Cours-TD d'introduction à l'Intelligence Artificielle

Partie II

Le Neurone Formel

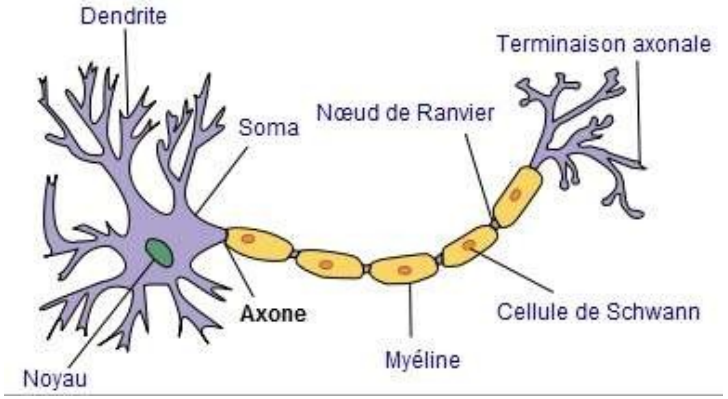
Simon Gay

Introduction à l’Intelligence Artificielle

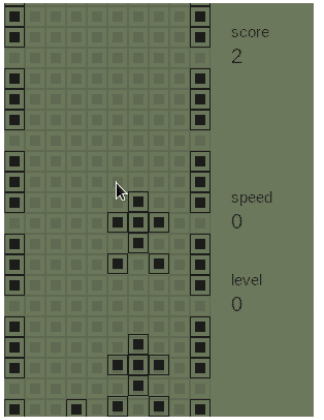
- **Menu :**
  - Théorie :
    - le neurone formel
  - Pratique :
    - implémentation d’un neurone formel
    - Création d’outils pour l’entraînement et le test de neurones
    - Apprentissage sur une base d’images

Introduction à l'Intelligence Artificielle

- **Vers un modèle simplifié du neurone :**
  - Le neurone biologique :
    - Des dendrites qui collectent les signaux d'entrée
    - Un corps qui intègre les signaux
    - Un axone pour transmettre le signal
    - Un fonctionnement par impulsion (fréquence)



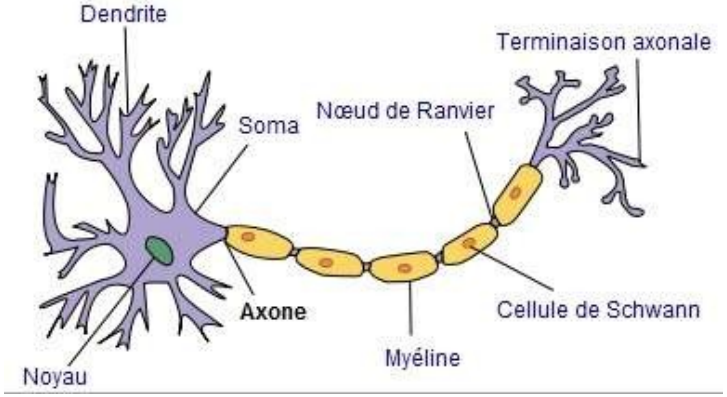
  - Il faut simplifier le modèle et l'adapter au fonctionnement d'un ordinateur
    - /!\ Le neurone formel est une version TRES simplifiée du neurone biologique



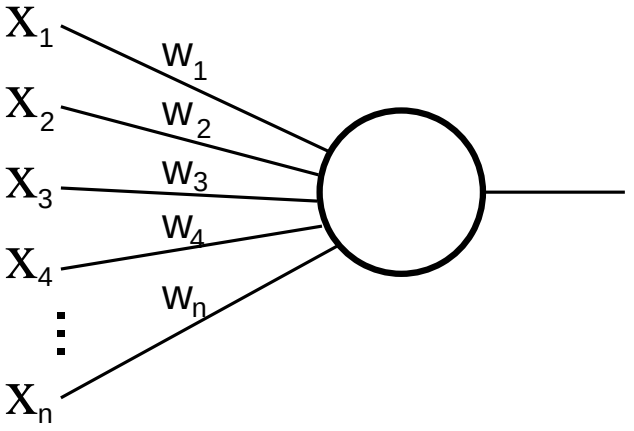
https://github.com/tshiamobhuda/Racing

Introduction à l'Intelligence Artificielle

- **Vers un modèle simplifié du neurone :**
  - Le neurone biologique :
    - Des dendrites qui collectent les signaux d'entrée
    - Un corps qui intègre les signaux
    - Un axone pour transmettre le signal
    - Un fonctionnement par impulsion (fréquence)

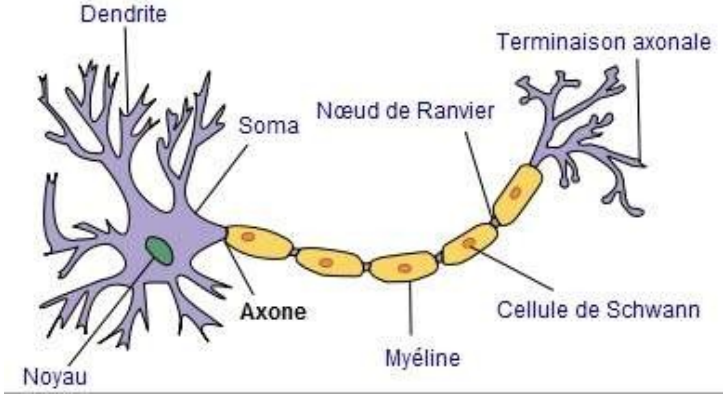


  - Il faut simplifier le modèle et l'adapter au fonctionnement d'un ordinateur
    - On remplace la fréquence par une valeur réelle
    - En entrée : un vecteur de valeurs
      - Vecteur $X = \{ x_1, x_2, x_3, \dots , x_n \}$
    - Un vecteur de réel pour les synapses
      - Vecteur $W = \{ w_1, w_2, w_3, \dots , w_n \}$

