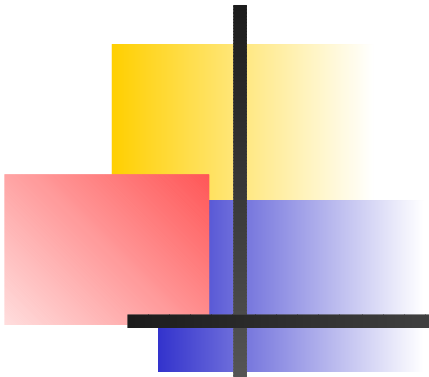


Réseaux de neurones

Liva Ralaivola

liva@lif.univ-mrs.fr

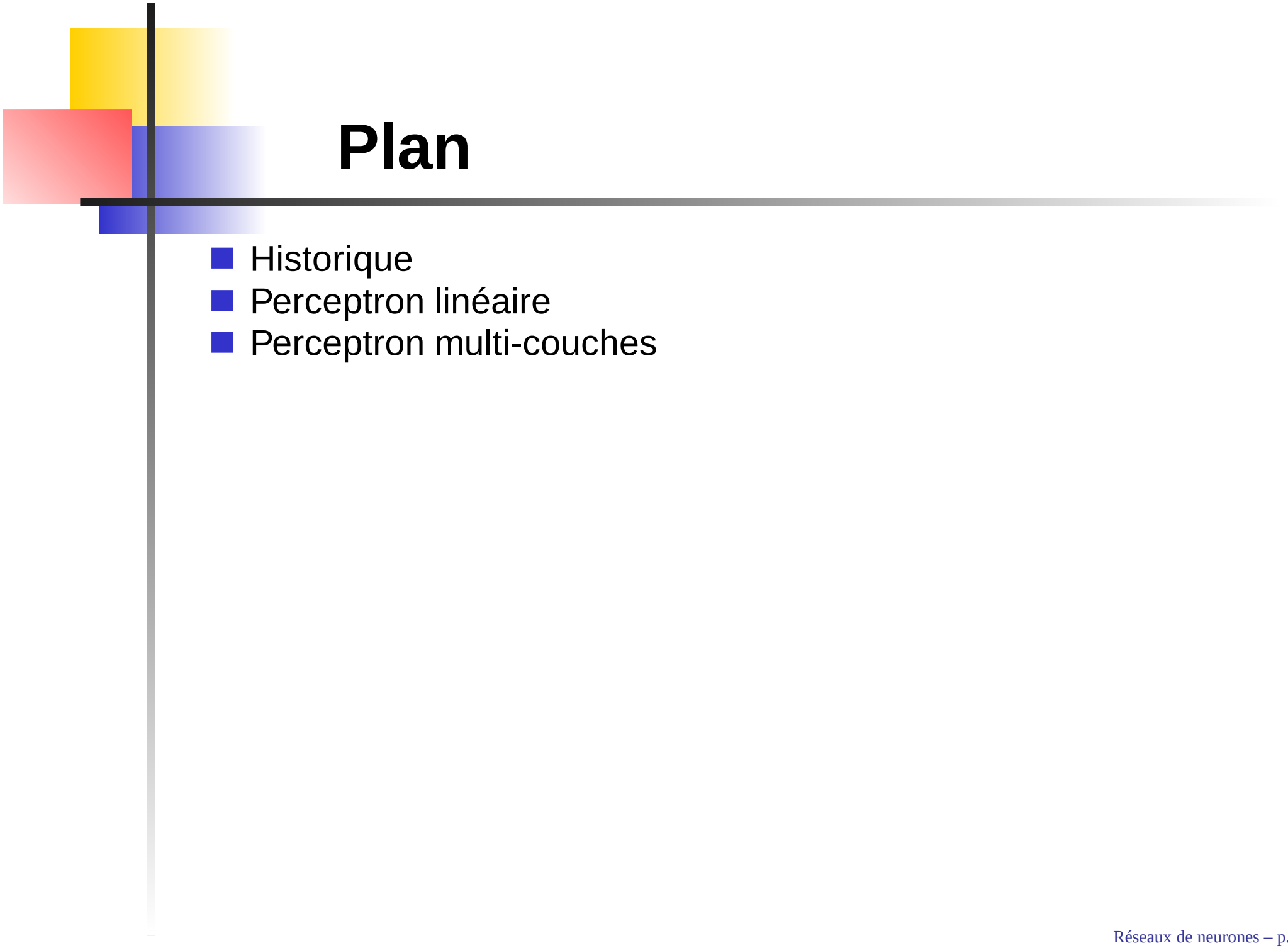
Laboratoire d'Informatique Fondamentale de Marseille

