Les Réseaux de Neurones à fonctions à base radiale

Yann MORERE

Le 10 mai 2001

Résumé

Les réseaux de neurones à fonctions à base radiale on connu un essor considérable. Un résumé sur tout ce qu'il faut savoir sur les réseau de neurones à fonctions à base radiale est contenu dans ces pages.

Table des matières

Table des figures

Chapitre 1

Les Réseaux à fonctions à base de radiale

1.1 Les fonctions radiales

C'est une classe de fonctions spéciales. Leur réponse croit où décroît de façon monotone par rapport à la distance d'un point central. Le centre, la distance, et la forme de la fonction à base radiale sont les paramètres du modèle qui est linéaire si ils sont fixent.

Une fonction à base radiale typique est de la forme :

$$h(x) = exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{1.1}$$

ces paramètres sont le centre c le rayon r. Une fonction à base radiale Gaussienne décroît quand la distance par rapport au centre augmente. A l'opposé, une fonction à base radiale multiquadratique croît la distance par rapport au centre augmente. Elle a la forme suivante :

$$h(x) = \frac{\sqrt{r^2 + (x-c)^2}}{r^2} \tag{1.2}$$

Les RBF gaussiennes sont les plus utilisées car elles sont plus plausibles biologiquement et elles ont une réponse finie ce qui est plus pratique (Cf. figure 1.1).

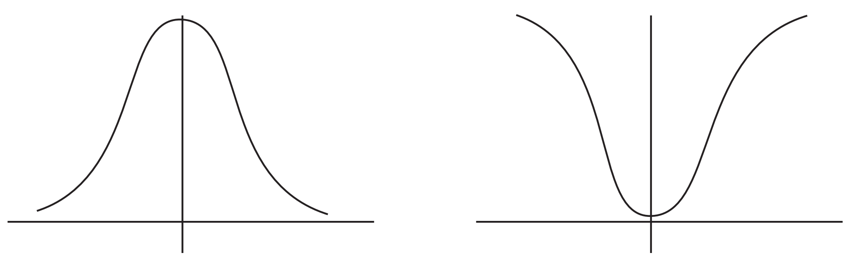

FIG. 1.1 – RBF Gaussienne et RBF Multiquadratique

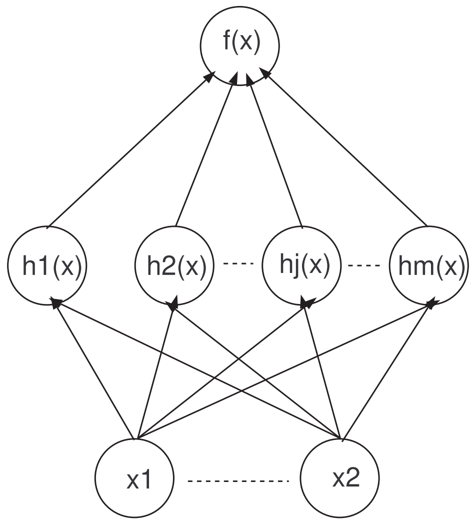

FIG. 1.2 – Réseau de type Perceptron avec une seule couche cachée

1.2 Les réseaux de neurones à fonctions à base radiale

Les réseaux RBF est une architecture de réseaux Feedforward puissante. Ce type de réseau a été introduit par Hardy [HARDY, 1971], et la théorie correspondante a été développée par Powell [POWELL, 1985]. Ces réseaux étaient à l'origine appliqués aux problèmes d'interpolation multivariable et ils ont pris le terme de réseaux de neurones grâce à Broomhead et Lowe.

Au départ le nombre d'unités de base radiale correspondait au nombre de points de données. Ce qui était une sérieuse limitation car si le nombre de points est élevé la méthode devient impraticable. Maintenant le nombre d'unité est largement inférieur aux nombre de points de données.

Ces réseaux de neurones ont trouvé des domaines d'applications variés comme le traitement de la parole, traitement de l'image et traitement du signal.

Les fonctions à base radiale sont une classes simple de fonction. En principe elle peuvent être employées dans toutes sortes de modèles linéaires ou non, et tout type de réseaux à couche simple ou multicouche. Mais on remarque que les réseaux à fonction à base radiales sont souvent utilisés dans des réseaux à une seule couche cachée (Cf. figure 1.2).

Chaque composant n du vecteur d'entrée x nourrissent les m neurones à fonction à base radiale dont les sorties sont combinées linéairement avec les poids $\{\omega_j\}_{j=1}^m$ dans la sortie du réseau $f(x)$ (Cf. figure 1.2).
Un réseau RBF est non-linéaire si les fonctions à base radiales peuvent modifier leur centre ou leur forme ou encore s'il y a plus d'une couche cachée.
D'après Baker [BAKER et FARRELL, 1992] il est naturel d'imposer des contraintes au comportement d'un réseau de neurones.

- Contraintes d'approximation universelle.
- Contrainte de couverture : qui s'exprime par le fait que pour chaque sortie y_i associé

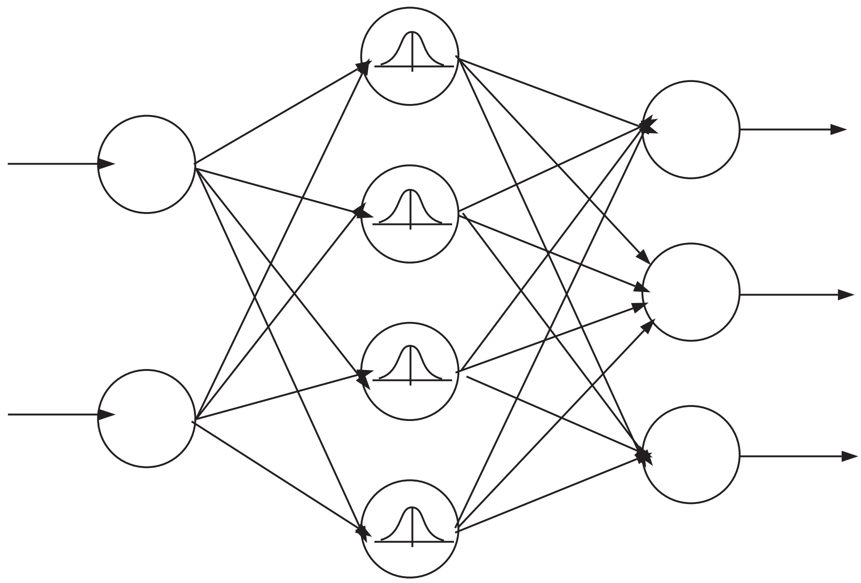

FIG. 1.3 – Exemples d'architectures

au vecteur d'entrée $\vec{u}$ il existe au moins un paramètre w_j tel que $|\frac{\partial y_i(u;\omega)}{\partial \omega_i}|$ (fonction de sensibilité) soit très grande au voisinage de $\vec{u}$.

- Contrainte de généralisation locale : si la valeur $\vec{u}$ il existe au moins un paramètre w_j tel que $|\frac{\partial y_i(u;\omega)}{\partial \omega_i}|$ est très grande au voisinage de $\vec{u}$, elle doit être relativement proche de 0 une fois que l'on s'éloigne de ce voisinage.

Les réseaux de neurones à fonction de base radiale satisfont à ces contraintes. Ils sont composés en outre d'une couche d'entrée et de sortie, une couche cachée de n unités qui ne réagissent significativement à une partie de l'espace d'entrée (échantillon) suivant un fonction d'activation de type gaussien. Ainsi pour $p = 1$ (une seule sortie), en utilisant ce type de réseau, la valeur de sortie est donnée par :

$$y = \frac{\sum_{j=1}^{n} \omega_j p_j(u)}{\sum_{j=1}^{n} p_j(u)} \tag{1.3}$$

où $\sum_{j=1}^{n} p_j(u)$ est le terme de normalisation et où

$$p_j(u) = exp\left(-\frac{1}{2}\sum_{k=1}^{m}\frac{(u_k - c_{k,j})^2}{\sigma_{k,j}^2}\right) = exp\left(-\frac{1}{2}(u - c_j)^t\, \mathcal{Q}_j\,(u - c_j)\right) \tag{1.4}$$

Les vecteurs $c_j (j = 1, \ldots, n)$ sont les centres des gaussiennes (même dimension que le vecteurs d'entrée $\vec{u}$).

Dans les tâches de classification, les centres de gaussiennes sont choisis par des techniques de regroupement de données (ou technique de clustering de type K-means), le but étant de choisir les centres comme points représentatifs des zones de grande concentrations de données. Mais quand on travaille en ligne ces méthodes ne sont plus applicables. L'idée est alors de disposer les centres en treillis régulier pour couvrir uniformément la partie utile de l'espace d'entrée et respecter les contraintes de couverture et généralisation localisée.

