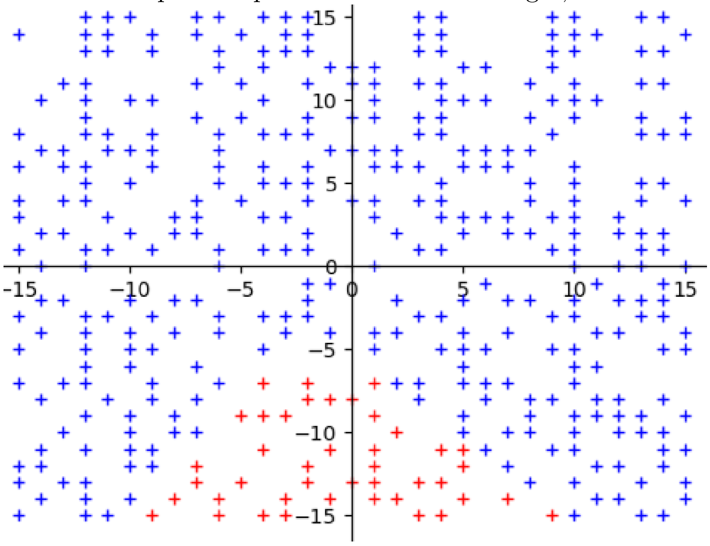
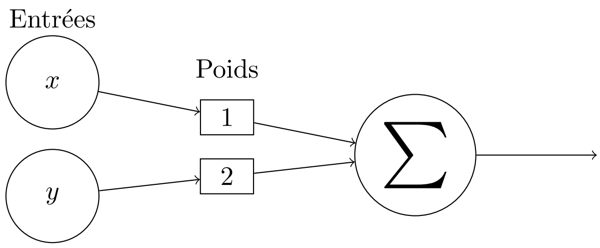
On considère environ 390 points dans un repère du plan : certains sont rouges, d'autres bleus.




On peut s'attendre à ce qu'une IA correctement entraînée à partir de ces points, soit capable de prévoir la couleur d'un nouveau point. Ainsi, un point de coordonnées $(-1; -12)$ ou un point de coordonnées $(5; -15)$ seront probablement rouges alors qu'un point de coordonnées $(-5; 14)$ sera certainement bleu. Comment une IA capable d'estimer la couleur d'un nouveau point de cette figure peut-elle fonctionner et comment l'entraîner ?

# 1 Le perceptron linéaire.

On implémente un neurone à deux entrées : $x$ et $y$, de poids synaptiques respectifs 1 et 2.




**Exercice 1 :**

1. Implémenter une fonction `f(x,y)` prenant deux arguments `x` et `y` et renvoyant $1 \times x + 2 \times y$ soit $x + 2y$.
2. Que renvoie `f(5,2)` ? `f(-5,2)` ? `f(-5,-2)` ?

On choisit comme fonction d'activation du neurone la fonction de Heaviside : $H(x) = \begin{cases} 1 \text{ si } x \geq 0, \\ 0 \text{ si } x < 0. \end{cases}$

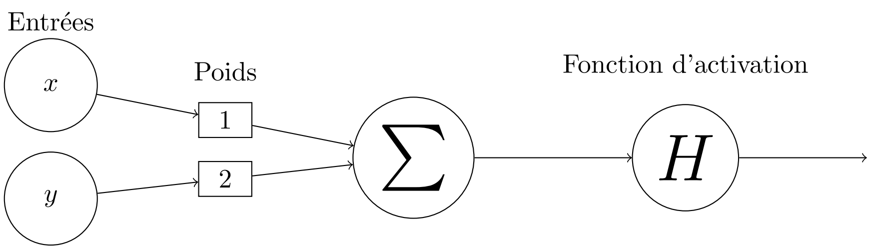
**Exercice 2 :**

1. Implémenter une fonction `Heaviside(x)` renvoyant 1 si `x` $\geq 0$ et 0 sinon.
2. Que renvoie `Heaviside(f(5,2))` ? `Heaviside(f(-5,2))` ? `Heaviside(f(-5,-2))` ?

