Une Modélisation à Base De Règles Floues*

Manfred MÄNNLE
Centre de Recherche en Automatique de Nancy (CRAN)
Faculté de Sciences
BP 239
F-54506 Vandœuvre Cedex
s_maennl@ira.uka.de, maennle@cran.u-nancy.fr

11 mai 1995

Mots clefs : Modèle à base de règles floues, sigmoïdes, algorithme de Sugeno, algorithme gradient, RPROP

Résumé

Le modèle flou proposé dans cet article est consideré pour trouver un modèle MISO et statique. Il s'agit donc d'une approximation d'une fonction multivariable. L'algorithme de modélisation est une modification de l'algorithme «TSK» de Takagi, Sugeno et Kang [12, 10]. Les fonctions d'appartenance utilisées sont des sigmoïds et les paramètres sont calculés par RPROP (resilient propagation), une amélioration simple de l'algorithme gradient.

Cet article provisoire est considéré pour l'utilisation interne et n'est pas encore finalisé.

1 Introduction

Une bonne motivation pour la modélisation floue se trouve dans l'article [16] de L. Zadeh. L'idée c'est d'introduire le flou pour mieux traiter l'incertitude et l'imprécision. On espère trouver quelques informations sur des systèmes complexes, où les approches classiques ne donnent pas de bons résultats. La différence par rapport à l'approche neuronale, c'est qu'on peut (plus ou moins) interpréter les règles floues trouvées et gagner de l'information qualitative sur le système regardé. Cependant, lorsqu'on introduit le flou, le résultat aussi sera plus flou et donc mois précis. Mais, en augmentant le nombre de règles ou des ensembles flous pour partitionner l'espace d'entrée, on peut aussi augmenter la précision du résultat.

*Travail dans le cadre du program d'échange de l'Institut Franco-Allemand IAR.

Contrairement aux modèles flous à base des relations, dont on trouve des vues d'ensemble et des comparaisons par exemple dans [2, 9], les modèles à base des règles ont un mécanisme d'inférence fixe mais des ensembles flous variables. Parfois, comme pour le TSK, le nombre de règle est variable aussi. Ici, l'inférence floue est définie par le produit (Larsen). Pour $\tilde{A}$ avec $\mu_A : \mathcal{U} \to [0,1]$, $\tilde{B}$ avec $\mu_B : \mathcal{V} \to [0,1]$ on a donc

$$\mu_{[A \to B]}(u,v) := \mu_A(u) \cdot \mu_B(v) \tag{1}$$

Les ensemble flous des prémisses de règles sont donné par des sigmoïds. Au contraire aux «systèmes de type Mamdani», où les conséquences sont des ensembles flous aussi, ici, dans le «système de type Sugeno», elles sont données par une constante ou une surface et de ce fait elles ne sont pas floues. L'intersection des ensembles floues est définie par le produit comme t-norm. On a ainsi :

$$\mu_{[A \cap B]}(u,v) := \mu_A(u) \cdot \mu_B(v) \tag{2}$$

La défuzzification est accomplie par le centre de gravité.

Si on prend des gaussiennes comme fonctions d'appartenance, c'est pas très difficile à montrer, qu'un système de type Sugeno est (en théorie) capable d'approximer chaque fonction continue avec une exactitude arbitraire [14], mais le nombre de règles peut devenir infinitivement grand. Lorsque le produit de deux fonctions sigmoïdals donne approximativement une gaussienne, on peut attendre le même résultat pour le système proposé ici.

Le modèle flou examiné dans cet article sera concrétisé dans le chapitre 2. Chapitre 3 et 4 décrivent l'identification des paramètres et l'algorithme heuristique de Sugeno. Dans le chapitre 5, deux exemples sont présentés, enfin chapitre 6 discute quelques questions encore ouvertes.

2 Le modèle flou

Soi N la dimension des vecteurs d'entrée $\vec{u}$, M le nombre des exemples mesurés $(\vec{u}, y)$ et R le nombre actuel des règles floues. Les règles floues sont ainsi de la forme :

$$\begin{aligned} &\text{if } u_1 \text{ is } F_{1k} \text{ and } \ldots \text{and } u_N \text{ is } F_{Nk} \text{ then} \\ &\quad \hat{y}_k = p_{0k} + p_{1k} \cdot u_1 + \ldots + p_{Nk} \cdot u_N \end{aligned} \quad (3)$$

À ce sujet les fonctions d'appartenance sont définies par des sigmoïds et pour $i = 1, \ldots, M$; $k = 1, \ldots, R$; $j = 1, \ldots, N$ on a alors

$$F_{jk}(u_{ji}) = \frac{1}{1 + e^{\sigma_{jk} \cdot (u_{ji} - \mu_{jk})}} \quad (4)$$

Soi pour $k = 1, \ldots, R$; $\vec{u} \in U_1 \times \ldots \times U_N$; $U_j \subset \mathbb{R}$

$$\begin{aligned} w_k(\vec{u}) &= \bigwedge_{j=1}^{N} F_{jk}(\vec{u}) \\ &= \prod_{j=1}^{N} F_{jk}(\vec{u}) \quad \text{(t-norm : produit)} \end{aligned} \quad (5)$$

La conséquence de la règle k; $k = 1, \ldots, R$ est calculée par

$$\begin{aligned} f_k(\vec{u}) &= \vec{p_k}\vec{u} \\ &= p_{0k} + p_{1k} \cdot u_1 + \ldots + p_{Nk} \cdot u_N \end{aligned} \quad (6)$$

Avec l'expression (1) et le centre de gravité, on obtient la sortie précise

$$\hat{y}(\vec{u}) = \frac{\sum_{r=1}^{R} w_r(\vec{u}) \cdot f_r(\vec{u})}{\sum_{r=1}^{R} w_r(\vec{u})} \quad (7)$$

En regardant

$$v_k(\vec{u}) = \frac{w_k(\vec{u})}{\sum_{r=1}^{R} w_r(\vec{u})} \qquad k = 1, \ldots, R \quad (8)$$

on a $\forall \vec{u} \in U_1 \times \ldots \times U_N$

$$\sum_{r=1}^{R} v_r(\vec{u}) = 1 \quad (9)$$

et v_r peut être considéré comme une distribution de possibilité.

