TP1 Apprentissage

Étudiant: Isaías Faria

Partie 2 - Le perceptron

1 -

Avec TangH:

Code:

```
net_output = tanh(dot(neuron.weights',patt_in));
```

Figure

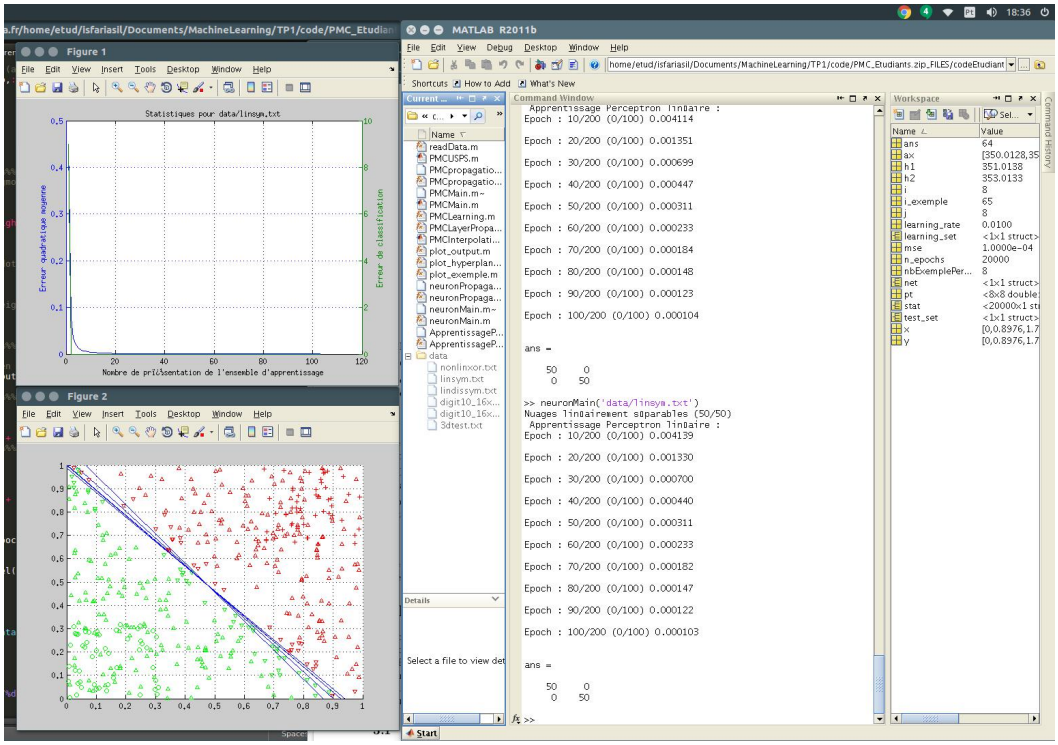

Figure 1: L'apprentissage avec tangH.

Avec Sigmoïde:

Code:

```
net_output =
2.0/(1.0+exp(-1*(dot(neuron.weights',patt_in)))) -1.0;
```

