Cours-TD d'introduction à l'Intelligence Artificielle

Partie III

Le Perceptron et les réseaux mono-couche

Simon Gay

Introduction à l’Intelligence Artificielle

- **Menu :**

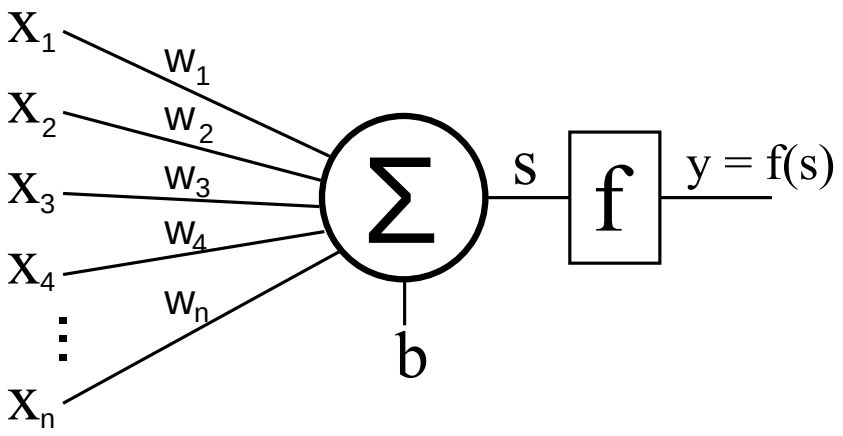
  - Théorie :
    - le neurone formel, pourquoi ça marche
    - Le principe du perceptron et réseaux mono-couche

  - Pratique :
    - implémentation d’un réseau mono-couche
    - Optimisation du réseau

Introduction à l'Intelligence Artificielle

- **Le neurone formel**

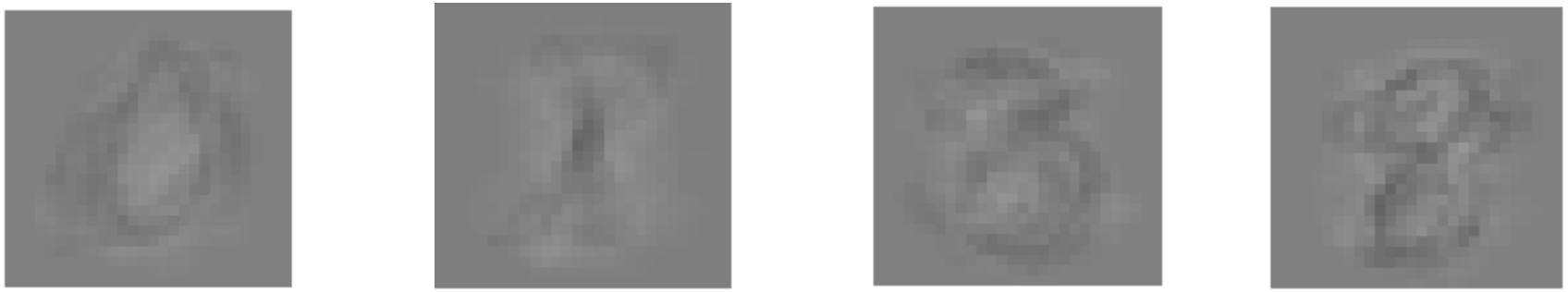
  - Un ensemble de poids
  - Apprentissage sur des exemples
  - Règle de Widrow-Hoff
  - Résultats concluants sur des exemples non connus



Introduction à l’Intelligence Artificielle

- **Le neurone formel**

  - Le neurone formel fait une moyenne des résultats positifs et des résultats négatif pour former une ‘image moyenne’ faisant ressortir les caractéristiques de l’élément à détecter

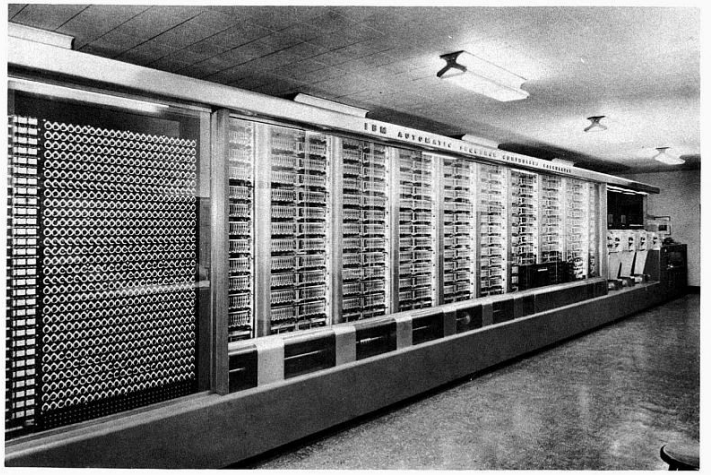


  - Cette ‘image moyenne’ permet de détecter l’élément sur un vecteur d’entrées non connu

