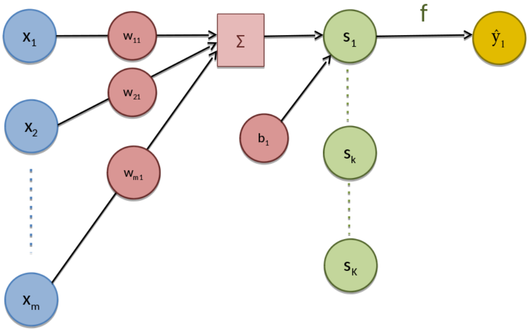
Neural Networks and Deep Learning: Neural Networks

Nicolas Thome

Conservatoire National des Arts et Métiers (Cnam)
Département Informatique

Perceptron and Multi-Class Classification

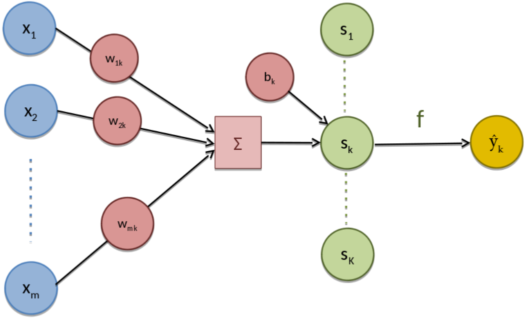
- Formal Neuron: limited to binary classification
- <u>**Multi-Class Classification:**</u> use several output neurons instead of a single one ! $\Rightarrow$ **Perceptron**



- Input $\mathbf{x}$ in $\mathbb{R}^m$
- Output neuron $\hat{y}_1$ is a formal neuron:
  - Linear (affine) mapping: $s_1 = \mathbf{w_1}^\top \mathbf{x} + b_1$
  - Non-linear activation function: f: $\hat{y}_1 = f(s_1)$
- Linear mapping parameters:
  - $\mathbf{w_1} = \{w_{11}, ..., w_{m1}\} \in \mathbb{R}^m$
  - $b_1 \in \mathbb{R}$

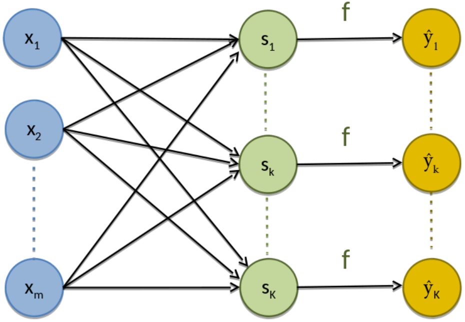
Perceptron and Multi-Class Classification

- Input $\mathbf{x}$ in $\mathbb{R}^m$
- Output neuron $\hat{y}_k$ is a formal neuron:
  - Linear (affine) mapping: $s_k = \mathbf{w_k}^\top \mathbf{x} + b_k$
  - Non-linear activation function: f: $\hat{y}_k = f(s_k)$
- Linear mapping parameters:
  - $\mathbf{w_k} = \{w_{1k}, ..., w_{mk}\} \in \mathbb{R}^m$
  - $b_k \in \mathbb{R}$



Perceptron and Multi-Class Classification

- Input $\mathbf{x}$ in $\mathbb{R}^m$ $(1 \times m)$, output $\hat{y}$: concatenation of K formal neurons
- Linear (affine) mapping ~ matrix multiplication: $\mathbf{s} = \mathbf{x}\mathbf{W} + \mathbf{b}$
  - $\mathbf{W}$ matrix of size $m \times K$ - columns are $\mathbf{w_k}$
  - $\mathbf{b}$: bias vector - size $1 \times K$
- Element-wise non-linear activation: $\hat{\mathbf{y}} = f(\mathbf{s})$

