

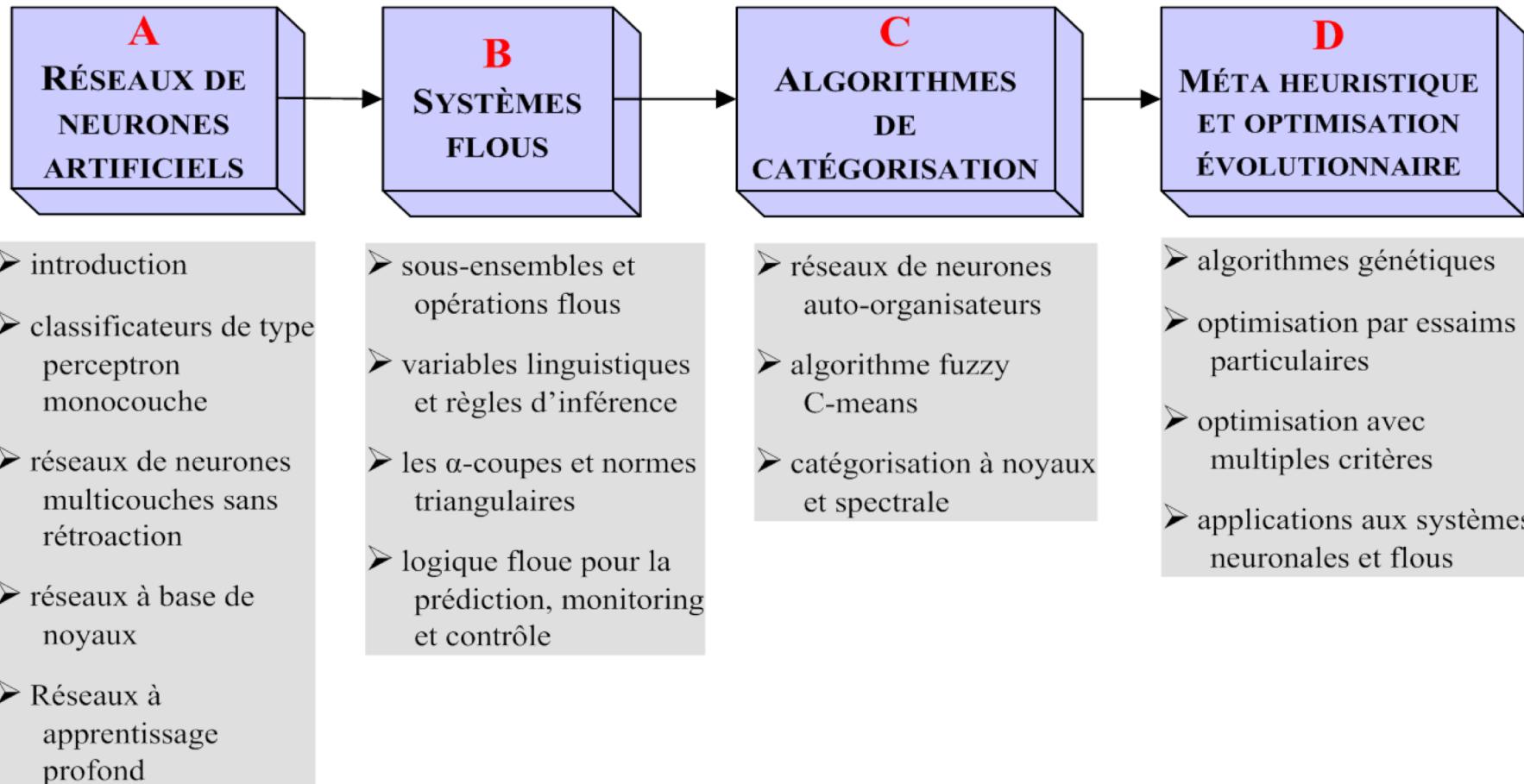
SYS843

A.5 Réseaux de neurones à base de noyaux

**Eric Granger
Ismail Ben Ayed**

Hiver 2017

CONTENU DU COURS



Sommaire – Section A.5

A.5 Réseaux de neurones à base de noyaux

1. Séparabilité des données
2. Problème d'interpolation
3. Réseaux classifieurs RBF
4. SVM: Séparateurs à vaste marge

1. Séparabilité des données

Théorème sur la séparabilité de patrons (Cover, 1965)

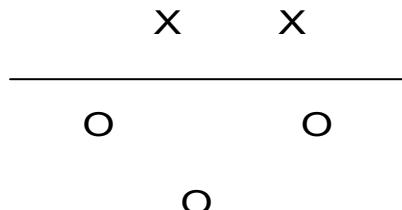
- ▶ Un problème de classification qui est transposé de façon non linéaire dans un espace de haute dimensionnalité a une plus grande probabilité d'être séparable qu'en basse dimensionnalité
- ▶ Séparabilité – problème à 2 classes:
 - transpose \mathbf{x} dans *l'espace image* de haute dimensionnalité avec des fonctions cachées $\varphi(\mathbf{x})$ non-linéaires à valeur réelle:
$$\varphi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \dots, \varphi_{m_1}(\mathbf{x})]$$
 - le problème est ‘ φ – séparable’ s’il existe un vecteur de paramètres \mathbf{w} à m_1 dimensions tel que:
$$\begin{cases} \mathbf{w}^T \varphi(\mathbf{x}) > 0, & \mathbf{x} \in C_1 \\ \mathbf{w}^T \varphi(\mathbf{x}) < 0, & \mathbf{x} \in C_2 \end{cases}$$

1. Séparabilité des données

Théorème sur la séparabilité de patrons (Cover, 1965)

► Exemple 2D – cas des fonctions cachées φ polynomiales:

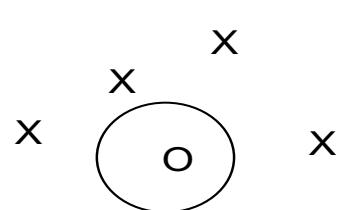
- classe de transformations obtenue à partir d'une combinaison linéaire des produits des coordonnées de $\mathbf{x} = (x_1, x_2)$
- variété d'ordre r : $y = \sum_{0 \leq i_1 \leq i_2 \dots \leq i_r \leq m_1} w_{i_1 i_2 \dots i_N} x_{i_1} x_{i_2} \dots x_{i_r} = 0$



$$\varphi(\mathbf{x}) = [x_1, x_2]$$

$$w_1 x_1 + w_2 x_2 + w_0 = 0$$

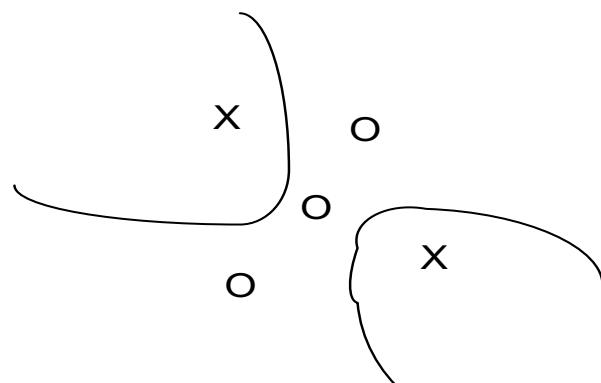
(2 fonctions de base)



$$\varphi(\mathbf{x}) = [x_1^2, x_2^2, x_1, x_2]$$

$$w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_0 = 0$$

(4 fonctions de base)



$$\varphi(\mathbf{x}) = [x_1^2, x_2^2, x_1, x_2, x_1 x_2]$$

$$w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + w_0 = 0$$

(5 fonctions de base)

1. Séparabilité des données

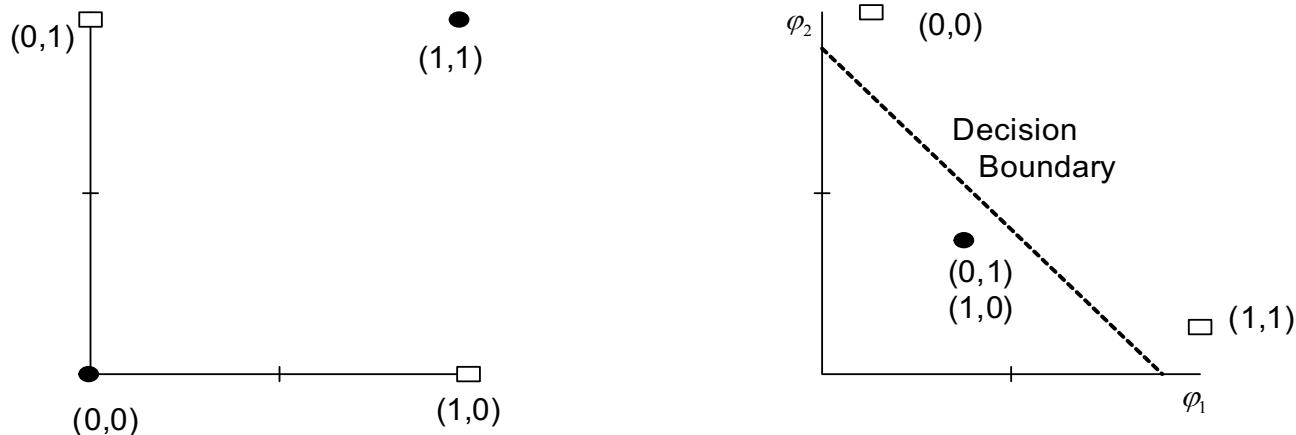
Théorème sur la séparabilité de patrons (Cover, 1965)

- Exemple – cas des fonctions cachées Gaussiennes: problème de la fonction logique XOR (linéairement non séparable)

$$\varphi_1(\mathbf{x}) = e^{-\|\mathbf{x}-\boldsymbol{\mu}_1\|^2}, \quad \boldsymbol{\mu}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\varphi_2(\mathbf{x}) = e^{-\|\mathbf{x}-\boldsymbol{\mu}_2\|^2}, \quad \boldsymbol{\mu}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

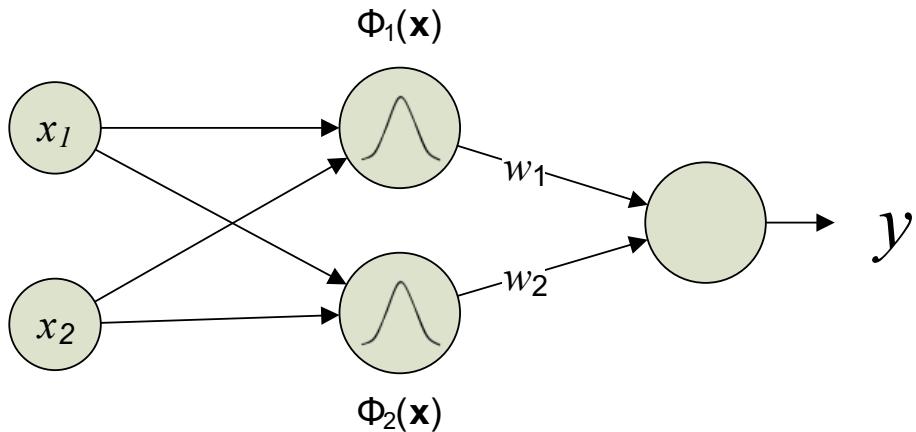
Input pattern \mathbf{x}	First Hidden Function $\varphi_1(\mathbf{x})$	Second Hidden Function $\varphi_2(\mathbf{x})$
(1,1)	1	0.1353
(0,1)	0.3678	0.3678
(0,0)	0.1353	1
(1,0)	0.3678	0.3678



1. Séparabilité des données

Théorème sur la séparabilité de patrons (Cover, 1965)

- **Exemple – cas des fonctions cachées Gaussiennes:** problème de la fonction logique XOR (linéairement non séparable)



Dans l'espace (caché) des descripteurs: Lorsque projetés dans l'espace des descripteurs (Φ_1 , Φ_2), C1 et C2 deviennent *séparables linéairement*. Donc, un classificateur linéaire avec $\Phi_1(\mathbf{x})$ et $\Phi_2(\mathbf{x})$ comme entrées peut être utilisé pour résoudre le problème du XOR (OU exclusif).

