L'intégration des méthodes Monte Carlo par chaînes de Markov dans l'apprentissage des réseaux de neurones

Linda OTMANI [1] et Abdelkader BENYETTOU [2]

[1] Université de sciences et de technologie d'ORAN-Mohamed Bodiaf-
Faculté des sciences-Département d'informatique-Laboratoire SIMPA
otmani_linda@yahoo.fr

[2] Université de sciences et de technologie d'ORAN-Mohamed Bodiaf-
Faculté des sciences-Département d'informatique-Laboratoire SIMPA
aek.benyettou@univ-usto.dz

Résumé : Les méthodes Monte Carlo par chaînes de Markov (MCMC) , s'inscrivent dans le cadre du formalisme Bayésien .On peut dire que l'application de ces méthodes à l'apprentissage des réseaux de neurones peut apporter plusieurs avantages, tout d'abord, l'introduction de connaissances a priori est susceptible d'améliorer l'estimation des paramètres. De plus, le formalisme Bayésien permet une analyse complète des incertitudes et des probabilités des données. Dans cet article, nous présenterons brièvement le principe de ces méthodes, qui présentent l'avantage d'être utilisables, et nous allons montrer leurs applications pour l'apprentissage des réseaux de neurones.

Mot clés
Méthodes Bayésiennes, Réseaux de neurones, Méthodes Monte Carlo par chaînes de Markov (MCMC),

1 Introduction

Les méthodes bayésiennes ont été appliquées ces dernières années aux réseaux de neurones par différents auteurs, notamment dans les travaux de MacKay 1992 a [1], MacKay 1992 b[2],Neal 1992[3] , Neal 1994[4], repris dans Neal 1996 [5], et Buntine et Weigened 1991, Bishop 1995, et plus récemment Freitas 1998, Vehtari 1999 et Freitas 2000.

Weigened (1991), Mackay (1992) et Neal (1994) ont montré que les méthodes bayésiennes pour l'apprentissage des réseaux de neurones peuvent apporter plusieurs avantages, car il n'est pas nécessaire de limiter la taille du réseau pour éviter le sur ajustement, et que le nombre de neurones cachés peut tendre vers l'infini, le seul facteur qui doit limiter la taille du réseau est la capacité des ordinateurs utilisés et le temps disponible pour effectuer les calculs nécessaires, mais comme les paramètres utilisés sont issus d'une distribution de probabilité, il est nécessaire pour connaître un paramètre de calculer des intégrales faisant intervenir les distributions des autres paramètres.

Il est, en général, impossible de calculer ces intégrales analytiquement, et plusieurs approches ont été proposées pour effectuer ces calculs. Mais soit ces méthodes sont très lourdes à implémenter, soit elles reposent sur des approximations qui peuvent fausser les résultats, et cela à cause des paramètres utilisés (paramètres du réseau de neurone) qui sont issus d'une distribution de probabilité, pour inférer ces paramètres on est confronté soit à un problème d'intégration ou d'optimisation comme celui de notre cas.

Pour résoudre ce problème nous avons opté pour l'algorithme appelé «reversible Jump MCMC Simulated Annealing» proposé par [6] [10] et qui s'inscrit toujours dans le cadre des méthodes MCMC , et nous allons par la suite vérifier son application pour l'approximation des fonctions.

2 L'inférence Bayésienne

Dans un formalisme Bayésien, toute inférence repose sur la densité à posteriori des paramètres qui nous intéressent conditionnellement aux observations. [7]
Supposant que $y = (y_1, ..., y_N)$ soit le vecteur des observations et $\theta = (\theta_1, ..., \theta_d) \in \Theta$ soit le vecteur de paramètres à estimer.
La densité ***à posteriori*** $p(\theta \backslash y)$ conjointe de tous les paramètres du modèles est définie à partir du théorème de Bayes :

$$p(\theta / y) = \frac{p(y/\theta) \quad p(\theta)}{p(y)} \qquad (1)$$

$p(y\backslash\theta)$ est la densité de probabilité conditionnelle des observations connaissant les paramètres du modèle : fonction de ***vraisemblance***. [8] [9]
$p(\theta)$ est la densité ***à priori*** des paramètres θ choisie en fonction des connaissances disponibles sur θ avant la prise en compte des observations.
$p(y)$ est une constante, indépendante de θ, aussi appelée ***évidence*** Bayésienne .