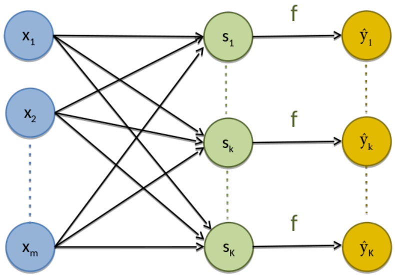


Perceptron and Multi-Class Classification

- **Soft-max Activation**:

$$\hat{y}_k = f(s_k) = \frac{e^{s_k}}{\sum_{k'=1}^{K} e^{s_{k'}}}$$

- **Probabilistic interpretation for multi-class classification**:
  - Each output neuron $\Leftrightarrow$ class
  - $\hat{y}_k \sim P(k/\mathbf{x}, \mathbf{w})$



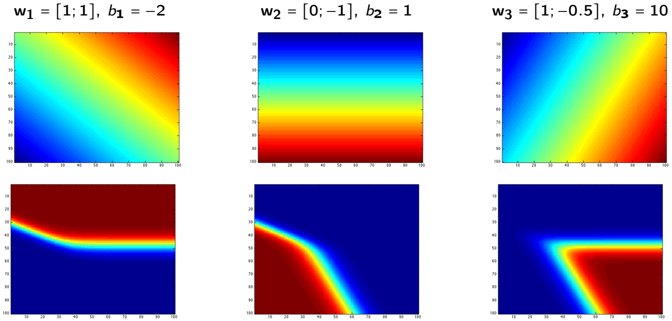
$\Rightarrow$ **Logistic Regression (LR) Model !**

2d Toy Example for Multi-Class Classification

- $\mathbf{x} = \{x_1, x_2\} \in [-5;5] \times [-5;5]$, $\hat{y}$: 3 outputs (classes)

Linear mapping for each class: $\mathbf{s}_k = \mathbf{w_k}^\top \mathbf{x} + b_k$

Soft-max output: $P(k/\mathbf{x}, \mathbf{W})$

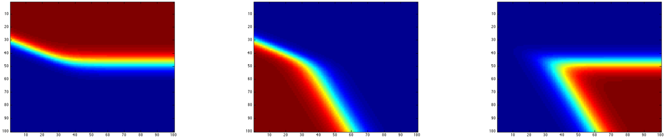


2d Toy Example for Multi-Class Classification

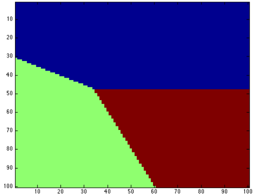
- $\mathbf{x} = \{x_1, x_2\} \in [-5;5] \times [-5;5]$, $\hat{y}$: 3 outputs (classes)

Soft-max output:
$P(k/\mathbf{x}, \mathbf{W})$

$\mathbf{w_1} = [1;1]$, $b_1 = -2$ $\quad$ $\mathbf{w_2} = [0;-1]$, $b_2 = 1$ $\quad$ $\mathbf{w_3} = [1;-0.5]$, $b_3 = 10$

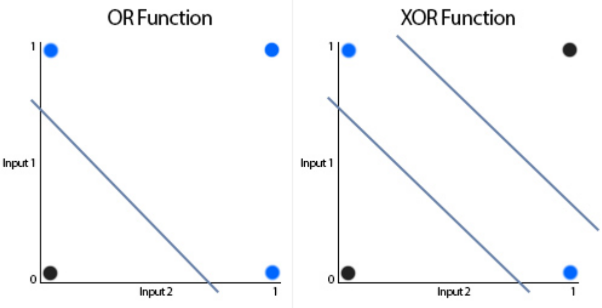


Class Prediction:
$k^* = \arg\max_k P(k/\mathbf{x}, \mathbf{W})$



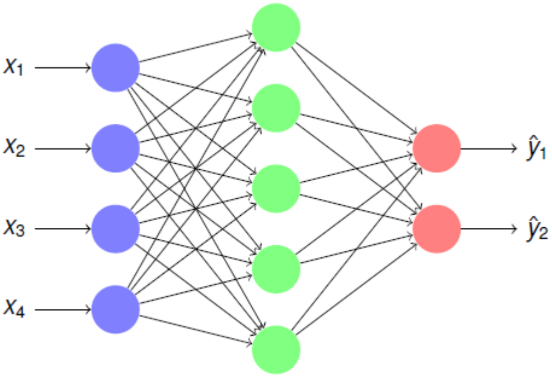
X-OR Problem

- Logistic Regression (LR): NN with 1 input layer & 1 output layer
- LR: limited to linear decision boundaries
- **X-OR**: NOT 1 and 2 OR NOT 2 AND 1
    - **X-OR: Non linear decision function**