  - **Mais pourquoi ça marche ?**

Introduction à l'Intelligence Artificielle

- **Le perceptron**


II Front View of the Calculator

  - Un des plus ancien réseau supervisé
    - inventé en 1957 par Frank Rosenblatt
    - Un ou plusieurs neurones formels
    - Règle de Hebbs (puis Widrow-Hoff plus tardivement)

  - Classifieur binaire : chaque neurone retourne 0 ou 1
    - Fonction d'activation à seuil

Introduction à l'Intelligence Artificielle

- **Fonction linéaire dans un plan**

  exemple :

  3.y + 2.x + 1 = 0 (y = -(2/3).x – 1/3)

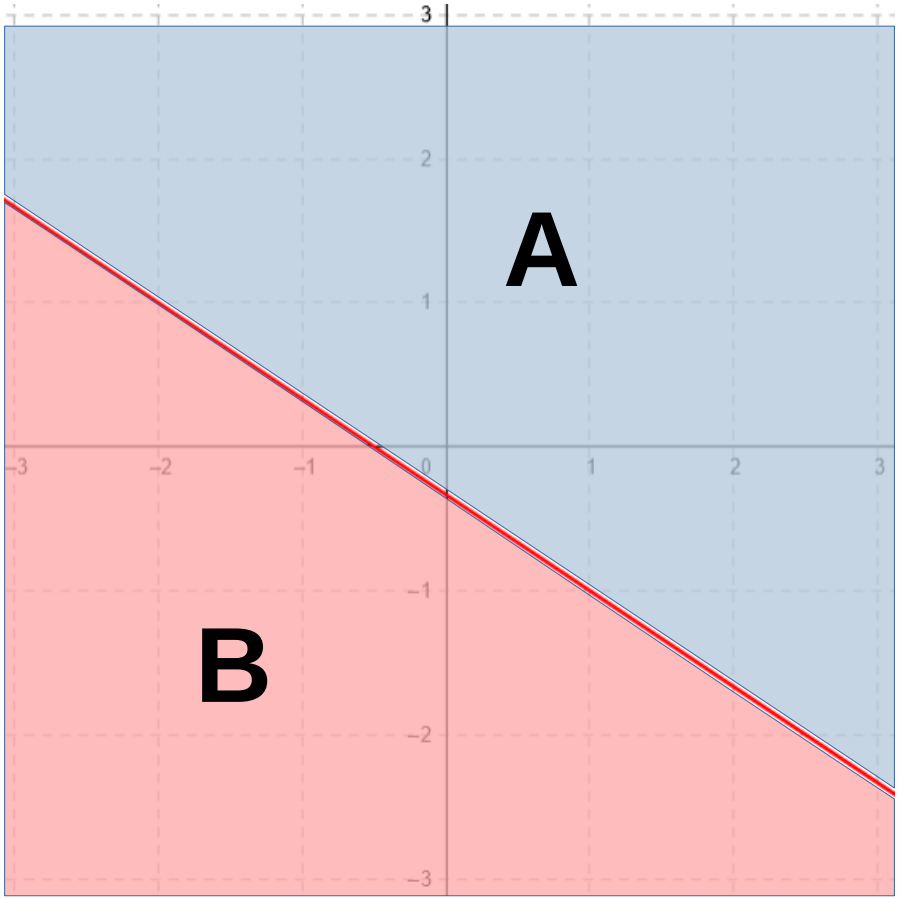
  Cette fonction sépare le plan en 2 :