Les matrices $\mathcal{Q}$ sont les matrices des rayons choisies le plus souvent diagonales $\mathcal{Q}_j^{-1} = diag(\sigma_{1,j}^2, \sigma_{2,j}^2, \ldots, \sigma_{m,j}^2)$.
Dans le cas où les centres sont disposés en treillis la valeurs des rayons sont fixées par la distance entre centres voisins afin d'assurer un certain taux de recouvrement entre les différentes gaussiennes. Ce degrés de recouvrement résulte d'un compromis entre les propriétés de localité et de généralisation. L'étendue de la généralisation est donc contrôlable.
Mais ces réseaux offrent d'autres avantages

- On a la possibilité d'alléger les calculs en utilisant le caractère local de l'activation des neurones ("champs d'influence restreint") et la propriété de généralisation locale : pour une entrée donnée lors du calcul de la valeur de sortie il n'est pas nécessaire de calculer l'activation de tous les neurones de la couche cachée : seuls les neurones dont le champ d'influence touche significativement l'entrée considérée doivent être pris en compte. Il en est de même lors de l'apprentissage si on prend une méthode type descente de gradient.
- On peut aussi remarquer la similitude de l'approche avec des techniques de modélisation floue. Il est aisé de montrer l'équivalence entre le réseau à fonction radiale de base décrit précédemment et un raisonnement flou basé sur un certain nombre de règles floues du types : Si (U_1 est $A_j; U_2$ est $B_j \ldots U_m$ est C_j) Alors Y est $D_j (A_j, B_j, C_j, D_j)$ sont des termes linguistiques définissant des ensembles flous de références, j est l'indice de la règle.
  La notation X indique une variable floue résultant de l'estompage de la variable nette x.
  L'équivalence s'obtient par un choix bien spécifique des différents opérateurs flous (conjonction, implication, désestompage...) et des fonctions d'appartenance définissant les termes linguistiques.
  Ces fonctions doivent être des gaussiennes (de moyennes $c_{j,k}$ et d'écart-type $\sigma_{k,j}$) ayant pour équations :
  $$\mu_j(u_k) = exp\left(-\frac{1}{2}\sum_{k=1}^{m}\frac{(u_k - c_{k,j})^2}{\sigma_{k,j}^2}\right) \tag{1.5}$$
- On peut remarquer que ce choix correspond à une vision très appauvrie de la modélisation floue.
- On peut remarquer que chaque unité de la couche cachée correspond à une règle floue. La valeur ω_j (le mode de la fonctions d'appartenance) correspond à une valeur nette caractéristique du terme linguistique D_j intervenant dans la conséquence de la règle j. Cette équivalence permet de donner une interprétation "humaine" des réseaux à fonction à base radiale.

En réalité la comparaison n'est pas aussi claire, car une fonction à base radiale peut être obtenue par une combinaison judicieuse de deux fonctions d'activation sigmoidales si bien que les réseaux à fonctions à base radiale peuvent être considérés comme des réseaux sigmoïdaux à architecture particulière.
Ces réseaux présentent aussi des défauts. Leur caractère local n'est pas toujours souhaitable surtout si le support est infini et la dimension de l'espace d'entrée est élevé. On peut alors utiliser les réseaux à bases radiales modifiées que l'on nomme "Unités semi-locales" [HARP et SAMAD, 1992].

à base radiale	sigmoïdale	
Approximation Universelle	oui	oui
Dépendance entre les paramètres (poids)	linéaire	non-linéaire
Effet d'un changement paramétrique	local	global
Domaine de définition fini (borné)	infini	
Interprétation des paramètres	possible	non
Charges de calcul réductibles	oui	non

TAB. 1.1 – Comparaisons des propriétés des réseaux de neurones à fonctions à base radiale et sigmoïdale.

La plupart des applications qui utilisent les fonctions à base radiale possèdent des centres et des distances fixées une fois pour toutes. Les seuls paramètres ajustables sont les poids associés à chaque unité cachée. On sait que la sortie du réseau est une fonction linéaire des paramètres ce qui permet d'utiliser les méthodes classiques d'identification paramétriques et ainsi d'obtenir un apprentissage rapide et précis.

La disposition des centres a priori est parfois impraticable car le nombre de centre requis est trop élevé. On propose alors d'utiliser un nombre plus restreint et de les laisser se disposer. C'est alors l'algorithme d'apprentissage qui aura la charge d'ajuster les poids et de positionner les centres et les valeurs des rayons.

Le réseau s'autorganisera pour placer un maximum de centre dans les endroits accidentés de la fonctions et les espacera dans les zones plus lisses. Mais alors la sortie n'est plus linéaire par rapport aux paramètres et le risque de minima locaux apparaît. Mais leurs apparitions n'est pas aussi critique que dans le cas des réseaux sigmoïdaux car l'adjonction de connaissances a priori sous la forme de la position des centres, des distances et des poids permet de trouver une solution pas trop éloigné de l'optimal.
On parle alors de réseaux à base radiale généralisés. On doit donc oublier les algorithmes d'apprentissage type filtre de Kalman et autres moindres carrés. On peut utiliser la méthode de la descente du gradient avec laquelle la parallélisation au niveau du neurone peut être conservée.

Chaque neurone j de la couche cachée est capable d'ajuster les poids ω_j ainsi que les centre c_j et les rayons σ_j indépendamment des autres neurones de la couche cachée car il n'utilise que ses propres paramètres pour calculer ses corrections (erreur de sortie et facteur de normalisation). Avec une seule sortie les valeurs des gradients sont données par

Soit $\mathcal{J} = \frac{1}{2}(y - y^*)^2$ le critère à minimiser.
Posons $\mathcal{R}(u) = \sum_{i=1}^{n} \rho_i(u)$ facteur de normalisation.

$$\frac{\partial \mathcal{J}}{\partial \omega_j} = (y - y^*)\frac{\rho_j(u)}{\mathcal{R}(u)}$$

$$\frac{\partial \mathcal{J}}{\partial c_{j,k}} = \frac{\partial \mathcal{J}}{\partial \rho_j}\frac{\partial \rho_j}{\partial c_{j,k}}$$

$$\frac{\partial \mathcal{J}}{\partial \sigma_{j,k}} = \frac{\partial \mathcal{J}}{\partial \rho_j}\frac{\partial \rho_j}{\partial \sigma_{j,k}}$$

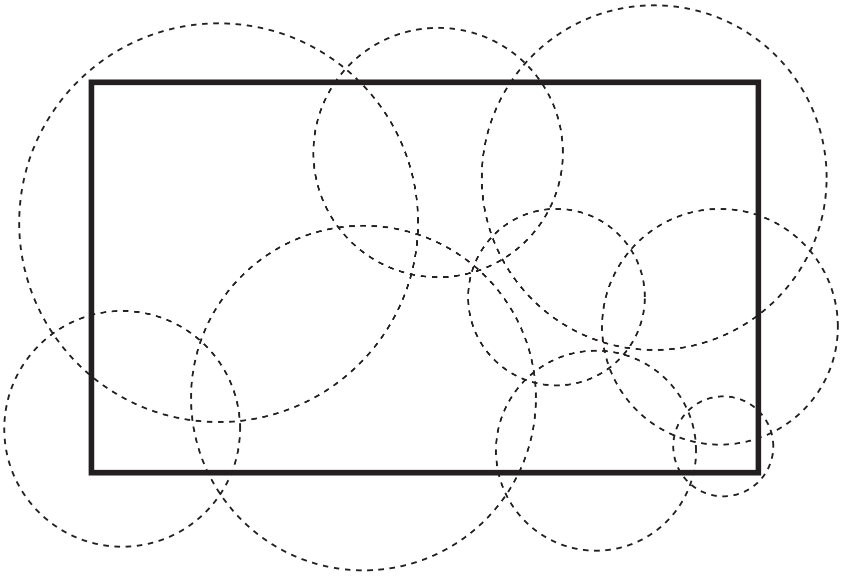

Fig. 1.4 – Dallage flou par un réseau à fonction à base radiale

$$\frac{\partial \mathcal{J}}{\partial \rho_j} = (y - y*) \left[\frac{w_j \mathcal{R}(u) - \sum_{i=1}^{n} \rho_i(u)\omega_i}{\mathcal{R}(u)^2} \right]$$

$$\frac{\partial \mathcal{J}}{\partial c_{j,k}} = \rho_j \cdot \frac{u_k - c_{j,k}}{\sigma_{j,k}^2}$$

$$\frac{\partial \mathcal{J}}{\partial \sigma_{j,k}} = \rho_j \cdot \frac{(u_k - c_{j,k})^2}{\sigma_{j,k}^3}$$

Les corrections s'obtiennent par :

$$\Delta\omega_j = -\eta_\omega \frac{\partial \mathcal{J}}{\partial \omega_j}$$

$$\Delta c_{j,k} = -\eta_c \frac{\partial \mathcal{J}}{\partial c_{j,k}}$$

$$\Delta\sigma_{j,k} = -\eta_{sigma} \frac{\partial \mathcal{J}}{\partial \sigma_{j,k}}$$

Il y a donc maintenant $(1 + 2m)n$ paramètres à ajuster au lieu des n précédents. (m est le nombre d'entrées).

On peut transposer l'algorithme de corrections des paramètres d'un réseau à fonction à base radiale à certains problème de modélisation de données par raisonnement flou. Les moyennes et variances gaussiennes définissant les fonctions d'appartenance des termes linguistiques des règles floues.

La disposition des centres et des rayons définit un dallage de l'espace d'entrée (Cf. figure 1.4). Il correspond à une partition floue de l'espace en raison du recouvrement. Ce dallage n'offre pas de topologie régulière. Mais il manque toujours la moyen de déterminer le nombre d'unités sur la couche cachée c'est à dire le nombre de règles floues utilisées.