L'optimisation des paramètres de conséquences est une optimisation linéaire est peut être poursuivie par la méthode des moindres carrés [7] ou, comme ici, par un algorithme itératif, par exemple l'algorithme RPROP [3, 4], qui est expliqué en détail dans le chapitre 3.1. L'optimisation nonlinéaire des paramètres de prémisses est le sujet du chapitre 3.2. Pour économiser le temps de calcul, uniquement l'algorithme RPROP est appliqué, car il faut optimiser les paramètres des conséquences et des prémisses de maniére alternatif.

Ici on utilise donc le modèle flou TSK de Sugeno modifié : des sigmoïds pour les ensembles flous et RPROP pour optimiser les paramètres.

Des autres approches qui ressemblent à celui de Sugeno sont par exemple

- Yager et Filev [15, 5]. Ils utilisent aussi une architecture TSK, mais avec des fonctions de cloche et un algorithme gradient pour calculer les paramètres.
- Tanaka et Tanino [13] utilisent un algorithme génétique pour trouver les paramètres.
- Sugeno et Yasukawa [11] proposent une approche «inverse». Ici l'espace de *sortie* est partitionné ensuite on cherche les entrées appartenant aux partitions trouvées.
- Nakamori et Ryoke [8] font aussi une approche «inverse» comme Sugeno et Yasukawa, mais avec des partitions «hyperéllipsoidales» dans l'espace de sortie.
- Abe et Lan [1] calculent une partition de l'espace d'entrée selon la valeur de la sortie et extraient directement des règles floues.

3 Identification des parametres

L'identification des paramètres se passe par minimisation de l'écart $\|\vec{\varepsilon}\|^2 = \|\vec{y} - \vec{\hat{y}}\|^2$. Cet écart depend seulement des paramètres et on a au minimum

$$\begin{aligned} \frac{\partial \|\vec{\varepsilon}\|^2}{\partial p_{jr}} = 0, \qquad \frac{\partial \|\vec{\varepsilon}\|^2}{\partial \mu_{jr}} = 0 \\ \text{et} \qquad \frac{\partial \|\vec{\varepsilon}\|^2}{\partial \sigma_{jr}} = 0 \end{aligned} \quad (10)$$

L'algorithme RPROP [3, 4] additionne dans chaque itération t une certaine valeur Δ_i^t au parametre p_i. Si le signe du gradient partiel ne change pas, le dernier pas est bien. Donc on continue de progresser dans la même direction

$$p_i^t = \begin{cases} p_i^{t-1} - \Delta_i^t & \text{si } \frac{\partial \|\vec{\varepsilon}^t\|^2}{\partial p_i} > 0 \\ p_i^{t-1} + \Delta_i^t & \text{si } \frac{\partial \|\vec{\varepsilon}^t\|^2}{\partial p_i} < 0 \\ p_i^{t-1} & \text{si } \frac{\partial \|\vec{\varepsilon}^t\|^2}{\partial p_i} = 0 \end{cases} \quad (11)$$

Si le signe change, le dernier pas est trop grand, on a passé le minimum. Pour réparer cela, on retire le dernier pas

$$p_i^t = p_i^{t-2} \text{ si } \frac{\partial \|\vec{\varepsilon}^{t-1}\|^2}{\partial p_i} \cdot \frac{\partial \|\vec{\varepsilon}^{t}\|^2}{\partial p_i} < 0 \quad (12)$$

Aussi selon le changement de signe le pas est augmenté, si on allait dans la bonne direction. Si on passe le minimum, le pas diminue. L'adaption itérative du *pas* est alors

$$\Delta_i^t = \begin{cases} \eta^+ \cdot \Delta_i^{t-1} & \text{si } \frac{\partial \|\vec{\varepsilon}^{t-1}\|^2}{\partial p_i} \cdot \frac{\partial \|\vec{\varepsilon}^{t}\|^2}{\partial p_i} > 0 \\ \eta^- \cdot \Delta_i^{t-1} & \text{si } \frac{\partial \|\vec{\varepsilon}^{t-1}\|^2}{\partial p_i} \cdot \frac{\partial \|\vec{\varepsilon}^{t}\|^2}{\partial p_i} < 0 \\ \Delta_i^t & \text{autrement} \end{cases}$$

avec les constantes $\eta^+ = 1.2$ et $\eta^- = 0.5$. Il est donc nécéssaire de calculer la dérivée partielle $\frac{\partial \|\vec{\varepsilon}^{t}\|^2}{\partial p_i} > 0$ à chaque itération t pour tous les paramètres p_i.

3.1 Identification des paramètres de conséquence

En choisissant la norme euclidienne pour $\|\cdot\|$, on obtient

$$\begin{aligned} \|\vec{\varepsilon}\|^2 &= \sum_{i=1}^{M} \varepsilon_i^2 \\ &= \sum_{i=1}^{M} \left(y_i - \sum_{k=1}^{R} v_k(\vec{u}_i) \cdot f_k(\vec{u}_i) \right)^2 \quad (13) \end{aligned}$$

et avec cela pour $r = 1, \ldots, R;\ j = 0, \ldots, N$

$$\begin{aligned} \frac{\partial \|\vec{\varepsilon}^{t}\|^2}{\partial p_{jr}} &= (-2) \sum_{i=1}^{M} (y_i - \hat{y}_i) \sum_{k=1}^{R} v_k(\vec{u}_i) \frac{\partial f_k(\vec{u}_i)}{\partial p_{jr}} \\ &= 2 \sum_{i=1}^{M} (\hat{y}_i - y_i) v_r(\vec{u}_i) u_{ji} \quad (14) \end{aligned}$$

lorsque $\frac{\partial f_k(\vec{u}_i)}{\partial p_{jr}} = u_{ji}$ et avec $u_{0i} := 1$.

