B12
OUTSIDE
THE BOX

# Making the data talk. Loud and clear.

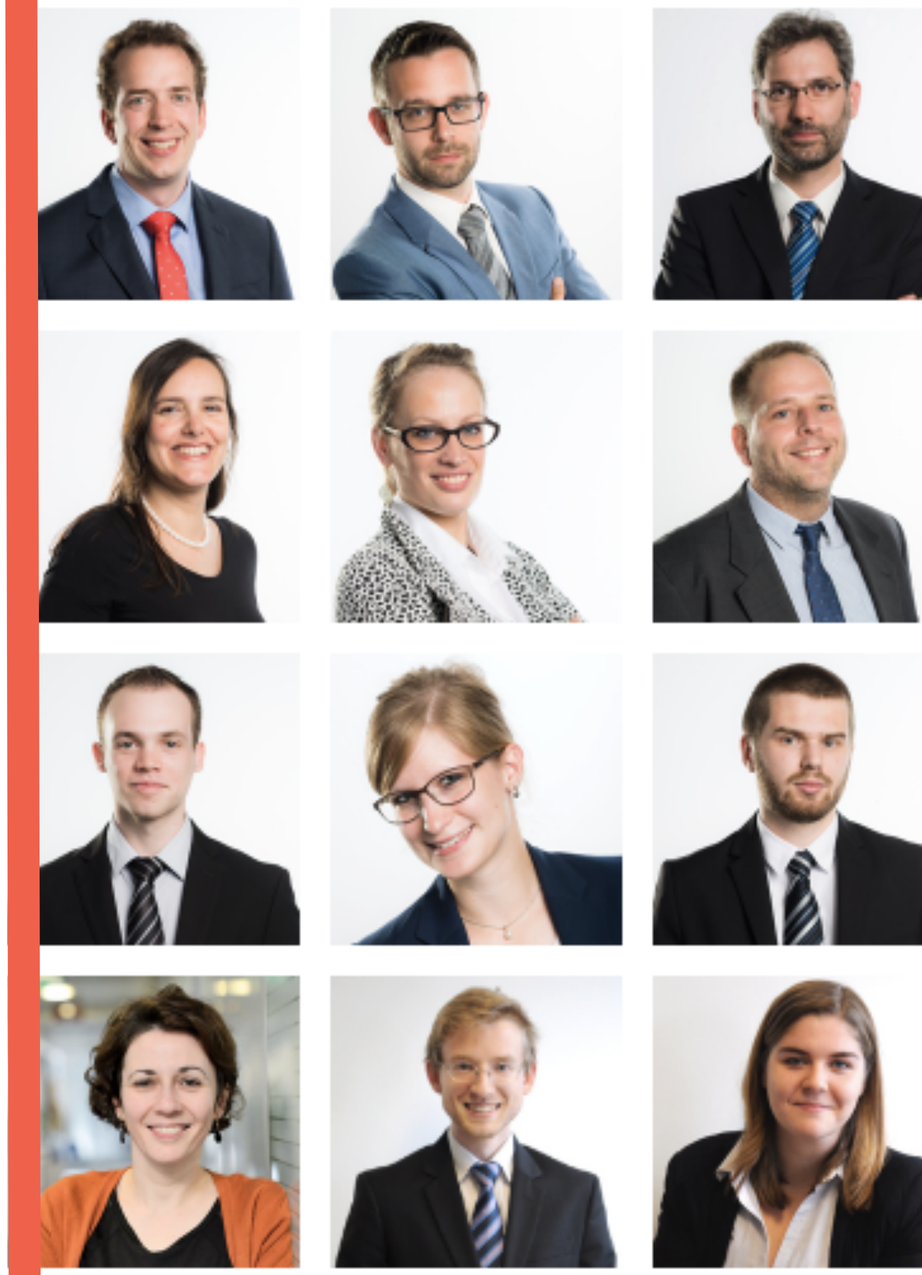
Introduction to Machine Learning and its "business" applications

Michel HERQUET - B12 Consulting - MCNet - 6/7/2017

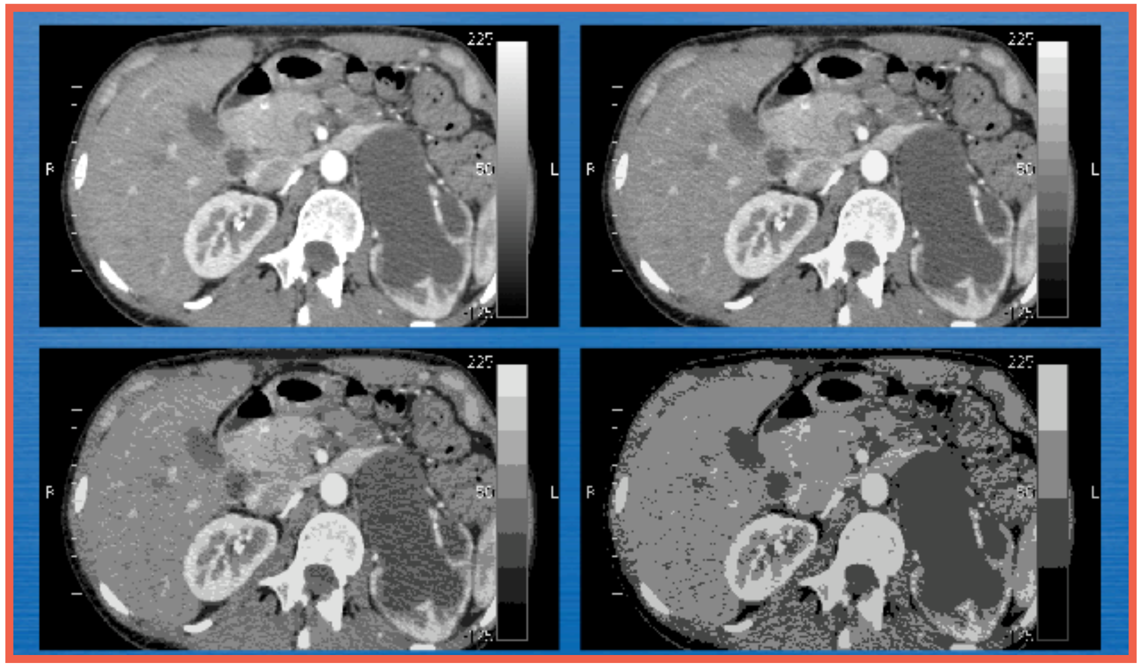
B12
OUTSIDE
THE BOX

# B12 CONSULTING

- Data Analytics/Software Development **service** firm funded in 2012 by three Physics PhD’s
- Team of over 10 consultants mainly **scientific backgrounds** in quantitive Sciences including Computer Science, Mathematics, Engineering and… Physics!
- **Continuous learning** environment focusing on most recent technical and technological advances
- **“Outside the box”** approach emphasising custom solution tailored to our Clients exact needs over standardised products
- Specific **expertise** in:
    - Algorithm development (Python, C++, etc.)
    - Machine learning algorithms (supervised and unsupervised, including Deep Learning)
    - “Big” data processing and visualisation technologies