  - Si 3.y+2.x+1 > 0

    → espace A au dessus de la droite

  - Si 3.y+2.x+1 < 0

    → espace B en dessous de la droite



Introduction à l'Intelligence Artificielle

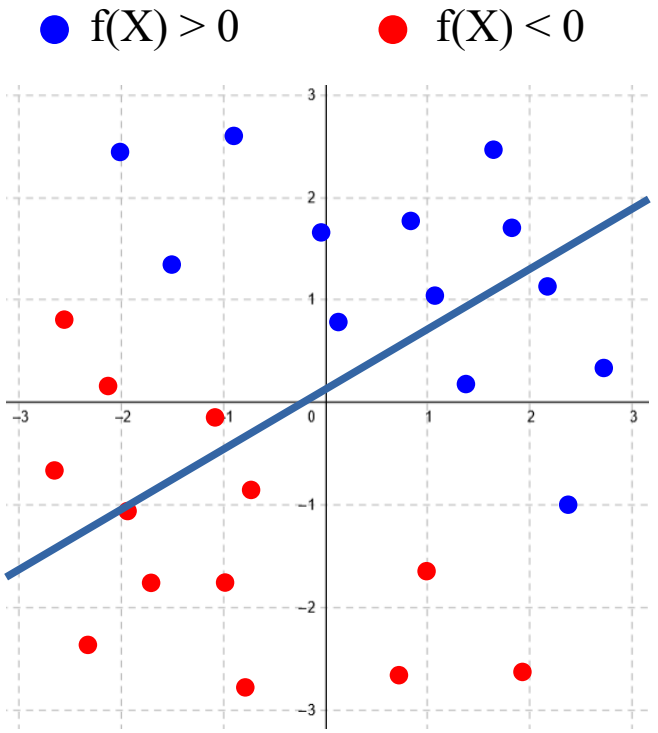
- **Et si on ne connaît pas la fonction ?**

  Soit une fonction $a.x + b.y + c = 0$

  Avec un ensemble de points $X_i = [x_i , y_i]$ tels que $f(X_i) > 0$ ou $f(X_i) < 0$

  → On doit trouver un triplet (a , b , c) respectant toutes les contraintes

- On prend un triplet au hasard
- On teste chaque exemple
  - Si $f(X_i)>0$ mais $ax_i+by_i+c<0$
    - Il faut augmenter ax_i+by_i+c
  - Si $f(X_i)<0$ mais $ax_i+by_i+c>0$
    - Il faut réduire ax_i+by_i+c
  - Sinon, on ne fait rien



Introduction à l'Intelligence Artificielle

- **Et si on ne connaît pas la fonction ?**

  Pour chaque paramètre a, b et c, il faut légèrement augmenter ou diminuer la valeur proportionnellement, respectivement, à x, y et 1

  $a \leftarrow a + \alpha \cdot x_i$ ou $a \leftarrow a - \alpha \cdot x_i$

  $b \leftarrow b + \alpha \cdot y_i$ ou $b \leftarrow b - \alpha \cdot y_i$

  $c \leftarrow c + \alpha \cdot 1$ ou $c \leftarrow c - \alpha \cdot 1$

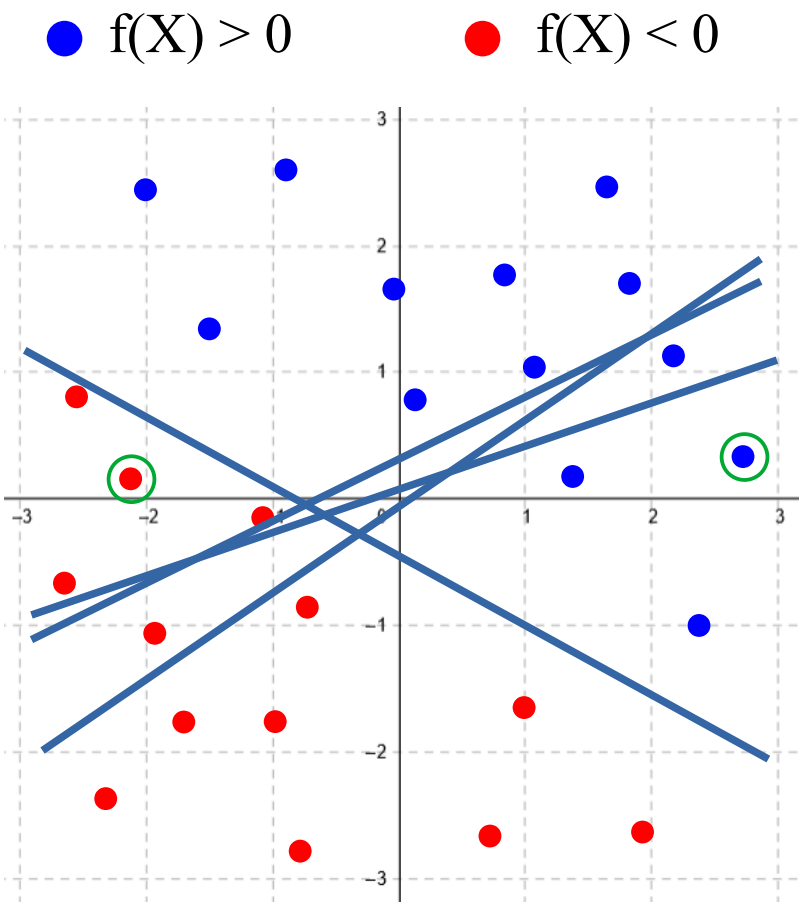
- **Si on note $r_i = 1$ si $f(X_i)>0$ et $r_i = -1$ si $f(X_i)<0$**
  - Alors
    - $a \leftarrow a + \alpha \cdot r_i \cdot x_i$
    - $b \leftarrow b + \alpha \cdot r_i \cdot y_i$
    - $c \leftarrow c + \alpha \cdot r_i$
  - On teste les points Xi, jusqu'à ce qu'ils soient tous du bon côté de la courbe