3.2 Identification des paramètres de prémisse

Les dérivées partielles sont pour $j = 1, \ldots, N$ und $r = 1, \ldots, R$

$$\begin{aligned} &\frac{\partial \|\vec{\varepsilon}\|^2}{\partial \mu_{jr}} \\ &= 2 \sum_{i=1}^{M} \left((y_i - \hat{y}_i) \cdot (-1) \cdot \sum_{k=1}^{R} f_k(\vec{u}_i) \cdot \frac{\partial v_k(\vec{u}_i)}{\partial \mu_{jr}} \right) \\ &= 2 \sum_{i=1}^{M} \left((\hat{y}_i - y_i) \cdot \sum_{k=1}^{R} f_k(\vec{u}_i) \cdot \frac{\partial v_k(\vec{u}_i)}{\partial \mu_{jr}} \right) \quad (15) \end{aligned}$$

et également

$$\begin{aligned} &\frac{\partial \|\vec{\varepsilon}\|^2}{\partial \sigma_{jr}} \\ &= 2 \sum_{i=1}^{M} \left((\hat{y}_i - y_i) \cdot \sum_{k=1}^{R} f_k(\vec{u}_i) \cdot \frac{\partial v_k(\vec{u}_i)}{\partial \sigma_{jr}} \right) \quad (16) \end{aligned}$$

La dérivée partielle de (8) est pour $k \neq r$

$$\frac{\partial v_k(\vec{u}_i)}{\partial \mu_{jr}} = \frac{-w_k(\vec{u}_i)}{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right)^2} \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \mu_{jr}} \quad (17)$$

et pour $k = r$

$$\begin{aligned} &\frac{\partial v_r(\vec{u}_i)}{\partial \mu_{jr}} \\ &= \frac{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right) \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \mu_{jr}} - w_r(\vec{u}_i) \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \mu_{jr}}}{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right)^2} \\ &= \frac{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right) - w_r(\vec{u}_i)}{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right)^2} \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \mu_{jr}} \quad (18) \end{aligned}$$

avec

$$\frac{\partial w_r(\vec{u}_i)}{\partial \mu_{jr}} = \prod_{\substack{q=1 \\ q \neq j}}^{N} F_{qr}(u_{qi}) \cdot \frac{\partial F_{jr}(u_{ji})}{\partial \mu_{jr}} \quad (19)$$

et

$$\begin{aligned} \frac{\partial F_{jr}(u_{ji})}{\partial \mu_{jr}} &= \frac{-e^{\sigma_{jr} \cdot (u_{ji} - \mu_{jr})} \cdot (-\sigma_{jr})}{\left(1 + e^{\sigma_{jr} \cdot (u_{ji} - \mu_{jr})}\right)^2} \\ &= \sigma_{jr} \cdot F_{jr}(u_{ji}) \cdot \frac{1 + e^{\sigma_{jr} \cdot (u_{ji} - \mu_{jr})} - 1}{1 + e^{\sigma_{jr} \cdot (u_{ji} - \mu_{jr})}} \\ &= \sigma_{jr} \cdot F_{jr}(u_{ji}) \cdot (1 - F_{jr}(u_{ji})) \quad (20) \end{aligned}$$

Conformément aux équations (17) et (18), on obtient pour $k \neq r$

$$\frac{\partial v_k(\vec{u}_i)}{\partial \sigma_{jr}} = \frac{-w_k(\vec{u}_i)}{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right)^2} \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \sigma_{jr}} \quad (21)$$

et pour $k = r$

$$\begin{aligned} &\frac{\partial v_r(\vec{u}_i)}{\partial \sigma_{jr}} \\ &= \frac{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right) \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \sigma_{jr}} - w_r(\vec{u}_i) \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \sigma_{jr}}}{\left(\sum_{s=1}^{R} w_s(\vec{u}_i) \right)^2} \end{aligned}$$

$$= \frac{\left(\sum_{s=1}^{R} w_s(\vec{u}_i)\right) - w_r(\vec{u}_i)}{\left(\sum_{s=1}^{R} w_s(\vec{u}_i)\right)^2} \cdot \frac{\partial w_r(\vec{u}_i)}{\partial \sigma_{jr}} \quad (22)$$

avec

$$\frac{\partial w_r(\vec{u}_i)}{\partial \sigma_{jr}} = \prod_{\substack{q=1 \\ q \neq j}}^{N} F_{qr}(u_{qi}) \cdot \frac{\partial F_{jr}(u_{ji})}{\partial \sigma_{jr}} \quad (23)$$

et

$$\begin{aligned} &\frac{\partial F_{jr}(u_{ji})}{\partial \sigma_{jr}} \\ &= \frac{-e^{\sigma_{jr}\cdot(u_{ji}-\mu_{jr})} \cdot (u_{ji}-\mu_{jr})}{\left(1+e^{\sigma_{jr}\cdot(u_{ji}-\mu_{jr})}\right)^2} \\ &= (\mu_{jr}-u_{ji}) \cdot F_{jr}(u_{ji}) \cdot \frac{1+e^{\sigma_{jr}\cdot(u_{ji}-\mu_{jr})}-1}{1+e^{\sigma_{jr}\cdot(u_{ji}-\mu_{jr})}} \\ &= (\mu_{jr}-u_{ji}) \cdot F_{jr}(u_{ji}) \cdot (1-F_{jr}(u_{ji})) \quad (24) \end{aligned}$$

4 L'algorithme heuristique de Sugeno

L'algorithme heuristique de Sugeno est expliqué en détail dans [12, 10, 6].