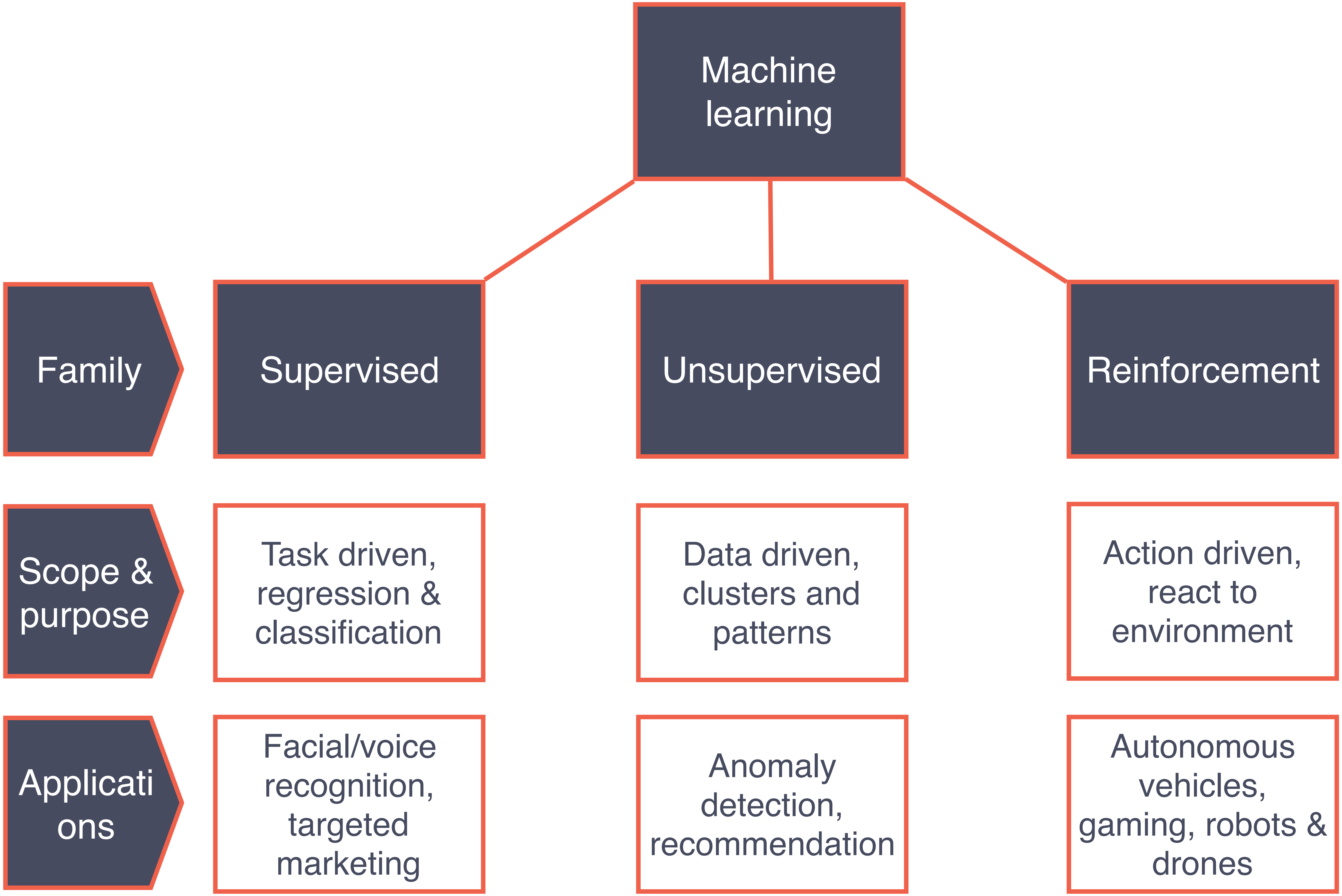


B12 OUTSIDE THE BOX



# Machine Learning is... everywhere

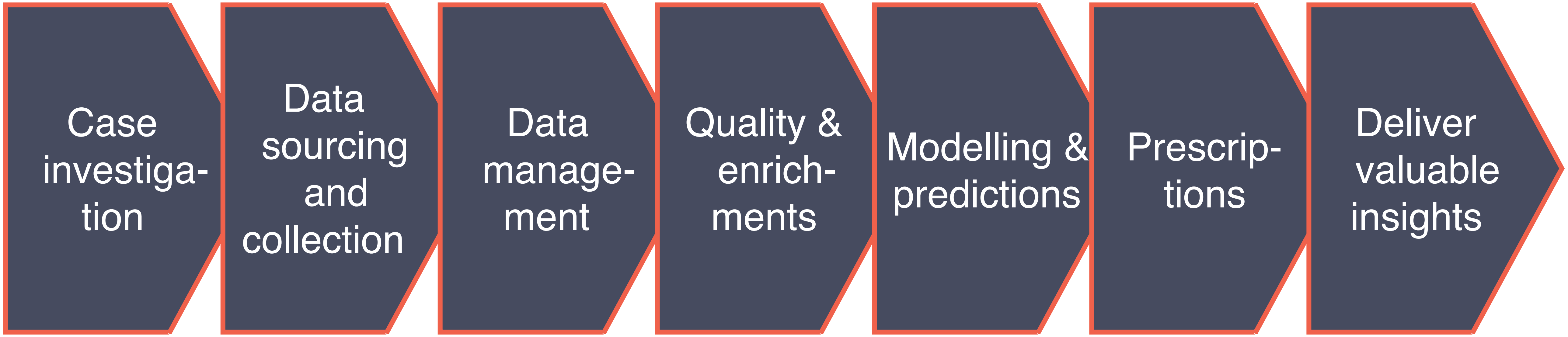
# THE MACHINE LEARNING ALGORITHM FAMILIES



