Réseaux à fonctions de base radiales

Emmanuel Viennet

Chapitre 4

Réseaux à fonctions de base radiales

4.1. Présentation

Les réseaux à fonctions de base radiales (RBF) sont des modèles connexionnistes simples à mettre et œuvre et assez intelligibles, et sont très utilisés pour la régression et la discrimination. Leur propriétés théoriques et pratiques ont été étudiées en détail depuis la fin des années 80 ; il s'agit certainement, avec le Perceptron multicouche, du modèle connexionniste le mieux connu.

Une fonction de base radiale (RBF) est une fonction ϕ symétrique autour d'un centre $\boldsymbol{\mu}_j$: $\phi_j(\mathbf{x}) = \phi(\|\mathbf{x} - \boldsymbol{\mu}_\mathbf{j}\|)$, où $\|.\|$ est une norme [BUH 03]. Par exemple, la fonction gaussienne est une RBF avec la norme euclidienne et $\phi(r) = e^{-r^2/2\sigma^2}$. En général, les RBF sont paramétrées par σ qui correspond à la « largeur » de la fonction :

$$\phi_j(\mathbf{x}) = \phi(||\mathbf{x} - \boldsymbol{\mu}_\mathbf{j}||, \sigma_\mathbf{j})$$

Un modèle ou *réseau* RBF calcule une combinaison linéaire de fonctions radiales de centres $\boldsymbol{\mu}_j$:

$$y(\mathbf{x}) = \sum_{\mathbf{j=1}}^{\mathbf{N}} \mathbf{w_j} \phi(||\mathbf{x} - \boldsymbol{\mu}_\mathbf{j}||, \sigma_\mathbf{j}) \qquad [4.1]$$

Chapitre rédigé par Emmanuel VIENNET.

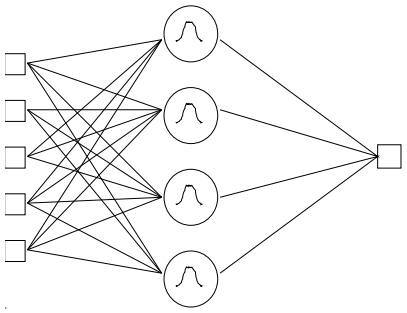

Figure 4.1. *Représentation connexionniste d'un réseau RBF. A gauche la couche d'entrée* $\mathbf{x}$*, au centre les centres RBF, à droite la sortie* y*. Pour traiter les problèmes de discrimination à* C *classes, on pourra utiliser* C *sorties* $y_1, \ldots, y_C$

On distingue trois couches (figure 4.1) : entrée $\mathbf{x}$, fonctions radiales, sortie, et trois jeux de paramètres : les centres $\boldsymbol{\mu}_j$, les largeurs σ_j et les poids w_j.

Les combinaisons linéaires de gaussiennes sont utilisées depuis les années 60 pour construire des interpolations ou approximations de fonctions [POW 87]. A la fin des années 80, la présentation de ces modèles comme des *réseaux connexionnistes* a suscité un regain d'intérêt [BRO 88, MOO 89] motivé en grande partie par la possibilité d'utiliser un algorithme d'apprentissage très rapide (sans recourir à des techniques d'optimisation non linéaire comme dans le cas du Perceptron multicouche), donnant en général des résultats voisins des meilleurs modèles connexionnistes.

L'apprentissage des modèles RBF est *supervisé* : il faut disposer d'un échantillon de l exemples (x_i, y_i). Comme les Perceptrons multicouches (MLP [1]), les RBF sont utilisés pour résoudre tant des tâches de discrimination (en général en choisissant $y_i \in \{-1, 1\}^C$) que des tâches de régression ou prévision de signal (monovarié $y_i \in \mathbb{R}$ ou multivarié $y_i \in \mathbb{R}^C$).

Les modèles RBF sont liés à de nombreuses autres approches utilisées en reconnaissance des formes ; les relations avec l'étude de l'approximation de fonctions (par exemple les splines [BIS 95]) sont évidentes. Mentionnons aussi les liens avec la théorie de la régularisation, l'interpolation, l'estimation de densité.

1. PMC pour les francophones, mais nous utiliserons l'abbréviation anglophone plus répandue dans la littérature.

4.2. Le problème d'approximation

Le problème général de de l'approximation [POG 90] d'une fonction multivariée se pose de la façon suivante : soit $f(\mathbf{x})$ une fonction continue de $\mathbb{R}^d \rightarrow \mathbb{R}$, et $F(\mathbf{w}, \mathbf{x})$ une fonction approximation dépendant continûment de $\mathbf{w} \in \mathbb{R}^{\mathbf{P}}$ et $\mathbf{x}$. On cherche le jeu de paramètres $\mathbf{w}^*$ tel que :

$$\forall \mathbf{w}, \;\; \mathbf{d}\,(\,\mathbf{F}(\mathbf{w}^*, .),\; \mathbf{f}(.)\,) \leq \;\mathbf{d}\,(\,\mathbf{F}(\mathbf{w}, .),\; \mathbf{f}(.)\,)$$

où d est une distance dans l'espace des fonctions.