Introduction à l'Intelligence Artificielle

- **Les paramètres vont évoluer jusqu'à converger vers une solution**

- Algorithme :

erreurs = vrai
tant que erreurs faire
    erreurs = faux
    pour chaque Xi faire
        si $f(X_i) \cdot (a.x_i+b.y_i+c) < 0$ faire
            $a \mathrel{+}= \alpha \cdot r_i \cdot x_i$
            $b \mathrel{+}= \alpha \cdot r_i \cdot y_i$
            $c \mathrel{+}= \alpha \cdot r_i$
            erreurs = vrai
        fin si
    fin pour
fin tant que



- On ne met à jour que si il y a erreur

Introduction à l'Intelligence Artificielle

- **Et le perceptron dans tout ça ?**

- Écrivons :
  - $a . x + b . y + c = 0 \quad \rightarrow \quad w_1 . x_1 + w_2 . x_2 + b = 0$

- On généralise à un espace à n dimensions :

$$\sum_k w_k . x_k + b = 0$$

- Les poids d'un neurone formel forment l'équation d'un hyperplan
- L'apprentissage modifie les poids pour séparer l'espace en deux pour séparer deux groupes de points de cet espace

Introduction à l'Intelligence Artificielle

- **Quelques propriétés :**
- **Si l'ensemble d'exemples peut être séparé par un plan, alors :**

  - **Convergence assurée en un nombre fini d'étapes**
    - **Quel que soit le nombre d'exemples**
    - **Quelle que soit la distribution**
    - **Quel que soit le nombre de dimensions de l'espace**

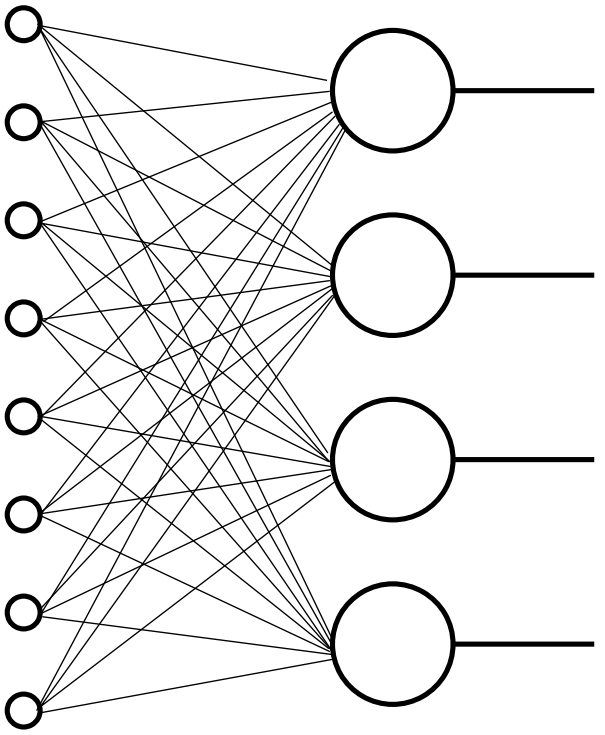
Introduction à l'Intelligence Artificielle

- **Perceptron avec plusieurs neurones**

- **Utilisé pour définir plus de deux classes**

  - Possibilité d'une sortie sur plusieurs bits
    - Exemples : code ascii d'une lettre, conversion binaire vers afficheur 7 segment…
  - Un neurone par classe
    - Compétition entre les neurones, peu utilisée car possibilité d'égalité entre deux neurones

