

## Réseaux de neurones

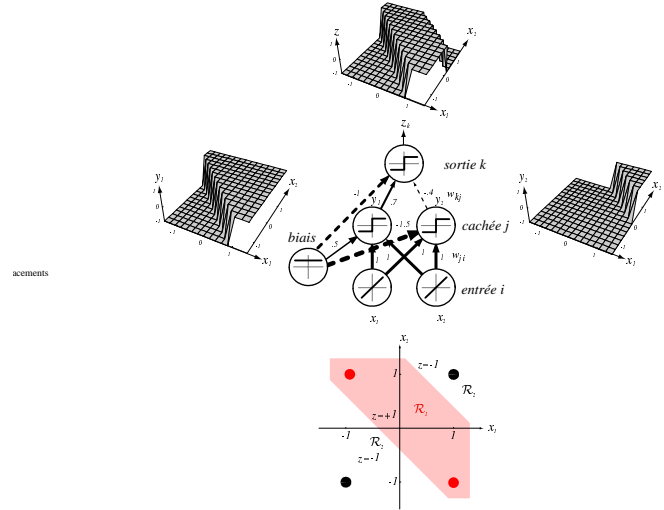
1

### • Objectif

- apprendre des classifieurs "plus riches" que linéaire
- apprendre la **transformation non-linéaire**
- algorithme: **rétropropagation d'erreur**
- régularisation – **sélection de modèle**

## Réseaux de neurones

2



## Réseaux de neurones

3

### • Neurone caché

- entrée = **activation** de réseau:

$$net_j = \sum_{i=1}^d w_{ji}x_i + w_{j0} = \sum_{i=0}^d w_{ji}x_i = \mathbf{w}_j^T \mathbf{x}$$

- sortie:

$$y_j = f(net_j)$$

- exemple:  $f(net) = \text{signe}(net) = \begin{cases} 1 & \text{si } net \geq 0 \\ -1 & \text{si } net < 0 \end{cases}$

- $f(\cdot)$ : **fonction d'activation** ou non-linéarité

- $\mathbf{w}$ : vecteur des **poids** "synaptiques"

## Réseaux de neurones

4

### • Neurone de sortie

- entrée:

$$net_k = \sum_{j=1}^{n_H} w_{kj}y_j + w_{k0} = \sum_{j=0}^{n_H} w_{kj}y_j = \mathbf{w}_k^T \mathbf{y}$$

- sortie:

$$z_k = f(net_k)$$

## Réseaux de neurones

5

### • La **puissance expressive** des réseaux de neurones

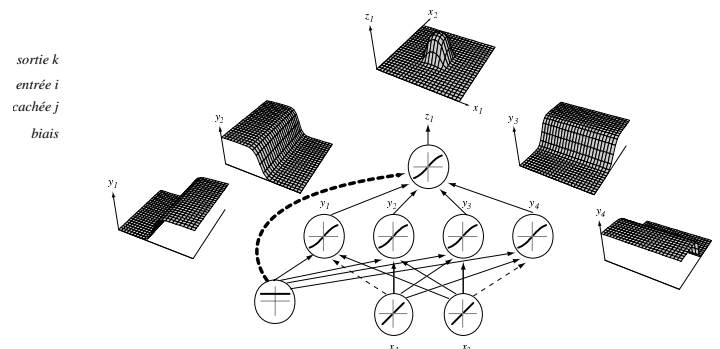
$$g_k(\mathbf{x}) = f \left( \sum_{j=1}^{n_H} w_{kj} f \left( \sum_{i=1}^d w_{ji} x_i + w_{j0} \right) + w_{k0} \right)$$

- erreur d'**approximation**
- toutes les **fonctions continues** peuvent être approchées
- mais:  $n_H \rightarrow \infty$  pour une approximation **exacte**