Dans ce contexte, le problème de l'apprentissage consiste à déterminer les paramètres d'une fonction d'approximation étant donné un ensemble fini de l points $(\mathbf{x_i}, \mathbf{f}(\mathbf{x_i}))$, ce qui revient à déterminer une hyper-surface passant le plus près possible des points donnés (exemples d'apprentissage). Mais dans le même temps, on cherche à obtenir une solution permettant une bonne *généralisation*, c'est-à-dire une estimation correcte de la fonction dans les zones où l'on ne dispose pas d'exemple. Pour cela, l'hyper-surface doit être la plus régulière possible. Durant l'apprentissage, on va donc minimiser un coût formé de deux termes [TIK 77] ; le premier exprime la qualité de l'interpolation (mesurée sur les exemples), tandis que le second pénalise les surfaces irrégulières. Ce terme s'exprime en général par l'intermédiaire de la norme d'un opérateur différentiel P :

$$\text{coût} = \sum_i (y_i - F(\mathbf{w}, \mathbf{x_i}))^{\mathbf{2}} \;+\; \lambda \| \,\mathbf{PF}\, \|^{\mathbf{2}} \qquad [4.2]$$

où λ est un paramètre de régularisation, souvent nommé *hyperparamètre de régularisation*. Typiquement, on cherche à minimiser la courbure de la solution en utilisant un opérateur laplacien [BIS 90]. L'existence et la qualité de la solution au problème d'approximation dépendent de la classe de fonctions à laquelle $F(\mathbf{w}, \mathbf{x})$ appartient [RIC 64]. Il existe de nombreuses classes de fonctions approximantes, comme par exemple les fonctions linéaires $F(\mathbf{w}, \mathbf{x}) = \mathbf{w}.\mathbf{x}$ et les Perceptrons multicouches :

$$F(\mathbf{w}, \mathbf{x}) = \sigma\left(\sum_{\mathbf{n}} \mathbf{w_n}\, \sigma\left(\sum_{\mathbf{i}} \mathbf{w_i}\, \sigma\left(\cdots\, \sigma\left(\sum_{\mathbf{j}} \mathbf{w_j x_j}\right)\, \cdots\right)\right)\right) \qquad [4.3]$$

où σ est la fonction sigmoïde (voir chapitre 3).

Un autre type d'approximation très utilisé repose sur l'emploi d'une base de fonctions Φ_i :

$$F(\mathbf{w}, \mathbf{x}) = \sum_{\mathbf{i}} \mathbf{w_i}\, \mathbf{\Phi_i}(\mathbf{x}) \qquad [4.4]$$

Les interpolations par splines, les expansions en séries de polynômes orthogonaux et modèles additifs généralisés [HAS 90] entrent, parmi d'autres, dans ce cadre. Un grand intérêt de cette représentation est que l'on dispose d'un approximant *non linéaire* en ayant à résoudre un problème linéaire (choix de w_i, les fonctions Φ_i étant déterminées). On peut facilement montrer [POG 90] que si l'opérateur de régularisation est invariant par translation et par rotation (hypothèses en général très raisonnables), la solution du problème de minimisation du coût 4.2 est nécessairement de la forme 4.4, $\Phi_i(\mathbf{x})$ étant une fonction *radiale*, ne dépendant que de la distance de $\mathbf{x}$ à un centre μ_i (équation 4.1). C'est pourquoi les réseaux RBF sont parfois appelés « réseaux régularisants » (*regularization networks*).

On peut établir pour ce type d'approximant le même type de résultats que pour les Perceptrons multicouches (voir chapitre 3) : toute fonction continue peut être approchée avec une précision arbitraire [PAR 91, POG 90].

Notons enfin que l'équation 4.1 est de la même forme qu'un estimateur à noyau comme celui de Parzen.

4.3. Apprentissage des modèles RBF

L'apprentissage d'un modèle RBF consiste à déterminer son architecture (le nombre N de fonctions radiales) et à fixer les valeurs des paramètres. La plupart des utilisateurs déterminent empiriquement la valeur de N en recourant à des techniques de validation croisée (voir chapitre 12).

L'apprentissage d'un réseau RBF est de type *supervisé* : on dispose d'un ensemble d'apprentissage constitué de l couples (vecteur d'entrée, valeur cible) :

$$(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_m}, \mathbf{y_m}), \quad \mathbf{x_i} \in \mathbb{R}^{\mathbf{d}}, \mathbf{y_i} \in \mathbb{R}$$

et du coût associé à chaque exemple :

$$E_i = \frac{1}{2}(y_i - F(\mathbf{x_i}))^{\mathbf{2}}$$

(auquel on ajoute éventuellement un terme de régularisation).

Une caractéristique intéressante des modèles RBF est que l'on peut diviser les paramètres en trois groupes : les centres $\boldsymbol{\mu}$, les largeurs σ et les poids w. L'interprétation de chaque groupe permet de proposer un algorithme d'apprentissage séquentiel, simple et performant [MOO 89].

4.3.1. *Approche séquentielle*

Cette technique d'apprentissage proposée dès la fin des années 1980 [MOO 89] est très couramment utilisée. Elle consiste à optimiser successivement les trois jeux de paramètres $(\boldsymbol{\mu}_j, \sigma_j, w_j)$. Cette technique a l'avantage d'être simple à mettre en œuvre, de demander peu de calculs et de donner des résultats acceptables. La solution obtenue n'est cependant pas optimale.

Dans un premier temps, on estime les positions des centres $\boldsymbol{\mu}_j$ et des largeurs σ_j à l'aide d'un algorithme *non supervisé* de type k-moyennes. Une fois ces paramètres fixés, il est possible de calculer les poids w_j optimaux par une méthode de régression linéaire. C'est certainement la simplicité et l'efficacité de cette méthode qui a fait le succès des RBF.

