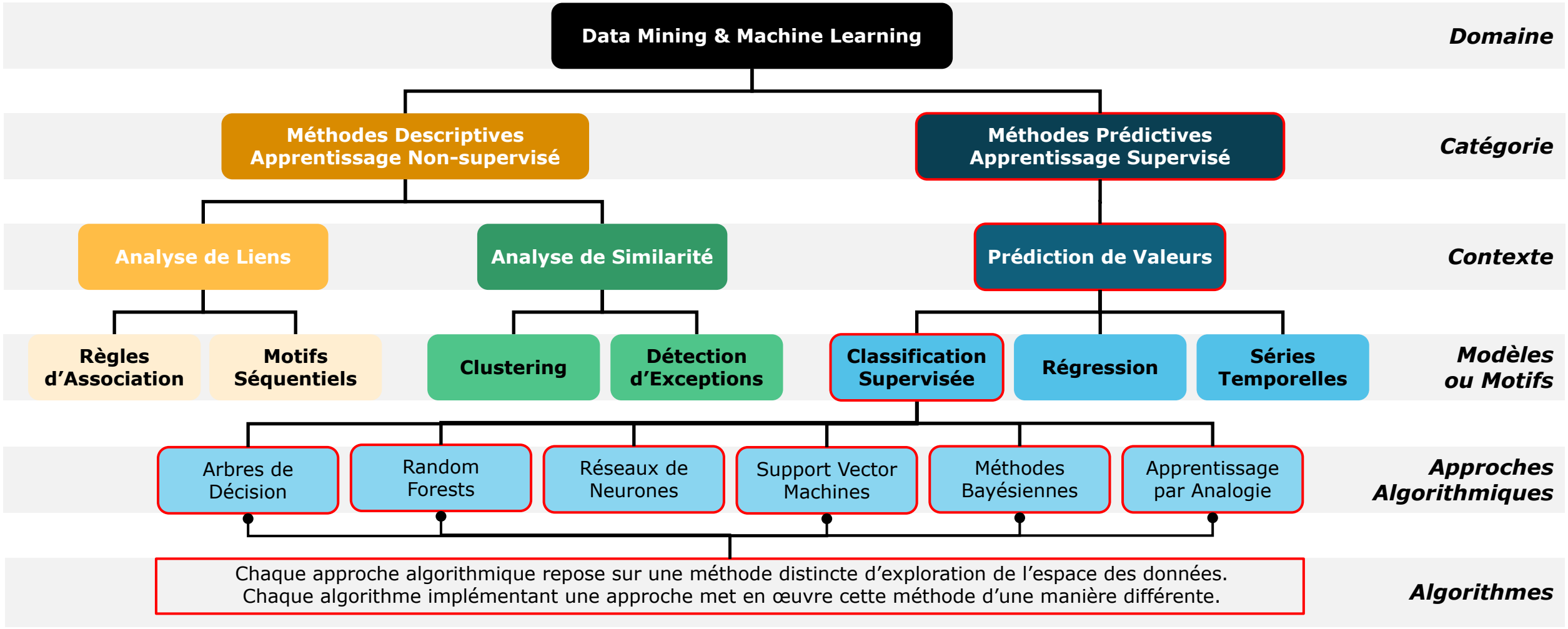
Module 2 : Réseaux de Neurones

Nicolas PASQUIER
Université Côte d'Azur
EUR DS4H (Digital Systems for Humans)
Laboratoire I3S (UMR-7271 UCA/CNRS)
http://www.i3s.unice.fr/~pasquier

Co-financé par :

Use cases réalisés par les masters :

