

# *Les réseaux de neurones*

Par:  
Dr. KOUAMO Stéphane

Institut National Polytechnique Houphouet Boigny - Master Big Data

November 4, 2024

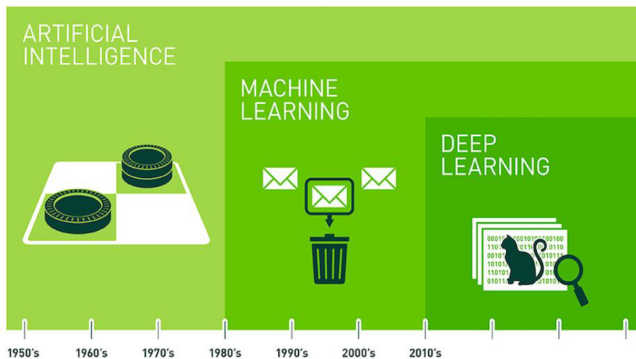


# Outline

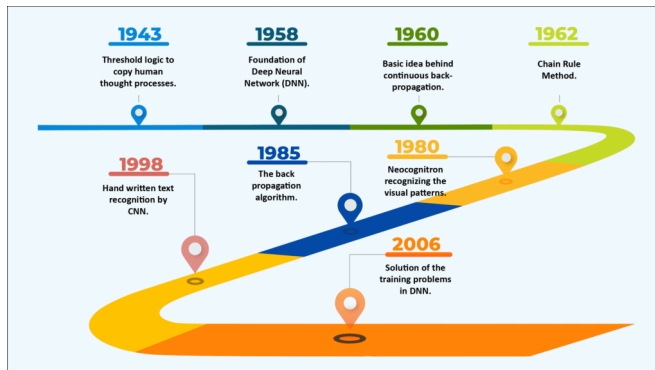
- ① Introduction
  - Historique, Définition, Utilités et limites
- ② Structure d'un Réseau de Neurone (RN)
- ③ Etapes de conception d'un RN
- ④ Différents types de RN
- ⑤ Deep learning
  - Définition et contexte
  - Pourquoi le deep learning?
  - Quelques algorithmes de deep learning
- ⑥ Présentation des projets + Test de connaissance



# Evolution



# Evolution



# Historique

## La préhistoire

- Warren Sturgis, McCulloch et Walter Pitts ont mené les premiers travaux sur les réseaux de neurones (RN) à la suite de leur article: " *What the frog's tells the frog's brain*" [10];
- apparition du concept de neurone formel en 1943, des poids synaptiques et du principe d'apprentissage.
- Seulement pas d'indication sur la méthode pour adapter les poids synaptiques. [Détection des minuties : exemple.](#)



# Historique

## Les premiers succès

- Franck Rosenblatt en 1957 propose le perceptron, le tout premier système artificiel capable d'apprendre par expérience y compris lorsque l'instructeur commet quelques erreurs [11];
- Cependant le perceptron montre quelque limites théorique, notamment l'impossibilité de traiter les problèmes non linéaires.
- Entre 1967 et 1982 les travaux sur les RN sont mis en berne.



# Historique

## Le renouveau

- En 1984, le système de rétro propagation du gradient de l'erreur est au coeur des débats.
- Minsky introduit le concept sur le perceptron multicouche, un système capable de résoudre les problèmes de non linéarité [8]



# Historique

## Le renouveau (suite)

- En 1986, Rumelhart peaufine les travaux sur le perceptron et propose une combinaison avec la rétro propagation du gradient de l'erreur [12].
- En 1992, Vapnik et al. ont fait bénéficier aux RN la technique de la régularisation statistique qui permet d'anticiper sur les problèmes liées au sur apprentissage.
- Le sur-apprentissage est une difficulté à laquelle doivent faire face tous les systèmes d'apprentissage par l'exemple (que ce soit par optimisation directe : régression linéaire ou optimisation itérative descente du gradient).





