Introduction to Machine Learning

Christophe Rappold

IEM – CSIC
GSI Helmholtz Centre for Heavy Ion Research – Darmstadt, Germany

10 May 2021

What is Machine Learning ?

A subset of artificial intelligence in the field of computer science that often uses statistical techniques to give computers the ability to "learn"with data, without being explicitly programmed

Samuel Arthur –1959 –ML in Checkers

- **Definition "to learn" from dictionary:**

  "Gain knowledge or understanding of, or skill in by study, instruction or experience"

  - Learning a set of new facts
  - Learning *how* to do something
  - Improving ability of something already learned

What is Machine Learning ?

- **Why learning ?**
  - Machine learning is programming computers to optimize a performance criterion using example data or past experience
  - Learning is used when :
    - Human expertise does not exist
    - Humans are unable to explain their expertise
    - Amount of knowledge is too large for explicit encoding
    - Solution changes in time
    - Relationships can be hidden within large amounts of data
    - Solution needs to be adapted to particular cases
    - New knowledge is constantly being discovered by humans

The automatic extraction of semantic information from raw signal is at the core of many applications (object recognition, speech processing, natural language processing, planning, etc).

Can we write a computer program that does that?

- **The (human) brain is so good at interpreting visual information that the gap between raw data and its semantic interpretation is difficult to assess intuitively:**

This is a mushroom.

This is a mushroom.

In [1]:
```python
from matplotlib.pyplot import imread
imread("mushroom-small.png")
```

Out[1]:
```
array([[[0.03921569, 0.03529412, 0.02352941, 1.        ],
        [0.2509804 , 0.1882353 , 0.20392157, 1.        ],
        [0.4117647 , 0.34117648, 0.37254903, 1.        ],
        ...,
        [0.20392157, 0.23529412, 0.17254902, 1.        ],
        [0.16470589, 0.18039216, 0.12156863, 1.        ],
        [0.18039216, 0.18039216, 0.14117648, 1.        ]],

       [[0.1254902 , 0.11372549, 0.09411765, 1.        ],
        [0.2901961 , 0.2509804 , 0.24705882, 1.        ],
        [0.21176471, 0.2       , 0.20392157, 1.        ],
        ...,
        [0.1764706 , 0.24705882, 0.12156863, 1.        ],
        [0.10980392, 0.15686275, 0.07843138, 1.        ],
        [0.16470589, 0.20784314, 0.11764706, 1.        ]],

       [[0.14117648, 0.12941177, 0.10980392, 1.        ],
        [0.21176471, 0.1882353 , 0.16862746, 1.        ],
        [0.14117648, 0.13725491, 0.12941177, 1.        ],
        ...,
        [0.10980392, 0.15686275, 0.08627451, 1.        ],
        [0.0627451 , 0.08235294, 0.05098039, 1.        ],
        [0.14117648, 0.2       , 0.09803922, 1.        ]],

       ...,
```

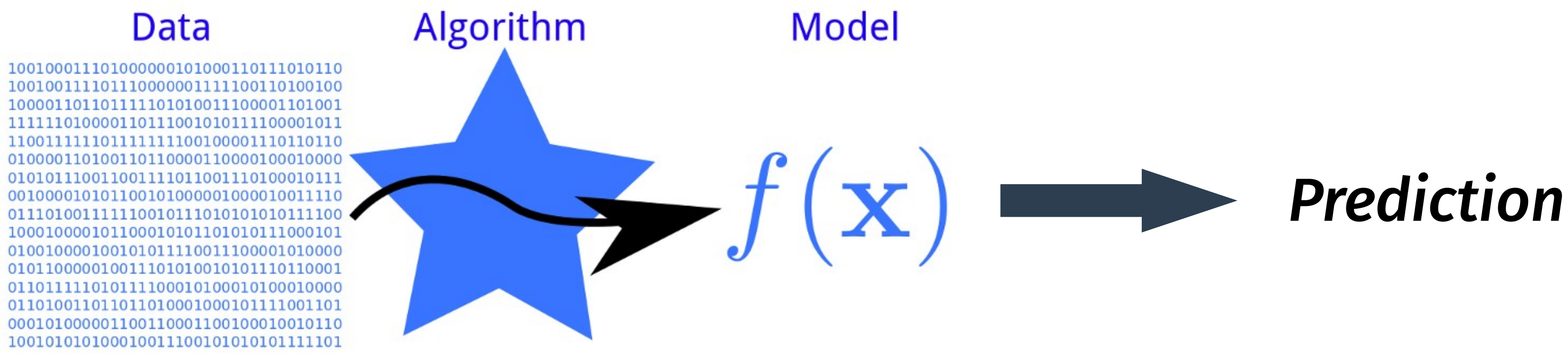
This is a mushroom.

- **Extracting semantic information requires models of high complexity.**
  - Cannot write a computer program that reproduces this process.
  - However, can write a program that learns the task of extracting semantic information.
- **A common strategy to solve this issue consists in:**
  - Defining a parametric model with high capacity
  - Optimizing its parameters by “making it work” on the training data

Learning → tuning the many parameters of the model

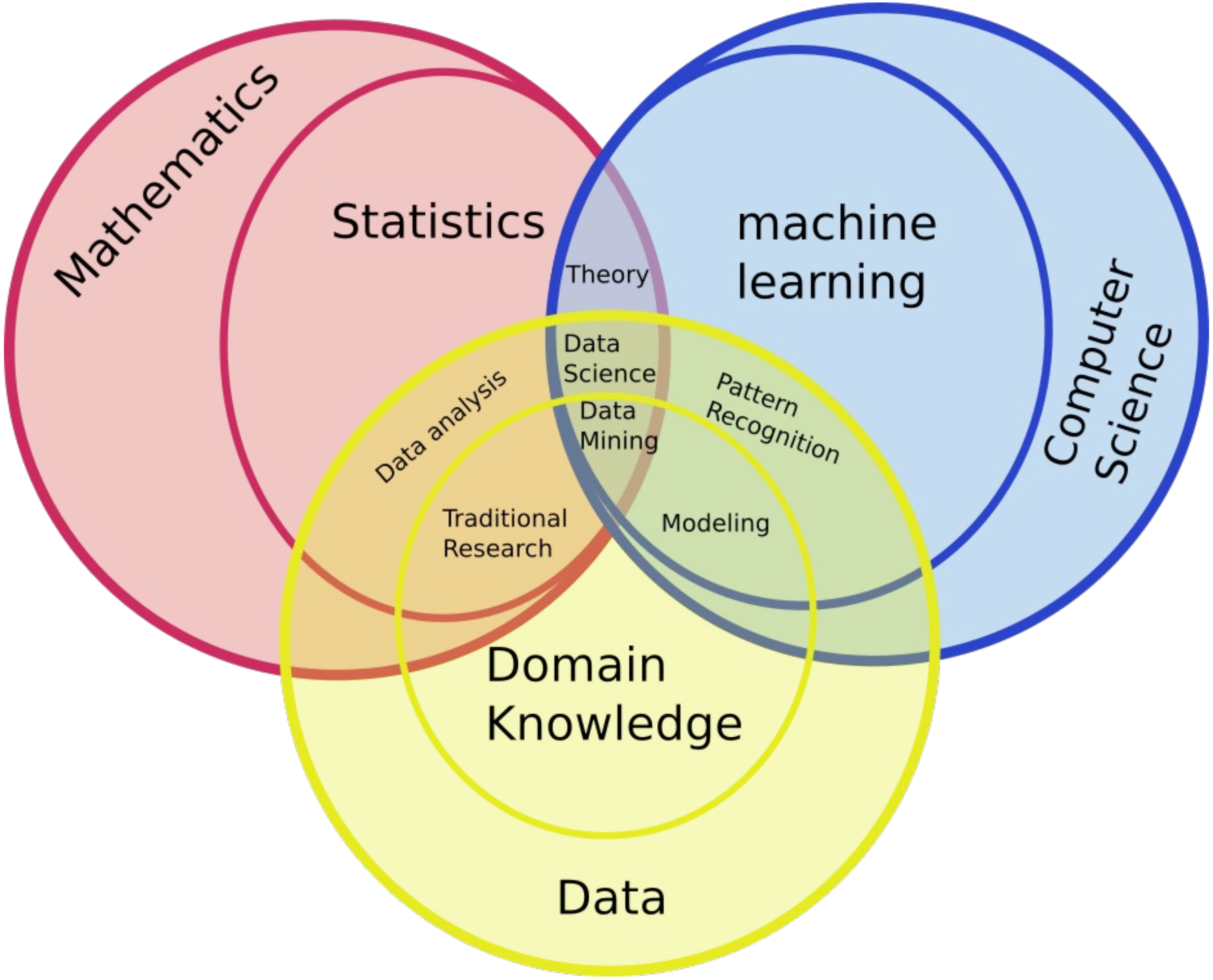
Machine learning is ...

- **Finding patterns or associations that can be used to make prediction**



- **ML is general term → many algorithms / methods**
- **Big Picture Goal : Learning useful *generalizations***

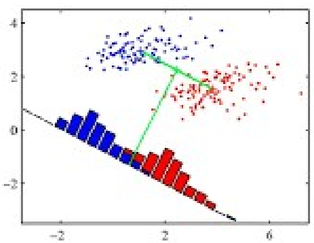
Fields cross sections



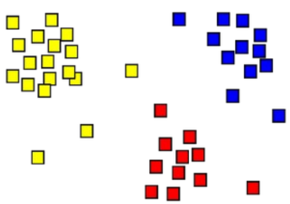
Statistics vs Machine Learning

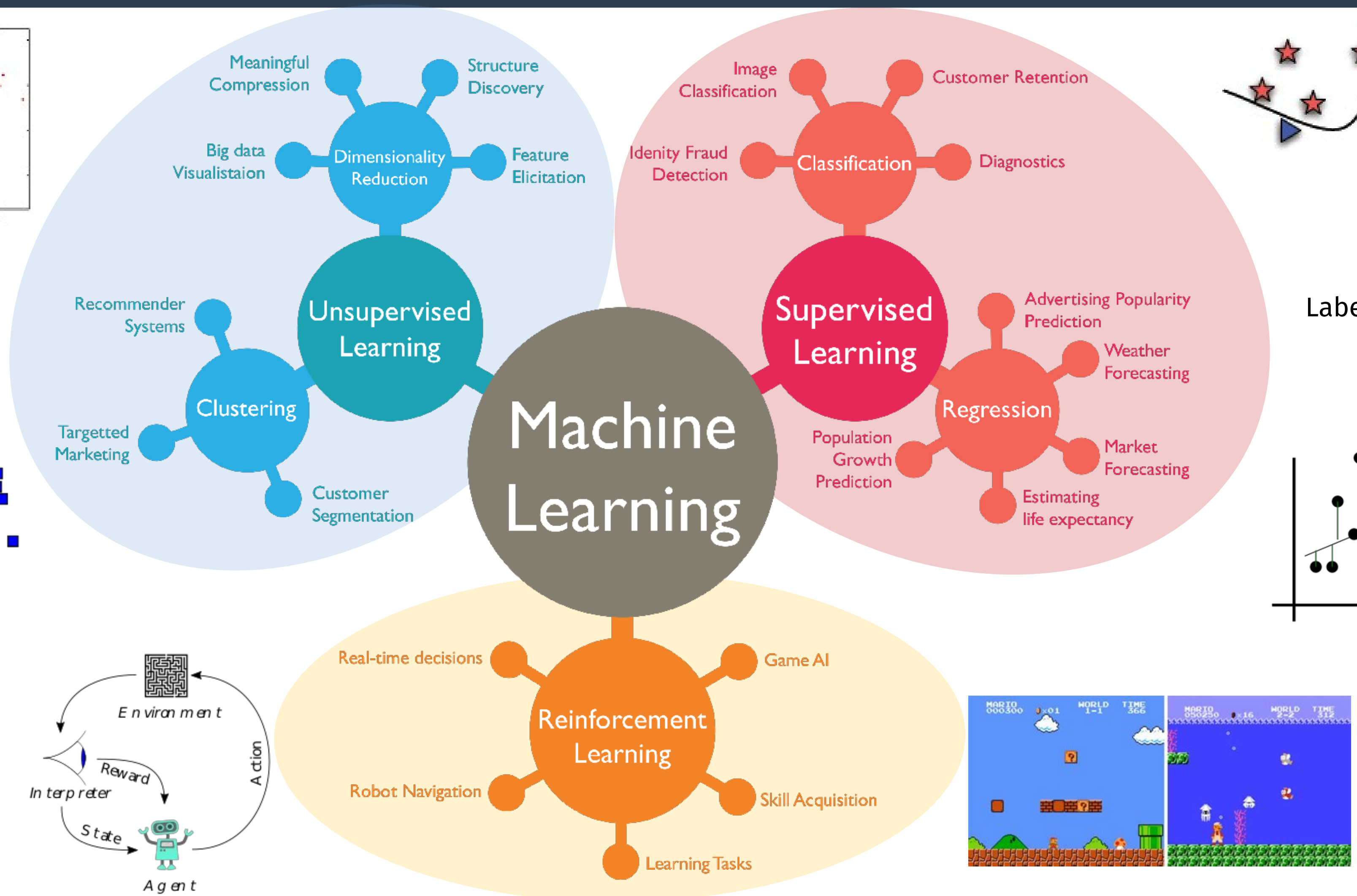
- **Largely overlapping fields:**
  - Both concerned with learning from data
  - Philosophical difference on ‘focus’ and ‘approach’.
- **Statistics:**
  - Founded in mathematics
  - Drawing valid conclusions based on analyzing existing data.
    - Making inference about a ‘population’ based on a ‘sample’
    - Tends to focus on fewer variables at once.
    - Precision and uncertainty are measures of model goodness.
- **Machine Learning:**
  - Founded in computer science
  - Focused on making predictions or seeking patterns (generalization).
    - Often considers a large number of variables at once.
    - Prediction accuracy to measure model goodness.

