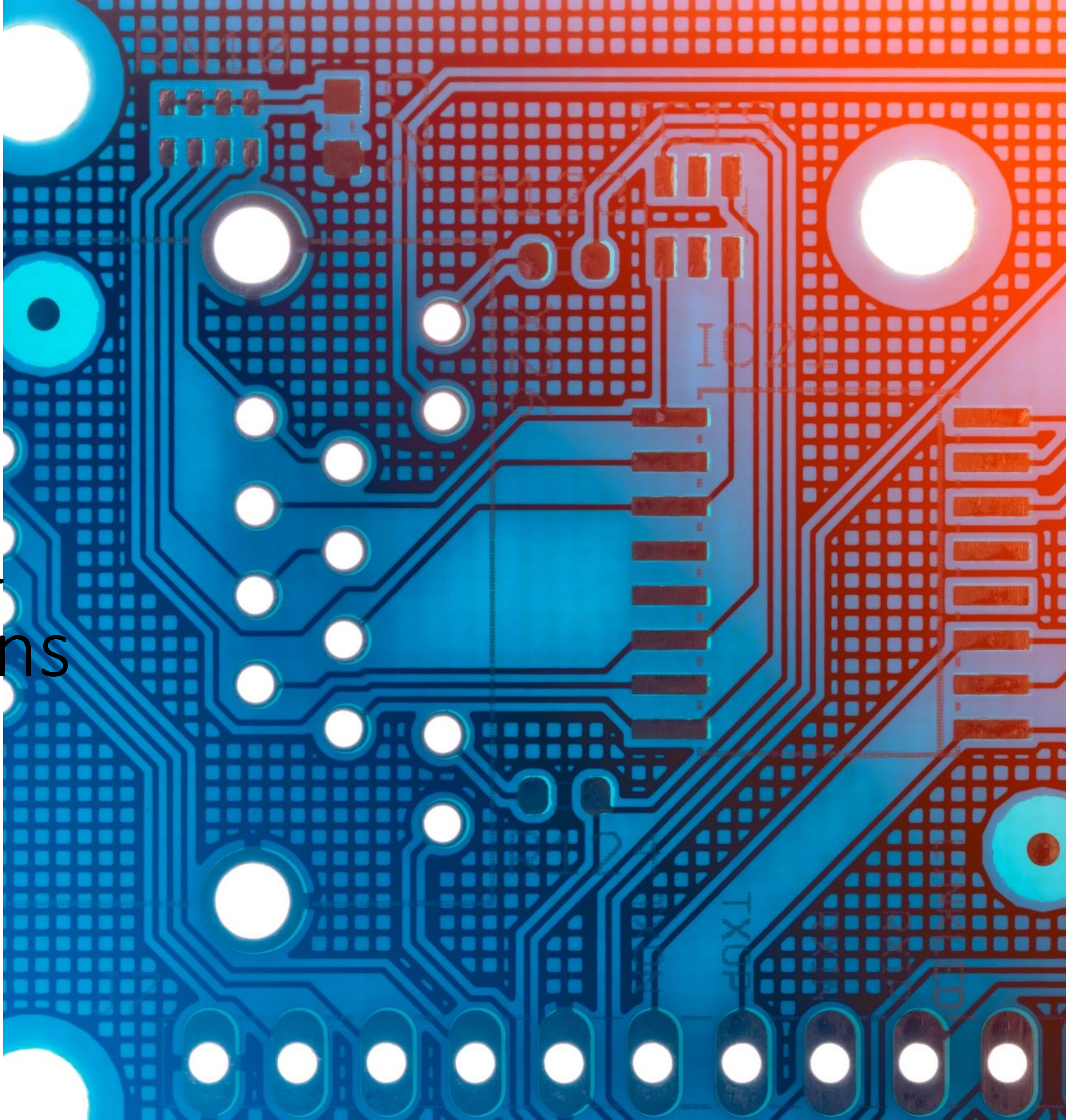
Susana Irene Díaz Rodríguez

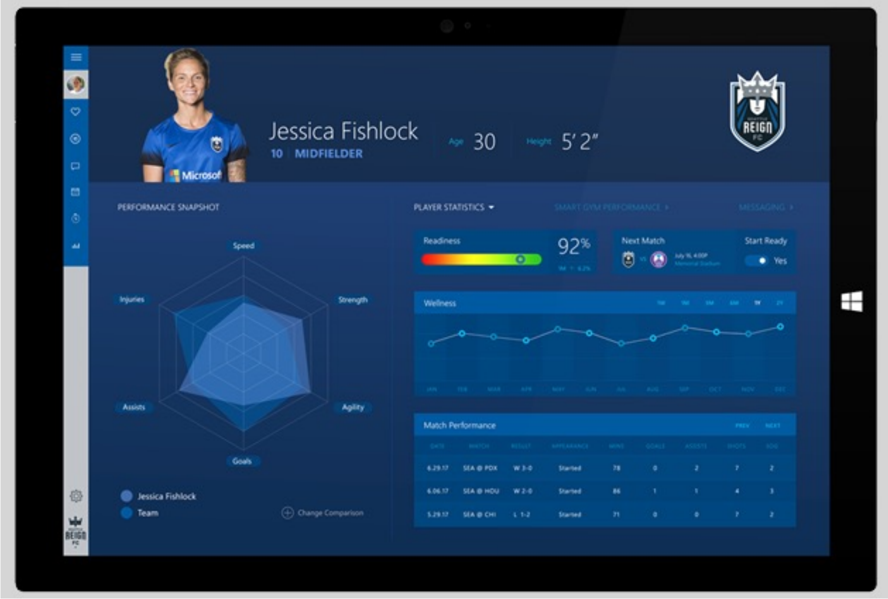
Full professor of Artificial Intelligence

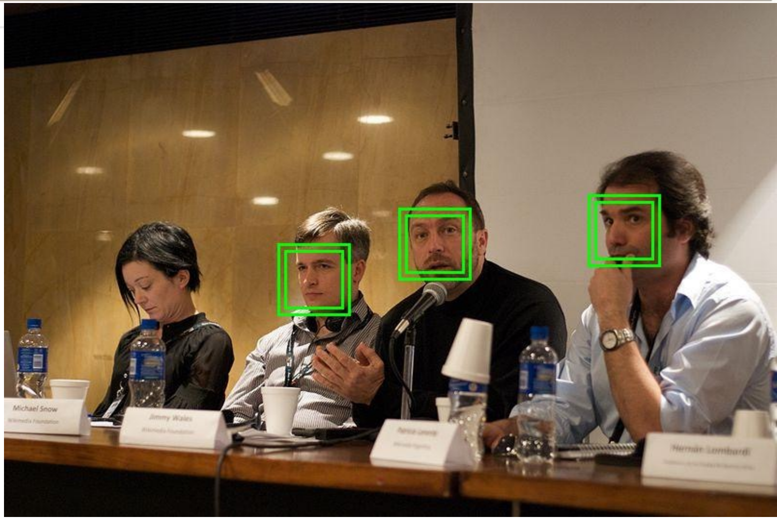
Universidad de Oviedo

Machine learning & Applications



The term *machine learning* refers to the automated detection of meaningful patterns in data

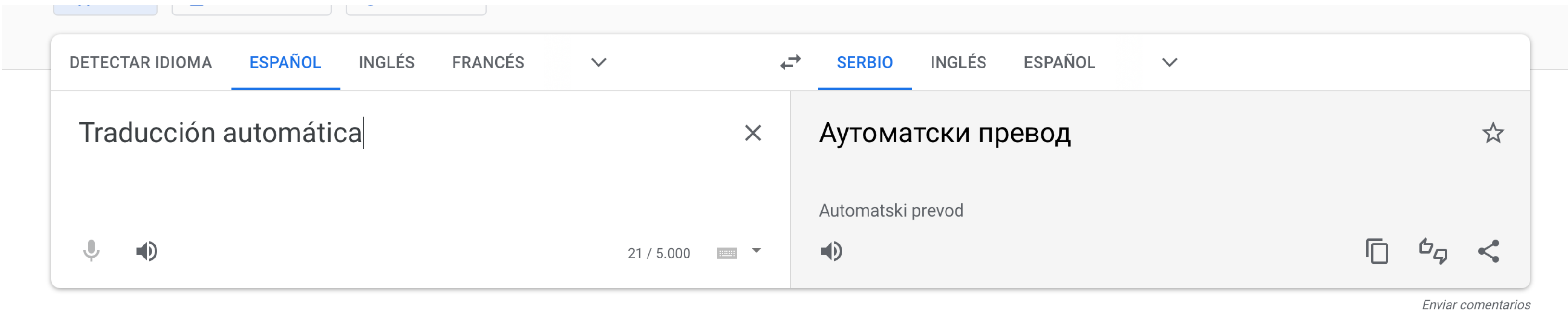




What learning is

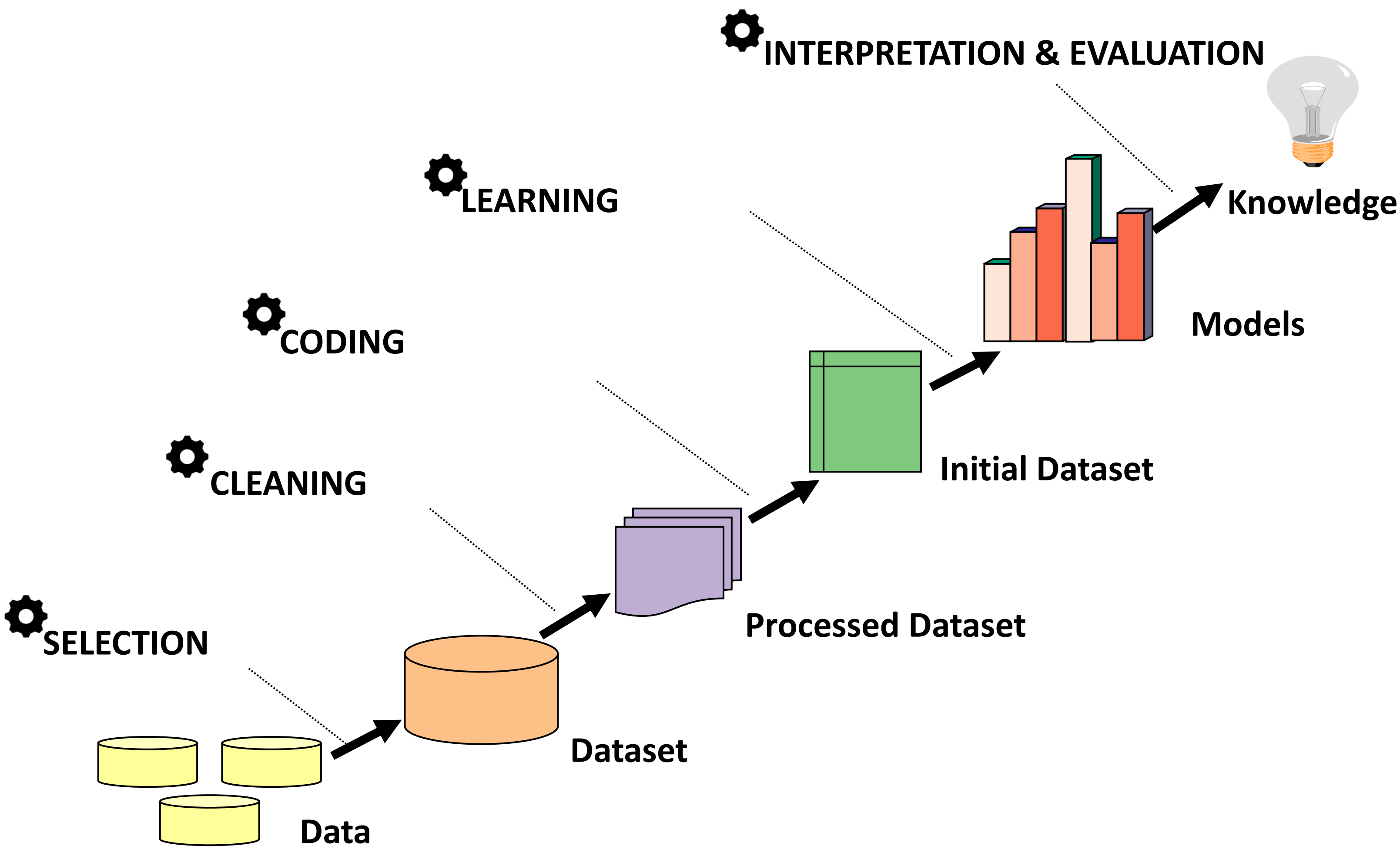
When do we need Machine Learning?