# Historique

## Le sommet de la pyramide

- En 1998, naissance d'un nouveau concept introduit par Yan Le Cun et al.
- En 2006, Fei-Fei Li commence à travailler sur la base de données visuelle ImageNet (introduite en 2009).
- A partir de 2010, Hinton et al.[3] poursuivent les travaux entamés par l'équipe de Yan LeCun et propose le Deep Neural Networks.
- En 2014, l'équipe de Yan LeCun propose le fameux algorithme pour Google, le GoogleNet un RN profond avec 22 couches, puis en 2015, le Deep Dream [2].



# Historique

## Le sommet de la pyramide

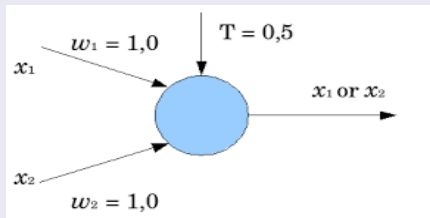
- En 2016, Kaiming He et al. proposent un RN profond avec 152 couches pour Microsoft/Facebook [6].
- 2018 et au-delà : Les architectures comme les transformateurs (introduites par le modèle BERT) révolutionnent le traitement du langage naturel, conduisant à des avancées significatives dans la compréhension et la génération de texte.



# Définition

## Neurone formel

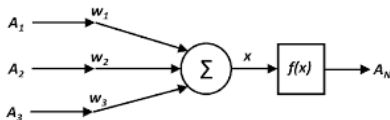
Entité unique pouvant être relié à d'autres entités, qui porte une information nécessaire pour l'activation d'une fonction produisant un résultat



# Définition

## Poids de connexion

Liens directs entre les neurones formels et les différentes couches du réseau de neurones.



$A_i$  : activité du neurone  $i$  (0 ou 1)

$w_i$  : poids du neurone d'entrée  $i$

$\Sigma$  : calcule la somme  $x$  des produits  $A_i \cdot w_i$

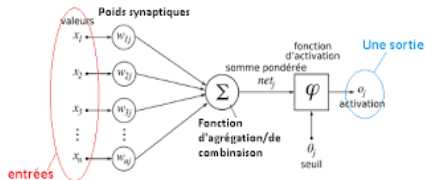
$f(x)$  : fonction d'activation (ou de transfert)



# Définition

## Réseau de Neurones Artificiel (RNA) ou Perceptron Multi-Couche

Un réseau de neurones (ou Artificial Neural Network en anglais) est un modèle de calcul dont la conception est très schématiquement inspirée du fonctionnement des vrais neurones (humains ou non)



# Définition

## Caractéristiques

- Au moins une couche avec une fonction d'activation non linéaire.
- Les neurones de la couche  $i$  servent d'entrées à ceux la couche  $i + 1$ .
- Neurones d'entrée, Neurone de sortie, Neurones cachés.
- Généralement : tous les neurones d'une couche sont connectés à tous les neurones des couches précédentes et suivantes.
- Toutes les connections  $i \rightarrow j$  sont pondérées par les poids  $w_{ij}$ .



# Définition

## Apprentissage profond

- Ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.



# Définition

## Apprentissage profond

- Ensemble de méthodes d'apprentissage automatique tentant de modéliser avec un haut niveau d'abstraction des données grâce à des architectures articulées de différentes transformations non linéaires.
- Il s'exerce sur un réseau de neurones profond, c-a-d un RN avec plusieurs couches cachées dont les sorties des couches de niveau inférieur constituent les entrées des couches de niveau supérieur et sont étroitement liées entre elles.





# Utilités

## Pourquoi recourir à des réseaux de neurones ?

Les réseaux de neurones s'avèrent plus performants que les techniques de régressions pour des tâches de Machine Learning.

- capacité à s'adapter aux données traitées (apprentissage) ;
- possibilité d'effectuer les calculs en parallèles.