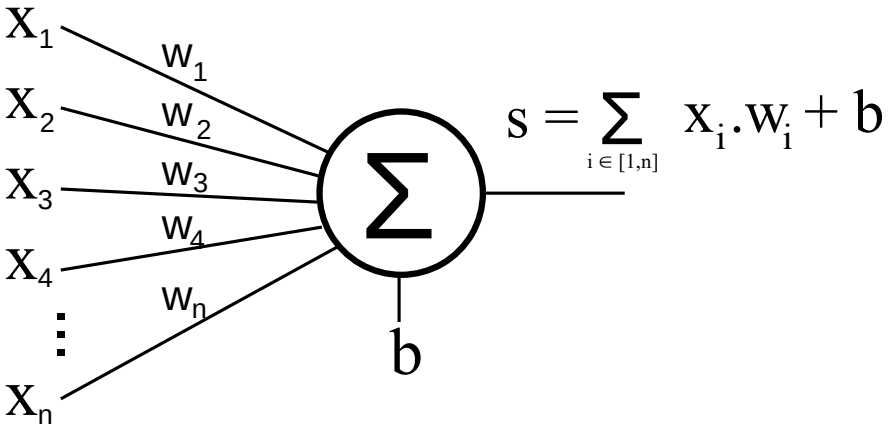


Introduction à l'Intelligence Artificielle

- **Vers un modèle simplifié du neurone :**
  - Le neurone biologique :
    - Des dentrides qui collectent les signaux d'entrée
    - Un corps qui intègre les signaux
    - Un axone pour transmettre le signal
    - Un fonctionnement par impulsion (fréquence)

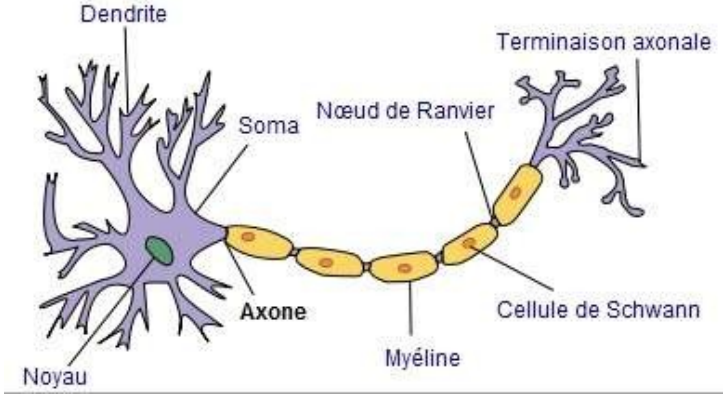


  - Il faut simplifier le modèle et l'adapter au fonctionnement d'un ordinateur
    - Pour l'intégration des entrées :
      - Somme des entrées pondérées par les poids synaptiques
      - Un biais b

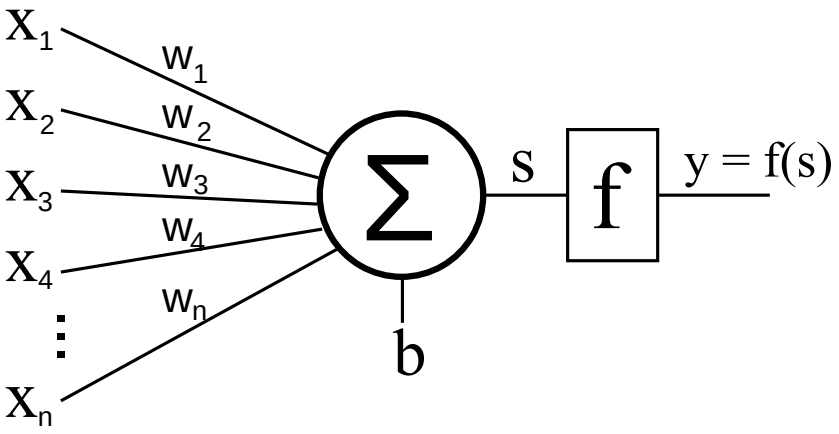


Introduction à l'Intelligence Artificielle

- **Vers un modèle simplifié du neurone :**
  - Le neurone biologique :
    - Des dentrides qui collectent les signaux d'entrée
    - Un corps qui intègre les signaux
    - Un axone pour transmettre le signal
    - Un fonctionnement par impulsion (fréquence)



