Recherche Operationnelle Modélisation Paramétrique, Filtrage Optimal et Adaptatif Introduction au Data Mining Réseaux de Neurones

Mauricio Caceres, *Master, SISEA, ENSSAT, Lannion*

***Abstract*—Ce projet nous amenne à découvrir et faire une première application des concepts de machine learning. Les concepts de machine learning, deep learning, et réseau de neurones sont très couramment nommes. Sont très utilises pour la reconnaisances d'"images et la ré"slution de problèmes difficiles à modéliser et aussi avec une mathematique difficile à résoudre. Les techniques de machine learning ont une nature très empirique mais aussi une partie teorique et des fondaments mathematiques. Dans ce rapport les résultats de l'implementation d'une réseau des neurones multi-couches seront détailles avec differentes notions qui permetent de fondamenter les prise de décision au niveuau de l'implementation.**

***Keywords*—Réseaux de Neurones, perceptron multi-couches, convergence, backpropagation, descente maximale**

I. Introduction

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

LATEX

Fig. 1. Simulation Results

M. Pascal Scalart, Pôle Electronique, Enssat, Lannion, France e-mail: (pascal.scalart@univ-rennes1.fr).

A. Objectifs

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

II. Modélisation d'un neurone

Le modèle utilisé dans ce cas, inspiré dans le fonctionnement de une neurone biologique est le perceptron. Les

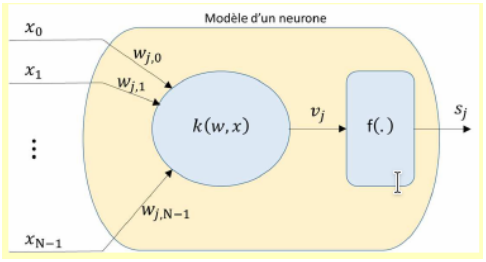

Fig. 2. Modèle de perceptron utilise dans le réseau de neurones

éléments suivant sont partie du modèle et necessaire pour comprendre le fonctionnement du perceptron.

- nombre de signaux d'entrée $x_0, ..., x_{N-1}$
- poids de connexions $w_{j,0}, ..., w_{j,L-1}$
- fonction d'activation $v = k(w, x)$
- fonction de transition f
- un état de sortie $s_j = f(v_j)$
- un entrée fixe: le bias

On a utilisé la fonction sigmoide comme fonction de transition. La fonction de activation est la fonction produit escalaire. Les connections entre neurones est faite par le bias du poids. Si la connections a un poid negative est une connection dite

exhibitrice si est positif est dite *excitatrice*. C'est une manière de modeliser la force de la connection synaptique entre ce perceptron là et une autre à lequel est connecté.

A. Le perceptron : fonctions de transition classiques

Ce sont les fonctions linaire, sigmoide, tangente hyperbolique ou autres.Cette fonction nous permets avoir la valeur de sortie de la neurone, si elle est positive est dite une neurone actif. Dans le cas contraire es dit inactif. Pour notre cas la function sigmoide de la Fig, 3 c'est la que nous permet avoir des valeur entre 0 et 1. Cela permettra de fixer le seuil d'activité de la neurone j.

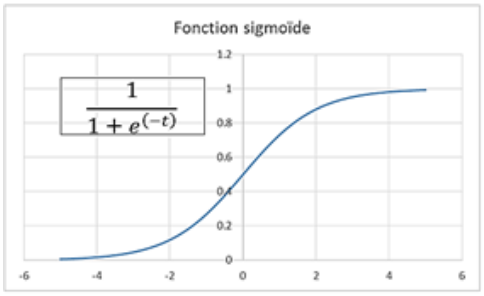

Fig. 3. Fonction sigmoide et sa derivé

B. Le perceptron : rôle de l'unité de biais

Dans la figure 2 on peut voir que la premier entrée $w_{j,0}$ es toujours à -1. La fonction de cette poids qu'on appelle bias est de fixer le seuil d'activation de la neurone. Permettant de déplacer l'hyperplan de la fonction d'activation qui peut-être vu comme une modification de la façon dont la neurone va se comporté face aux données, elle va classifier de manière différente

III. Le perceptron multicouche

À cause de l'impossibilité du perceptron pour classifier dans certain cas la connection de une couche de neurones a une autre supplémentaires nous donne une