# Domaines d'application

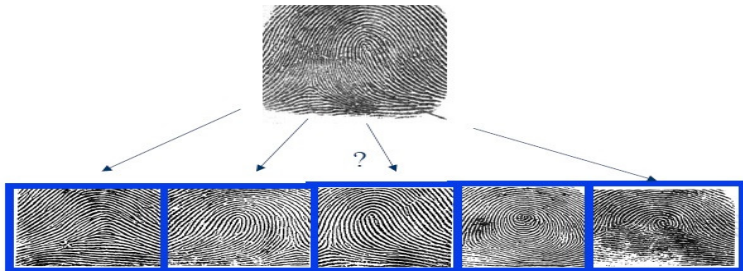
Ils sont souvent caractérisés par une relation *entrée-sortie* de la donnée d'information. Les RN sont utilisés entre autres pour :

- La reconnaissance d'image;
- Les classifications de textes ou d'images;
- Identification d'objets;
- Prédiction de données;
- Filtrage d'un ensemble de données;
- Le déplacement automatisé de robots mobiles autonomes;
- L'estimation boursière (apprentissage de la valeur d'une entreprise, prédiction sur la périodicité des cours boursiers, etc.);



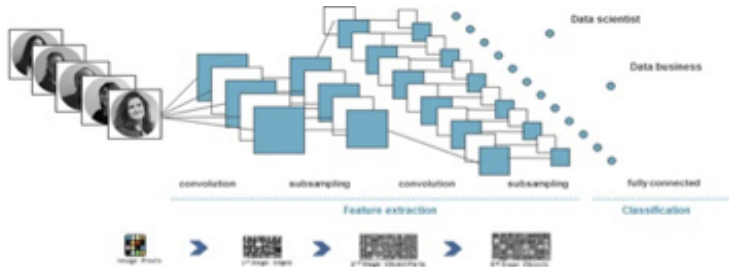
# Domaines d'application : exemple

- La classification



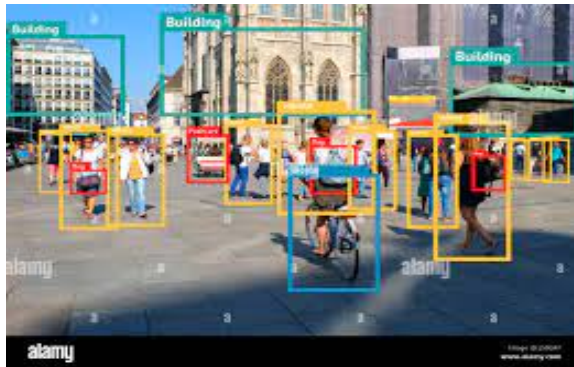
# Domaines d'application : exemple

- La reconnaissance d'image



# Domaines d'application : exemple

- La détection d'objet



# Limites

- Les RN ont besoin de servir d'exemples pour leur apprentissage. Ces cas sont en relation avec la complexité du problème à résoudre.;
- Impossibilité de réaliser la fonction du *XOR* pour le perceptron;
- Les RN ne fournissent pas les explications concernant leurs résultats ce qui limite l'analyse des phénomènes existants;
- Le choix du nombre de couches. Il n'existe pas de règle pour faire ce choix.



# Fonction de combinaison

Les  $x_i$  représentent les neurones d'entrée, les  $w_{ij}$  les poids de connexion et  $o_j$  les neurones en sortie. La fonction de combinaison est utilisée pour calculer la somme des entrées de la manière suivante :

$$o_j = \left( \sum_{i=1}^N w_{ij} x_i \right)$$



# Fonction d'activation

Soit un RN avec  $N$  cellules en entrée notées  $C_i; 1 \leq i \leq N$  et  $N$  poids de connexion  $W_{ij}$ . Pour obtenir la valeur  $Y$  en sortie on utilise une fonction d'activation qui peut être :

- la fonction logistique.

$$Y = \frac{2}{(1 + e^{(-2x)}) - 1}$$





# Fonction d'activation

- la fonction hyperbolique.

$$Y = \frac{2}{(1 + e^{(-2x)+1})}$$

- la fonction de Gauss.

$$Y = e^{(\frac{-x^2}{2})}$$

- la fonction à seuil.

$$Y = 0 \text{ si } X \leq 0 \text{ et } Y = 1 \text{ si } X > 0$$



## Une idée à partir d'exemples

Souvent, on ne sait pas définir le bon algorithme, mais on a des exemples (données). Imaginons que vous êtes un éleveur bovin.