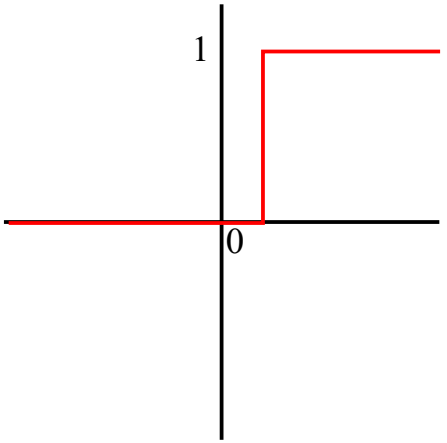
  - Il faut simplifier le modèle et l'adapter au fonctionnement d'un ordinateur
    - La sortie passe par une fonction d'activation
      - Permet de restreindre la valeur



Introduction à l'Intelligence Artificielle

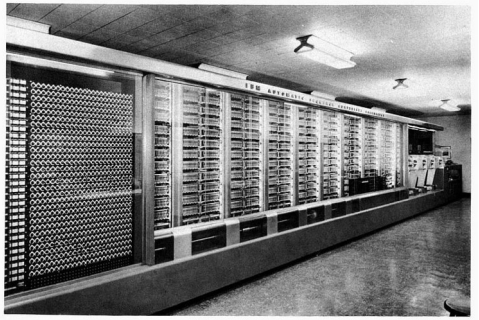
- **Quelle fonction d'activation ?**

  - Fonction seuil : S = 1 si y>seuil, S=0 sinon



(note : la valeur du seuil n'a pas d'importance grâce au biais)

  - Fonction utilisée sur les premiers modèles de neurones formels (proposé par Warren McCulloch et Walter Pitts en **1943**)

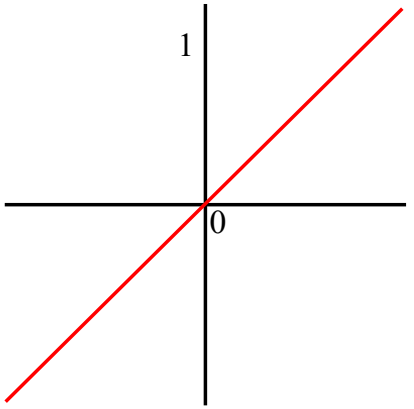

II Front View of the Calculator

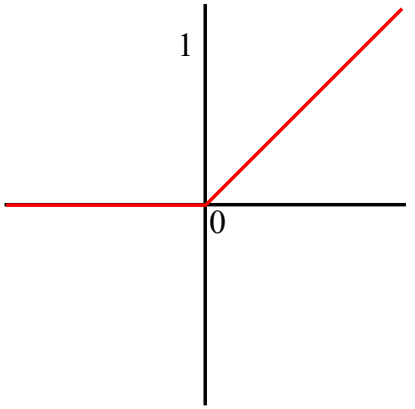
  - Quelques limitations :
    - On ne peut pas comparer les valeurs de plusieurs neurones
    - Les valeurs de l'information ne peuvent pas être « transmises »

Introduction à l'Intelligence Artificielle

- **Quelle fonction d'activation ?**

  - Fonctions linéaires et ReLU (Rectified Linear Unit)
    - y = x
    - y = max(0,x)





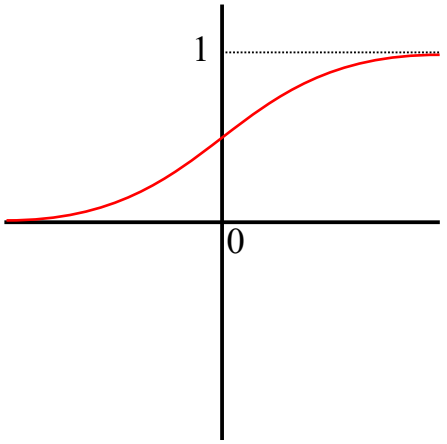
  - Fonctions très simples
  - ReLU permet de supprimer les valeurs négatives
    - très utilisée dans les approches de deep learning

  - La valeur de sortie peut dépasser 1 (la somme pondérée doit être égale à 1)
  - Gradient constant (une variation du poids a un même effet sur la sortie)

Introduction à l'Intelligence Artificielle

- **Quelle fonction d'activation ?**
  - Fonction sigmoïde
    - $y = \frac{1}{1 + e^{-x}}$

      (note : la fonction tanh est aussi utilisée)



  - Fonctions plus complexe
  - Toute grande valeur donne un résultat proche de 1 (ou de 0 en négatif)
  - Le gradient tend vers 0 pour les grandes valeurs
    - Stabilisation des poids (variations de plus en plus faible, mais toujours existantes)
    - Mais « perte » du gradient (problématique avec un grand nombre de couches)
  - Autre avantage que nous verrons plus tard

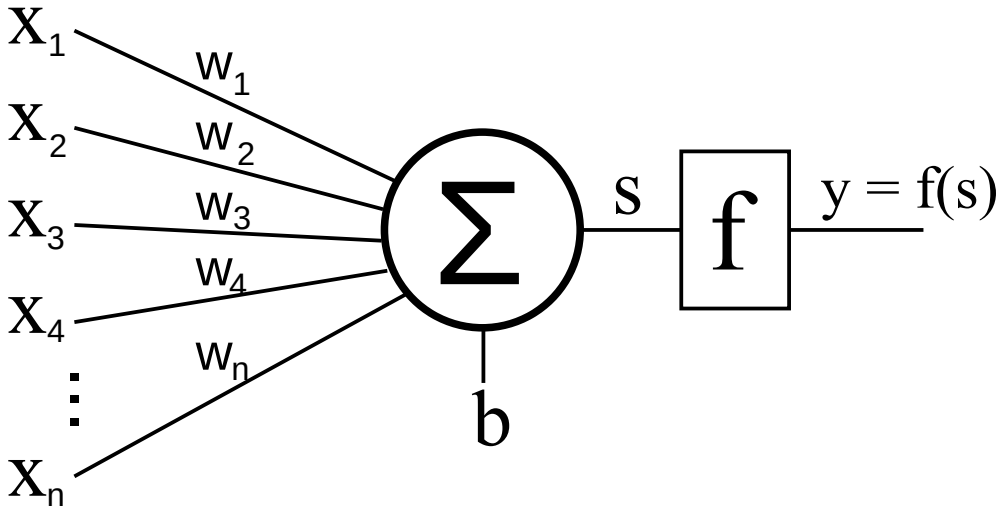
Introduction à l'Intelligence Artificielle