4.3.1.1. *Calcul des poids*

Si l'on suppose les centres et largeurs connus, les poids $\mathbf{w}$ optimaux se calculent aisément :

$$y(\mathbf{x}) = \sum_{\mathbf{j=1}}^{\mathbf{N}} \mathbf{w_j} \phi(||\mathbf{x} - \boldsymbol{\mu}_\mathbf{j}||, \sigma_\mathbf{j}) = \sum_{\mathbf{j=1}}^{\mathbf{N}} \mathbf{w_j h_j}(\mathbf{x})$$

On cherche la solution $\mathbf{w}$ qui minimise la différence e entre la sortie estimée et la sortie désirée. On a donc un système d'équations linéaires qui s'écrit :

$$\mathbf{y} = \mathbf{H}\,\mathbf{w} + \mathbf{e}$$

La matrice $\mathbf{H}$, de taille $l \times N$, donne les réponses des N centres RBF sur les l exemples, $\mathbf{y}$ est un vecteur regroupant les l sorties y_i sur l'ensemble d'apprentissage, et $\mathbf{e}$ est le vecteur d'erreur. Le critère à optimiser est :

$$E = \mathbf{e}^\mathsf{T}\,\mathbf{e}$$

Si l'on ajoute un terme de régularisation de type *ridge regression* [HOE 62], qui pénalise les solutions avec de grandes valeurs des poids, on écrit :

$$E = \mathbf{e}^\mathsf{T}\,\mathbf{e} + \lambda\,\mathbf{w}^\mathsf{T}\,\mathbf{w}$$

La solution s'obtient par un calcul classique de pseudo-inverse, et s'écrit :

$$\mathbf{w} = (\mathbf{H}^\mathsf{T}\,\mathbf{H} \,+\, \lambda\,\mathbf{I})^{-\mathbf{1}}\,\mathbf{H}^\mathsf{T}\,\mathbf{y}$$

où $\mathbf{I}$ est la matrice identité de taille l.

La régression de type *ridge* est très utilisée en apprentissage statistique. Dans le contexte des réseaux connexionnistes (par exemple les Perceptrons multicouches), on l'appelle souvent *weight decay*. Le paramètre λ est libre et doit être déterminé par validation croisée ou, de manière plus sophistiquée, en employant des méthodes bayésiennes de réestimation, voir [ORR 95]).

En pratique, il est recommandé de résoudre le système d'équation en utilisant une décomposition en valeurs singulières (SVD), qui résiste bien aux problèmes de mauvais conditionnement numérique.

4.3.1.2. *Estimation non supervisée des centres et des largeurs*

Afin de déterminer les positions et largeurs des centres gaussiens, on les interprète comme représentant la densité de probabilité des données et on cherche une solution *locale* (chaque fonction va s'activer dans une « petite » région de l'espace d'entrée). On désire qu'au moins un centre soit activé, c'est-à-dire que la valeur de la fonction radiale soit non négligeable, dans toutes les régions où l'on a des données. La dimension de l'ensemble des points associés à un centre va permettre d'estimer la largeur de ce centre.

Ce point de vue suggère d'utiliser une approche non supervisée, qui estime la densité de probabilité des données. Dans ce cadre, les valeurs cibles y_i ne vont pas être utilisées pour l'estimation des centres $\boldsymbol{\mu}_j$ et des largeurs σ_j. Notons que cela peut dans certains cas être un avantage : il est courant que les données non étiquetées soit beaucoup plus faciles à obtenir en grande quantité que les données étiquetées. Par exemple, en reconnaissance d'images de visages, il est facile de collecter des images de visages quelconques, mais plus coûteux de réunir un ensemble de visages d'identité connue. Dans ces situations, les approches basées sur l'estimation directe de la densité de probabilité des exemples sont toujours intéressantes (apprentissage semi-supervisé).

Le problème principal que rencontrent les modèles RBF est lié à leur comportement lorsque la dimension de l'espace d'entrée augmente (« malédiction de la dimension »). Si l'on veut couvrir l'espace d'entrée avec des sphères placées sur les centres RBF, le nombre de sphères nécessaires augmente exponentiellement avec la dimension d des entrées, affectant non seulement les temps de calcul mais aussi augmentant proportionnellement le nombre d'exemples requis pour l'estimation correcte des paramètres.

Un autre problème, lié au précédent, est la sensibilité au bruit : puisque l'estimation des centres et largeurs est faite de manière non supervisée, il n'est pas possible de distinguer les variables corrélées à la valeur cible de celles qui n'apportent que du bruit.

Pour ces raisons, on observe facilement en pratique que les performances des modèles RBF se dégradent rapidement lorsque la dimension des entrées augmente. Il

est alors nécessaire de faire précéder le système RBF par une phase de réduction de dimension (sélection de variables supervisée ou non).

Avant de détailler plus la procédure d'apprentissage séquentielle, insistons sur son caractère sous-optimal : même si les résultats observés en pratique sont généralement corrects, il est facile d'imaginer des situations dans lesquelles la densité de probabilité des données diffère beaucoup de la valeur cible à estimer. La figure 4.2, empruntée à [BIS 95] illustre parfaitement ce cas.

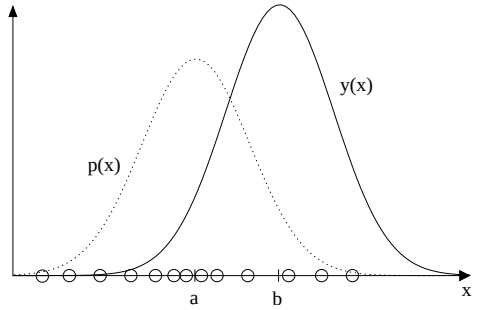

Figure 4.2. *Exemple simple d'un cas dans lequel la densité de probabilité $p(x)$ des données (représentées par les ronds sur l'axe horizontal) ne coïncide pas avec la fonction cible $y(x)$. L'apprentissage séquentiel va centrer la fonction au point a, alors que la valeur optimale est b*

Utilisation de tous les exemples

L'approche la plus simple pour choisir les centres RBF $\boldsymbol{\mu}_i$ est de retenir tous les exemples disponibles. Cette approche, qui rappelle celle employée pour le système de discrimination « plus proche voisin », est rarement utilisée car elle possède deux désavantages majeurs : lenteur d'exécution dès que l'on dispose d'un nombre significatif d'exemples, et surtout nombre beaucoup trop important de paramètres à estimer, donc sur-apprentissage garanti. Elle peut cependant constituer la première étape pour des approches à base d'élagage, consistant à supprimer progressivement les centres les moins utiles.