La première chose à faire est de recueillir des données sur le poids (en kg) et l'âge (en mois) de plusieurs vaches.



Age (mois)	Poids (kg)
18	450
6	200
30	700
12	350
24	600
96	1200
30	700
36	800
72	1100
48	900
84	1150
108	1250
60	1000
54	950
42	?



On a vite l'idée de tracer une courbe de la forme  $y = ax + b$ .

Le but est donc de trouver les valeurs de  $a$  et  $b$ .



On a vite l'idée de tracer une courbe de la forme  $y = ax + b$ .

Le but est donc de trouver les valeurs de  $a$  et  $b$ .

$$a = \frac{y_2 - y_1}{x_2 - x_1}$$

$$b = y_1 - ax_1 = y_2 - ax_2 = \frac{x_2y_1 - x_1y_2}{x_2 - x_1}$$

Si  $a = 0$  alors la fonction est constante Si  $b = 0$  alors la fonction est linéaire.

Une fois que les valeurs de  $a$  et  $b$  sont corrects, on devine le poids de la vache dont l'âge est de 42*mois*, soit environ 850*kg*



# Exercice

## Réaliser la fonction XOR

Le but est de trouver les valeurs de  $x$  à partir de celles de  $a$  et  $b$ .

$$X = A \oplus B$$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0



## Choix et préparation de l'échantillon

- L'élaboration d'un RN commence toujours par le choix et la préparation des échantillons de données.
- Cette étape est cruciale et va aider le concepteur à déterminer le type de réseau le plus approprié pour résoudre son problème.
- La façon dont se présente l'échantillon conditionne le type de réseau (le nombre de cellules d'entrée, le nombre de cellules de sortie et la façon dont il faudra mener l'apprentissage, les tests et la validation).

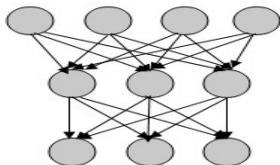
Une fois que l'échantillon est correct, on peut élaborer la structure du RN.



## Elaboration de la structure du réseau

La structure du RN dépend étroitement du type d'échantillon. Le type de réseau peut être : un perceptron standard, un réseau de Hopfield [4], un réseau de Kohonen [1], un ARTMAP, etc. Pour le cas du Perceptron, il faut choisir le nombre de neurone de la couche cachée.

**Propagation en avant.**

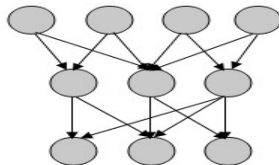


Réseau multicouche

Couche d'entrée

Couche cachée

Couche de sortie



Réseau à connexions locales

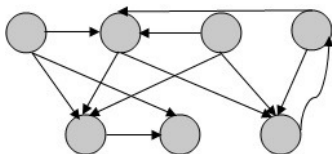




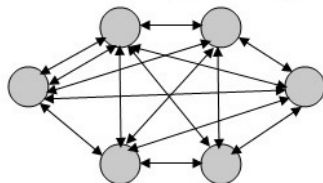
## Elaboration de la structure du réseau

**Modèle récurrent** Propagation des activations se fait de deux façons :

- synchrone : c'est à dire que toutes les unités sont mises à jour simultanément;
- asynchrone : les unités sont mises à jour de façon séquentielle.



Modèle récurrent asynchrone



Modèle récurrent synchrone

# Apprentissage

Cette notion n'est pas modélisable dans le cadre de la logique déductive. On procède à partir des connaissances déjà établies dont on tire une expérience dérivée. Deux réalités dans cette notion :

- Mémorisation : Assimiler sous une forme dense des exemples nombreux.
- Généralisation : Etre capable grâce aux exemples appris de traiter des exemples distincts encore non rencontrés mais similaires.