L'algorithme est composé de deux boucles imbriquées dont celle qui est à l'interieure optimise de mani'ere itératif les paramètres d'un modèle possedant une structure fixe jusquà ce que le critère d'arrêt augmente. Sugeno utilise dans [12, 10] le critère

$$UC = \sum_{i=1}^{M_A} (\hat{y}_i^A - \tilde{y}_i^A)^2 + \sum_{i=1}^{M_B} (\hat{y}_i^B - \tilde{y}_i^B)^2 \quad (25)$$

où ˆ signifie le modèle identifié avec A, ˜ le modèle identifié avec B et $\hat{y}^B$ est donc la sortie sur les exemples B du modèle identifié avec A. La boucle à l'extérieure cherche la bonne structure, c'est à dire le nombre de règles et les variables dans les prémisses. À chaque étape on prend la meilleure structure de l'étape précédente et on ajoute une règle en partitionnant l'espace de la variable d'entrée. On a donc à l'étape no. r : $N \cdot r$ nouvelles structures possibles à examiner (r règles qu'on peut diviser chaque fois en N variables). La boucle de l'extérieure aussi se termine selon le critère UC, qui influence ainsi beaucoup la qualité du résultat final.

Dans des premiers essais l'optimisation avec le critère UC ne marche pas, car ce dernier arrête trop tôt l'optimisation ; peut-être seulement parce qu'il y avait pas assez de différences entre les deux ensembles A et B. Le comportement de l'algorithme en présence du bruit est encore à examiner. Dans les exemples déjà regardés où il y a peu de bruit, le critère

$$\sum_{i=1}^{M_A} (y_i^A - \hat{y}_i^A)^2 + \sum_{i=1}^{M_B} (y_i^B - \hat{y}_i^B)^2 \quad (26)$$

fournit des résultats acceptables.

Le temps de calcul est de l'ordre $O(R^3N^2IM)$ si I est le nombre moyen d'itérations RPROP. Cela veut dire que pour un modèle avec deux fois plus de règles il faut environ huit fois plus de temps de calcul tandis qu'un redoublement du nombre d'exemples seulement redouble le temps de calcul.

5 Exemples

Deux exemples simples sont déjà examinés, mais plutôt pour tester le logiciel que pour examiner les propriétés de l'algorithme. Pourtant, on trouve déjà des premiers résultats.

Le logiciel attend des données normalisées, c'est-à-dire avec une moyenne égale à zéro et les valeurs x dans l'interval $[-0.5, 0.5]$. C'est pas très grave de ne pas respecter ce contraint, mais l'algorithme d'optimisation marche mieux car le programe choisit les paramètres initials selon ce contraint.

5.1 Une échelle en deux dimensions

Une échelle étant choisie pour tester l'algorithme RPROP, car ici il faut trouver des paramètres extrêmes et les algorithmes gradient ont souvent du mal à bien résoudre ce type de problème.

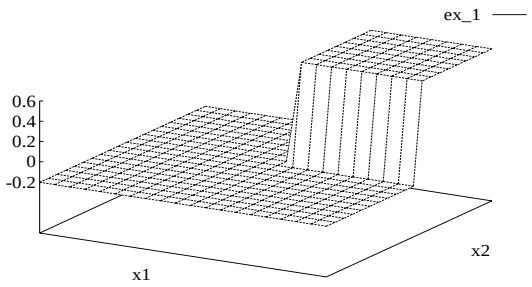

Figure 1: Une échelle

Il est interessant de suivre le dévéloppement des modèles obtenus. Chaque figure montre le meilleur modèle avec un certain nombre de règles. Figure 5.1 montre la sortie du modèle qui a une règle (c'est

donc un modèle linéaire), jusqu'à la figure 5.1 qui montre la sortie du modèle avec cinq règles. Ici on a quatre variables d'entrée dont deux contiennent que du bruit blanc et ne sont pas corrélées avec la sortie. C'est pourquoi les sorties des modèles, aussi celles du modèle linéaire, sont perturbées.