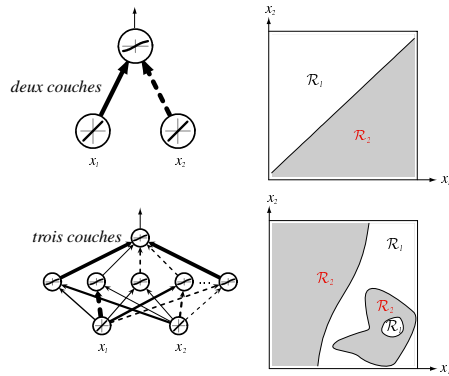
## Réseaux de neurones

6

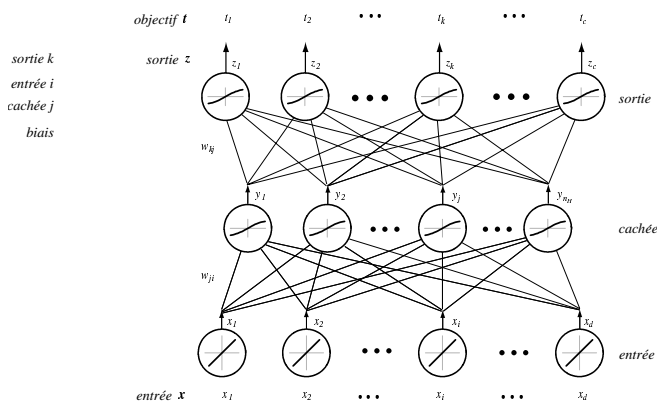
### • La **puissance expressive** des réseaux de neurones



- La **puissance expressive** des réseaux de neurones



- **Rétropropagation d'erreur**



- Couche de **sortie**

$$\frac{\partial J}{\partial w_{kj}} = \frac{\partial J}{\partial net_k} \frac{\partial net_k}{\partial w_{kj}} = -\delta_k \frac{\partial net_k}{\partial w_{kj}} = -\delta_k y_j$$

- **sensibilité:**

$$\delta_k = -\frac{\partial J}{\partial net_k} = -\frac{\partial J}{\partial z_k} \frac{\partial z_k}{\partial net_k} = (t_k - z_k) f'(net_k)$$

- **résultat final:**

$$\Delta w_{kj} = \eta \delta_k y_j = \eta (t_k - z_k) f'(net_k) y_j$$

- **Rétropropagation d'erreur**

- descente de gradient
- règle de chaîne
- extension de l'algorithme **LMS**

- **Rétropropagation d'erreur**

- **erreur d'entraînement:**

$$J(\mathbf{w}) = \frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2 = \frac{1}{2} \|\mathbf{t} - \mathbf{z}\|^2$$

- **descente de gradient:**

$$\Delta \mathbf{w} = -\eta \frac{\partial J}{\partial \mathbf{w}}; \quad \Delta w_{pq} = -\eta \frac{\partial J}{\partial w_{pq}}$$

- **mise à jour des poids:**

$$\mathbf{w}(m+1) = \mathbf{w}(m) + \Delta \mathbf{w}(m)$$

- Couche **cachée**

$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} f'(net_j) x_i$$

- **premier terme:**

$$\begin{aligned} \frac{\partial J}{\partial y_j} &= \frac{\partial}{\partial y_j} \left[ \frac{1}{2} \sum_{k=1}^c (t_k - z_k)^2 \right] \\ &= -\sum_{k=1}^c (t_k - z_k) \frac{\partial z_k}{\partial y_j} \\ &= -\sum_{k=1}^c (t_k - z_k) \frac{\partial z_k}{\partial net_k} \frac{\partial net_k}{\partial y_j} \\ &= -\sum_{k=1}^c (t_k - z_k) f'(net_k) w_{kj} \\ &= \sum_{k=1}^c \delta_k w_{kj} \end{aligned}$$

- Couche **cachée**

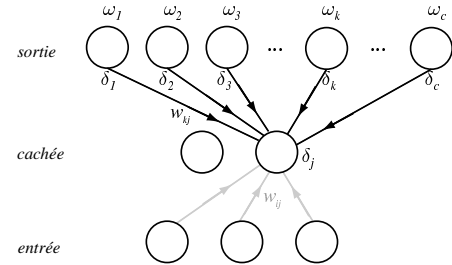
$$\frac{\partial J}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} \frac{\partial y_j}{\partial net_j} \frac{\partial net_j}{\partial w_{ji}} = \frac{\partial J}{\partial y_j} f'(net_j) x_i$$

- résultat **final**:

$$\Delta w_{ji} = \eta \delta_j x_i = \eta \underbrace{\left[ \sum_{k=1}^c w_{kj} \delta_k \right]}_{\delta_j} f'(net_j) x_i$$

- Rétropropagation des **sensibilités**

sortie  $k$   
entrée  $i$   
cachée  $j$   
biais



- Protocoles d'entraînement

- stochastique**: mise à jour après chaque point d'entraînement tiré par hasard
- (en-ligne)**: comme stochastique, mais chaque point est traité seulement une fois)

RETROPROPAGATIONSTOCHASTIQUE( $\Theta, \eta$ )

- 1 initialiser  $w$
- 2 **faire**
- 3  $x \in D_n$  choisi par hasard
- 4  $w_{ji} \leftarrow w_{ji} + \eta \delta_j x_i$
- 5  $w_{kj} \leftarrow w_{kj} + \eta \delta_k y_j$
- 6 **jusqu'à**  $\|\nabla J(w)\| < \Theta$
- 7 **retourner**  $w$