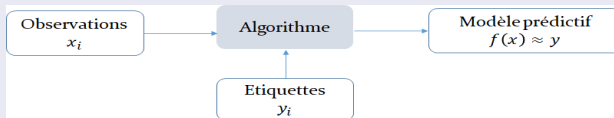
Deux phénomènes opposés : c'est la *généralisation* qui est l'objet de notre attention.



# Apprentissage

## Apprentissage supervisé

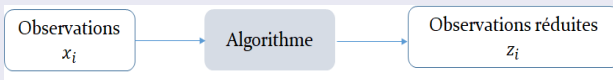
ici, on force le réseau à converger vers un état final précis (connu à l'avance), en même temps qu'on lui présente un motif.



# Apprentissage

## Apprentissage non supervisé

A l'inverse, lors d'un apprentissage non supervisé, le réseau est laissé libre de converger vers n'importe quel état final lorsqu'on lui présente un motif.



# Apprentissage

## Apprentissage par renforcement

l'apprentissage par renforcement consiste, pour un agent autonome (robot, etc.), à apprendre les actions à entreprendre, à partir d'expériences, de façon à optimiser une récompense quantitative au cours du temps. L'agent est plongé au sein d'un environnement, et prend ses décisions en fonction de son état courant. En retour, l'environnement procure à l'agent une récompense, qui peut être positive ou négative.



# Apprentissage

Plusieurs types d'algorithmes d'entraînement qui consiste à modifier les poids synaptiques en fonction d'un jeu de données :

- Sur apprentissage : Il arrive qu'à force de faire apprendre un RN toujours sur le même échantillon, celui-ci devienne inapte à reconnaître autre chose que les éléments présents dans l'échantillon.
- L'élagage (Pruning en anglais) : consiste à supprimer des connexions (ou synapses), des entrées ou des neurones du réseau une fois l'apprentissage terminé.
- Rétro propagation: consiste à rétro propager l'erreur commise par un neurone à ses synapses et aux neurones qui y sont reliés. Pour les RN on parle de rétro propagation du gradient de l'erreur (corrige les poids synaptiques qui contribuent le plus à engendrer l'erreur).



# Apprentissage

## Principe de l'algorithme de rétro propagation

- (1) Initialiser les poids de connexions par des valeurs aléatoires (généralement petites, dans l'ordre de 0 et 1);
- (2) Choisir un pattern et l'appliquer à la couche d'entrée;
- (3) Propager le signal à travers le réseau :

$$v_i^m = g(h_i^m) = g\left(\sum_j w_{ij} \times v_j^{m-1}\right)$$

jusqu'au calcul de  $v^N$ ;

- (4) Calculer les deltas sorties;

$$\delta_i^N = g'(h_i^N) \times (y_i^m - v_i^m);$$



# Apprentissage

## Principe de l'algorithme de rétro propagation

(5) Calculer les deltas sur les autres couches;

$$\delta_i^{m-1} = g'(h_i^{m-1}) \sum w_{ji}^m \delta_j^m;$$

$$m = N, N - 1, \dots 2$$

(6) Modifier les poids de connexion en utilisant;

$$\Delta w_{ij}^m = \eta \delta_i^m v_j^{m-1};$$

$$w_{ij}^m = w_{ij}^{m-1} + \Delta w_{ij}^m;$$

$$\eta = \frac{1}{\eta};$$

(7) Aller à l'étape 2 si encore des patterns;

(8) Itération suivante.