      → neurones à fonction d'activation continue



Introduction à l'Intelligence Artificielle

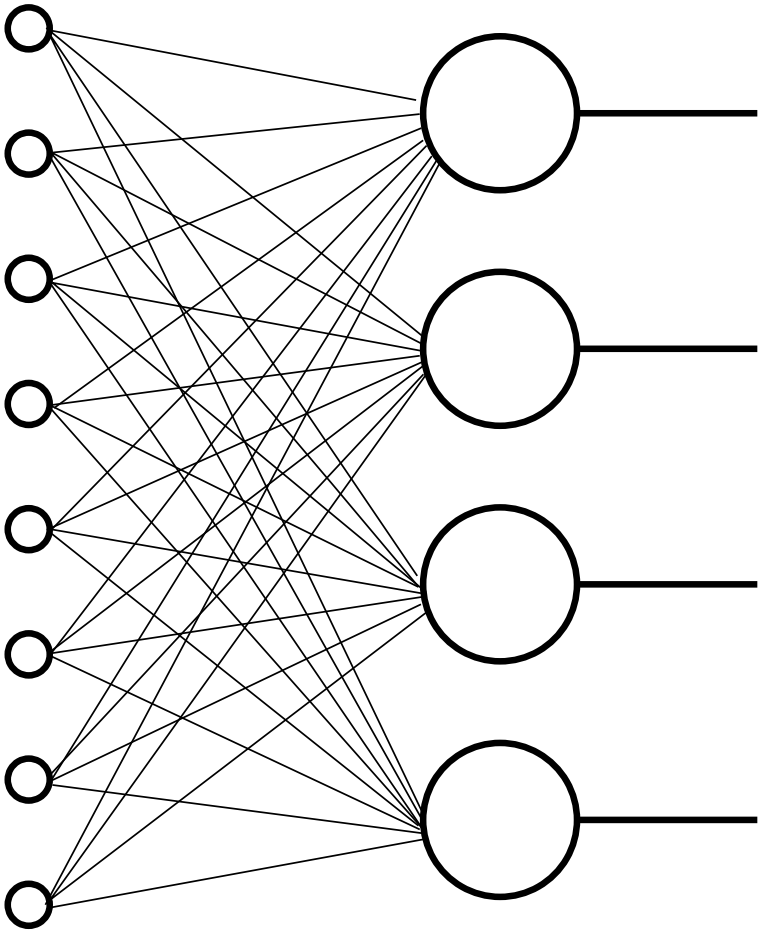
- **Perceptron avec fonction d'activation continue : le réseau mono-couche**

- Fonctions sigmoïde, tanh, linéaire, RELU ...
  - Ajoute une information supplémentaire : le niveau de confiance dans le résultat
    - Plus on est proche de 0 ou de 1, plus on est sûr du résultat
    - Compétition entre les neurones : on considère le neurone le plus actif (pas de seuillage des résultats, résolution d'ambiguïté)

(1 : 0,013) (2 : 0,658) (3 : 0,553) (4 : 0,350) (5 : 0,112) (6 : 0,092)...

- Apprentissage par la méthode Widrow-Hoff
  - Prend en compte l'erreur → recherche plus efficace de solutions

Introduction à l'Intelligence Artificielle

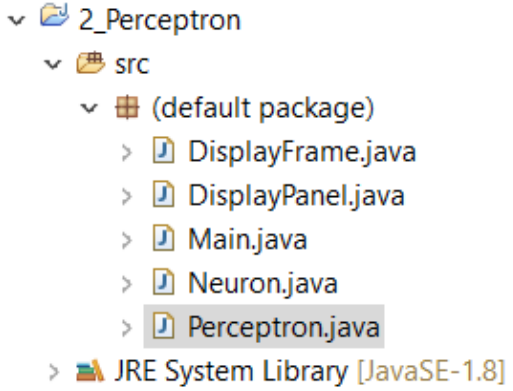


Passons à la pratique !