1.1 Estimation paramétrique Bayésienne

1.1.1. Estimateur (MMSE)

L'estimateur MMSE est défini par la moyenne de la densité à posteriori considérée.
Etant donné un vecteur de paramètres θ et un vecteur d'observation y on a :

$$\hat{\theta}_{MMSE} = \int_{\Theta} \theta \; p(\theta / y) \; . \; d\theta \qquad (2)$$

1.1.2. Estimateur MAP

L'estimateur MAP est déterminé par le mode de la densité à posteriori considérée :

(3)

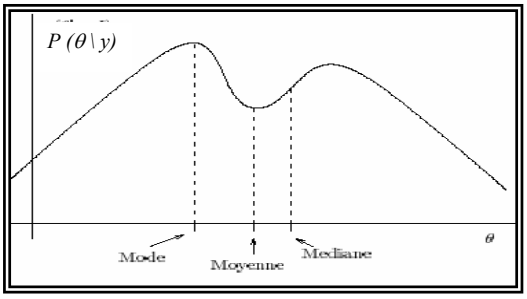


Figure1 : Estimateurs paramétriques Bayésien

Pour résoudre un tel problème, nous proposons une méthode de simulation de densités de probabilité telle que la méthode Monte Carlo par Chaîne de Markov, basée sur la génération de variables aléatoires distribuées suivant une loi π à simuler.

2 Le modèle adopté

Pour réaliser notre travaille, nous allons adopter le schéma d'approximation proposé par Holmes et Mallicke (1998) [10], qui consiste à mixer les k RBFs et la régression linéaire. Ce modèle est donné par :

M_0: $y_t = b + \beta^{-1} x_t + n_t \quad k=0$; (4)

M_k: $y_t = \sum_{j=1}^{k} a_j \phi(\|x_t - \mu_j\|) \; b + \beta^{-1} x_t + n_t \quad k \geq 1$ (5)

D'où $1 \leq k \leq k_{max}$
$\|.\|$: distance (euclidienne ou de mahalanobis).
$\mu_j \in R^d$: $j^{ème}$ centre RBF pour un modèle avec k RBFs,
$a_j \in R^c$: $j^{ème}$ coefficient (poids) RBF,

$b \in R^c$ et $\beta \in R^d \times R^c$: paramètres de la régression linéaire,
$n_t \in R^c$: séquence de bruit.
ϕ :Une fonction RBF, (fonction gaussienne).
Le schéma suivant représente notre modèle adopté

$$y = D\left(\mu_{1:k,1:d}, x_{1:N,1:d}\right)\alpha_{1:1+d+k,1:c} + n_t$$

⇨ Le traitement du bruit est normalement (6) distribué comme suit :

$$n_t \sim \mathrm{N}\left(\begin{bmatrix}0\\0\\.\\.\\.\\0\end{bmatrix}, \begin{bmatrix}\sigma_1^2 & 0 & .... & 0\\ 0 & \sigma_2^2 & .... & 0\\ & & . & \\ & & . & \\ & & . & .\\ 0 & 0 & .... & \sigma_c^2\end{bmatrix}\right)$$

Un bruit additif, blanc, gaussien, de moyenne nulle et variance = σ^2 .
⇨ Les inconnus sont :

✓ Le nombre de RBFs : *k,*
✓ Les paramètres des *k* RBFs : α*, μ et* σ^2.