A. Architecture utilise

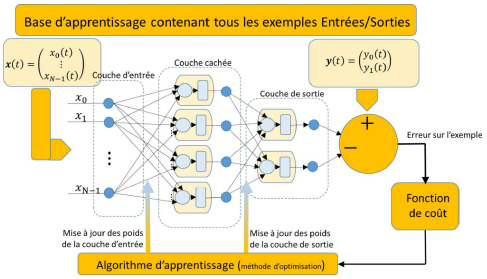

Fig. 4. Architecture de la réseaux utilisé. MLP conventional

B. L'erreur du perceptron : une fonction multidimensionelle

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

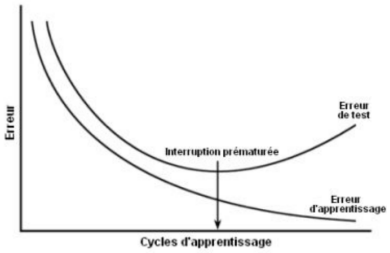

Fig. 5. Courbes indicant l'erreur d'apprentissage (risque empirique) et l'erreur de test(risque réel)

IV. L'apprentissage supervisé

Dans le programme du projet on a implimenté une processus d'apprentissage supervisé. Ce algorithme a comme finalité la modificatioon des poids de la couche d"entreé et de sortie permettant de l'adapter aux entrées d'accord à la fonction de coût. Il s'agit de presenter dans l'entrée des examples $x(n)$ correspondant à la base de données de entrenaiment et ensuite comparer la sortie $y(n)$ obtenue à la sortie theorique (classe correcte dans le cadre de la classification) ou correcte. Le fait de savoir à priori lequelle est la sortie correcte fait que l'algorithme soit supervisé.
La modification des poids initiaux va nous amener vers une réseaux entrainé correctement lorsque l'erreur diminue. Notre strategie a été de presenter les examples à l'entrée $x(n)$ selon le type de son, toujours en respectant l'ordre son 1, son 2, son 3. De cette façon on connais toujours quelle classe de son on a en entrée pour après calculer l'erreur.

V. Algorithme BackPropagation : calcul du vecteur gradient

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut

metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Avec ce type de réseau on peut traiter different types d'information comme signaux (images, sons). Sont capables de résoudre des problèmes de classification par le bias d'un processus d"apprentissage. L'algorithme d'apprentissage es s'agite de

VI. Cas d'un perceptron multicouche : implementations

Dans cette section on parlera sur les détails d'implementation de notre réseaux multi-couche. Les strategies suivis et les choix de parametres et de techniques de convergence. Il y a quatre codes avec des implementations et techniques differentes. On montrera leur performance et comportement. A la fin on aura des conclusions.

A. Base d'apprentissage et de test

L'apprentissage va se faire sur les caractéristiques plus fortes de notre données, soit des signaux ou des images ou n'importe quelle autre type de donne. Mais il faut trouver un moyen de représenter ou les extraire. On appelle attributs à les descriptors de la donnée

Aussi pour des limitations pratiques et d'implementation cette répresentation doit être échantillonné. Pour notre application on s'en sert de l'estimation de la DSP (densité spectrale de puissance) du signal à l'aide de la méthode du périodogramme fenêtré.

Listing 1. Code puor initialisation des variables

```
[dataTest,fs,Nbits] = wavread(name);
% calcul des data_test
L = size(dataTest,1);
X = fft(dataTest.*hamming(L),Nfft);
Sxx = 1/Nfft*abs(X).^2;
data_test(:,in) = Sxx(1:size(data_test,1));
```

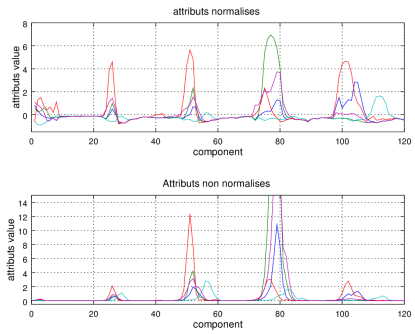

Fig. 6. Comparation entre les attributs normalisées,non normalisés

B. Parametrage du MLP

Notre MLP a 4098 entrées dans la couche d'entrée, connectes chaqu'une a 100 neurones dans la couche cachée. Dans la sortie intermediaire on a donc 100 sortie connectés à l'entrée du la couche de sortie qui a 3 neurones. La sortie de chacune des trois dernières neurones sont la sortie de notre réseaux.
Le taux d'apprentissage nous donne selon l'algorithme de backpropagation le rythme à lequel les poids sont modifies pour s'ajuster aux entrées (fit data).

```
L_in = 4097+1;
L_cachee = 100+1;
L_out = 3;
mu = 0.5 %taux d'apprentissage
a=-0.5;
b = 0.5;
C = [ a + (b-a).*rand(L_in,L_cachee)];
W = [a + (b-a).*rand(L_cachee,L_out)];
```

Le parametre μ est tres importante parce que modifie aussi la duree de la algortihme pour convergence. Si on converge tres rapidement avec un taux d'apprentisage grand on ne va pas bien apprendre la base de données donc on a le risque d'avoir une basse performance au niveau de la distinctions des examples de test parce que la classification est très generaliste. Si le taux d'apprentissage es plus petit l'entrainement prendera plus de temps et ça faire que le réseaux apprenne beacoup mieux les examples. Par contre si on entraine beacoup les neurones il y a le risque qu'elle aprenne la base de données tel comme elle est, en commetant ce que on appele *overfitting* ou *surapprentissage*. Dans ce cas il faut appliquer des techniques especifiques pour obtenir des résultats differentes.On parlera plus tard sur ce sujet là.

