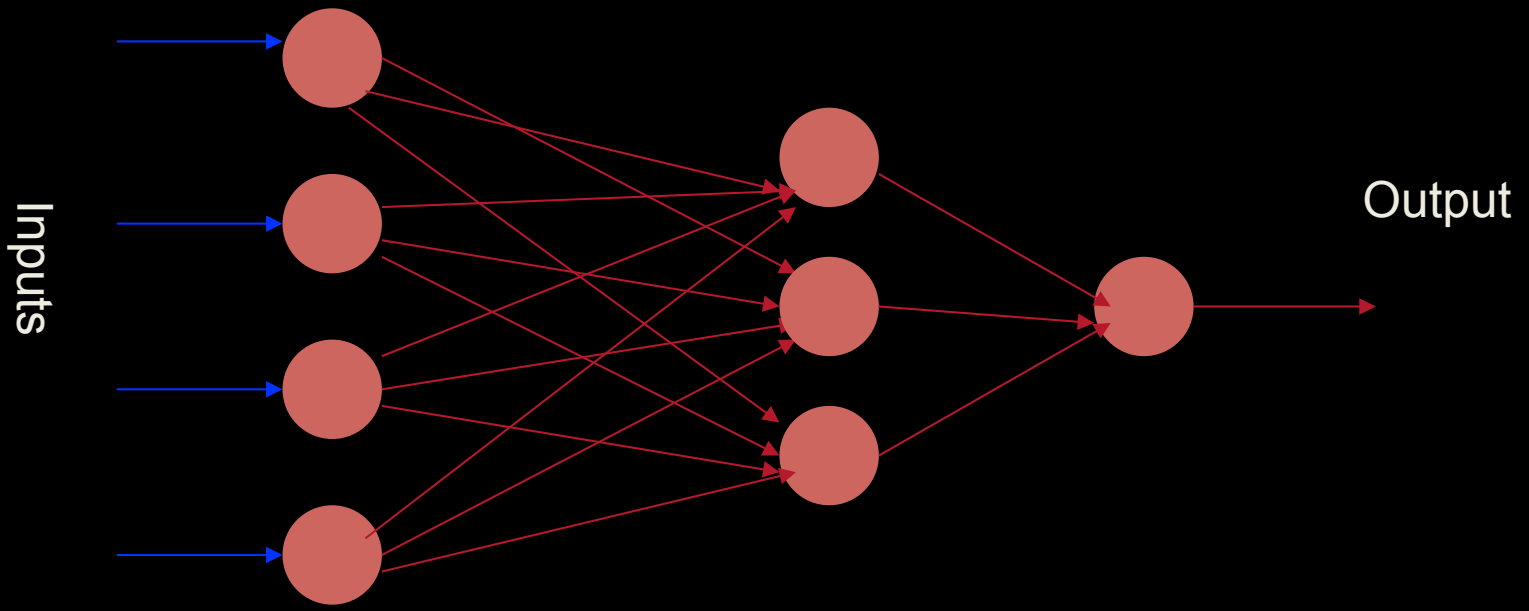
Apprentissage automatique
Projet
Notes sur la partie Neural Networks