• *Tasks Performed by Humans: Learn from experience*



• *Tasks beyond human capacities*

Data Mining

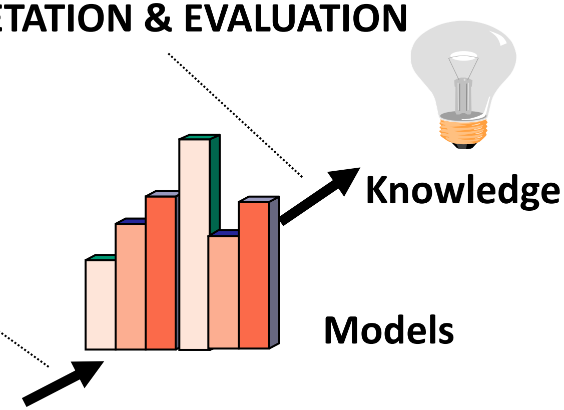


INTERPRETATION & EVALUATION
LEARNING
CODING
CLEANING
SELECTION
Knowledge
Models
Initial Dataset
Processed Dataset
Dataset
Data

Data Mining

INTERPRETATION & EVALUATION

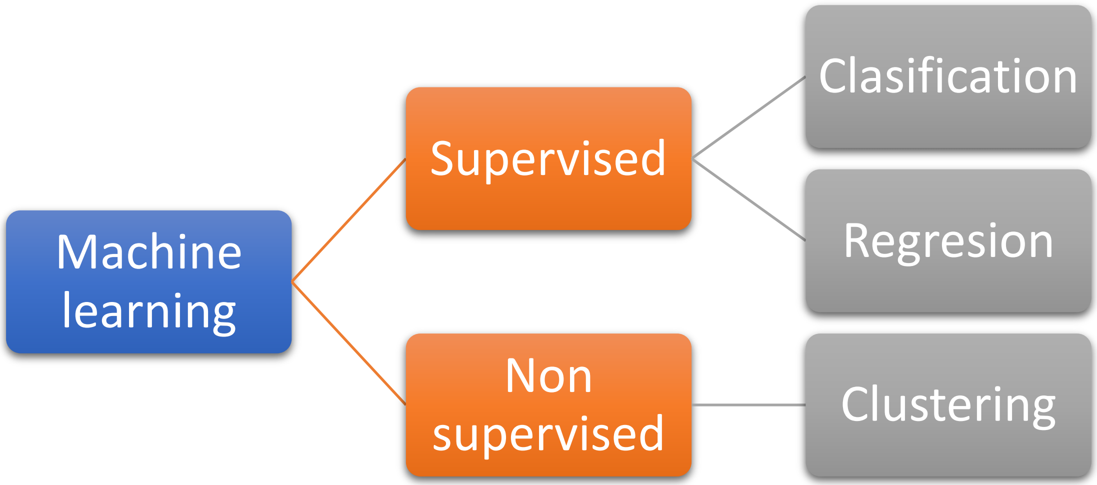
LEARNING



Outlook	Temp.	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Taxonomy

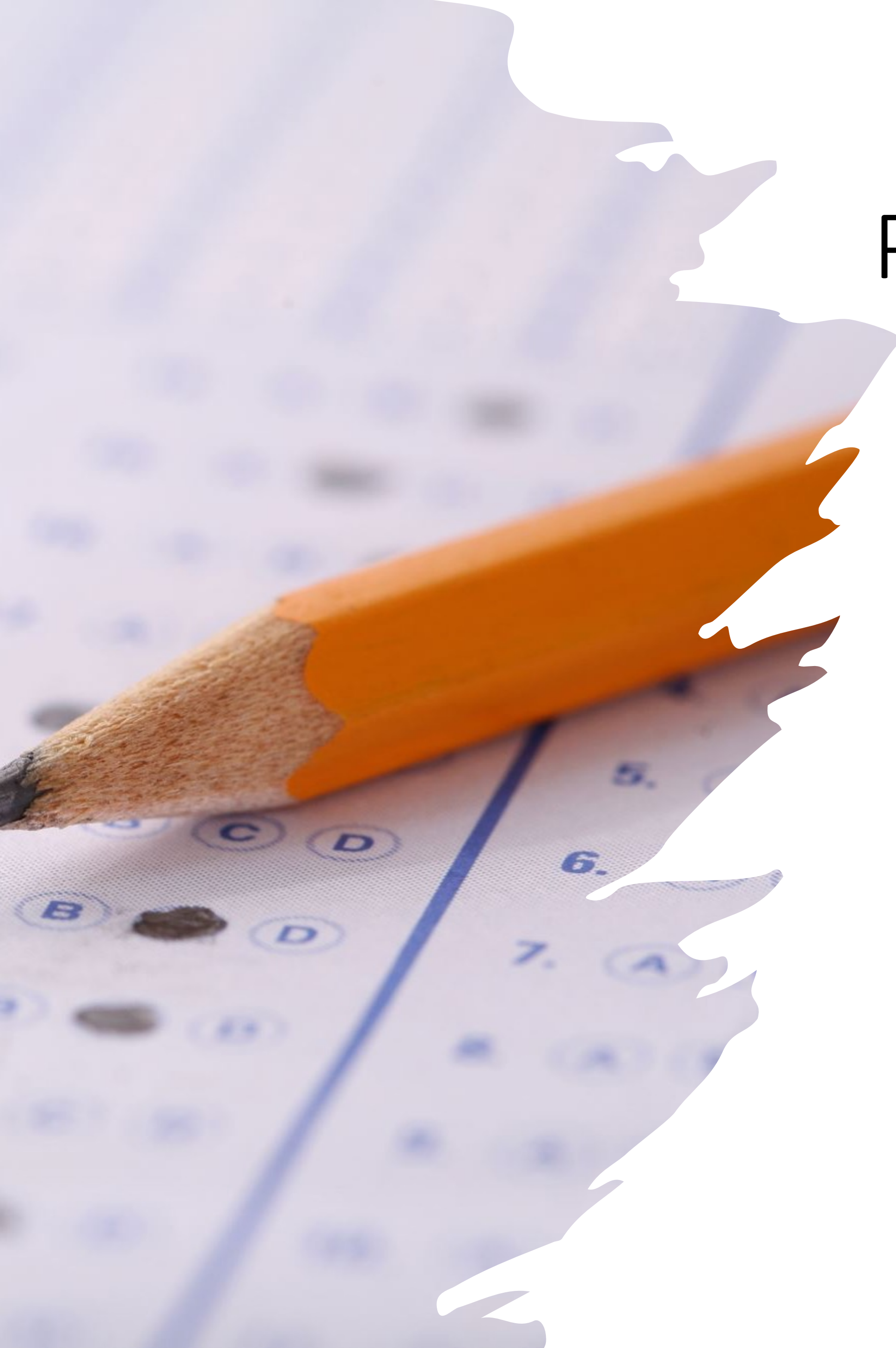
Learn a function mapping inputs to outputs using labeled training data (you get instances/examples with both inputs and ground truth output)



Learn something about just data without any labels (harder!), for example clustering instances that are “similar”

Supervised ML

- Inpu tdata
- Input to the function(features/attributes of data)
- The function or model you choose
- The optimization algorithm you use to explore space of functions



Problem statement

- Set of possible instances $\mathcal{X}$
- Set of possible labels $\mathcal{Y}$
- Unknown target function $f : \mathcal{X} \to \mathcal{Y}$
- Set of function hypotheses $H = \{h \mid h : \mathcal{X} \to \mathcal{Y}\}$

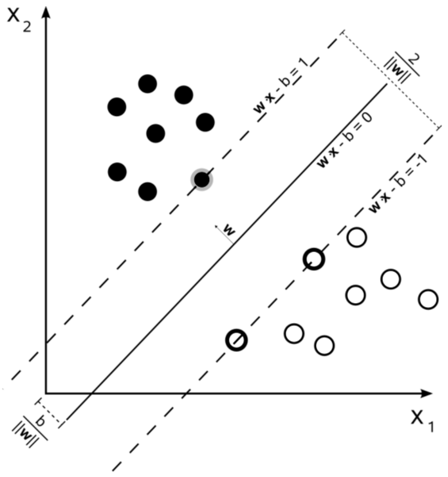
Input: Training examples of unknown target function f

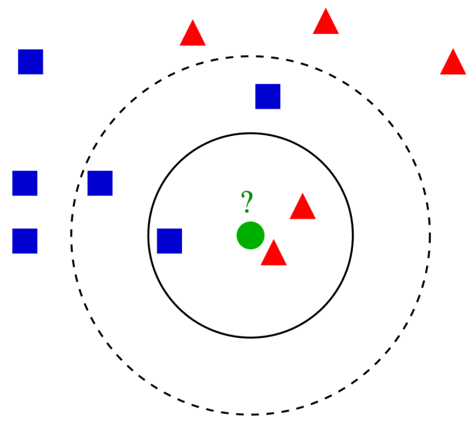
$$\{\langle \boldsymbol{x}_i, y_i \rangle\}_{i=1}^{n} = \{\langle \boldsymbol{x}_1, y_1 \rangle, \ldots, \langle \boldsymbol{x}_n, y_n \rangle\}$$

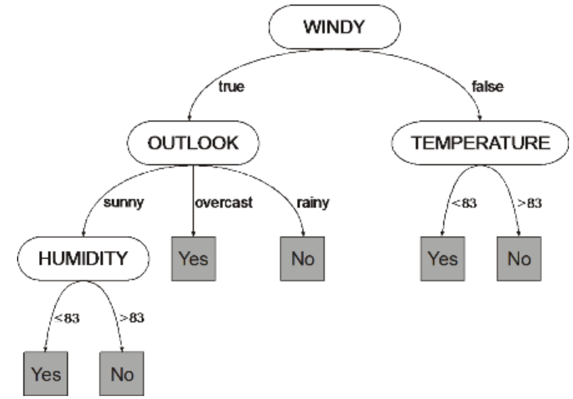
Output: Hypothesis $h \in H$ that best approximates f

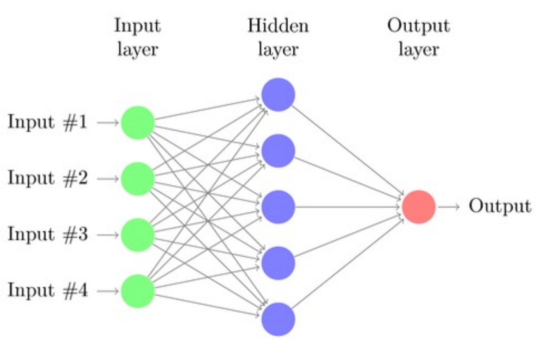
Which one is the best solution?

- $h^* = argmax_{\{h \in H\}}[P(h|Data)]$
- **Select the simplest solution** (Ockham principle)



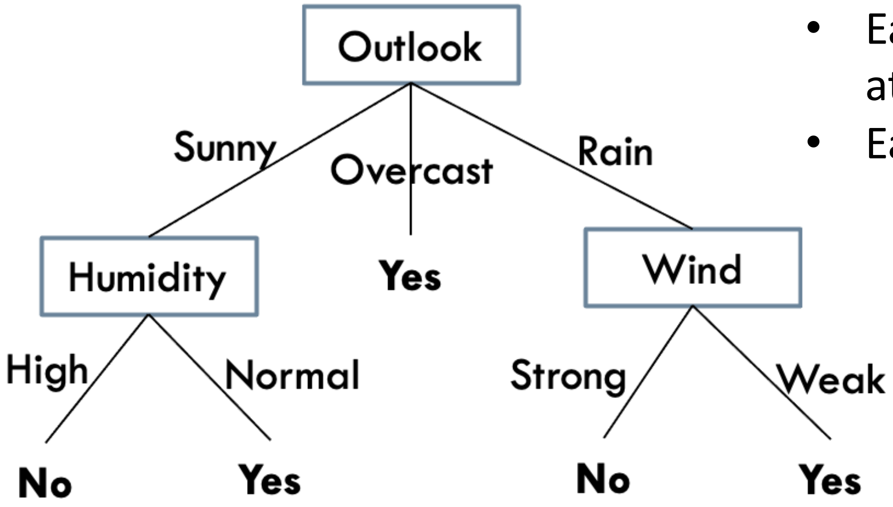






Different paradigms

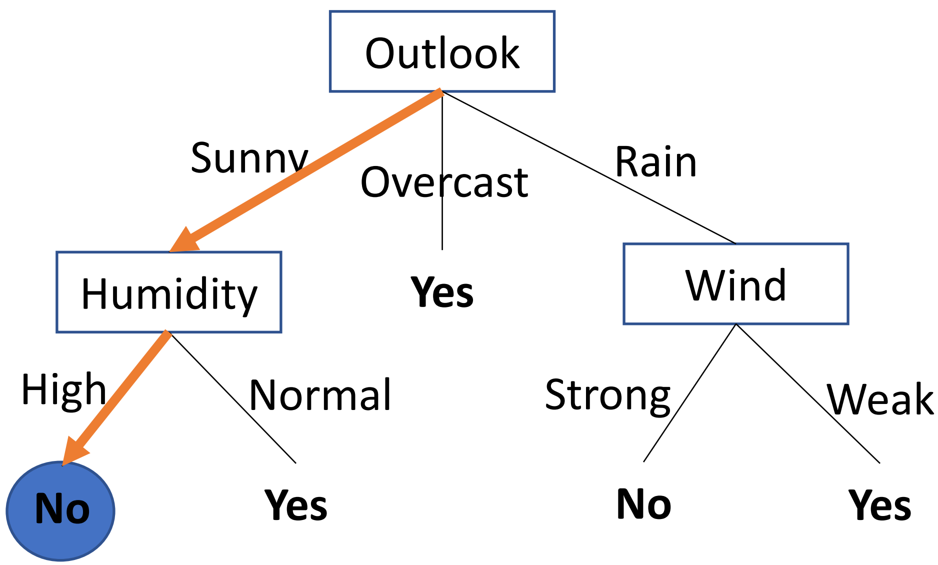
Decision Trees



- Each internal node is a test on one attribute
- Each leaf node: Prediction

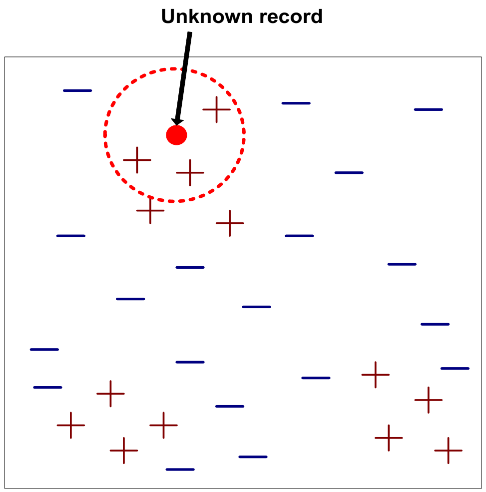
Do we play tennis ?