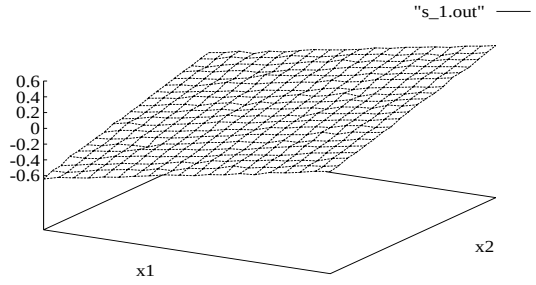

Figure 2: Une règle

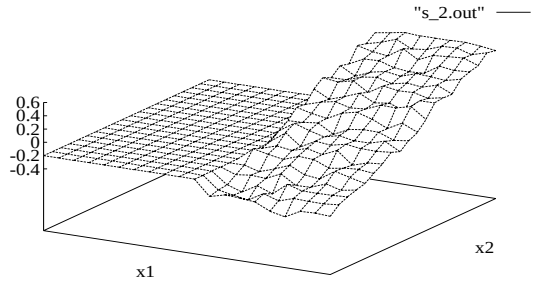

Figure 3: Deux règles

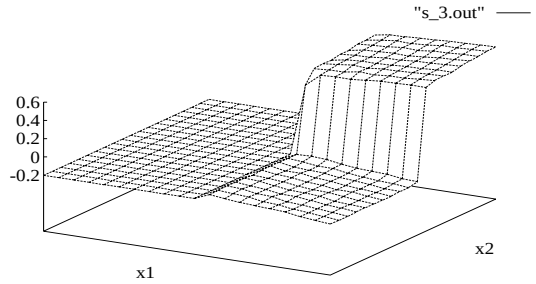

Figure 4: Trois règles

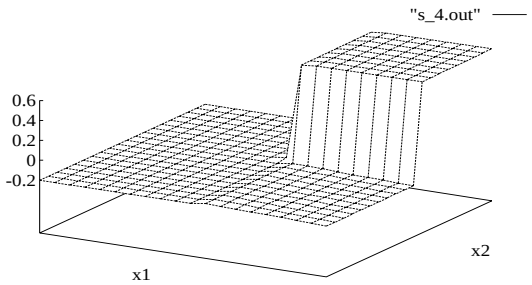

Figure 5: Quatre règles

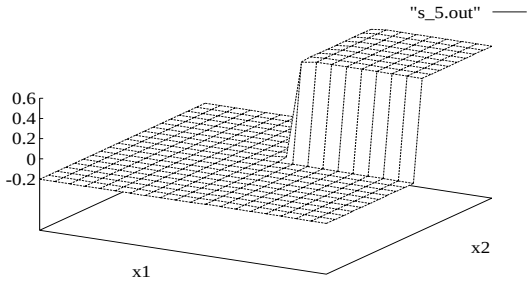

Figure 6: Cinq règles

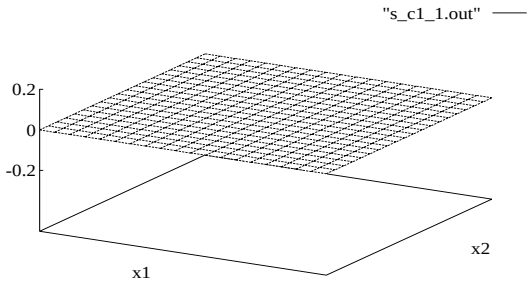

Figure 7: Une règle, conséquence constante

Déjà avec trois règles, on a un modèle qui approxime à peu près la vraie fonction, mais il reste une perturbation et un écart. L'algorithme a trouvé que seulement les variables $x1$ et $x2$ sont importantes, c'est-a-dire que les variables $x3$ et $x4$ n'apparaissent pas dans les prémisses. Cependant, elles sont toujours dans les conséquences et causent des perturbations.

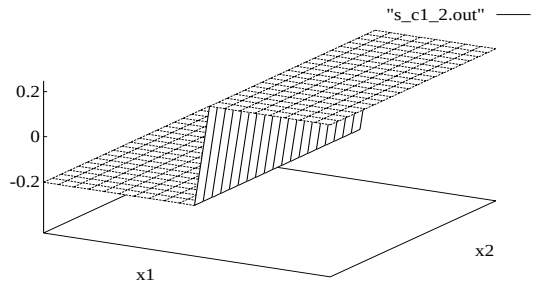

Figure 8: Deux règles, conséquence constante

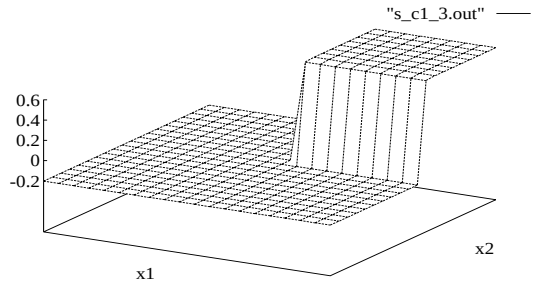

Figure 9: Trois règles, conséquence constante

Pour éviter cela et aussi pour accélérer l'algorithme, des modèles sont examinés, qui n'ont pas des surfaces comme conséquence mais des constantes*. Ces modèles sont plus robusts, le temps de calcul est reduit (car il y a moins de paramètres à identifier) et le résultat est beaucoup plus facile à interpréter. Cet exemple est aussi examiné avec seulement les deux variables $x1$ et $x2$ qui sont non bruitées. Dans ce cas les modèles avec des surfaces comme conséquences sont aussi bien que les autres.

Voici comme exemple les règles floues du modèle avec trois règles où les ensembles flous sont définis par leurs paramètres. L'ensemble $[0.09, -107.5]$ a un $\mu = 0.09$ et un $\sigma = -107.5$. Un grand σ indique une pente raide.

```
R1: if X1 is [ 0.09,-107.5]
   and X2 is [ 0.10, -90.0] then Y =  0.80
R2: if X1 is [ 0.01, 96.90] then Y = -0.20
R3: if X1 is [ 0.09,-107.5]
   and X2 is [-0.01,  76.6] then Y = -0.20
```

*Pour la preuve que ces modèls sont des approximateurs universels dans [14] aussi une consequénce constante suffit.

5.2 Deux gaussiennes

Comme exemple nonlinéaire une fonction en deux dimensions avec deux gausiennes est essayée. Aussi la première fois avec des surfaces et puis avec des constantes comme conséquences.