Machine learning

| Family | Supervised | Unsupervised | Reinforcement |
|---|---|---|---|
| Scope & purpose | Task driven, regression & classification | Data driven, clusters and patterns | Action driven, react to environment |
| Applications | Facial/voice recognition, targeted marketing | Anomaly detection, recommendation | Autonomous vehicles, gaming, robots & drones |

# TYPICAL MACHINE LEARNING PROJECT FLOW



Case investigation

Data sourcing and collection

Data management

Quality & enrichments

Modelling & predictions

Prescriptions

Deliver valuable insights

# Selected business cases

B12 OUTSIDE THE BOX

# CHEMICAL PRODUCTION PROCESS IMPROVEMENT

1

- **Client**: Major player of the chemical industry
- **Business challenge**: Improve the quality of a specific end product with high technological applications
- **Resolution**: Quantitative study on real production data pointing out different specific issues and suggesting improvement strategies
- **Technologies**: Production process modelling and offline supervised Machine Learning analysis (Decision Trees) in Python starting from static data files
- **Outcome**: recommendations implemented and project positioned internally as a starting point for a global data valorisation effort

# DECISION TREES (1980'S)

- A decision tree is a **flowchart-like structure** in which internal node represents a « test » on an attribute, each branch represents the outcome of the test and each leaf node represents a class label

- Several building algorithms exists, most of them leveraging the notion of **information entropy**

- Most popular ones are ID3, C4.5, C5 (R.Quinlan) and CART

# ID3 ALGORITHM

- **Information entropy (Shannon, 1948)**
  - Average **amount of information** contained in a sample drawn from a distribution or data stream. It characterises our **uncertainty** about our source of information. If entropy decreases after adding information, change is called an **information gain**

$$E(S)=\sum_{i=1}^{c}-p_i\log_2 p_i$$

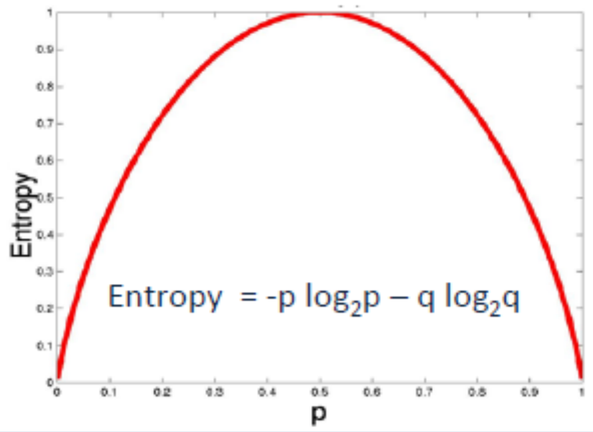
| Play Golf | |
|---|---|
| Yes | No |
| 9 | 5 |

**Entropy(PlayGolf) =** Entropy (5,9)

= Entropy (0.36, 0.64)

= - (0.36 $\log_2$ 0.36) - (0.64 $\log_2$ O.64)

= 0.94



Entropy = -0.5 $\log_2$0.5 – 0.5 $\log_2$0.5 = 1

$$E(T,X)=\sum_{c\in X}P(c)E(c)$$

| | | Play Golf | | |
|---|---|---|---|---|
| | | Yes | No | |
| Outlook | Sunny | 3 | 2 | 5 |
| | Overcast | 4 | 0 | 4 |
| | Rainy | 2 | 3 | 5 |
| | | | | 14 |

**E**(PlayGolf, Outlook) = **P**(Sunny)***E**(3,2) + **P**(Overcast)***E**(4,0) + **P**(Rainy)***E**(2,3)

= (5/14)*0.971 + (4/14)*0.0 + (5/14)*0.971

= 0.693

# ID3 ALGORITHM (CTD.)

Choose attribute splitting which highest information gain

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Outlook | Sunny | 3 | 2 |
| | Overcas | 4 | 0 |
| | Rainy | 2 | 3 |
| Gain = 0.247 | | | |

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Temp. | Hot | 2 | 2 |
| | Mild | 4 | 2 |
| | Cool | 3 | 1 |
| Gain = 0.029 | | | |

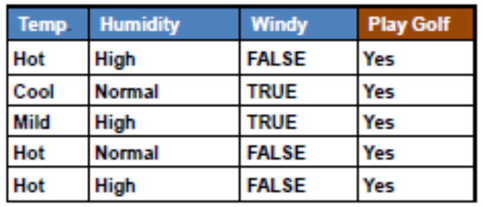
| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Humidity | High | 3 | 4 |
| | Normal | 6 | 1 |
| Gain = 0.152 | | | |

| | | Play Golf | |
|---|---|---|---|
| | | Yes | No |
| Windy | False | 6 | 2 |
| | True | 3 | 3 |
| Gain = 0.048 | | | |

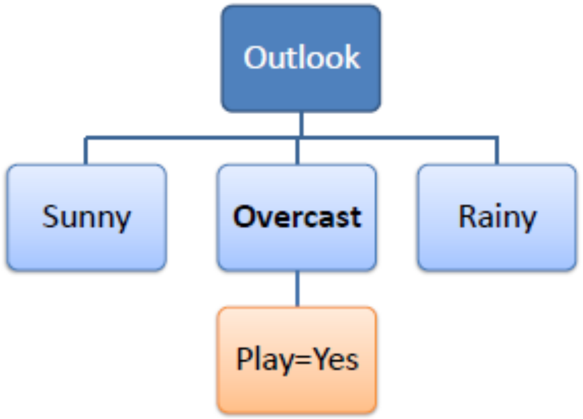
$$\boxed{Gain(T,X) = Entropy(T) - Entropy(T,X)}$$