- The prediction is :
  - {Outlook:Sunny, Temperature: Hot, Humidity: High, Wind: Strong}



K-NN

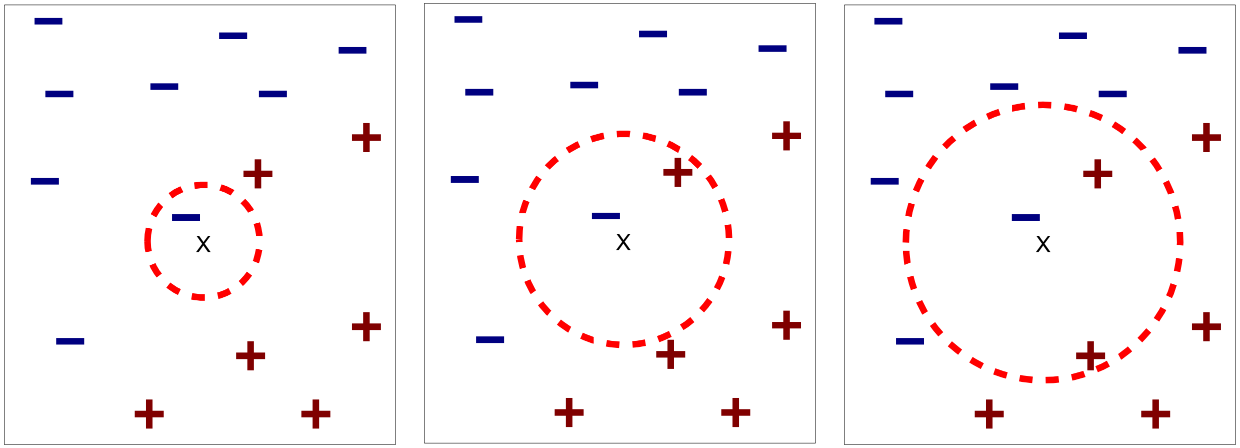
- Maybe the simplest method
- Instance based learning



Require 3 inputs

1. Training set
2. A distance
3. *k*, the number of neighbors

K-NN



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

Support Vector Machines

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2, \text{ subject to } y_i(\boldsymbol{w}\boldsymbol{x_i} + b) \geq 1$$

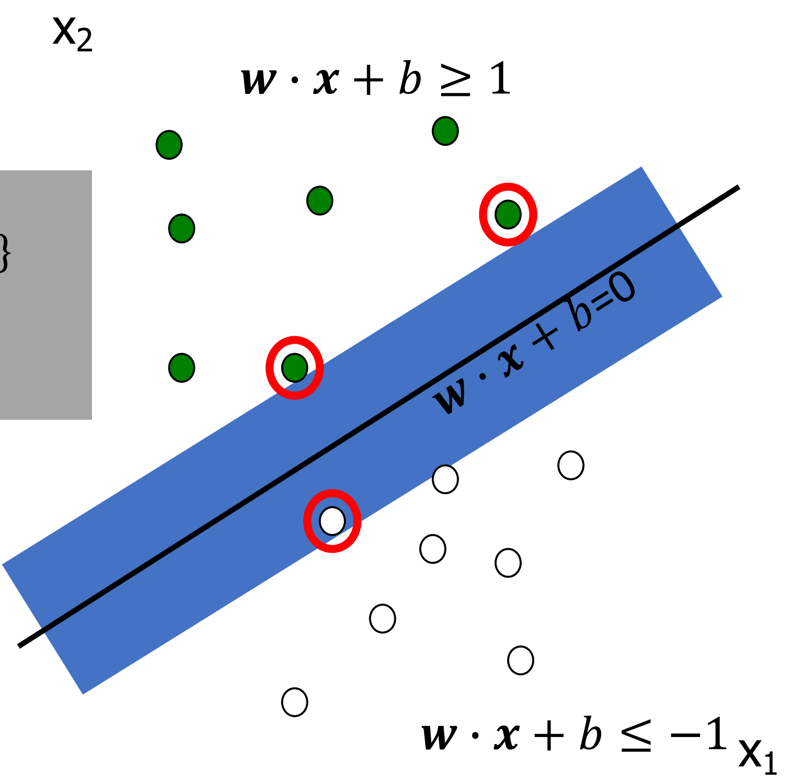
$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{n} \boldsymbol{\alpha_i}\{y_i(\boldsymbol{w}\boldsymbol{x_i} + b) - 1\}$$

$\alpha_i \geq 0$ $\forall i$=1,...., n

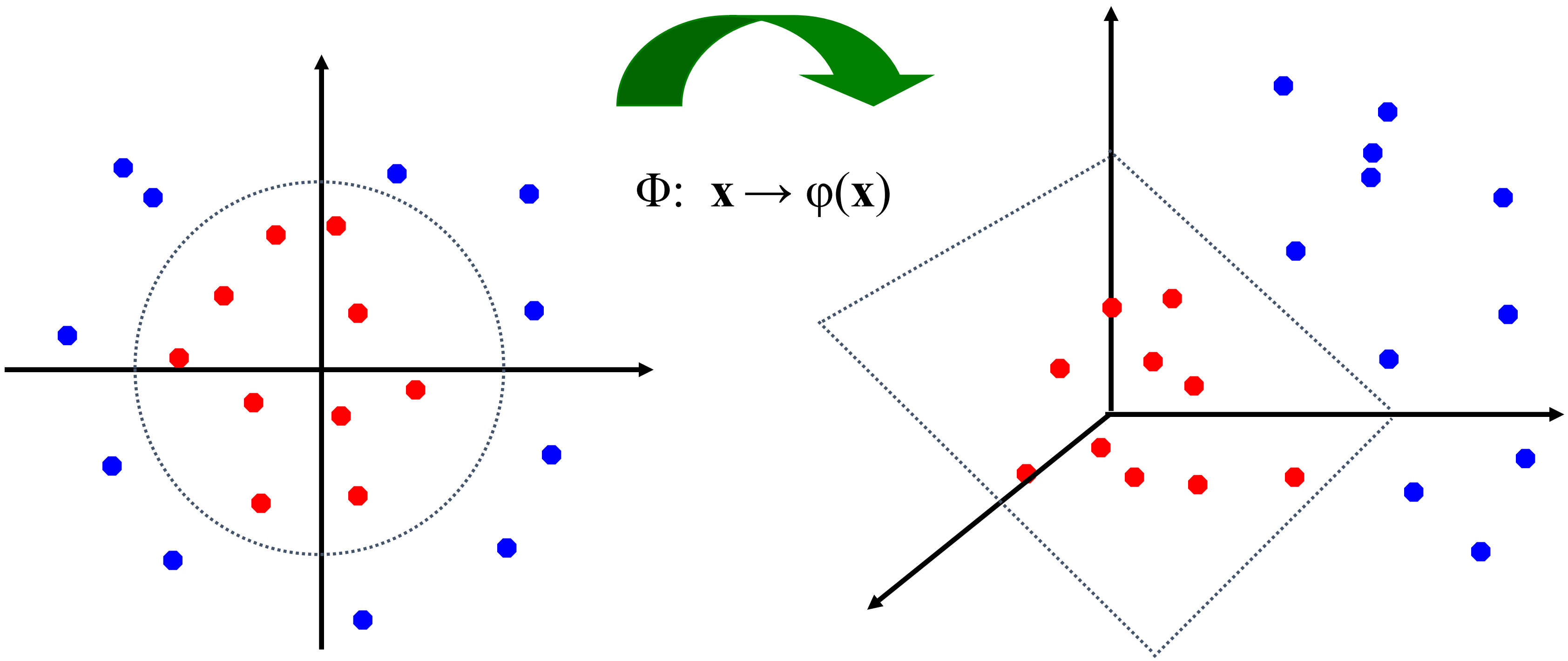
$$max_\alpha L_d(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y_i y_j \boldsymbol{x_i}\boldsymbol{x_j}$$

Subject to

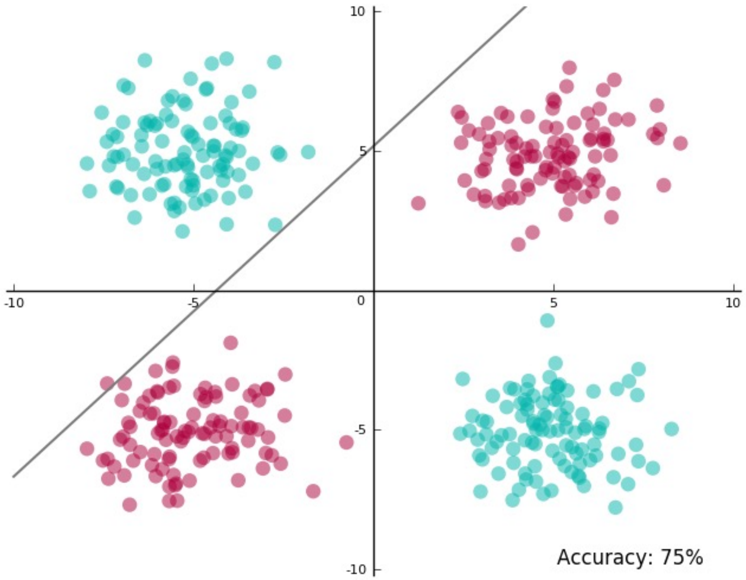
$\alpha_i \geq 0\ \forall i, \qquad 0 = \sum_{i=1}^{n} \alpha_i y_i$

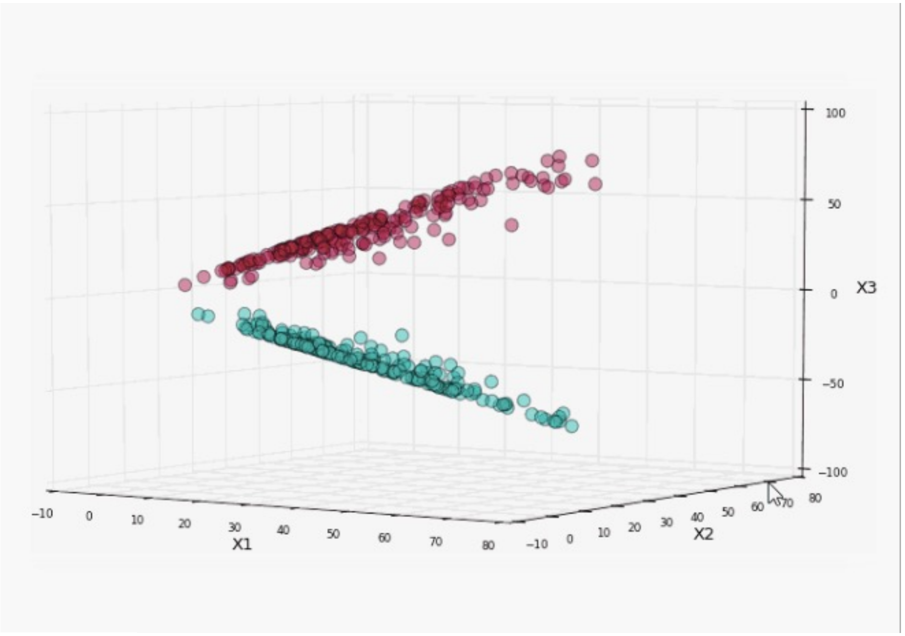


Support Vector Machines



Support Vector Machines



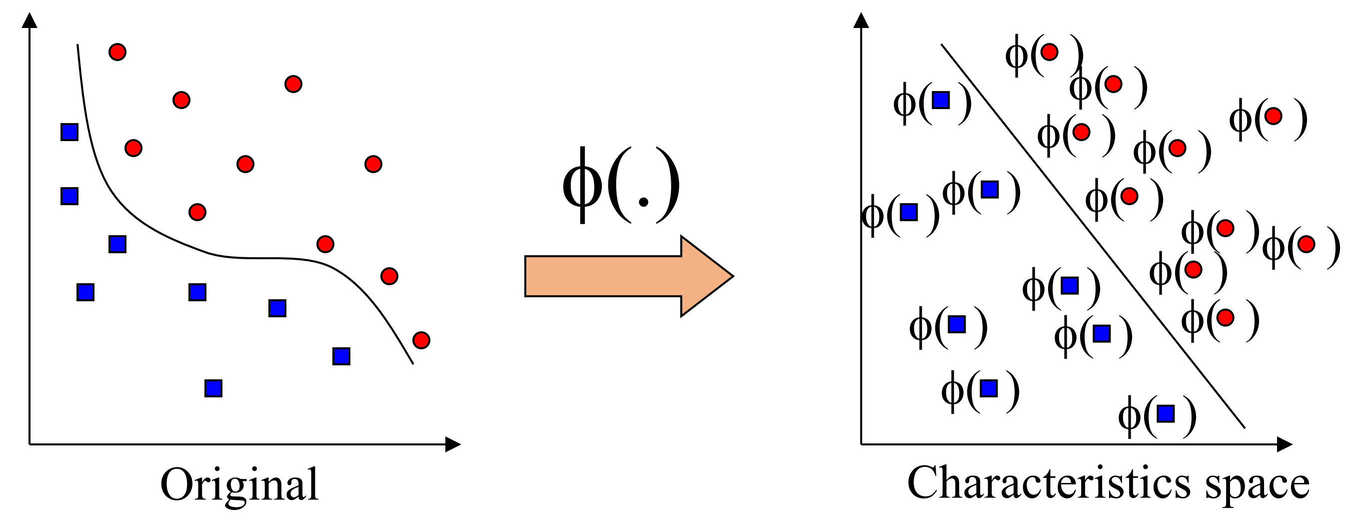


$$X_1 = x_1^2$$

$$X_2 = x_2^2$$

$$X_3 = \sqrt{2}x_1x_2$$

Support Vector Machines



Support Vector Machines

$$\min_{\boldsymbol{w},b} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i,$$