Figure

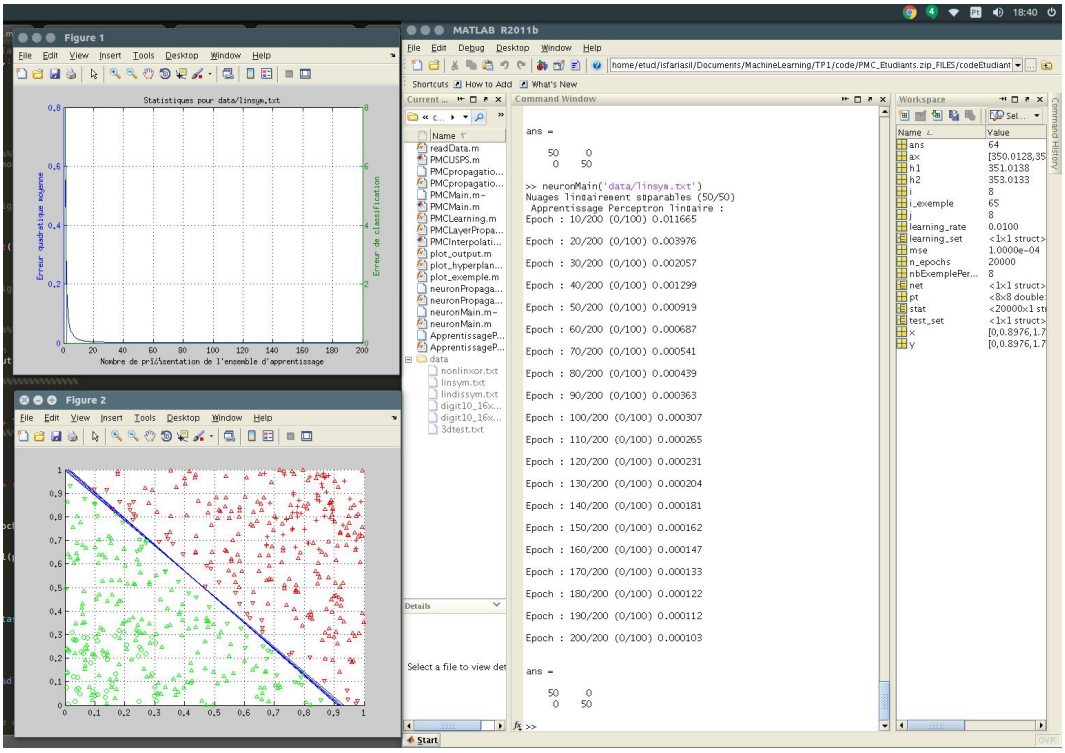

Figure 2: L'apprentissage avec Sigmoïde.

Avec seuil:

Code:

```
net_output = sign(dot(neuron.weights',patt_in));
```

Figure

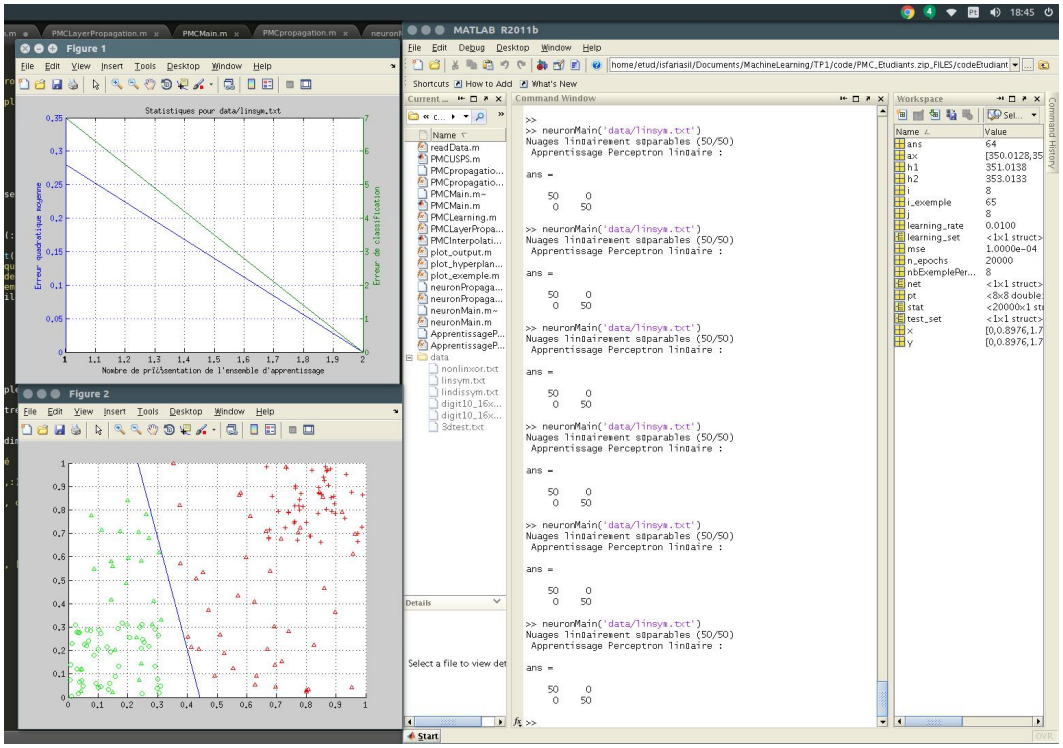

Figure 3: L'apprentissage avec seuil.

2-

Code

```
neuron.weights=
neuron.weights+learning_rate*(error_output.*patt_in)';
```

3-

Il influe sur la vitesse de convergence à chaque étape. Il represente la taille de pas.

4 - Il existe des exemples d'un même type que l'autre, la fonction linéaire est décalée.

5 -

Code

```
function output = neuron_propagation (neuron , exemple)
        %Pour TangH
         output=tanh(dot(neuron.weights',[exemple 1]));
end
```

Partie 3 - Le perceptron multicouches

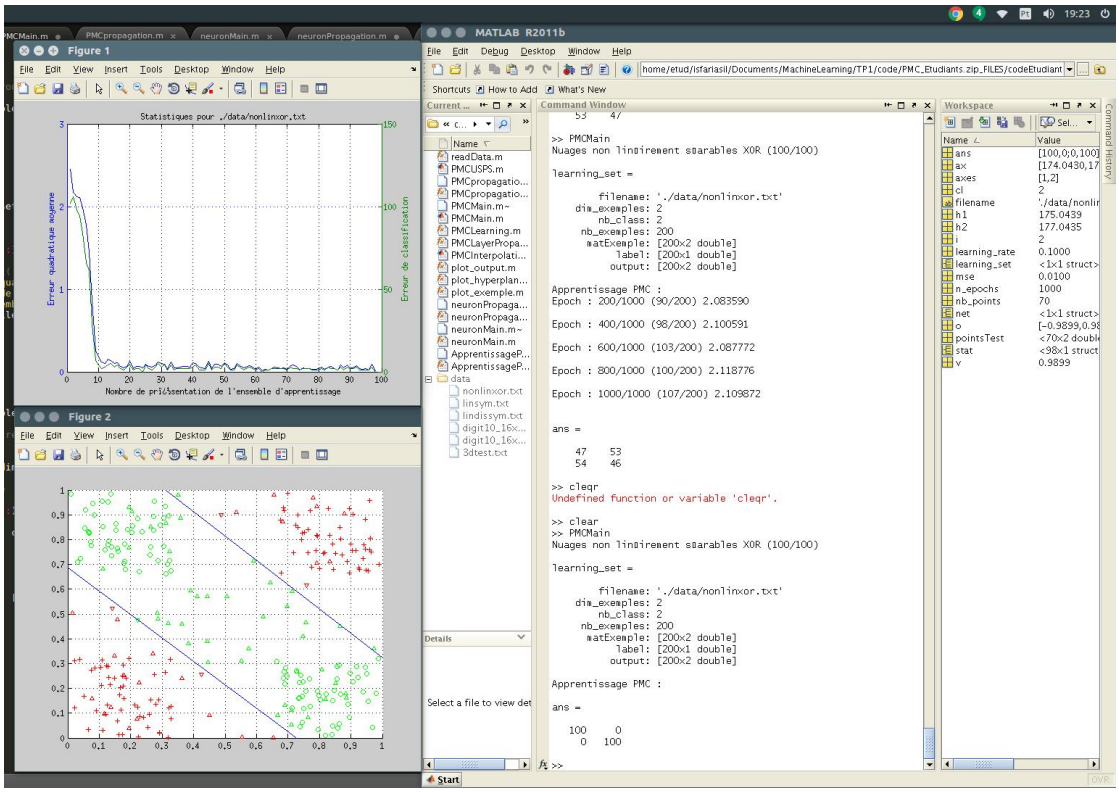

Figure 4: L'apprentissage du PMC avec données non lineaires.