Utilisation d'une méthode de clustering

La sélection des centres la plus utilisée fait appel à un algorithme de *clustering*, type k-moyennes, permettant de calculer rapidement k vecteurs minimisant l'erreur de quantification (ou erreur empirique) sur l'ensemble d'apprentissage :

$$E(\boldsymbol{\mu}) = \frac{1}{2} \sum_i \min_k (x_i - \mu_k)^2$$

Le nombre k de centres à rechercher doit être spécifié *a priori*, c'est le principal inconvénient de cette méthode en pratique (en général, on utilise là aussi des méthodes de validation croisée pour déterminer le nombre optimal). Notons qu'ici les centres ne coïncident plus nécessairement avec des exemples de l'ensemble d'apprentissage. La méthode des k-moyennes est ancienne [MAC 67] mais a fait l'objet de nombreuses études et raffinements (voir par exemple [BOT 95]).

Une autre approche [ORR 95] consiste à rechercher incrémentalement les centres RBF parmi les exemples, en recherchant à chaque étape le centre susceptible d'apporter la plus grande diminution de l'erreur. Cette approche peut être combinée avec une régularisation de type *ridge*.

Détermination des largeurs

Une fois les centres RBF placés, reste à déterminer les valeurs de largeurs (paramètres σ). On utilise en général une heuristique, basée soit sur la distance de chaque centre à son voisin, soit sur le calcul de la variance de l'ensemble d'exemples rattachés à un centre (ceux pour lesquels ce centre est le plus proche).

4.3.2. *Apprentissage par descente de gradient*

Une alternative à l'apprentissage séquentiel décrit dans la section précédente consiste à optimiser les paramètres du modèles RBF par descente de gradient, comme on le fait pour d'autres modèles connexionnistes. Il faut pour cela calculer les dérivées du coût (éventuellement régularisé) par rapport aux différents paramètres.

Pour une fonction gaussienne :

$$\Phi_{ij} = \exp(-\frac{\|\mathbf{x_i} - \boldsymbol{\mu_j}\|^\mathbf{2}}{2\sigma_j^2})$$

et un coût $E_i = \frac{1}{2}(y_i - y(x_i))^2$, les dérivées partielles s'écrivent :

$$\frac{\partial E_i}{\partial w_j} = -w_j(y_i - \sum_j w_j \Phi_{ij})\Phi_{ij}$$

$$\frac{\partial E_i}{\partial \sigma_j} = -w_j \frac{\|\mathbf{x} - \boldsymbol{\mu_j}\|^\mathbf{2}}{\sigma_j^3}(y_i - \sum_j w_j \Phi_{ij})\Phi_{ij}$$

$$\frac{\partial E_i}{\partial \boldsymbol{\mu}_j^k} = -w_j \frac{\mathbf{x_i^k} - \boldsymbol{\mu_j^k}}{\sigma_j^2}(y_i - \sum_j w_j \Phi_{ij})\Phi_{ij}$$

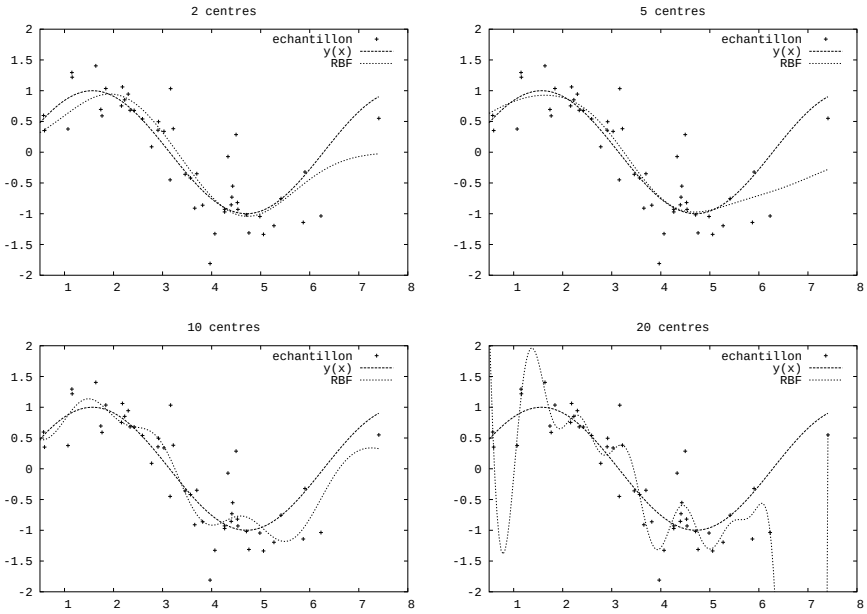

Figure 4.3. *Approximation d'une fonction avec un modèle RBF. Les 50 points de l'échantillon d'apprentissage, représentés sur les 4 courbes, sont générés comme suit :* $x = \mathcal{N}(3.2, 1.6), y = \sin(x) + \mathcal{N}(0, 0.3)$. *La courbe* $y(x)$ *est la « vraie » fonction (sinus), et la courbe « RBF » est la sortie du modèle. Avec 5 centres, l'approximation est correcte (sauf dans la partie droite où l'on a très peu de points). Avec seulement deux centres, l'erreur est supérieure, tandis qu'avec 10 voire 20 centres on note un* overfitting *important : la solution oscille et est très sensible au bruit.*

A partir de ces équations, on peut mettre en œuvre un algorithme d'apprentissage standard de minimisation de l'erreur, en version *batch* (calcul de l'erreur sur l'ensemble des exemple avant mise à jour des paramètres) ou en ligne (mises à jour après chaque exemple, approche qui en général offre de meilleures performances). Il s'agit cependant d'un problème non linéaire, et l'algorithme d'optimisation a de grandes chances de rester bloqué dans un minimum local de la fonction de coût. La réussite de l'optimisation dépend donc beaucoup des conditions initiales. Il est donc recommandé de n'utiliser l'optimisation globale des paramètres par descente de gradient qu'après un apprentissage séquentiel classique (voir section précédente). La descente de gradient permet alors d'effectuer un réglage fin des paramètres qui améliore les performances [FOG 93].