UMR 6166 CNRS – Université de Provence

http://www.lif.univ-mrs.fr

Contexte

- Classification supervisée (*pattern recognition*)
  - $\mathcal{S} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_\ell, y_\ell)\}$ ensemble d'apprentissage
  - $\mathcal{X} = \mathbb{R}^d,\ \mathcal{Y} = \{0,1\}^m$
    - $\mathbf{x}_i$ de classe $c \Leftrightarrow y_i = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$ '1' en c^{eme} position
- Utilisation
  - temps d'apprentissage long autorisé
  - et/ou apprentissage *en ligne* requis
  - nombre de données assez grand
  - interprétabilité du modèle non nécessaire
  - possible bruit sur les données

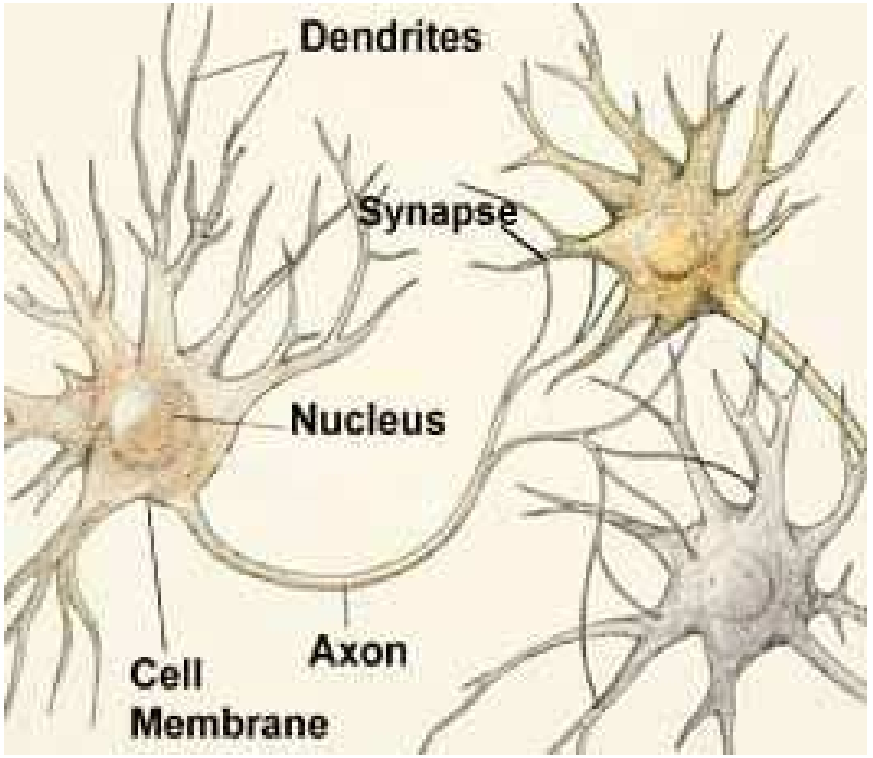


Plan

- Historique
- Perceptron linéaire
- Perceptron multi-couches

Historique (1/2)

- Motivations biologiques
  - systèmes apprenants composés de réseaux connectés de plusieurs unités
  - capacités de mémoire/adaptabilité de ces systèmes