Méthodes d'Extraction de Modèles de Connaissances

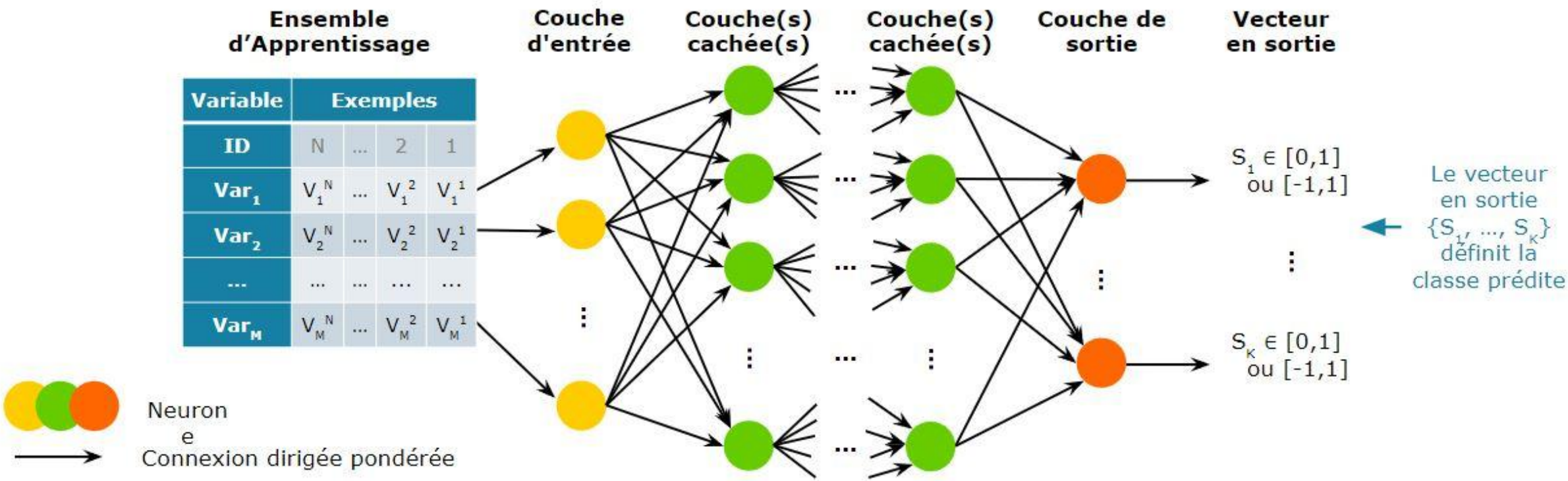


Réseaux de Neurones Artificiels : Principe de l'Approche

- « Artificial Neural Networks » ou « ANN » (W. McCulloch & W. Pitts, 1950).
- Objectif : reproduire le fonctionnement des neurones biologiques.
- Réseaux de nœuds appelés **neurones artificiels** interconnectés.
- Les neurones sont organisés en couches, chacune correspondant à un rôle, dans le réseau :
  - **Couche d'entrée :**
    - Neurones qui reçoivent chacun les valeurs d'une variable de l'ensemble d'apprentissage.
  - **Couche(s) cachée(s) :**
    - Neurones de calculs qui ont un rôle de « mémoire ».
    - Différents cas : zéro (classifieur linéaire), une ou plusieurs couches cachées .
  - **Couche de sortie:**
    - Neurones qui génèrent le résultat final (prédictions des classes).