# Validation et tests

## Validation

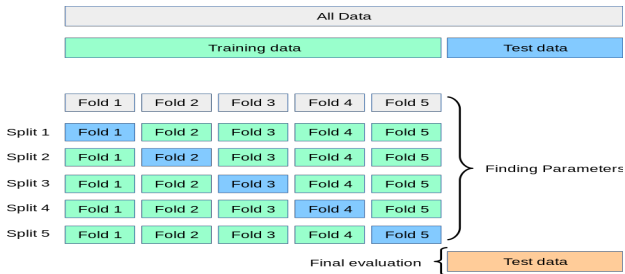
- Une fois le réseau calculé, il faut toujours procéder à des tests afin de vérifier que le réseau réagit correctement.
- Pour les tests dans le cas général, une partie des données est simplement écartée de l'échantillon d'apprentissage et conservée pour les tests hors échantillon : **Cross validation**.

On peut par exemple utiliser 60% de l'échantillon pour l'apprentissage, 20% pour la validation et 20% pour les tests. Le but final étant de faire que toutes les données participent au moins une fois à l'apprentissage et au test.



# Validation et tests

## Cross validation ou Validation croisé .



# Critères de performance

## Matrice de confusion

On appelle **vrais positifs (true positives)** les exemples positifs correctement classifiés ; **aux positifs (false positives)** f les exemples négatifs étiquetés positifs par le modèle ; et réciproquement pour les **vrais négatifs (true negatives)** et les **faux négatifs (false negatives)** .

		Classe réelle	Classe réelle
		0	1
Classe prédite	0	True Negative	False Negative
Classe prédite	1	False Positive	True Positive



## Critères de performance

### Rappel

On appelle **rappel (recall en anglais)**, ou sensibilité (sensitivity en anglais), le taux de vrais positifs, c'est-à-dire la proportion d'exemples positifs correctement identifiés comme tels.

$$Rappel = \frac{TP}{TP + FN} \text{ et la Precision} = \frac{TP}{TP + FP}$$

### Score

$$Score = \frac{TP + TN}{TP + TN + FP + FN}$$



## Que retenir?

Les RN permettent de résoudre des problèmes précis.

- on utilise en ensemble de données de départ,
- on trouve le lien qui existe entre ces données,
- on fait une généralisation,
- on prédit/devine une nouvelle valeur.

C'est en cela que consiste le machine learning. Seulement on peut avoir des exemples beaucoup plus complexes avec plus de paramètres à traiter. Par exemple : l'algorithme du flux de facebook qui présente les pages dans un certain ordre en fonction de : qui a posté? Quand? Nature de la page, Sujet, Réactions des autres, ... On essaie de prédire si cela vous intéresse ou pas? L'équation de la droite  $y = ax + b$  sera dépassée d'où l'utilisation des RN.



## Que retenir?

- Pour un neurone  $x$  avec  $i$ , il existe des poids  $p_i$  et une activation  $y$ .

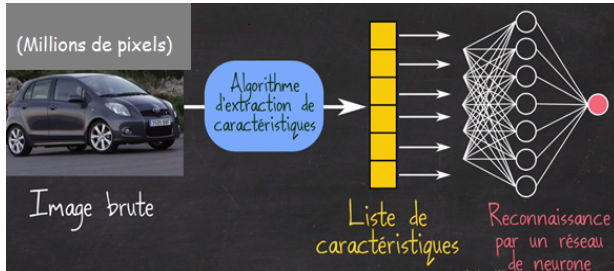
$$\text{Si } \sum x_i p_i \succ s, \quad y = 1 \text{ sinon } y = 0$$

- Pour le RN nous avons plusieurs entrées, des sorties et une phase d'apprentissage. Une phase de prédiction.



## Que retenir?

- Pour le **traitement d'image** il existe une étape intermédiaire : la détection et l'extraction des caractéristiques, avant la phase d'apprentissage.



- Le traitement de l'image consistera à traiter les vecteurs caractéristiques de l'image.

# Différents types de RN

## RN à apprentissage non supervisé

Les (RN a apprentissage non supervisé) les plus connus sont :

- Réseau de kohonen: Réseau a apprentissage non supervise avec retro propagation (Quantification Vectorielle, Self Organisation Map, Learning Vector Quantization (utilise la recherche du plus proche voisin)).
- Discrete Hopfield.
- Continuos Hopfield
- Apprentissage compétitif.