1.3 Critère de sélection de réseaux RBF

1.3.1 La validation croisée

C'est le principal outil de mesure de l'erreur de prédiction.

Mise en oeuvre : Lorsque nous avons beaucoup d'échantillons on peut alors diviser l'échantillon de départ en deux. Une partie servira à l'apprentissage l'autre à la validation du réseau ainsi obtenu. Si on a plusieurs modèles, on réalise l'apprentissage sur le jeu d'échantillons d'apprentissage et l'autre jeu d'échantillons servira à sélectionner le meilleur.
Une autre méthode, consiste à partitionner le jeu d'échantillons initial en différents sous-ensembles et de calculer une partition moyenne. Ceci évitera le biais possible induit par l'utilisation d'une seule partie des données. Cette méthode est meilleure.
Une variante encore plus précise consiste à diviser l'échantillon de départ de p composants en un échantillon d'essai de p-1 échantillons et d'avoir un seul échantillon de test. On peut alors recommencer en changeant l'échantillon de test. Dans ce sens toutes les données peuvent être utilisées comme échantillons de test et d'entraînement. Il s'agit de la validation croisée LOO "Leave One-Out".

1.4 Les réseaux de neurones RBF retardés

Afin de reconnaître des signaux variables dans le temps le perceptron est combiné avec des retards pour donner les TDNN Time Delay Neural Network. Mais pour de longues séquences temporelles les TDNN ont besoin de beaucoup de paramètres et d'un apprentissage long.
Au contraire les réseaux RBF n'ont besoin que de peu de paramètres et le temps d'apprentissage est plus court. Mais leur principal défaut est qu'il ne détecte pas les caractéristiques temporelles. Il faut donc combiner les caractéristiques des deux réseaux.

1.4.1 Architecture

Pour que les réseaux RBF puissent détecter les caractéristiques temporelles on doit lui présenter non pas une entrée à l'instant présent, mais plusieurs. Le réseau est alors en prise avec les entrées d'une fenêtre temporelle (cf. figure 1.5). Le principal problème de cette structure est son instabilité face aux décalages temporels. D'un coup sur l'autre la présentation d'une même entrée peut donner des réponses différentes. C'est pourquoi on lui ajoute une deuxième couche qui effectue une simple intégration temporelle.
Le réseau résultant est du type de la figure 1.6.

On constate que le réseau se contente de prendre en compte les données sur une petite fenêtre temporelle. La seconde couche effectue une somme des activations de la couche RBF. Le domaine d'application principal est la reconnaissance de la parole.

Les réseaux feedforwards qui utilisent les fonctions à base radiale dans la couche cachée d'après Park [PARK et SANDBERG, 1991] et Sandberg [PARISINI et ZOPPOLI, 1993] constituent une classe d'approximateurs universels pour une fonction non-linéaire.

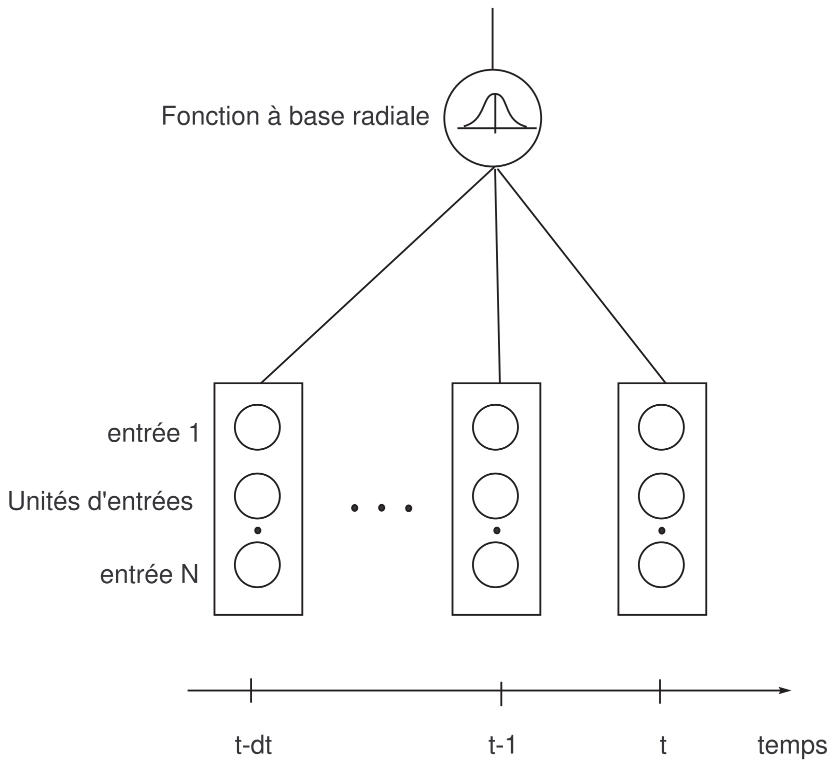

FIG. 1.5 – Réseau RBF avec retard

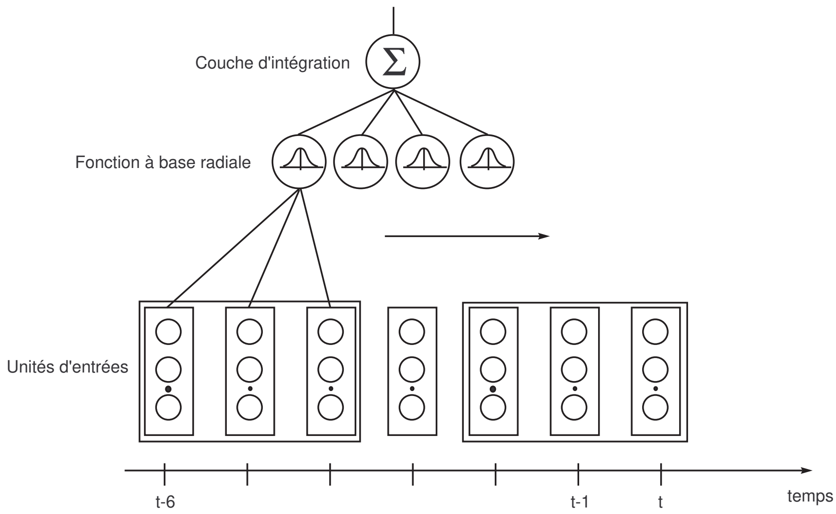

FIG. 1.6 – Structure d'un réseau RBF à retard

1.5 Application des réseaux RBF à l'émulation de fonction

Le réseau RBF monocouche décrit par Broomhead et Lowe [BROOMHEAD et LOWE, 1988](RBFN) constitue une famille multi-paramètres de fonctions avec des paramètres modifiables par apprentissage qui sont les paramètres de contrôle. Dans un réseau de ce type on a une seule couche cachée et la sortie du i ème neurones est donnée par :

$$f_i(x) = \sum_j \omega_{i,j} g_j(x)$$

avec

$$f_i(x) = g\left(\frac{\|x - \bar{x}_j\|}{\sigma_j}\right)$$

où g est une fonction radiale symétrique (gaussienne).

On Utilise des réseaux de neurones à fonctions à base radiale pour l'émulation de fonction. Ici ce n'est pas la copie de la fonction en terme quantitatif (approximation de la fonction avec une erreur minimale) mais une approximation qualitative qui permet une meilleure généralisation pour un système d'entrée-sortie inconnu. Ceci est basé sur le fait que l'humain lors d'un apprentissage donne l'avantage à l'approche global du problème. Pour l'apprentissage d'une fonction on s'attachera à la forme globale.

1.6 Les réseaux RBF pour la généralisation de scénarii réalisables et non-réalisables

Dans l'apprentissage supervisé une erreur importante à prendre en compte est l'erreur de généralisation. Une mesure entre la sortie désirée et la sortie réelle. L'erreur de généralisation est composée de deux erreurs :

- Erreur d'approximation
- Erreur d'estimation

Sur une architecture de réseau (entraîné) apprenti l'erreur d'approximation est celle donnée par le meilleur (réseau entraîné) apprenti de cette architecture.

Et cette erreur est due à l'incapacité du réseau à représenter le procédés à la perfection. Si l'erreur d'approximation est nulle le problème est alors dit réalisable sinon il est dit irréalisable.

L'erreur d'estimation est due au fait de ne pas avoir sélectionner un (réseau entraîné) apprenti optimal pour l'architecture choisie. Ceci est du au fait du nombre de données insuffisante pour l'apprentissage, donnée bruitée ou encore un algorithme d'apprentissage qui ne garanti pas un minimum global.

En théorie on peut éliminer l'erreur d'approximation en ayant un très grand nombre d'échantillons. D'après Hartman et Al. [HARTMAN *et al.*, 1990] les réseaux de neurones à fonction à base radiale sont des approximateurs universels parcimonieux car, avec un nombre suffisant d'unités à fonctions à base radiale, n'importe quelle fonction peut être approximée.