Sommaire – Section A.5

A.5 Réseaux de neurones à base de noyaux

1. Séparabilité des données
- 2. Problème d'interpolation**
3. Réseaux classifieurs RBF
4. SVM: Séparateurs à vaste marge

2. Problème d'interpolation

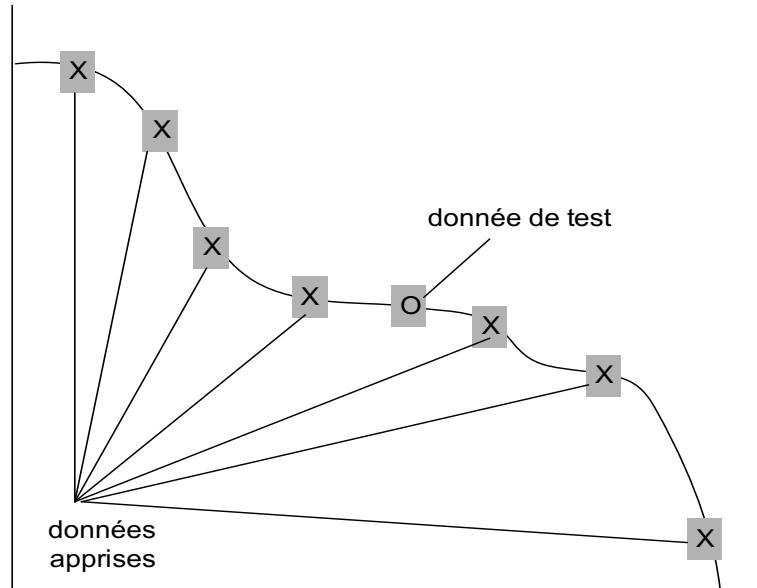
Définitions

- ▶ **Méthodes à fonctions à bases radiales – un domaine de recherche actif an analyse numérique depuis environ 1980 (*kernel methods*)**
 - solution au problème d'interpolation multi variable
 - approximation de fonctions dans des espaces multidimensionnels
- ▶ **Réseau RBF – un réseau de neurone pour la classification et la régression (Broomhead et Low, 1988)**
 - un réseau de neurones non-récurrent ('feed forward') à apprentissage supervisé
 - conception inspirée de méthodes d'interpolation qui exploitent les RBF

2. Problème d'interpolation

► Méthodes d'interpolation:

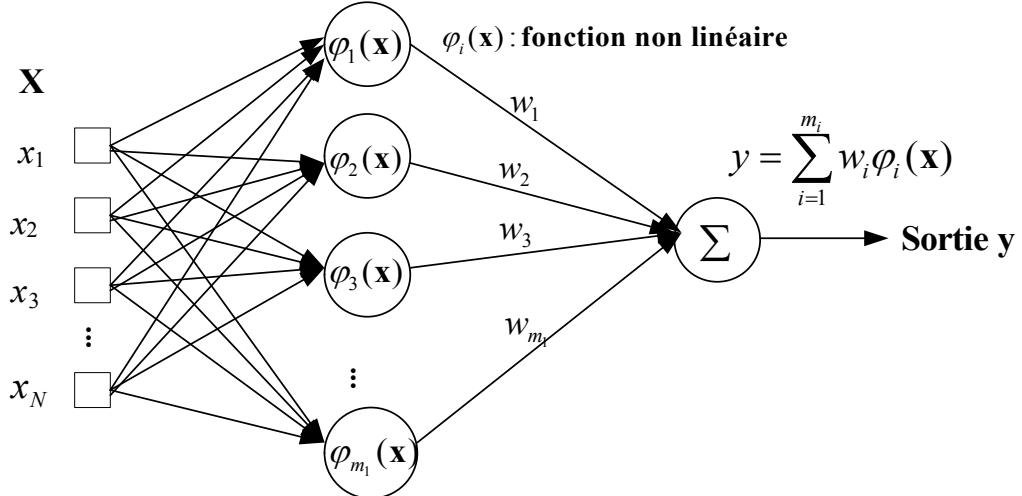
- **Apprentissage:** trouver une surface multidimensionnelle qui correspond le mieux avec les données d'entraînement
- **Généralisation:** utiliser cette surface pour interpoler les données de test



2. Problème d'interpolation

► Structure générale d'un réseau de neurones RBF:

- **couche cachée:** transformation non linéaire $\mathbf{x} \rightarrow \varphi(\mathbf{x})$
 - chaque neurone constitue une fonction cachée $\varphi_i(\mathbf{x})$ (*i.e.*, RBF) pour la transformation non linéaire des patrons d'entrée \mathbf{x}
 - le nombre de neurones est généralement bien plus grand que le nombre de nœuds d'entrée
- **couche de sortie:** transformation linéaire $\varphi(\mathbf{x}) \rightarrow \mathbf{y}$
 - combinaison linéaire des fonctions $\varphi_i(\mathbf{x})$ pour produire une sortie



2. Problème d'interpolation

Interpolation

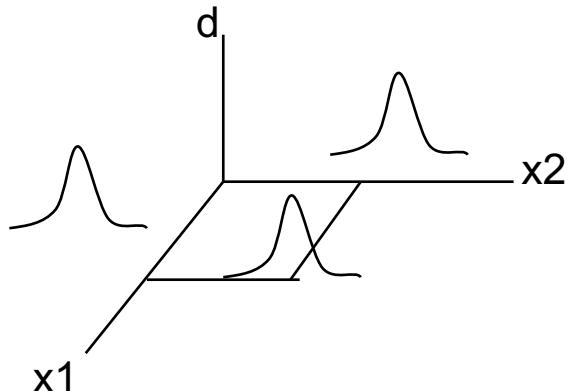
- ▶ **Problème:** étant donnée N patrons différents $\{\mathbf{x}_i \in R^{m_0} | i = 1, 2, \dots, N\}$ et les N réponses désirées correspondantes $\{d_i \in R^1 | i = 1, 2, \dots, N\}$, trouvez une fonction $y : R^N \rightarrow R^1$ qui satisfait: $y(\mathbf{x}_i) = d_i$
- ▶ **La technique RBF consiste à utiliser une fonction de la forme:**

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \varphi_{13} & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \varphi_{21} & \varphi_{2N} \\ \vdots & & & \\ \varphi_{N1} & \varphi_{N2} & \varphi_{N3} & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \text{ avec } \varphi_{ji} = \varphi(\|\mathbf{x}_j - \mathbf{x}_i\|)$$

$$y(\mathbf{x}) = \sum_{i=1}^N w_i \varphi(\|\mathbf{x} - \mathbf{x}_i\|)$$

ou $\Phi = \begin{bmatrix} \varphi_{11} & \varphi_{12} & \varphi_{13} & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \varphi_{21} & \varphi_{2N} \\ \vdots & & & \\ \varphi_{N1} & \varphi_{N2} & \varphi_{N3} & \varphi_{NN} \end{bmatrix}; \mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix}; \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}$

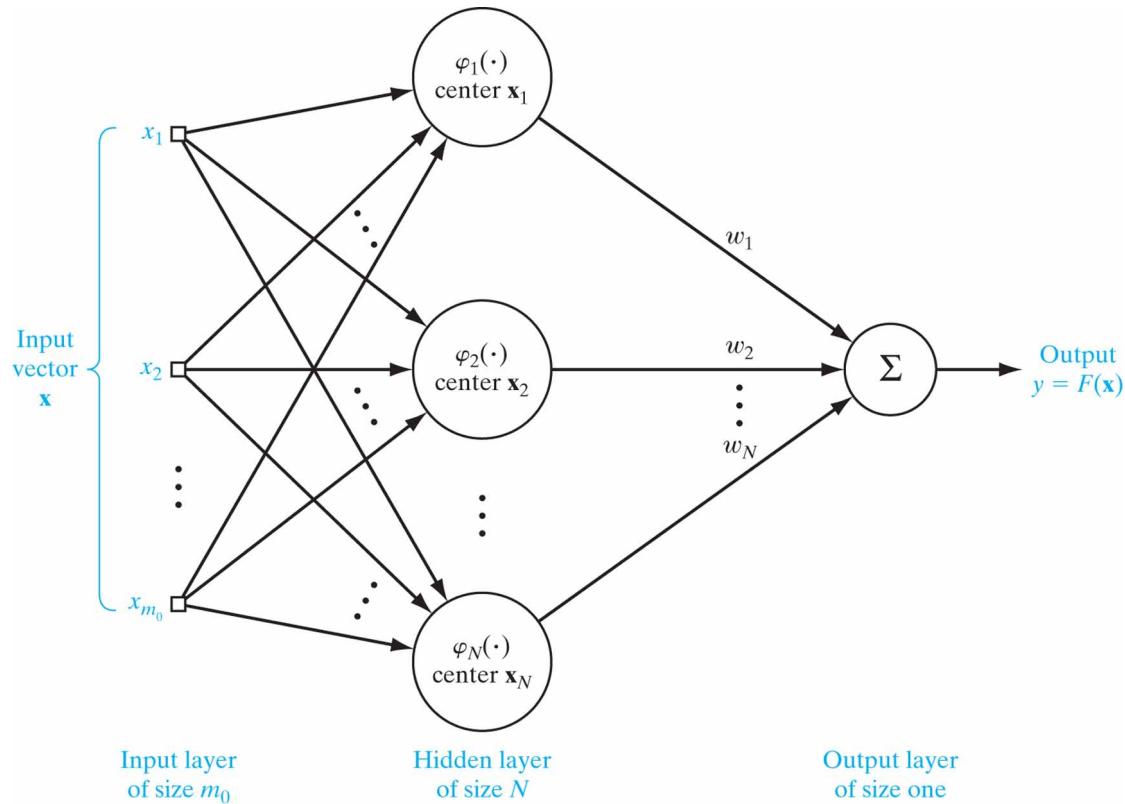
$$\Phi \mathbf{w} = \mathbf{d}, \text{ alors } \mathbf{w} = \Phi^{-1} \mathbf{d}$$



2. Problème d'interpolation

Interpolation

► Réseau RF basé sur la théorie d'interpolation



2. Problème d'interpolation

Interpolation

► Théorème de Micchelli (1986):

- Si $\{\mathbf{x}_i\}_{i=1}^N$ est un ensemble de patrons distincts, alors la matrice N -par- N d'interpolation Φ est non singulière (i.e., une solution existe).