- **Vers un modèle simplifié du neurone :**
  - Le neurone formel
    - Un vecteur d'entrée X
    - Un ensemble de poids W
    - On ajoute un biais b
      - Équivalent à une entrée à 1

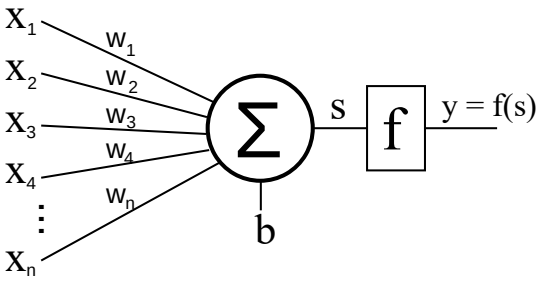
$$s = \sum_{i \in [1,n]} x_i . w_i$$

    - Une fonction d'activation

      Y = f (S)



Introduction à l'Intelligence Artificielle



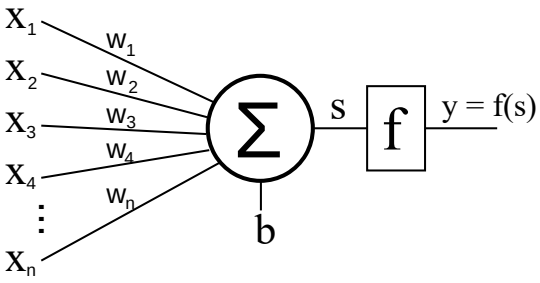
- **Méthode d'apprentissage supervisé**

  - Loi de Hebb (1949) : "*cells that fire together, wire together*"
    - $w_i^{t+1} = w_i^t + \alpha \, . \, x_i \, . \, r$: si *r* et x_i sont de même signe, w_i est renforcé

      α est le taux d'apprentissage (limite la variation des poids)

  - Règle Delta : modifier chaque poids w_i proportionnellement à :
    - la valeur d'entrée (plus l'entrée a une valeur élevée, plus elle influe sur le résultat)
    - la différence entre le résultat et la sortie attendue (plus l'écart est grand plus il faut modifier les poids)

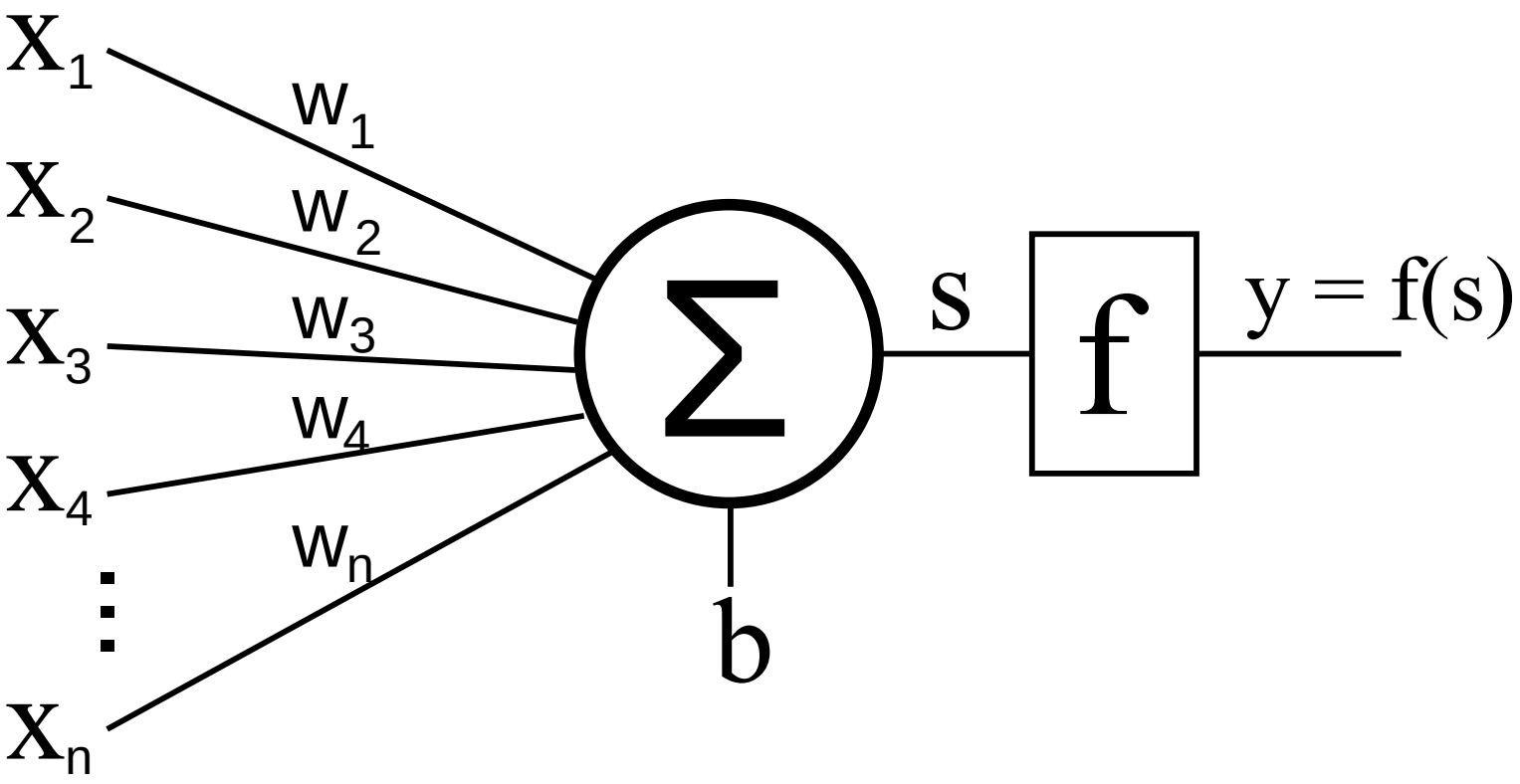
Introduction à l'Intelligence Artificielle