1.7 Le réseau RBF

Le réseau RBF est composé de deux couches complètement interconnectées. Chaque unité cachée est paramètrée par deux valeurs :

- Un centre m de l'espace d'entrée, qui correspond au vecteur définit par les poids des connexions entre les unités d'entrées et les unités cachées
- une largeur σ_B^2

Ces paramètres sont supposés être fixés par un algorithme adapté comme un algorithme de segmentation (clustering) ou de maximisation de ressemblance des paramètres en regard des données d'apprentissage. La fonction d'activation des unités cachées est radialement symétrique dans l'espace d'entrée.

La magnitude (puissance, grandeur) de l'activation est une fonction décroissante de la distance euclidienne entre le vecteur d'entrée x d'un point de donnée et le centre de la fonction à base radiale. La couche de sortie calcule une combinaison linéaire des sorties des fonctions d'activations. A l'intérieur du réseau les fonctions à base radiale sont gaussiennes et les largeurs sont toutes identiques sur la couche cachée. D'où la fonction totale calculée par le réseau est de la forme :

$$f(x,\omega) = \sum_b \omega_b exp\left(-\frac{\|x-m_b\|^2}{2\sigma_b^2}\right)$$

Bien que les fonctions gaussiennes soient beaucoup utilisées dans le cadre des réseaux RBF il existe d'autres fonctions d'activation radiale. L'algorithme d'apprentissage utilisé est stochastique : On ajoute un bruit gaussien de moyenne nulle aux poids du réseaux de neurones. La minimisation de l'erreur d'apprentissage revient donc à maximiser la ressemblance des données.

Les algorithmes de descente de gradient peuvent aussi être utilisé (résolution des équations du type Folker-Planck). Afin d'éviter l'interdépendance de la distribution des vecteurs de poids on doit adjoindre un facteur de régularisation qui peut être vu comme une première distribution sur l'espaces des (réseaux entraînés) apprentis : $P(\omega,\gamma) = \frac{-\gamma E_\omega}{Z_\omega}$ où E_ω est le terme de pénalité sur la puissance (grandeur) du vecteur des poids des (réseaux entraînés) apprentis et $Z_\omega = \int_\omega d\omega exp(-\gamma E_\omega)$.

1.7.1 Calcul de l'erreur de généralisation

D'après Freeman et Saad [HOLCOMB et MORARI, 1991] il y a plusieurs façon de définir l'erreur de généralisation. Une définition simple et intuitive serait de dire que c'est la prédiction sur la distribution d'entrée du carré de la différence entre la fonction cible réelle et une fonction estimante :

$$E = \int_X P(x)\left(f(x,\omega)\right)^2\,dx$$

D'un point de vue pratique on considère qu'un seul peut accéder au risque empirique : $\frac{1}{P_{test}}\sum_{p=1}^{P_{test}}\left(y_p - f(x,\omega)\right)^2$ Cette valeur est l'approximation du risque espéré.

1.8 Algorithme d'apprentissage supervisé en ligne pour les réseaux RBF

L'utilisation des réseaux RBF principales entraîne la sélection des spécifications du réseau (structure) et la stratégie (algorithme) de mise à jour des poids (paramètres). Le premier concerne le choix de la dimension des entrées du réseau, la sélection du type de non-linéarité RBF et le nombre d'unités dans la couche cachée. La deuxième sélection se fait dans la description d'un algorithme performant pour déterminer les centres ainsi que les poids du réseau ainsi que les paramètres de largeur du réseau (fonction RBF).

Lorsque la structure du réseau est fixe on peut opérer séparément l'apprentissage des centres et des poids du réseau en deux phases successives.

Les centres sont souvent choisis aléatoirement au départ et leur valeur est affinée pas à pas par des méthodes de segmentation. Avec ces centres ainsi définit on peut effectuer l'optimisation des poids du réseau par une routine d'estimation de paramètres standard. Avec cette stratégie de mise à jour des paramètres les centres sont mis à jour de façon non-supervisée et les poids sont mis à jour de façon supervisée. L'avantage en adoptant cette stratégie hybride est de réduire sensiblement l'explosion calculatoire par rapport à une méthode complètement supervisée.

Différents algorithmes efficaces ont été développé pour ces réseaux à structure statique.

D'un point du vue de modélisation de système si on ne connaît rien sur le système le réseau choisi sera sûrement surdimensionné. Si on laisse de coté le fait que les calculs sont plus fastidieux, du point de vue de la généralisation ce surdimensionnement peut être catastrophique et peut même remettre en cause le principe de parcimonie des réseaux de neurones. Et ceci est d'autant plus important dans le cas où l'on travail en temps réel avec des systèmes variant largement dans le temps.

Une nouvelle approche d'optimisation est ici étudiée : en étudiant la variance de la contribution des incréments successifs de chaque regresseur, la structure du modèle est adaptée en ligne et un sous-ensemble de centre est sélectionné afin d'avoir un coût de calcul raisonnable. Cette algorithme combine la sélection de candidat en ligne par des régresssseurs avec une routine de moindres carrés récursifs, afin de d'obtenir un apprentissage adaptatif en ligne d'un réseau qui pourra être utilisé dans la reconnaissance de forme ainsi que dans l'acquisition des séquences temporelles.

1.8.1 Représentation NARMAX pour les systèmes non-linéaires

L'étude des systèmes non-linéaires est importante car de nombreux systèmes physiques comportent des non-linéarités en degrés plus ou moins élevé. Une bonne représentation est alors très importante pour la modélisation de ces systèmes. Comme une extension naturel des équations différentielles linéaires une modélisation non-linéaire est définie : le modèle NARMAX : Non-linear Autoregressive Moving Average with eXogenous inputs.

Soit un vecteur $\tilde{x}(k)$ à l'instant composé de $(mN_u + nN_y + nN_e)$.
On a

$$\tilde{x}(k) \equiv \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-N_u) \\ y(k-1) \\ \vdots \\ y(k-N_y) \\ e(k-1) \\ \vdots \\ e(k-N_e) \end{bmatrix}$$

où $u(k) \in \mathbb{R}^m, y(k) \in \mathbb{R}^n, e(k) \in \mathbb{R}^n$ sont les entrées, les sorties et le bruit du système avec les plages d'entrée maximum N_u, N_y, N_e à l'instant k respectivement des m entrées et n sorties. La sortie du système est donnée par la relation : où $y(k) = f\left(\tilde{x}(k)\right) + e(k)$ où $f(\cdot) \in \mathbb{R}^m$ est une fonction valuée d'un vecteur de réels. Le choix de la fonction $f(\cdot)$ détermine les caractéristiques dynamiques du systèmes non-linéaires car le bruit n'est ajouté qu'en sortie. La fonction f est souvent une fonction très complexe et obtenir une bonne représentation de cette dernière n'est pas une chose facile. On doit donc utiliser une approximation de cette dernière. Les réseaux de neurones RBF seront utilisés ici pour approximer ces fonctions. Donc la sortie prédite du système $\hat{y}(k) \in \mathbb{R}^n$ sera la sortie du réseau RBF $\hat{y}(k) = \hat{f}\left(x(k)\right)$ où $x(k) \in \mathbb{R}^{N_i}$ est le vecteur d'entrée définit par :

$$\tilde{x}(k) \equiv \begin{bmatrix} u(k-1) \\ \vdots \\ u(k-N_u) \\ y(k-1) \\ \vdots \\ y(k-N_y) \\ e(k-1) \\ \vdots \\ e(k-N_e) \end{bmatrix}$$

1.8.2 Représentation RBF d'un modèle NARMAX

Un réseau RBF consiste en une seule couche cachée composé d'un tableau d'unités non-linéaires. Chacune des ces unités est associée avec une valeur scalaire qui est le paramètre de largeur. Pour N_i entrées et n sortie avec un réseau à N_c unités cachées à l'instant k, la i^{e}unité cachée effectue un fonction non-linéaire :

$$\Phi = \underbrace{\mathbb{R}^{N_i}}_{\textit{Entrée du réseau}} \times \underbrace{\mathbb{R}^{N_i}}_{\textit{Centre}} \times \underbrace{\mathbb{R}_+}_{\textit{Paramètre de largeur}} \rightarrow \underbrace{\mathbb{R}}_{\textit{Sortie de la couche cachée}}$$

qui est caractérisée par la valeur courante du centre $c_i(k) \in \mathbb{R}^{N_i}$ et la paramètre de largeur $\omega_i(k) \in \mathbb{R}$ sur la dimension N_i du vecteur des entrées $x(k) \in \mathbb{R}^{N_i}$ pour produire la i^{e}sortie de la couche cachée :

$$\varphi_i(k) = \phi\left(\|x(k) - c_i(k)\|, \omega_i(k)\right), 1 \leq i \leq N_c$$

où $\|\cdot\|$ dénote un norme dans l'espace d'état à N_i dimensions. La fonction non-linéaire $\phi(\cdot)$ est généralement symétrique radialement autour de son centre. On peut alors utiliser une fonction du type $\phi(x) \equiv x^2 \log_e x$ qui est une spline plate. (choisie car elle ne possède pas de paramètre de largeur ce qui réduit le coût de calcul) Avec la fonction RBF spécifiée précédemment on a :

$$\varphi_i(k) = \phi\left(\|x(k) - c_i(k)\|, \omega_i(k)\right), i = 1, 2, \cdots, N_c$$

La sortie de chaque unité de la couche de sortie est donc la somme pondérée des sorties des unités de la couche cachée. On peut donc dire que la j^esortie associée au vecteur de poids estimé à l'instant k $\theta_j(k) = \{\theta_{ij}(k)\}_{i=1}^{N_c} \in \mathbb{R}^{N_c}$ et la sortie j du réseau $\hat{y}_j(k) = \varphi^T(k)\theta_j(k), 1 \leq j \leq n$ où $\varphi_j(k) \equiv \{\varphi_{ij}(k)\}_{i=1}^{N_c}$. La réponse du réseau de dimension n peut être exprimée par $\hat{y}(k) = \{\varphi^T(k)\theta_j(k)\}_{j=1}^{n}$.