► Théorème de Light (1971):

- Si $\{\mathbf{x}_i\}_{i=1}^N$ est un ensemble de patrons distincts, une matrice d'interpolation Φ avec éléments de la forme

$$\varphi_{ji} = \varphi(\|\mathbf{x}_j - \mathbf{x}_i\|)$$

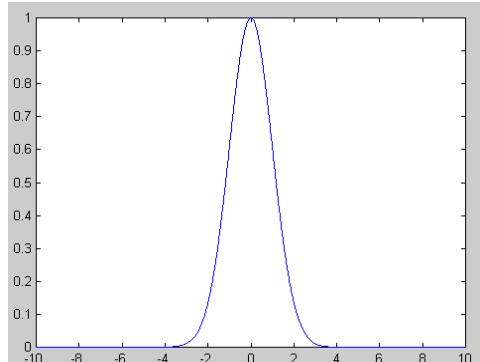
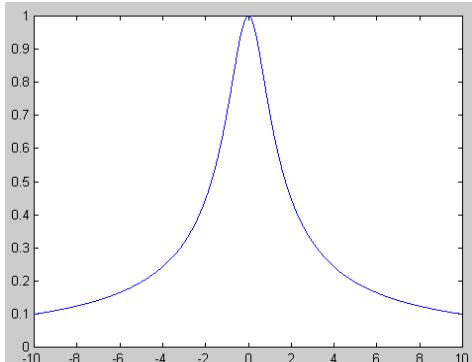
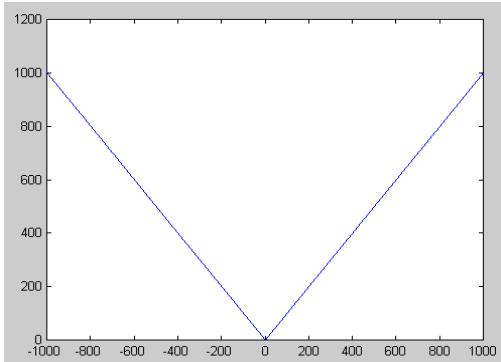
est positive et définie.

2. Problème d'interpolation

Interpolation

► RBF communes qui respectent le théorème de Light:

- fonctions multi quadratiques: $\varphi(r) = (r^2 + c^2)^{1/2}$ pour un $r > 0$, $r \in \mathbb{R}$
- fonctions multi quadratiques inverse: $\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}}$ pour un $r > 0$, $r \in \mathbb{R}$
- fonctions Gaussiennes: $\varphi(r) = \exp\left\{-\frac{r^2}{2\sigma^2}\right\}$ pour un $r > 0$, $r \in \mathbb{R}$



2. Problème d'interpolation

Interpolation

- **Exemple en 1-D:** données d'entraînement $\{(x_i, y_i)\} = \{(-1, 1), (0, 2), (1, 1)\}$ avec RBF multi quadratique:

$$\varphi(r) = \sqrt{(r^2 + 0.5)}$$

$$\begin{aligned} y(\mathbf{x}) &= w_1 \varphi(\|\mathbf{x} - \mathbf{x}_1\|) + w_2 \varphi(\|\mathbf{x} - \mathbf{x}_2\|) + w_3 \varphi(\|\mathbf{x} - \mathbf{x}_3\|) \\ &= w_1 \sqrt{(\mathbf{x} + 1)^2 + 0.5} + w_2 \sqrt{\mathbf{x}^2 + 0.5} + w_3 \sqrt{(\mathbf{x} - 1)^2 + 0.5} \end{aligned}$$

$$\begin{bmatrix} \sqrt{0.5} \sqrt{1.5} \sqrt{4.5} \\ \sqrt{1.5} \sqrt{0.5} \sqrt{1.5} \\ \sqrt{4.5} \sqrt{1.5} \sqrt{0.5} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} \sqrt{0.5} \sqrt{1.5} \sqrt{4.5} \\ \sqrt{1.5} \sqrt{0.5} \sqrt{1.5} \\ \sqrt{4.5} \sqrt{1.5} \sqrt{0.5} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

2. Problème d'interpolation

Interpolation

- Selon les théorèmes de Michelli et Light, on peut résoudre:

$$\mathbf{w} = \boldsymbol{\Phi}^{-1} \mathbf{d}$$

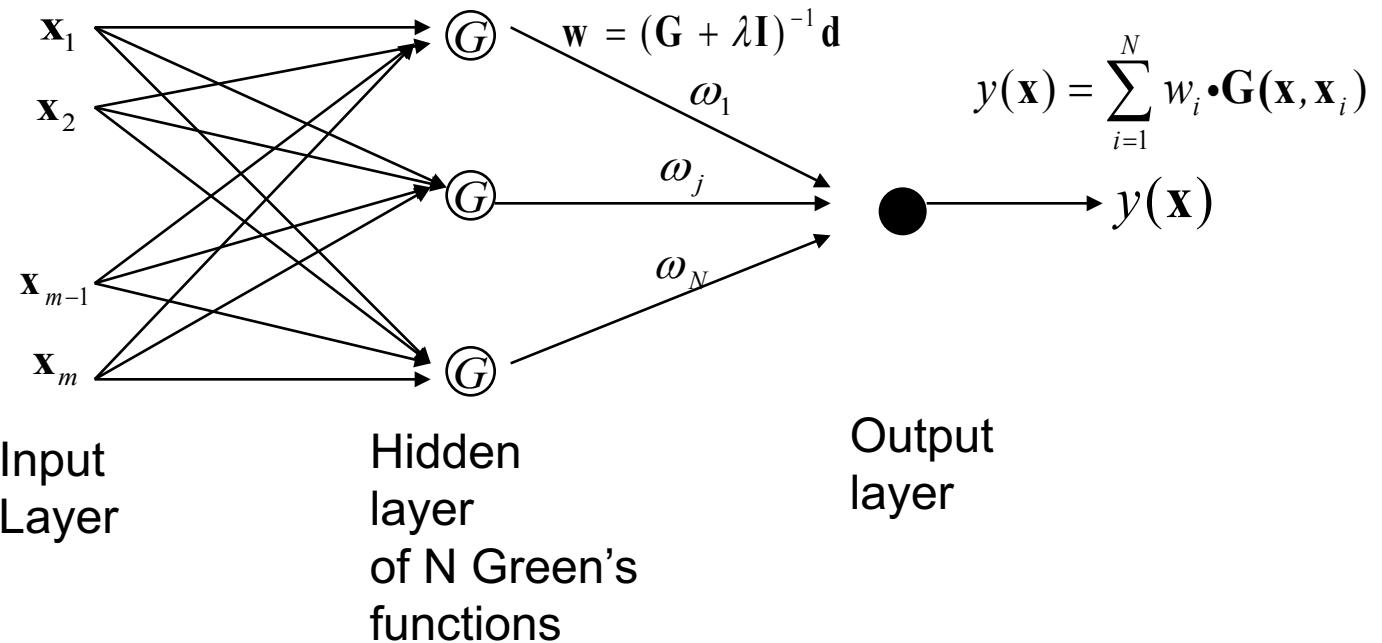
- mais, en pratique, on veut résoudre quand $\boldsymbol{\Phi}$ est arbitrairement proche d'une matrice singulière
- pour avoir des solutions uniques:
 - théorie de régularisation (solution exacte)
 - réseau RBF généralisé (estimation)

2. Problème d'interpolation

Réseau de régularisation

► Réseau RBF de régularisation:

- la sortie est la somme pondérée des sorties de la couche cachée



2. Problème d'interpolation

Réseau de régularisation

- ▶ **Propriétés d'un réseau RBF de régularisation:**
[Poggio et Girosi, 1990]

😊 **approximateurs universels:** ils peuvent approximer n'importe quelle fonction continue arbitrairement bien avec un nombre suffisant de neurones cachées:

- étant donnée une fonction non linéaire inconnue f , il existe toujours un choix de coefficients w qui approxime f mieux que tout autre choix possible;
- ses solutions sont optimales: il minimise la fonction de coût $\varepsilon(y)$.

😩 **complexité de calculs:** la correspondance des neurones cachées avec N patrons donne un réseau très coûteux

- calcul des poids w est $O(N^3)$ car on doit inverser une matrice N-par-N
- difficile à réaliser en pratique pour de grands N ...

2. Problème d'interpolation

Réseaux RBF généralisés

► Un approximation de la solution régularisée:

- solution sous optimale dans l'espace à dimension $M < N$, qui est alors moins coûteuse
- on peut dériver un approximation avec la méthode de Galerkin's [Poggio and Girosi, 1990]:

$$y^*(\mathbf{x}) = \sum_{i=1}^M w_i \bullet \varphi_i(\mathbf{x}) = \sum_{i=1}^M w_i \bullet G(\|\mathbf{x} - \mathbf{t}_i\|)$$

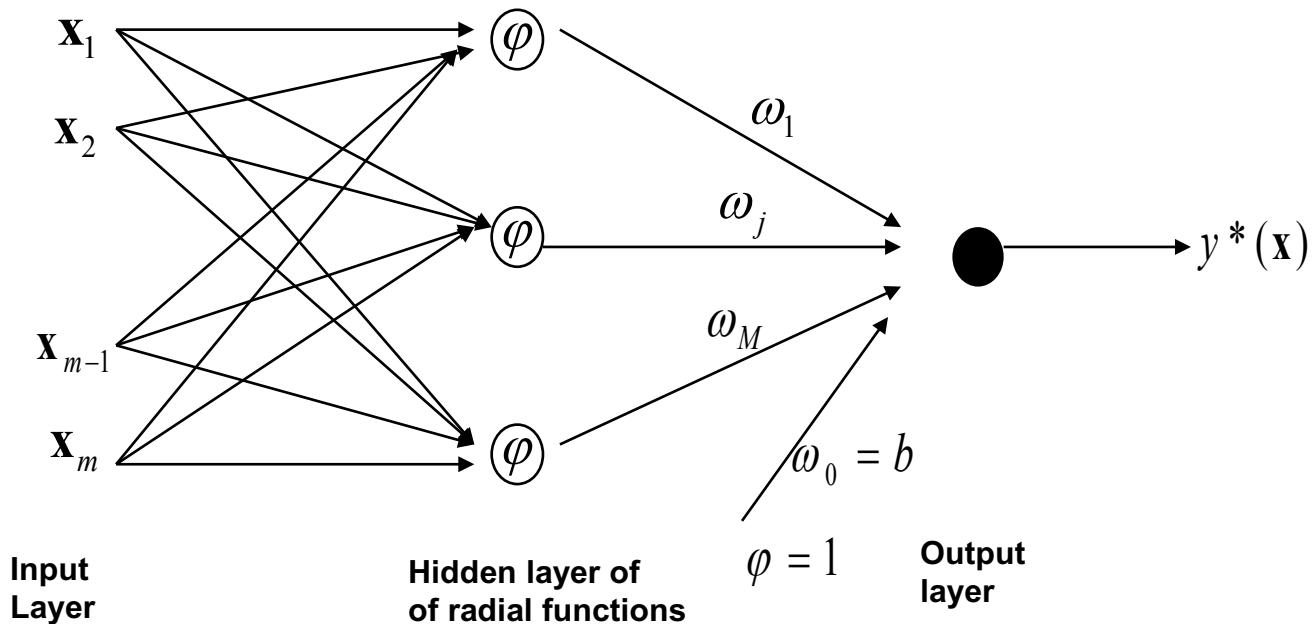
car: $\varphi_i(\mathbf{x}) = G(\|\mathbf{x} - \mathbf{t}_i\|) \quad i = 1, 2, \dots, M \quad M < N$

où les centre \mathbf{t}_i et les poids w_i sont à déterminer.