**G**(PlayGolf, outlook) = **E**(PlayGolf) - E(PlayGolf, Outlook)
= 0.940 - 0.693 = 0.247

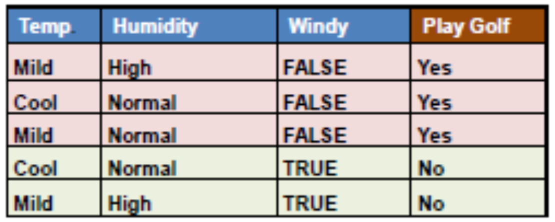
- A branch with entropy of 0 is a leaf node



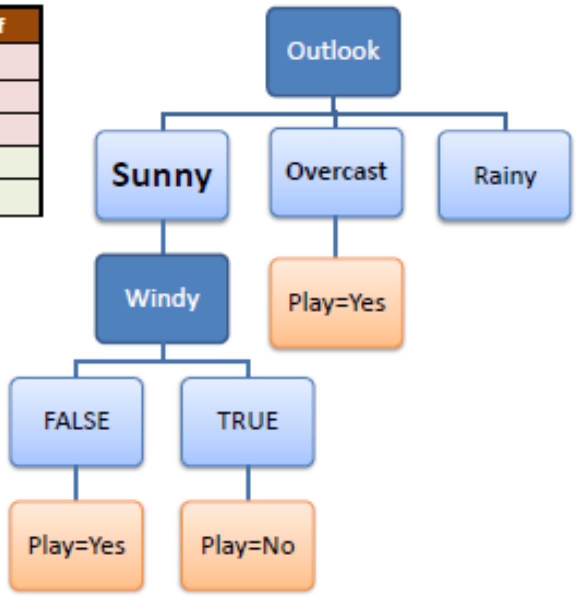
| Temp. | Humidity | Windy | Play Golf |
|---|---|---|---|
| Hot | High | FALSE | Yes |
| Cool | Normal | TRUE | Yes |
| Mild | High | TRUE | Yes |
| Hot | Normal | FALSE | Yes |
| Hot | High | FALSE | Yes |



- A branch with entropy more than 0 needs further splitting



| Temp. | Humidity | Windy | Play Golf |
|---|---|---|---|
| Mild | High | FALSE | Yes |
| Cool | Normal | FALSE | Yes |
| Mild | Normal | FALSE | Yes |
| Cool | Normal | TRUE | No |
| Mild | High | TRUE | No |



# CAR FUEL CONSUMPTION OPTIMISATION

- **Client**: Major actor of the automotive industry
- **Business challenge**: Leverage data generated by cars to dynamically decrease fuel consumption
- **Resolution**: Precise fuel consumption modelling based on actual car data, and optimisation strategy definition and implementation
- **Technologies**: Supervised (SVR) and unsupervised (k-Mean) Machine Learning techniques in Python, Big Data database technologies
- **Outcome**: Patentable algorithms included in a prototype product which helped to successfully demonstrate the relevance of the approach

# SUPPORT VECTOR MACHINES (BOSER, GUYON, VAPNIK 1992)

- **Goal** : find the hyperplane (i.e. decision boundary) linearly separating classes
- **Basic idea** : maximise distance between the two boundaries demarcating the classes

$$\frac{2}{\sqrt{\mathbf{w}^{\mathbf{T}}\mathbf{w}}} \longrightarrow$$

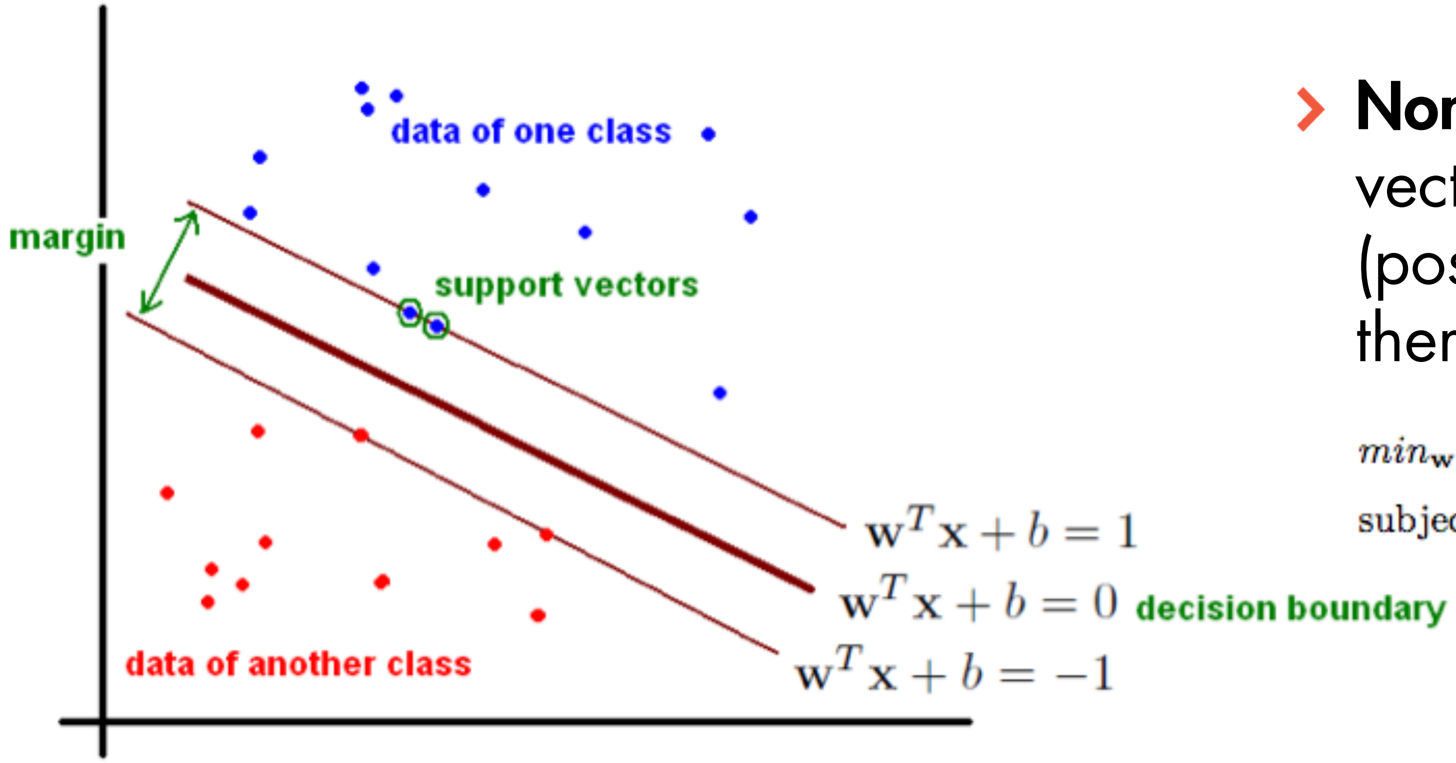
$$min_{\mathbf{w},b} \frac{\mathbf{w}^{\mathbf{T}}\mathbf{w}}{2}$$

subject to: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1$ ($\forall$ data points $\mathbf{x_i}$).