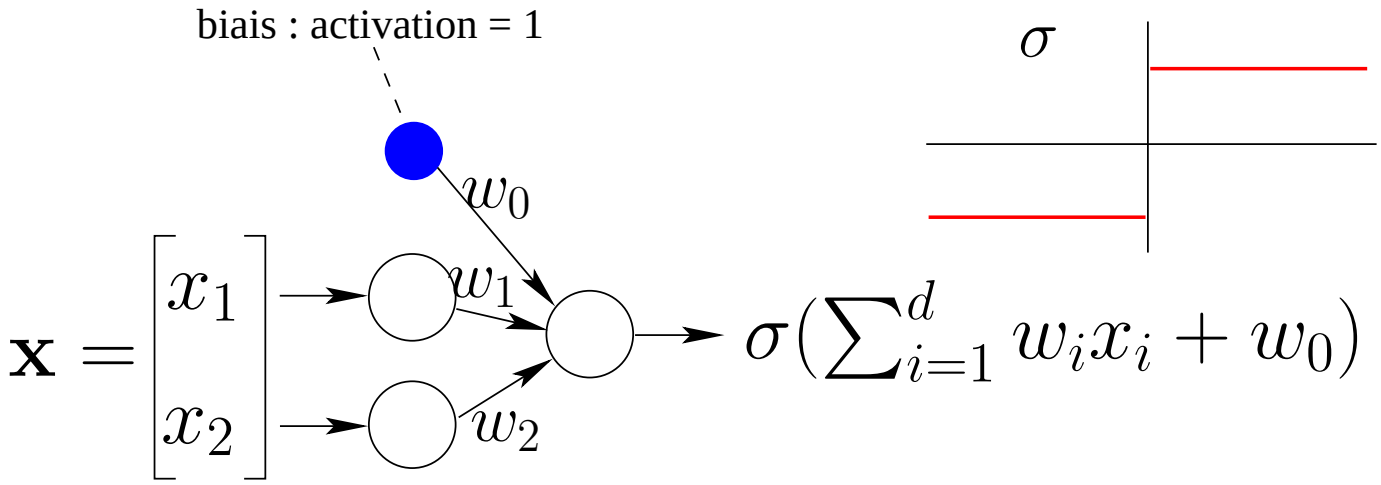


Historique (2/2)

- Caractéristiques réseau de neurones biologique
  - nombre de neurones dans le cerveau : 10^{11} neurones chacun étant connecté à 10^4 autres neurones
  - temps de transmission de l'information entre deux neurones du cerveau : 10^{-3}
  - mais temps d'activation du réseau très rapide : 10^{-1} secondes pour la reconnaissance d'un proche
  - connexions en boucles
- Caractéristiques réseau de neurones artificiels
  - nombre de neurones : de l'ordre de quelques centaines au maximum avec quelques dizaines de connexions
  - temps de transmission de l'information entre deux neurones : 10^{-10} secondes
  - difficulté d'apprentissage avec des connexions en boucle

Perceptron linéaire (1/4)

- Séparateur linéaire



- Classification de $\mathbf{x}$ effectuée en fonction de l'hyperplan

$$\mathbf{w} \cdot \tilde{\mathbf{x}} = 0 \text{ où } \mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} \text{ et } \tilde{\mathbf{x}} = \begin{bmatrix} 1 \\ \mathbf{x} \end{bmatrix}$$

Perceptron linéaire (2/4)

- Algorithme d'apprentissage I
  - Classification binaire $y_i \in \{-1, +1\}$
  - Initialisation $\mathbf{w} = \mathbf{0}$
  - Répéter jusqu'à convergence ou bien atteinte d'un nombre max d'itérations
    - pour tous les exemples $(\mathbf{x}_p, y_p)$ faire
      - si $\sigma(\mathbf{w} \cdot \tilde{\mathbf{x}}_p) = y_p$
        ne rien faire
      - sinon
        $\mathbf{w} \leftarrow \mathbf{w} + y_p \tilde{\mathbf{x}}_p$

Perceptron linéaire (3/4)