2. Problème d'interpolation

Réseaux RBF généralisés

► Architecture générale:



2. Problème d'interpolation

Réseaux RBF généralisés

► Détermination des poids w_i :

- Un nouvelle fonction de coût:

$$\varepsilon(y^*) = \sum_{i=1}^N \left(d_i - \sum_{j=1}^M w_j \bullet G(\|\mathbf{x}_i - \mathbf{t}_j\|) \right)^2 + \lambda \|\mathbf{D}y^*\|^2$$

- la minimisation de cette fonction de coût en fonction du vecteur de poids donne: $(\mathbf{G}^T \mathbf{G} + \lambda \mathbf{G}_0) \mathbf{w} = \mathbf{G}^T \mathbf{d}$

$$\mathbf{G}_0 = \begin{bmatrix} G(\mathbf{t}_1, \mathbf{t}_1) & G(\mathbf{t}_1, \mathbf{t}_2) & \dots & G(\mathbf{t}_1, \mathbf{t}_M) \\ G(\mathbf{t}_2, \mathbf{t}_1) & G(\mathbf{t}_2, \mathbf{t}_2) & \dots & G(\mathbf{t}_2, \mathbf{t}_M) \\ \ddots & \ddots & \vdots & \vdots \\ G(\mathbf{t}_M, \mathbf{t}_1) & G(\mathbf{t}_M, \mathbf{t}_2) & \dots & G(\mathbf{t}_M, \mathbf{t}_M) \end{bmatrix}$$

2. Problème d'interpolation

Réseaux RBF généralisés

Il a été démontré que lorsque le paramètre $\lambda \rightarrow 0$, le vecteur \mathbf{w} converge vers une solution pseudo-inverse pour $M \leq N$ [Broomhead et Lowe, 1988]:

$$\mathbf{w} = \mathbf{G}^+ \mathbf{d}$$

$$\mathbf{w} = (\mathbf{G}^T \mathbf{G})^{-1} \mathbf{G}^T \mathbf{d}$$

2. Problème d'interpolation

Réseaux RBF généralisés

► Comparaison entre réseaux RBF régularisés (exact) et généralisés (estimation):

- le nombre de neurones de la couche cachée:
 - régularisation: N
 - généralisés: $M \leq N$ (selon la complexité de problème)
- les paramètres inconnus à estimer:
 - régularisation: poids de la couche de sortie
 - généralisés: les poids de la couche de sortie et la position des centres

Sommaire – Section A.5

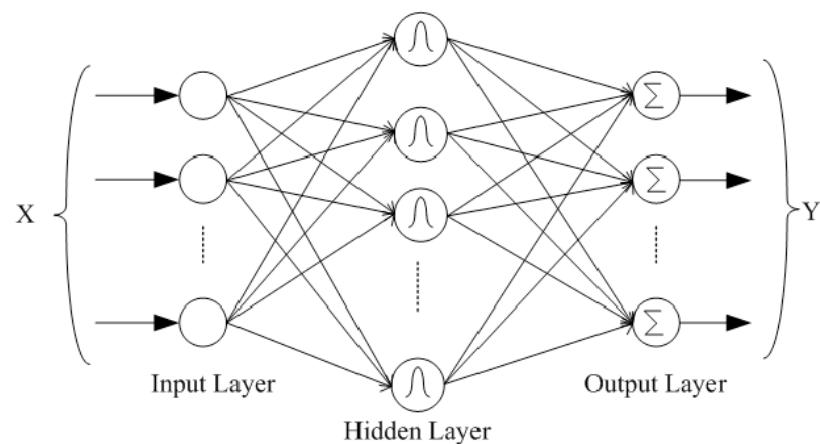
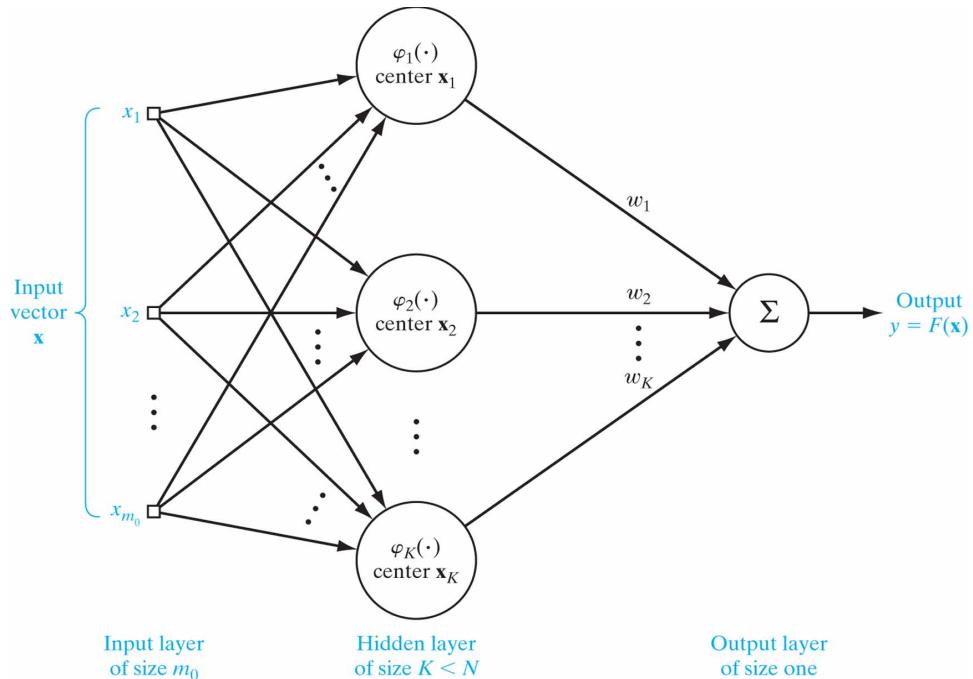
A.5 Réseaux de neurones à base de noyaux

1. Séparabilité des données
2. Problème d'interpolation
- 3. Réseaux classifieurs RBF**
4. Séparateurs à vaste marge (SVM)

3. Réseaux classifieurs RBF

Structure d'un réseau RBF classificateur

- couche cachée:** transformation non linéaire $\mathbf{x} \rightarrow \varphi(\mathbf{x})$
- couche de sortie:** transformation linéaire $\varphi(\mathbf{x}) \rightarrow \mathbf{y}$
(combinaison linéaire des fonctions cachées)



3. Réseaux classifiants RBF

Structure d'un réseau RBF classificateur

► L'activation des neurones:

- couche cachée: (par noyau j) $\varphi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \mu_j\|^2}{2\sigma_j^2}\right)$

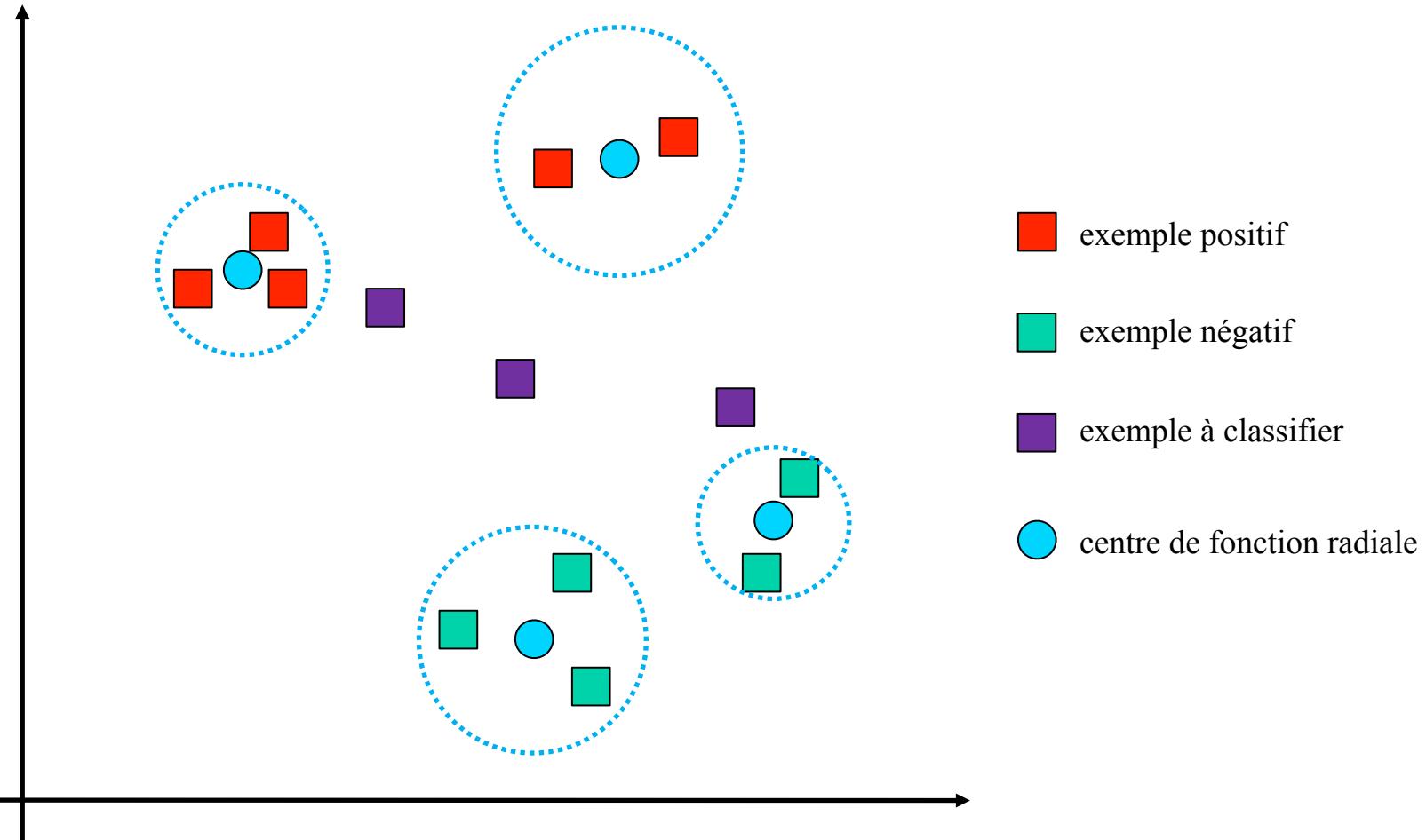
(fonction de forme radiale Gaussienne)

- couche de sortie: (par classe k) $y_k(\mathbf{x}) = \sum_{j=1}^M w_{kj} \bullet \varphi_j(\mathbf{x})$
- superposition linéaire de RBF capables d'approximation universelle

► Apprentissage: optimiser les centres μ_i , les dispersions σ_j et les poids w_{kj}

3. Réseaux classifiants RBF

► Illustration – problème à 2 classes en 2-dimensions



3. Réseaux classifieurs RBF

Stratégies d'apprentissage

- ▶ **les neurones cachées** – plusieurs différentes stratégies d'apprentissage en littérature selon la façon dont les centres sont déterminés:
 1. centres fixes, sélectionnés de façon aléatoire
 2. **centres obtenus par apprentissage non-supervisé**
 3. centres obtenus par apprentissage supervisé
- ▶ **les poids de sortie** – s'ajustent *rapidement* selon une stratégie d'optimisation supervisée linéaire
 - algorithme Least Mean Squared, etc.