Ainsi, la fonction `neurone(x,y)` ci dessous :

```
def neurone(x,y)
    return Heaviside(f(x,y))
```

permet d'obtenir le neurone :

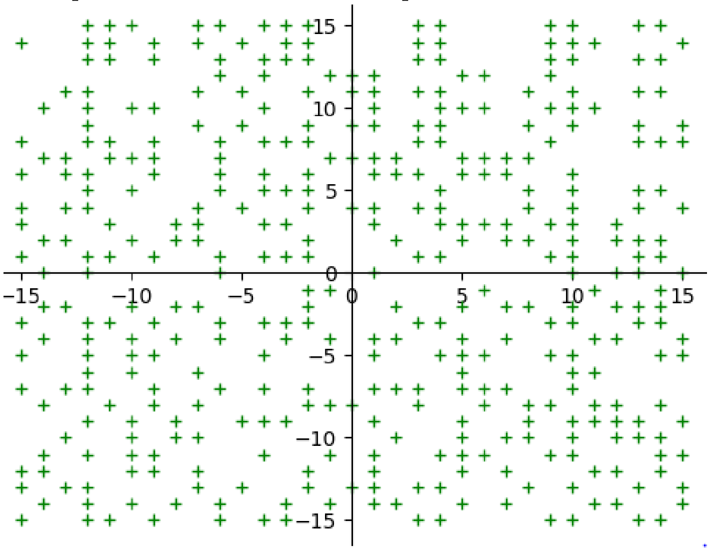


Avec ce neurone, à tout point du plan de coordoonnées $(x, y)$ on associe un nombre : 0 ou 1. On convient que si le nombre associé est 1 alors le point aura la couleur bleue, sinon il aura la couleur rouge.

## Exercice 3 :

1. Quelle est la couleur du point $A(5; 2)$ ?
2. Quelle est la couleur du point $B(-5; 2)$ ?
3. Quelle est la couleur du point $C(-5; -2)$ ?

On prend le même nuage de points qu'au début du TP : ils sont pour l'instant verts. On note $(E)$ cet ensemble de points.



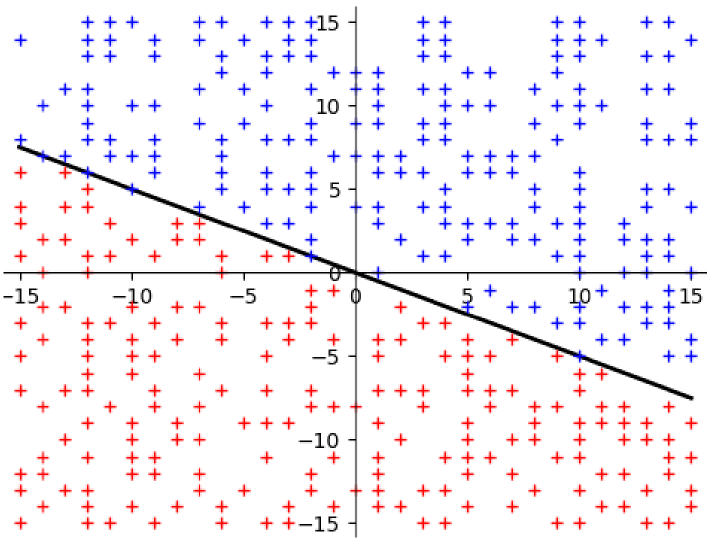

On choisit de colorer un point en bleu si le nombre associé par le neurone à ce point est 1, et de le colorer en rouge sinon. Ainsi $A(5; 2)$ sera bleu par exemple.