- **Méthode d'apprentissage supervisé**
  - Loi Delta : on calcule l'écart entre entre le résultat r voulu et le résultat y du neurone :
    - $\Delta = r - y$
    - Perceptron (Rosenblatt, 1957) : fonction d'activation seuil, donc $\Delta \in \{ -1 ; 0 ; 1 \}$
    - Widrow-Hoff rule (1960) : fonctions d'activation continues, donc $\Delta \in \mathbb{R}$
  - On met chaque poids à jour :
    - Principes : les poids sont modifiés en fonction de :
      - La valeur d'entrée (une entrée 'active' aura plus d'impact sur la sortie)
      - La différence (Δ) entre la sortie et le résultat (plus l'erreur est grande, plus il faut corriger les poids)
    - $w_i^{t+1} = w_i^t + \alpha \cdot x_i \cdot \Delta$

      α est le *taux d'apprentissage.* Même mise à jour pour le biais b
  - L'apprentissage s'effectue sur un (très) grand nombre d'exemples

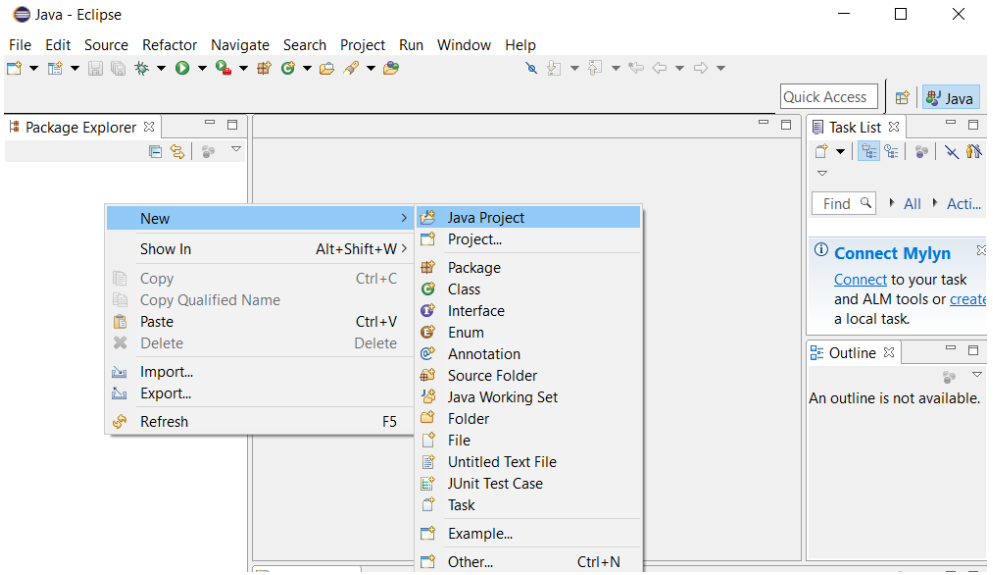
Introduction à l'Intelligence Artificielle