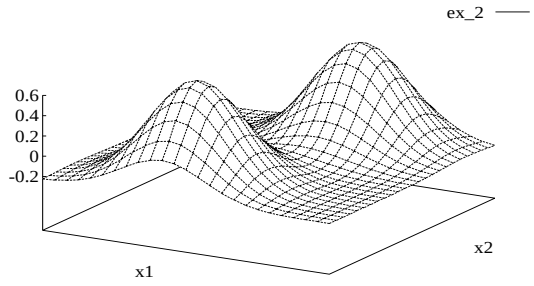

Figure 10: Deux gaussiennes

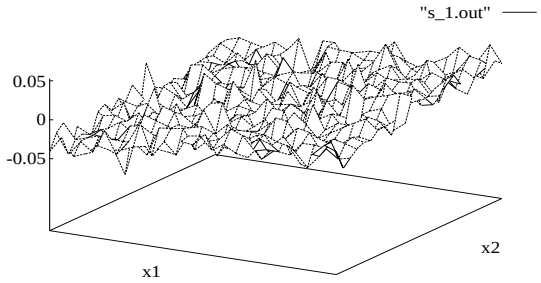

Figure 11: Une règle

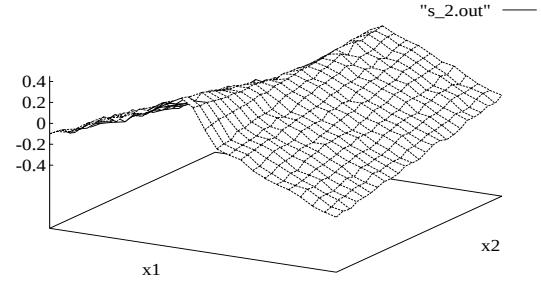

Figure 12: Deux règles

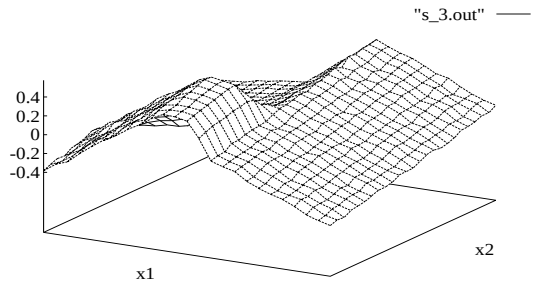

Figure 13: Trois règles

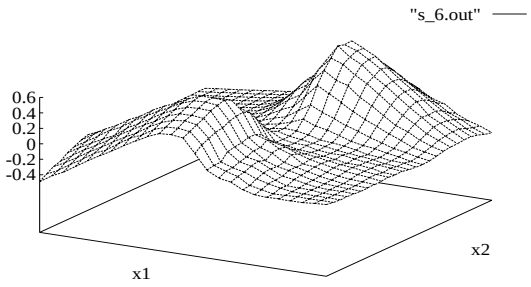

Figure 16: Six règles

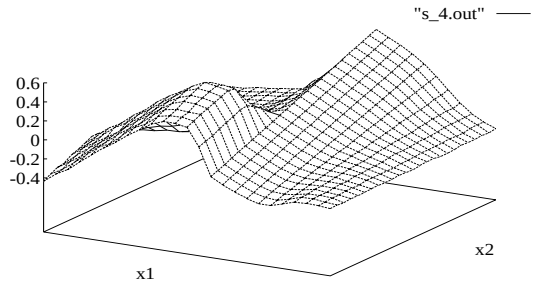

Figure 14: Quatre règles

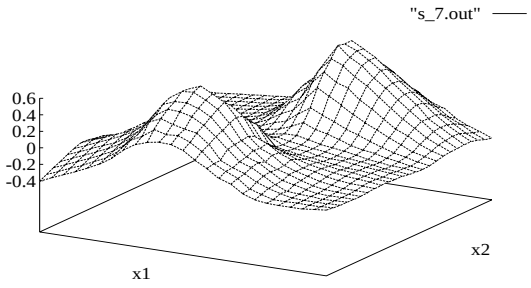

Figure 17: Sept règles

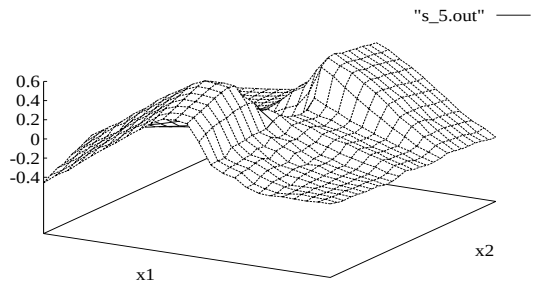

Figure 15: Cinq règles

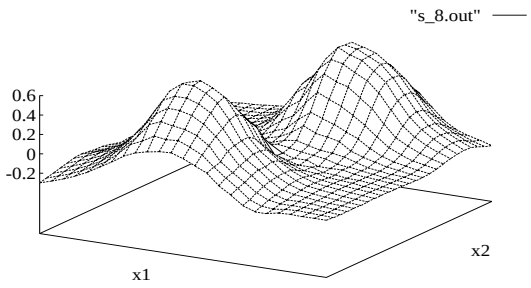

Figure 18: Huit règles

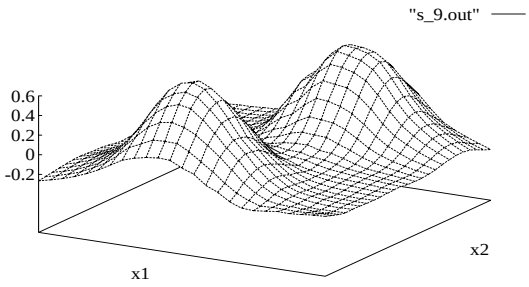

Figure 19: Neuf règles

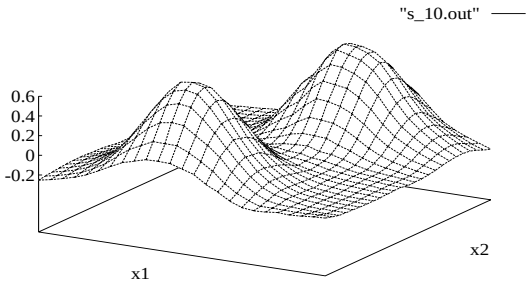

Figure 20: Dix règles

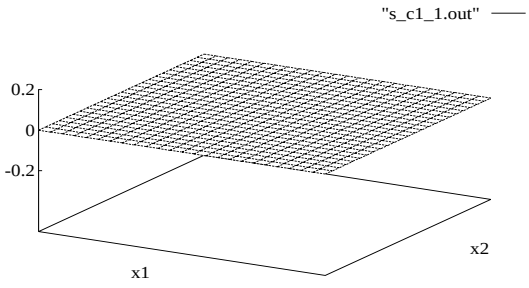

Figure 21: Une règle, conséquence constante

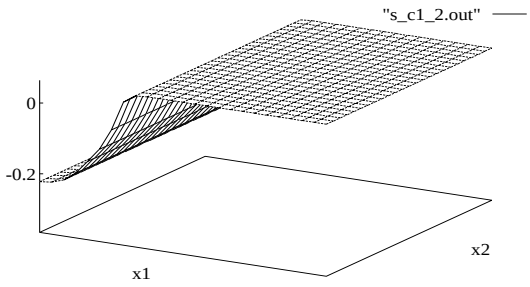

Figure 22: Deux règles, conséquence constante