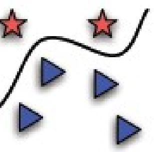
Types of Machine learning



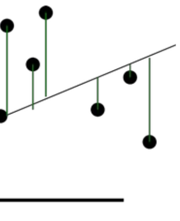
Unlabeled Data



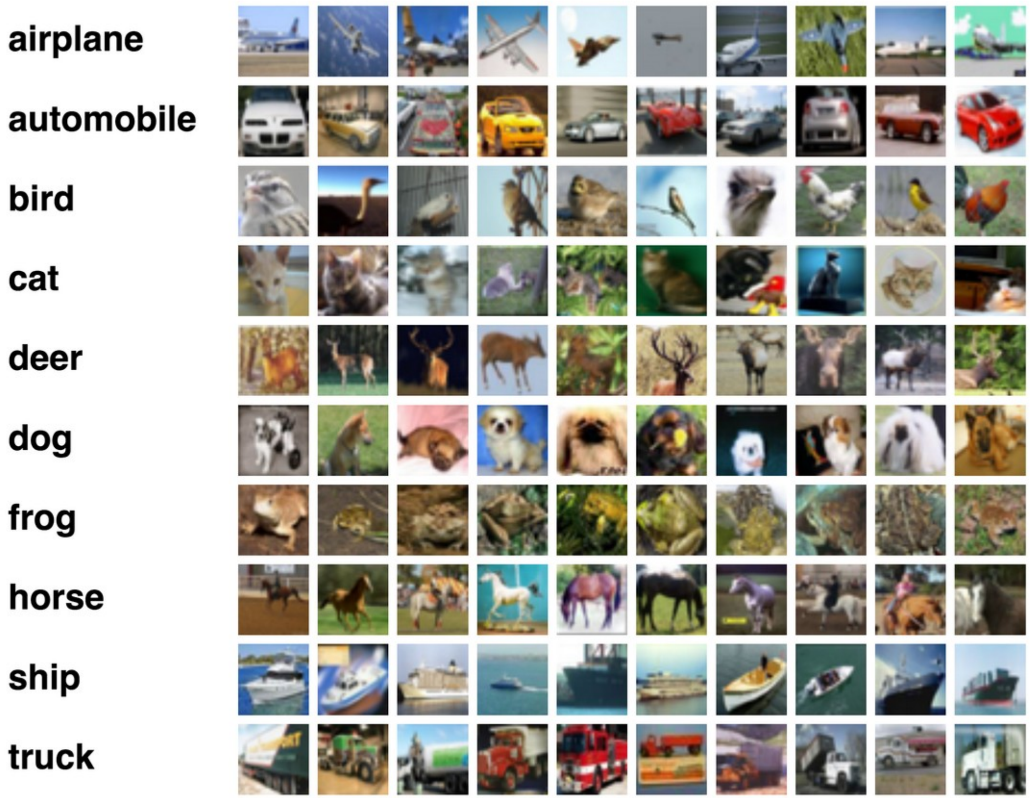




Labeled Data



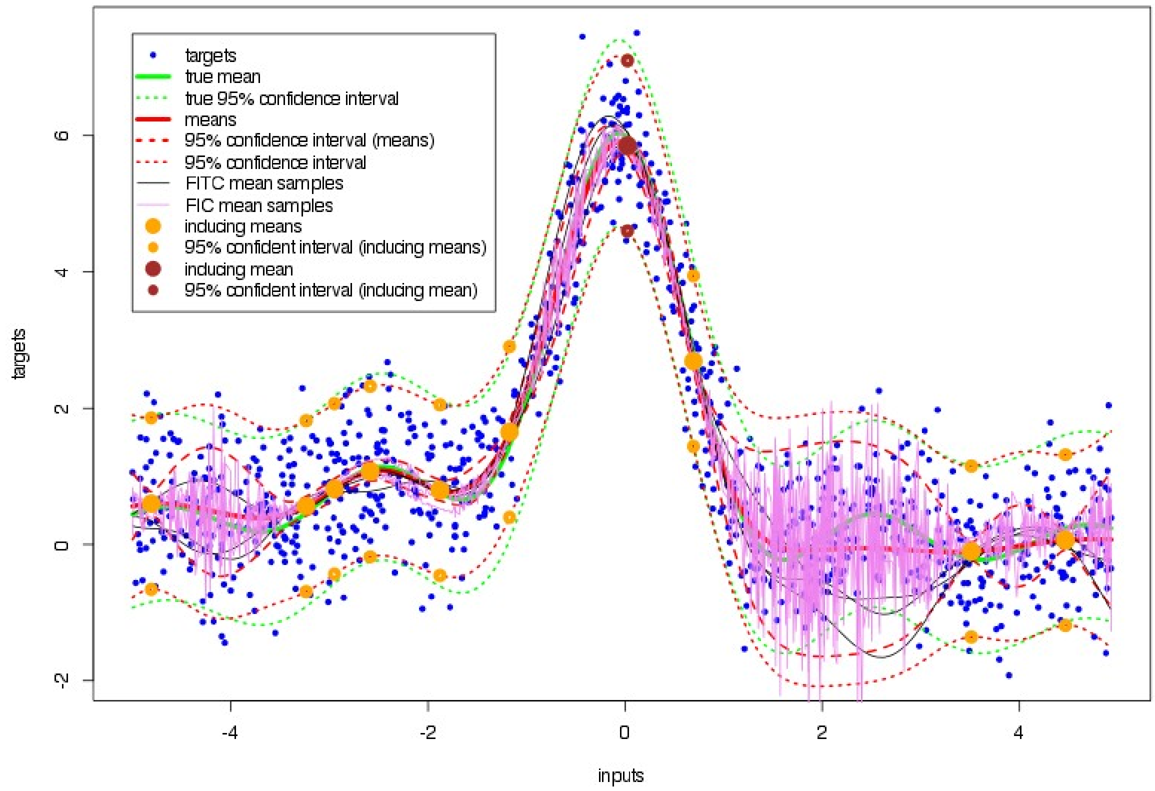
Some illustrative examples



Classification

CIFAR10 dataset (50k images 32x32x3)

Some illustrative examples



Regression

Some illustrative examples

Object detection and segmentation

K. He et al., *Mask R-CNN* (2017) arXiv:1703.06870

Some illustrative examples

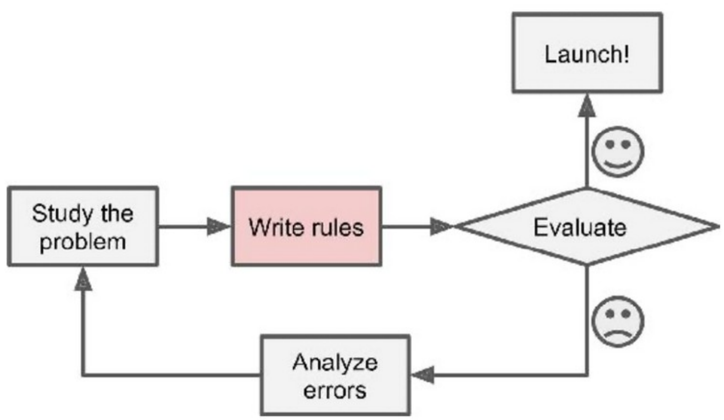
Human pose estimation
Y. Chen et al, *Adversarial PoseNet* (2017) arXiv:1705.00389

Some illustrative examples

Data generation
M. Arjovsky et al, *Wasserstein GAN*, (2017) arXiv:1701.07875

Example : Spam detection

- **Naive approach**
  - Observe what is a spam and detect recurrent patterns
  - write an algorithm of these patterns
  - If a new email contains these patterns then classify it as a spam
  - iterate until convergence