- **Soft margin exception :** allow some data points of one class to appear on the other side of the boundary (slack variables)

$$min_{\mathbf{w},b,\epsilon} \frac{\mathbf{w}^{\mathbf{T}}\mathbf{w}}{2} + C\sum_i \epsilon_i$$

subject to: $y_i(\mathbf{w}^T\mathbf{x_i} + b) \geq 1 - \epsilon_i$ and $\epsilon_i \geq 0$ ($\forall$ data points $\mathbf{x_i}$).


data of one class
margin
support vectors
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$ decision boundary
data of another class
$\mathbf{w}^T\mathbf{x} + b = -1$

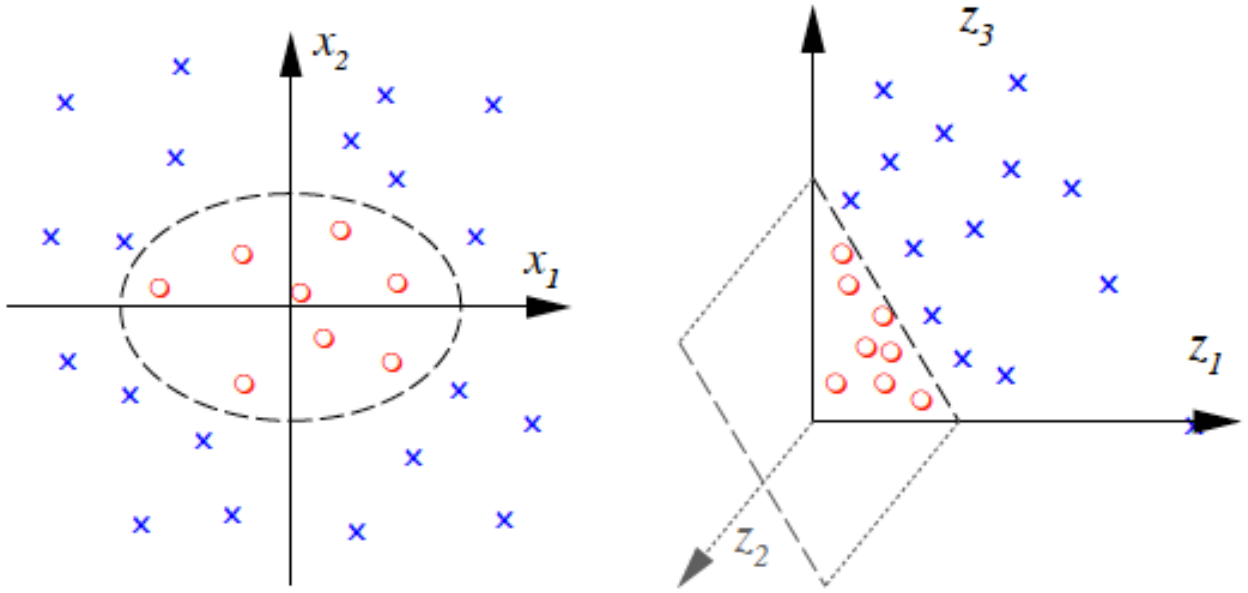
- **Non linear decision boundaries :** Data vectors are mapped to higher dimensional (possibly) infinite feature space to make them linearly separable in that space

$$min_{\mathbf{w},b,\epsilon} \frac{\mathbf{w}^{\mathbf{T}}\mathbf{w}}{2} + C\sum_i \epsilon_i$$

subject to: $y_i(\mathbf{w}^T\phi(\mathbf{x_i}) + b) \geq 1 - \epsilon_i$ and $\epsilon_i \geq 0$ ($\forall$ data points $\mathbf{x_i}$).

# SUPPORT VECTOR MACHINES (CTD.)

$$\Phi : R^2 \to R^3$$
$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{(2)}x_1x_2, x_2^2)$$