3. Réseaux classifieurs RBF

1. centres fixes sélectionnés de façon aléatoire

- ▶ Choix aléatoire des centres à partir des patrons de TRAIN
- ▶ On utilise un RBF Gaussien:

$$G(\|x - \mu_j\|^2) = \exp\left(-\frac{m}{d_{\max}^2}\|x - \mu_j\|^2\right), \quad i = 1, 2, \dots, m$$

- RBF normalisé et centré à μ_j
- m : nombre de centres
- d_{\max} : distance maximum entre centres
- pour éviter des RBF trop étirés ou comprimés, l'écart type est fixé selon:

$$\sigma = \frac{d_{\max}}{\sqrt{2m}}$$

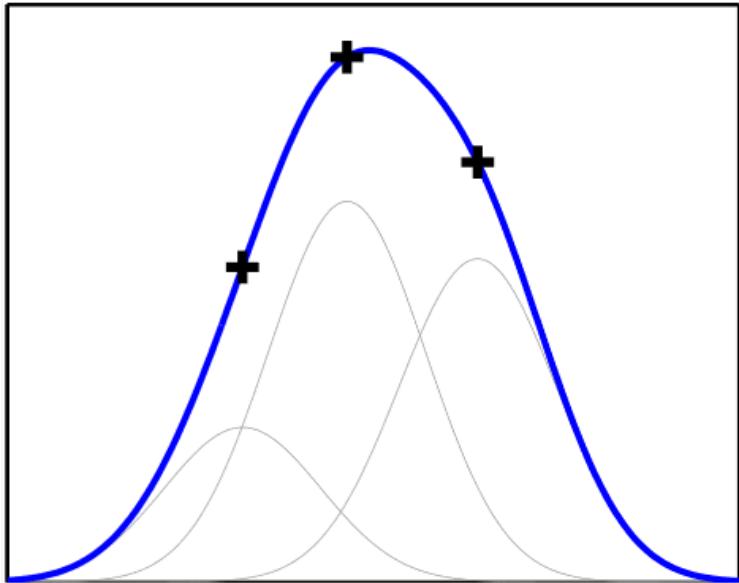
- ▶ On peut utiliser différents centres et écarts types suite à un analyse des données de TRAIN

3. Réseaux classifieurs RBF

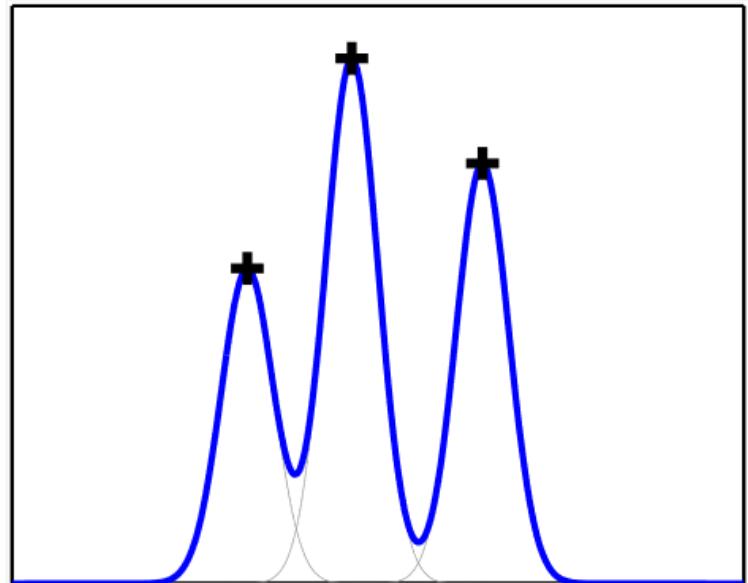
1. centres fixes sélectionnés de façon aléatoire

- ▶ Impact du choix de l'écart σ :

grand σ



petit σ



3. Réseaux classifieurs RBF

1. centres fixes sélectionnés de façon aléatoire

► Défis avec la méthode à centres fixés:

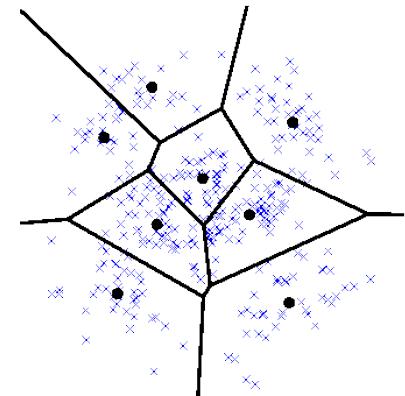
- justifiée seulement si TRAIN est distribué d'une façon représentative pour le problème
- complexité: peut nécessiter plusieurs neurones cachées, et un grand TRAIN pour obtenir des performances adéquates
.... donc beaucoup de paramètres, et de redondance.

3. Réseaux classifieurs RBF

2. centres obtenus par apprentissage non-supervisé

► Apprentissage hybride:

1. **couche cachée:** apprentissage non-supervisé des données pour estimer les centres des RBFs
 - *e.g.*, algorithme k -means ou GMM



2. **couche de sortie:** apprentissage supervisé pour estimer les poids linéaires
 - *e.g.*, algorithme LMS, règle delta (à base d'erreurs)

3. Réseaux classifieurs RBF

2. centres obtenus par apprentissage non-supervisé

► Algorithme classique *k-means*

Algorithme qui produit une *partition dure* selon le *critère du plus-proche-voisin*:

- **Objectif:** regrouper les patrons \mathbf{x}_i d'une ensemble $\mathbf{D}_n = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ dans un des K catégories, où chaque catégorie est représenté par une *catégorie* ou un centroïde $\boldsymbol{\mu}_k$, avec $k = 1, 2, \dots, K$
- **Étapes (processus itératif en 2 étapes):**
 1. **partitionnement:** assigner le centroïde $\boldsymbol{\mu}_k$ le plus proche à chacun des patrons \mathbf{x}_i
 2. **apprentissage:** mise-à-jour les centroïdes $\boldsymbol{\mu}_k (k=1, 2, \dots, K)$

- **Fonction de coût:** on minimise la somme des erreurs au carré

$$C = \sum_{i=1}^n \sum_{k=1}^K r_{ik} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|^2 = \sum_{i=1}^n \sum_{k=1}^K r_{ik} \left(\sqrt{(\mathbf{x}_i - \boldsymbol{\mu}_k)^2} \right)^2$$

3. Réseaux classifieurs RBF

2. centres obtenus par apprentissage non-supervisé

► **Exemple: apprentissage k -means par lot:**

- \mathbf{X}_k : ensemble de patrons \mathbf{x}_i assigné à la catégorie k
- N_k : nombre de patrons assigné à la catégorie k

1. **initialise** K centroïdes $\boldsymbol{\mu}_k$ ($k = 1, 2, \dots, K$) ou la partition de façon aléatoire
2. **partitionnement:** assigne la catégorie la plus *proche* à chaque patron \mathbf{x}_i
 → pour $i = 1, 2, \dots, n$ $k^* = \arg \min \left\{ \|\mathbf{x}_i - \boldsymbol{\mu}_k\| : k = 1, 2, \dots, K \right\}$
3. **apprentissage:** mise-à-jour du centroïde de chaque catégorie en optimisant \mathbf{C} par rapport aux prototypes $\boldsymbol{\mu}_k$
 → pour $k = 1, 2, \dots, K$ $\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{\mathbf{x}_i \in \mathbf{X}_k} \mathbf{x}_i$
4. **critère d'arrêt:** si la valeur des centroïdes $\boldsymbol{\mu}_k$ ou bien le coût change par rapport à la dernière itération, revenir à l'étape 2

3. Réseaux classifieurs RBF

2. centres obtenus par apprentissage non-supervisé

► **Exemple: apprentissage séquentielle k -means (en-ligne):**

1. **Initialisation** – choisir aléatoirement les centres
2. **Échantillonnage** – choisir un patron \mathbf{x} de TRAIN
3. **Calcul de proximité** – déterminer J , l'index du centre qui est le plus proche de \mathbf{x}

$$J = \arg \min \left\{ \|\mathbf{x}(n) - \boldsymbol{\mu}_j(n)\| : j = 1, 2, \dots, m \right\}$$

4. **Ajuster les centres** –

$$\boldsymbol{\mu}_j(n+1) = \begin{cases} \boldsymbol{\mu}_j(n) + \eta [\mathbf{x}(n) - \boldsymbol{\mu}_j(n)], & j = J \\ \boldsymbol{\mu}_j(n), & \text{sinon} \end{cases} \quad 0 < \eta < 1$$

retourner à l'étape 2

3. Réseaux classifieurs RBF

Calcul des poids (2^e couche)

- **Analytique:** On peut calculer la valeurs des poids qui minimisent l'erreur quadratique totale (MSE) à l'aide l'équation matricielle $\mathbf{w} = \underline{(\Phi^\top \Phi)^{-1} \Phi^\top \mathbf{y}}$

On utilise la pseudo inverse de la matrice rectangulaire Φ

$$\underbrace{\begin{bmatrix} \exp(-\gamma \|\mathbf{x}_1 - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_1 - \boldsymbol{\mu}_K\|^2) \\ \exp(-\gamma \|\mathbf{x}_2 - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_2 - \boldsymbol{\mu}_K\|^2) \\ \vdots & \vdots & \vdots \\ \exp(-\gamma \|\mathbf{x}_N - \boldsymbol{\mu}_1\|^2) & \dots & \exp(-\gamma \|\mathbf{x}_N - \boldsymbol{\mu}_K\|^2) \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{bmatrix}}_{\mathbf{w}} \approx \underbrace{\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix}}_{\mathbf{y}}$$

- **Apprentissage:** On peut utiliser des algorithmes qui minimisent l'erreur par descente de gradient

3. Réseaux classifieurs RBF

Calcul des poids (2^e couche)

- ▶ Algorithme *Least-Mean Squares* (LMS) pour la couche de sortie:

TABLE 3.1 Summary of the LMS Algorithm

Training Sample: Input signal vector = $\mathbf{x}(n)$
 Desired response = $d(n)$

User-selected parameter: η

Initialization. Set $\hat{\mathbf{w}}(0) = \mathbf{0}$.

Computation. For $n = 1, 2, \dots$, compute

$$e(n) = d(n) - \hat{\mathbf{w}}^T(n)\mathbf{x}(n)$$

$$\hat{\mathbf{w}}(n + 1) = \hat{\mathbf{w}}(n) + \eta \mathbf{x}(n)e(n)$$

3. Réseaux classifieurs RBF

3. centres sélectionnés par apprentissage supervisé

- ▶ Tous les paramètres libres sont ajustés par un processus d'apprentissage supervisé à base d'erreur
- ▶ Exploite des la descente de gradient, avec la fonction de coût:

$$E = \frac{1}{2} \sum_{j=1}^N e_j^2$$

et avec:

$$\begin{aligned} e_j &= d_j - F^*(x_j) \\ &= d_j - \sum_{i=1}^M w_i G(\|x_j - t_i\|_{C_i}) \end{aligned}$$

- ▶ Détermine tous les paramètres qui minimisent E .