Elisa Fromont

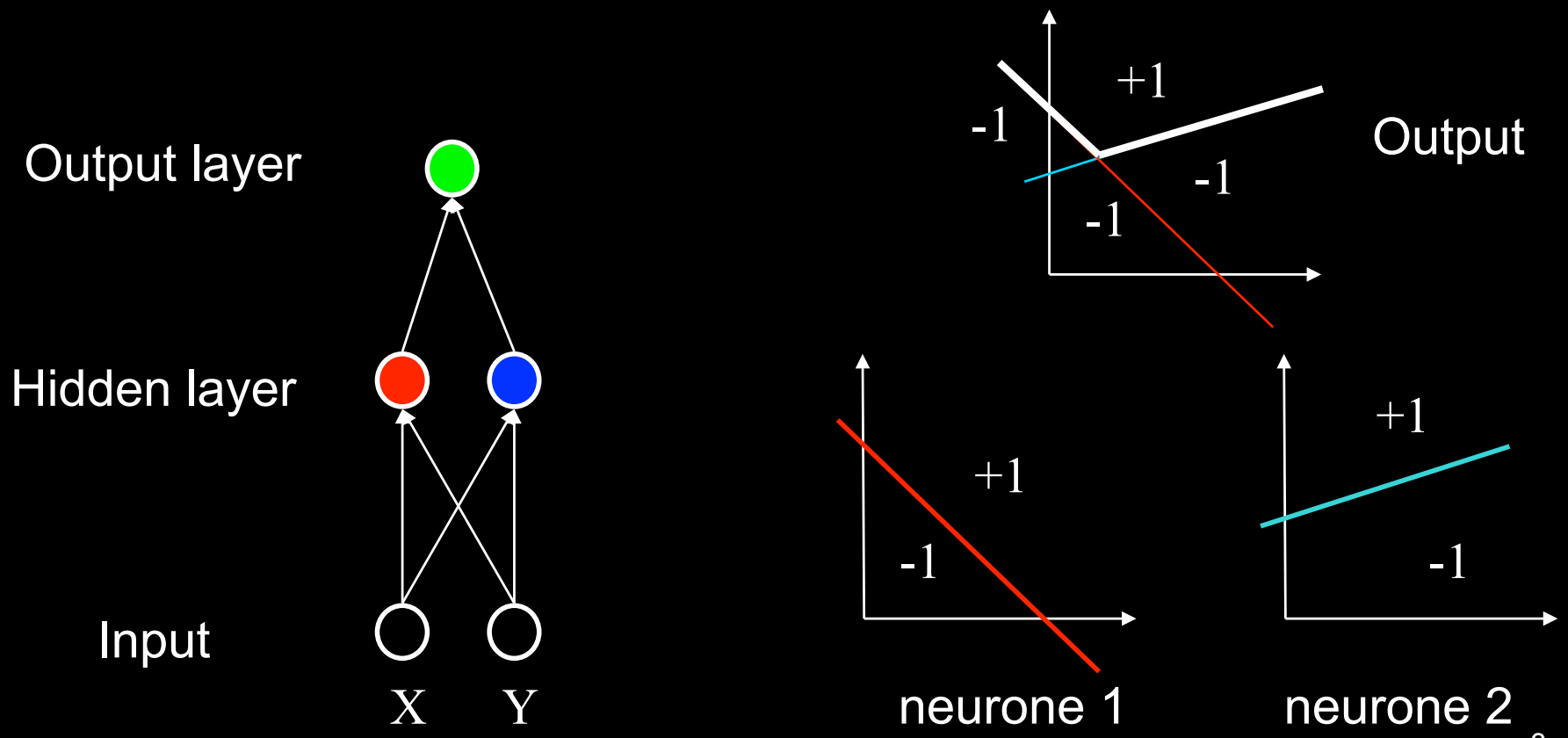
Neural networks (NN)



An artificial neural network is composed of many artificial neurons that are linked together according to a specific network architecture. The objective of the neural network is to transform the inputs into meaningful outputs by changing the weights between the connections.

Multiple layers network

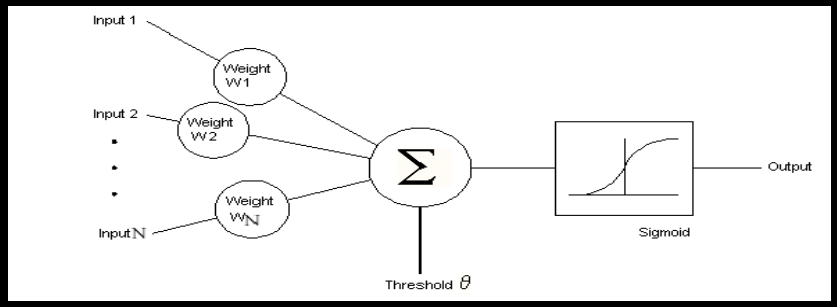
Can approximate any function $f : \Re^d \rightarrow \Re^C$



Artificial neurons

y is the neuron's output, x is the vector of inputs, and w is the vector of synaptic weights.

$$y = f(x, w)$$



Nowadays, the activation function is often chosen to be the logistic sigmoid or the hyperbolic tangent (tanh).