## Exercice 4 :

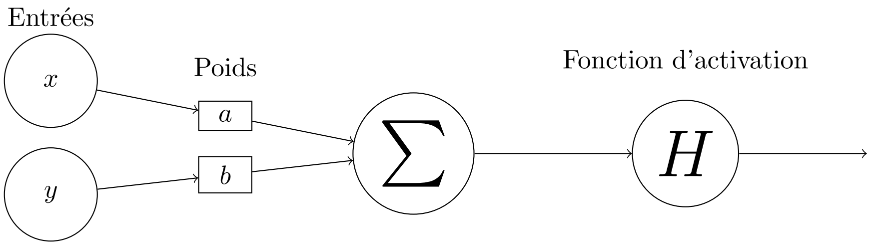
1. Ouvrir et compléter le fichier `neurone.py` afin que chaque point du plan soit donné en entrée du neurone implementé et que chaque point soit coloré correctement. La liste `x` contient toutes les abscisses des points et la liste `y` contient toutes les coordonées dans le même ordre. Ainsi `(x[10], y[10])` sont les coordonnées d'un point du plan.
2. Que constate-t-on ? Quelle est l'action du neurone sur l'ensemble $(E)$ des points donnés ?
3. Peut-on prévoir, à l'aide du neurone, la couleur d'un point de coordonnées $(20; -6)$ ?

Le neurone a partagé le nuage de points selon la droite d'équation $y = -\frac{1}{2}x$. En effet :

$$x + 2y = 0 \Longleftrightarrow y = -\frac{1}{2}x$$



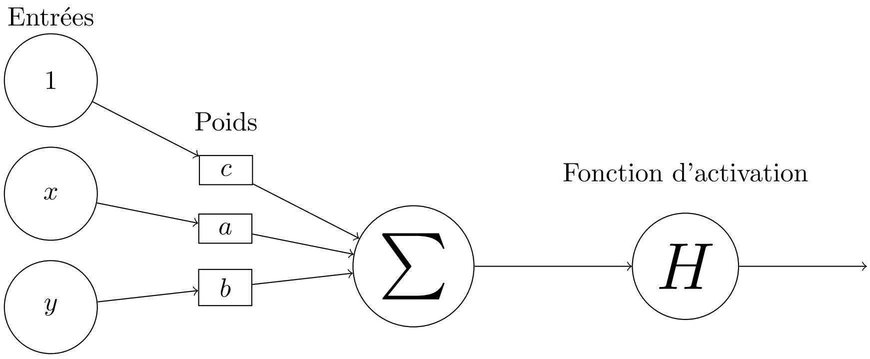
On souhaite désormais obtenir un neurone dont les poids synaptiques $a$ et $b$ peuvent être passés en argument :




## Exercice 5 :

1. Modifier la fonction `f(x,y)` en fonction `f(x,y,a,b)` prenant quatre arguments : `x`, `y` et leurs poids respectifs `a` et `b`, et renvoyant $a \times x + b \times y$.
2. Tester l'action du neurone sur le nuage de points $(E)$ pour différentes valeurs de `a` et `b`.

On ajoute désormais une troisième entrée au neurone, toujours égale à 1, appelée biais. Son poids synaptique sera noté c.




## Exercice 6 :

1. Implémenter une fonction `g(x,y,a,b,c)` prenant cinq arguments : `x`, `y` et leurs poids respectifs `a` et `b`, le poids `c` du biais 1, et renvoyant $ax + by + c$.
2. Implémenter la fonction `neurone_biais(x,y,a,b,c)`.
3. Tester l'action du neurone avec biais sur le nuage de points $(E)$ pour différentes valeurs de `a`, `b` et `c`.

## Remarque(s) :

Un perceptron linéaire permet de séparer un nuage de points en deux parties selon une droite.