Notons que les solutions RBF obtenues avec un apprentissage par descente de gradient sont souvent assez différentes de celles obtenues par apprentissage séquentiel.

En particulier, rien ne garantit plus la *localité* de chaque fonction de base (autrement dit, les largeurs peuvent prendre des valeurs élevées).

4.3.3. *Modèles hybrides MLP-RBF*

Nous avons vu plus haut que les modèles RBF étaient peu appropriés au traitement de données de grande dimension (augmentation rapide du nombre de paramètres, sensibilité au bruit). Il est donc naturel de faire précéder ces modèles d'une phase de réduction de dimension, soit par sélection de variables, soit par une technique linéaire de type Analyse en composantes principales (ACP), soit encore en utilisant un réseau de neurones de type Perceptron multicouches (MLP, voir équation 4.3). Cette dernière solution [FOG 93] offre la possibilité d'utiliser des traitements non linéaires et permet un apprentissage conjoint des deux modules (MLP et RBF) par descente de gradient, afin d'assurer l'optimalité de la solution.

Les réseaux MLP peuvent être utilisés pour la compression de données (réseaux auto-associatifs) ou pour la discrimination. Parmi leurs avantages bien connus figure la possibilité d'effectuer des traitements calculs de type filtrage non linéaire très utiles pour le traitement des images et des signaux temporels (emploi de masque de poids partagés selon la technique nommée *Time delay Neural Networks*), et la bonne résistance au bruit. Leur apprentissage est toutefois nettement plus long que celui des réseaux RBF.

La motivation de l'association MLP/RBF est de profiter des capacités des réseaux MLP pour l'*extraction de caractéristiques*, qui seront traitées par le réseau RBF.

L'apprentissage d'un système hybride MLP/RBF se fait en plusieurs temps (figure 4.4) :

– initialiser les paramètres du réseau MLP par descente de gradient ; il s'agit d'un apprentissage supervisé ou auto-associatif ;

– remplacer la dernière couche du réseau MLP par un réseau RBF et effectuer un apprentissage séquentiel rapide (voir section 4.3.1) ;

– optimiser conjointement tous les paramètres du système MLP/RBF par descente de gradient. Cette dernière phase permet en général d'améliorer légèrement les performances du système.

4.3.4. *Autres approches*

D'autres approches ont été proposées pour l'apprentissage des modèles RBF, mais sont moins utilisées que la méthode séquentielle exposée plus haut car plus lourdes à mettre en œuvre. Elles sont toutefois intéressantes car elles permettent de situer les

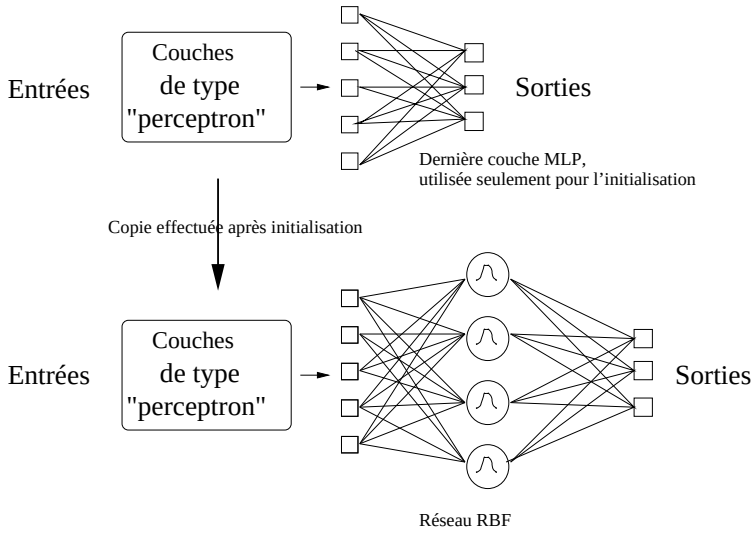

Figure 4.4. *Apprentissage d'un système hybride MLP/RBF. Afin d'éviter les minimums locaux et d'accélérer l'apprentissage, on commence par initialiser les couches MLP seules ; après convergence, on remplace la dernière couche du réseau MLP par un réseau RBF*

modèles RBF dans le contexte plus large de l'apprentissage statistique. Nous décrirons brièvement les techniques basées sur l'algorithme *Expectation Minimization* (EM) et les machines à vecteurs de support (SVM).

4.3.4.1. *Méthode EM*

La méthode EM [DEM 77] est un algorithme général qui peut s'utiliser pour calculer une estimation de type Maximum de vraisemblance des paramètres d'un modèle. Cet algorithme a été utilisé pour l'apprentissage de nombreux systèmes connexionnistes, dans un cadre supervisé ou non supervisé, pour la discrimination, la classification ou l'approximation de fonctions.

L'idée est de considérer le modèle RBF comme un modèle de mixture, la densité de probabilité des données étant de la forme :

$$p(\mathbf{x}) = \sum_{\mathbf{i}} \mathbf{P}(\mathbf{i}) \mathbf{\Phi_i}(\mathbf{x})$$

où les paramètres $P(i)$ sont les coefficients de la mixture, ici interprétés comme les probabilités *a priori* que les données aient été générées par la composante i de la mixture. La fonction de vraisemblance est :

$$\mathcal{L} = \prod_{n} p(\mathbf{x^n})$$

L'algorithme EM a été appliqué à l'apprentissage des RBF par plusieurs auteurs ([BIS 95, ORR 98], et plus récemment [LÁZ 03] qui proposent une formulation permettant un apprentissage EM rapide).