3. Réseaux classifieurs RBF

3. centres sélectionnés par apprentissage supervisé

► **poids linéaires:** $\frac{\partial E(n)}{\partial w_i(n)} = \sum_{j=1}^N e_j(n)G(\|\mathbf{x}_j - t_i(n)\|)$

$$\mathbf{w}_i(n+1) = \mathbf{w}_i(n) - \eta_1 \frac{\partial E(n)}{\partial \mathbf{w}_i(n)}, \quad i = 1, 2, \dots, m_1$$

► **position des centres:**

$$\frac{\partial E(n)}{\partial t_i(n)} = 2w_i(n) \sum_{j=1}^N e_j(n)G'(\|\mathbf{x}_j - t_i(n)\|) \sum_i^{-1} [\mathbf{x}_j - t_i(n)]$$

$$\mathbf{t}_i(n+1) = \mathbf{t}_i(n) - \eta_2 \frac{\partial E(n)}{\partial \mathbf{t}_i(n)}, \quad i = 1, 2, \dots, m_1$$

► **dispersion des centres:**

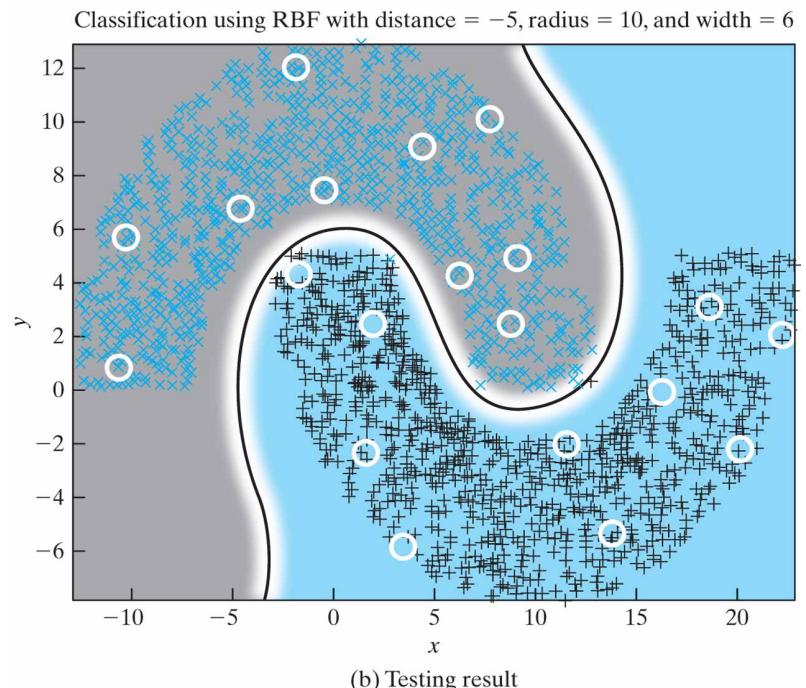
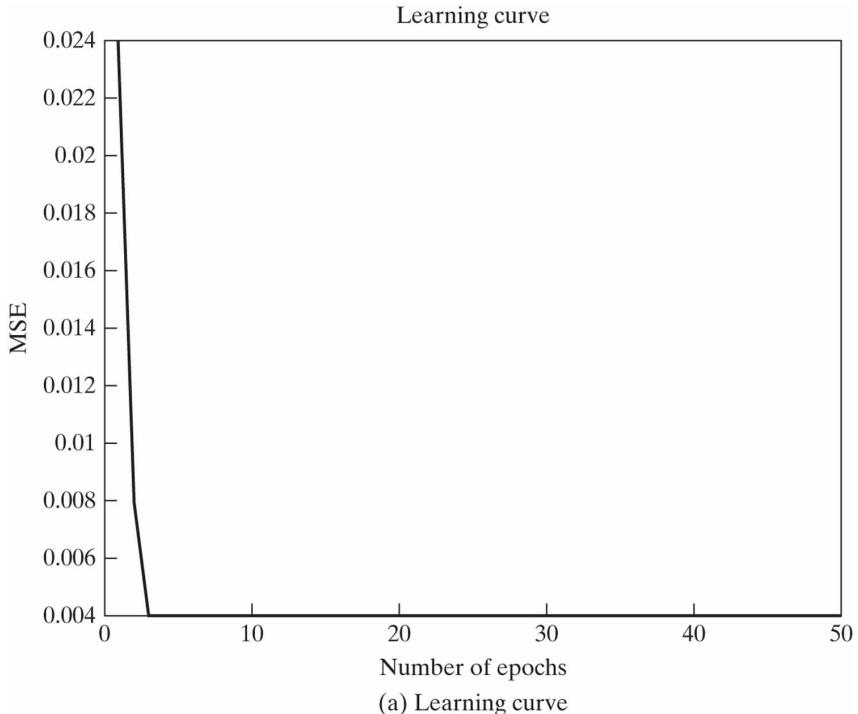
$$\frac{\partial E(n)}{\partial \sum_i^{-1}(n)} = -w_i(n) \sum_{j=1}^N e_j(n)G'(\|\mathbf{x}_j - t_i(n)\|) Q_{ji}(n)$$

$$\sum_i^{-1}(n+1) = \sum_i^{-1}(n) - \eta_3 \frac{\partial E(n)}{\partial \sum_i^{-1}(n)} \quad Q_{ji}(n) = [\mathbf{x}_j - \mathbf{t}_i(n)][\mathbf{x}_j - \mathbf{t}_i(n)]^T$$

3. Réseaux classifieurs RBF

Exemple

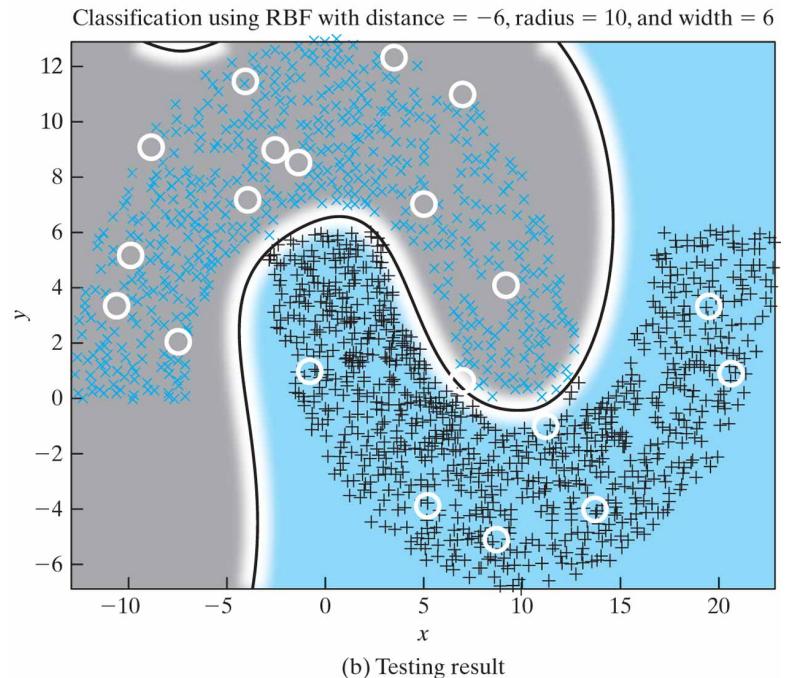
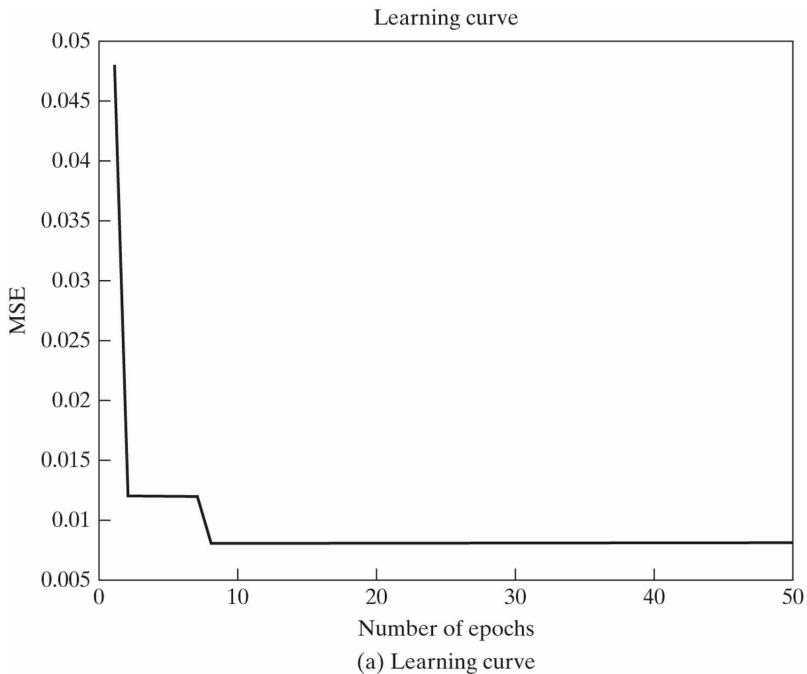
- ▶ Performance d'un réseau RBF entraînés avec k-means et LMS pour avec une distance $d = -5$, $M = 20$.



3. Réseaux classifieurs RBF

Exemple

- Performance d'un réseau RBF entraînés avec k-means et LMS pour avec une distance $d = -6$, $M = 20$.



3. Réseaux classifieurs RBF

Théorie de Bayes

► Problème de classification à k classes:

- sachant tous les statistiques sous-jacentes d'un problème de classification, la règle de décision de la probabilité a posteriori maximum (MAP) est optimale:

$$k^* = \arg \max \left\{ p(c_k | \mathbf{x}) : k = 1, 2, \dots, K \right\}$$

- selon le théorème de Bayes, on calcule la probabilité a posteriori comme:

$$p(c_k | \mathbf{x}) = \frac{p(\mathbf{x} | c_k)P(c_k)}{\sum_{h=1}^K p(\mathbf{x} | c_h)P(c_h)}$$

3. Réseaux classifieurs RBF

Théorie de Bayes

► Transposition dans le réseau RBF:

- si plusieurs RBF ($j = 1, 2, \dots, M$) sont utilisées pour représenter les classes, on peut remplacer dans l'équation de probabilité a posteriori:
 - les probabilités conditionnelles:

$$p(\mathbf{x} | c_k) = \sum_{j=1}^M p(\mathbf{x} | j) \cdot p(j | c_k)$$

- la probabilité non-conditionnelle:

$$p(\mathbf{x}) = \sum_{h=1}^K p(\mathbf{x} | c_h) P(c_h) = \sum_{j=1}^M p(\mathbf{x} | j) P(j)$$

3. Réseaux classifieurs RBF

Théorie de Bayes

► Transposition dans le réseau RBF:

- **couche cachée:** l'activation φ_j est interprétée comme la probabilité à posteriori de la présence du RBF j sachant \mathbf{x} :

$$\varphi_j(\mathbf{x}) = \frac{p(\mathbf{x} | j) p(j)}{\sum_{m=1}^M p(\mathbf{x} | j) p(j)} = p(j | \mathbf{x})$$

- **poids de la couche de sortie:** interprété comme la probabilité à posteriori d'appartenance à une classe étant donnée les RBFs

$$w_{kj} = \frac{p(j | c_k) P(c_k)}{p(j)} = p(c_k | j)$$

3. Réseaux classifieurs RBF

Théorie de Bayes

► Remarques:

- le réseau RBF classificateur est une réalisation parallèle du test d'hypothèse Bayesien
- chaque sorties du réseaux RBF est interprétée comme une probabilité a posteriori
- la distribution de chaque classe est modélisée comme un mélange de Gaussiennes (peut accommoder des classes multimodales, ou non-Gaussiennes)
- des RBF Gaussiennes permettent de régulariser
- s'adapte bien à la détection de nouveauté, détection d'ambiguïté, etc.