3 Position du problème

Etant donné l'ensemble de données d'entrée /sortie $\{x, y\}$ tel que :
$O = \{x_1, x_2, \ldots, x_N ; y_1, y_2, \ldots, y_N\}$:
Notre objectif est d'estimer *k* et $\theta \in \Theta_k$. D'où
$\Theta_0 \triangleq (R^{d+1})^c \times (R^+)^c$, et
$\Theta_k \triangleq (R^{d+1+k})^c \times (R^+)^c \times \Omega_k$ pour $k \in \{1, \ldots, k_{max}\}$.
C à d $\alpha \in (R^{d+1+k})^c$; $\sigma \in (R^+)^c$ et $\mu \in \Omega_k$
On note que :
↳ Le nombre maximal de RBFs est défini par :

↳ Ω_k es $k_{\max} = (N - (d+1))^2$. (7)

$$\Omega_k = \{\mu ; \mu_{1:k,i} \in [\min x_{1:N,i} - \iota\,\Xi_i, \max x_{1:N,i} + \iota\,\Xi_i]$$

pour $i = 1, \ldots, d\}$ (8)

ι : un paramètre utilisateur. La prémisse ici est de placer les fonctions où les données sont condensées. Les centres sont échantillonnés d'un espace dont l'hyper volume est [Freitas 98]

$$\psi^k = \left(\prod_{i=1}^{d}(1+2\iota)\,\Xi_i\right)^k$$

avec

$$\Xi_i = \left\| \max\left(x_{1:N,i}\right) - \min\left(x_{1:N,i}\right) \right\|$$

4 Les MCMC pour l'apprentissage bayésien

L'inférence de *k* et Θ_k est basée sur la distribution jointe à posteriori $p(k, \Theta_k \backslash x, y)$, obtenue par le théorème de Bayes. Notre but est d'estimer cette distribution pour obtenir « théoriquement » tous les éléments du posterior.
L'idée principale des MCMC, est de construire une chaîne de Markov dont la distribution stationnaire est la distribution à posteriori désirée $p(k, \Theta_k \backslash x, y)$. $\left(k^{(i)}, \Theta_k^{(i)}\right)_{i \in \mathrm{N}}$
Comme nous l'avons déjà dit, une méthode MCMC simple n'est pas capable de « *sauter* » entre les sous espaces de Θ_k (de dimensions différentes). Cependant, récemment Green a introduit une nouvelle classe flexible, d'échantillonneurs MCMC, appelée « *reversible jump MCM C»* [10]. Capable de sauter entre les espaces des paramètres de dimensions différentes. [11].

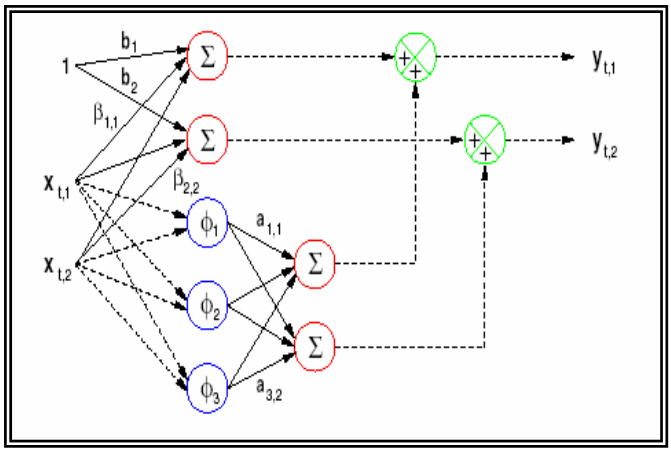


Figure2 : Le modèle linéaire d'approximation d'un RBF à trois fonctions RBF, deux entrées et deux sorties.1

Une stratégie de *recuit simulé* peut être adoptée à cet l'algorithme, pour tirer des échantillons aléatoires à partir de la chaîne de Markov. Ces échantillons sont utilisés pour approximer l'inférence désirée. [10] [12] [13].

$$p\left(k, \mu_1, ..., \mu_k / x, y\right) \propto \left[\prod_{i=1}^{c} \left(y_{1:N,i}^T \; P_{i,k}^* \; y_{1:N,i} \right)^{-1/2} \right] p(k)$$

4.1. Estimation des poids

Étant donné $k, \mu 1, \ldots, \mu k$, l'estimation des moindres carrés de α est donnée par :

$$\hat{\alpha}_{1:m,i} = \left[D^T (\mu_{1:k}, x) D (\mu_{1:k}, x) \right]^{-1} D^T (\mu_{1:k}, x) \; y_{1:N,i} \qquad (9)$$