4.3.4.2. *Machines à vecteurs support (SVM)*

Les SVM, proposées par Vapnik au début des années 90 [BOS 92, VAP 95], sont basées sur le principe de minimisation structurelle du risque (MSR). Nous rappelons brièvement ci-après les principes de base de ces approches. Le lecteur intéressé pourra consulter [BUR 98] pour une excellente introduction aux SVM et [VAP 98, SCH 98] pour des études détaillées.

Modèle statistique de l'apprentissage

Nous présentons ici très brièvement la formulation statistique de l'apprentissage proposée par Vapnik [VAP 95] et qui débouche sur les SVM. Le chapitre 10 discute ces thèmes de façon plus appronfondie.

Nous nous plaçons ici dans le cas de la discrimination à 2 classes. On dispose d'un échantillon de m exemples, $x_i \in \mathbb{R}^n$, la classe de x_i étant donnée par $y_i = \pm 1$. On suppose qu'il existe une distribution de probabilité $P(x, y)$, inconnue, de laquelle est issu notre échantillon. Le but de l'apprentissage est ici d'estimer l'application associant y à x. Le système discriminant est défini par l'ensemble des applications $x \mapsto f(x; \alpha) = \pm 1$, la fonction f étant choisie dans un espace Φ. Par exemple, si Φ désigne l'ensemble des polynômes de degré n (seuillés vers $-1, 1$), α dénotera le vecteur des coefficients du polynôme. De façon générale, α indice les fonctions dans l'espace de recherche Φ, et n'est pas nécessairement un paramètre dans $\mathbb{R}^k$. L'apprentissage consiste à déterminer la valeur optimale de α, qui minimise le taux d'erreur moyen sur toutes les formes possibles, que l'on appelle le *risque* :

$$R(\alpha) = \int \frac{1}{2} |y - f(x; \alpha)| dP(x, y)$$

P étant inconnu, le risque l'est aussi ; par contre, on peut mesurer le risque *empirique* sur l'échantillon :

$$R_m(\alpha) = \frac{1}{2l} \sum_{i=1}^{m} |y_i - f(x_i; \alpha)|$$

La restriction de la recherche aux fonctions de l'espace Φ est très importante. Si on lève cette restriction, il est très facile de construire un système discriminant donnant un risque empirique nul (n'importe quelle fonction f telle que $f(x_i) = y_i$), mais une telle fonction ne minimisera pas forcément le risque R (sur-apprentissage, ou *overfitting*).

A l'opposé, si l'on choisit une classe trop petite, il peut être impossible d'y trouver une bonne solution. La dimension VC, notée h, permet de caractériser de ce point de vue la « richesse » d'une classe de fonctions[2].

Vapnik propose des bornes reliant le risque vrai R au risque empirique mesuré sur l'ensemble d'apprentissage. Avec la probabilité $1 - \eta$, l'inégalité suivante est vraie [VAP 95] :

$$R(\alpha) \leq R_m(\alpha) + \sqrt{\frac{1}{m}\left(h(\log(2m/h) + 1) - \log(\eta/4)\right)} \qquad [4.5]$$

L'approche « minimisation structurelle du risque » (MSR) consiste à minimiser cette borne (en choisissant la bonne classe de fonctions) au lieu de minimiser simplement le risque empirique.

Machines à vecteurs de vSupport (SVM)

Les SVM sont inspirées de la méthode MSR. L'idée de base est de minimiser la borne sur R en fixant (lorsque c'est possible) le risque empirique R_m à 0 puis en minimisant le deuxième terme pour trouver la meilleure valeur de h. Dans le cas général, il n'est pas possible de fixer le risque empirique : on recherche alors un compromis entre la minimisation de l'erreur sur l'ensemble d'apprentissage et la capacité de généralisation (minimisation du membre droit de l'équation 4.5).

Supposons dans un premier temps que l'échantillon $(x_i, y_i), i = 1, ..., m$ (où $y_1 = \pm 1$ donne la classe de chaque exemple) soit linéairement séparable ; il existe alors un hyperplan $w.x + b$ tel que :

$$w_i.x_i + b \geq 1 \quad \text{si} \quad y_i = 1 \qquad [4.6]$$

$$w_i.x_i + b \leq -1 \quad \text{si} \quad y_i = -1 \qquad [4.7]$$

Dans ce cas, on peut montrer [VAP 95] que la dimension VC h de la famille des classifieurs basés sur les fonctions $f(x; \alpha) = w.x + b$ avec $\|w\| < A$ est bornée par :

$$h \leq \min(r^2 A^2,\, n) + 1$$

2. On utilise la dimension VC de la classe de fonctions de perte associées aux modèles, voir chapitre 10.

où n est la dimension des entrées x, r le rayon de la plus petite sphère entourant tous les points x_i.

Ainsi, si l'on minimise $\|w\|$, on minimise le deuxième terme de l'inégalité (4.5). On peut très facilement constater que l'hyperplan qui minimise $\|w\|$ tout en respectant (4.6) et (4.7) est tel qu'il maximise la *marge*, c'est-à-dire la distance au point x_i le plus proche. Il est assez naturel que le système généralisant le mieux soit celui qui passe le plus loin de tous les exemples.

Pour trouver cet hyperplan, on peut minimiser $\Psi = \frac{1}{2}\, w.w$ sous les contraintes $y_i.f(x_i, w, b) \geq 1,\ \ i = 1...m$, qui résument les inégalités (4.6) et (4.7). Nous ne détaillons pas ici l'obtention de la solution (voir [BUR 98] ou [VAP 95]) ; en écrivant le lagrangien, on montre que la solution f s'écrit sous la forme :

$$f(x) = w_0.x + b = \sum_i \alpha_i\, x_i.x + b \qquad [4.8]$$

où α_i est le multiplicateur de Lagrange associé à chaque exemple. Les exemples x_i qui interviennent dans la solution ($\alpha_i \neq 0$) sont nommés *vecteurs de support*. Ils correspondent aux exemples proches de la frontière de décision, et sont situés sur la marge (voir figure 4.5).