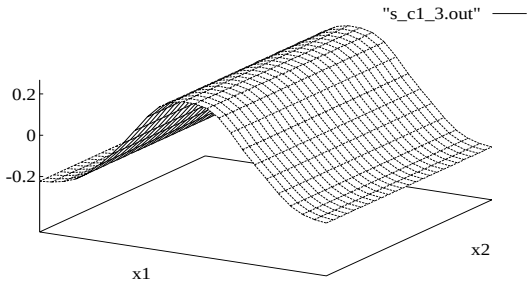

Figure 23: Trois règles, conséquence constante

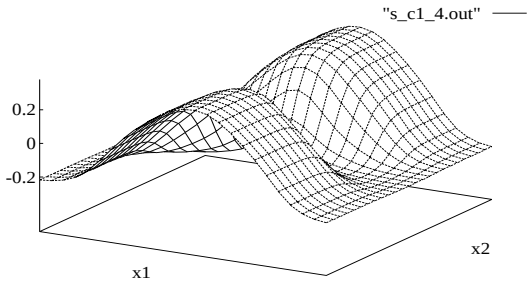

Figure 24: Quatre règles, conséquence constante

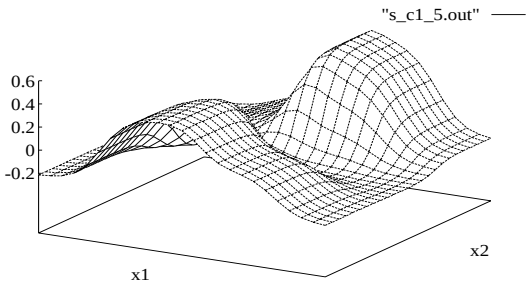

Figure 25: Cinq règles, conséquence constante

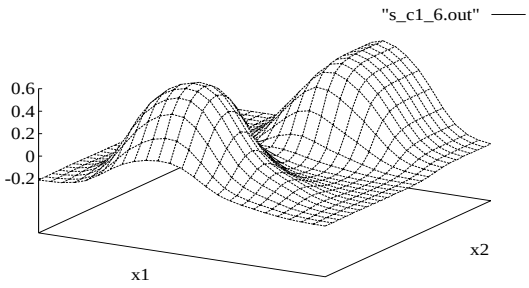

Figure 26: Six règles, conséquence constante

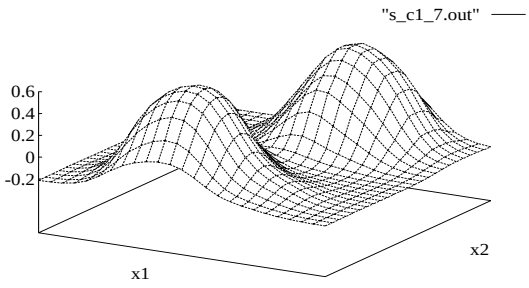

Figure 27: Sept règles, conséquence constante

À nouveau le modèle avec les conséquences constantes se montrent plus robust et fournit avec sept règles une approximation passable. Mais ici, dans les essais sans bruit, les modèles avec des surfaces sont meilleurs.

L'écart du dernier modèle avec sept règles et des conséquences constantes est montré dans la figure 5.2. Figure 5.2 donne le développement du critère R^2 pour les modèles avec conséquences constantes jusqu'à douze règles.