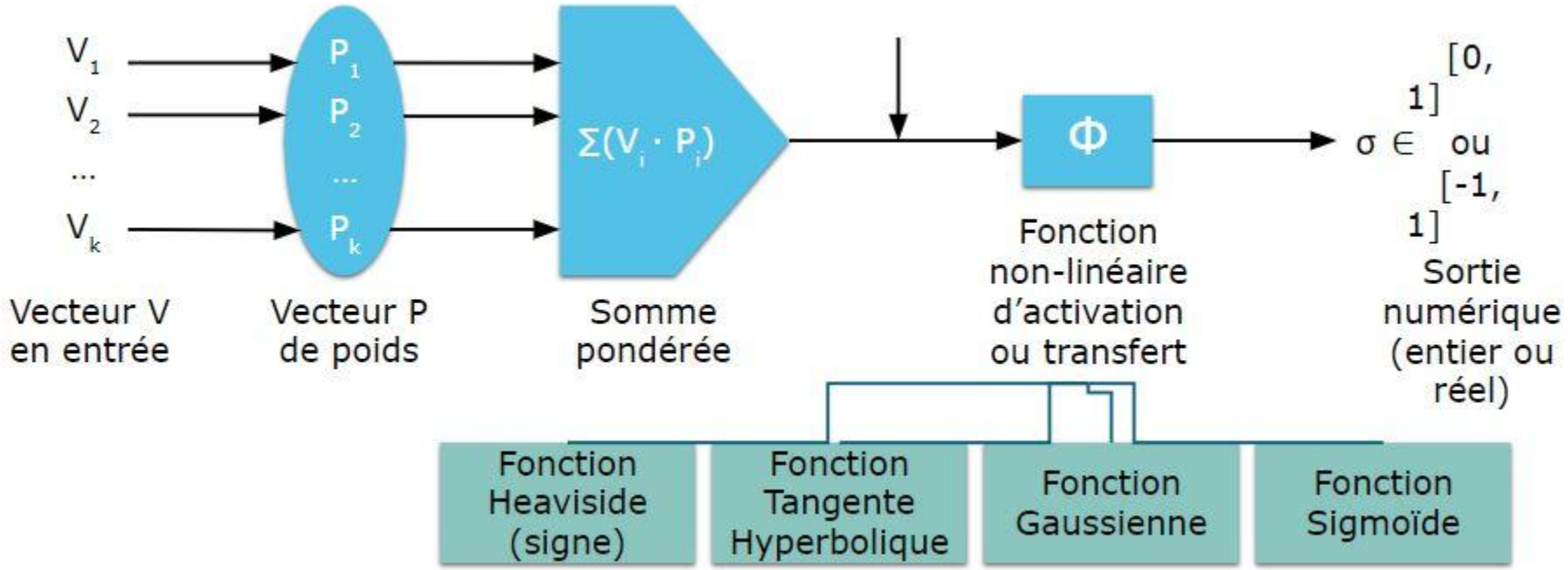
Réseaux de Neurones de Type Perceptron

- L'information (données calculées en fonction des valeurs des variables) circule dans le réseaux de la couche d'entrée vers la couche de sortie;