En définissant les différentes matrices d'entrées, de données et de poids comme suit :

- Matrice des sorties prédites : $\hat{Y}(k) \equiv [\hat{y}_1(k), \cdots, \hat{y}_n(k)] \equiv \begin{bmatrix} \hat{y}^T(1) \\ \vdots \\ \hat{y}^T(k) \end{bmatrix} \in \mathbb{R}^{k \times n}$
- Matrices des données : $\Phi(k) \equiv \begin{bmatrix} \varphi^T(1) \\ \vdots \\ \varphi^T(1) \end{bmatrix} \in \mathbb{R}^{k \times N_c}$
- La matrice des poids : $\Theta(k) \equiv [\theta_1(k), \cdots, \theta_n(k)] \equiv \begin{bmatrix} \theta_{11} & \cdots & \theta_{1n} \\ \vdots & & \vdots \\ \theta_{N_c1} & \cdots & \theta_{N_cn} \end{bmatrix} \in \mathbb{R}^{N_c \times n}$

la réponse du réseau peut donc être exprimée : $\hat{Y}(k) = \Phi(k)\Theta(k)$

Des études récentes par Poggio et Girosi [POGGIO et GIROSI, 1990] ont montré que la convergence de l'apprentissage peut être améliorée par l'adjonction de contributions directes des entrées aux sommes pondérées des sorties du réseau. Ce qui est avantageux est que l'on peut aussi ajuster les valeurs de ces liens (pondérations modifiables). Ceci est facilement implémentable en modifiant le vecteur d'entrée par : $\varphi'(k) = \begin{bmatrix} \varphi(k) \\ x(k) \\ 1 \end{bmatrix} \begin{matrix} \}N_c \\ \}N_i \\ \}1 \end{matrix}$

Ici les unités d'entrée fournissent les contribution directes. on a donc :

$$\varphi(k) = \begin{bmatrix} \varphi_1(k) \\ \vdots \\ \varphi_{N_c}(k) \\ \varphi_{N_c+1}(k) \\ \vdots \\ \varphi_{N_c+N_i}(k) \\ \varphi_{N_c+N_i+1}(k) \end{bmatrix} = \begin{bmatrix} \varphi_1(k) \\ \vdots \\ \varphi_c(k) \\ x_1(k) \\ \vdots \\ x_{N_i}(k) \\ 1 \end{bmatrix}$$

Le vecteur des poids de connexion doit être redéfinit par :

$$\theta_j(k) = \begin{bmatrix} \theta_{1j}(k) \\ \theta_{2j}(k) \\ \vdots \\ \theta_{N_c+N_i+1j}(k) \end{bmatrix}, j = 1, 2, \cdots, n$$

Les dimensions des vecteurs $\varphi(k)$ et $\theta_j(k)$ changent de N_c à $N_h \equiv N_c + N_i + 1$. Tout ceci étant défini, on doit maintenant entraîner le réseau de neurones RBF.

1.9 Estimateur orthogonal moindre carrés et sélection avant

La méthode classique utilisée pour la modélisation de données par régression est la méthode des moindres carrés linéaires. Une approche alternative à cette méthode classique est la méthode orthogonal des moindres carrés (OLS) pour contourner les problèmes dus au système non-linéaire. Cette méthode contient une détection de structure de modèle en considérant un modèle auxiliaire équivalent dans un espace de vecteurs orthogonal.

Au lieu de considérer l'estimation des paramètres du modèle définit par $y_i(k) = \Phi(k)\theta(k) + e_i(k)$ on utilise le modèle équivalent donné par $y_i(k) = \Omega(k)g_i(k) + e_i(k)$ où on considère $\Omega(k) = [\omega_1(k)\omega_2(k)\cdots\omega_{N_h}(k)]$ avec $\omega_i^T(k)\omega_j = L_{ij}\delta_{ij}, \forall i,j = 1,2,\cdots,N_h$. $g_i(k) \in \mathbb{R}^{N_h}, \delta_{ij}$ est le Delta de Kronecker, $L_{ij} \in \mathbb{R}$ est une constante de normalisation et $e_i(k)$ est un vecteur d'erreur de moyenne nulle. L'estimation du j^{e}composant du vecteur de paramètres du vecteur $g_{ij}, \hat{g}_{ij}(k)$ à l'instant k peut être calculé par $\hat{g}_{ij}(k) = \frac{y_i^T(k)\omega_j(k)}{\|\omega_j(k)\|^2}, 1 \leq j \leq N_h$. De là le jeu de vecteur $\{\omega_j(k); j = 1,2,\cdots,N_h\}$ forme une base orthogonale qui couvre le même espace que celui couvert par le jeu des vecteurs de bases $\{\varphi_j(k); j = 1,2,\cdots,N_h\}$ dans le modèle original de régression. En terme géométrique la valeur $\hat{g}_{ji}(k)\|\omega_j(k)\|$ peut être considérée comme la valeur signée de la magnitude de la projection orthogonale du vecteur $y_i(k)$ observé sur la j $^{\text{e}}$coordonnées du nouvel espace de vecteur.

Le critère de sélection du plus important régresseur depuis l'ensemble complet de régresseurs à inclure dans le modèle auxiliaire peut être pris comme la proportion en pourcentage de l'énergie donnée par chacun des régresseurs. Un tel critère peut être exprimé par : $ERR_i^{(j)}(k) \equiv \frac{\hat{g}_{ij}^2(k)\|\omega_j(k)\|^2}{\|y_i(k)\|^2} \times 100\%$ qui peut être définit comme le rapport d'erreur de réduction (Billing et Chen 1989 [BILLINGS et CHEN, 1989]) pour le j^{e}composant du modèle auxiliaire à l'instant k.

1.10 Apprentissage du réseau basé sur la segmentation par k-moyennes et les moindres carrés récursifs de Givens.

Quand on reformule un problème d'approximation de fonction par les réseaux RBF, la tâche d'apprentissage implique la détermination d'un jeu de centres appropriés, des paramètres de largeur et les poids du réseau associés (Moody, Darken 1989 [MOODY et DARKEN, 1985]). On est souvent amené à utiliser un sous-ensemble de candidats par rapport à l'ensemble de départ pour ces paramètres.

On sait qu'un apprentissage optimal (une erreur minimale) ne peut être atteint qu'avec une réseau important ce qui réduit considérablement la vitesse de convergence de l'algorithme. On est donc obligé de passer par des méthodes de réduction de structure du réseau.

La sélection des centres peut être considérée séparément comme un problème de sélection de sous-modèle qui sélectionne seulement un sous-ensemble de centres appropriés. On passe ensuite à l'optimisation des poids du réseau.

C'est donc un optimisation en deux parties :

- ◦ Sélection des centres

◦ Modifications des poids

1.10.1 Segmentation en k-moyennes des centres

Afin de mettre à jour les centres une méthode de segmentation à k-moyennes basée sur des méthodes de segmentation mono-hiérarchique. Dans sa version la plus simple l'algorithme dirige chaque donnée vers le centre le plus proche. Le nombre de centre étant spécifié l'algorithme effectue ce calcul sur le vecteur d'entrée du réseau à chaque instant et assigne les centres successivement au segment de données dans le domaine de l'espace d'entrée du réseau.

Les centres étant mis à jour par l'algorithme de segmentation, l'algorithme de sélection avant de Givens traite toutes les sorties des unités cachées comme des candidats potentiels pour les centres ou régresseurs dans l'estimation des poids du réseau par les moindres carrés. Seulement le sous-ensemble le plus significatif des sorties de la couche cachée est sélectionné. L'algorithme minimise l'erreur entre la sortie désirée et la sortie actuelle du réseau de neurones en effectuant des séries de transformations orthogonales de Givens. La mise à jour des paramètres est considérée comme une apprentissage supervisé.

Pour simplifier l'algorithme on considérera un système avec un seule sortie. On considère un système avec N_i entrées, une sortie, N_c centres initialisés à des valeurs aléatoires dans le voisinage de l'espace d'entrée. Donné un gain initial de segmentation, $\kappa(0)$, l'algorithme de segmentation à k-moyennes calcul la distance euclidienne $d_i(k)$ entre le vecteur d'entrée courants du réseau $x(k) \in \mathbb{R}^{N_i}$ et le i^{e}centre $c_i(k) \in \mathbb{R}^{N_i}$ pour tous les N_c centres à chaque instant k tel que $d_i(k) = \|x(k) - c_i(k)\|, i = 1, 2, \cdots, N_c$. La mise à jour des centres est faite en fonction de la distance d'éloignement du vecteur d'entrée courant et des centres mis à jour précédemment mesurés par la distance euclidienne. $c_i(k) = c_i(k-1) + \kappa(k)\delta_{ij}\left[x(k) - c_i(k-1)\right], i = 1, 2, \cdots, N_c$ où $j = argmin_{i=1,2,\cdots,N_c}\{d_i(k)\}$ et le gain de segmentation $\kappa(k) \in [0, 1]$ est donné par $\kappa(k) = \frac{\kappa(k-1)}{\sqrt{1+\int\left(\frac{k}{N_h}\right)}}$ où $\int(x)$ représente la partie entière de x. D'autres choix peuvent être pris en considération pour la mise à jour du gain de segmentation comme de choisir une décroissance monotone jusqu'à zéro. L'expérience montre que le choix du gain initial n'est pas prépondérant mais que de bonnes valeurs de sont proches de 1.