4.2. Estimation des paramètres du bruit

En utilisant l'estimation conventionnelle de la distribution gaussienne, l'estimation de σ^2 est donnée par :

$$\hat{\sigma}_i^2 = \frac{1}{N} \left(y_{1N,i} - D\left(\hat{\mu}_{1k}, x\right) \hat{\alpha}_{1m,i} \right)^T \left(y_{1N,i} - D\left(\hat{\mu}_{1k}, x\right) \hat{\alpha}_{1m,i} \right) = \frac{1}{N} y_{1N,i}^T \; P_k^* \; y_{1N,i} \qquad (10)$$

D'où $P_{i,k}^*$ est une matrice orthogonale de projection des moindres carrés :

$$P_k^* = I_N - D\left(\hat{\mu}_{1:k}, x\right) \left[D^T\left(\hat{\mu}_{1:k}, x\right) D\left(\hat{\mu}_{1:k}, x\right) \right]^{-1} D^T\left(\hat{\mu}_{1:k}, x\right) \qquad (11)$$

4.3. La distribution jointe à posteriori

➢ On peut imposer la distribution à priori sur k :

$$P(k) \propto \exp\left[-P\right] \qquad (12)$$

D'ou P est un terme de pénalité qui dépend de l'ordre du modèle. En se basant sur le critère du *minimum description length* (MDL),

$P_{\text{MDL}} = \xi/2 \; log \; (N)$.

$\xi =$ nombre de paramètres du modèle $= k(c+1) + c(1+d)$ en cas d'un réseau RBF.

$$P(k) = \exp\left[-\left(\frac{k(c+1) + c(1+d)}{2} \right) \log N \right] \qquad (13)$$

➢ La fonction de vraisemblance est comme suit :

$$p \propto \left[\prod_{i=1}^{c} \left(y_{1:N,i}^T \; P_{i,k}^* \; y_{1:N,i} \right)^{-N/2} \right] \qquad (14)$$

➢ Sachant que les échantillons du bruit sont gaussiens, la distribution jointe à posteriori $p\ (k, \mu_1, \ldots, \mu_k \backslash x,y)$ est donnée par : Freitas 98

$$p\left(k, \mu_1, ..., \mu_k / x, y\right) \propto \left[\prod_{i=1}^{c} \left(y_{1:N,i}^T \; P_{i,k}^* \; y_{1:N,i} \right)^{-1/2} \right] p(k) \qquad (15)$$

➢ Par conséquent l'évaluation du maximum à posteriori (MAP) de ces paramètres est obtenue par la maximisation du côté droit de (Equ.15).

$$p\left(k, \mu_{1k}\right)_{MAP} \propto \underset{k,\mu_i,k \in \Omega}{\operatorname{argmax}} \left[\prod_{i=1}^{c} \left(y_{1:N,i}^{T} \; P_{i,k}^{*} \; y_{1:N,i} \right)^{-1/2} \right] \exp\left[-\left(\frac{k(c+1)+c(1+d)}{2} \right) \log N \right] \quad (16)$$

Nous allons utiliser l'algorithme Reversible jump Markov Chain Monte Carlo Simulated Annealing (R-J-MCMC SA), pour estimer jointement l'ordre du modèle k ($k <= k$ max) et les centre RBF $\mu 1, \ldots, \mu k$, à chaque itération. [14] [15]

5. L'algorithme R-j-MCMC simulated annealing

1. Initialisation de: $(k^{(0)}, \theta^{(0)}) \in \Theta$;
2. Itération i
- ✧ Echantillonner $u \sim$ и $_{[0, 1]}$ et initialiser la température ;
- ✧ Si $\left(u \le b_{k(i)} \right)$ Alors « ***birth*** » ;

Sinon si $\left(u \le b_{k(i)} + d_{k(i)} \right)$ Alors « ***death*** » ;

Sinon si $\left(u \le b_{k(i)} + d_{k(i)} + s_{k(i)} \right)$ Alors «***split*** » ;

Sinon si $\left(u \le b_{k(i)} + d_{k(i)} + s_{k(i)} + m_{k(i)} \right)$ Alors «***merge*** » ;

Sinon mettre à jour les centres RBF ; Fin si
- ✧ Réaliser une étape MH avec le rapport d'acceptation recuit.