- Protocoles d'entraînement

- batch**: mise à jour après avoir traité tous les points d'entraînement
- plusieurs **époques**

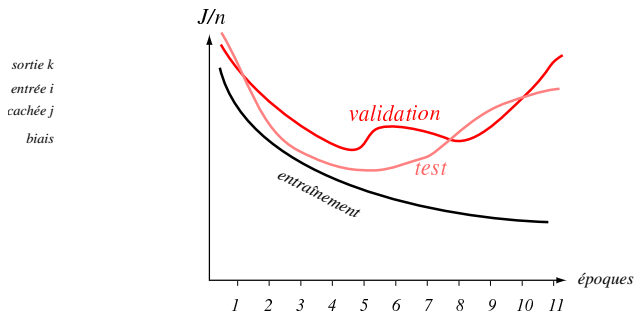
RETROPROPAGATIONBATCH( $\Theta, \eta$ )

- 1 initialiser  $w$
- 2 **faire**
- 3  $\Delta w_{ji} \leftarrow \Delta w_{kj} \leftarrow 0$
- 4 **pour**  $m = 1$  à  $n$  **faire**
- 5  $x \leftarrow x_m$
- 6  $\Delta w_{ji} \leftarrow \Delta w_{ji} + \eta \delta_j x_i$
- 7  $\Delta w_{kj} \leftarrow \Delta w_{kj} + \eta \delta_k y_j$
- 8  $w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$
- 9  $w_{kj} \leftarrow w_{kj} + \Delta w_{kj}$
- 10 **jusqu'à**  $\|\nabla J(w)\| < \Theta$
- 11 **retourner**  $w$

- Courbes d'apprentissage

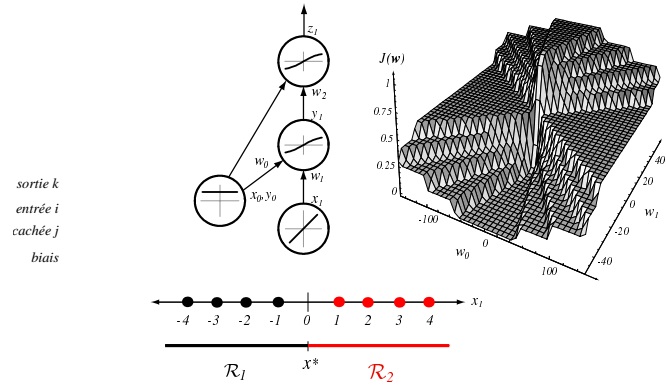
- ensemble d'**entraînement**
- ensemble de **test**: **évaluation** de la performance
- ensemble de **validation**: **ajuster** les paramètres (par exemple: nombre d'époques)

• Courbes d'apprentissage



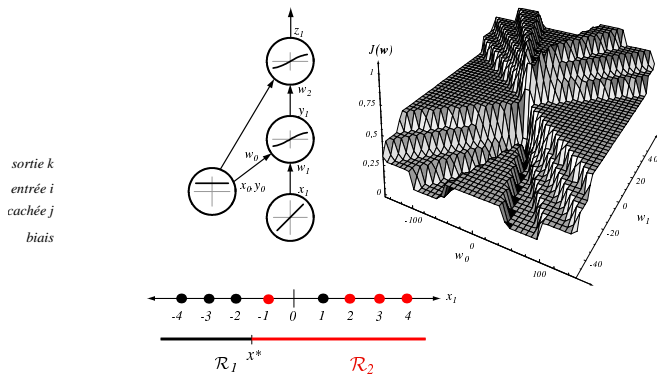
sortie k  
entrée i  
cachée j  
biais

• Surface d'erreur



sortie k  
entrée i  
cachée j  
biais

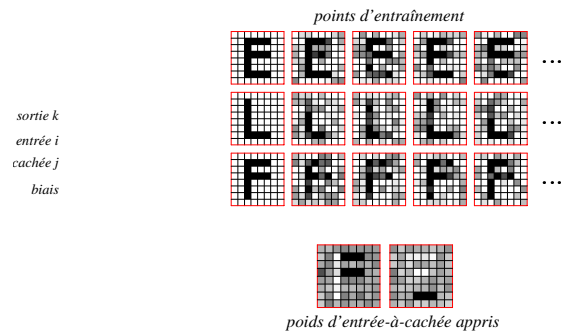
• Surface d'erreur



sortie k  
entrée i  
cachée j  
biais

• Représentation des poids

- extraction des traits, filtrage



sortie k  
entrée i  
cachée j  
biais

• La fonction d'activation

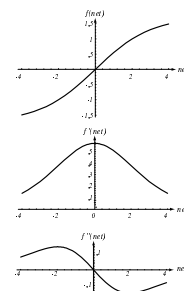
- non-linéaire
- bornée
- continue et lisse
- monotone (?)
- linéaire sur les petites entrées