1.10.2 L'algorithme d'estimation de sélection avant

Donné un ensemble de centres fixes déterminés par l'algorithme de segmentation, la tâche restante consiste à calculer le vecteur de poids $\theta_i(k) \in \mathbb{R}^{N_h}$ $(1 \leq i \leq n)$ pour chaque sortie du réseau. La détermination de ces poids est en fait un problème de moindres carrés en résolvant l'équation donnée par : $[\Lambda^{\frac{1}{2}}(k)\Phi(k)]^T[\Lambda^{\frac{1}{2}}(k)\Phi(k)]\theta_i(k) = [\Lambda^{\frac{1}{2}}(k)\Phi(k)]y_i(k)$ où

$$\Lambda(k) \triangleq \begin{bmatrix} \lambda^{k-1} & & & 0 \\ & \lambda^{k-2} & & \\ & & \ddots & \\ 0 & & & 1 \end{bmatrix} \equiv \begin{bmatrix} & & & 0 \\ & \lambda\Lambda K-1 & & \vdots \\ & & & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

$\lambda \in (0, 1)$est le facteur d'oubli de l'algorithme récursif et la matrice $\Lambda^{\frac{1}{2}}(k)$ est le facteur de Cholesky de la matrice $\Lambda(k)$. Le principal avantage de cette méthode est la facilité de parallélisation. Le calcul de $\theta_i(k)$ à chaque instant k peut être effectué en deux phases bien séparées basées sur la résolution récursive de : $[\Lambda^{\frac{1}{2}}(k)\Phi(k)]^T[\Lambda^{\frac{1}{2}}(k)\Phi(k)]\theta_i(k) = [\Lambda^{\frac{1}{2}}(k)\Phi(k)]y_i(k)$.

La phase d'apprentissage se résume donc à trois phases :

- Mise à jour des centres
- Sélection des centres
- Estimation des poids

1.10.3 Test de validité des structures

Test de validité de modèle off-line Billing et Zhu [BILLINGS et ZHU, 1995] ont estimé qu'un modèle MIMO est considéré comme représentation acceptable des données si les corrélations suivantes sont satisfaites :

$$\left|\frac{\sum_{k-1}^{N}\alpha(k-\tau)\gamma(k)}{\sqrt{\sum_{k=1}^{N}\alpha^2(k)}\cdot\sqrt{\sum_{k=1}^{N}\gamma^2(k)}}\right| < \frac{1,96}{\sqrt{N}}$$

et

$$\left|\frac{\sum_{k-1}^{N}\beta(k-\tau)\gamma(k)}{\sqrt{\sum_{k=1}^{N}\beta^2(k)}\cdot\sqrt{\sum_{k=1}^{N}\gamma^2(k)}}\right| < \frac{1,96}{\sqrt{N}}$$

où

$$\alpha(k) \triangleq \sum_{j=1}^{n}\frac{\varepsilon_j^2(k)-\frac{1}{N}\sum_{k=1}^{N}\varepsilon_j^2(k)}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[\varepsilon_j^2(k)-\frac{1}{N}\sum_{l=1}^{N}\varepsilon_j^2(l)\right]^2}}$$

$$\beta(k) \triangleq \sum_{i=1}^{m}\frac{u_i^2(k)-\frac{1}{N}\sum_{k=1}^{N}u_i^2(k)}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[u_i^2(k)-\frac{1}{N}\sum_{l=1}^{N}u_i^2(l)\right]^2}}$$

$$\gamma(k) \triangleq \sum_{j=1}^{n}\frac{y_j(k)\varepsilon_j(k)-\frac{1}{N}\sum_{k=1}^{N}y_j(k)\varepsilon_j(k)}{\sqrt{\frac{1}{N}\sum_{k=1}^{N}\left[y_j(k)\varepsilon_j(k)-\frac{1}{N}\sum_{l=1}^{N}y_j(l)\varepsilon_j(l)\right]^2}}$$

Test de validation récursive Le test précédent oblige à avoir toutes les données de $k = 1$ à $k = N$. On peut alors reformuler ce test avec sous une forme récursive en définissant :

$$\rho_1(k) = \eta\rho_1(k-1) + (1-\eta)\times\left[\frac{1}{\tau_{Max}+1}\sum_{\tau=0}^{\tau_{Max}}\left|\frac{\alpha'(k-\tau)-\mu_{\alpha'}(k)}{\sigma_{\alpha'}(k)}\right|\right]\times\left|\frac{\gamma'(k)-\mu_{\gamma'}(k)}{\sigma_{\gamma'}(k)}\right|$$

avec $\rho_1(0) = 0$

$$\rho_2(k) = \eta\rho_2(k-1) + (1-\eta)\times\left[\frac{1}{\tau_{Max}+1}\sum_{\tau=0}^{\tau_{Max}}\left|\frac{\beta'(k-\tau)-\mu_{\beta'}(k)}{\sigma_{\beta'}(k)}\right|\right]\times\left|\frac{\gamma'(k)-\mu_{\gamma'}(k)}{\sigma_{\gamma'}(k)}\right|$$

avec $\rho_2(0) = 0$

où $\alpha'(k) \triangleq \sum_{j=1}^{n}\varepsilon_j^2(k)$ et où $\beta'(k) \triangleq \sum_{i=1}^{m}u_i^2(k)$ et où $\gamma'(k) \triangleq \sum_{j=1}^{n}y_i(k)\varepsilon_j(k)$

$\mu_{\alpha'}(k) = \eta\mu_{\alpha'}(k-1) + (1-\eta)\alpha'(k)$ avec $\mu_{\alpha'}(0) = 0$
$\mu_{\beta'}(k) = \eta\mu_{\beta'}(k-1) + (1-\eta)\beta'(k)$ avec $\mu_{\beta'}(0) = 0$
$\mu_{\gamma'}(k) = \eta\mu_{\gamma'}(k-1) + (1-\eta)\gamma'(k)$ avec $\mu_{\gamma'}(0) = 0$
et
$\sigma_{\alpha'}(k) = \sqrt{\eta\sigma_{\alpha'}(k-1) + (1-\eta)][\alpha'(k) - \mu'(k)]^2}$ avec $\sigma_{\alpha'}(0) = \delta'$
$\sigma_{\beta'}(k) = \sqrt{\eta\sigma_{\beta'}(k-1) + (1-\eta)][\beta'(k) - \mu'(k)]^2}$ avec $\sigma_{\beta'}(0) = \delta'$ $\sigma_{\gamma'}(k) = \sqrt{\eta\sigma_{\gamma'}(k-1) + (1-\eta)][\gamma'(k) - \mu'(k)]^2}$ avec $\sigma_{\gamma'}(0) = \delta'$

où δ' est un très petit nombre positif et η est la facteur d'oubli constant qui contrôle le degrés de compromis entre une détection lente et une trop rapide.

Application à la reconnaissance de forme Il s'agit de la classification des formes en différentes classes : le mode de classification est toujours le même. On obtient un ensemble de données par des mesures, et des règles de classification place une donnée présentée à sa classe avec un minimum d'erreur. Le rôle du réseau de neurones peut être vu comme l'extraction de cette base de règles d'extraction. Mais comme le problème posé à toujours un nombre fini de classes, les règles deviennent alors non-linéaires (exemple du XOR) (si linéaire 4 classes mais on en impose 2).

- Application à la modélisation de système SISO non-linéaire.
- Application des RBF à la modélisation d'un niveau d'eau non-linéaire.
- Application à la modélisation de système MIMO non-linéaire.

Identification d'un système à 2 entrées et 2 sorties régit par les équations suivantes :

$$\left.\begin{array}{rcl} y_1(k) & = & 0.5y_1(k-1) + u_1(k-2) \\ & + & 0.1y_2(k-1)u_1(k-1) + 0.5e_1(k-1) \\ & + & 0.2y_1(k-2)e_1(k-2) + e_1(k) \\ y_2(k) & = & 0.9y_2(k-2) + u_2(k-1) \\ & + & 0.2y_2(k-1)u_2(k-2) + 0.5e_2(k-1) \\ & + & 0.2y_2(k-1)e_2(k-2) + e_2(k) \end{array}\right\}$$

où $\{u_1(k)\}$ est une séquence gaussienne de moyennes nulle et de variance unitaire. $\{u_2(k)\}$ est un séquence uniformément distribuée sur $\left]-\sqrt{3}, \sqrt{3}\right]$ et $\{e_1(k)\}$,$\{e_2(k)\}$ sont des séquences gaussiennes de moyenne nulle et de variance 0.04.