1. $i \leftarrow i + 1$ et aller à 2 ;
2. Calculer les coefficients $\alpha_{1:m}$;

5.1. Les sauts de l'algorithme

Supposons que l'état courant de la chaîne de Markov est (k, θ_k). A chaque itération, cet algorithme effectue l'un des sauts suivants : [10] [16]

5.1.1. Saut de naissance « Birth jump»

- Proposer aléatoirement un nouveau centre RBF à partir de l'intervalle (**Equ.8**) ;
- Evaluer A_{birth}, et échantillonner $u \sim$ и $_{[0, 1]}$;
- Si $u \le A_{birth}$ alors l'état de la chaîne de Markov devient $(k+1, \mu_{1:k+1})$ sinon elle reste $(k, \mu_{1:k})$.

$$r_{birth} = \prod_{i=1}^{c} \left[\left(\frac{y_{1:N,i}^{T} \; p_{i,k}^{*} \; y_{1:N,i}}{y_{1:N,i}^{T} \; p_{i,k+1}^{*} \; y_{1:N,i}} \right)^{N/2} \right] \frac{\psi \;\; \exp\left(-C\right)}{(k+1)}$$

d'où p_k^* est donné dans (11).

$C = [(c+1) \log (N)/2]$ selon le critère MDL.

$A_{birth} = \min (1, r_{birth})$.

5.1.2. Saut de mort « death jump »

- Supprimer aléatoirement un des k centres RBF dans (Equ.5);
- Evaluer A_{death}, et échantillonner $u \sim$ и $_{[0, 1]}$;
- Si $u \le A_{death}$ alors l'état de la chaîne de Markov devient $(k-1, \mu_{1:k-1})$ sinon elle reste $(k, \mu_{1:k})$.

$$r_{death} = \prod_{i=1}^{c} \left[\left(\frac{y_{1:N,i}^{T} \; p_{i,k}^{*} \; y_{1:N,i}}{y_{1:N,i}^{T} \; p_{i,k-1}^{*} \; y_{1:N,i}} \right)^{N/2} \right] \frac{k \;\; \exp\left(C\right)}{\psi}$$

$A_{death} = \min (1, r_{death})$.

5.1.3. Saut de scission « split jump »

- Choisir aléatoirement un des centres RBF μ ;
- Le remplacer par ses voisins les plus proches $\mu 1, \mu 2$ tel que $\mu 1 = \mu - u_s \zeta$ et $\mu 2 = \mu + u_s \zeta$;
- Le nouveau centre doit être lié à l'espace Ω_k dans (Equ.8) ;

ζ est une constante (paramètre de simulation) et $u \sim$ и $_{[0, 1]}$;

- Evaluer A $_{split}$, et échantillonner $u \sim$ и $_{[0, 1]}$;
- Si u ≤ A s_{plit} alors l'état de la chaîne de Markov devient $(k+1, \mu_{1:k+1})$ sinon elle reste $(k, \mu_{1:k})$.

$$r_{split} = \prod_{i=1}^{c}\left[\left(\frac{y_{1:N,i}^{T}\, p_{i,k}^{*}\, y_{1:N,i}}{y_{1:N,i}^{T}\, p_{i,k+1}^{*}\, y_{1:N,i}}\right)^{N/2}\right] \frac{k\ \zeta\ \exp(-C)}{(k+1)}$$

$$A_{split} = \min(1, r_{split}).$$

5.1.4. Saut de fusion « merge jump »

- Choisir aléatoirement un des k terme RBF μ_1 ;
- Trouver son voisin proche μ_2 ; (en utilisant la distance euclidienne) ;
- Si $\| \mu 1 - \mu 2 \| < 2\zeta$, alors remplacer les deux fonctions RBF par une seul RBF dont la location est $\mu = (\mu 1 - \mu 2)/2$;
- Evaluer A $_{merge}$, et échantillonner $u \sim$ и $_{[0, 1]}$;
- Si $u \le$ A $_{merge}$ alors l'état de la chaîne de Markov devient $(k-1, \mu_{1:k-1})$ sinon elle reste $(k, \mu_{1:k})$.