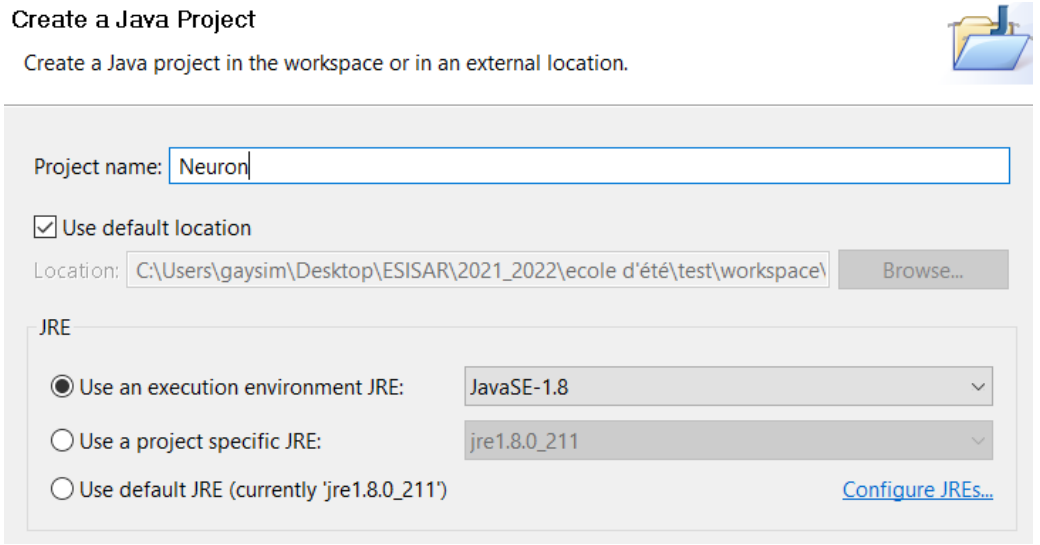


Passons à la pratique !

Introduction à l'Intelligence Artificielle

- **L'environnement**
  - Ouvrez Eclipse
  - Créez un nouveau projet Java '1_Neuron'
    - Clic droit dans package explorer → new → Java project, environnement 'JavaSE-1.8' (ou 'J2SE-1.5')





Introduction à l'Intelligence Artificielle

- **L'environnement**
  - Créez une classe **Main** et une classe **Neuron**
    - Clic droit sur le projet → new → class
    - Pensez à cocher 'public static void main(String[ ] args)' pour la classe Main

```
1
2
3  public class Main {
4
5
6      public Main(){
7
8      }
9
10     public static void main(String[] args) {
11
12         new Main();
13     }
14
15 }
16
```

Introduction à l'Intelligence Artificielle

- **L'environnement**

  - Téléchargez sur ***https://github.com/gaysimon/MachineLearning*** les classes DisplayFrame et DisplayPanel, et intégrez-les au projet (clic droit sur le projet→ import → File System)

  - Ajoutez dans main une instance de *Neuron* et de *DisplayFrame*, que vous initialiserez

```
3  public class Main {
4
5      private DisplayFrame display;        // display panel
6
7      public Neuron neuron;                // single neuron
8
9      public Main(){
0
1          neuron=new Neuron();
2
3          display=new DisplayFrame(this);
4      }
5
6      public static void main(String[] args) {
7          new Main();
8      }
9  }
```

Introduction à l'Intelligence Artificielle

- **Le neurone**
  - Dans la classe *Neuron* :
    - Un vecteur (tableau) de float pour les poids
    - Un float pour le biais
    - Un float pour le résultat
    - Un float pour le delta (→ on l'utilisera pour mesurer les performances)
  - Le constructeur de *Neuron* doit avoir le nombre de poids en paramètre
    - Initialisez le vecteur de poids avec ce nombre
      - Dans Main, vous ajouterez la valeur 1 au constructeur du neurone
  - Créez une fonction **`float compute(float[ ] img)`**
    - Implémentez la fonction qui calcule le résultat du neurone
    - Écrivez la fonction d'activation sigmoide à part (note : **`Math.exp( )`**)
    - La fonction doit enregistrer le résultat dans la variable **`output`**

Introduction à l'Intelligence Artificielle

- **Le neurone**

```java
public class Neuron {

    public float[] synaps;
    public float bias;
    public float output=0;
    public float delta=0;

    ////////////////////////////////////////
    public Neuron(int size){
        synaps=new float[size];
    }
```

```java
    ////////////////////////////////////////
    public float compute(float[] img){

        float sum=0;
        for (int i=0;i<synaps.length;i++){
            sum+=img[i]*synaps[i];
        }
        sum+=bias;

        output=activation(sum);

        return output;
    }

    ////////////////////////////////////////
    private float activation(float x){
        return (float) (1 / (1+Math.exp(-x)));
    }
}
```

Introduction à l’Intelligence Artificielle

- **Le neurone**
  - Implémentez la fonction `void learn(float[] img, int res)`
    - Ajoutez à la classe une variable learnRate initialisée à 0,01
    - N’oubliez pas le biais !
    - Enregistrez le delta avec la variable globale **`delta`**