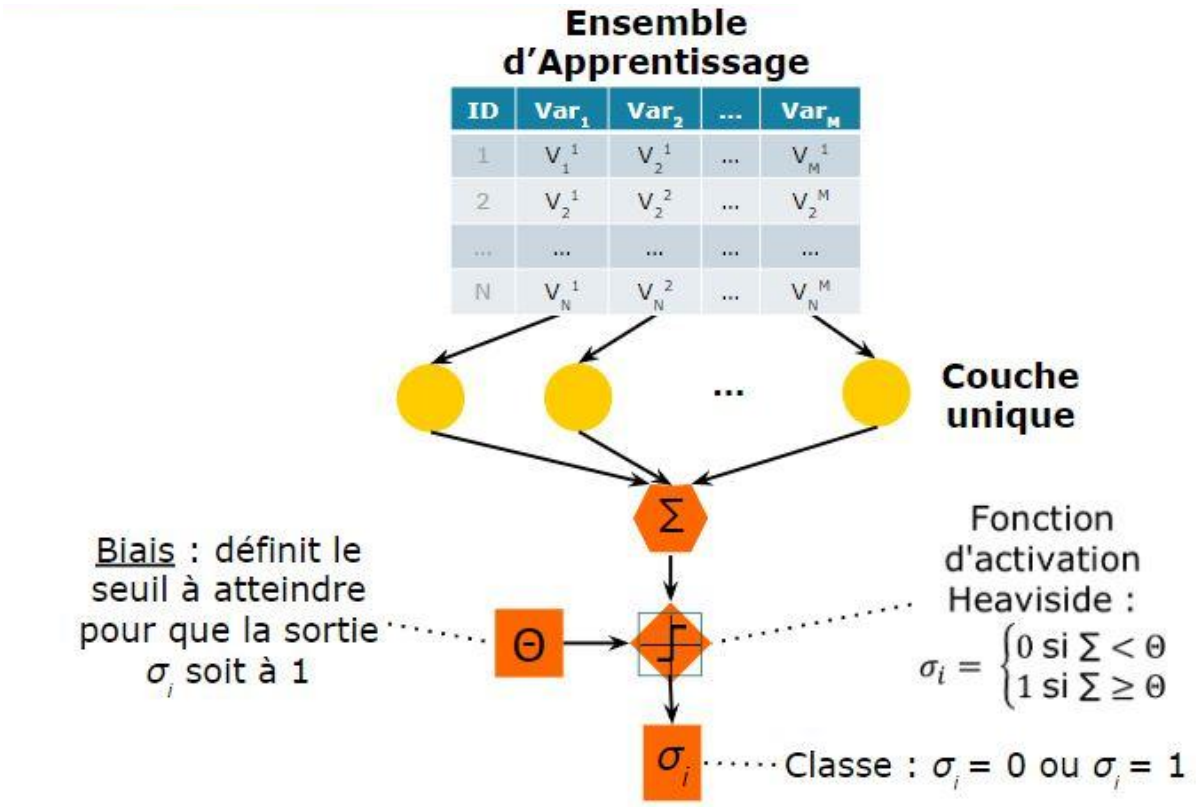
Structure des Neurones Artificiels

- Chaque neurone reçoit en entrée de une (neurone de la couche d'entrée) à N (neurones des couches cachées et de sortie) valeurs numériques V_i.



Perceptrons Mono-Couche

- « The Perceptron – A Perceiving and Recognizing Automaton » (Rosenblatt, 1957).
- Type de réseau de neurones artificiels le plus simple : une unique couche de neurones assure les fonctions d'entrée, calcul et sortie.
- Classifieur linéaire : calcule la décision par combinaison linéaire des valeurs des variables.
- Efficace pour les données de grande dimension.
- Incapable de traiter des problèmes non linéaires (M. L. Minsky & S. Papert, 1969).



Perceptrons Mono-Couche

- « Back-Propagation Learning » (D. E. Rumelhart & Y. LeCun, 1986).
- **Couche d'entrée :**
  - Taille (nombre de neurones) définie automatiquement par l'algorithme en fonction de l'ensemble d'apprentissage.
  - Nombre et types des variables détermine la taille : un neurone par valeur (variables discrètes) ou par intervalles de valeurs (variables continues) le plus souvent.
- **Couche(s) cachée(s) :**
  - Le nombre de couches (1 ou plus) et la taille de chacune (nombre de neurones) sont des paramètres définis par l'utilisateur.
- **Couche de sortie :**
  - Taille (nombre de neurones) définie automatiquement par l'algorithme en fonction de l'application (principalement le nombre de classes à distinguer).

Perceptrons Mono-Couche

- Capables de traiter des problèmes non-linéaires de différents types grâce aux différentes topologies et fonctions d'activation ou transfert.
- Apprentissage : détermination des poids P_{nm} optimaux pour chaque connexion inter-neuronale entre les neurones *n* et *m* de deux couches adjacentes.
- Problème d'optimisation basé sur la minimisation de l'erreur quadratique.
- Erreur quadratique locale *E(i)* pour l'exemple (instance) i : différence entre la sortie attendue y_i et la sortie calculée par le réseau σ_i
- $$E(i) = \frac{1}{2}(\gamma_i - \sigma_i)^2$$
- Erreur quadratique globale *E(EA)* pour l'ensemble d'apprentissage *EA* : cumul des différences pour les N exemples.
- $$E(EA) = \frac{1}{2N}\sum_{i=1}^{i=N}(\gamma_i - \sigma_i)^2$$