6 -

Le PMC utilise la fonction d'activation TangH, alors ici, la dérivé a été utilisé, ce qui est **(1-tang(w'X)²)**. Le delta_HO est equivalent à delta_k qui a été presenté dans le diaporama du cours.

Code:

```
delta_HO = error_output.*(1.0-(net_output).^2.0);
```

7 -

Ici le calcul du vecteur delta_j a été calculé.

Code:

```
calcula_hidd=(1-hidden_output.^2);

tmpp_HO=net.HO;

%remove the artificial line - bias
tmpp_HO(net.dim_cachees+1,:)=[];

%le delta_J du diaporama
delta_IH = calcula_hidd.*(delta_HO*tmpp_HO');
```

8 -

Code:

```
%backpropagation!!! Slide 35 (NN.pdf)
net.HO =net.HO + learning_rate*[hidden_output 1]'*delta_HO;
net.IH = net.IH +learning_rate*(delta_IH'*[patt_in 1])';
```

9-

Code

```
function output = PMCLayerPropagation (input , weights)
%%Il y a le biais!
       output=tanh([input 1]*weights);
end
```

10 -

Le taux d'apprentissage définit la vitesse de convergence. Pour la valeur 0,1 n'est pas nécessaire 1000 fois. Mais il peut perdre de la puissance de généralisation.

11 -

Les hyperplanos sont définis en fonction de la qualité de la solution. Ils empirent lorsque la puissance de généralisation est moins.

Partie 4 - Etude d’un cas réel

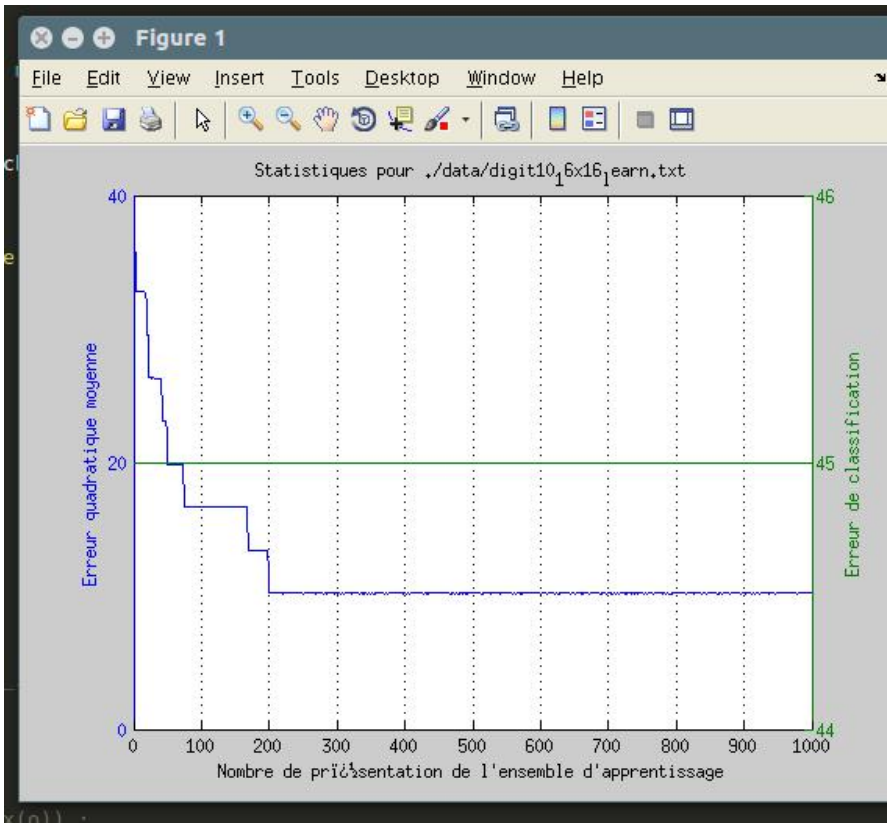

Figure 5: Convergente du PMC pour les données USPS.