Introduction à l'Intelligence Artificielle

- **Nous allons implémenter un réseau mono-couche pour reconnaître les chiffres**

- Pour commencer :
  - Dupliquez le projet 1_neuron et appelez la copie 2_perceptron
  - Vérifiez que le nouveau projet est toujours fonctionnel
  - Ajoutez une classe 'Perceptron' au projet



Introduction à l'Intelligence Artificielle

- **Un réseau mono-couche n'est rien de plus qu'une liste de neurones formels**

- Dans la classe Perceptron, ajoutez :
  - un vecteur (tableau) de Neuron que vous appelerez layer
  - Un vecteur de float 'results' pour collecter les résultats
  - Un float 'sum_delta' (on s'en servira pour mesurer les performances)

- Ajoutez le constructeur de la classe perceptron
  - Paramètres : nombre d'entrées, nombre de sorties
  - Initialisez correctement les vecteurs

Introduction à l'Intelligence Artificielle

- **Un réseau mono-couche n'est rien de plus qu'une liste de neurones formels**

```
2  public class Perceptron {
3
4
5      public Neuron[] layer;
6      public float[] result;
7      public float sum_delta;
8
9
.0     public Perceptron(int nb_input, int nb_output){
.1
.2         layer=new Neuron[nb_output];
.3         for (int i=0;i<layer.length;i++) layer[i]=new Neuron(nb_input);
.4
.5         result=new float[layer.length];
.6     }
.7
```

Introduction à l'Intelligence Artificielle

- **Un réseau mono-couche n'est rien de plus qu'une liste de neurones formels**

- Ajoutez une fonction 'compute' qui doit calculer la sortie de chaque neurone.

- Ajoutez une fonction 'learn' qui applique l'apprentissage sur chaque neurone
  - Pensez aux paramètres de cette fonction, et comment ils sont transmis aux neurones :
    - Le vecteur d'entrées
    - Le vecteur de sorties
  - La fonction doit réinitialiser le sum_delta et ajouter la valeur absolue du delta de chaque neurone

Introduction à l'Intelligence Artificielle

- **Un réseau mono-couche n'est rien de plus qu'une liste de neurones formels**

```
public void compute(float[] input){
    for (int i=0;i<layer.length;i++){
        result[i]=layer[i].compute(input);
    }
}

public void learn(float[] input, int[] output){
    sum_delta=0;
    for (int i=0;i<layer.length;i++){
        layer[i].learn(input, output[i]);
        sum_delta+=Math.abs(layer[i].delta);
    }
}
```

- **Le réseau est prêt à l'emploi !**

Introduction à l'Intelligence Artificielle

- **Intégration du réseau mono-couche**
- Dans Main, remplacez le pointeur du Neuron par un Perceptron
  - Paramètres pour instancier le perceptron :
    - En entrée : toujours *size_x*size_y* éléments
    - En sortie : *nb_values* sorties

```
private DisplayFrame display;       // display panel

//-----------------
public Perceptron perceptron;       // perceptron
```

```
// initialize structures
img=new float[size_x*size_y];
perceptron=new Perceptron(size_x*size_y, nb_values);
```

Introduction à l'Intelligence Artificielle

- **Intégration du réseau mono-couche**

- Le résultat est un vecteur dont un seul élément est à 1 (la bonne réponse)
- Corrigez les appels de fonction *compute* et *learn*

```
int res=0;
if (y==number) res=1;

neuron.compute(img);
neuron.learn(img, res);
```

```
int[] output=new int[nb_values];
output[y]=1;

perceptron.compute(img);
perceptron.learn(img, output);
```

```
float res=neuron.compute(img);
```

```
perceptron.compute(img);
```

Introduction à l'Intelligence Artificielle

- **Modification de l'affichage**

- Nous devons afficher les poids des dix neurones du réseau
- Agrandissez le Frame d'affichage (700px au lieu de 500)
- On va afficher les neurones sur deux rangées (avec / et %)

```
for (int n=0;n<Main.nb_values;n++){
    for (int i=0;i<Main.size_x;i++){
        for (int j=0;j<Main.size_y;j++){
            val=(int)(main.perceptron.layer[n].synaps[i+Main.size_x*j]*50)+128;
            if (val<0) val=0;
            if (val>255) val=255;
            g.setColor(new Color(val,val,val));
            g.fillRect(180+3*i+100*(n%5), 10+3*j+100*(n/5), 3, 3);
        }
    }
}
```

Introduction à l'Intelligence Artificielle

- **Analyse de l'apprentissage**

- Modifiez la lecture du delta pour permettre l'affichage du delta moyen

```
sumdelta+=Math.abs(neuron.delta);
display.repaint();
try {Thread.sleep(10);
} catch (InterruptedException e) {e.pri
```

→

```
sumdelta+=perceptron.sum_delta;
display.repaint();
try {Thread.sleep(10);
} catch (InterruptedException e) {e.pr
```

```
System.out.println("epoch n°"+epoch+" : "+(sumdelta/(nb_values*10 * nb_samples)));
```

