

# Introduction à l'Intelligence Artificielle

- **L'environnement**

- Ouvrez Eclipse
- Créez un nouveau projet Java '1\_Neuron'
  - Clic droit dans package explorer → new → Java project
- 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() {
10
11         neuron=new Neuron();
12
13         display=new DisplayFrame(this);
14     }
15
16     public static void main(String[] args) {
17         new Main();
18     }
19 }
```

# 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 sigmoïde à part (note : **Math.exp( )** )
  - La fonction doit enregistrer le résultat

# Introduction à l'Intelligence Artificielle

- Le neurone

```
1
2 public class Neuron {
3
4     public float[] synaps;
5     public float bias;
6     public float output=0;
7     public float delta=0;
8
9     //////////////////////////////////////|
10    public Neuron(int size){
11        synaps=new float[size];
12    }
13
```

```
16
17 //////////////////////////////////////
18 public float compute(float[] img){
19
20     float sum=0;
21     for (int i=0;i<synaps.length;i++){
22         sum+=img[i]*synaps[i];
23     }
24     sum+=bias;
25
26     output=activation(sum);
27
28     return output;
29 }
30
31 //////////////////////////////////////
32 private float activation(float x){
33     return (float) (1 / (1+Math.exp(-x)));
34 }
```

# 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

# Introduction à l'Intelligence Artificielle

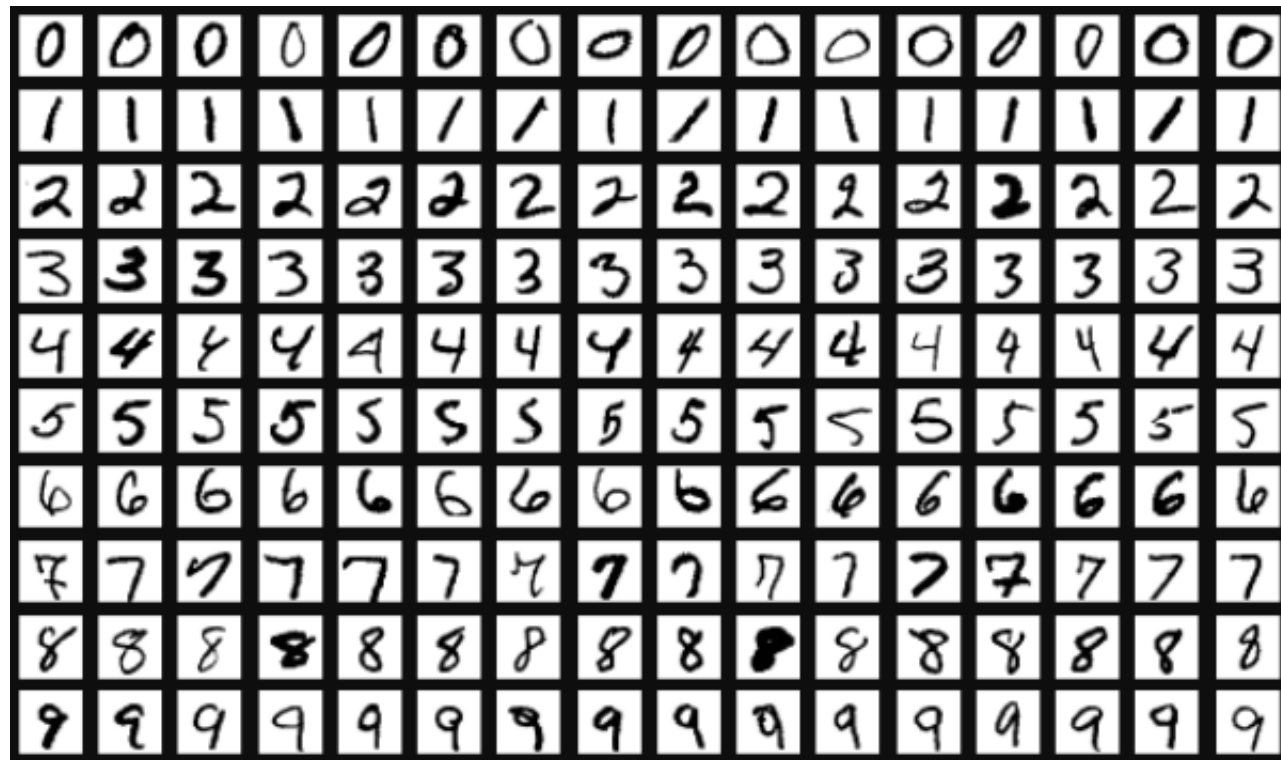
- Le neurone

```
2 public class Neuron {
3
4     public float learnRate=0.01f;
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
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
  - Images 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
```

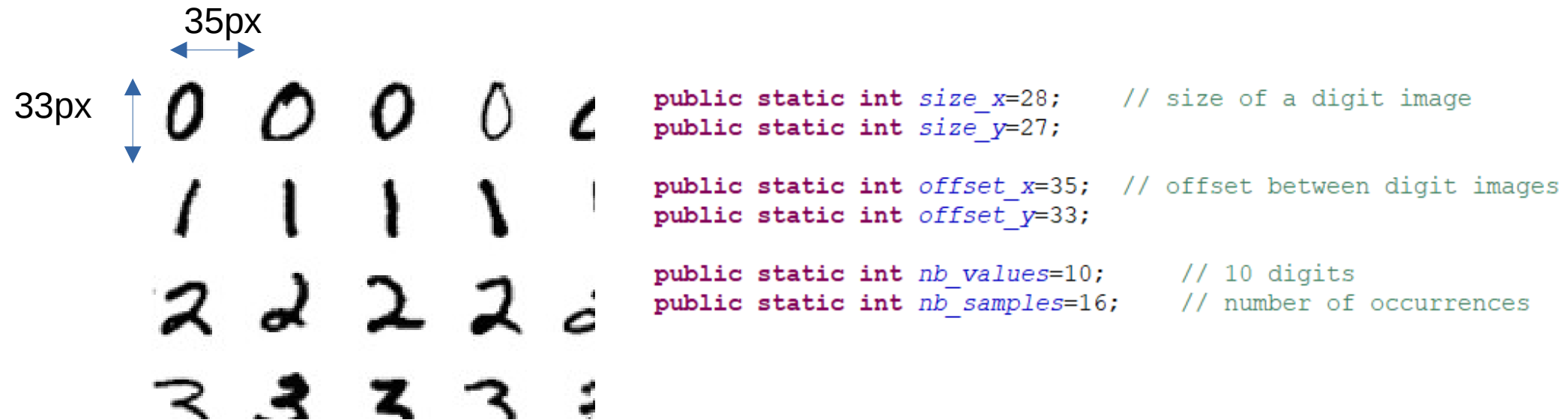


# Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**

- Dans la classe Main :

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



# Introduction à l'Intelligence Artificielle

- **Le système d'entraînement**

- Dans la classe Main :

- Créez un vecteur de *int* 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