12 -

La taux de succès est très faible. Seulemente 16% de succès..

13 -

Avec une taux d'apprentissage plus grand le système converge plus vite. Mais la taux d'erros est plus grand. Si on modifier le nombre de neurones, c'est nécessaire modifier la manière de l'apprendissage (comparer 2 types du caracter)

Partie 5 - Regression

14 - Code por générer le test.

```
test_set = [] ;
test_set.filename = '';
test_set.dim_exemples   = 2 ;   % surface dans R^3
test_set.nb_class     = 1 ;
test_set.nb_exemples   = 121 ;  % Nombre de points d'interpolation
test_set.pmatExemple=zeros(test_set.nb_exemples,
   test_set.dim_exemples) ;
test_set.label       = zeros (test_set.nb_exemples , 1) ;
test_set.output      = zeros (test_set.nb_exemples , 1) ;

nbExemplePerDim = sqrt(learning_set.nb_exemples) ;

x = linspace (0,2*pi,nbExemplePerDim) ;
y = linspace (0,2*pi,nbExemplePerDim) ;
pt = zeros (nbExemplePerDim , nbExemplePerDim) ;

for i = 1 : nbExemplePerDim
    for j = 1 : nbExemplePerDim
       pt(i,j) = PMCpropagation(net , [x(i) y(j)]);;
    end
end

figure,surf (x , y , pt) ;
```

Les Résultats:

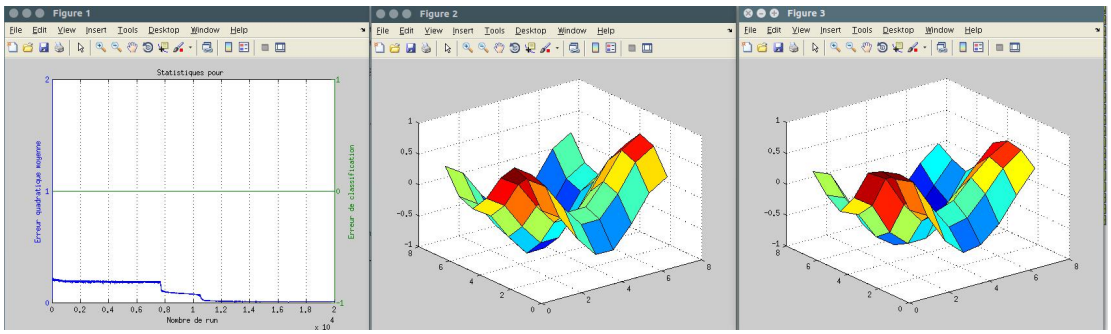

Figure 6.1: Convergente du PMC pour le Interpolation.
Figure 6.2: Surface interpolée pour les données de aprendissage.
Figure 6.3: Surface interpolée pour les données de **test**.

15 -

Avec la Figure 7 est possible vérifier que le modèle n'a pas bien adapté aux intervalles extrapolées. Tous les points entre 0 et 2*pi étaient bien prévus.