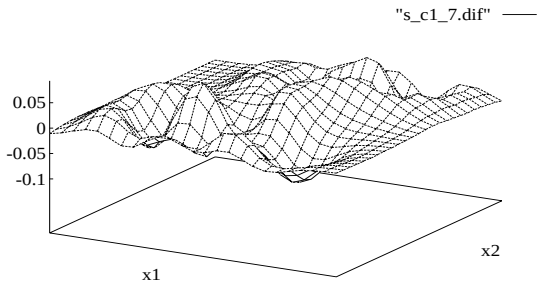

Figure 28: Écart du modèle c1_7

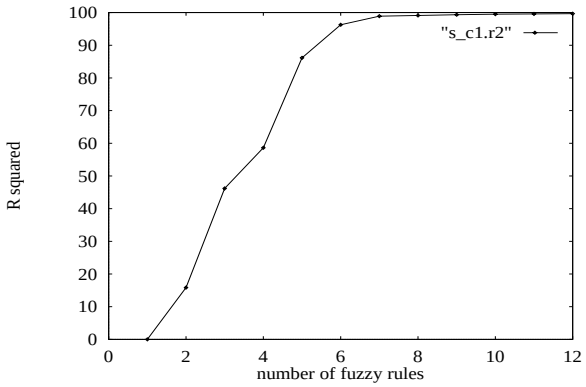

Figure 29: Dévéloppement du critére R^2

Voici les sept règles du dernier modèle avec des conséquences constantes. En examinant R5 on constate que l'y est très grand lorsque x_1 est compri entre 0.08 et 0.29 et x_2 entre 0.19 et 0.44. Dans cette partition d'espace d'entée se trouve une gaussienne. La deuxiéme est plus difficile à trouver, parce que pour cela il faut accumuler R4 et R7 qui donnent ensemble aussi une valeur d'y environ de 1.90 et leurs prémisses sont très proches.

```
R1: if X1 is [-0.25,-18.9]
   and X2 is [ 0.19,-25.8]
   and X1 is [ 0.04,-17.1] then Y = -0.19
R2: if X1 is [-0.60, 6.37] then Y = -0.21
R3: if X1 is [ 0.16,-19.4] then Y = -0.23
R4: if X1 is [-0.25,-18.9]
   and X1 is [ 0.29, 21.1]
   and X2 is [-0.31, 15.0]
   and X2 is [-0.28,-15.2] then Y = 0.90
```

```
R5: if X1 is [ 0.29, 21.1]
   and X2 is [ 0.19,-25.8]
   and X1 is [ 0.08,-23.6]
   and X2 is [ 0.44, 41.8] then Y = 1.85
R6: if X1 is [-0.25,-18.9]
   and X1 is [ 0.29, 21.1]
   and X2 is [-0.58, 25.1] then Y = 0.04
R7: if X1 is [ 0.29, 21.1]
   and X1 is [ 0.08,-23.6]
   and X2 is [ 0.26,-78.4] then Y = 1.05
```

Enfin en peut résumer que l'algorithme marche bien sur les exemples simples regardés ici. Le plus de règles utilisées, le meilleure est l'approximation, mais il est aussi plus difficile à interpréter les règles. Il faut donc, par exemple en regardant le développement de R^2, choisir un modèle assez exact avec peu de règles.

6 Questions ouvertes

Il restent des questions ouvertes qui seront encore examinées partiellement dans les semaines qui me restent à Nancy. Ce sont notamment

- La sensibilité de l'algorithme face aux bruits.
- La recherche des variables importantes. Est-ce que l'algorithme heuristique trouve les variables significatives ?
- L'influence du critère d'arrêt. Pourquoi le critère UC ne marchait-il pas dans les deux premiers exemples ? Marche-t-il mieux en cas du bruit élevé ?
- Une réduction du temps de calcul, par exemple en évitant d'examiner les variables d'entrée qui plusieures fois n'ont pas augmenté la qualité du modèle.
- L'amélioration de l'interprétabilité des règles par effacer les règles qui n'ont presque pas d'influence sur le résultat du modèle.
- L'applicabilité aux problèmes réels et durs (par exemple la machine à papier).

Références

[1] Abe S., Lan M., *Fuzzy Rules Extraction Directly from Numerical Data for Function Approximation*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 25, No. 1, pp. 119-129, 1995

[2] Böhler A., *Lösung von Fuzzy-Relationen-Gleichungen*, Diplomarbeit, Universität Karlsruhe, Fakultät für Informatik, Institut für Prozeßrechentechnik und Robotik, Oktober 1994

[3] Braun H., Riedmiller M., *Rprop: A fast adaptive learning algorithm*, Proceedings of the Int. Symposium on Computer and Information Science VII, 1992

[4] Braun H., Riedmiller M., *Rprop: A fast and robust backpropagation learning strategy*, Proceedings of the ACNN, 1993

[5] Filev D., *Fuzzy Modeling of Complex Systems*, Int. Journal of Approximate Reasoning, Vol. 5, pp. 281-290, 1991

[6] Hihi J., *Evaluation de méthodes d'identification de systèmes non linéaires en régime permanent : Méthode de Traitement des Données par Groupes ; Identification floue*, Thése de l'Université de Nancy 1, Mars 1993

[7] Klema V., Laub A., *The Singular Value Decomposition: Its Computation and Some Applications*, IEEE Transactions on Automatic Control, Vol. AC-25, No. 2, pp. 164-176, 1980

[8] Nakamori Y., Ryoke M., *Identification of Fuzzy Prediction Models Through Hyperellipsoidal Clustering*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 8, pp. 1153-1173, 1994

[9] Postlethwaite B., *Empirical comparison of methods of fuzzy relational identification*, IEE Proceedings-D, Vol. 138, No. 3, pp. 199-206, 1991

[10] Sugeno M., Kang G., *Structure Identification of Fuzzy Model*, Fuzzy Sets and Systems, Vol. 26, No. 1, pp. 15-33, 1988

[11] Sugeno M., Yasukawa T., *A Fuzzy-Logic-Based Approach to Qualitative Modeling*, IEEE Transactions on Fuzzy Systems, Vol. 1, No. 1, pp. 7-31, 1993

[12] Takagi T., Sugeno M., *Fuzzy Identification of Systems and Its Application to Modeling and Control*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 15, No. 1, pp. 116-132, 1985

[13] Tanaka M., Tanino T., *Identification of Nonlinear Systems Using Fuzzy Logic an Genetic Algorithms*, SYSID, Vol. 1, pp. 301-306, 1994

[14] Wang L.-X., *Fuzzy Systems are Universal Approximators*, IEEE International Conference on Fuzzy Systems, San Diego, CA, pp. 1163-1170, 1992

[15] Yager R., Filev D., *Unified Structure and Parameter Identification of Fuzzy Models*, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No. 4, pp. 1198-1205, 1993

[16] Zadeh L., *The role of fuzzy logic in modeling, identification and control*, Modeling, Identification and Control, Vol. 15, No. 3, pp. 191-203, 1994