3. Réseaux classifieurs RBF

Comparaison MLP vs RBF

► Neurones de la couche cachée:

MLP: calculent une fonction non-linéaire du produit scalaire entre entrée et poids

- l'activation dépend de la somme pondérée avec entrées, et d'une fonction d'activation monotonique
- donc, l'activation est constante sur surfaces d'hyperplans

RBF: calculent une fonction non-linéaire de la distance entre entrée et centres

- activation dépend de la distance radiale entre entrée et centroïde, et une fonction d'activation locale
- donc, l'activation est constante sur hyper ellipses

3. Réseaux classifieurs RBF

Comparaison MLP vs RBF

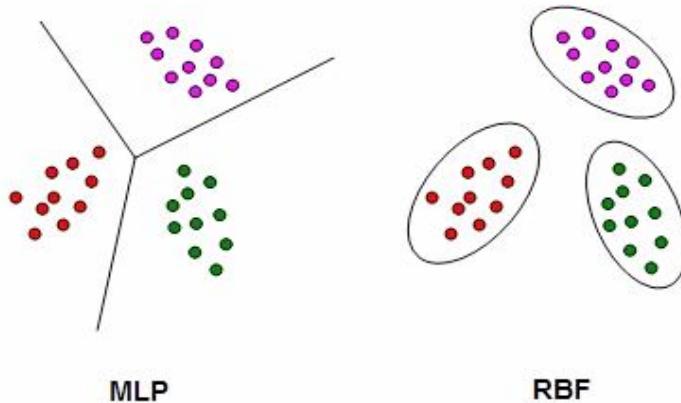
► Séparation des classes:

MLP: les neurones cachés forment des hyperplans dans l'espace d'entrée

- discriminatif – bornes de décision explicites

RBF: les neurones cachées représentent des RFB locales

- génératif – bornes des décision implicites



3. Réseaux classifieurs RBF

Comparaison MLP vs RBF

► **Représentation des connaissances dans l'espace des neurones cachées p/r à l'espace d'entrée:**

MLP: distribuée

- plusieurs neurones cachés vont s'activer pour contribuer à la sortie
- car optimisation globale de tous les poids donne une approximation globale

RBF: locale

- très peu de neurones cachés vont s'activer pour contribuer à la sortie
- une approximation locale

3. Réseaux classifieurs RBF

Comparaison MLP vs RBF

► Architecture neuronique:

MLP: peut avoir plusieurs couches cachées et des patrons complexes d'interconnexions

- tous les neurones partagent le même modèle neuronique
- couches cachées et de sorties non-linéaires

RBF: simple, consistant généralement d'une couche cachée

- la couche cachée est différente de la couche de sortie
- couche cachée non-linéaires et couche de sortie linéaires

3. Réseaux classifieurs RBF

Comparaison MLP vs RBF

► Processus d'apprentissage de paramètres:

MLP: tous les paramètres sont appris en même temps, via un processus supervisé global

- problème d'optimisation complexe qui peut converger lentement, et trouver des minimums locaux

RBF: les paramètres sont appris en deux étapes

- centres et dispersions sont apprises par apprentissage non-supervisé (sans effectuer une optimisation complexe)
- les poids w sont appris par apprentissage supervisé rapide (solution à un problème linéaire)

Sommaire – Section A.5

A.5 Réseaux de neurones à base de noyaux

1. Séparabilité des données
2. Problème d'interpolation
3. Réseaux classifieurs RBF
- 4. SVM: Séparateurs à vaste marge**

4. Séparateurs à vaste marge

‘**Support Vector Machines**’: (Vapnik, 1992) reconnu comme un des meilleurs classificateurs statistiques

- classificateur binaire à 2 classes
- **plusieurs applications pratiques:** détection de visage, reconnaissance de caractères, etc.
- un sujet de recherche intensif depuis~2001

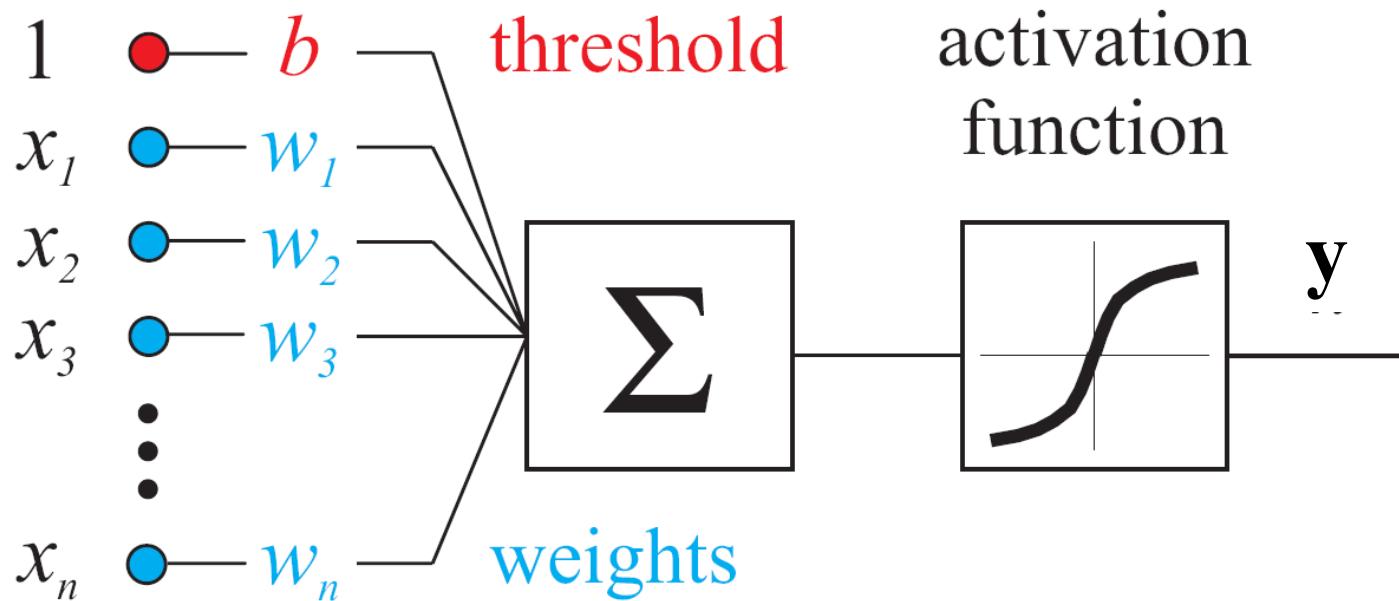
Variantes pertinentes:

- a. **SVM linéaires (LSVM)** – approche discriminative
- b. **SVM non-linéaires (méthode à noyau)** – approche hybride discriminative-générative

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

- ▶ **LSVM** – un classificateur linéaire à 2 classes (dichotomie):



4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

► Problème de conception:

- soit l'ensemble de données pour l'apprentissage:

$$\mathbf{D}_n = \{z_1, z_2, \dots, z_n\}$$

$$= \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)\} \in \Re^d \times \{-1, 1\}$$

- l'objectif est de trouver un hyperplan:

$$\mathbf{w}\mathbf{x} + b, \quad \mathbf{w} \in \Re^d, \quad b \in \Re$$

qui sépare les données des deux classes

4. Séparateurs à vaste marge

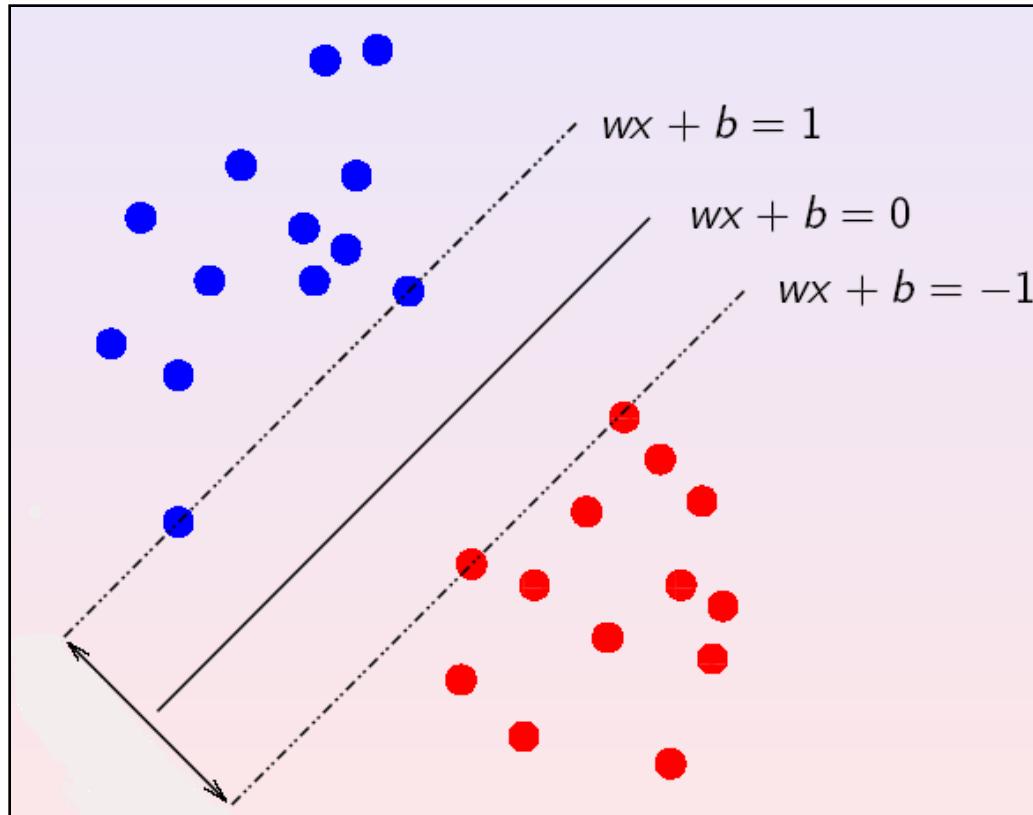
(a) SVM linéaires – cas séparable

- ▶ **Approche SVM:** déterminer l'hyperplan séparateur qui maximise *marge M* entre patrons des deux classes
- ▶ La *marge de l'hyperplan*, $M = d_+ + d_-$, avec
 - d_+ ≡ la distance la plus courte entre cet hyperplan et le patron *positif* (classe +1) le plus proche
 - d_- ≡ la distance la plus courte entre cet hyperplan et le patron *négatif* (classe -1) le plus proche

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

- Exemple: un cas de données linéairement séparables



4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

► Calcul de la marge M :

- on peut définir les contraintes d'optimisation suivantes:

$$\mathbf{w}\mathbf{x}_i + b \geq +1, \text{ pour } y_i = +1$$

$$\mathbf{w}\mathbf{x}_i + b \leq -1, \text{ pour } y_i = -1$$

qu'on peut combiner comme suit :

$$c_i = y_i (\mathbf{w}\mathbf{x}_i + b) - 1 \geq 0, \text{ pour } i = 1, 2, \dots, n$$

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

► Calcul de la marge M (suite)

- on peut démontrer que $d_+ = d_- = 1 / \|\mathbf{w}\|$, où $\|\mathbf{w}\|$ est la norme Euclidienne de \mathbf{w}
- la marge de l'hyperplan devient donc:

$$M = d_+ + d_- = \frac{2}{\sqrt{\mathbf{w} \cdot \mathbf{w}}} = \frac{2}{\|\mathbf{w}\|}$$

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

► Problème d'optimisation sous contraintes:

- **Apprentissage** → recherche d'un hyperplan (dans l'espace des \mathbf{w} et b) avec la *marge maximum*, permettant de classifier tous les patrons dans \mathbf{D}_n
- le problème consiste alors à *minimiser* la fonction de coût:

$$L(\mathbf{w}) = \frac{\|\mathbf{w}\|^2}{2}, \text{ sujet aux contraintes :}$$

$$c_i = y_i (\mathbf{w} \mathbf{x}_i + b) - 1 \geq 0, \text{ pour } i = 1, 2, \dots, n$$