Apprentissage d’un Réseau de Neurones

- **Rétropropagation** du gradient de l’erreur
  - Les poids des connexions inter-neuronales sont modifiés selon l'importance de leur contribution à l’erreur.
  - Les connexions inter-neuronales sont traitées de la dernière couche vers la première couche.
- Entrée de l’algorithme de rétropropagation :
  - Configuration du réseau de neurones (e.g. nombre et taille des couches).
  - Ensemble d’apprentissage *EA*.
- Paramètres de l’algorithme de rétropropagation :
  - Condition d’arrêt de l’apprentissage : nombre d’itérations (i.e. parcours de l’ensemble d’apprentissage), temps de calcul ou seuil de mesure d’évaluation à atteindre (e.g. erreur globale).
  - Taux d’apprentissage $\alpha \in [0.0, 1.0]$: définit la vitesse d’évolution du réseau (facteur de la m-à-j des poids).

Apprentissage des Poids : Algorithme de Rétropropagation

répéter
  pour chaque exemple $e \in$ EA faire
    présenter e au perceptron
    si sortie calculée $\sigma \neq$ sortie attendue γ alors
      pour chaque neurone $n \in$ couche de sortie faire
        $\delta_n \leftarrow \sigma_n (1 - \sigma_n) (\gamma_n - \sigma_n)$ // Dérivée
      finpour
      pour chaque couche adjacente précédente C
        pour chaque neurone $n \in$ couche C faire
          $\delta_n \leftarrow \sigma_n (1 - \sigma_n) \Sigma_{k \in \text{Succ}(n)} \delta_k P_{nk}$
        finpour
      finpour
    finsi
    pour chaque poids P_{nm} faire
      $P_{nm} \leftarrow P_{nm} + \alpha \delta_n e_{nm}$
    finpour
  finpour
jusqu'à condition d'arrêt

δ_n : facteur de modification des poids pour n
σ_n : sortie du neurone n
γ_n : valeur attendue pour le neurone n
Succ(n) : neurones successeurs de n
P_{nm} : poids de la connexion de n vers m
e_{nm} : valeur transmise entre n et m pour e
α : taux d'apprentissage

Rétropropagation du Gradient de l'Erreur

Deux modes d'application possibles :

- Mode **on-line** : la mise-à-jour a lieu pour chaque exemple d'apprentissage.
  - Avantage : tend plus rapidement vers la solution optimale.
  - Inconvénient : un trop grand nombre d'itérations peut conduire à un sur-apprentissage (*overfitting*).
  - Le réseau est alors très performant pour les exemples de l'EA, mais ne parvient que peu, ou pas, à généraliser pour d'autres exemples.
- Mode **batch** : la mise-à-jour des poids a lieu après introduction de tous les exemples de l'ensemble d'apprentissage.
  - Avantage : diminue les risques de sur-apprentissage.
  - Inconvénient : corrige les poids sur la globalité des exemples et l'impact de chaque exemple sur la correction apportée est faible.
  - Nécessite beaucoup de temps (itérations) avant de tendre vers une bonne solution.

Problème du Sur-Apprentissage

- Classement d'un exemple : l'exemple transite par le réseau et le vecteur en sortie définit la prédiction de classe.
- « **Sur-ajustement** » (*over-fitting*) peut être causé par :
  - Un dimensionnement inadéquat des couches du réseau (taille et nombre des couches trop important).
  - Un apprentissage trop poussé (trop d'itérations).
- Caractérisation :
  - Le réseaux décrit intégralement chaque exemple d'apprentissage, c-à-d tient compte systématiquement de toutes les valeurs des variables de l'exemple appris.
- Conséquence :
  - Un nouvel exemple ne pourra être classé correctement que s'il est intégralement identique à un exemple d'apprentissage.
  - Mauvaises propriétés de généralisation.