subject to $y_i(\boldsymbol{w}\boldsymbol{x_i} + b) \geq 1 - \xi_i$

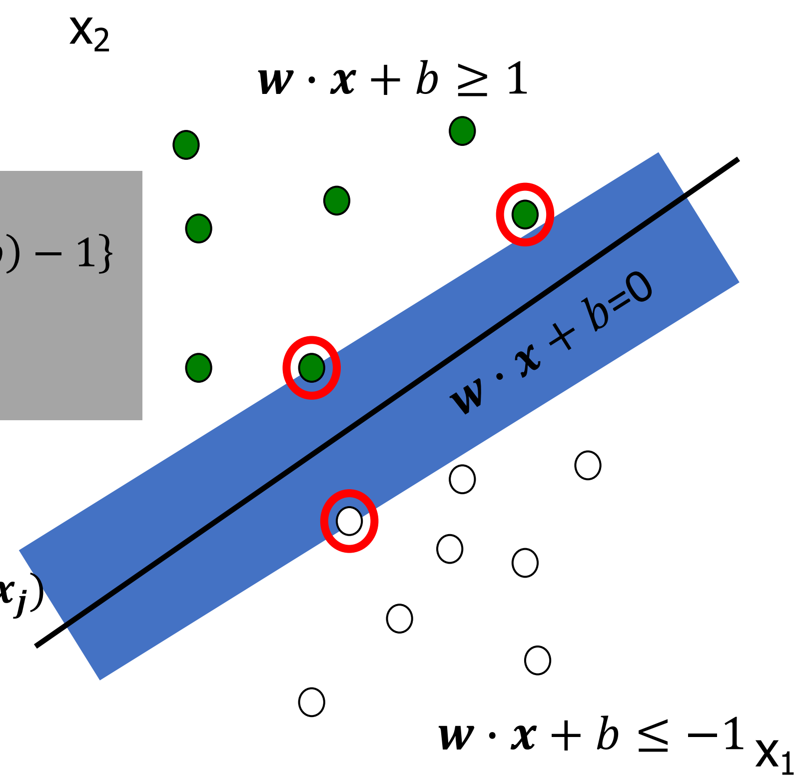
$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\boldsymbol{\alpha_i}\{y_i(\boldsymbol{w}\boldsymbol{x_i} + b) - 1\}$$

$0 \leq \alpha_i \leq C \quad \forall i=1,\ldots, n$

$$max_{\alpha} L_d(\boldsymbol{\alpha}) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{K}(\boldsymbol{x_i}\boldsymbol{x_j})$$

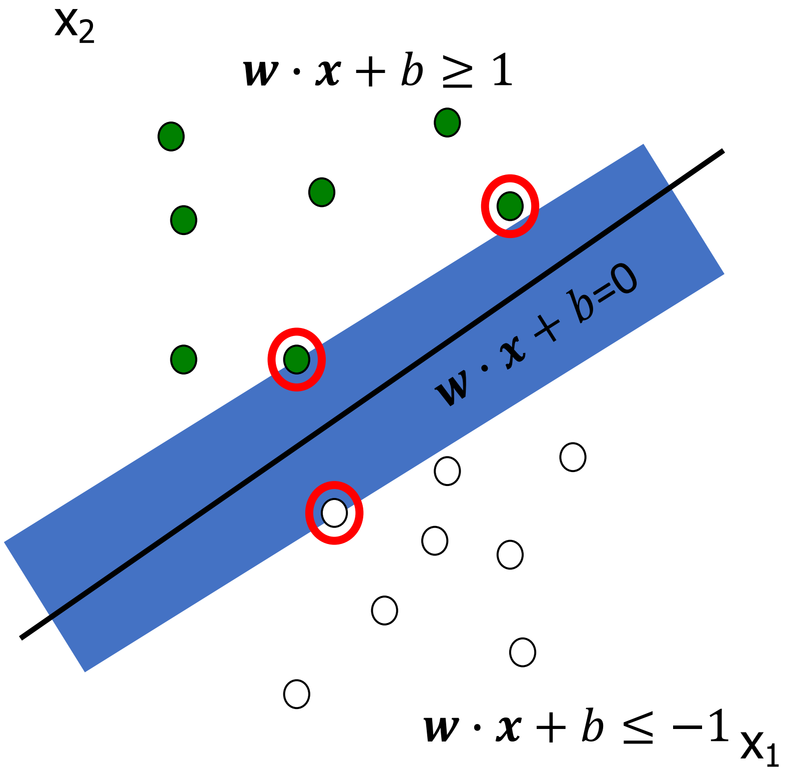
Subject to

$0 \leq \alpha_i \leq C \quad \sum_{i=1}^{n}\alpha_i y_i = 0$



Support Vector Machines

$$g(\boldsymbol{x}) = \boldsymbol{w}\boldsymbol{x} + b = \sum_{i \in VS} \alpha_i y_i \boldsymbol{x_i}\boldsymbol{x} + b$$



Support Vector Machines

- Polynomial (degree d)

$$K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}\boldsymbol{y} + 1)^d$$

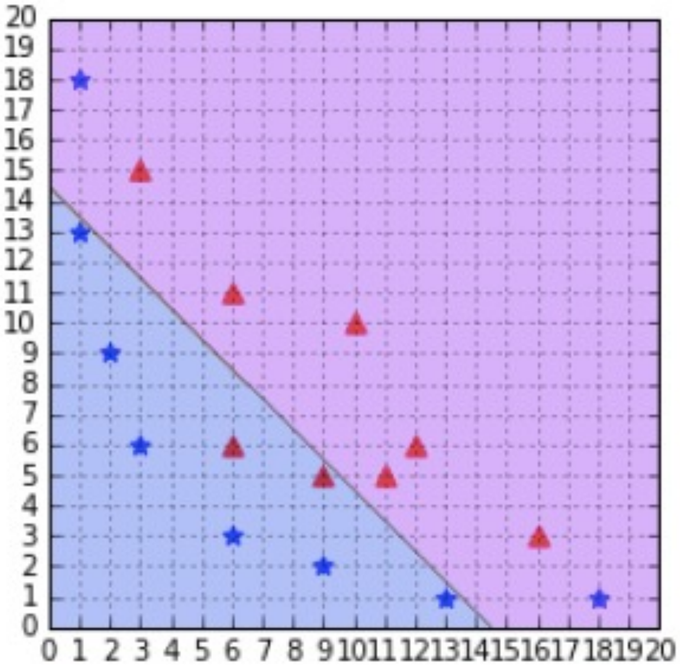
- Radial (width σ)

$$K(\boldsymbol{x}, \boldsymbol{y}) = e^{-\|\boldsymbol{x}-\boldsymbol{y}\|^2/(2\sigma^2)}$$

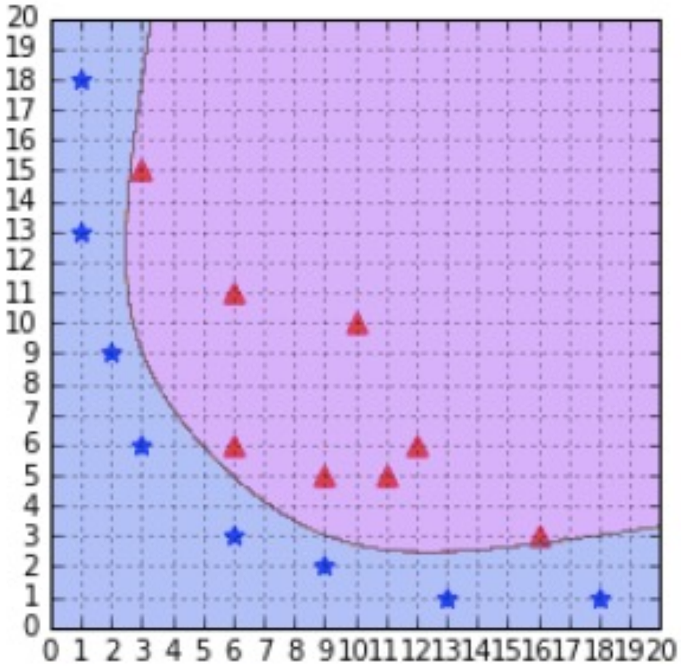
- Sigmoidal (parameters κ and θ)

$$K(\boldsymbol{x}, \boldsymbol{y}) = \tanh(\kappa\boldsymbol{x}\boldsymbol{y} + \theta)$$