1.11 Différents algorithmes d'apprentissage pour les réseaux RBFs

Comme précédemment l'apprentissage se décompose en deux phases :

- Apprentissage des centres et des largeurs de la couches cachées
- Apprentissage des poids des connexions entre la couche cachée et la couche de sortie

Ces algorithmes d'apprentissage utilisent le plus souvent des méthodes de segmentation pour partitionner l'espace d'entrée. La structure du réseau, elle, est déterminée.

- Travail de Chen [CHEN *et al.*, 1990a] [CHEN *et al.*, 1990b]: L'apprentissage se fait par l'utilisation de l'algorithme orthogonal des moindres carrés. Le critère d'information d'Akaike est utilisé pour déterminer le nombre d'unité sur la couche cachée et un rapport de réduction d'erreur est utilisé pour la sélection des centres tels que les erreurs d'approximation du réseau soient réduites à chaque étape de sélection.
- Travail de Hocomb et Morari [HOLCOMB et MORARI, 1991] : Cet algorithme

traite les RBFs associées aux neurones de la couche cachée comme approximativement orthogonal entre elles. Le réseau commence avec un seul neurone et des unités sont ajoutées quand cela est nécessaire.
- Travail de Lee et Rhee [LEE et RHEE, 1991], Musavi [MUSAVI *et al.*, 1992] : ces algorithmes sont basés sur des méthodes de segmentation hiérarchiques supervisées.
- Dans les travaux de Lee et Rhee l'apprentissage commence avec un seul neurone sur la couche cachée avec une grande largeur : des unités sont ajoutées si nécessaire. Les largeurs associées ainsi que les emplacements sont modifiés.
- Dans les travaux de Muravi, l'apprentissage débute avec un large nombre d'unités sur la couche cachées que l'on fusionne quand cela est possible. Les largeurs et les emplacements sont modifiés en conséquence.

Tous ces algorithmes ne peuvent sélectionner que le unités de la couches cachées. On peut aller plus loin et proposer un algorithme qui sélectionne aussi les entrées en tenant compte de la pertinence de la sortie du réseau.

1.12 Information mutuelle et sélection des unités

L'information mutuelle est une mesure de l'information fondamentale dans la théorie de l'information. Pour les réseaux de neurones Linsker (1989)?????Reference????? a dérivé une règle d'apprentissage en maximisant l'information mutuelle entre un signal d'entrée et un signal de sortie afin de générer une structure ordonnée. L'information mutuelle entre les entrées et les sorties de la couche cachée a été introduite par (Deco 1995)?????Reference????? dans l'algorithme de rétropropagation qui pénalise l'extraction massive d'information par les unités de la couche cachée et empêche le surdimensionnement du réseau.

En 1994 Battiti ?????Reference????? applique l'information mutuelle afin d'effectuer une sélection de caractéristiques de formes pour des applications de reconnaissances de formes.

1.12.1 L'information mutuelle

Soit Y une variable aléatoire avec un ensemble de N_y sorties $Y_1, Y_2, \cdots, Y_{N_y}$ avec les probabilités $P(Y_j) = p_j (j = 1, 2, \cdots, N_y)$. Soit U une variable aléatoire avec un ensemble de N_u sorties $U_1, U_2, \cdots, U_{N_u}$ avec les probabilités $P(U_k) = q_k (k = 1, 2, \cdots, N_u)$. La connexion entre les deux variables est obtenu en exprimant leur probabilités jointes $P(Y_j \cap U_k) = p_{jk} (j = 1, 2, \cdots, N_y, k = 1, 2, \cdots, N_u)$.

En général l'incertitude de la variable Y est réduite à la suite de la connaissance de la variable U si elle sont statistiquement jointes. La quantité de réduction de cette incertitude est $I(Y, U) = H(Y) - (H(Y/U)) = \sum_{j=1}^{N_Y} \sum_{k=1}^{N_U} p_{jk} \log\left(\frac{p_{jk}}{p_j q_k}\right)$ ou $I(Y/U)$ est l'information mutuelle, $H(Y)$ est l'entropie, une mesure de l'incertitude de la variable Y, $H(Y/U)$ est l'entropie conditionnelle, une mesure de l'incertitude la variable Y sachant U. On peut montrer que $0 \leq I(Y/U) \leq H(Y)$. L'information mutuelle est nulle seulement si les deux variables sont indépendantes et l'information mutuelle vaut $H(Y)$ quand les deux variables sont parfaitement associées.

L'information mutuelle est souvent considérée comme une mesure de la dépendance entre deux variables. Une autre alternative consiste à considérer l'information mutuelle comme une mesure de l'information d'une variable par rapport à une autre ou encore une mesure du degrés de prédiction d'une variable de sortie en connaissant la variable d'entrée.

On peut remarquer que l'information mutuelle mesure la dépendance entre variables en intégrant les non-linéarités. Ceci peut donc être utilisé dans l'apprentissage des réseaux de neurones. On peut donc voir l'apprentissage du réseau de neurones comme une extraction d'information. Il s'agit de l'apprentissage des relations entre les données d'entrées et les données de sorties.

Il est clair qu'il sera intéressant de supprimer les données d'entrées sans rapport avec la sortie. Car les prendre en compte augmenterait le coût de calcul. On suppose un ensemble S_m avec m variables d'entrée. La tâche consiste à trouver un sous-ensemble S_n avec $n < m$ variables d'entrées qui comporte le plus d'information sur la sortie Y. on veut donc : $\bar{S}_n = \max_{S_n \subset S_m} I(Y, S_n)$. Le nombre possible de sous-ensemble de S_n avec n variables d'entrées est $\begin{pmatrix} m \\ n \end{pmatrix}$. Si on considère tous ces sous-ensembles les calculs sont très lourds et le nombre de données pour l'extraction de l'information mutuelle sera vraiment très grand. On diminue ce nombre d'échantillons à 1000 voir 100. Une autre alternative consiste à calculer une information mutuelle d'un degrés moindre et de construire le sous-ensemble pas à pas. Pour qu'une variable soit sélectionnée il semble logique qu'elle satisfasse les critères heuristiques suivant :

- Elle ne doit pas être très dépendante d'une autre variable sélectionnée (sinon elle ne contient que peut d'information supplémentaire sur la sortie et peut être redondante)
- Elle doit être informative sur la sortie

1.12.2 Algorithme de sélection d'unités d'entrées

Il semble bien adapté au problème de reconnaissance de forme, mais inadéquate pour l'identification de systèmes. En effet dans ce type de problème les sorties futures du réseaux sont très dépendantes des entrées et des sortie précédentes.

Cette très grande dépendance met à défaut l'algorithme que ne peut sélectionner un sous-ensemble représentatif. une modification de l'algorithme permet de passer cette limitation. On peut en effet considérer qu'une très grande information mutuelle (en valeur) peut déterminer les retards dans un système. On va utiliser cette caractéristiques dans la modélisation de systèmes complexes.

- Application à la reconnaissance de forme
- Application à l'identification de système

1.13 Réseau de neurones RBF et commande basée sur la logique floue

Les réseaux RBF sont une alternative aux réseaux de neurones de type perceptron dans les application basées sur les réseaux de neurones. Leurs principaux avantages résident dans la vitesse d'apprentissage plus rapide (quand la tâche à apprendre est formulée de façon linéaire) et leur capacité d'effectuer le rôle de système d'inférence floue. Les réseau RBF sont intéressant car les neurones individuels de la couche cachées correspondent exactement à une règle floue elle aussi unique.

1.13.1 Plusieurs méthodes

Voici un récapitulatif des méthodes utilisées :