Les matrices C et W ont le poids du connections entre

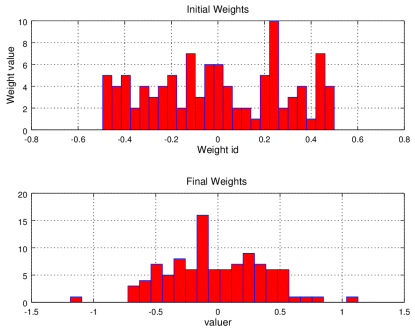

Fig. 7. Histogramme des poids de la couche de sortie

neurones et entrées et sorties. Dans la Fig.12 et la Fig.8 on voit les histogrammes des valeurs des poids pour les connections d'une neurone.

C. Test sur une version simple du MLP

Les courbes les plus importantes sont celles ci que nous montrent comment se comporte l'erreur. L'erreur commence

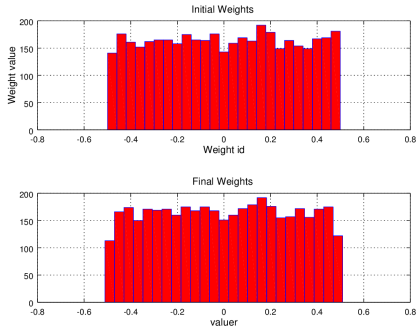

Fig. 8. Histogramme des poids de la couche de cachée

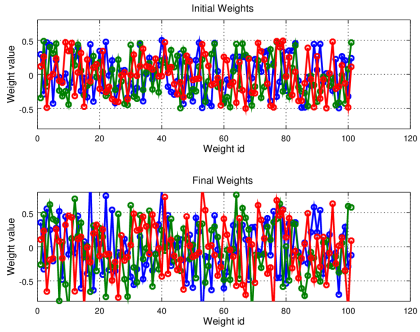

Fig. 9. Graphique pour comparation des poids de sortie initiaux et finales

au départ à diminuer et ensuite commence à ce stabiliser. Le critère d'arrêt de l'algorithme est arriver à un'erreur plus petit que 1e-4.

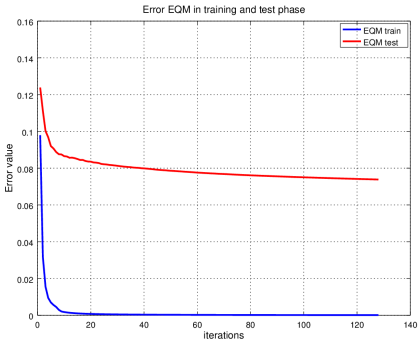

Fig. 10. Résultats de implementation simple MLP, $\mu = 0.5$,initialisation poids non normalement distribuée

VII. Amelioration de l'algorithme pour une meuilleur convergence

Entre plusiers techniques pour faire avoir une convergence dans une période des temps aceptable.

- permutation des examples de la base de données
- même frequence pour classe
- normalisation de l'entrée

Tous ces astuces sont déjà implementes dans la version base du code. Dans les programmes suivantes on a la mise en place de :

- rs2 : initialisation des poids avec distribution normal
- rs3 : On rajoute un taux d'apprentissage variable
- rs4 : On rajoute deux taux d'apprentissage variable

VIII. Programme rs2 : initialisation des poids avec distribution normal

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

La initialisation des poids de les matrices W et C sont fais à l'aide d'une function qui genere des numeros aleatoires avec une distribution normal centrée et paramètre $\sigma = \sqrt{m}$ en étant m les nombres de connection avec la couche précédente.

```
L_in = 4097+1;
L_cachee = 100+1;
L_out = 3;

%sigmas de distrib. de proba
sigma1=1/sqrt(L_in);
sigma2 = 1/sqrt(L_cachee);

%creation the matrices de poids
C = normrnd (0, sigma1, L_in, L_cachee);
W =   normrnd(0, sigma2, L_cachee, L_out);
```