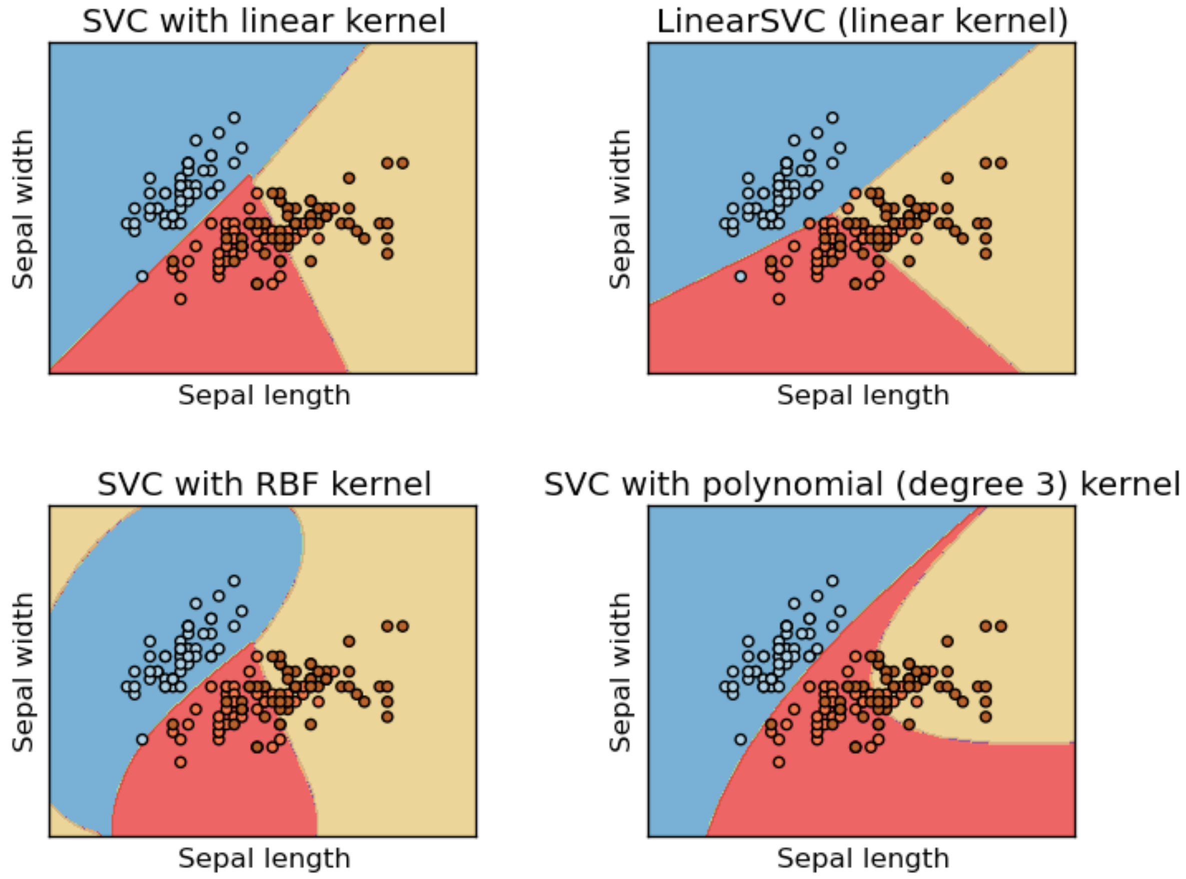
- **Kernel trick :** solving the above linear optimisation problem implies to be able to quickly compute scalar products such as

$$\boxed{\phi(\mathbf{x_i})^T\phi(\mathbf{x_j})}$$

this can be intractable in high dimensional spaces, but it turns out that there are special kernel functions that operates on the lower dimensional space to produce an equivalent measure

$$\boxed{K(\mathbf{x_i}, \mathbf{x_j}) = \phi(\mathbf{x_i})^T\phi(\mathbf{x_j})}$$

# SUPPORT VECTOR MACHINES (CTD.)



Gaussian RBF

$$k(\boldsymbol{x},\boldsymbol{z}) = \exp\left(\frac{-\|\boldsymbol{x}-\boldsymbol{z}\|^2}{c}\right)$$

Polynomial

$$k(\boldsymbol{x},\boldsymbol{z}) = \left(\left(\boldsymbol{x}^\top \boldsymbol{z}\right)+\theta\right)^d$$

Sigmoidal

$$k(\boldsymbol{x},\boldsymbol{z}) = \tanh\left(\kappa\left(\boldsymbol{x}^\top \boldsymbol{z}\right)+\theta\right)$$

Inverse multi-quadric

$$k(\boldsymbol{x},\boldsymbol{z}) = \frac{1}{\sqrt{\|\boldsymbol{x}-\boldsymbol{z}\|^2+c^2}}$$

- SVM can also be used for regression purposes and prediction (Support Vector Regression, or SVR)

# ANOMALY DETECTION IN RETAIL DATA

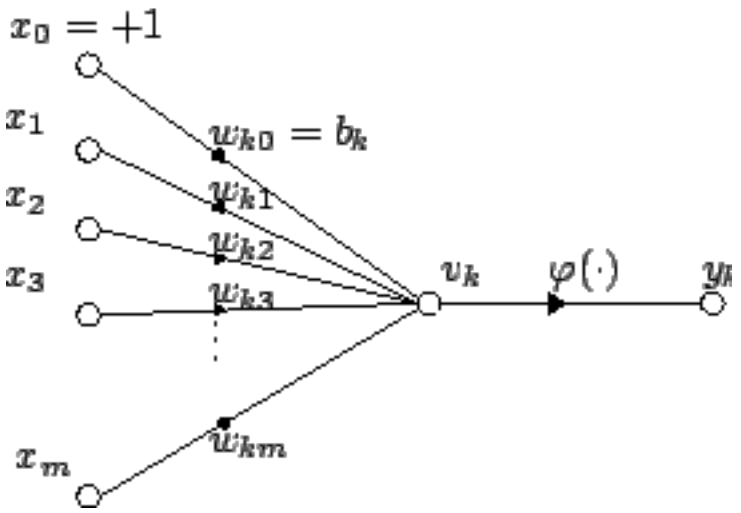
- **Client**: Mid size player in a local healthcare retail industry
- **Business challenge**: Isolate and characterise a large variety of anomalies following unknown patterns in the context of sales data
- **Resolution**: Design and implementation a new type of smart algorithm to automatically identify and report anormal recurring patterns
- **Technologies**: Unsupervised Machine Learning techniques in Python (autoencoders), large data files advanced manipulation technologies and connectors to business data infrastructure
- **Outcome**: Relevance of the approach proven by detecting specific anomalies in real data. Ongoing solution productisation effort

# ARTIFICIAL NEURAL NETWORKS

- Computational models **inspired by an animal's central nervous systems** (in particular the brain)
- Systems of **interconnected "neurones"** which can compute values from inputs
- Two defining properties :
  - consist of sets of **adaptive weights**, i.e. numerical parameters that are tuned by a learning algorithm
  - capable of approximating **non-linear** functions of their inputs
- Frequently use(d) for **pattern recognition**, e.g., computer vision
- Used for supervised learning (categorisation and regression), unsupervised learning (e.g., autoencoders for anomaly detection) and reinforced learning

# THE PERCEPTRON ALGORITHM (1958)

System of interconnected neurones that compute values from inputs by feeding information through the network