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

► **Problème d'optimisation avec contraintes:** (suite)

- pour résoudre un problème d'optimisation avec coût $L(\mathbf{w})$ et paramètre \mathbf{w} , on peut fixer $\partial L(w) / \partial \mathbf{w} = 0$

Exemple (1D): minimiser $L(w) = \frac{w^2}{2} - 3w$

alors : $\frac{\partial L(w)}{\partial w} = w - 3 = 0 \Rightarrow w = 3$

- **mais**, lorsqu'il y a des contraintes $c_i \geq 0$, on utilise les *multiplicateurs Lagrangiens*, et on vérifie notre solution avec les *conditions Karush-Kuhn-Tucker (KKT)*

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

► Problème d'optimisation avec contraintes: (suite)

- **Lagrangien** – fonction de coût formée en soustrayant un terme pour chaque contrainte $c_i \geq 0$, pondéré par un multiplicateur Lagrangien positif: $L(\mathbf{w}, \alpha) = L(\mathbf{w}) - \sum_i \alpha_i c_i$
- on peut alors résoudre le *problème dual* – maximiser $L(\mathbf{w}, \alpha)$ en fonction de α , sujet au contraintes:

$$\frac{\partial L(\mathbf{w}, \alpha)}{\partial \mathbf{w}} = 0 \text{ et } \alpha_i \geq 0, \text{ pour } i = 1, 2, \dots, n$$

- le problème général consiste donc à trouver la solution:

$$\max_{\alpha} \min_{\mathbf{w}} L(\mathbf{w}, \alpha)$$

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

► **Problème d'optimisation avec contraintes:** (suite)

- on introduit un multiplicateur Lagrangien α_i ($i = 1, 2, \dots, n$), pour chaque contrainte d'inégalité:

$$L(\mathbf{w}, b, \alpha) = \frac{\|\mathbf{w}\|^2}{2} - \sum_{i=1}^n \alpha_i (y_i (\mathbf{w} \cdot \mathbf{x}_i + b) - 1)$$

- $L(\mathbf{w}, b, \alpha)$ doit être (1) minimisé p/r aux variables primaires (\mathbf{w} et b), tout en (2) maximisant p/r aux variables duales (α_i)
- aux extreums, nous avons:

$$\frac{\partial L(\mathbf{w}, b, \alpha)}{\partial \mathbf{w}} = 0 \text{ et } \frac{\partial L(\mathbf{w}, b, \alpha)}{\partial b} = 0$$

4. Séparateurs à vaste marge

(a) SVM linéaires – cas séparable

- ▶ La formulation duale – on cherche à *maximiser*:

$$L = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

sujet aux contraintes linéaires :

$$\alpha_i \geq 0 \quad \forall i \quad \text{et} \quad \sum_{i=1}^n \alpha_i y_i = 0$$

- on peut résoudre avec des techniques d'optimisation classiques en **programmation quadratique** (basées sur, *e.g.*, l'ascente de gradient avec contraintes)

4. Séparateurs à vaste marge

(a) SVM linéaires – cas non-séparable

► **Solution – *marge molle* ('soft margin')** pour modéliser le chevauchement ou le bruit:

- $\xi_i \equiv$ distance entre patron erroné \mathbf{x}_i et la droite définie par les SV de sa classe
- contraintes pour l'optimisation:

$$\mathbf{w}\mathbf{x}_i + b \geq +1 - \xi_i, \quad \text{pour } y_i = +1$$

$$\mathbf{w}\mathbf{x}_i + b \leq -1 + \xi_i, \quad \text{pour } y_i = -1$$

4. Séparateurs à vaste marge

(a) SVM linéaires – cas non-séparable

► **Solution** – marge *molle* pour modéliser le chevauchement ou le bruit:

- on repose les contraintes p/r à une marge dur
- on cherche alors à *minimiser* le critère d'optimisation quadratique suivant:

$$\frac{\|\mathbf{w}\|^2}{2} + C \sum_{i=1}^n \xi_i$$

sujet aux contraintes:

$$c_i = y_i(\mathbf{w}\mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad \forall i$$

4. Séparateurs à vaste marge

(a) SVM linéaires – cas non-séparable

- ▶ La formulation duale – on cherche à *maximiser*:

$$L = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

sujet aux contraintes :

$$0 \leq \alpha_i \leq C, \quad \forall i \quad \text{et} \quad \sum_{i=1}^n \alpha_i y_i = 0$$

- on obtient \mathbf{w} et b selon: $\mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$

$$\alpha_i [1 - \xi_i - y_i (\mathbf{w} \mathbf{x}_i + b)] = 0$$

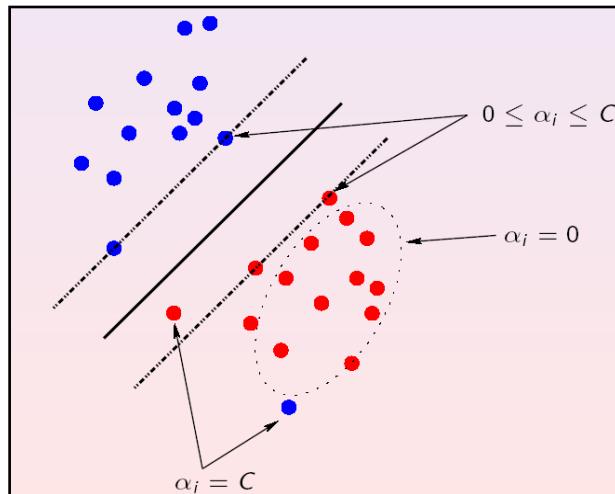
4. Séparateurs à vaste marge

(a) SVM linéaires – cas non-séparable

► Fonction de décision:

$$\hat{y} = \text{sign}(\mathbf{w}\mathbf{x} + b) = \text{sign}\left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \mathbf{x} + b\right)$$

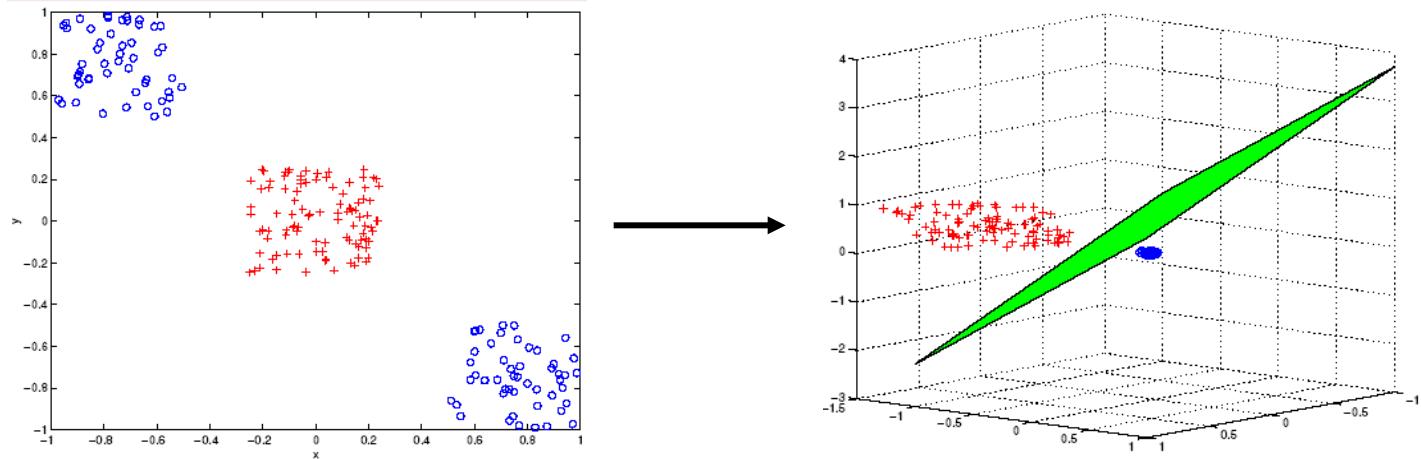
► Vecteurs de support: patrons d'entraînement \mathbf{x}_i de D_n ($i = 1, 2, \dots, n$) avec $\alpha_i \neq 0$



4. Séparateurs à vaste marge

(b) SVM non-linéaires

- **Solution:** projeter les patrons dans un espace de plus grande dimensionnalité avec une transformation non-linéaire
- cet espace devrait séparer les deux classes plus facilement
 - étant donné une fonction $\varphi: \Re^d \rightarrow F$, travailler avec une *l'espace image du patron* $\varphi(\mathbf{x}_i)$ au lieu de celle du patron \mathbf{x}_i



4. Séparateurs à vaste marge

(b) SVM non-linéaires

- ▶ Avec un SVM, on doit calculer les produits $\varphi(\mathbf{x}_i)\varphi(\mathbf{x}_j)$ pour passer à l'espace image
 - cependant, ce calcul peut être très coûteux dans un espace de grande dimensionnalité
- ▶ Fonction noyaux: on utilise plutôt une fonction noyau $k(\mathbf{x}_i, \mathbf{x}_j)$, qui représente un produit dans un espace image
$$k(\mathbf{x}_i, \mathbf{x}_j) = \varphi(\mathbf{x}_i)\varphi(\mathbf{x}_j)$$

4. Séparateurs à vaste marge

(b) SVM non-linéaires

► **Fonction noyaux communs:**

- Polynomial:

$$k(\mathbf{x}_i, \mathbf{x}_j) = (u \mathbf{x}_i \mathbf{x}_j + v)^p \quad u, v \in \Re, \quad p \in \mathbb{N}_+^*$$

- Gaussien ou ‘Radial Basis Function’ (RBF):

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp \left\{ -\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2} \right\}, \quad \sigma \in \Re_+^*$$

4. Séparateurs à vaste marge

(b) SVM non-linéaires

- ▶ La formulation duale – on cherche à *maximiser*

$$L = \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$$

sujet aux contraintes:

$$0 \leq \alpha_i \leq C \quad \text{et} \quad \sum_i \alpha_i y_i = 0$$

4. Séparateurs à vaste marge

(b) SVM non-linéaires

► **La formulation duale:**

- on détermine les variables primaires \mathbf{w} et b avec:

$$\mathbf{w} = \sum_{i=1}^n \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}_j)$$

$$1 - y_i \left[\sum_j \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j) + b \right] = 0$$

pour: $0 < \alpha_i < C$

4. Séparateurs à vaste marge

(b) SVM non-linéaires

► Mode opérationnel:

- fonction de décision:

$$\hat{y} = \text{sign} \left[\sum_i \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b \right]$$

4. Séparateurs à vaste marge

(b) SVM non-linéaires

► Synthèse des propriétés:

- **objectif:** conçu pour maximiser la marge dans l'espace des partons
- **méthode Lagrangienne:** permet de formuler l'apprentissage comme un problème d'optimisation quadratique (sous contraintes)
- **populations non linéairement séparables:** utilise une marge molle et/ou une fonction noyau
- **relations non-linéaires:** projeter les patrons dans une espace de haute dimensionnalité
- **fonctions noyaux:** permettent de simplifier le calcul

4. Séparateurs à vaste marge

(b) SVM non-linéaires

► Considérations pratiques:

- pour optimiser la capacité du modèle: le choix du noyau est le paramètre le plus important
 - **noyau polynomial:** si on augmente le degré du polynôme, on augmente la capacité
 - **noyau Gaussien:** si on augmente la variance, on diminue la capacité