Introduction à l’Intelligence Artificielle

- **Analyse des résultats**

- Après l’apprentissage, on va compter le nombre d’erreurs sur le jeu d’essais

- Pour chaque test, récupérez l’index du neurone le plus actif
- Comparez-le au nombre testé
- Si l’index et le nombre ne coïncident pas, incrémentez une variable

- Affichez le nombre d’erreurs après les tests

Introduction à l'Intelligence Artificielle

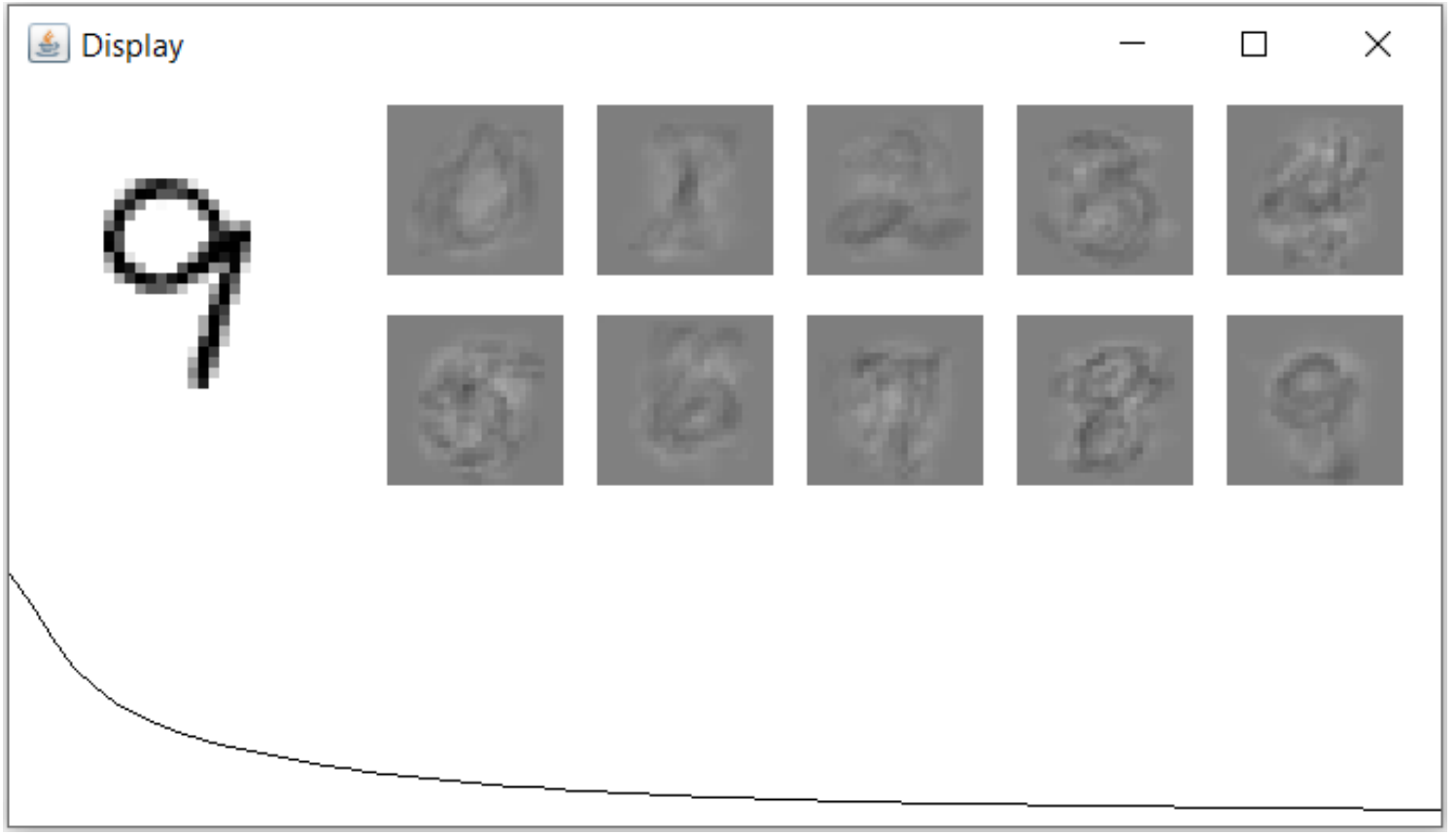
- **Analyse des résultats**

```
                perceptron.compute(img);

                int imax=0;
                float max=0;
                for (int i=0;i<perceptron.result.length;i++){
                    if (perceptron.result[i]>max){
                        imax=i;
                        max=perceptron.result[i];
                    }
                }

                if (imax!=y) errors++;
            }
        }

        System.out.println("nb errors : "+errors);

    }
```