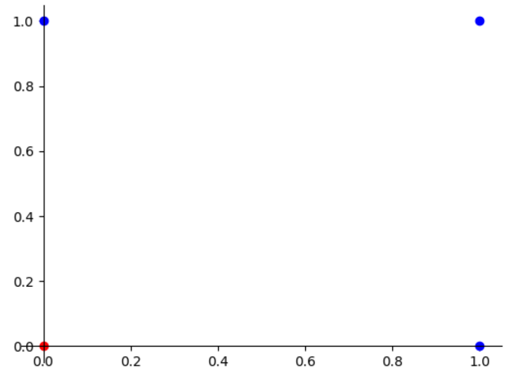
Il faut désormais considérer le problème inverse : soit un nuage de points rouges et bleus pouvant être séparés par une droite, comment retrouver les poids synaptiques `a`, `b` et `c` du neurone permettant d'effectuer cette séparation ? Si on y parvient, le neurone permettra de prévoir la couleur de tout nouveau point.

# 2 Quelques exemples logiques.

## 2.1 Le OU logique.

| x | y | x OU y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

1 correspond au bleu et 0 au rouge.



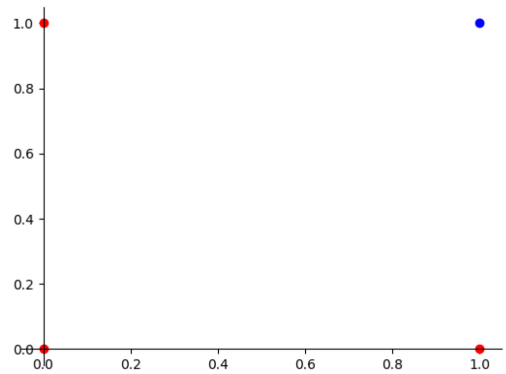

**Exercice 7 :**

1. Peut-on trouver une droite partageant le nuage de points en deux parties, l'une contenant les points bleus, l'autre contenant le point rouge ?
2. Déterminer les poids synaptiques `a`, `b` et `c` d'un neurone permettant ce partage.
3. Ouvrir le fichier `logique.py` et vérifier les résultats de la question précédente.

## 2.2 Le ET logique.

| x | y | x ET y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

1 correspond au bleu et 0 au rouge.



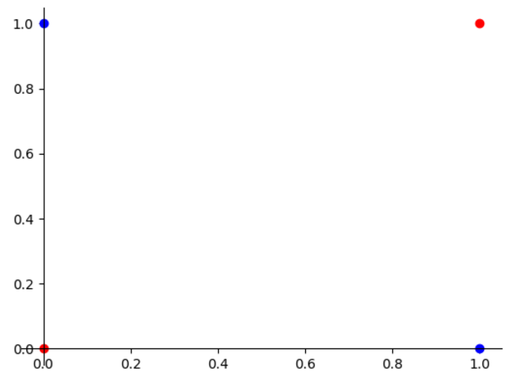

**Exercice 8 :**

1. Peut-on trouver une droite partageant le nuage de points en deux parties, l'une contenant le points bleu, l'autre contenant les point rouges ?
2. Déterminer les poids synaptiques `a`, `b` et `c` d'un neurone permettant ce partage.
3. Reprendre le fichier `logique.py` et vérifier les résultats de la question précédente.

### 2.3 Le OU exclusif logique (XOR).

| x | y | x XOR y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

1 correspond au bleu et 0 au rouge.




**Exercice 9 :**

1. Peut-on trouver une droite partageant le nuage de points en deux parties, l'une contenant les points bleus, l'autre contenant les points rouges ?
2. Ouvrir le fichier `XOR.py` et regarder comment on peut obtenir le XOR à l'aide d'un réseau de 3 neurones.
3. Dessiner ce réseau.

**Remarque(s) :**

- L'action d'un seul neurone est finalement assez limitée et ne permettra pas d'obtenir le nuage de points présenté en début de TP.
- Si un nuage de points est partageable par un perceptron linéaire, on dit qu'il est linéairement séparable.

## 3 Apprentissage par correction d'erreur.

On considère la donnée un nuage de points linéairement séparable. Peut-on automatiser la détermination des poids synaptiques `a`, `b` et `c` d'un neurone permettant ce partage ?