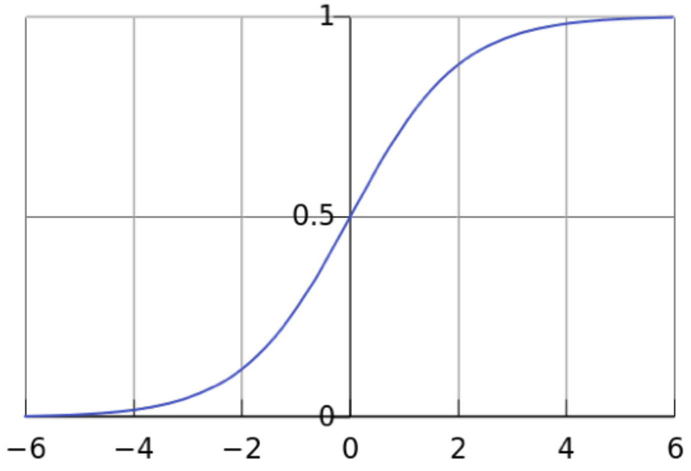
Basic structure of an artificial neurone:

One or more inputs are received and summed up (weighted) passing through a transfer function to produce the output

Neuron output

$$y_k = \varphi\left(\sum_{j=0}^{m} w_{kj}\, x_j\right)$$

Transfer function **φ**



# THE PERCEPTRON ALGORITHM (CTD.)

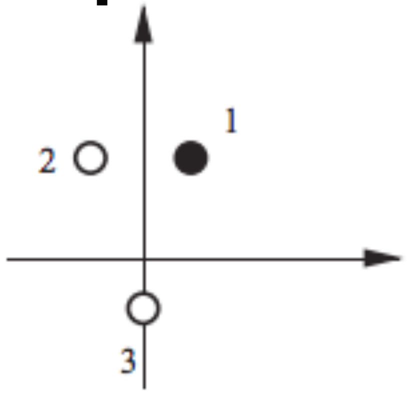
## The perceptron algorithm

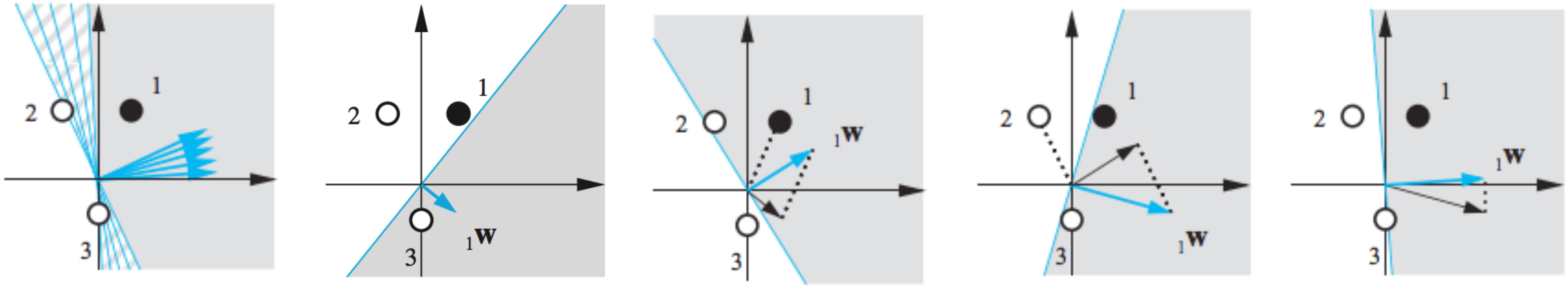
- Binary classifier that maps inputs x to one of possible output values f(x), e.g., true or false

## Training

- For each point in training set :
    - calculate actual output and
    - update the weights

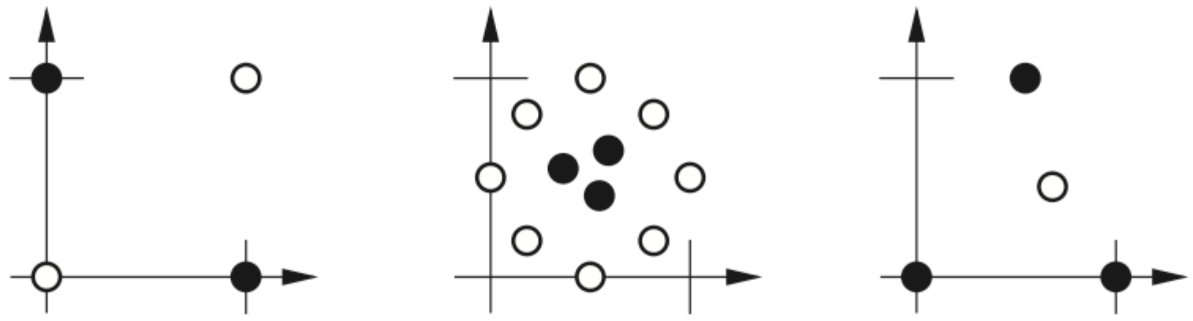
## Example

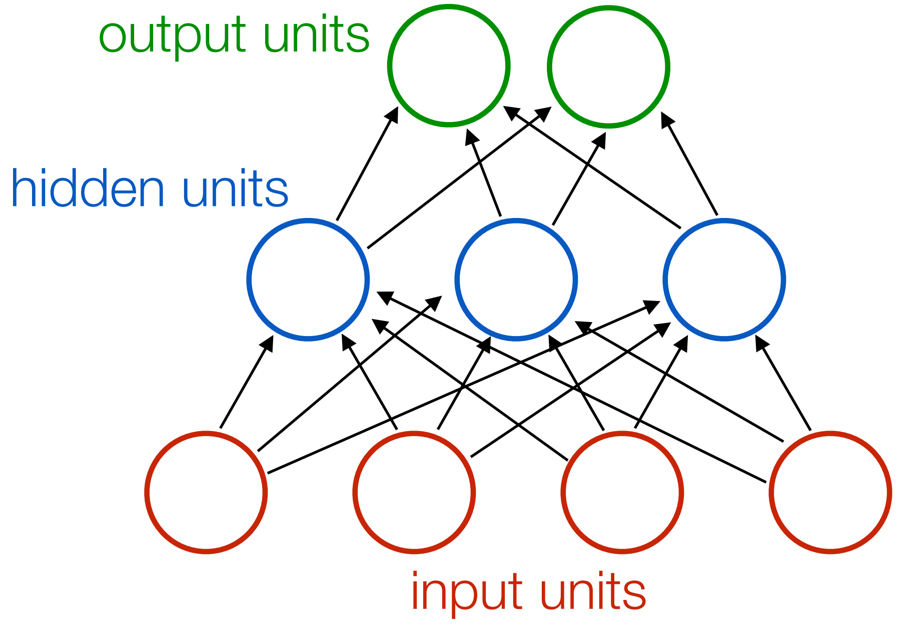




# BEYOND THE PERCEPTRON

- Perceptron problem: linearly inseparable (related to the so-called Vapnik–Chervonenkis dimension)





- Solution: multilayer perceptrons (units)

When hidden layers exist, perceptron algorithm cannot be applied.