- Méthodes classiques : centres déterminé de manière aléatoire, largeurs uniformes, poids de la couches de sortie déterminé par une méthode pseudo-inverse. Réseau de neurones RBF classique : On spécifie le nombre d'unités cachées le seul paramètres. *Les centres des gaussiennes sont initialisés à des valeurs aléatoires de l'espaces des échantillons. La largeurs est uniformes et est donnée par $\sigma = \frac{d}{\sqrt{2N}}$ ou d est la distance euclidienne maximale entre chaque paire de centres choisis. Les poids de la couche de sortie sont optimisés par la méthode pseudo-inverse (SVD).
- Méthode de Moody et Darken [MOODY et DARKEN, 1985] : détermination des centres par la segmentation k-means, détermination des largeurs par une méthode heuristique des P plus proches voisins, et détermination des poids par une méthodes des moindres carrés. L'apprentissage des paramètres est hybride car la couche cachée est optimisée par une voie d'auto-organisation. Pendant que la couche de sortie est optimisée par un superviseur. Le nombre N d'unités cachée doit être défini. L'algorithme de segmentation à k-moyennes et utilisé pour déterminer les centres de la couches cachée. Les largeurs de RBF sont déterminées par une méthode heuristique des P plus proches voisins. Les poids de la couche de sortie sont optimisés par une méthode des moindres carrés.
- Méthode d'estimation des segments ou méthode Poggio et Girosi [POGGIO et GIROSI, 1990] modifiée, méthode de Mountain modifiée basé sur un algorithme de segmentation par force d'attraction (attracteurs). Pas besoin de spécifier le nombre d'unités cachées. La méthode calcul une mesure du potentiel de chaque échantillons comme une fonction de la distance à tous les autres points de l'échantillon d'apprentissage définie par $P_i = \sum_{j=1}^{N} \exp\left(-\alpha\left(x_i - c_j\right)^2\right)$ où $\alpha = \frac{4}{r_a^2}, r_b > 0$.
  L'échantillon avec le plus fort potentiel est choisi. Si le potentiel est plus bas que le minimum requis la méthode est arrêtée. Si elle est supérieur à la valeur bornée un nouveau neurone est ajouté à la couche cachée. Son centre est initialisé à la valeur actuel de l'échantillon traité. D'une autre manière des critères additionnels de décision sont utilisés. En général à la suite de l'obtention du i^{e}centre chaque échantillon est révisé par la formule $P_i = P_i - P_k^* \exp\left(-\beta\left(x_i - x_k^*\right)^2\right)$ où $\beta = \frac{4}{r_b^2}, r_b > 0$ ou P_k^* est le potentiel du k^{e}centre de segment x_k^2 et $r_b > r_a$. Les largeurs des RBF sont déterminées par une méthode heuristiques des P plus proches voisins et les poids de la couche de sortie sont optimisés par un algorithmes de types moindres carrés.
- Méthode Poggio et Girosi modifiée [POGGIO et GIROSI, 1990] : ce modèle a ses racines dans la théorie de la régularisation et l'utilise comme la simplification de ses solutions une gaussienne généralisée pour les neurones de la couche cachée : $\varphi(x) = \exp\left(-\frac{1}{2}(x-c)^T K^{-1}(x-c)\right)$ ou x est le vecteur d'entrée, c est le centre de la RBF, et $\frac{1}{K}$ est l'inverse de la matrice de covariance.
- Algorithme original de segmentation : il est basé sur la force d'attraction entre deux points de masse dans les lois newtoniennes. la force d'attraction entre deux points de masse a et b séparés par une distance c est donnée par $F = G\frac{ab}{c^2}$ et G est la constante de gravitation. Cette procédure détermine la positions des centres pour l'apprentissage adaptatif des réseaux RBF. Le premier échantillon est le premier centre de segment et sa masse est initialisée à 1. Pour chaque nouvel échantillon x_p la force d'attraction est calculée par rapport à tous les segments existants. Le segment avec le plus grande valeur est recherche si il est supérieur au seuil F^* on

met à jour sa position par l'équation $c_i = \frac{m_i c_i + x_p}{m_i + 1}$.

Utilisation à simulation de la commande de parking d'un camion dans son emplacement.

Il est bien connu que la modélisation et la commande de système ne peut par être exacte pour les systèmes complexes réels. Ces systèmes sont actuellement commandés par des humains sans l'aide de modèles mathématiques, d'algorithmes ou d'un savoir approfondie sur le système mis en jeu. Les contrôleurs humains dérivent intègrent un modèle qualitatif du procédés et produisent des actions ou réponses basées sur des valeurs qualitatives à des situations données qui sont vaguement paramètrées par les conditions opérationnelles. Le problème consiste donc à représenter et à calculer des procédés imprécis. Cf. Travail de Zadeh calcul de la logique floue.

Implémentation de la commande basée sur a logique floue par une réseau RBF. Deux approches :

- Utilisation d'un réseau RBF flou pour implémenter directement les inférences floues du système utilisant des entrées et des sorties floues
- Utilisation d'un réseau RBF précis (par opposition à flou) ou les entrées et les sorties sont des valeurs réelles et le réseau sera entraîné par pour produire des sorties défuzzifiées en utilisant un superviseur d'apprentissage basé sur des règles floues

Bibliographie

[BAKER et FARRELL, 1992] W. BAKER et J. FARRELL. *An introduction to connectionist learning control.* D. WHITE & D. SOFGE. Van Nostrand Reinhold, 1992.

[BERTHOLD, 1994] M. R. BERTHOLD. A time delay radial basis function network for phoneme recognition. In *Proceedings of the international conference on Neural Networks*, Orlando, 1994. Intel Corporation, Santa Clara.

[BILLINGS et CHEN, 1989] S. A. BILLINGS et S. CHEN. Extended model set, global data and threshold model identification of severely non-linear systems. In *International Journal of Control*, volume 50, pages 1897–1923, 1989.

[BILLINGS et ZHU, 1995] S. A. BILLINGS et Q. M. ZHU. Model validation tests for multivariable non-linear models including neural networks. In *International Journal of Control*, volume 62, pages 749–766, 1995.

[BROOMHEAD et LOWE, 1988] D. S. BROOMHEAD et D. LOWE. Multivariate functionnal interpolation and adaptive networks. In *Complex Systems*, volume 2, pages 321–355, 1988.

[CHEN *et al.*, 1990a] S. CHEN, S. A. BILLINGS, C. F. N. COWAN, et P. M. GRANT. Non-linear systems identification using radial basis functions. In *International Journal of Systems Science*, volume 21, pages 2513–2539, 1990.

[CHEN *et al.*, 1990b] S. CHEN, S. A. BILLINGS, C. F. N. COWAN, et P. M. GRANT. Practical identification of narmax models using radial basis functions. In *International Journal of Control*, volume 52, pages 1327–1350, 1990.

[FREEMAN et SAAD, 1996] J. A. FREEMAN et D. SAAD. Radial basis function network : Generalization in over-realisable and unrealisable scnenarios. In *Neurals Networks*, volume 9.9, pages 1521–1529, 1996.

[FUNG *et al.*, 1996] C. F. FUNG, S. A. BILLINGS, et W. LUO. On-line supervised adaptative training using radial basis function networks. In *Neurals Networks*, volume 9.9, pages 1597–1617, 1996.

[GABRIJEL *et al.*,] I. GABRIJEL, A. DOBNIKAR, et N. STEELE. Rbf neural networks and fuzzy logic based control - a case study.

[GABRIJEL et DOBNIKAR,] I. GABRIJEL et A. DOBNIKAR. Adaptative rbf neural network.

[HARDY, 1971] R. L. HARDY. Multiquadric equations of topography and other irregular surfaces. In *Journal of geophysical research*, volume 76.8, pages 1905–1915, 1971.

[HARP et SAMAD, 1992] S. HARP et T. SAMAD. *Genetic synthesis of neural network architecture.* L. DAVIS. Van Nostrand Reinhold, New-York, 1992.

[HARTMAN *et al.*, 1990] F. J. HARTMAN, J. D. KEELER, et J. M. KOWALSKI. Layered neural networks with gaussian hidden units as universal approximators. In *Neural Computation*, volume 2, pages 210–215, 1990.

[HOLCOMB et MORARI, 1991] T. HOLCOMB et M. MORARI. Local training for radial basis function networks. toward solving the hidden unit problem. In *Proceedings of American Control Conference*, pages 2331–2336, 1991.

[HWANG et BANG, 1996] Y. S. HWANG et S. Y BANG. An efficient method to construct a radial basis function neural networ classifier. In *Neurals Networks*, volume 10.8, pages 1495–1503, 1996.

[HWANG et BANG, 1997] Y. S. HWANG et S. Y BANG. An efficient method to construct a radial basis function neural networ classifier. In *Neurals Networks*, volume 10.8, pages 1495–1503, 1997.

[LEE et RHEE, 1991] S. LEE et M. K. RHEE. A gaussian potential function network with hierarchically self-organizing learning. In *Neural Network*, volume 4, pages 207–224, 1991.

[MOODY et DARKEN, 1985] J. E. MOODY et C. J. DARKEN. Fast learning in networks of locally tuned processing units. In *Neural Computations*, volume 1, pages 281–294, 1985.

[MUSAVI *et al.*, 1992] M. T. MUSAVI, W. AHMED, K. H. CHANG, K. B. FARIS, et D. M. HUMMELS. On the training of radial basis function classifiers. In *Neural Networks*, volume 5, pages 595–603, 1992.

[ORR,] M. J. L. ORR. Introduction to radial basis function networks.

[PARISINI et ZOPPOLI, 1993] T. PARISINI et R. ZOPPOLI. Radial basis functions and multilayer feedforward neural networks for optimal control of non-linear stochastic systems. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1853–1858, San Francisco, 1993.

[PARK et SANDBERG, 1991] J. PARK et I. SANDBERG. Universal approximation using radial-basis function networks. In *Neural Computations*, volume 3, pages 246–257, 1991.

[POGGIO et GIROSI, 1990] T. POGGIO et F. GIROSI. Network for approximation and learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1496, 1990.

[POWELL, 1985] M. J. D. POWELL. Radial basis function for multivariable interpolations : A rewiew. In *IMA conference Algorithms for approximation of functions and data*. RMCS, 1985.

[RENDERS,] J.M. RENDERS. *Algorithmes Génétiques et Réseaux de Neurones*. Collections Systèmes Complexes. Edition Hermès.

[UNAR et MURRAY-SMITH,] M. ALI UNAR et D. J. MURRAY-SMITH. Radial basis function networks for ship steering control.

[ZHENG et BILLING, 1996] G. L. ZHENG et S. A. BILLING. Radial basis function network configuration using mutual information and the orthogonal least squares algorithm. In *Neurals Networks*, volume 9.9, pages 1619–1637, 1996.