Polynomial kernel

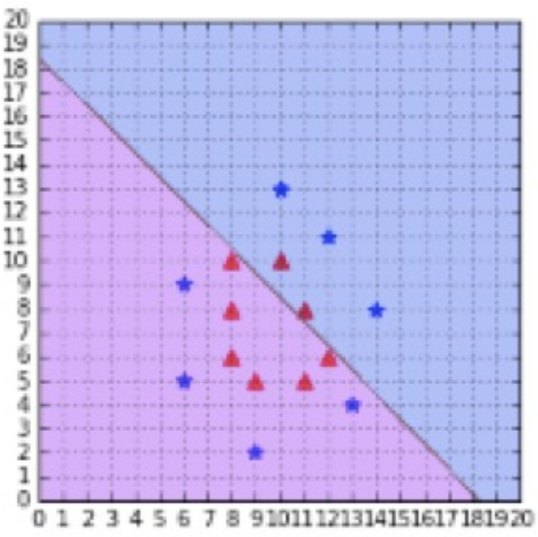


degree 1

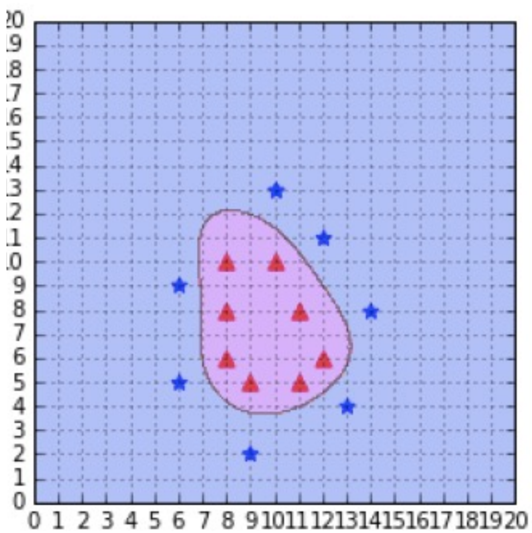


degree 6

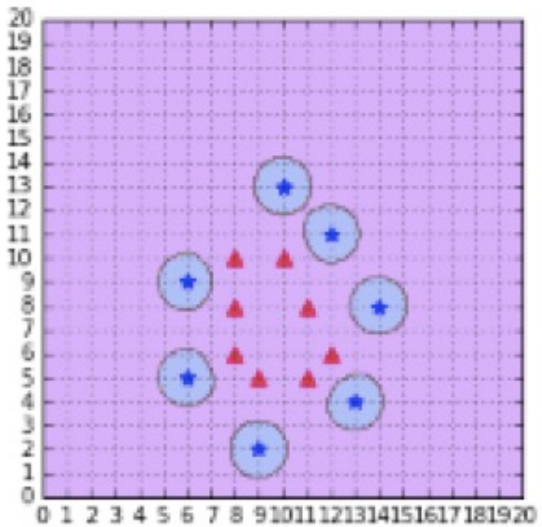
Radial Kernel



Radial σ = 1e-5



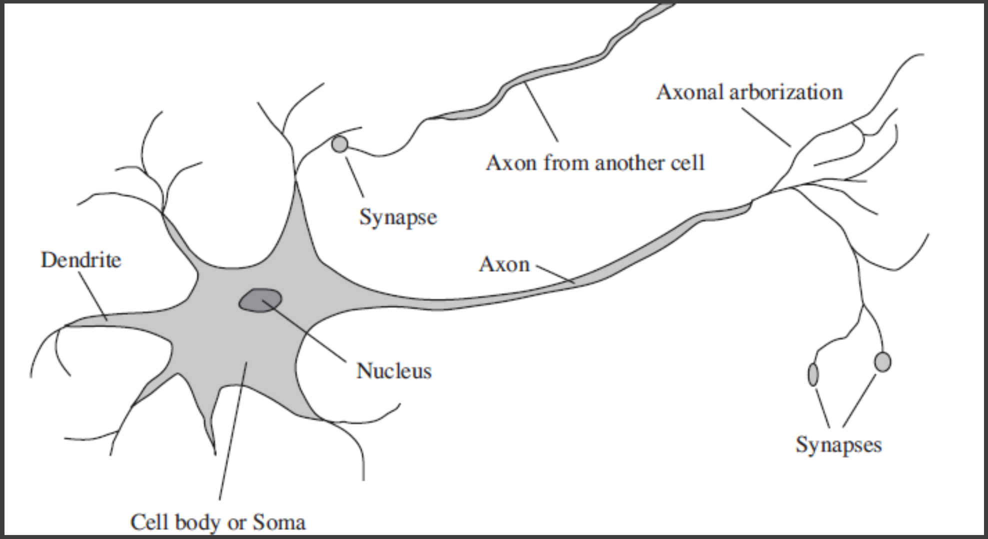
Radial σ = 0.1



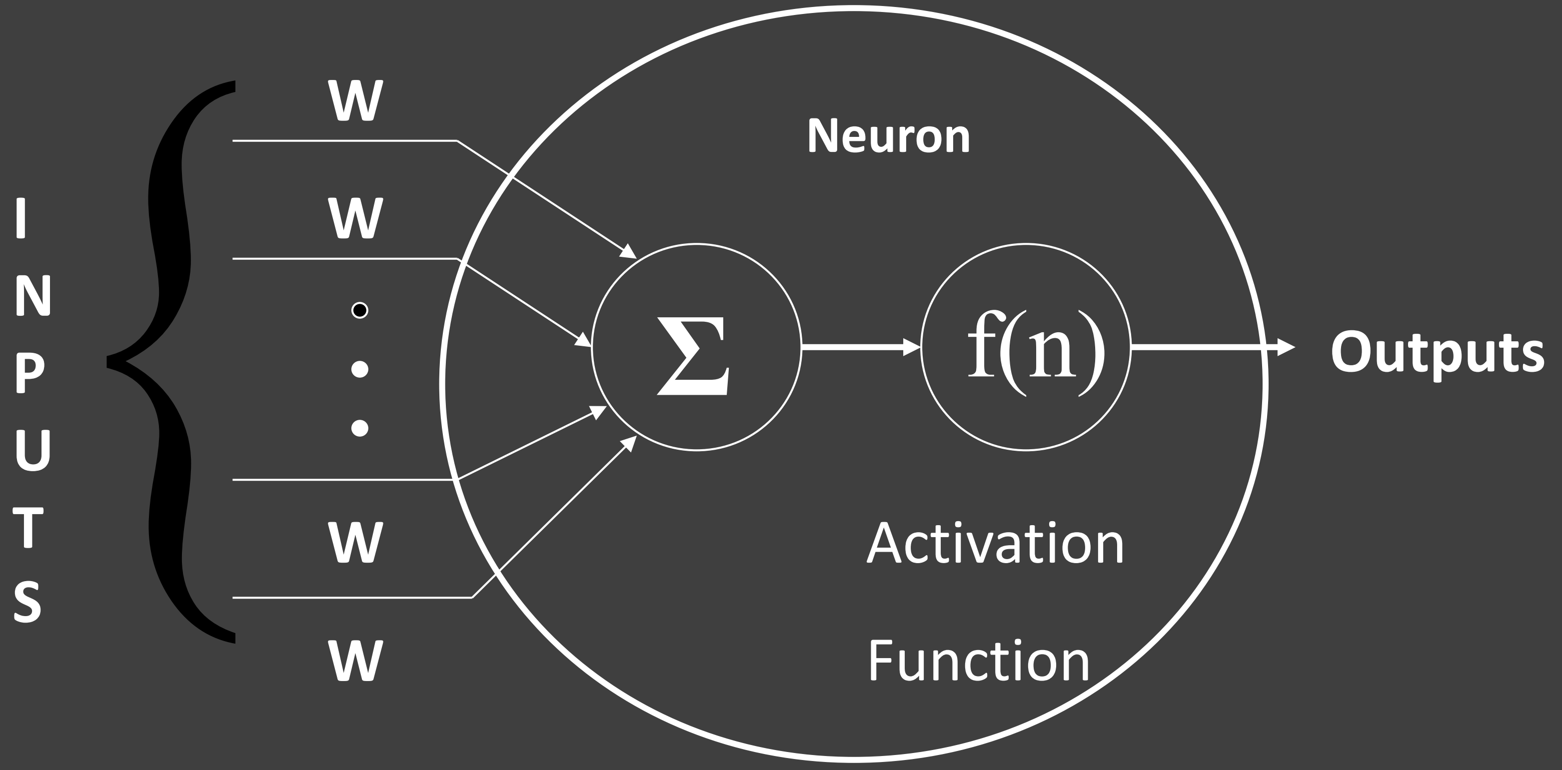
Radial σ = 2

σ low, linear SVM . σ high, overfitting

Neurons

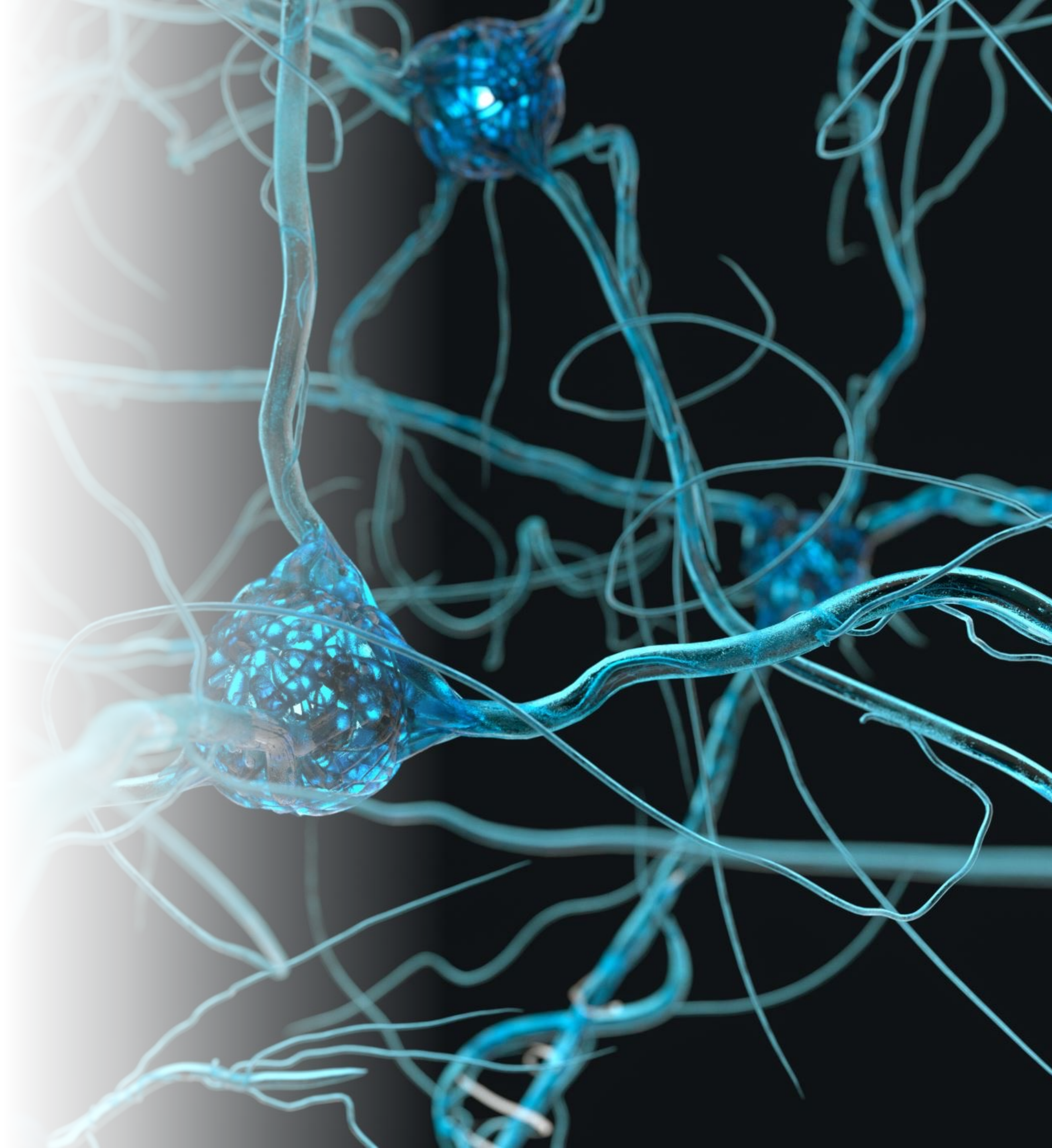

Axonal arborization
Axon from another cell
Synapse
Dendrite
Axon
Nucleus
Synapses
Cell body or Soma

Artificial Neuron



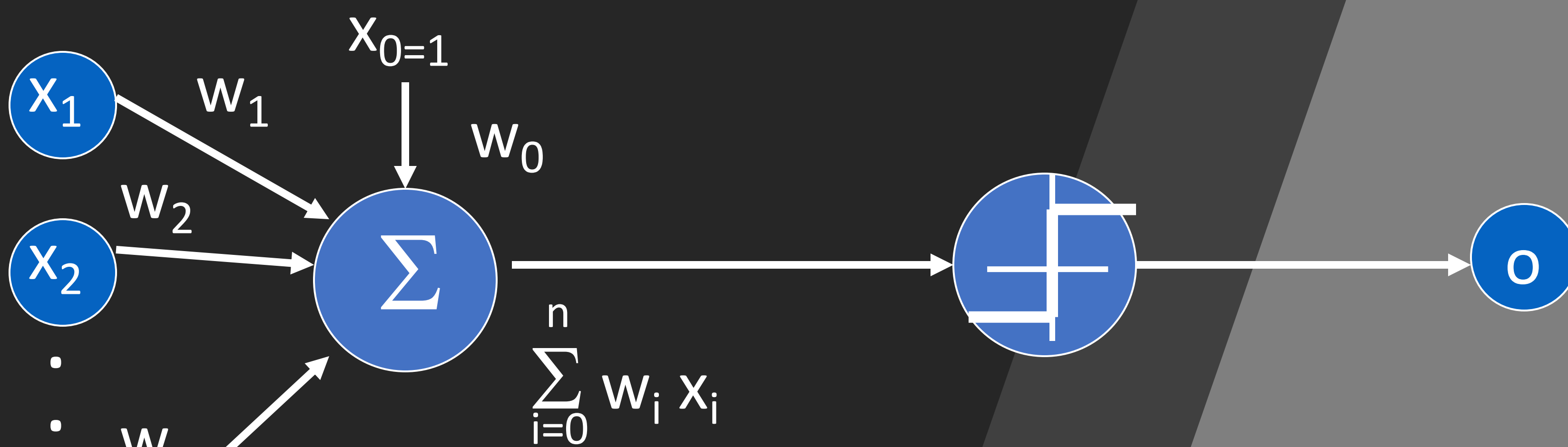
W=Weight

Neural networks



Perceptron

- Linear treshold unit (LTU)



$$o(x_i)=\begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$$

Perceptron Learning Rule

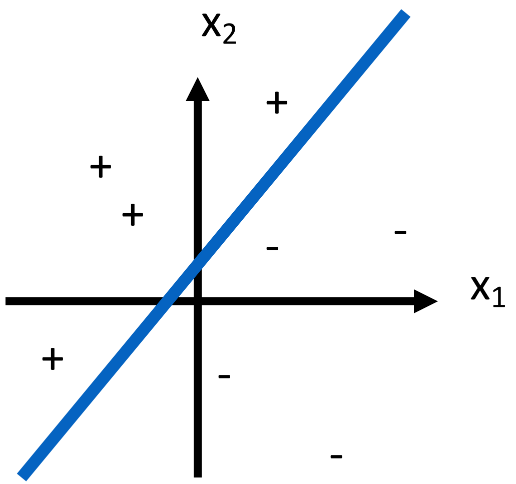
$w_i = w_i + \Delta w_i$

$\Delta w_i = \eta\ (t - o)\ x_i$

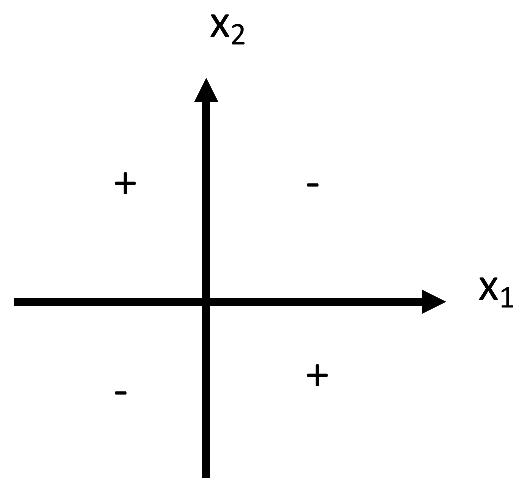
η *learning rate*

- If the output is incorrect ($t \neq o$) the weights w_i are changed such that the output of the perceptron for the new weights is *closer* to t.
- The algorithm converges to the correct classification
    - if the training data is linearly separable
    - and η is sufficiently small