$$r_{merge} = \prod_{i=1}^{c}\left[\left(\frac{y_{1:N,i}^{T}\, p_{i,k}^{*}\, y_{1:N,i}}{y_{1:N,i}^{T}\, p_{i,k-1}^{*}\, y_{1:N,i}}\right)^{N/2}\right] \frac{k\ \exp(C)}{\zeta(k-1)}$$

$$A_{merge} = \min(1, r_{merge}).$$

5.1.5. Mise à jour « update move »

L'échantillonnage des centres RBF est difficile, puisque leur distribution est non linéaire. Là on échantillonne un à la fois en utilisant un ensembles d'étapes MH.

Pour $j = 1, 2, \ldots, k$

- Tirer un échantillon $u \sim$ и $_{[0, 1]}$;
- Si $u < 0.5$

❖ Echantillonner aléatoirement un centre RBF à partir de l'intervalle initialement fixé dans (**Equ.8**).

❖ Calculer $\mu_j^{\bullet}$

$$r_{update} = \prod_{i=1}^{c}\left(\frac{y_{1:N,i}^{T}\, p_{i,k}^{*}\, y_{1:N,i}}{y_{1:N,i}^{T}\, p_{i,k}^{*}\, y_{1:N,i}}\right)^{N/2}$$

D'où $p_{i,k}^{\bullet}$ est le même que $p_{i,k}^{*}$ avec $\mu_{1:k,1:d}$, remplacé par : $\{\mu_{1,1:d}, \mu_{2,1:d}, \ldots, \mu_{j-1}, \mu_{j,1:d}^{\bullet}, \mu_{j+1,1:d}, \ldots, \mu_{j,1:d}\}$.

❖ Si $v \sim u_{[0,1]} \le \min\{1, r_{update}\}$ alors l'état devient $(k, \mu_1, \mu_2, \ldots, \mu_{j-1}, \mu_{j+1}, \ldots, \mu_k)$ sinon il reste inchangé.

- Si $u \ge 0.5$

❖ Echantillonner aléatoirement un centre RBF $\mu_j^{\bullet}$ à partir de la distribution : $\mu_{j,1:d}^{*} \setminus \mu_{j,1:d} \sim N(\mu_{j,1:d}, \sigma_{RW}^{2} \boldsymbol{I}_d)$.

❖ $r_{update} = \prod_{i=1}^{c}\left(\frac{y_{1:N,i}^{T}\, p_{i,k}^{*}\, y_{1:N,i}}{y_{1:N,i}^{T}\, p_{i,k}^{\bullet}\, y_{1:N,i}}\right)^{N/2}$

❖ Si $v \sim u_{[0,1]} \le \min\{1, r_{update}\}$ alors l'état devient $(k, \mu_{1,1:d}, \mu_{2,1:d}, \ldots, \mu_{j-1,1:d}, \mu_{j,1:d}^{\bullet}, \mu_{j+1,1:d}, \ldots, \mu_{k,1:d})$, sinon il reste inchangé.

5.2. L'optimisation

D'une perspective MCMC, on peut résoudre un problème d'optimisation, comme celui posé dans notre cas, en adoptant la stratégie du récuit simulé. Cette dernière implique la simulation de chaîne de Markov non homogène dont la distribution à l'itération i, n'est pas $\pi(z)$ mais :

$\pi_i(z) \propto \pi^{1/T_i}(z)$ avec T : la température

Lorsque $i \to \infty$ $T = 0$, la densité $\pi^{\infty}(z)$ se concentre sur l'ensemble des maximaux globaux $\pi(z)$.