- Algorithme d'apprentissage II (descente de gradient)
- Initialiser $\mathbf{w}$ aléatoirement
- Répéter jusqu'à convergence ou atteinte d'un nombre max d'itérations
  - Initialiser Δw_i à 0
  - Pour tous les exemples $(\mathbf{x}_p, y_p)$
    - calculer la sortie $s = \mathbf{w} \cdot \tilde{\mathbf{x}}_p$
    - pour chaque composante $\mathbf{w}_i$

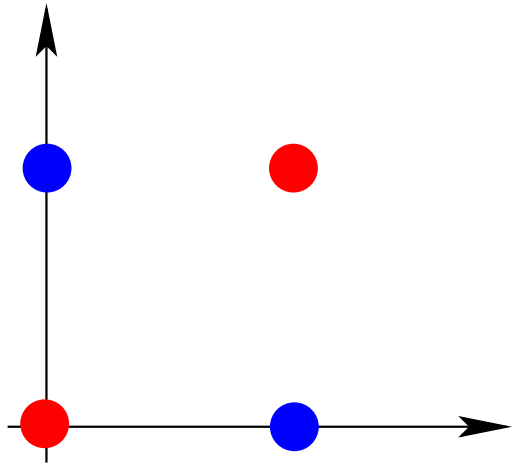
$$\Delta w_i \leftarrow \Delta w_i + \eta(y_p - s)x_{pi}$$

  - pour chaque composante $\mathbf{w}_i$ faire

$$w_i \leftarrow w_i + \Delta w_i$$

Perceptron linéaire (4/4)

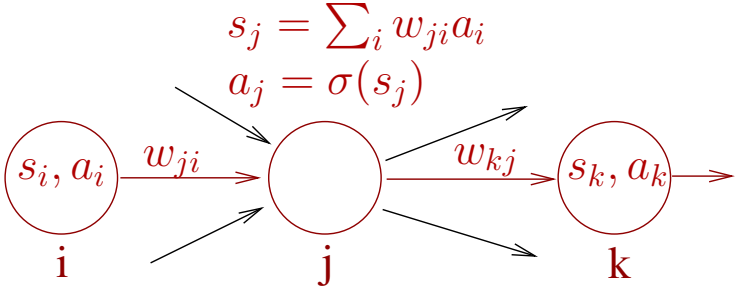
- Propriétés du perceptron linéaire
  - convergence de l'algorithme du perceptron garantie si séparabilité des exemples possible
  - séparation impossible du XOR

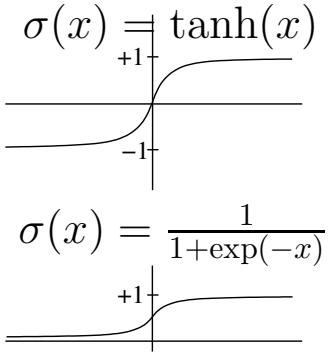


- Extensions
  - perceptron multi-couches
  - kernel adatron [Friess et al., 1998]
  - voted perceptron [Freund and Schapire, 1999]

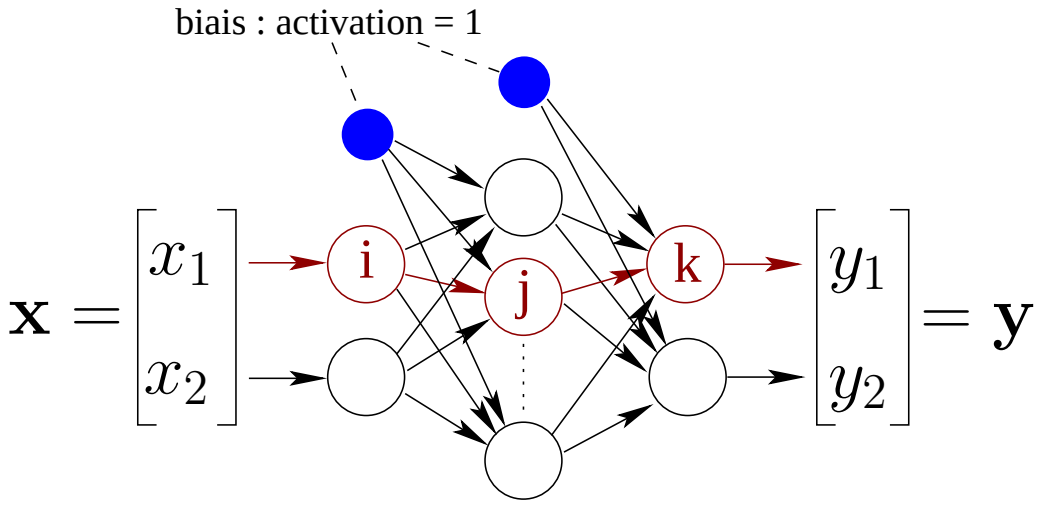
Perceptron multi-couches (1/9)