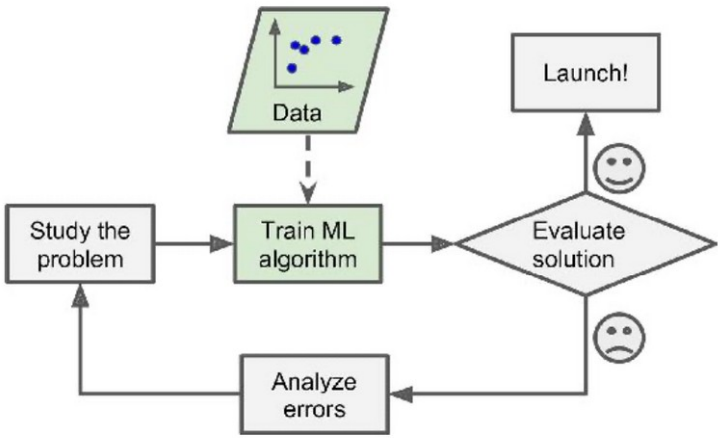


- **Complex task**
- **High nb of rules**
- **Difficult to update**

Example : Spam detection

- **Machine learning**

    1. A ML spam filter automatically learns relevant patterns

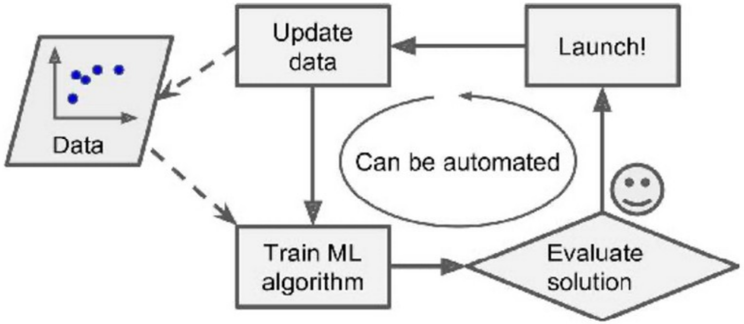


Example : Spam detection

- **Machine learning**

    1. A ML spam filter automatically learns relevant patterns

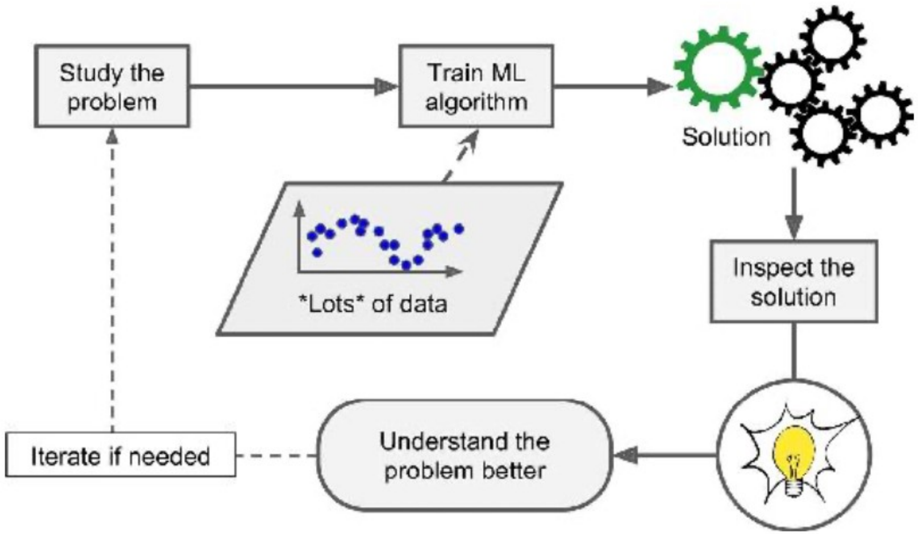
    2. Automatic adaptation



Example : Spam detection

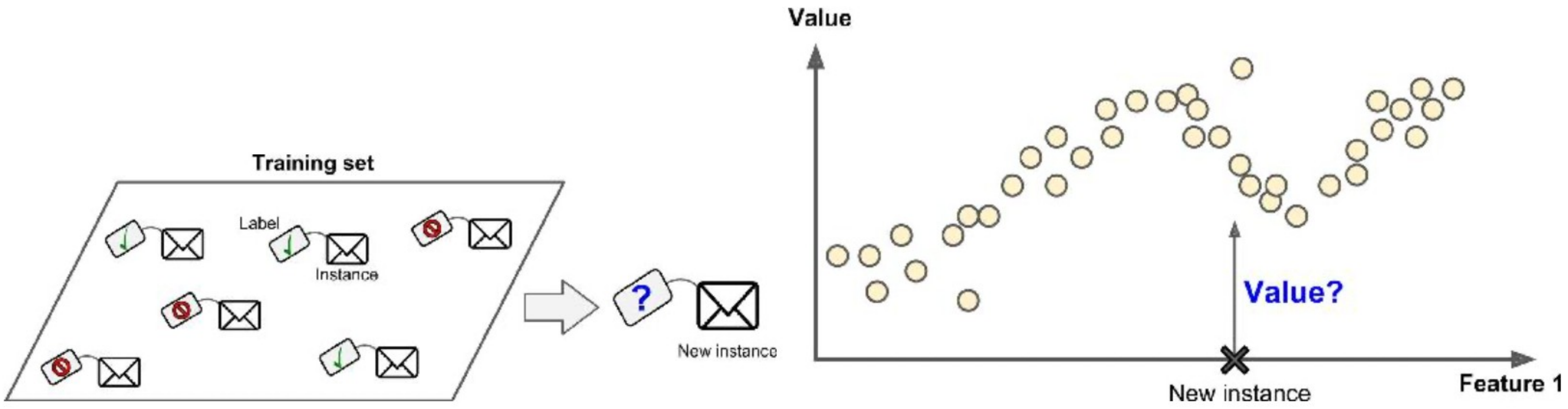
- **Machine learning**

    1. A ML spam filter automatically learns relevant patterns
    2. Automatic adaptation
    3. Can help humans to learn → Data Mining

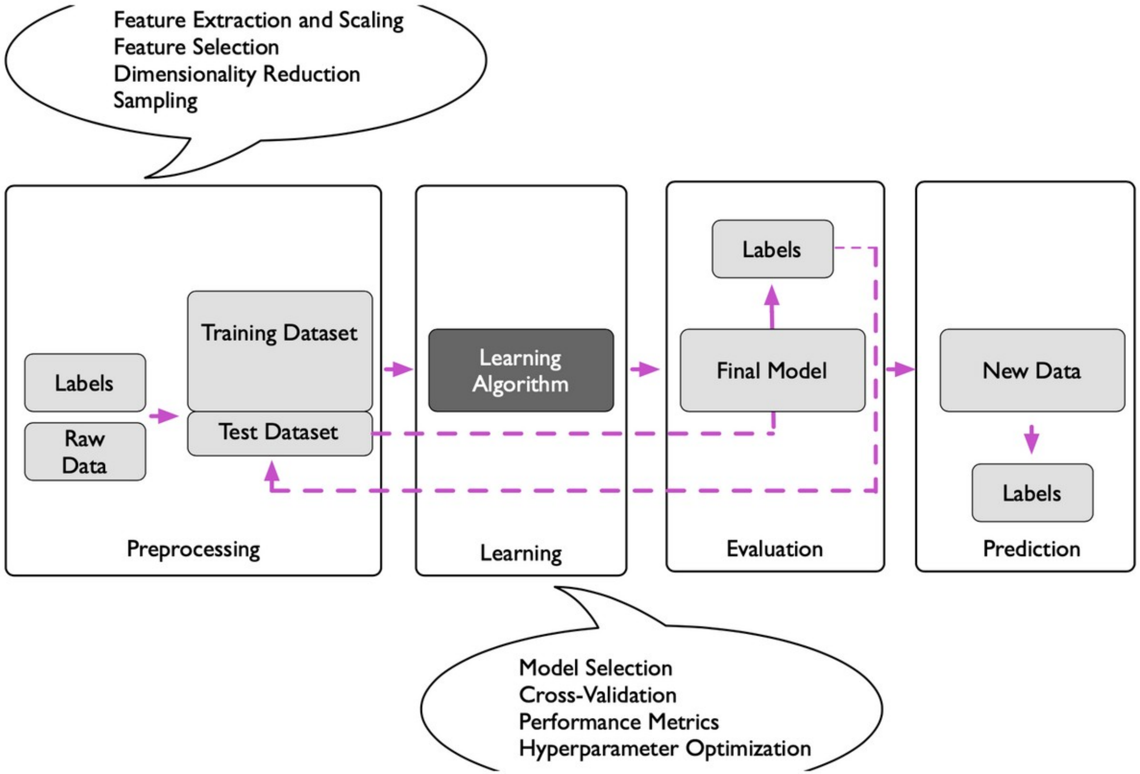


Supervised learning

- **Important aspects :**
  - Labeled data
  - Direct feedback
  - Predict outcome



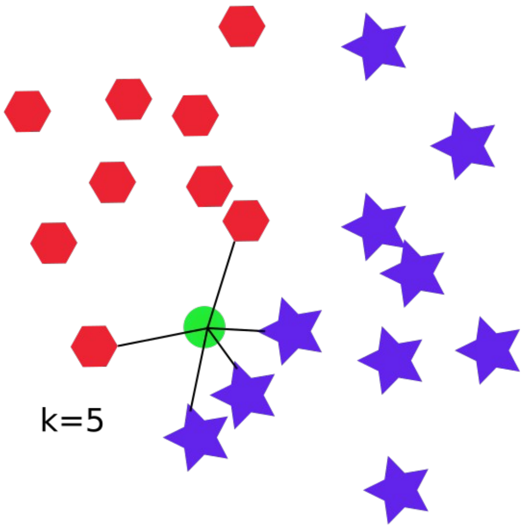
- **Workflow**



- Instance: *A specific observation of data.*
- Feature: *An measurable property of instance.*
- Criterion/Outcome: *The feature that you want to predict.*
- Model: *Representation or simulation of reality. Typically a simplification based on assumptions*

Supervised learning

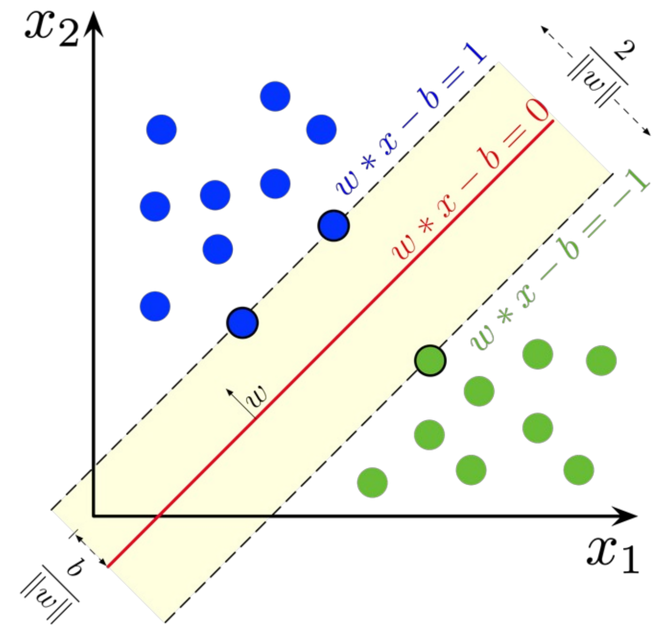
- **Main algorithms:**
  - K-nearest neighbors
    - Within the dataset take k nearest neighbors (with defined norm)
    - Each neighbor provide a class → vote
    - Most vote gives an estimate of the class of the new data



Supervised learning

- **Main algorithms:**
  - Support vector machine
    - Dataset : (x_i, y_i) with i=1…n and $y=\{-1,1\}$
    - Goal is to find hyperplane :

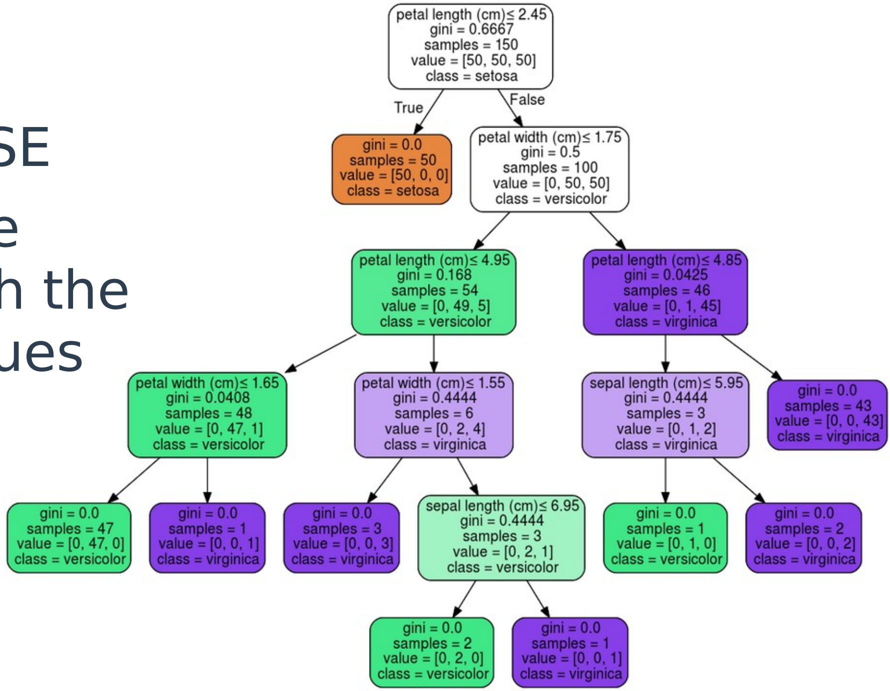
      $w^T x - b = 0$
    - Minimization : $||w||_2$ such that $y_i(w^T x_i - b) \geq 1$ for i=1…n
    - Classifier : $x \rightarrow \text{sgn}(w^T x - b)$



Supervised learning

- **Main algorithms:**
  - Decision Trees :
    - The criterion is modeled as a sequence of logical TRUE or FALSE
    - Recursively partitions the feature space such that the samples with the same labels or similar target values are grouped together.
    - Minimize the impurity:

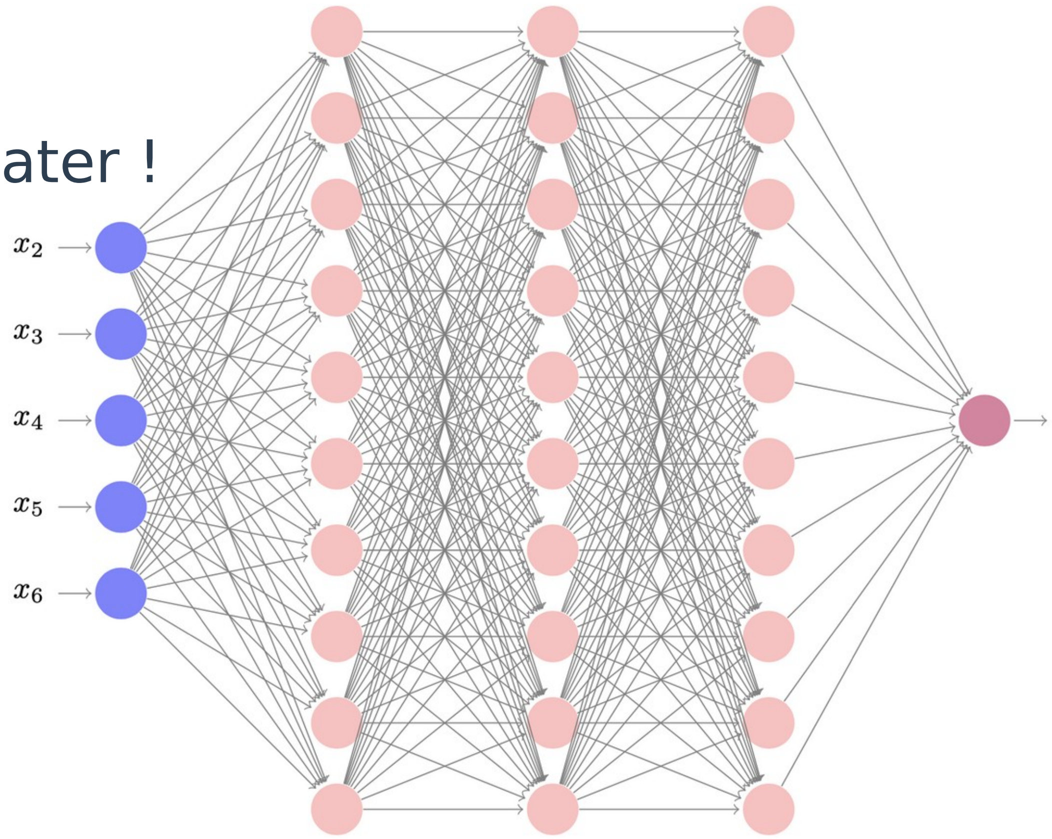
$$G = \frac{N^{left}}{N} H\left(Set_{left}\right) + \frac{N^{right}}{N} H\left(Set_{right}\right)$$



Supervised learning

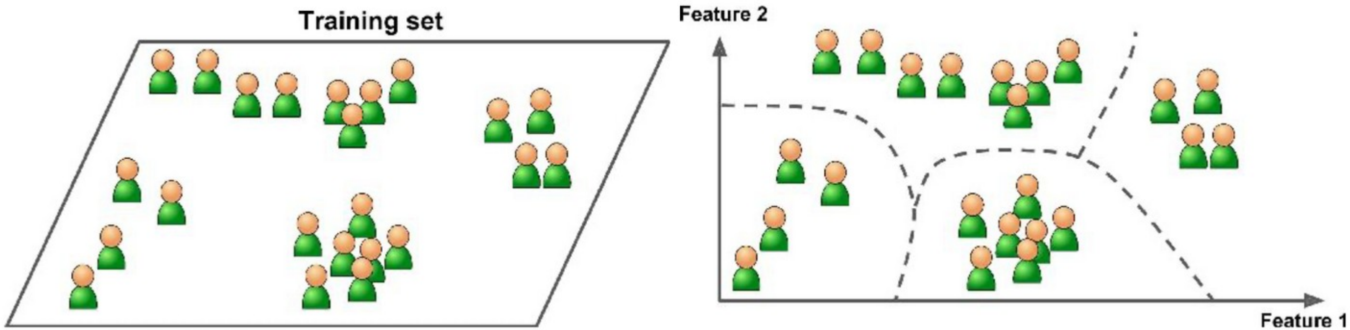
- **Main algorithms:**
  - Artificial neural network

    → be presented in details later !



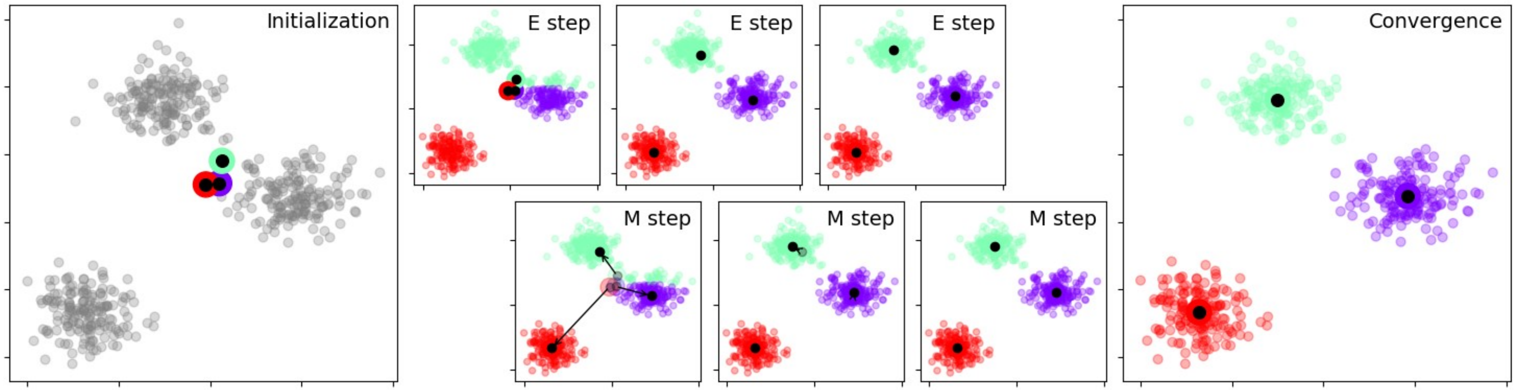
Unsupervised learning

- **Important aspects :**
  - No Labels or targets
  - No feedback
  - Find hidden structures



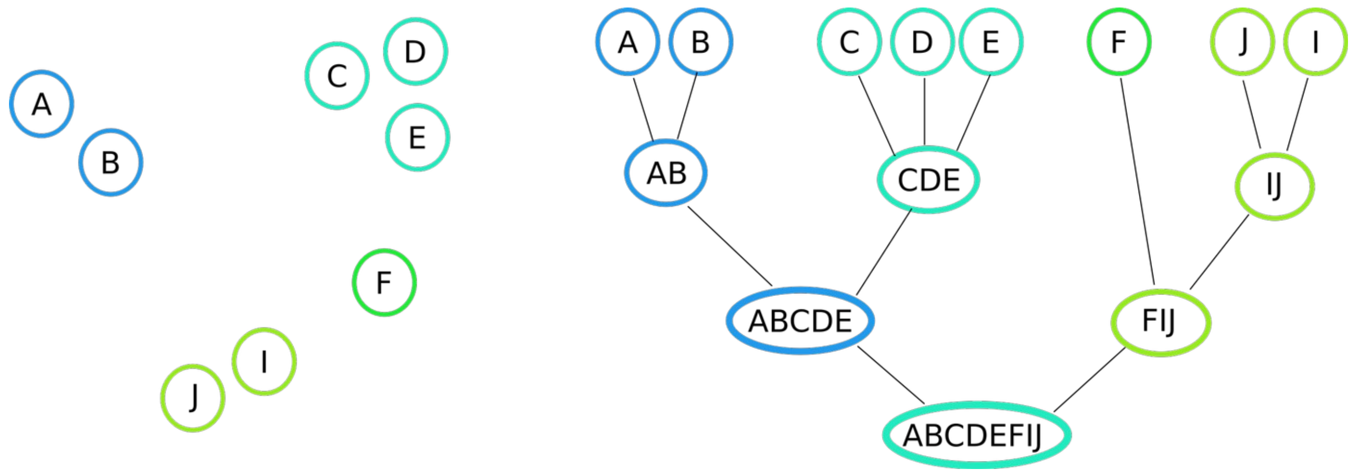
Supervised learning

- **Main algorithms:**
  - Clustering
    - K-means and variants
      - Partition N obs into K-cluster
      - Minimization of the within-cluster sum-of-squares criterion: $\sum_{i=0}^{n} \min_{\mu_j \in C_i} \left(||x_i - \mu_j||^2 \right)$
      - Iterative process by updating the centroid of each cluster



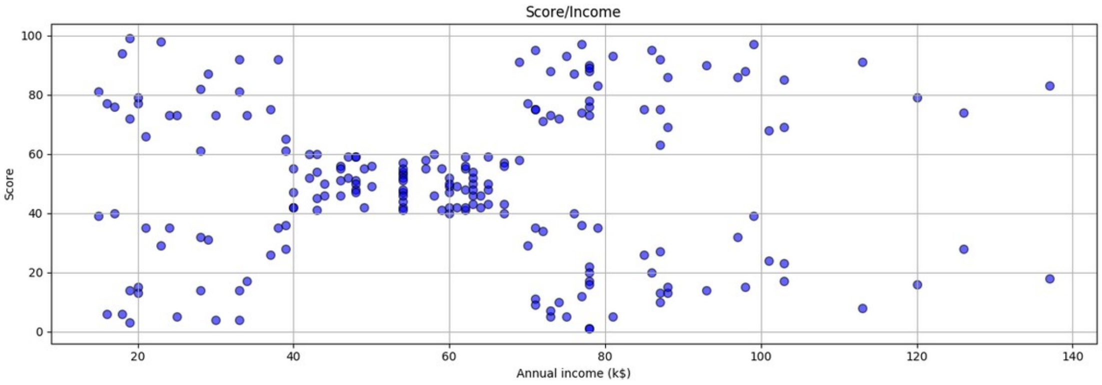
Unsupervised learning

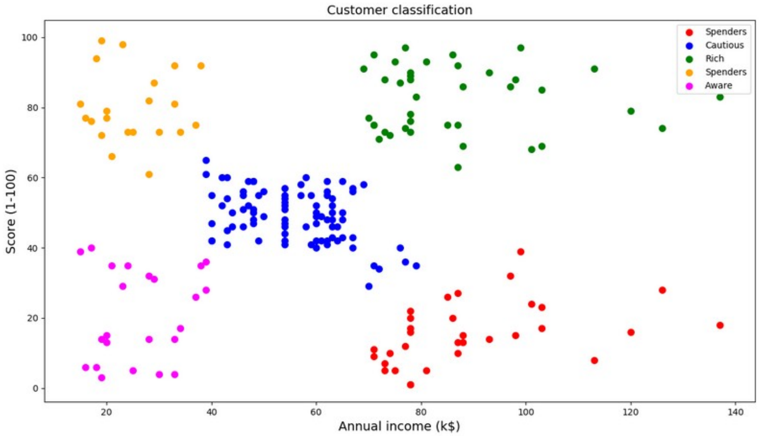
- **Main algorithms:**
  - Clustering
    - Hierarchical cluster analysis
      - Needs one metric ($||.||_2$)
      - linkage criteria: d between clusters as a function of the d between observations (complete-linkage clustering $max\{d(a,b):a\in A, b\in B\}$)