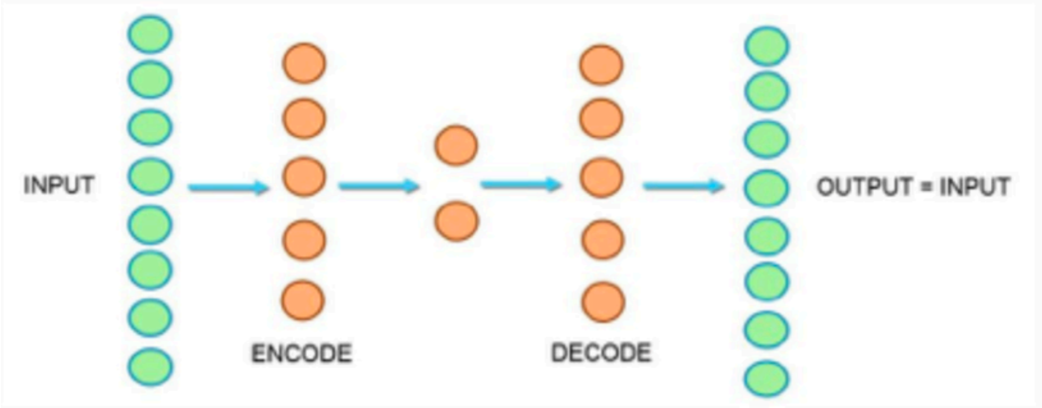
**Back propagation algorithm is one possible solution:**

Modify the weights of the connections leading to the hidden units based on the strength of each unit's contribution to the final prediction... But original attempts had a lot of issues.

**Recent breakthrough (Deep Learning) thanks to the explosion of available computer ressources (data, calculation)**

# AUTOENCODERS

## A particular type of n.n. :



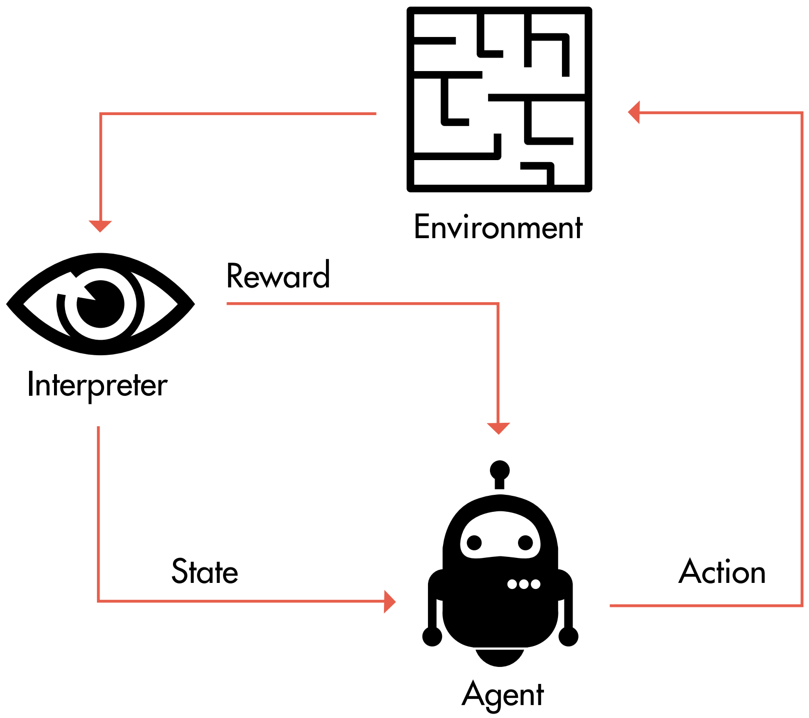
- Structure of a n.n. with autoencoder :
  - **Input data** (features)
  - **Hidden layer(s)** : compress the input data in a lower dimensional space —> obtain the true nature of data without uninteresting features and noise
  - **Output nodes** : from the hidden layer(s) reconstruct the original space
- In simple words, the autoencoder learns the pattern of the data by learning an identity function h(x)≈x in the training process

## How to detect anomalies with autoencoders :

- **Train the autoencoder on data with no anomalies**
- The training process outputs a MSE = $\frac{1}{2n}\Sigma_i||t_i - o_i||^2$ as indicator of the difference between the original data and its low-dimensional reconstruction
- **Apply the auto encoder to the test dataset**
- Each row of the output for the test dataset comes out with a reconstruction error, to be compared with the MSE coming the training
- **Rows with high reconstruction error w.r.t. MSE coming from the training are likely to be outliers**

# REINFORCED LEARNING

## Working principle



- **Technologies :**
  - TensorFlow (Deep Learning by Google)
  - Keras: high-level Deep Learning API for Python
  - Keras RL: RL implementation on Keras

- **Advantages :**
  - Extend ML to decision making, beyond insights
  - Agent can handle very different environmental situation and start extrapolating results
  - Lot of flexibility to favour one behaviour over another (reward function)

- **Disadvantages :**
  - Requires **a lot** of training to converge (i.e., CPU time) !
  - Reward function definition is critical and highly impacts results
  - The entire approach relies on the possibility to accurately model environment, possible agent actions and state changes

B12 OUTSIDE THE BOX

# KEY TAKEAWAYS

- Machine Learning is an increasingly popular approach to solve concrete "real world" issues
- It relies on the presence of valuable datasets, but also on a large variety of (old and new) algorithms and technologies to learn from this data, and the skills and knowledge required to use them efficiently
- A lot has been happening recently, thanks to both significant theoretical advances and improve in computer power (parallelisation), in particular "reinforced learning" is a promising new area.
- But a lot also remains to be done: stay tuned!