$$y = \frac{1}{1 + e^{-w^T x - a}}$$

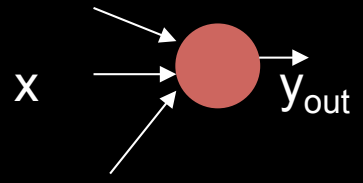
sigmoidal neuron

MLP neural networks

MLP = multi-layer perceptron

Perceptron:

$$y_{out} = w^T x$$



MLP neural network:

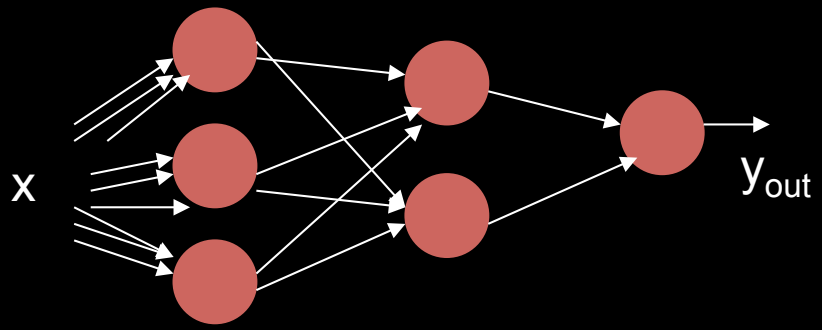
$$y_k^1 = \frac{1}{1+e^{-w^{1kT}x - a_k^1}}, k = 1,2,3$$

$$y^1 = (y_1^1, y_2^1, y_3^1)^T$$

$$y_k^2 = \frac{1}{1+e^{-w^{2kT}y^1 - a_k^2}}, k = 1,2$$

$$y^2 = (y_1^2, y_2^2)^T$$

$$y_{out} = \sum_{k=1}^{2} w_k^3 y_k^2 = w^{3T} y^2$$



Les problèmes que vous ne pourrez pas éviter…

- Choisir une architecture pour votre réseau
- Décider des Meta paramètres (même si vous utilisez une implémentation existante)
- Représenter votre problème (reconnaissance de caractères manuscrits) pour que votre réseau puisse apprendre à classifier (choix des entrées, des sorties, méthodes d'apprentissage)

Quelle architecture pour le projet ?

- Couche d'entrée: nb de neurones dépend de la représentation choisie (nb de pixels ou taille de l'histogramme par exemple)
- Nb de couches cachées + nb neurones dans chaque couches: utilisation ou pas de convolutions (1 ou 2 couches cachées semblent suffisantes pour limiter le nombre de paramètres à apprendre)
- Couche de sortie: nb de neurone = nb de classes dans le pb (0,1,2,…,9)

Reconnaissance de caractères, quelles entrées ?

- Plusieurs possibilités:
- Un exemple d'apprentissage = (M, l) M = vecteur (matrice) codant la valeur (binaire) des pixels d'entrée du caneva (taille fixe). L = classe de l'exemple (vecteur binaire à C dimension où C est le nombre de classe)
- Utilisation des codes de Freeman en entrée (trouver une représentation de taille fixe, par exemple, histogramme des codes)

Quels paramètres ?

- Comment initialiser les poids ?

(aléatoire entre -1 et 1 ou « Derrick Nguyen and Bernard Widrow. Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights. In Neural Networks, 1990., 1990 IJCNN International Joint Conference on, pages 21–26. IEEE, 1990. »

- Quel est le « learning rate » ? (0.001, 0.01, 0.1,0.2)
- Combien de neurones dans la couche cachée et combien de couches ? (80, 100, 200, …)

(tout cela peut être tester par cross-validation ou sur un ensemble de validation)