Soit $(E)$ une ensemble de points de coordonnées $(x;y)$ dont chacun a une couleur dans l'ensemble $\{0;1\}$, telle que 1 correspond au bleu et 0 au rouge.

On initialise les poids synaptiques `a`, `b` et `c` du neurone à 0 et on donne l'algorithme suivant :

---

**Tant que** le neurone ne renvoie pas la bonne couleur de chaque point de $(E)$
Choisir un point de $(E)$ de coordoonées (`x, y`) et de couleur `couleur`.
`p` ⟵ `neurone(x, y, a, b, c)`
`a` ⟵ `a + (couleur - p)` × `x`
`b` ⟵ `b + (couleur - p)` × `y`
`c` ⟵ `c + (couleur - p)`
**Fin Tant que**

---

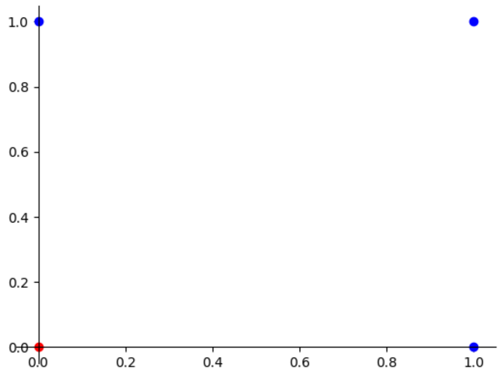
**Remarque(s) :**

Attention : cet algorithme n'est pas fini si l'ensemble $(E)$ n'est pas linéairement séparable.

### 3.1 Application au OU logique.

| x | y | couleur |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

1 correspond au bleu et 0 au rouge.



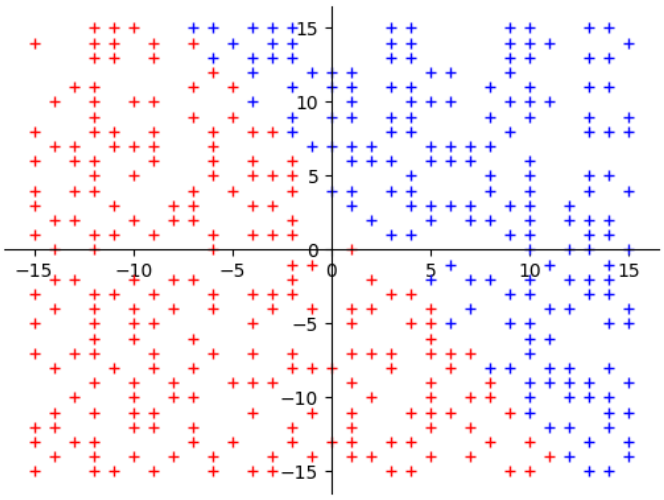

**Exercice 10 :**

Recopier et compléter le tableau suivant (il y a plus de 10 étapes !).

| Etape | a | b | c | x | y | couleur | p | couleur = p ? | a | b | c |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | | | | | | | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | OUI | - | - | - |
| 2 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | NON | +0 | +1 | +1 |
| 3 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | OUI | - | - | - |
| 4 | 0 | 1 | 1 | 1 | 1 | 1 | | | | | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

### 3.2 Application à un nuage de 390 points.

On donne le nuage de point suivant, qui semble linéairement séparable :




Chaque point est caractérisé par ses coordonnées $(x; y)$ et par un nombre de l'ensemble $\{0; 1\}$ définissant sa couleur.

Ouvrir le fichier `apprentissage.py`. La liste `x` contient toutes les abscisses des points, la liste `y` contient toutes les coordonées dans le même ordre et la liste `couleur` contient toutes les couleurs dans le même ordre. Ainsi `(x[10], y[10])` sont les coordonnées d'un point du plan et `couleur[10]` est sa couleur.