Si l'on considère le noyau de transition de l'algorithme (R-J-MCMC) $T(z, z^*)$ comme loi de proposition, le rapport d'acceptation recuit est donné par :

$$A_{RJSA} = \min\left\{1, \frac{\pi^{(1/T_i - 1)}(z^*)}{\pi^{(1/T_i - 1)}(z)}\right\}$$

6 Explication des étapes de l'algorithme

1- Initialisation :

Les valeurs initiales de $\mu_1, \ldots, \mu_k$ sont aléatoirement choisies selon (**Equ.10**).

2- La boucle :

♦ Les sauts « ***birth*** » et « ***death*** » permettent au réseau respectivement de s'augmenter de k *à* $k+1$, et de se baisser de k à $k-1$.

♦ Les sauts « ***split*** » et « ***merge*** » permettent aussi de changer la dimension de k à $k+1$ et de k à $k-1$.

♦ Le saut « ***merge*** » sert à éviter de placer plusieurs fonctions RBF dans le même voisinage. D'autre part le saut « ***split*** » est utilisé dans les régions de données où il y a des composantes étroites.

Remarques :

♦ Le noyau résultant de la simulation de la chaîne de Markov est donc un mélange de plusieurs noyaux de transition liés aux mouvements décrits ci-dessus.

♦ A chaque itération, un des sauts (b, d, m, s, u) est choisis aléatoirement. Les probabilités pour choisir ces sauts sont respectivement b_k, d_k, m_k, s_k et u_k, tel que $b_k + d_k + m_k + s_k + u_k = 1$ avec $(0 \le k \le k_{max})$. Le saut n'est effectué que si l'algorithme l'accepte.

✓ Pour $k = 0$, les sauts mort, scission et fusion sont impossibles, donc :
$d_0 = 0$; $m_0 = 0$; $s_0 = 0$.

✓ Pour $k = 1$, le saut fusion est interdit. Donc $m1 = 0$.

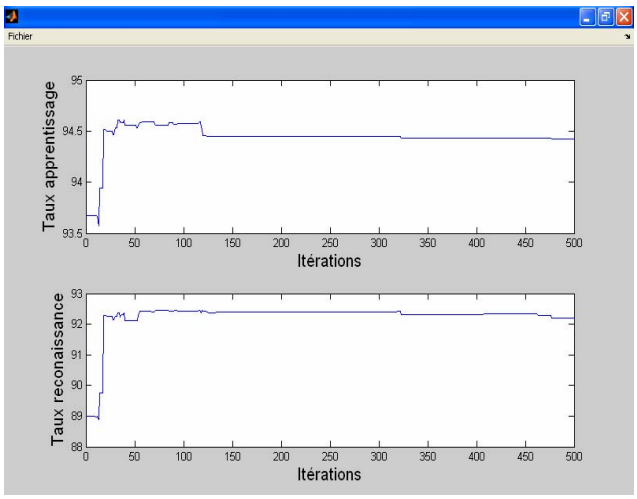
✓ Pour $k = k_{max}$, les saut naissance et scission, ne sont pas autorisés et pour cela ; $b_{kmax} = 0$; $s_{kmax} = 0$.

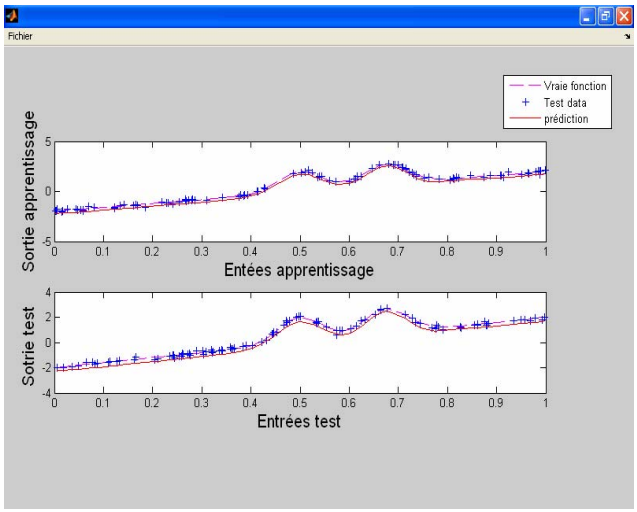
♦Notre algorithme, donne la MAP jointe estimation de $\mu_{1:k}$, et k avec $b_k = d_k = m_k = s_k = u_k = 0.2$.