# Les RN pour les pré traitements

## Transformation

- RNC (Réseau de neurones cellulaires) : Un neurone cellulaire est une unité de calcul dont l'état a un instant donné dépend des sorties précédentes des cellules de son voisinage et de leurs entrées.
- La relation entre état et entrées s'établit sous la forme d'une équation différentielle:

$$\frac{de_i}{dt} = -\frac{1}{\tau}e_i + \sum_{vj} a_{ij}i_j + \sum_{vj} b_{ij}s_j + e_0$$

Où  $e_i$  désigne l'état du neurone  $i$ ,  $i_j$  et  $s_j$  désignent l'entrée et la sortie du neurone  $j$  et  $V_i$  désigne le voisinage du neurone  $i$ . Les paramètres  $a_{ij}$  et  $b_{ij}$  sont les poids (matrices A et B) et  $e_0$  est une constante. La sortie d'un neurone est déterminée par une fonction croissante de l'état.



# Les RN pour les pré traitements

## Prédiction

- La définition d'un modèle prédictif  $\tilde{y} = f(x)$  est un problème d'approximation de fonction [9].
- Une des applications des réseaux de neurones, en particulier les perceptrons multicouches (MLP) et l'apprentissage par retro propagation permettent de construire des modèles prédictifs non linéaires.

## Segmentation

- La classification est également une application des RN.
- RNC pour séparer les pixels d'une image selon qu'ils appartiennent a une zone texturée ou non.



## Les RN pour la quantification scalaire et vectorielle

- La quantification scalaire uniforme d'une valeur  $x$  ne demande qu'une seule opération (Cependant, si les valeurs à quantifiées ne sont pas réparties uniformément, il faut comparer chacune d'elles à  $x$ ).
- La quantification vectorielle consiste à approximer un ensemble de valeur successive d'une image. Le tout traite en parallèle.



## Les RN pour la quantification scalaire et vectorielle (suite)

Le codage d'un vecteur  $x$  de dimension  $k$  consiste à lui associer un entier  $i$  compris entre 1 et  $N$ . La distorsion introduite est généralement mesurée par l'*erreur quadratique moyenne*:

$$d(x, y) = \sum_{j=1}^k (x_j - y_{ij})^2$$

Afin de minimiser l'erreur quadratique moyenne, le codage suit la règle du plus proche voisin, on choisit  $i$  de tel que :

$$\forall j, d(x, y_i) \leq d(x, y_j)$$



## Les RN pour la quantification scalaire et vectorielle (suite et fin)

Algorithme de l'apprentissage compétitif simple :

- (1) Initialiser le dictionnaire;
- (2) Tirer un vecteur d'apprentissage  $x$ ;
- (3) Rechercher le neurone  $i$  vainqueur de la compétition tel que :

$$d(x, w_i) \leq d(x, w_j) \quad \forall j$$

- (4) Mobiliser son vecteur poids:  $y_i \leftarrow y_i + \epsilon (x, -y_i)$
  - (5) Re boucler en 2 tant que le critère d'arrêt n'est pas atteint.
- Avec  $d$  étant une mesure de distance, et  $w_i$  le meilleur représentant du vecteur  $x$  pris en entrée. Le paramètre est un coefficient d'apprentissage, il est compris entre 0 et 1, et décroît au cours de l'apprentissage. Le critère d'arrêt est généralement d'atteindre un nombre donné d'itérations fixé à l'avance [5].



