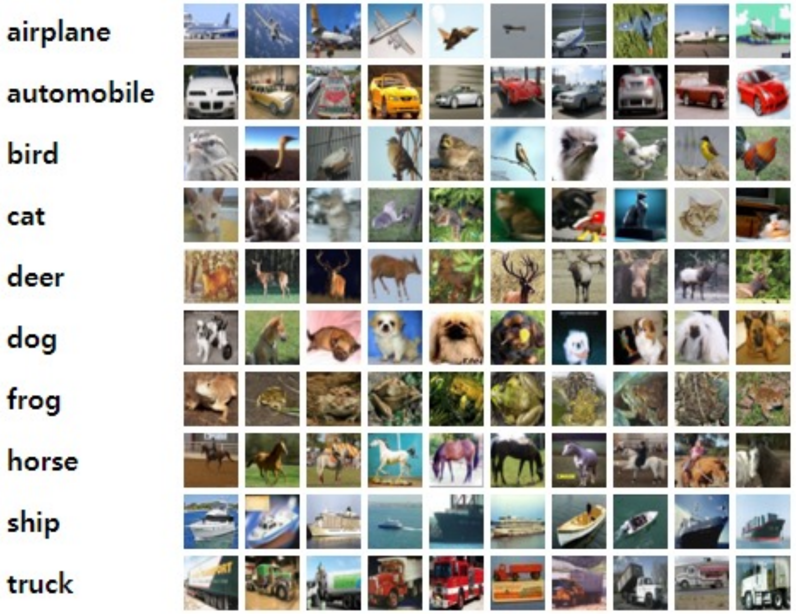
MNIST dataset

- Hand written digit
- 28x28 pixels, 70,000 images
- http://yann.lecun.com/exdb/mnist/



CIFAR - 10

- Color images in 10 classes
- 32x32 pixels, 60,000 images
- https://www.cs.toronto.edu/~kriz/cifar.html



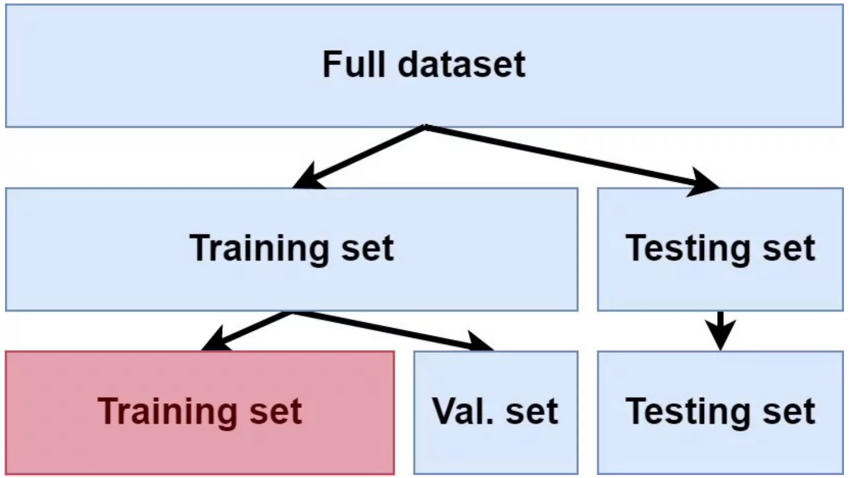
Popular open data repositories

UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/)

Kaggle (http://www.kaggle.com/datasets)

Amazon's AWS datasets (http://aws.amazon.com/fr/datasets/)

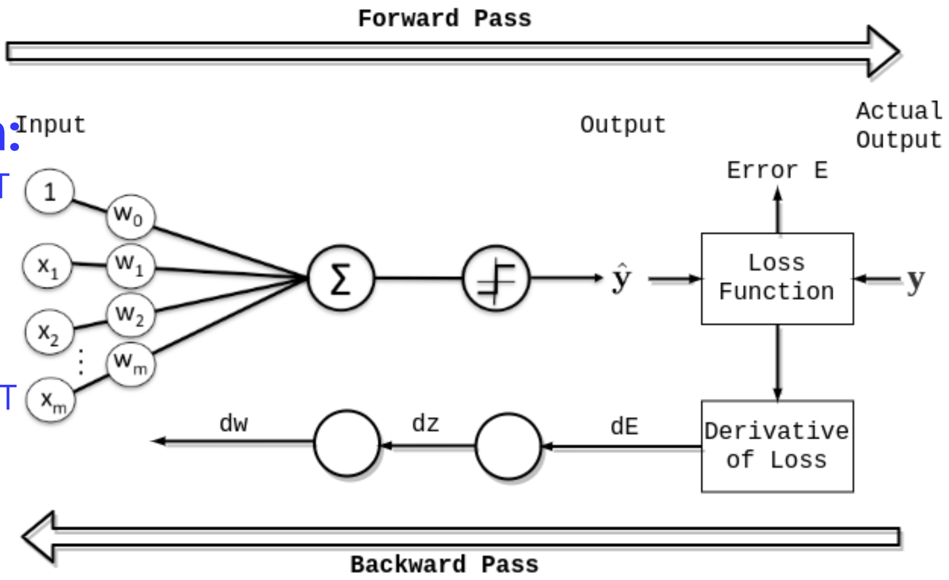
Dataset: batch size, iterations, epochs



Training set $(x^{(i)}, y^{(i)})$, $i=1,2,....,N$

Input neuron: $\mathbf{x}=(x_1,x_2,...,x_m)^T$

Input data: $\mathbf{x}^{(\mathbf{i})}=(x_1^{(i)},x_2^{(i)},...,x_m^{(i)})^T$



(Backpropagation)

Output neuron: y

Output data: y $^{(\mathbf{i})}$