- Neurone formel





- Perceptron multi-couches



Perceptron multi-couches (2/9)

- Fonction implémentée : $f : \mathbb{R}^d \to \mathbb{R}^m$
- Erreur quadratique, en posant $\mathbf{o}_p = f(\mathbf{x}_p)$

$$E(\mathbf{w}) = \sum_{p=1}^{\ell} E_p(\mathbf{w}) \quad \text{avec} \quad E_p(\mathbf{w}) = \frac{1}{2}\|\mathbf{o}_p - \mathbf{y}_p\|^2 = \frac{1}{2}\sum_{q=1}^{m}(o_{pq} - y_{pq})^2$$

autres fonctions possibles (e.g. cross-entropy)

- Descente de gradient

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta \nabla_{\mathbf{w}} E(\mathbf{w}),\ \eta > 0$$

- Descente de gradient stochastique, à la présentation de l'exemple $(\mathbf{x}_p, \mathbf{y}_p)$

$$\mathbf{w}^{t+1} = \mathbf{w}^t - \eta_t \nabla_{\mathbf{w}} E_p(\mathbf{w}),\ \eta > 0$$

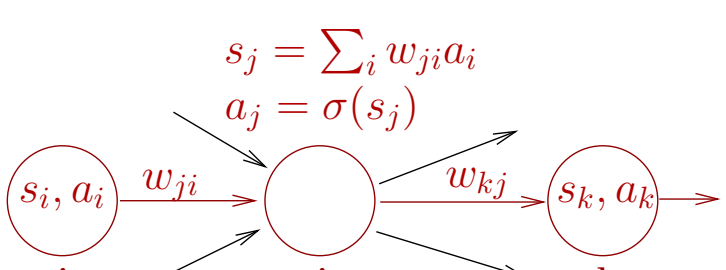
Perceptron multi-couches (3/9)

- η, η_t : pas d'apprentissage, pas d'apprentissage adaptatif
- Propriétés de η_t pour descente de gradient stochastique

$$\sum_t \eta_t \to \infty, \qquad \sum_t \eta_t < \infty$$

- Exercices
  - Montrer que pour η assez petit la descente de gradient permet de diminuer à chaque étape l'erreur E
  - De quelle forme peut être η_t ?

Perceptron multi-couches (4/9)



- Rétropropagation du gradient (j neurone caché, k neurone de sortie), activation logistique
  - dérivation en chaîne : $\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial s_j}\frac{\partial s_j}{\partial w_{ji}} = \frac{\partial E_p}{\partial s_j} a_i = \delta_j a_i$
  - Montrer que pour k et tous les neurones de la couche de sortie

$$\delta_k = \frac{\partial E_p}{\partial s_k} = -(y_{pk} - a_k) a_k (1 - a_k)$$

  - Montrer que pour j et tous les neurones de la couche cachée

$$\delta_j = a_j (1 - a_j) \sum_{k \in Succ(j)} \delta_k w_{kj}$$

Perceptron multi-couches (5/9)