On a implementé l'algorithme précédent. La variable `score` donne le nombre de points, sur les 390, pour lesquels le neurone renvoie la bonne couleur avec les poids synaptiques calculés à chaque étape.

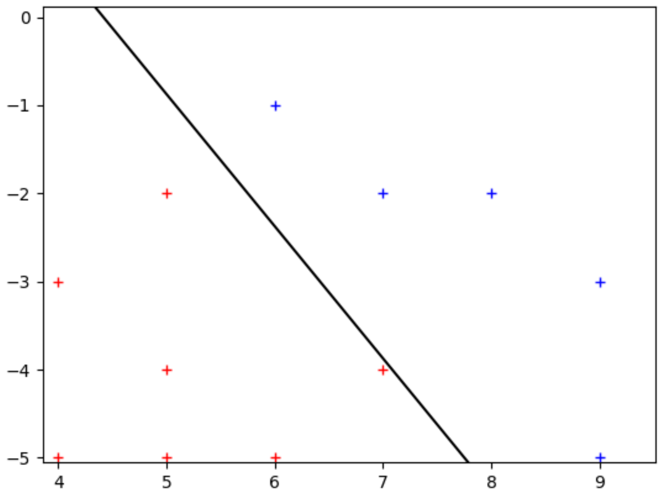
En exécutant le code, on constate qu'en 11 étapes, on parvient au résultat recherché et **on visualise l'évolution de la droite de séparation**.

```
Etape : 1      a= 28 b= 14 c= -15      score= 340
Etape : 2      a= 27 b= 11 c= -27      score= 343
Etape : 3      a= 33 b= 21 c= -35      score= 343
Etape : 4      a= 36 b= 18 c= -47      score= 346
Etape : 5      a= 32 b= 15 c= -58      score= 353
Etape : 6      a= 32 b= 17 c= -66      score= 356
Etape : 7      a= 26 b= 14 c= -76      score= 368
Etape : 8      a= 32 b= 21 c= -84      score= 364
Etape : 9      a= 24 b= 15 c= -92      score= 384
Etape : 10     a= 33 b= 22 c= -98      score= 369
Etape : 11     a= 24 b= 16 c= -106     score= 390
```

### 3.3 Limites.

#### Exercice 11 :

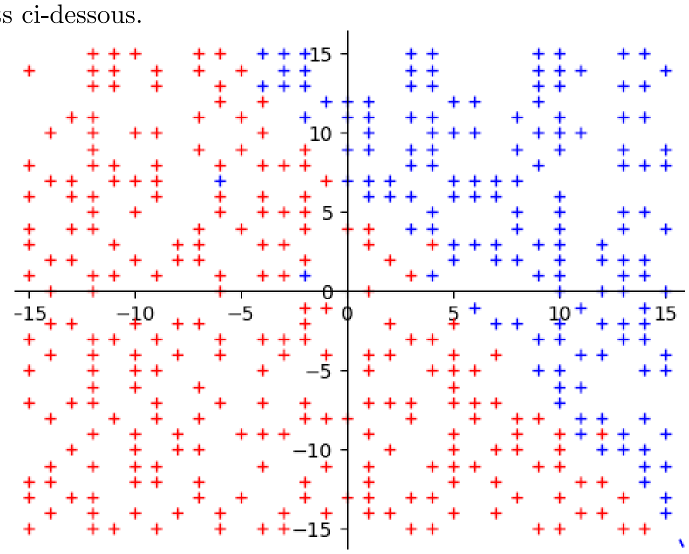
On donne le nuage de points ci-dessous. On a trouvé les poids synaptiques permettant à un neurone de séparer les points en deux sous-ensembles de points de même couleur. La droite de séparation est tracée.




1. On ajoute un nouveau point rouge de coordonnées $(6; -2)$. La solution proposée reste-t-elle correcte ?
2. L'ensemble de points est-il toujours linéairement séparable ?

#### Exercice 12 :

On donne le nuage de points ci-dessous.




Est-il linéairement séparable ?