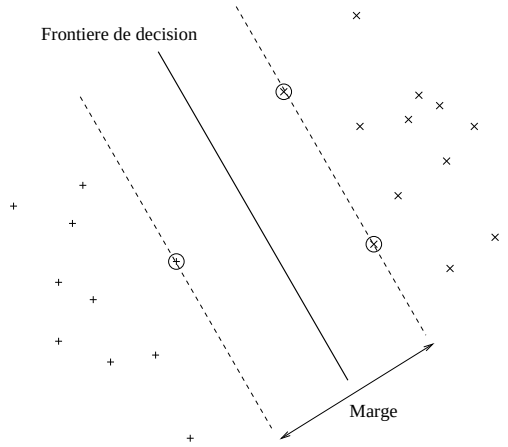

Figure 4.5. *Séparation linéaire optimale. Les trois vecteurs de support sont cerclés*

Cette approche s'étend au cas où les données ne sont pas séparables par f en relâchant les contraintes (4.6) et (4.7) grâce à des variables d'écart $\xi_i \geq 0$ qui permettent

à certains points de se situer du mauvais côté de la frontière :

$$w_i.x_i + b \geq 1 - \xi_i \quad \text{si} \quad y_i = 1 \qquad [4.9]$$

$$w_i.x_i + b \leq -1 + \xi_i \quad \text{si} \quad y_i = -1 \qquad [4.10]$$

Il faut alors minimiser $\sum_i \xi_i$, et on montre [VAP 95] que l'on obtient alors la même solution que précédemment, avec une contrainte supplémentaire : $\alpha_i \leq C$, où C est une constante positive qui permet de doser l'importance que l'on accorde *a priori* aux écarts. L'introduction de cet hyperparamètre, dont la détermination est le plus souvent effectuée par validation croisée, est lié au fait que l'approche SVM ne permet pas une optimisation directe de l'équation 4.5 (voir aussi sur ce thème le chapitre 10).

Les SVM s'étendent très élégamment pour construire des modèles non linéaires en remarquant que dans la solution (4.8), seul intervient le produit scalaire entre deux points, et que nous n'avons à aucun moment utilisé de propriétés spéciales de ce produit scalaire. On peut utiliser comme produit scalaire toute fonction noyau symétrique $K(x, y)$ respectant certaines conditions (conditions de Mercer [VAP 95, BUR 98]), et obtenir un comportement non linéaire :

$$f(x) = \sum_{i=1}^{m} y_i \alpha_i K(x_i, x) + b$$

(notons que les coefficients α_i sont nuls pour tous les points qui ne sont pas vecteurs de support.)

Par exemple, on construit des machines polynômiales de degré p avec $K(x, y) = (x.y + 1)^p$, et les machines « radiales » avec :

$$K(x, x_i) = \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right)$$

où σ fixe le caractère local de l'estimation.

Dans ce dernier cas, la fonction calculée à la même forme qu'un système RBF :

$$f(x) = \sum_{i=1}^{m} y_i \alpha_i \exp\left(-\frac{(x - x_i)^2}{2\sigma^2}\right) + b$$

L'apprentissage SVM détermine automatiquement les centres (un sous-ensemble des exemples) et les poids. Le paramètre σ est global, il concerne tous les centres, contrairement aux modèles décrits dans les sections précédentes.

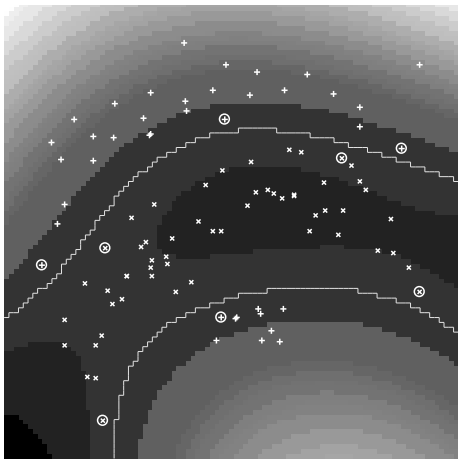

Figure 4.6. *Solution SVM avec noyau RBF sur un problème de dimension 2. Il s'agit de distinguer les croix droites des croix obliques. La solution utilise 8 vecteurs supports (points cerclés)*

On trouvera une comparaison expérimentale détaillée des SVM RBF et des RBF « traditionnelles » sur des tâches de discrimination dans [SCH 97]. Les principales conclusions de cette étude sont :

– les centres choisis par la méthode SVM ne sont pas situés près des centres de gravité des exemples, mais au contraire près des frontières entre les classes (voir figure 4.6). Cela permet en général d'annuler l'erreur de classement sur l'ensemble d'apprentissage ;

– dans tous les cas testés, l'erreur de généralisation obtenue par les SVM est plus faible.

Les approches SVM présentent cependant quelques inconvénients, en particulier liés à la difficulté de traiter de grands ensembles de données (plus de 10^4 exemples environ) et la difficulté du choix des paramètres (C et σ) qui oblige à recourir à de laborieuses procédures de validation croisée.

4.4. Conclusion

Les modèles RBF ont été très utilisés depuis la fin des années 80, et il serait impossible d'en dresser la liste des applications. On trouve de nombreuses utilisations en prévision des séries temporelles (voir par exemple [YEE 01]). Dans le domaine de la reconnaissance d'image ou de signal, il est d'usage de faire précéder le modèle RBF d'une extraction de caractéristiques (filtrage, contours, textures...) afin de réduire la dimension des entrées.