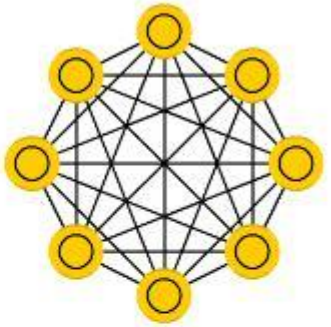
Réseaux de Neurones Récurrents

- Perceptrons : réseaux de neurones à propagation avant
  - « *Feed-forward Neural Networks* ».
  - Pas de cycle : l'information n'est transmise que vers l'avant dans le réseau, des nœuds d'entrée vers les nœuds de sortie.
- Réseaux de neurones **récurrents**, ou à **propagation arrière** :
  - « *Feedback Neural Networks* » ou « *Recurrent Neural Networks* ».
  - Cycles dirigés : l'information est transmise vers l'avant (couche suivante) mais également aux neurones de la couche courante et éventuellement aussi à ceux des couches précédentes.
  - Information générée à l'itération t+1 tient compte de celle générée à l'itération t : renforcer « l'effet mémoire ».
  - Variantes dans les topologies, connectivités et types de neurones : Hopfield networks, Boltzmann machines, Bidirectional Associative Memory, Gated Recurrent Unit, Long Short-Term Memory, etc.

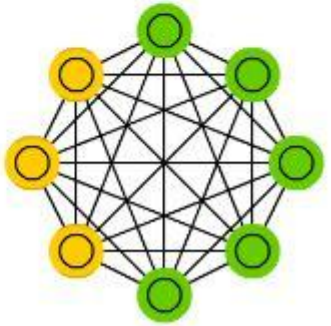
Réseaux de Neurones Intégralement Connectés

- Réseaux constitué de L neurones à états binaires ({-1,1} ou {0,1}) qui sont tous interconnectés.
- La sortie du réseau est la combinaison d'états finaux (état stable) des L neurones.
- Peut mémoriser environ 0,15 × L *patterns*, i.e. modèles à reconnaître.
- Ces réseaux ont démontré leur efficacité pour les problèmes de reconnaissances de formes : écriture manuscrite (mémorisation des caractères), identification d'objets dans des images (médicale, satellite, etc.).

Réseau de Hopfield



Réseau de Boltzmann



Topologies des Perceptrons et Réseaux Récurrents

Types de Neurones

Neurone d'entrée

Neurone caché

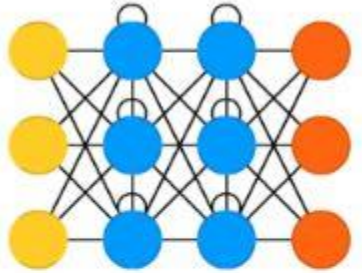
Neurone de sortie

Neurone récurrent

Perceptron monocouche

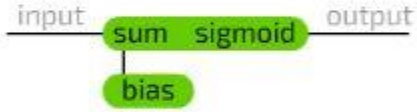
Perceptron multicouches

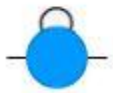
Réseau de neurones multicouches récurrent



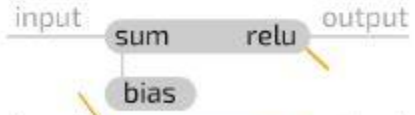
Structure interne des neurones

Neurone à propagation avant





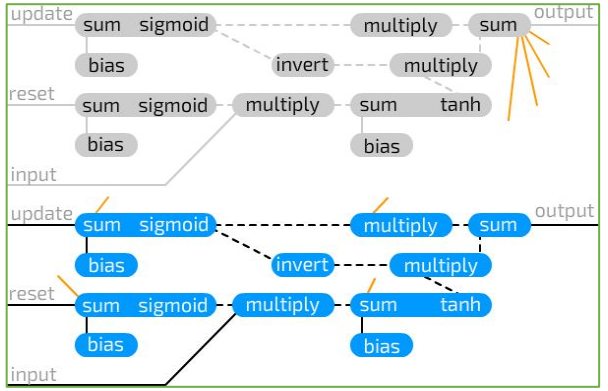
Neurone récurrent (itération précédente)