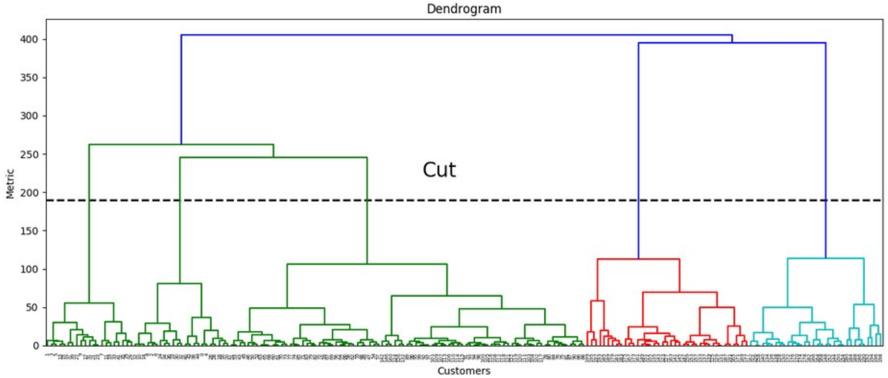


Unsupervised learning

- **Main algorithms:**
  - Clustering







Unsupervised learning

- **Main algorithms:**
  - Dimensionality reduction → Several aspects
    - high-dimensional datasets & the “curse of dimensionality”
      - When dimension UP, volume space unit hypercube UP, dataset become very sparse → problematic for statistics significance
        - 1D, unit interval & 100 uniformly distributed sample: distance spacing is 10^{-2}
        - 10D unit hypercube, for same lattice spacing needs 10^{20} samples.
    - Reduce dimension of dataset

      → Feature extraction: pre-processing steps for other algorithms

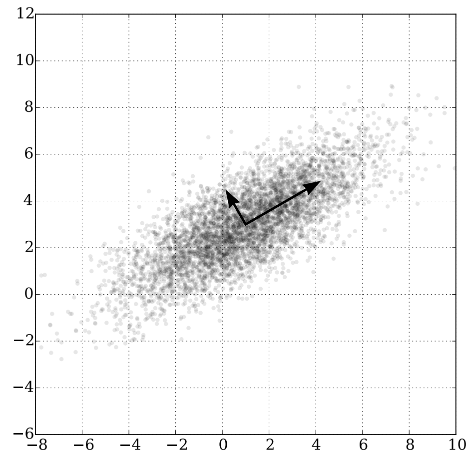
      → Data visualization: sometimes it is nice to also see the data

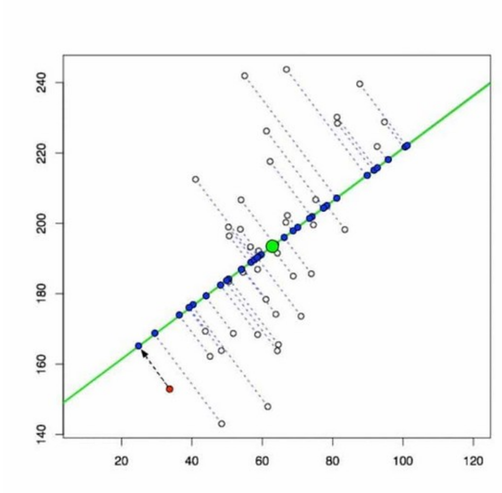
Unsupervised learning

- **Main algorithms:**
  - Dimensionality reduction
    - Principal component analysis:

      → Decompose a multivariate dataset in set of successive orthogonal components

      → In which a maximum amount of the variance.
    - Those are the eigenvector and eigenvalue of the covariance matrix of the dataset.





Reinforcement learning

- **Supervised Learning :** Explicit target signal of answer
- **Unsupervised Learning :** No answer
- **Reinforcement Learning :** No answer to a given task, but encourage the training through evaluation of agent's behavior

Agent: Subject

Reward: Evaluation
+ observation of the state

Action: manipulation

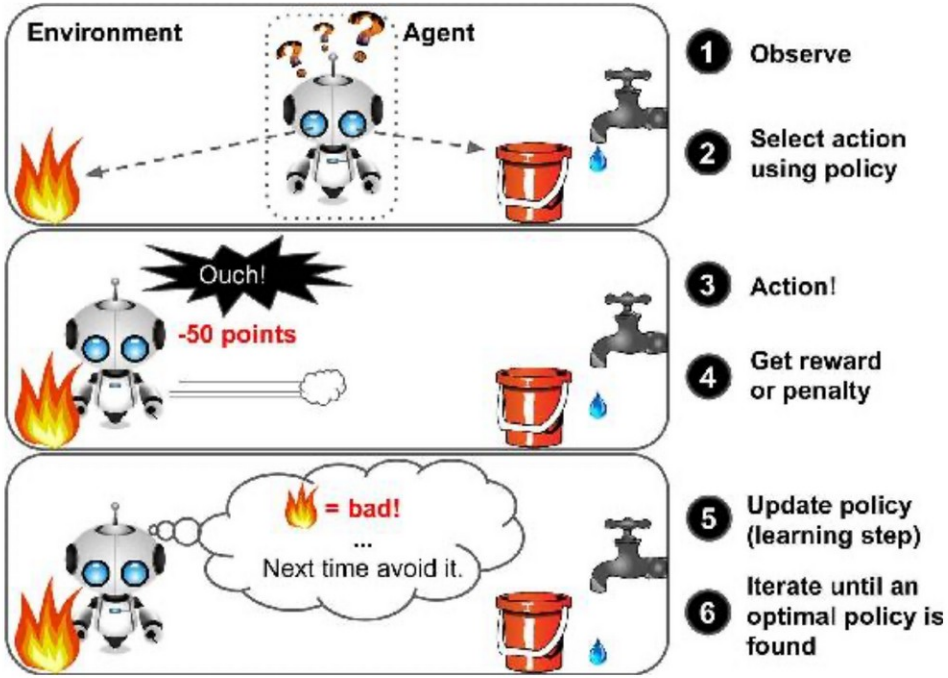
Environement : object

State change

Reinforcement learning

- **Reinforcement Learning :** No answer to a given task, but encourage the training through evaluation of agent’s behavior

  → Find the optimal policy: the strategy of the agent



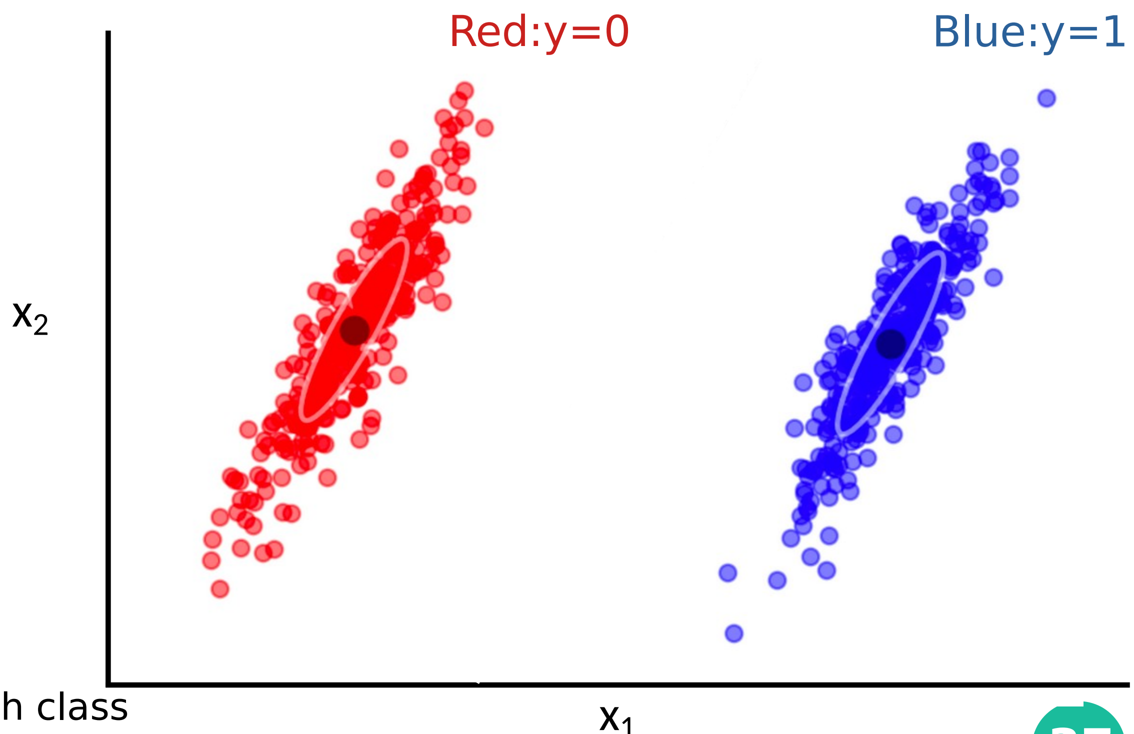
Logistic regression to neural network

- **Case : Separate dataset from 2 classes**
- **Data from joint distribution (X, y) ~ P(X,y)**
  - Features: $X \in \mathbb{R}^m$
  - Labels: $y \in \{0,1\}$
  - Joint distribution:

$$p(X, y) = p(x|y)\,p(y)$$

Likelihood function:
Distribution of the features
For a given class

Prior:
Probability of each class



Logistic regression to neural network

- **Separating classes → Predict the class of a point x:**

$$p(y=1|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=1)\,p(y=1)}{p(\boldsymbol{x})}$$

Bayes rule

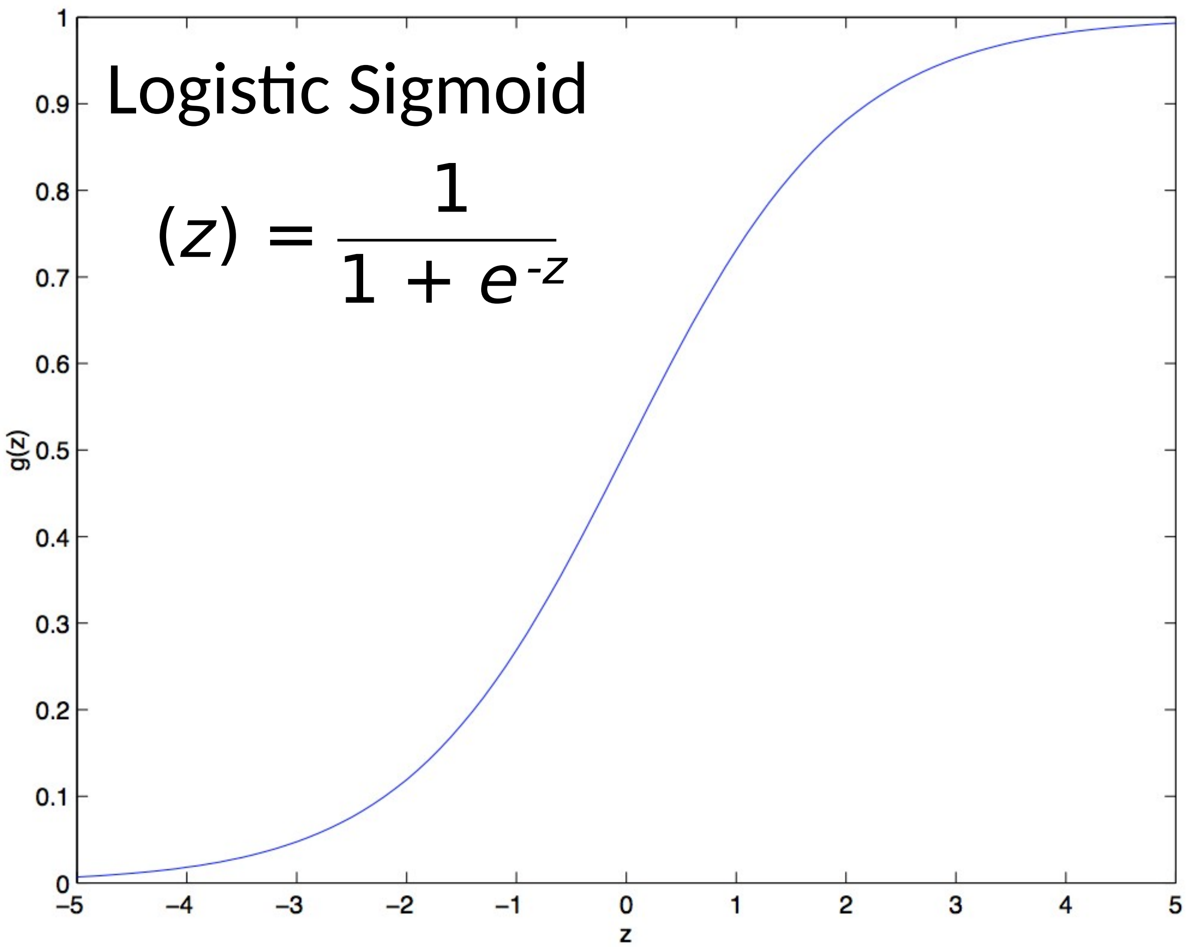
$$= \frac{p(\boldsymbol{x}|y=1)\,p(y=1)}{p(\boldsymbol{x}|y=0)\,p(y=0) + p(\boldsymbol{x}|y=1)\,p(y=1)}$$