## Les RN basés sur l'apprentissage de Hebb

Structure générale d'un réseau de neurone de Hebb. Règle d'apprentissage de Hebb : si deux neurones sont très actifs au même moment (qui est illustre par des grandes valeurs de leurs sorties et de leurs entrées), les poids de connexion entre ces deux neurones augmenteront :

$$[h] = [W]^T \cdot [X]$$



## Les RN basés sur l'apprentissage de Hebb

La règle d'apprentissage peut être décrite de la manière suivante :

$$W_i(t+1) = \frac{W(t) + \alpha h_i(t).X(t)}{||W_i(t) + \alpha h_i(t).X(t)||}$$

Où  $W_i(t+1) = W_{i1}, W_{i1} \dots W_{iN}$  est le  $i^{ieme}$  nouveau vecteur de couple de poids au temps  $(t+1)$ ; avec  $1 \leq i \leq M$  et  $M$  est le nombre de neurones en sortie.  $\alpha$  est le taux d'apprentissage.  $h_i(t)$  la  $i^{ieme}$  valeur en sortie.  $X(t)$  le vecteur en entrée correspondant a chaque bloc de l'image.  $||.||$  est la norme euclidienne utilisée pour normaliser les poids mis a jour et équilibrer l'apprentissage.



# Programmation des Réseaux de neurones avec Python

## Différentes étapes à suivre

- Télécharger les données (*load\_data()*).
- Définir le modèle *Keras* (*Keras.model.Sequential()*).
- Compiler le modèle *Keras* (*Keras.compile()*).
- Entraîner le modèle *Keras* (*Keras.fit()*).
- Evaluer le modèle *Keras* (*Keras.evaluate()*).
- Faire la prédiction.

## Prérequis

- Disposer de Python 2 ou 3.
- Se rassurer d'avoir SciPy installer.
- Se rassurer d'avoir Keras installer et configuré.










(L'apprentissage profond).

-  R. Cappelli, A. Lumini, D. Maio and D. Maltoni, "Synthetic fingerprint-Database Generation", proceeding of the 16<sup>th</sup> International Conference on Pattern Recognition (ICPR 2002), Québec City, Vol.3, pp 744 – 747, August 2002.
-  Y. Le Cun, Y. Bengio and G. Hinton. " *Deep learning*". Macmillan Publishers Limited. Nature, Vol.521, pp. 436 – 444. May 2015. DOI : 10.1038/nature14539.
-  Hinton G. and al. " *Deep Neural Networks for Acoustic Modelling in Speech Recognition*". IEEE Signal Processing Magazine. Digital Object Identifier 10 : 1109/MSP : 2012 : 2205597. November 2012.
-  J.J. Hopfield. "Neural networks and physical systems with emergent collective computational abilities". Proceedings of the National Academy of Sciences. (1982). pp. 2554 – 2558



-  J. Jiang. "Image compression with neural networks, A survey". Signal Processing : Image Communication, (1998).
-  Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. " *Deep Residual Learning for Image Recognition*". Computer Vision and Pattern Recognition. *arXiv* : 1512 : 03385[cs : CV], 2016.
-  Y.Minami, S.Tamura, HSawai, K.Shikano, " *Output Smoothing for TDNN Using Information on Input Vectors Neighborhood in Input Layer, Hidden Layer* ", Proc. ASJ, 1 : 3 : 18, pp.35 – 36 (Mar. 1990) (in Japanese).
-  M. Minsky and S. Papert. *Perceptrons*. MIT Press, Cambridge MA, 1969.
-  O. Moreira-Tamayo et J. Pineda de Gyvez. " *Preprocessing Operators for Image Compression Using Cellular Neural*



*Networks"*. Dans IEEE International Conference on Neural Networks, juin 1996. pp.1500 – 1505.



Lettvin, J. Y., McCulloch, Maturana, H.R., W. Sturgis. et W. H. Pitts. " *What the frog's eye tells the frog's brain*". Proceedings of the I.R.E. (1959). Vol.47, N° : 11, pp. 1940 – 1951.



F. Rosenblatt. " *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain*". Psychological Review. Vol. 65, N°. 6, pp. 386 – 408. 1958.



D.E. Rumelhart, G.E. Hinton and R.J. Williams. " *Learning Internal representations by error propagation*". In Parallel Distributed Processing, Cambridge, MA : MIT Press, 1 : 318 – 362, 1986.