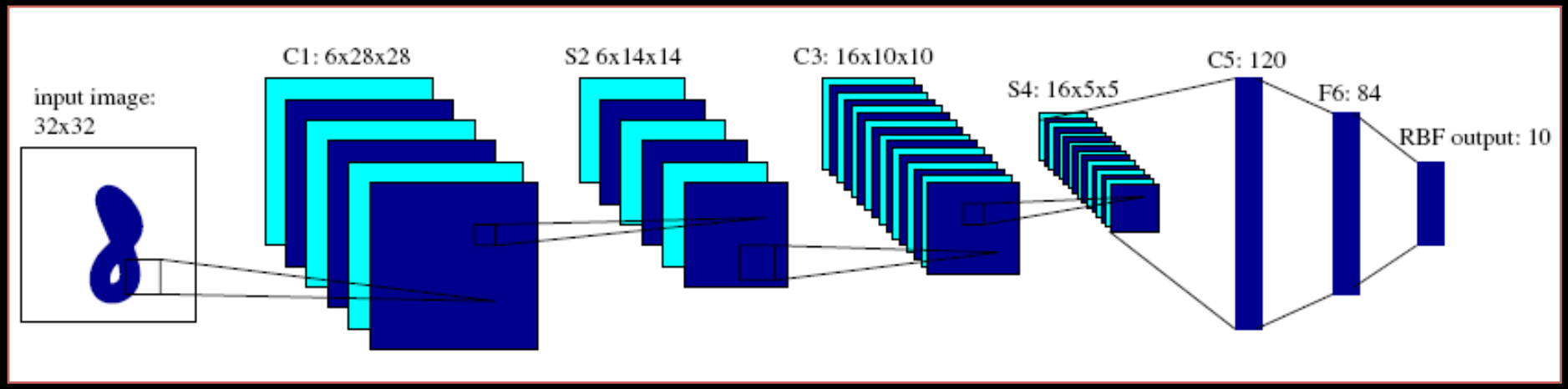
Où trouver des implémentations?

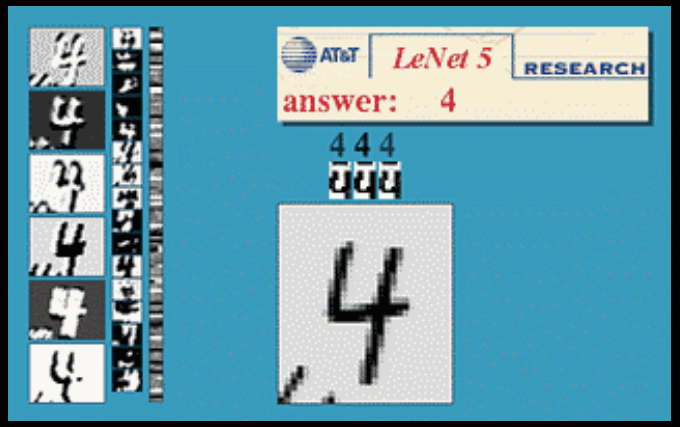
- Weka (implémentation Java): http://weka.sourceforge.net/doc.dev/weka/classifiers/functions/MultilayerPerceptron.html
- Matlab implementation
- Torch: http://torch.ch/
  - scientific computing framework with wide support for machine learning algorithms. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
- Theano:

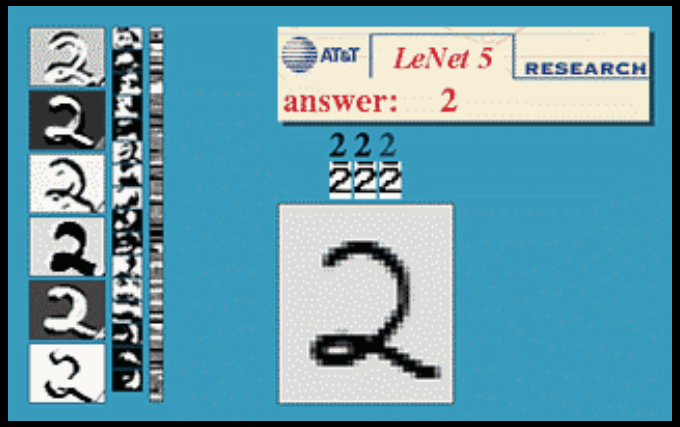
http://deeplearning.net/tutorial/lenet.html

Convolutional Neural Networks (CNN)

- A special kind of multi-layer neural networks.
- Implicitly extract relevant features.
- A feed-forward network that can extract topological properties from an image.
- Like almost every other neural networks CNNs are trained with a version of the back-propagation algorithm.
- Particularly suitable for signal processing applications (for example computer vision, speech recognition)
  - ex: digit recognition, image classification…







Try it out: http://cs.stanford.edu/people/karpathy/convnetjs/demo/mnist.html