Marginal definition

$$= \frac{1}{1 + \frac{p(\boldsymbol{x}|y=0)\,p(y=0)}{p(\boldsymbol{x}|y=1)\,p(y=1)}}$$

$$= \frac{1}{1 + \exp\left(\log\left(\frac{p(\boldsymbol{x}|y=0)\,p(y=0)}{p(\boldsymbol{x}|y=1)\,p(y=1)}\right)\right)}$$

Logistic Sigmoid Function



Predicting Classes with Gaussians

$$p(y=1|x)=\sigma(\log(\frac{p(\mathbf{x}|y=0)}{p(\mathbf{x}|y=1)})+\log(\frac{p(y=0)}{p(y=1)}))$$

Log-likelihood ratio

Constant w.r.t **x**

→ **With our Gaussian data :**

$$=\sigma(\log(p(\mathbf{x}|y=0))-\log(p(\mathbf{x}|y=1))+const)$$

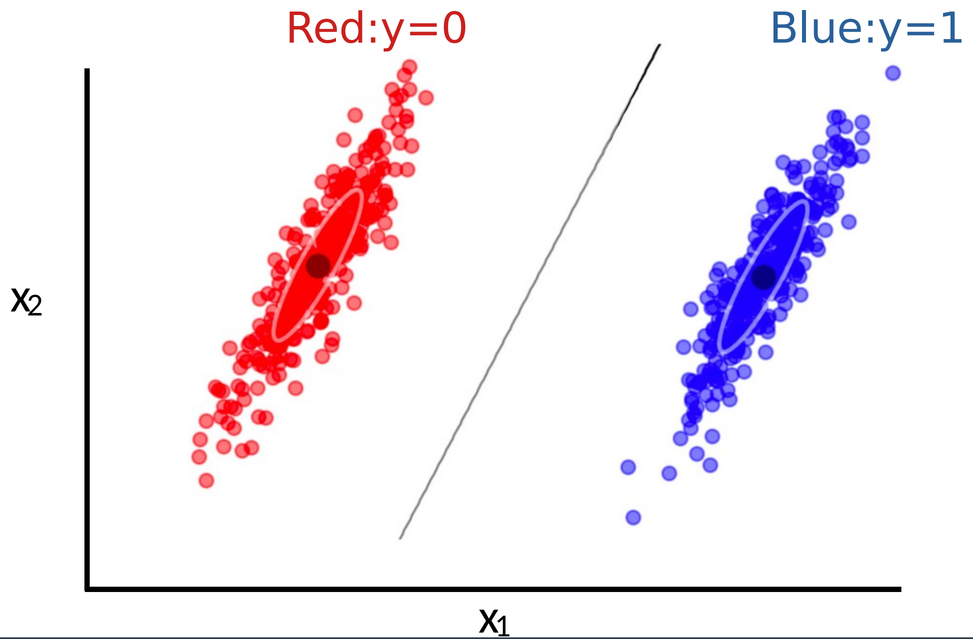
$$=\sigma(-1/2(\mathbf{x}-\mu_1)^T\Sigma^{-1}(\mathbf{x}-\mu_1)+1/2(\mathbf{x}-\mu_2)^T\Sigma^{-1}(\mathbf{x}-\mu_2)+const)$$

$$=\sigma((\mu_2-\mu_1)^T\Sigma^{-1}\mathbf{x}+1/2(\mu_2^T\Sigma^{-1}\mu_2-\mu_1^T\Sigma^{-1}\mu_1)+const)$$

$$=\sigma(\mathbf{w}^T\mathbf{x}+b)$$

Logistic regression

- **What did we learn ?**
  - For this data the log-likelihood ratio is linear
    - Line defines boundary to separate classes
    - Sigmoid turns distances from boundary into probability !



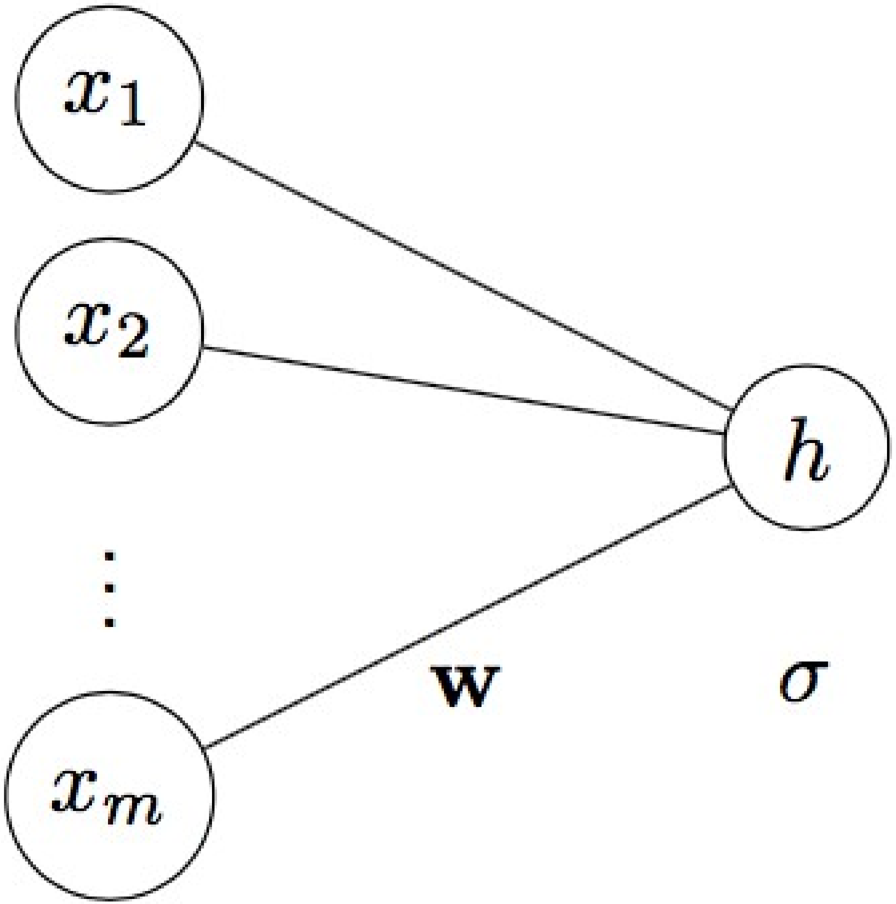
Logistic regression

- **What if we ignore Gaussian assumption on data?**
  - Model :

$$p\left(y=1|x\right)=\sigma\left(\boldsymbol{w}^T\boldsymbol{x}+b\right)\equiv h\left(\boldsymbol{x};\boldsymbol{w}\right)$$

- **Farther from boundary $w^Tx+b = 0$, more certain about class**
- **Sigmoid converts distance to class probability**

Logistic regression



$$p(y=1|x)=\sigma(\boldsymbol{w}^T\boldsymbol{x}+b)\equiv h(\boldsymbol{x};\boldsymbol{w})$$

$$=\frac{1}{1+\mathrm{e}^{-w^T\boldsymbol{x}-b}}$$

This unit is the main building block of Neural Networks!

Logistic regression

- **What if we ignore Gaussian assumption on data?**
  - Model : $p(y=1|x)=\sigma(\boldsymbol{w}^T\boldsymbol{x}+b)\equiv h(\boldsymbol{x};\boldsymbol{w})$

- **With** $p_i\equiv p(y_i=y|x_i)$

$$p(y_i=y|x_i)=Bernoulli(p_i)=(p_i)^{y_i}(1-p_i)^{1-y_i}=\begin{matrix} p_i & if\ y_i=1 \\ 1-p_i & if\ y_i=0 \end{matrix}$$

- **Log-likelihood :**

$$-\ln L=-\ln\prod(p_i)^{y_i}(1-p_i)^{1-y_i}$$

Binary cross entropy loss function

$$-\ln L=\sum -y_i\ln\sigma(\boldsymbol{w}^T\boldsymbol{x}+b)-(1-y_i)\ln(1-\sigma(\boldsymbol{w}^T\boldsymbol{x}+b))$$

Gradient descent

- **Likelihood function / Loss function L(θ) defined over a model parameters θ (i.e w & b)**
  - To minimize L(θ), gradient descent uses local linear information to iteratively move towards a (local) minimum.
  - First order approximation around θ_0 (Taylor expansion):

$$\hat{L}(\theta_0 + \epsilon) = L(\theta_0) + \epsilon \nabla_\theta L(\theta_0) + \frac{1}{2\gamma} \|\epsilon\|^2$$

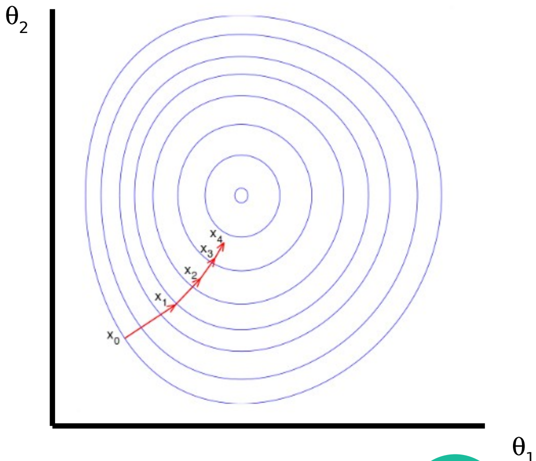
Gradient descent

A minimizer of the approximation $L(\theta_0+\varepsilon)$ is given for :

$$\nabla_\epsilon \hat{L}(\theta_0+\epsilon)=0=\nabla_\theta L(\theta_0)+\frac{1}{\gamma}\epsilon$$

- The best improvement is for the step: $\epsilon=-\gamma\nabla_\theta L(\theta_0)$
- Model parameters can be updated iteratively by :

$$\theta_{t+1}=\theta_t-\gamma\nabla_\theta L(\theta_t)$$



- θ_0 → initial parameters of the model
- γ → learning rate
  - Important for convergence of the minimization

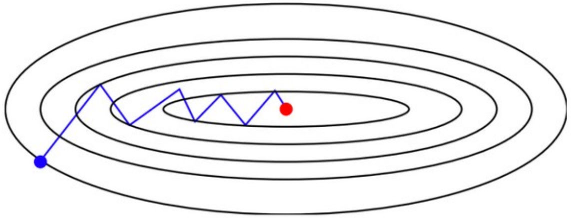
Stochastic gradient descent

- **Loss is composed of a sum over samples:**

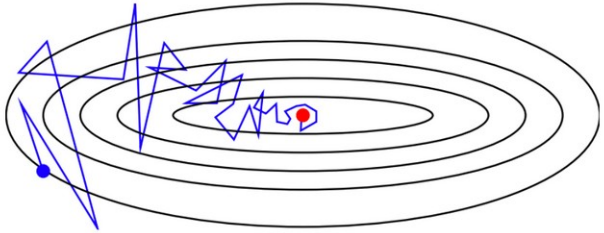
$$\nabla_\theta L(\theta) = \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta L(y_i, h(x_i, \theta))$$

→ Computing gradient grows linearly with N

- **Stochastic approach (SGB):**
  - Compute the gradient using a random sample (small size batch)
  - Gradient is unbiased → on average it moves in correct direction
  - Tends to be much faster the full gradient descent