Beyond Linear Classification

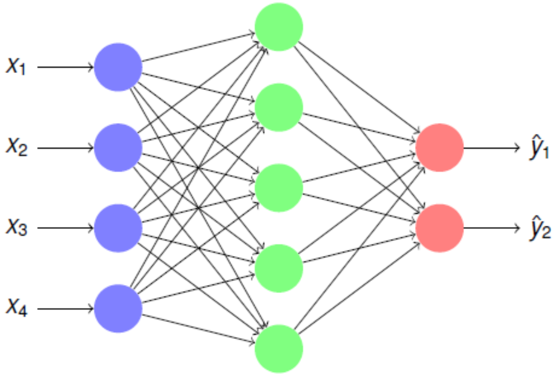
- LR: limited to linear boundaries
- **Solution:** add a layer !



- Input $\mathbf{x}$ in $\mathbb{R}^m$, *e.g.* $m = 4$
- Output $\hat{\mathbf{y}}$ in $\mathbb{R}^K$ (K # classes), *e.g.* $K = 2$
- **Hidden layer $\mathbf{h}$ in $\mathbb{R}^L$**

Multi-Layer Perceptron

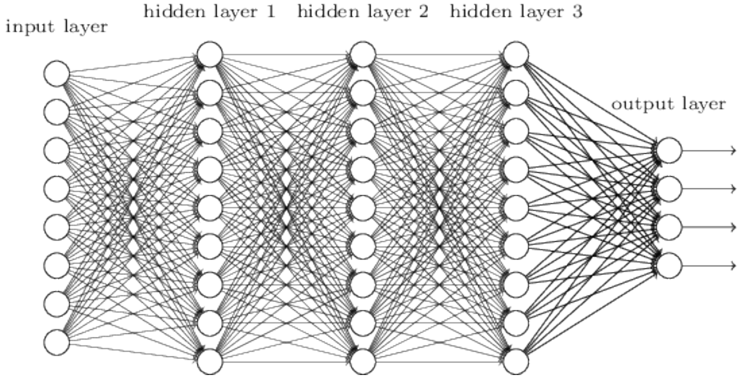
- **Hidden layer h**: **x** projection to a new space $\mathbb{R}^L$
- Neural Net with ≥ 1 hidden layer: **Multi-Layer Perceptron (MLP)**



- **h**: intermediate representations of **x** for classification $\hat{\mathbf{y}}$: $\mathbf{h} = f(\mathbf{xW} + \mathbf{b})$
- **Mapping from x to ŷ: non-linear boundary ! ⇒ activation f crucial!**

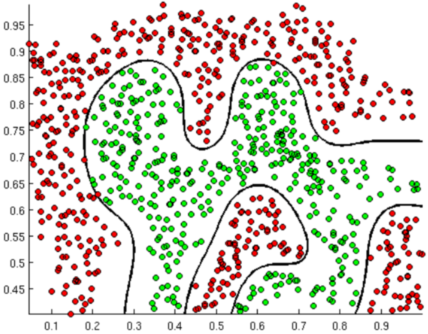
Deep Neural Networks

- Adding more hidden layers: Deep Neural Networks (DNN)
- Each layer $\mathbf{h}^l$ projects layer $\mathbf{h}^{l-1}$ into a new space
- Gradually learning intermediate representations useful for the task



Conclusion

- Deep Neural Networks: applicable to classification problems with non-linear decision boundaries



- Visualize prediction from fixed model parameters
- Reverse problem: **Supervised Learning** $\Rightarrow$ **following!**