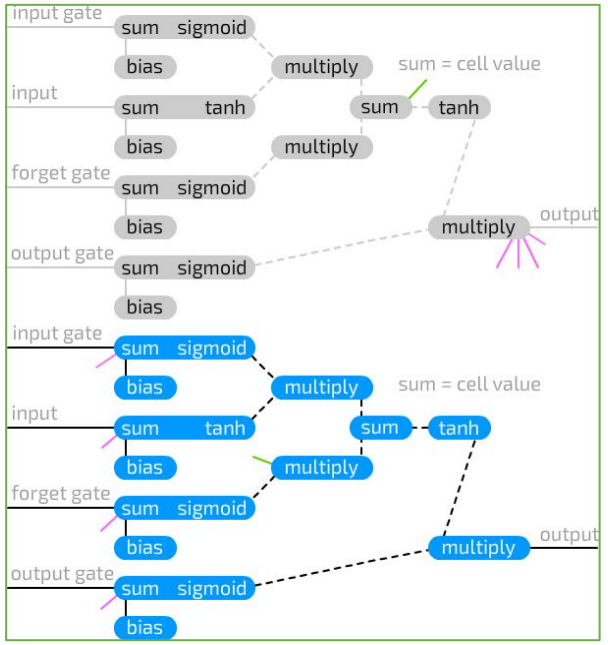
Neurone récurrent (itération actuelle)

Neurones Récurrents à Portes : Mémorisation Renforcée

GRU (Gated Recurrent Unit)
Combat la perte d'information grâce aux portes « reset » et « update ».
Reset : détermine quelle proportion de l'entrée est ajoutée à la valeur générée.
Update : détermine quelle proportion de la précédente valeur générée est ajoutée.