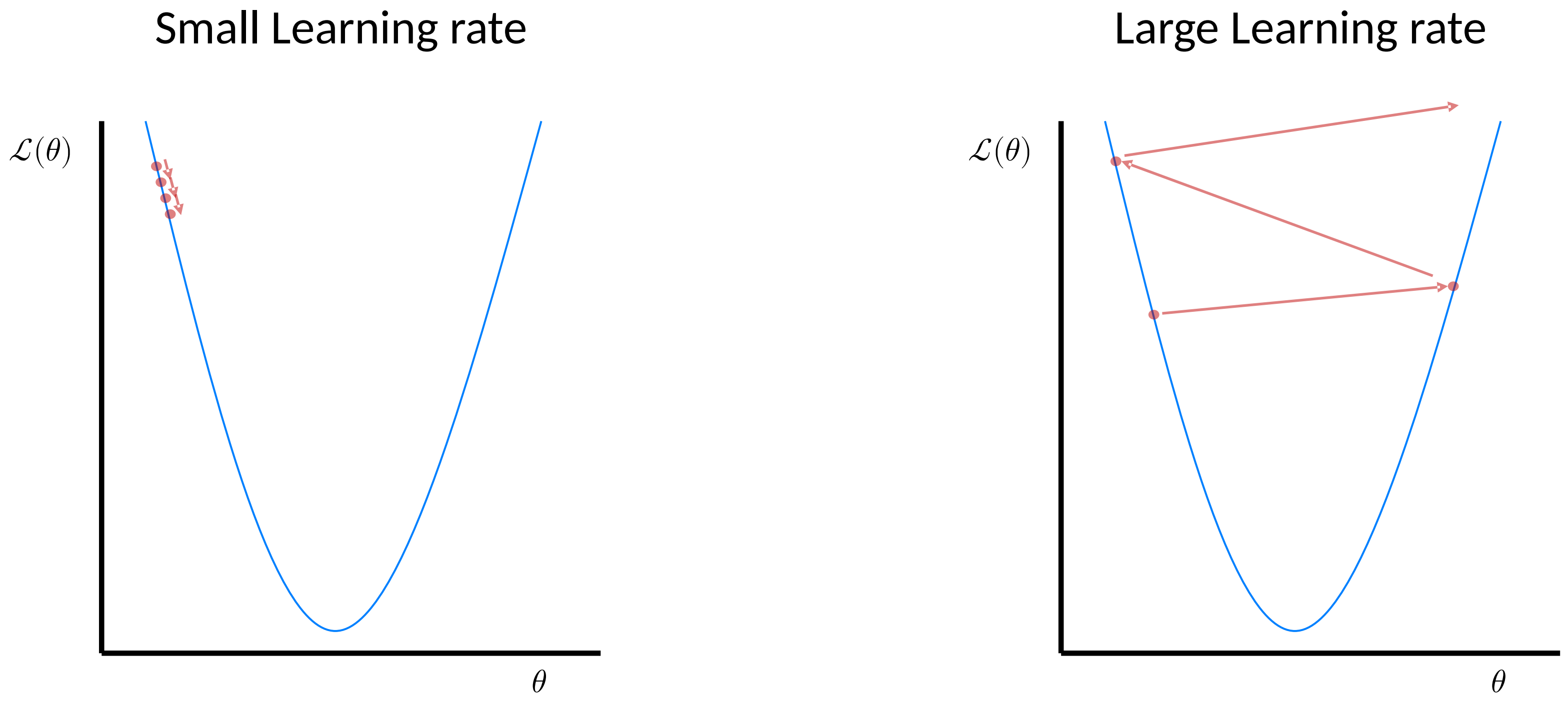
Batch gradient descent



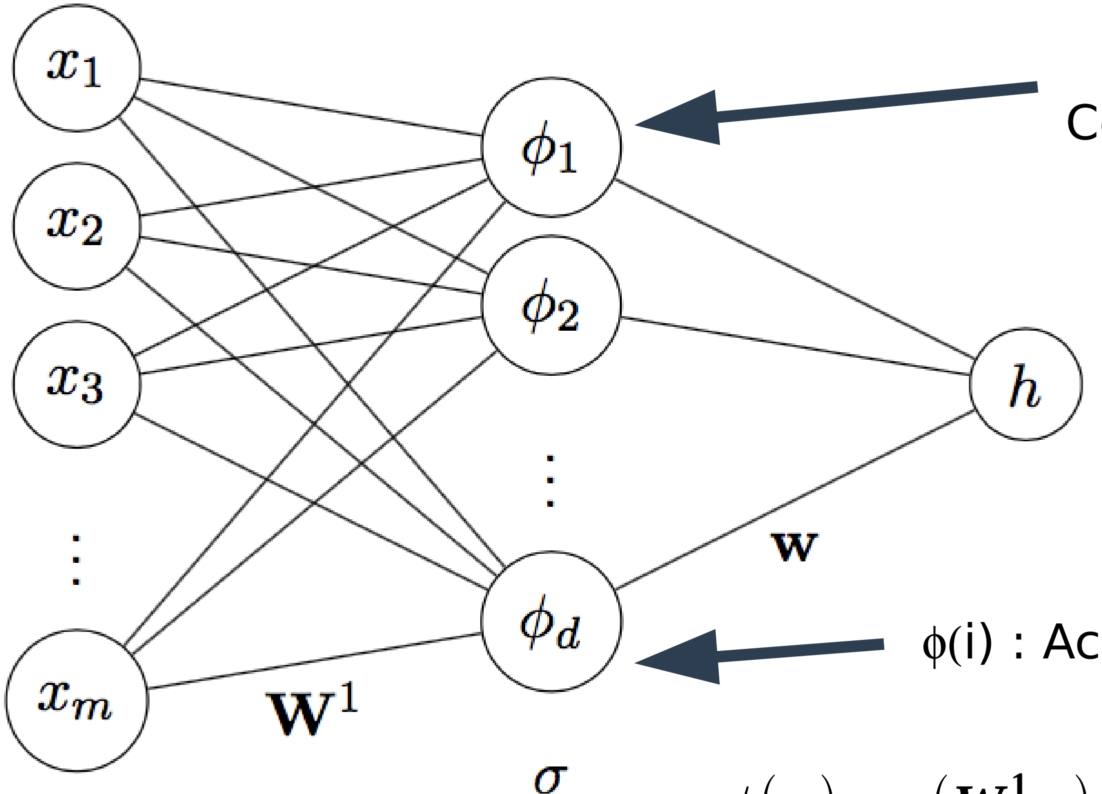
Stochastic gradient descent

Step sizes

- **Too small a learning rate, convergence very slow**
- **Too large a learning rate, algorithm diverges**



Feed Forward Neural Network



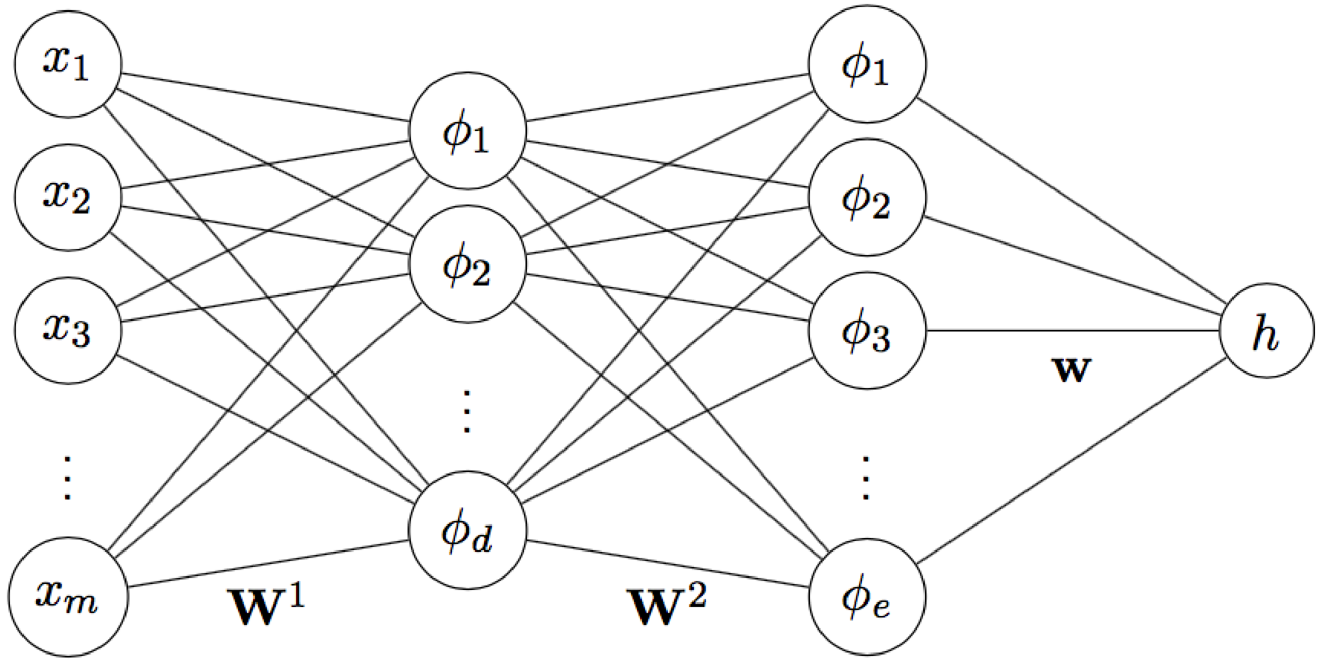
Hidden layer
Composed of *neurons*

$\phi(i)$: Activation function

$$\phi(\boldsymbol{x}) = \sigma(\boldsymbol{W}^{\mathbf{1}}\boldsymbol{x})$$

$$h(x) = w^T \phi(\boldsymbol{x})$$

Multi-layer Neural Network



- **Multilayer NN**
  - Each layer adapts basis functions based on previous layer

Neural Network Optimization Problem

- **Neural Network Model:** $h(x)=w^T\sigma(\boldsymbol{W^1 x})$
- **Classification: Cross-entropy loss function**

$$L(\boldsymbol{w},\boldsymbol{W^1})=\sum_i y_i \ln(p_i)+(1-y_i)\ln(1-p_i)$$

- **Regression: Square error loss function**

$$L(\boldsymbol{w},\boldsymbol{W^1})=\frac{1}{2}\sum_i (y_i-h(x_i))^2$$

- **Minimize loss with respect to weights :** w, W^1

Backpropagation

- **Loss function composed of layers of nonlinearity :**

$$L\left(\phi^N\left(\dots\phi^1\left(\boldsymbol{x}\right)\right)\right)$$

1. Forward step:

- Compute and save intermediate computations

$$\phi^N\left(\dots\phi^1\left(\boldsymbol{x}\right)\right)$$

2. Backward step:

$$\frac{\partial L}{\partial \phi^a} = \sum_j \frac{\partial \phi_j^{a+1}}{\partial \phi_j^a} \frac{\partial L}{\partial \phi_j^{a+1}}$$

3.Compute parameter gradients:

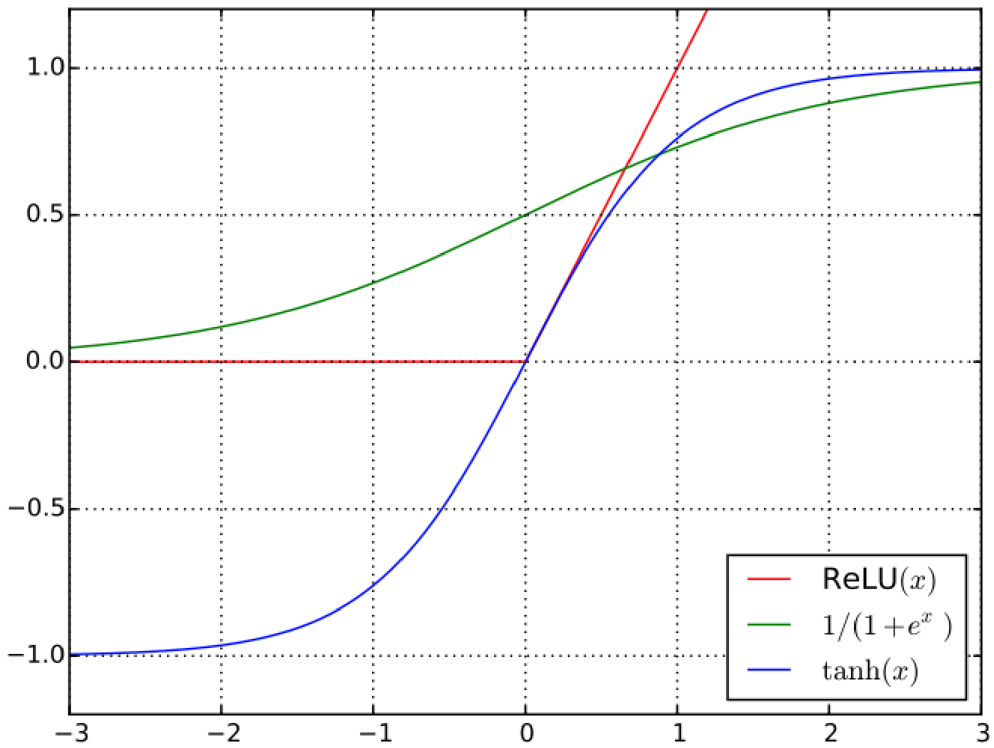
$$\frac{\partial L}{\partial \boldsymbol{w}^{\boldsymbol{a}}} = \sum_j \frac{\partial \phi_j^a}{\partial \boldsymbol{w}^{\boldsymbol{a}}} \frac{\partial L}{\partial \phi_j^a}$$

- **Why sigmoid ?** $\frac{\partial \sigma(x)}{\partial x} = \sigma(x)\left(1-\sigma(x)\right)$

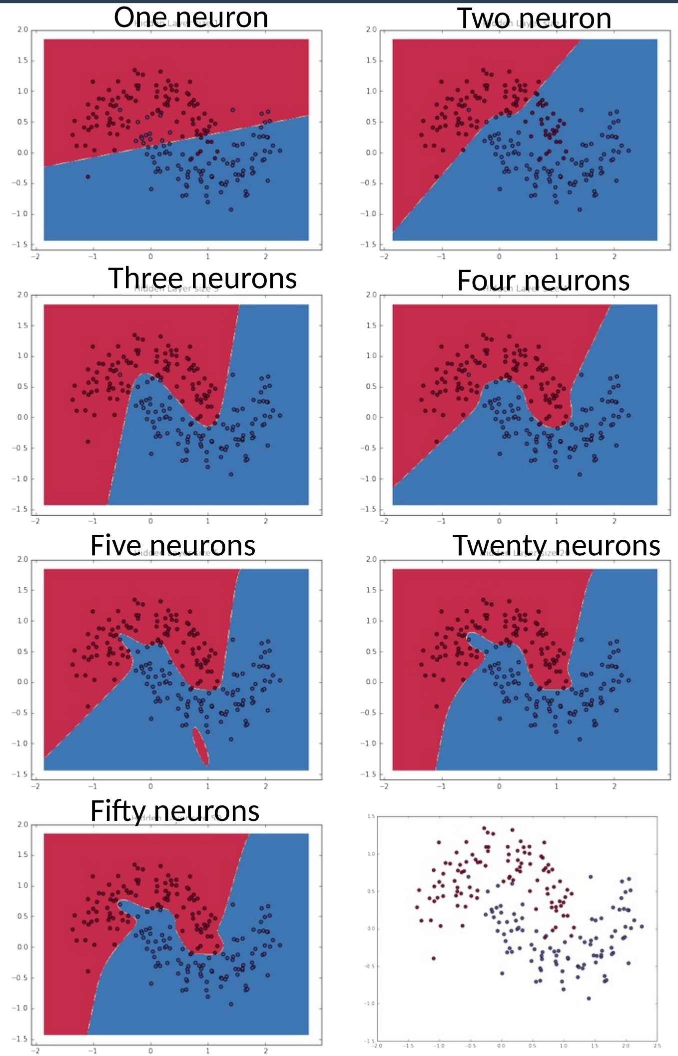
Easy to compute !

Activation functions

- **Started with sigmoid, but any function can be used**
- **Requirement :**
  - Easy/simple derivative
  - That can be expressed as function of itself
- **Examples:**
  - tanh,
  - sigmoid,
  - ReLU = max {0,x}

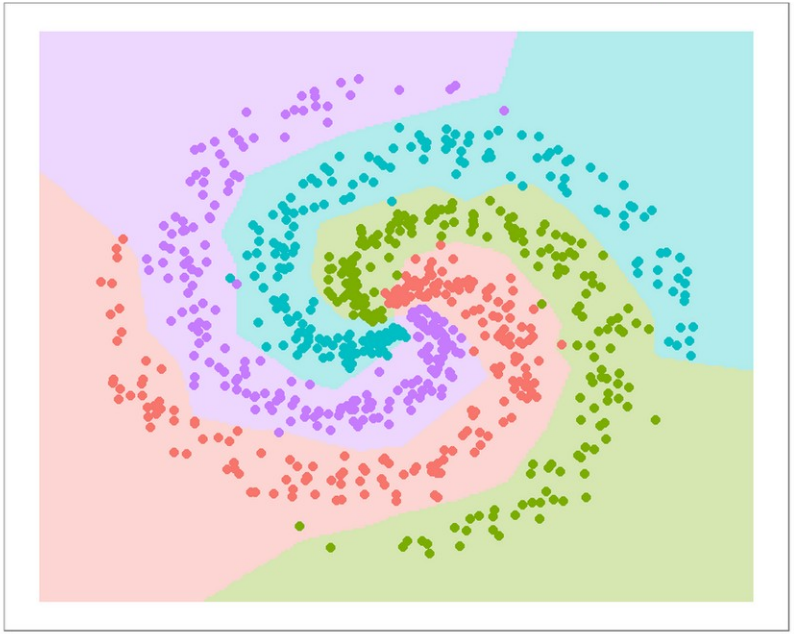


Neural Network Decision Boundaries

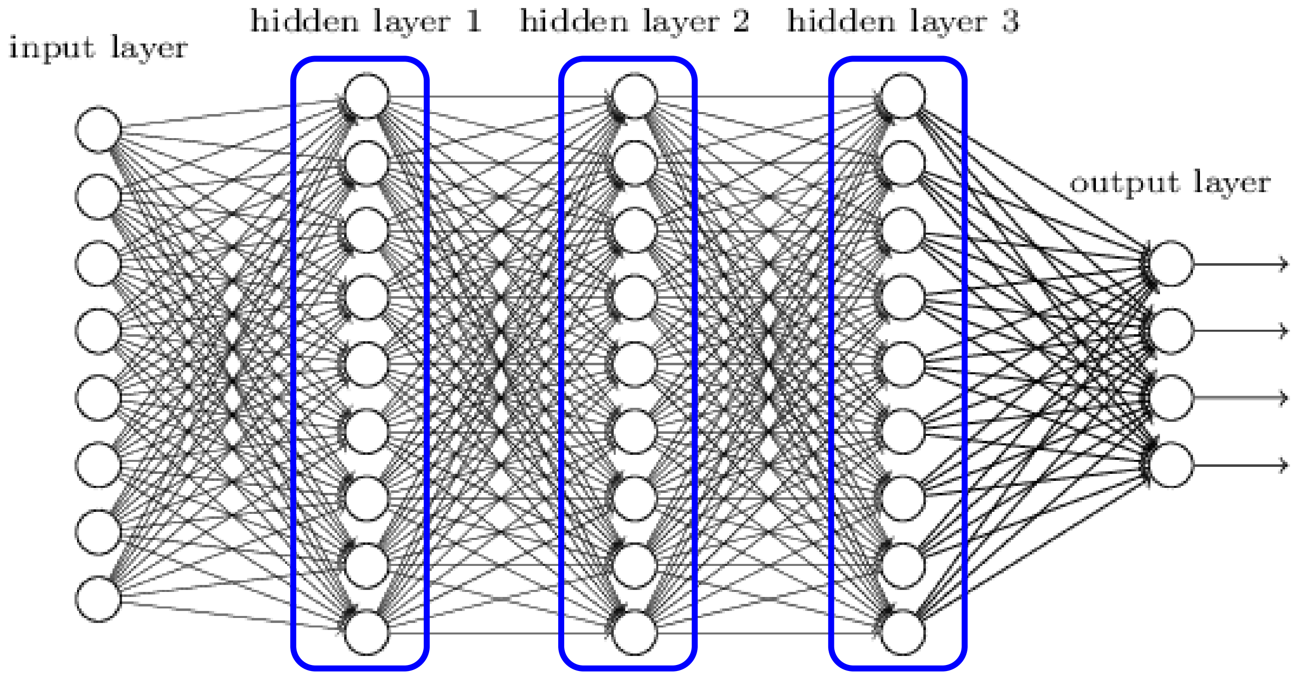


2-class classification
1-hidden layer NN
L2 norm regularization

4-class classification
2-hidden layer NN
ReLU activations
L2 norm regularization



Deep Neural Networks



- As data complexity grows, need exponentially large number of neurons in a single-hidden-layer network to capture all structure in data
- Deep neural networks factorize the learning of structure in data across many layers

Demystify neural networks

- **Full implementation of training of 2-layer NN :**

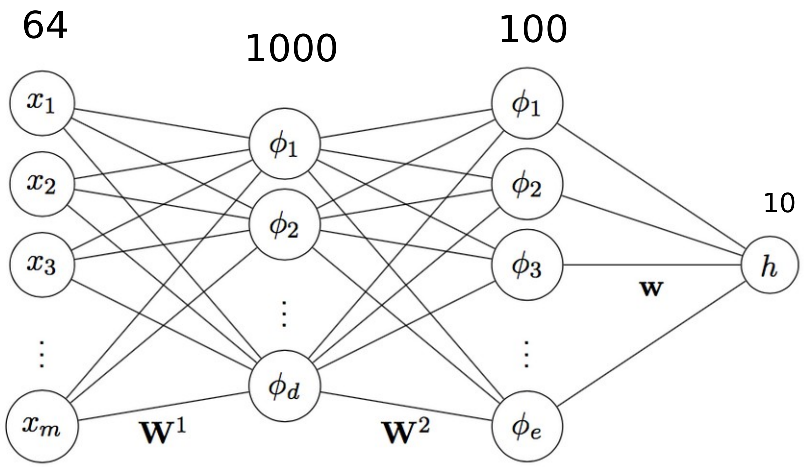
```python
import numpy as np
from numpy.random import randn

N, D_in, H, D_out = 64, 1000, 100, 10
x, y = randn(N, D_in), randn(N, D_out)
w1, w2 = randn(D_in, H), randn(H, D_out)

for t in range(2000):
  h = 1 / (1 + np.exp(-x.dot(w1)))
  y_pred = h.dot(w2)
  loss = np.square(y_pred - y).sum()
  print(t, loss)

  grad_y_pred = 2.0 * (y_pred - y)
  grad_w2 = h.T.dot(grad_y_pred)
  grad_h = grad_y_pred.dot(w2.T)
  grad_w1 = x.T.dot(grad_h * h * (1 - h))

  w1 -= 1e-4 * grad_w1
  w2 -= 1e-4 * grad_w2
```



Optimization part:
gradient descent
via back propagation

Cooking recipe in ML

- **Get data (loads of them)**
- **Get good hardware**
- **Define the neural network architecture as a composition of differentiable functions**
- **Optimize with (variants of) stochastic gradient descent**
- **But pitfalls to be aware of:**
  - Data quality : Garbage In → Garbage Out / Missing data ?
  - Underfitting / Overfitting
  - Simplicity don’t imply better generalization
  - Appropriate evaluation metric
  - Mistaking correlation for causation & confounding variables

Any questions ?