- **Testez votre réseau mono-couche !**

Introduction à l'Intelligence Artificielle

- **Analyse des résultats**

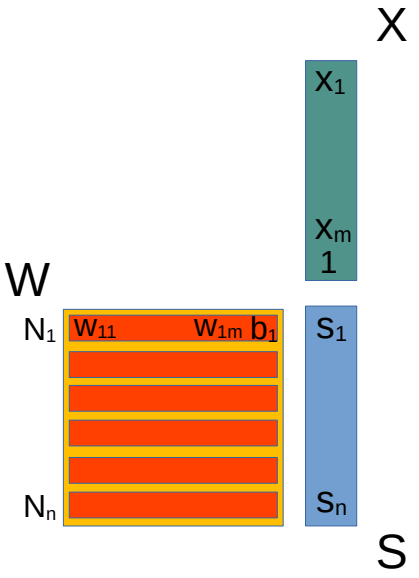


- **50 epochs → 8 erreurs ; 100 epoch → 6 erreurs ; 200 epochs → 5 erreurs**

Introduction à l'Intelligence Artificielle

- **Un peu d'optimisation algorithmique**
- **La classe perceptron fait appel aux instances de neurones**

- **Pour chaque neurone *n* : $s_n = W_n . X$**
- **Du point de vue du réseau, on peut regrouper les vecteurs de poids dans une matrice W et les résultats dans un vecteur S :**
  - **$S = W . X$**



- **Nous allons supprimer la classe Neuron et intégrer le calcul matriciel dans la classe Perceptron**

Introduction à l'Intelligence Artificielle

- **Un peu d'optimisation**

- Dupliquez le projet 2_Perceptron et appelez-le 2_PerceptronV2

- Dans la classe Perceptron, remplacez le vecteur de Neuron par
  - Une matrice 'weights' pour les poids (le biais est dans cette matrice)
  - Un vecteur 'result' pour récupérer les sorties des neurones
  - Un vecteur 'deltas' pour enregistrer les deltas
  - Récupérez le learnrate et la fonction d'activation de la classe Neuron

```
2  public class Perceptron {
3
4      public float learnRate=0.01f;
5
6      public float[][] weights;
7      public float[] deltas;
8      public float[] result;
9
10     public float sum_delta;
11
12     public Perceptron(int nb_input, int nb_output){
13         weights=new float[nb_output][nb_input+1];
14         deltas=new float[nb_output];
15         result=new float[nb_output];
16     }
```

Introduction à l'Intelligence Artificielle

- **Un peu d'optimisation**

- Modifiez la fonction compute pour effectuer le calcul des neurones
  - Récupérez la fonction d'activation du neurone

```
public void compute(float[] input){
    for (int n=0;n<result.length;n++){
        result[n]=0;
        for (int i=0;i<input.length;i++){
            result[n]+=weights[n][i]*input[i];
        }
        result[n]+=weights[n][input.length];
        result[n]=activation(result[n]);
    }
}
```

Introduction à l'Intelligence Artificielle

- **Un peu d'optimisation**

- Modifiez la fonction learn pour calculer le delta de chaque neurone, les additionner et mettre à jour les poids

```
public void learn(float[] input, int[] output){

    sum_delta=0;
    for (int n=0;n<deltas.length;n++){
        deltas[n]=output[n]-result[n];
        sum_delta+=Math.abs(deltas[n]);
    }

    for (int n=0;n<result.length;n++){
        for (int i=0;i<input.length;i++){
            weights[n][i]+=learnRate * deltas[n] * input[i];
        }
        weights[n][input.length]+=learnRate * deltas[n];
    }
}
```

Introduction à l'Intelligence Artificielle

- **Un peu d'optimisation**
- Corrigez les pointeurs dans l'afficheur

```
for (int i=0;i<Main.size_x;i++){
    for (int j=0;j<Main.size_y;j++){
        val=(int)(main.perceptron.weights[n][i+Main.size_x*j]*50)+128;
        if (val<0) val=0;
        if (val>255) val=255;
```

- Supprimez la classe Neuron (vérifiez qu'il n'y a pas d'erreurs)
- Testez le réseau mono-couche : les résultats doivent être identiques
- **Conservez bien votre projet, il servira de base au prochain TP !**