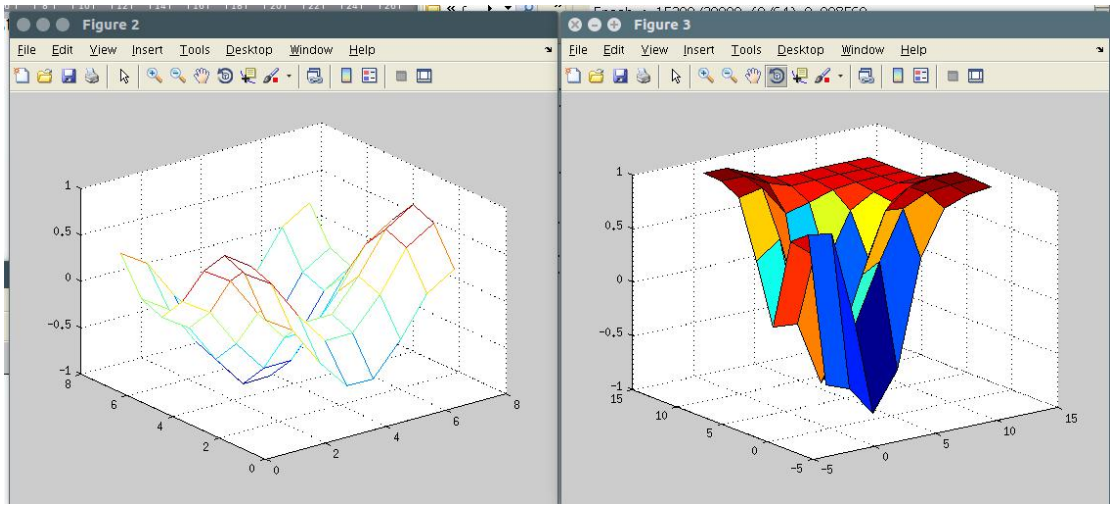

Figure 7.1: Surface interpolée pour les données de aprendissage.
Figure 7.2: Surface interpolée pour les données de test entre -1 et 4*pi.

16 -

Avec la Figure 8 est possible vérifier que le modèle avec 16 couches cachées le mieux adapté aux intervalles extrapolées. Tous les points entre 0 et 2*pi étaient mieux prévus.

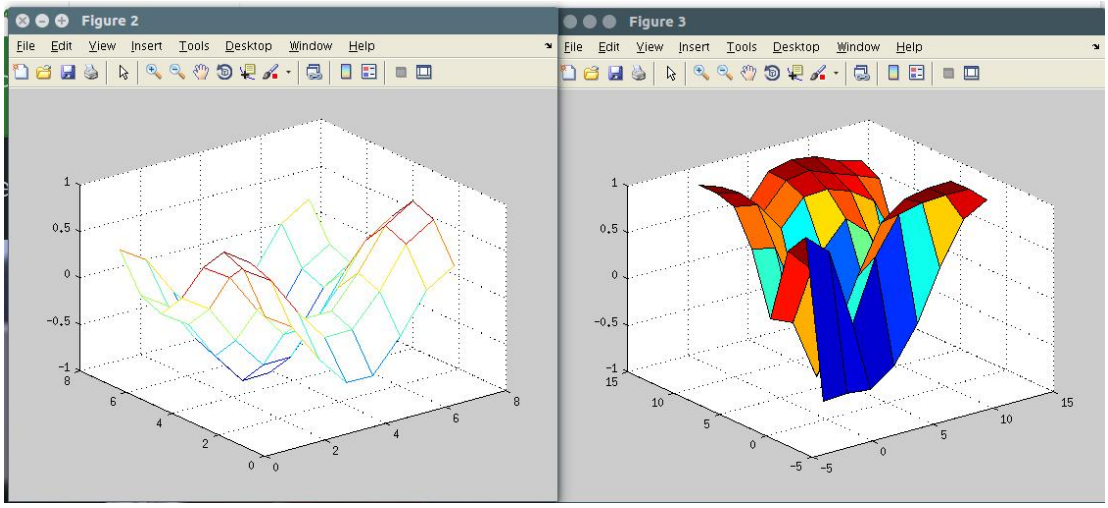

Figure 8.1: Surface interpolée pour les données de aprendissage entre 0 et 2*pi.
Figure 8.2: Surface interpolée pour les données de test entre -1 et 4*pi avec 16 couche cachées.