Decision Surface of a Perceptron



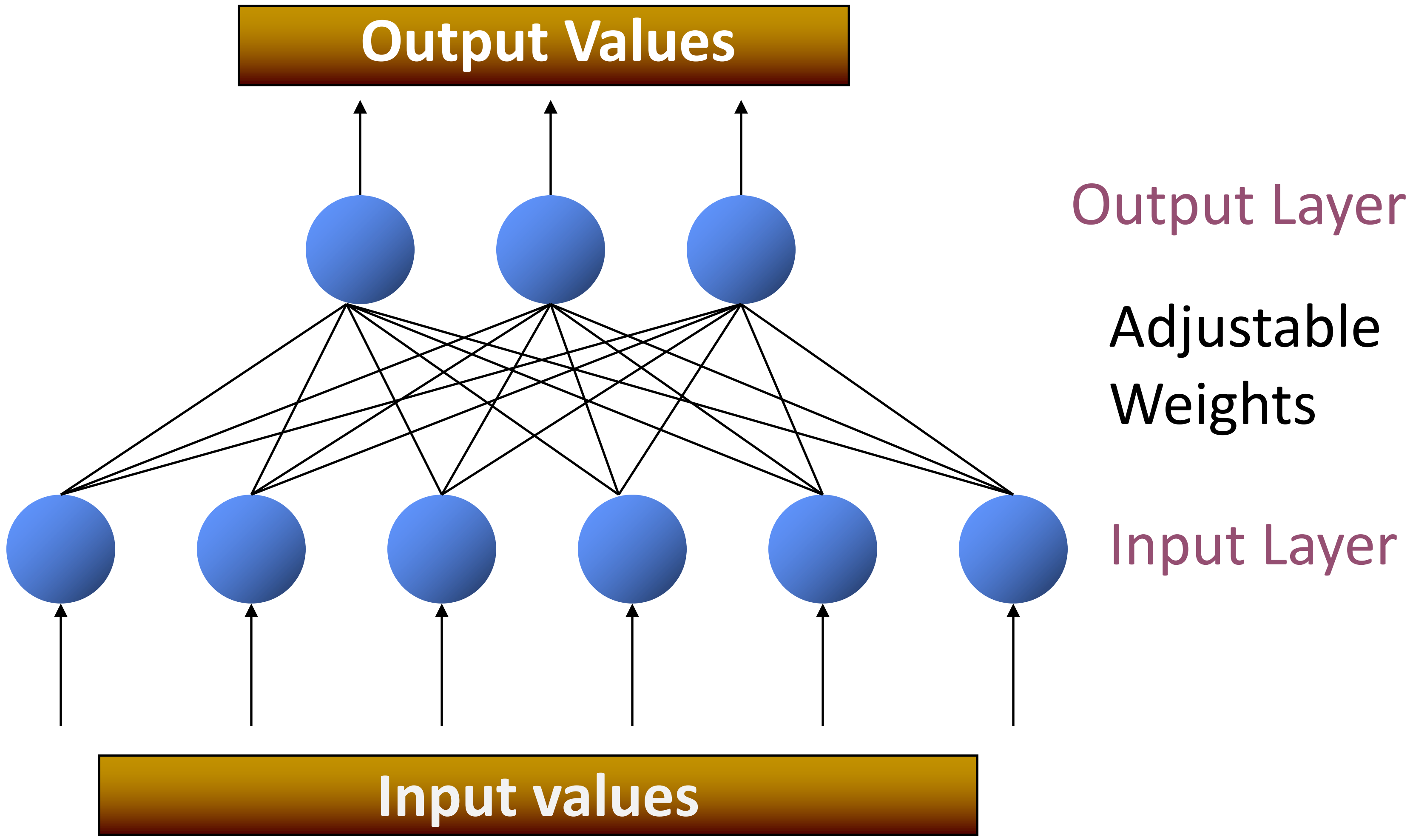
Linearly separable



Non-Linearly separable

- Perceptron is able to represent some useful functions
- AND(x_1,x_2) choose weights w_0=-1.5, w_1=1, w_2=1
- But functions that are not linearly separable (e.g. XOR) are not representable

Multi layer netwoks



NN for Machine Learning

1. Given training data:

$$\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$$

2. Choose:

 – Decision function

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

 – Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

3. Define goal:

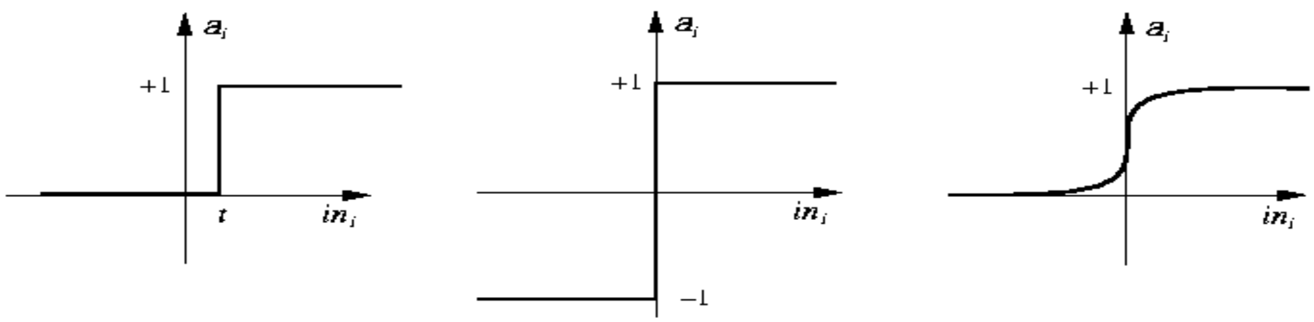
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{i=1}^N \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

4. Train

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

Multi layer networks

- Transforms neuron's input into output.
- Features of activation functions:
  - A squashing effect is required
    - Prevents accelerating growth of activation levels through the network.
  - Simple and easy to calculate



(a) Step function (b) Sign function (c) Sigmoid function

Multi layer networks

The hard-limiting threshold function

- Corresponds to the biological paradigm
  - either fires or not

Sigmoid functions ('S'-shaped curves)

- The logistic function

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

- The hyperbolic tangent (symmetrical)

Backpropagation

- Can theoretically perform “any” input-output mapping;
- Can learn to solve linearly inseparable problems.

Gradient descent

Backpropagation Algorithm

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - For each training example <$(x_1,\ldots x_n)$,y> Do
    - Input the instance $(x_1,\ldots,x_n)$ to the network and compute the network outputs o_k
    - For each output unit k
      - $\delta_k=o_k(1-o_k)(t_k-o_k)$
    - For each hidden unit h
      - $\delta_h=o_h(1-o_h)\sum_k w_{h,k}\,\delta_k$
    - For each network weight $w_{,j}$ Do
    - $w_{i,j}=w_{i,j}+\Delta w_{i,j}$ where

$$\Delta w_{i,j}=\eta\,\delta_j\,x_{i,j}$$

Neural Network Architectures

Even for a basic Neural Network, there are many design decisions to make:

1. # of hidden layers (depth)
2. # of units per hidden layer (width)
3. Type of activation function (nonlinearity)
4. How to update the weight

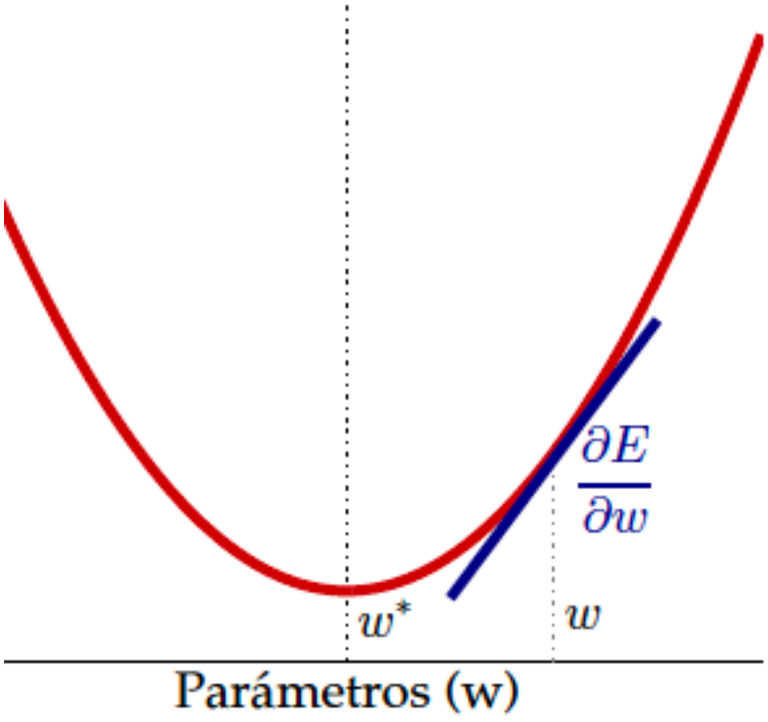
$$g(x) = \frac{1}{1-e^{-x}}$$

$$g(x) = x$$

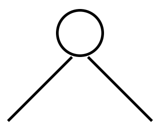
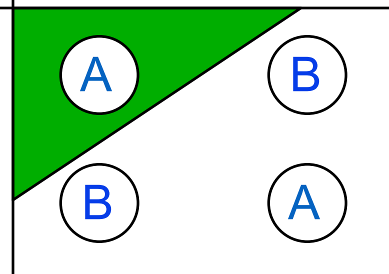
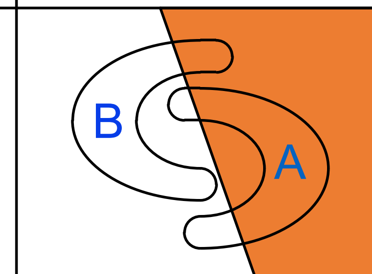
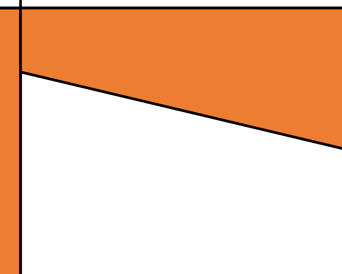
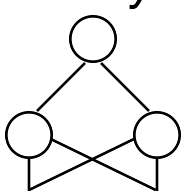
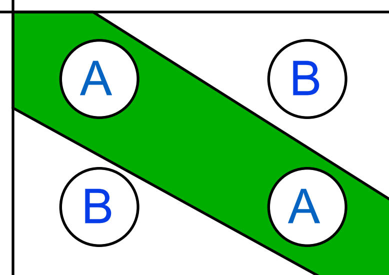
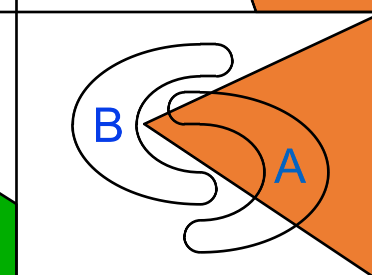
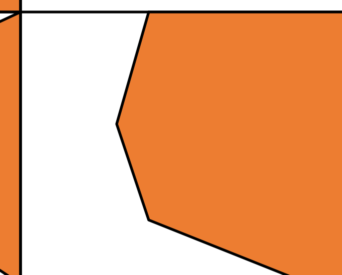
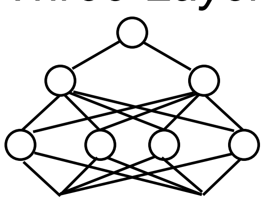
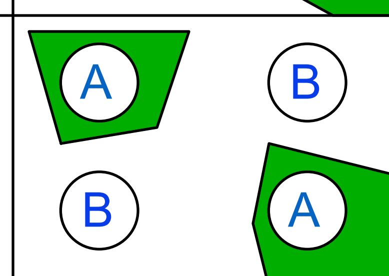
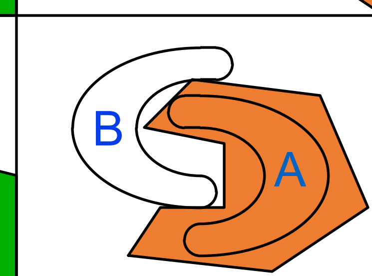
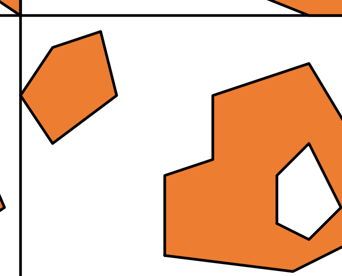
$$g(x) = \begin{cases} 1 \text{ if } \boldsymbol{wx}>0 \\ -1 \text{ otherwise} \end{cases}$$

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{d}\sum_{k}(y_{kd} - y'_{kd})^2$$

Gradient descent

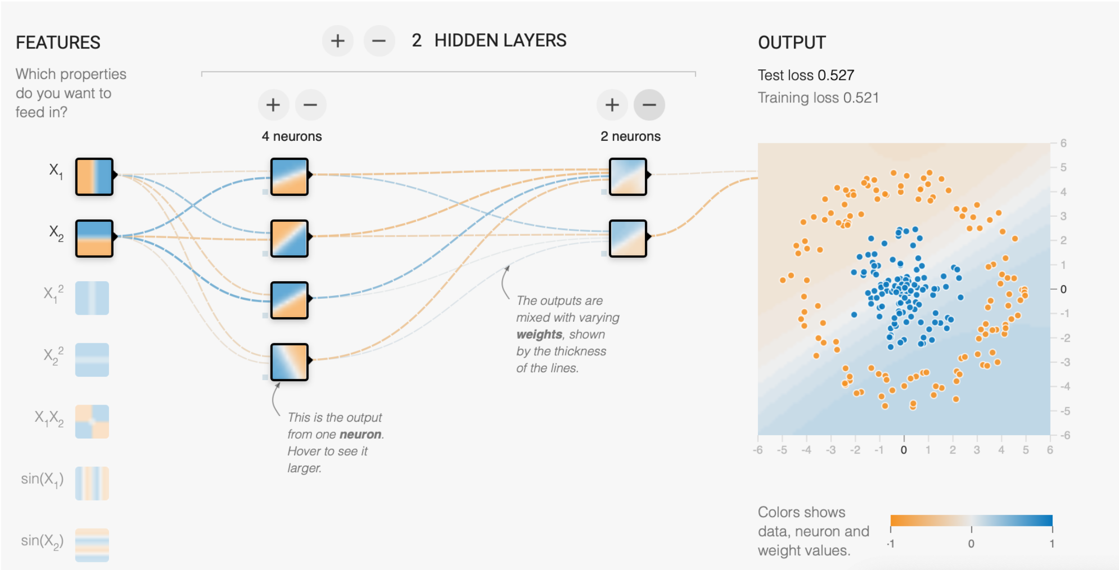


Which surfaces can we learn?