Il est facile de tester les divers modèles RBF grâce aux nombreuses implémentations libres disponibles (citons Torch en C++ [COL 02] ou les codes Matlab de Mark Orr[3] [ORR 00]).

L'étude des publications scientifiques sur le domaine montre qu'après une certaine effervescence au début des années 90, les recherches sur ce thème se sont stabilisées depuis quelques années, au profit en particulier des méthodes à base de noyaux (évolutions des machines à vecteur de support mentionnées dans ce chapitre).

4.5. Bibliographie

[BIS 90] BISHOP C., « Curvature-driven smoothing in backpropagation neural networks », *Proceedings of INNC Paris*, vol. 2, page749, 1990.

[BIS 95] BISHOP C. M., *Neural networks for pattern recognition*, Oxford University Press, Oxford, UK, UK, 1995.

[BOS 92] BOSER B. E., GUYON I. M., VAPNIK V. N., « A Training Algorithm for Optimal Margin Classifiers », HAUSSLER D., Ed., *5th Annual ACM Workshop on COLT*, Pittsburgh, PA, ACM Press, p. 144-152, 1992.

[BOT 95] BOTTOU L., BENGIO Y., « Convergence Properties of the K-Means Algorithm », *Advances in Neural Information Processing Systems*, vol. 7, Denver, MIT Press, 1995.

[BRO 88] BROOMHEAD D., LOWE D., « Multivariate functional interpolation and adaptative networks », *Complex Systems*, vol. 2, 1988.

[BUH 03] BUHMANN M. D., *Radial Basis Functions*, Cambridge University Press, Cambridge, 2003.

[BUR 98] BURGES C. J. C., « A Tutorial on Support Vector Machines for Pattern Recognition », *Data Mining and Knowledge Discovery*, vol. 2, n° 2, p. 1-47, 1998.

[COL 02] COLLOBERT R., BENGIO S., MARIÉTHOZ J., Torch : a modular machine learning software library, Rapport n° 02-46, IDIAP, 2002.

[DEM 77] DEMPSTER A. P., LAIRD N., RUBIN D. B., « Maximum likelihood from incomplete data via the EM algorithm (with discussion) », *Journals of Royal Statistics Society B*, vol. 39, p. 1-38, 1977.

[FOG 93] FOGELMAN SOULIÉ F., LAMY B., VIENNET E., « Multi-Modular Neural Networks Architectures for Pattern Recognition : Applications in Optical Characters Recognition and Human Face Recognition », *Int. J. Pattern Recognition and Artificial Intelligence*, vol. 7, n° 4, p. 721-755, 1993, Extended version as Tech. Report 827, LRI 1993.

[HAS 90] HASTIE T. J., TIBSHIRANI R. J., *Generalized additive models*, London : Chapman & Hall, 1990.

3. disponibles sur `http ://www.anc.ed.ac.uk/~ mjo/rbf.html`

[HOE 62] HOERL A., « Application of ridge analysis to regression problems », *Chemical Engineering Progress*, vol. 58, p. 54-59, 1962.

[LÁZ 03] LÁZARO M., SANTAMARÍA I., PANTALEÓN C., « A new EM-based training algorithm for RBF networks », *Neural Networks*, vol. 16, p. 69-77, janvier 2003.

[MAC 67] MACQUEEN J., « Some Methods for Classification and Analysis of Multivariate Observations », *Proc. of the fifth Berkeley Symposium on Mathematics, Statistics and Probabilities*, vol. 1, p. 281-297, 1967.

[MOO 89] MOODY J., DARKEN C., « Fast Learning in Networks of Locally-tuned Processing Units », *Neural Computation*, vol. 1, p. 281-294, 1989.

[ORR 95] ORR M. J. L., « Regularization in the Selection of Radial Basis Function Centers », *Neural Computation*, vol. 7, p. 606-623, 1995.

[ORR 98] ORR M., « An EM Algorithm for Regularised RBF Networks », *International Conference on Neural Networks and Brain*, 1998.

[ORR 00] ORR M., HALLAM J., MURRAY A., LEONARD T., « Assessing RBF networks using DELVE », *International Journal of Neural Systems*, vol. 10, p. 397-415, 2000.

[PAR 91] PARK J., SANDBERG I. W., « Universal Approximation Using Radial-Basis-Function Networks », *Neural Computation*, vol. 3, p. 246-257, 1991.

[POG 90] POGGIO T., GIROSI F., « Networks for Approximation and Learning », *Proceedings of the IEEE*, vol. 78, n° 9, p. 1481-1497, septembre 1990.

[POW 87] POWELL M. J. D., « Radial basis functions for multivariable interpolation : a review », *Algorithms for approximation*, New York, NY, USA, Clarendon Press, p. 143-167, 1987.

[RIC 64] RICE J., *The Approximation of Functions*, vol. 1, Addison-Wesley, Reading, MA, 1964.

[SCH 97] SCHÖLKOPF B., SUNG K., BURGES C., GIROSI F., NIYOGI P., POGGIO T., VAPNIK V., « Comparing support vector machines with Gaussian kernels to radial basis function classifiers », *IEEE Trans. Sign. Processing*, vol. 45, p. 2758-2765, 1997, AI Memo n° 1599, MIT, Cambridge.

[SCH 98] SCHÖLKOPF B., BURGES C., SMOLA A., Eds., *Advances in Kernel Methods - Support Vector Learning*, MIT Press, Cambridge, USA, 1998.

[TIK 77] TIKHONOV A., ARSENIN V., *Solutions of Ill-posed Problems*, W.H. Winston, Washington, D.C., 1977.

[VAP 95] VAPNIK V., *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA, 1995.

[VAP 98] VAPNIK V., *Statistical Learning Theory*, Wiley, New York, NY, USA, 1998.

[YEE 01] YEE P. V., HAYKIN S., *Regularized Radial Basis Function Networks : Theory and Applications*, Wiley, New York, NY, USA, 2001.