• La fonction sigmoïde

$$f(x) = \text{atanh}(bx) = a \left( \frac{1 - e^{-bx}}{1 + e^{-bx}} \right); \quad a = 1.716; \quad b = 2/3$$

ts

sortie k  
entrée i  
cachée j  
biais



- Détails pratiques

- normaliser l'entrée:  $\mu = 0, \sigma = 1$
- valeurs d'objectif:  $\pm 1$
- entraînement avec bruit ( $\sigma \ll 1$ )
- fabriquer des données supplémentaires
- nombre d'unités cachées
- initialisation des poids – apprentissage uniforme:  $U[-1/\sqrt{d}, 1/\sqrt{d}]$

## Pré-traitement de la donnée

27

- Normaliser l'entrée:  $\mu = 0, \sigma = 1$ 
  - transformation blanchissante
  - égaliser le poids des attributs
- Valeurs d'objectif:  $\pm 1$ 
  - au milieu intervalle dynamique de la fonction d'activation
- Fabriquer des données supplémentaires
- Entraînement avec bruit ( $\sigma \ll 1$ )

## Choix des paramètres

29

- Initialisation des poids
  - apprentissage uniforme: les poids sont appris à la même vitesse
  - $net_j \in [-1, 1]$
  - $d$  paramètres aléatoires uniformes dans  $[-\tilde{w}, \tilde{w}]$
  - variance de la somme  $\approx \tilde{w}^2 d$
  - $\tilde{w} = 1/\sqrt{d}$
  - couche de sortie:  $\tilde{w} = 1/\sqrt{n_H}$

- Détails pratiques

- taux d'apprentissage
- impulsion (momentum)
- weight decay
- indices
- terminer au plus tôt (early stopping)
- nombre de couches cachées
- autres fonctions de critère

## Choix des paramètres

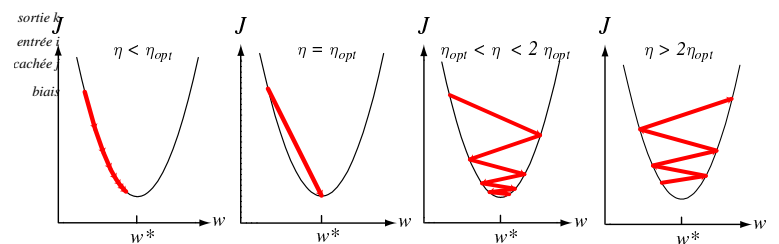
28

- Nombre d'unités cachées
  - régularisation – sélection de modèle
  - règle heuristique:  $n/10$
  - en utilisant un ensemble de validation

## Choix des paramètres

30

- Taux d'apprentissage

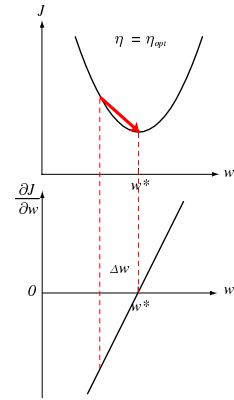


• Taux d'apprentissage

- $\frac{\partial^2 J}{\partial w^2} \Delta w = \frac{\partial J}{\partial w}$
- $\eta_{opt} = \left( \frac{\partial^2 J}{\partial w^2} \right)^{-1}$

• Taux d'apprentissage

sortie  $k$   
 entrée  $i$   
 cachée  $j$   
 biais



Autres heuristiques

• Impulsion (momentum)

- $\mathbf{w}(m+1) = \mathbf{w}(m) + (1-\alpha)\Delta\mathbf{w}(m) + \alpha\Delta\mathbf{w}(m-1)$

• Weight decay

- $w^{new} = w^{old}(1-\epsilon)$
- $J_{ef} = J(\mathbf{w}) + \frac{\epsilon}{2\eta} \mathbf{w}'\mathbf{w}$
- $J_{ef} = J(\mathbf{w}) + \frac{\epsilon}{2\eta} \sum_{i,j} \frac{w_{ij}^2 / (\mathbf{w}'\mathbf{w})}{1 + w_{ij}^2 / (\mathbf{w}'\mathbf{w})}$

• Terminer au plus tôt (early stopping)

Autres heuristiques

• Indices

sortie  $k$   
 entrée  $i$   
 cachée  $j$   
 biais