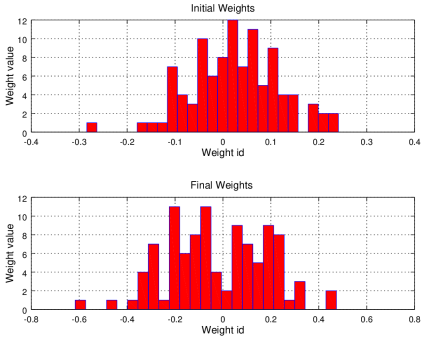

Fig. 11. Histogramme des poids de la couche de entrée d'une neurone.Distribution Normal

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

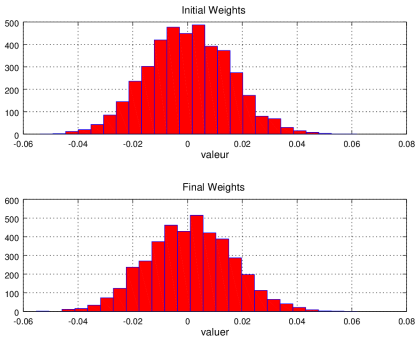

Fig. 12. Histogramme des poids de la couche de sortie d'une neurone.Distribution Normal

```
rs2 avec mu = 0.5

error =     9.9677e-05
error_test =  0.014817
elapsed_time =  37.162
----------------------
rs2 avec mu = 0.9

error =     9.9585e-05
error_test =  0.019871
elapsed_time =  26.649
----------------------
rs2 avec mu = 0.3

error =     9.9595e-05
error_test =  0.0095784
elapsed_time =  73.670
```

Voici la courbe d'erreur d'apprentisage et de test pour une valeur de $\mu = 0.5$

IX. Programme rs3 : On rajoute un taux d'apprentissage variable

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

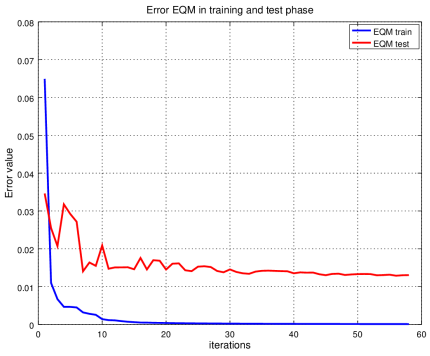

Fig. 13. Simulation Results

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

A cette étape là on fait varier le taux d'apprentissage

```
%Init taux d'apprentissage
alpha = 0.5;
muo = 0.5;
mu = muo/(1+alpha*1)
```

Et dans la boucle principal on fait la mise à jour

```
    for i = 1:60 %parcourir la base de donnees
            ...
            ...
            %calcul des sorties, erreur, update
                poids
            ...
            ...
                %update mu
                mu = muo/(1+alpha*i);
                storemu(iter)=mu; %store for graphs
        end
```

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum

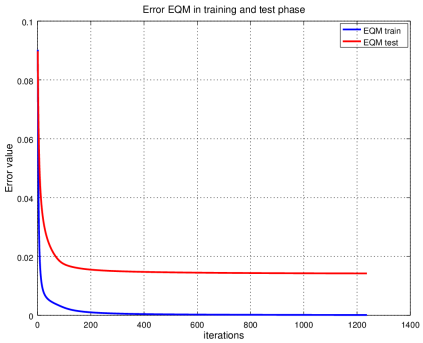

Fig. 14. Simulation Results

dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

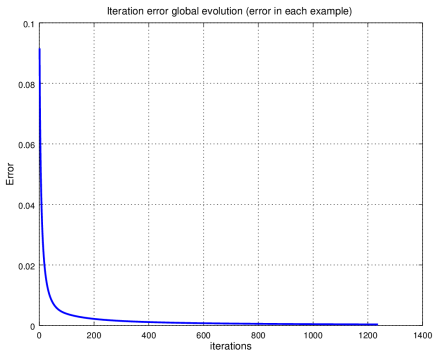

Fig. 15. Simulation Results

d

X. Programme Rs4 : On rajoute deux taux d'apprentissage variable

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper. Lorem ipsum

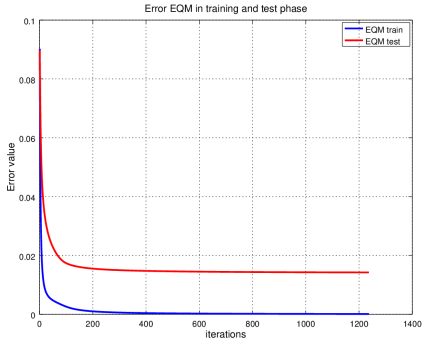

Fig. 16. Simulation Results

dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

XI. Conclusion

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Appendix A
Proof of the First Zonklar Equation

Some text for the appendix.

Acknowledgment

The authors would like to thank...

REFERENCES

[1] H. Kopka and P. W. Daly, *A Guide to LATEX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.