7 Application : Approximation des fonctions

Les fonctions a simuler sont des fonctions non linéaire à deux variables, où les entrées (x,y) sont tirées d'une distribution gaussienne avec une moyenne nulle, une variance =1 et un bruit v.

Un aperçue de la vraie fonction, et celle réalisée par l'algorithme





8 Conclusion

Dans cet article, on a présenté une technique pour l'apprentissage des réseaux de neurones, s'appuyant sur le principe des méthodes Monte Carlo par chaînes de Markov .L'intégration de ces méthodes à l'apprentissage des réseaux de neurones, nous a mené à des résultats satisfaisants par rapport à d'autres algorithmes classiques d'apprentissage.
L'application de ces méthodes à l'approximation des fonctions nous a indiqué clairement que cette technique d'apprentissage représente une alternative intéressante et prometteuse dans les méthodes existantes.
Dans cet article, nous avons donné une estimation du bruit, et du nombre de paramètres pour un modèle de réseau RBF d'une façon générale.

Nous avons adopté une inférence bayesienne, avec l'algorithme MCMC à sauts réversible pour effectuer les intégrales nécessaires, par conséquence une minimisation du temps d'apprentissage et de l'erreur pour le réseau de neurones.

9 Références

[1] D. J. C. MacKay. "*Bayesian interpolation". Neural Computation*, 4(3), 415-447, 1992.
[2] D. J. C. MacKay. "*A Practical Bayesian Framework for Backpropagation Networks*". Neural Computation, 4(3), 448-472, 1992.
[3] R. M. Neal. "***Bayesian Training of Backpropagation Networks by the Hybrid Monte Carlo Method"***. Technical Report CRG-TR-92-1, Department of Computer Science, University of Toronto, 1992.
[4] R. M. Neal. "***Bayesian Learning for Neural Network"***. Ph.D. thesis, University of Toronto, 1994.
[5] R. M. Neal. "***Bayesian methods for Neural Networks"***. New York : Springer- Verlag, 1996.
[6] W. Buntine, A.S "Weigend.***Bayesian back-propagation".*** *Complex Systems,* 5, 603-643, 1991.
[7] C. M. Bishop. "***Neural Networks for Pattern Recognition"***. Clarendon Press, Oxford, 1995.
[8] JFG de Freitas, M Niranjan, A H Gee, and A Doucet. "***Sequential Monte Carlo methods for optimisation of neural network models"***. Technical Report CUED/F-INFENG/TR 328, Cambridge University Engineering Department, July 1998.
[9] Aki Vehtari et Jouko Lampinen. « ***Bayesian neural networks with correlating residuals"***. In *Proc. IJCNN'99*, Washington, DC, USA, July 1999.
[10] JFG de Freitas. "***Robust full bayesian methods for neural network"***, Cambridge university , 2000.
[11] D. J. C. MacKay. "***The Evidence Framework Applied to Classification Networks"***. *Neural Computation*, 4(5), 698-714, 1992.
[12] Philippe Leray et Olivier François, « ***Etude comparative d'algorithme d'apprentissage et de structure dans les réseaux bayésiens »***, Laboratoire Perception, Systèmes, Information - FRE CNRS 2645.
[13] N. J. Gordon, D. J. Salmond, and A. F. M. Smith. "***Novel approach to nonlinear/non-gaussian bayesian state estimation"***. *IEEE Proceedings-F*, 140(2):107–113, April 1993.
[14] S. F. Gull. "***Bayesian inductive inference and maximum entrop***". G. J. Erickson,C.R. Smith eds. *Maximum-Entropy and Bayesian Methods in Science and Engineering, Vol.1: Foundations*, 53-74, Dordrecht: Kluwer, 1988.
[15]. J S Liu and R Chen. "***Sequential Monte Carlo methods for dynamic systems"***. *Journal of the American Statisti- cal Association*, 93:1032–1044, 1998.
[16] P. Muller. (1991). "***Monte Carlo integration in general dynamic models"***. *Contemporary Mathematics, 115*, 145–163.