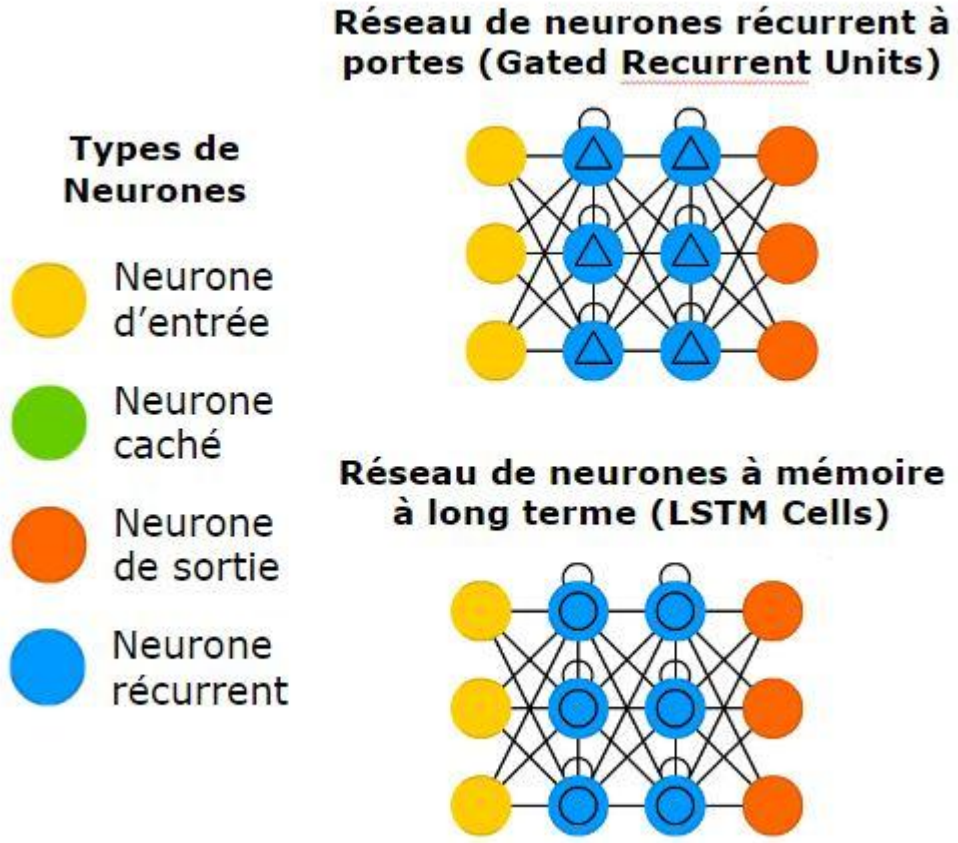


LSTM (Long Short-Term Memory Cell)
Combat la perte d'information grâce aux portes « input », « forget » et « output ».
Input : détermine quelle proportion de l'entrée est ajoutée à la valeur générée.
Forget : détermine quelle proportion de la précédente valeur générée est ajoutée.
Output : détermine quelle proportion de la valeur générée est transmise au reste du réseau.



Réseaux de Neurones Profonds ou Réseaux Convolutifs

- « *Deep Learning* » : mémorisation à long terme, problèmes séquentiels.
- Notion de « profondeur » : liée au nombre de couches cachées (plusieurs dizaines usuellement) et à leur taille (plusieurs milliers de neurones).
- Premières couches cachées (**convolutionnaires**) : extraient des caractéristiques simples (e.g. contours).
- Couches suivantes : combinent pour former des concepts de plus en plus complexes (e.g. assemblages de contours en motifs, de motifs en parties d'objets, de parties d'objets en objets, etc.).



Paramétrage de Réseaux de Neurones

- Normalisation des variables numériques continues :
  - Une magnitude importante de valeurs peut entraîner la saturation de plusieurs neurones et bloquer l'apprentissage.
- Topologie : nombre et taille des couches cachées.
- Taux d'apprentissage : détermine l'amplitude de l'apprentissage :
  - Trop petit : lenteur de la convergence (temps de calcul longs, problème des minimas locaux).
  - Trop grand : oscillations (mauvaise convergence).
  - Le plus souvent autour de 0.05 à 0.15 (définition empirique).
- Durée d'apprentissage :
  - Nombre d'itérations (usuellement plusieurs centaines).
  - Durée d'apprentissage (usuellement en minutes).
  - Seuil de mesure d'erreur à atteindre (e.g. erreur quadratique cumulée).

Réseaux de Neurones : Propriétés

- Avantages :
  - Bonne tolérance aux données bruitées (exceptions, erreurs de classes, etc.).
  - Efficaces dans le traitement des ensembles de données numériques continues de grande dimension.

- Inconvénients :
  - Effet « boîte noire » : impossible pour un humain d'interpréter l'information représentée par les poids des connexions et la topologie du réseau.
  - Temps d'apprentissage peut être long.
  - Difficulté de configuration (choix de topologie et taux d'apprentissage).
  - Risque de sur-apprentissage.

Références et Bibliographie

- Principales Librairies R
  - nnet : réseaux de neurones à couche cachée unique (inclus dans R base).
  - neuralnet : réseaux à couches cachées multiples offrant de nombreuses possibilités de paramétrisation (nombre et taille des couches, fonctions d'activation ou transfert, rétropropagation récurrente ou non, etc.).
  - deepnet : réseaux de neurones intégralement récurrents et profonds (réseaux de Hopfield et Boltzmann, réseaux profonds avec auto-encodeurs successifs, deep belief neural networks, etc.).
  - RSNNS : implémentations du Stuttgart Neural Network Simulator (SNNS) fournissant une grande variété de topologies de réseaux de neurones (perceptrons, réseaux récurrents, etc.).
  - tensorflow : interface vers la bibliothèque libre TensorFlow permettant la parallélisation des calculs sur CPU et GPU.
  - keras : interface vers la bibliothèque libre Keras proposant des réseaux profonds et récurrents, et fonctionne de façon transparente sur CPU et GPU.