- Algo. apprentissage : 1 couche cachée, $\sigma(x) = (1 + e^{-x})^{-1}$
  - Répéter jusqu'à convergence
    - pour chaque exemple $(\mathbf{x}_p, \mathbf{y}_p)$ faire
      - propager l'information
      - pour chaque neurone de sortie k calculer
        $$\delta_k \leftarrow a_k(1 - a_k)(y_{pk} - a_k)$$
      - pour chaque neurone caché j calculer
        $$\delta_j \leftarrow a_j(1 - a_j) \sum_{k \in Succ(j)} \delta_k w_{kj}$$
      - calculer le pas Δw_{ij}^t par
        $$\Delta w_{ij}^t = -\eta \delta_j a_i$$
      - mettre à jour les w_{ji} avec
        $$w_{ji} \leftarrow w_{ji} + \Delta w_{ij}^t$$
      - passer à l'itération suivante
        $$t \leftarrow t + 1$$

Perceptron multi-couches (6/9)

- Ajout d'un moment : mise à jour selon $w_{ji} \leftarrow w_{ji} + \Delta w_{ij}^t$ avec

$$\Delta^t w_{ij} \leftarrow -\eta \delta_j a_i + \alpha \Delta_{ij}^{t-1} w_{ij}, \quad 0 \leq \alpha < 1$$

  - accélère la convergence de l'algorithme
  - n'a pas d'effet sur la généralisation
- Couches cachées
  - rétropropagation du gradient marche pour n'importe quel nombre de couches cachées
  - couche cachée : changement de représentation (cf exercices)

Perceptron mutli-couches (7/9)

- Exemple d'autres fonctions d'erreurs
  - Erreur quadratique pénalisée (*weight decay*)

$$E(\mathbf{w}) = \sum_{p=1}^{\ell} E_p + \gamma \sum_{i,j} w_{ij}^2, \quad \gamma > 0$$

    - pénalise les réseaux avec des poids importants
    - exercice : ré-écrire l'équation de mise à jour de w_{ij} correspondante
  - Entropy croisée (cross-entropy), cas binaire $y_p \in \{0, 1\}$, 1 neurone de sortie

$$E(\mathbf{w}) = -\sum_{p=1}^{\ell} \left(y_p \log(o_p) + (1 - y_p) \log(1 - o_p) \right), \quad o_p = f(\mathbf{x}_p)$$

Perceptron multi-couches (8/9)

- Capacités de modélisation des perceptrons multi-couches [Cybenko, 1988, Cybenko, 1989]
  - toute fonction continue bornée peut être approchée avec une erreur arbitrairement petite par un PMC à une couche cachée
  - toute fonction peut être approchée avec une erreur arbitrairement petite par un PMC à deux couches cachées
- Importance de la régularisation [Bartlett, 1997]
  - contrôle de la taille des poids pour la généralisation

Perceptron multi-couches (9/9)

- Notions non présentées
  - partage de poids
  - sélection automatique de la structure
    - par ajout de neurones
    - par suppression de connexions
  - gradient conjugué
  - gradient exponentiel
  - . . .
- Réseaux récurrents
- Réseaux RBF

Conclusion

- Perceptron linéaire
  - algorithmes d'apprentissage
  - limitation du perceptron linéaire
- Perceptron multi-couches
  - neurone formel avec activation sigmoidale
  - calcul de gradient par rétropropagation
    - qualité de l'apprentissage malgré les minima locaux
  - gradient stochastique
  - choix de l'architecture
  - régularisation

Références

[Bartlett, 1997] Bartlett, P. L. (1997). For valid generalization the size of the weights is more important than the size of the network. In *Adv. in Neural Information Processing Systems*, volume 9, page 134.

[Cybenko, 1988] Cybenko, G. (1988). Continuous valued neural networks with two hidden layers are sufficient. Technical report, Department of Computer Science, Tufts University, Medford, MA.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2 :303–314.

[Freund and Schapire, 1999] Freund, Y. and Schapire, R. E. (1999). Large Margin Classification Using the Perceptron Algorithm. *Machine Learning*, 37(3) :277–296.

[Friess et al., 1998] Friess, T., Cristianini, N., and Campbell, N. (1998). The Kernel-Adatron Algorithm : a Fast and Simple Learning Procedure for Support Vector Machines. In Shavlik, J., editor, *Machine Learning : Proc. of the* 15^{th} *Int. Conf.* Morgan Kaufmann Publishers.