Structure	*Types of Decision Regions*	*Exclusive-OR Problem*	*Classes with Meshed regions*	*Most General Region Shapes*
Single-Layer 	*Half Plane Bounded By Hyperplane*			
Two-Layer 	*Convex Open Or Closed Regions*			
Three-Layer 	Arbitrary (Complexity Limited by No. of Nodes)			

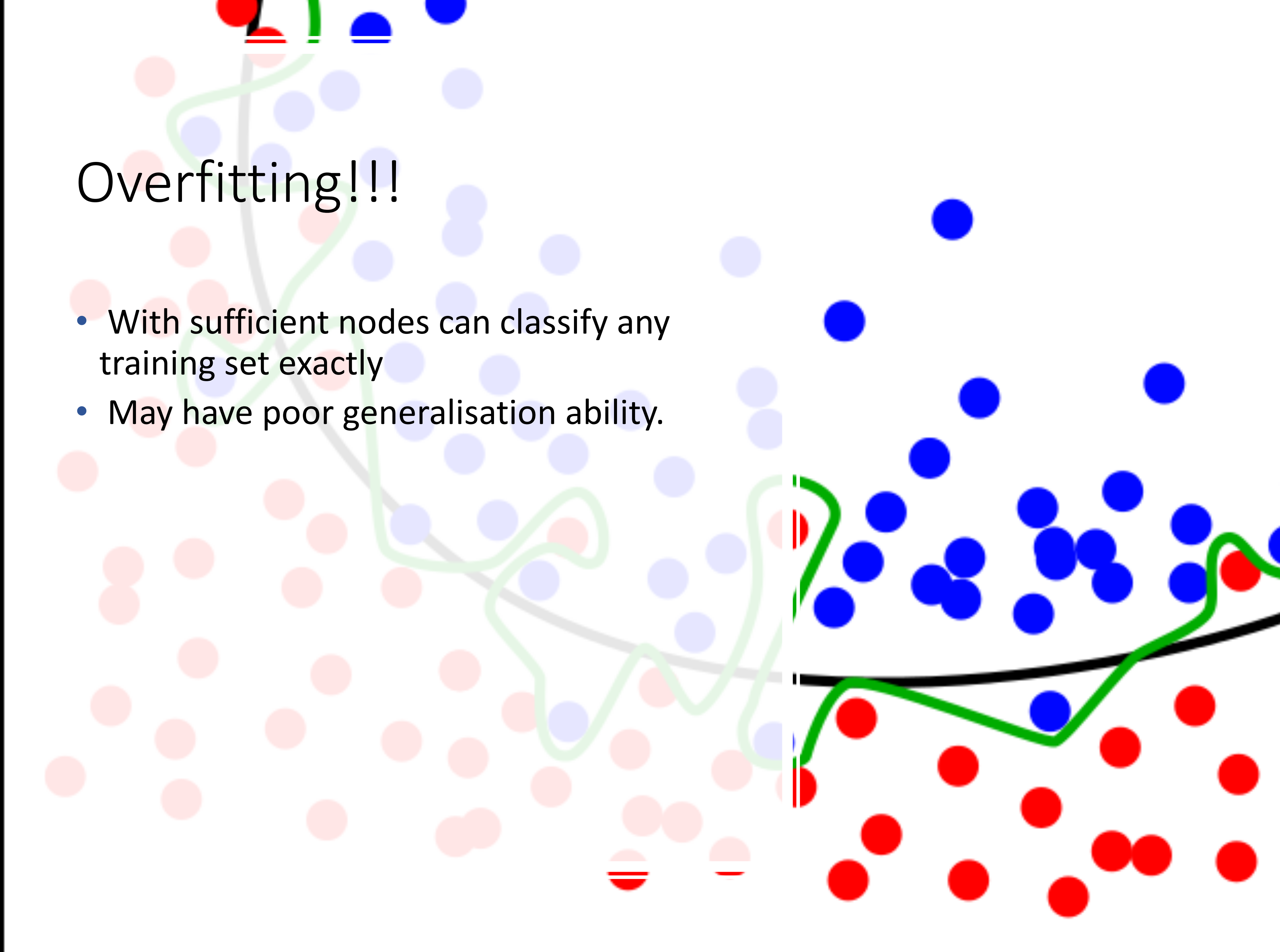
Expressive Capabilities of ANN

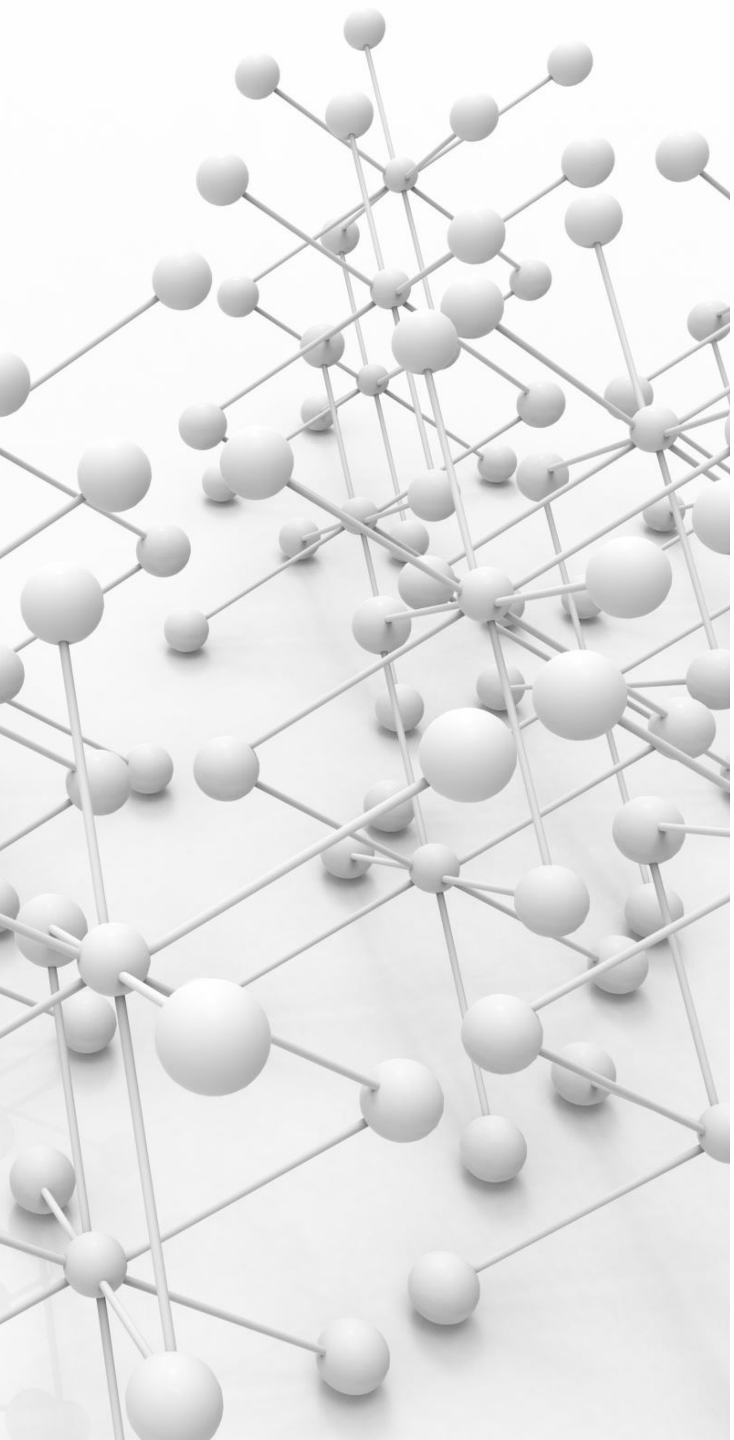
- With one hidden layer, it is possible to represent any boolean function or any continuous function
- With two hidden layers, it is possible to represent non continuous functions
- More complex problems… Deep learning



Overfitting!!!

- With sufficient nodes can classify any training set exactly
- May have poor generalisation ability.





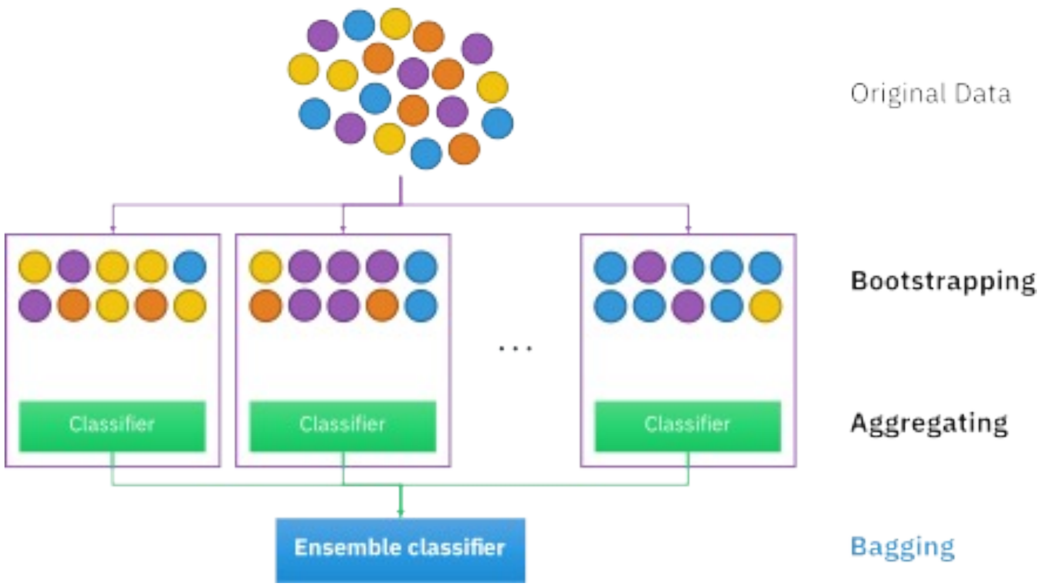
Evaluation

- How do the models generalize??

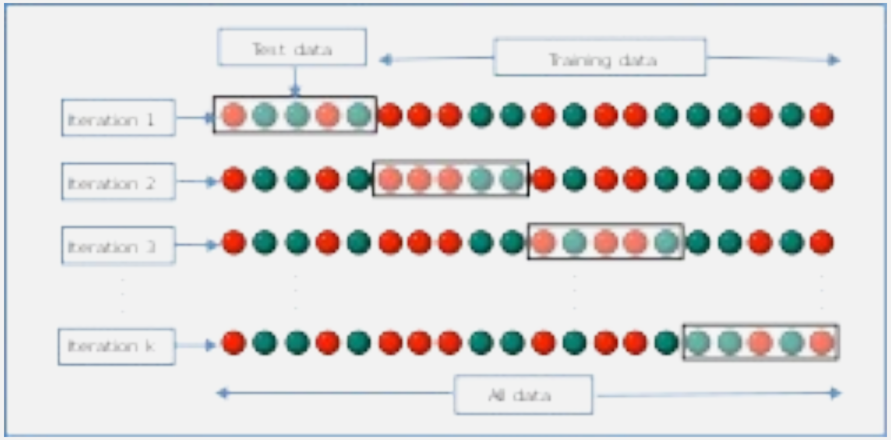
Training/test

- 1 data split
- Tipically 80% for training. 20% for testing
- **OK if we have enough dat**
- Otherwise, be careful with bias

Bootstrap

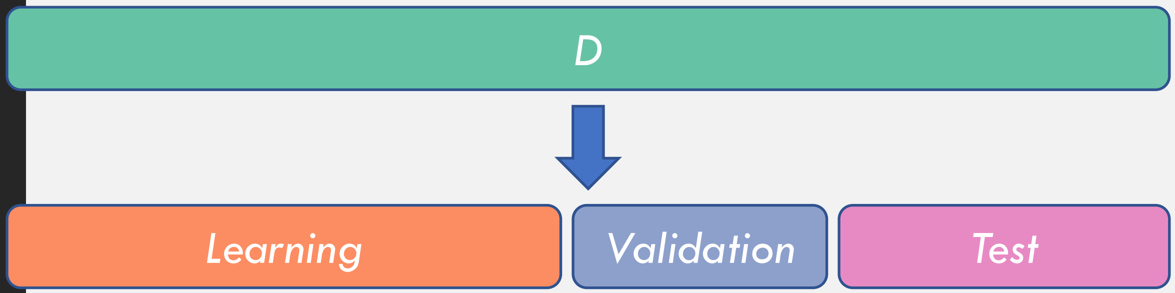

Original Data
Bootstrapping
Aggregating
Bagging
Classifier
Classifier
Classifier
Ensemble classifier

Cross validation



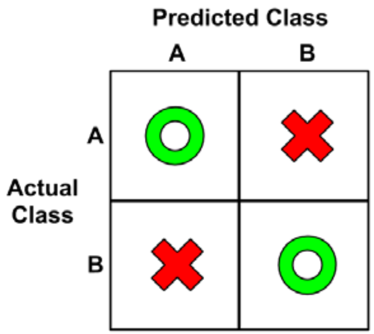
Meta Validation

- What to do if *h* functions require any parameter?
- We test several parameters and select the best
- How?