Introduction à l'Intelligence Artificielle

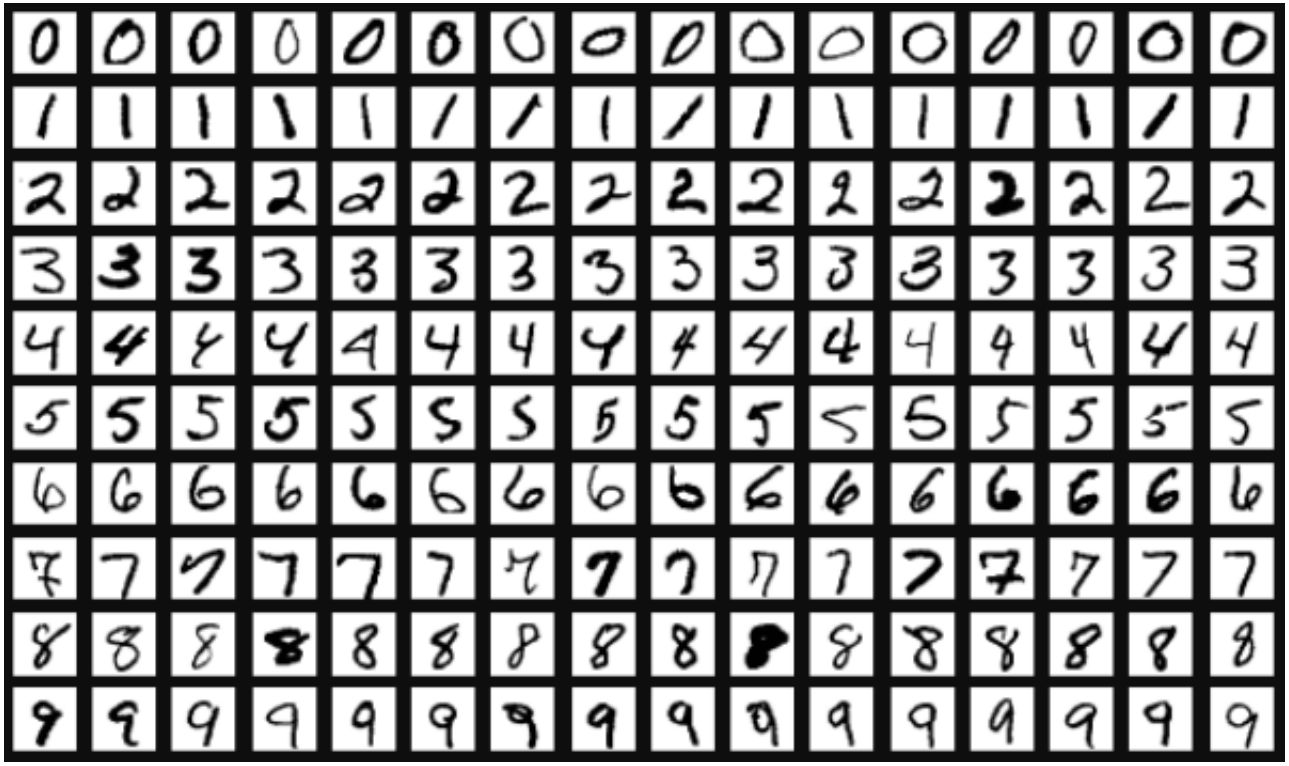
- **Le neurone**

```
2  public class Neuron {
3
4      public float learnRate=0.01f;
5
```

```
29
30     ////////////////////////////////////////
31     public void learn(float[] img, int res){
32
33         delta=res-output;
34
35         for (int i=0;i<synaps.length;i++){
36             synaps[i]+= learnRate * delta * img[i];
37         }
38         bias+=learnRate * delta;
39     }
```

Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**
  - Nous allons utiliser une base d'image issue de la base MNIST
  - Téléchargez l'image MnistExamples.png → 10 chiffres, 16 exemples
    - Imagettes de 28 x 27 pixels



Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**

- Dans la classe Main :
  - Définissez une variable *static String PATH* avec le chemin vers l'image
    - Ex : public static String PATH= ''/chemin/vers/mon/image/''
  - Chargez l'image au format *BufferedImage*

```
12    private BufferedImage image=null;   // image buffer
```

```
45        // load image
46        try {
47            image = ImageIO.read(new File(PATH+"MnistExamples.png"));
48        } catch (IOException e) {System.out.print(e);}
49
```

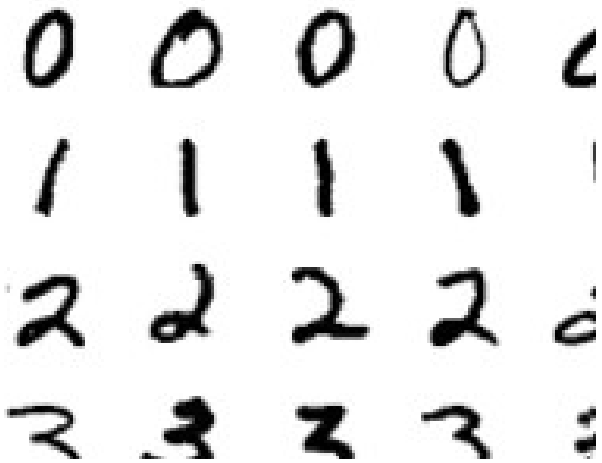
Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**

- Dans la classe Main :
  - On va ajouter les valeurs des offsets pour parcourir l'image globale
    - Taille des imagettes : 28 x 27 pixels
    - Images décalées de 35 pixels horizontalement et 33 pixels verticalement
    - 10 chiffres possibles, 16 exemples de chaque

35px

33px



```
public static int size_x=28;    // size of a digit image
public static int size_y=27;

public static int offset_x=35;  // offset between digit images
public static int offset_y=33;

public static int nb_values=10;     // 10 digits
public static int nb_samples=16;    // number of occurrences
```

Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**

- Dans la classe Main :
  - Créez un vecteur de *float* pour l'image de test (de taille *size_x*size_y*)
  - Ajoutez une double boucle for pour parcourir l'image (de respectivement nb_value et nb_sample itérations)
  - Dans la boucle, écrivez une fonction pour remplir le vecteur d'entrée à partir de la position dans la double boucle

    à noter : lecture d'un pixel sur une BufferedImage

    ```
    float pixel = (   (float)((image.getRGB(i, j)>> 16) & 0x000000FF)  )/255;
    ```

Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**

```
for (int x=0;x<nb_samples;x++){
    for (int y=0;y<nb_values;y++){

        // get digit image
        for (int i=0;i<size_x;i++){
            for (int j=0;j<size_y;j++){
                img[i+size_x*j]=
                        ((float)((image.getRGB(i+x*offset_x, j+y*offset_y)>> 16) & 0x000000FF))/255;
            }
        }
    }
}
```

Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**

- Nous avons le vecteur d'entrée, nous devons définir la valeur à obtenir
- Choisissez un nombre entre 0 à 9 (par exemple 3)
  - Si on a utilisé y comme itérateur de la boucle des nombres, on a pour résultat :
    - 1 si y == 3 (le nombre est bien celui choisi)
    - 0 si y != 3
- Ajoutez la variable res qui définit le résultat

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

Introduction à l'Intelligence Artificielle

```
public void paintComponent(Graphics g){

    g.setColor(Color.white);
    g.fillRect(0,0, this.getWidth(), this.getHeight());

    int val=0;
    for (int i=0;i<Main.size_x;i++){
        for (int j=0;j<Main.size_y;j++){
            val=(int)(main.img[i+Main.size_x*j]*255);
            g.setColor(new Color(val,val,val));
            g.fillRect(10+5*i, 10+5*j, 5, 5);
        }
    }

    for (int i=0;i<Main.size_x;i++){
        for (int j=0;j<Main.size_y;j++){
            val=(int)(main.neuron.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+5*i, 10+5*j, 5, 5);
        }
    }
```

Introduction à l'Intelligence Artificielle

- **Le système d'affichage**

- Dans Main, ajoutez dans la double boucle de test, ajoutez une temporisation et un rafraîchissement de l'affichage :

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

            display.repaint();

            try {Thread.sleep(500); // 500 ms
            } catch (InterruptedException e) {e.printStackTrace();}
        }
    }
```

- Puis lancez l'application pour vérifier que les valeurs d'entrée sont correctes

Introduction à l’Intelligence Artificielle

- **L’apprentissage (enfin!)**

- Dans la boucle d’apprentissage, après avoir défini l’image d’entrée et la sortie attendue, utilisez la fonction *compute* et la fonction *learn*
  - Vérifiez bien les paramètres

- Affichez à chaque test le delta et la sortie de votre neurone

Introduction à l'Intelligence Artificielle

- **L'apprentissage**

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

// process neuron
neuron.compute(img);
neuron.learn(img, res);

System.out.println("--- "+neuron.delta+" ; "+neuron.output);
```

Introduction à l’Intelligence Artificielle

- **L’apprentissage**

- On constate que le delta reste élevé pour le nombre choisi
  - Comme on a un learning rate faible, il faut faire l’apprentissage plusieurs fois
  - Chaque cycle est appelé ‘epoch’
- Ajoutez une boucle globale pour effectuer 50 fois l’apprentissage du jeu d’images

- On va aussi mesurer l’évolution du système :
  - Ajoutez une variable pour mesurer le delta moyen à chaque epoch
  - Affichez le delta moyen de chaque epoch

→ Pensez à baisser la tempo à 10 ms

Introduction à l'Intelligence Artificielle

- **L'apprentissage**

```
for (int epoch=0;epoch<50;epoch++){

    float sumdelta=0;

    for (int x=0;x<12;x++){
        for (int y=0;y<nb_values;y++){



            // add delta to the sum
            sumdelta+=Math.abs(neuron.delta);


            display.repaint();

            try {Thread.sleep(10);
            } catch (InterruptedException e) {e.printStackTrace();}
        }
    }

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

Introduction à l'Intelligence Artificielle

- **L'évaluation**

- L'évaluation consiste à tester notre réseau sur une base de données différentes de celles de test.
- On va diviser nos 16 échantillons en deux groupes :
  - 12 échantillons pour l'apprentissage
  - 4 échantillons pour l'évaluation
- Modifiez votre boucle d'échantillons pour apprendre sur les 12 premiers échantillons

```
for (int x=0;x<12;x++){
    for (int y=0;y<nb_values;y++){
```

- Après la boucle d'apprentissage, effectuez une nouvelle boucle sur les 4 derniers échantillons (de 12 à nb_samples)
  - Utilisez seulement la fonction compute
  - Affichez le nombre sur l'image testée et le résultat de votre neurone

    'Nombre : 3 ; prédiction : 0,988905'

Introduction à l'Intelligence Artificielle

- **L'évaluation**

```
int errors=0;
for (int x=12;x<nb_samples;x++){
    for (int y=0;y<nb_values;y++){

        // get digit image
        for (int i=0;i<size_x;i++){
            for (int j=0;j<size_y;j++){
                img[i+size_x*j]=
                        ((float)((image.getRGB(i+x*offset_x, j+y*offset_y)>> 16) & 0x000000FF))/255;
            }
        }

        // process neuron and display result
        float res=neuron.compute(img);

        if (res<0.001) res=0;
        System.out.println(y+" -> "+res);

        // count errors
        if ( (y==number && res<0.5) || (y!=number && res>0.5) ) errors++;
    }
}
```

- À noter : on peut aussi compter le nombre d'erreur

Introduction à l’Intelligence Artificielle

- **Bilan ?**

- **Neurone fonctionnel ?**
  - **Conservez bien vos fichier, on va s’en servir !**