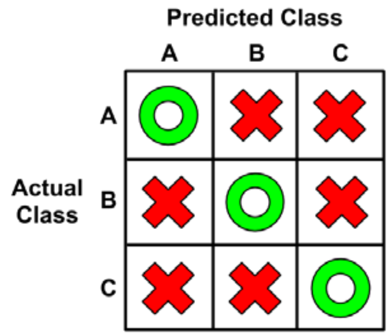


Evaluation

Binary problem



Multicategory problem



		Prediction	
		C_P	**C_N**
Truth	**C_P**	**TP**: True positive	**FN**: False negative
	C_N	**FP**: False positive	**TN**: True negative

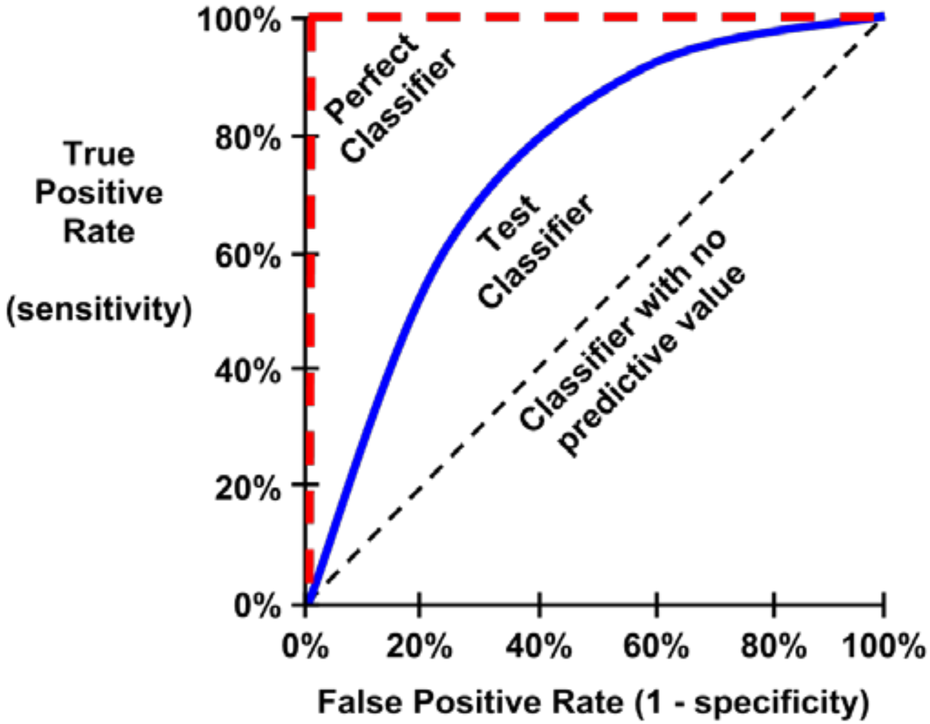
$$Accuracy = A = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precisión = P = \frac{TP}{TP + FP}$$

$$Cobertura = Recall = R = \frac{TP}{TP + FN}$$

$$F = \frac{2 * P * R}{P + R} = \frac{2 * TP}{2 * TP + FP + FN}$$

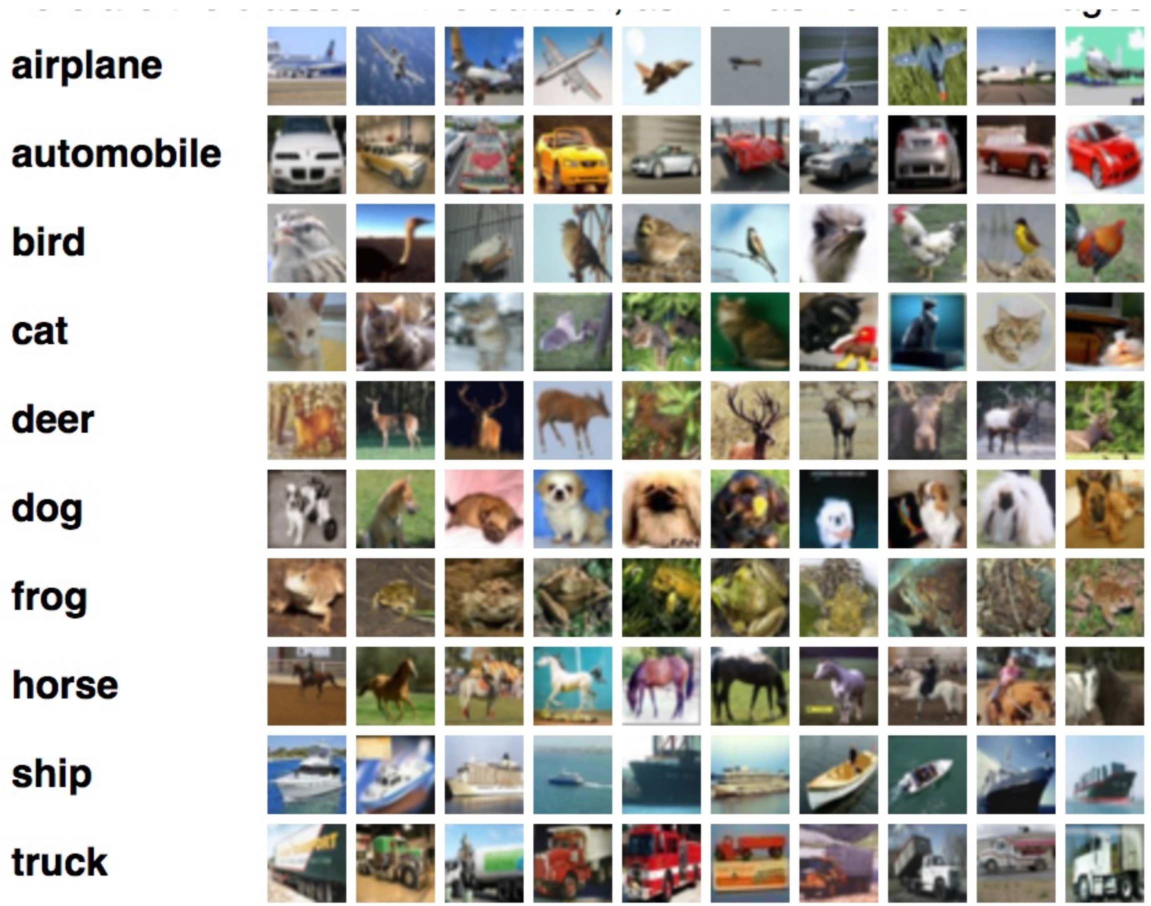
Evaluation



True Positive Rate
(sensitivity)
100%
80%
60%
40%
20%
0%
0%
20%
40%
60%
80%
100%
False Positive Rate (1 - specificity)
Perfect Classifier
Test Classifier
Classifier with no predictive value

Applications

Automatic image labelling

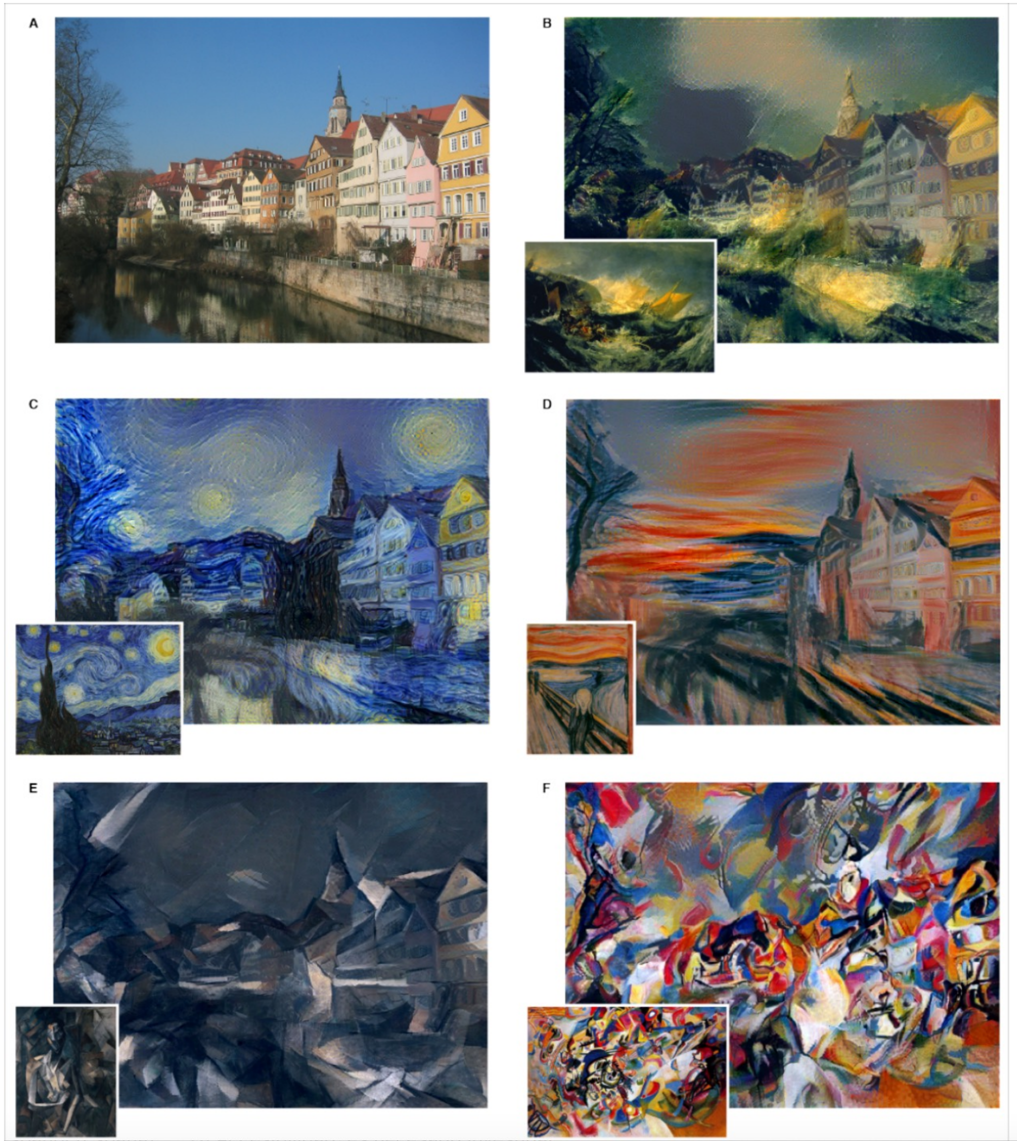


Object recognition

Example of Object Detection With Faster R-CNN on the MS COCO Dataset

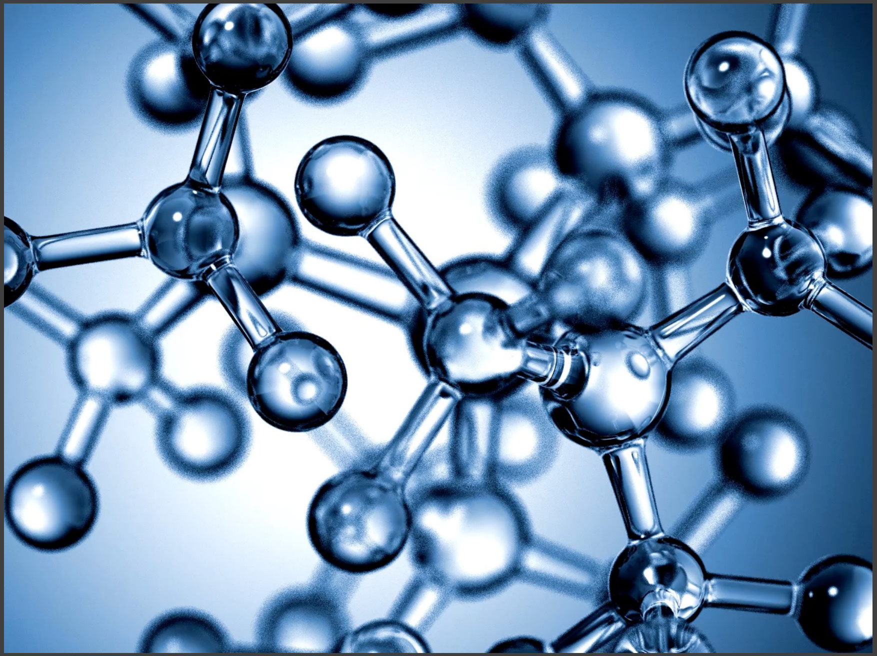
Image reconstruction

Style transfer



Medical Diagnosis

- **Treatment recomendation**
- **Recognition of cancerous cells**
- **Identification of features related to a disease**



References

- Tom Mitchell. *Machine Learning*. McGraw-Hill
- Ethem Alpaydin. *Introduction to machine learning*. The MIT Press
- Christopher Bishop. *Pattern recognition and machine learning*. Springer