Présentation débranchée d'une IA et mathématiques d'un neurone artificiel

1. **Jeu de Nim**
   Un apprentissage par renforcement
2. **Le perceptron**
   Un apprentissage supervisé
3. **Un premier réseau de neurones**
   Problème du XOR

Mickaël Barraud, pour l'IREM de Nantes, groupe EIEM

Enseignement Informatique Enseignement I.R.E.Mathématiques

Mathématiques et en NSI
Lycée De Lattre de Tassigny
La Roche-sur-Yon

Formateur MEEF 2D parcours NSI
INSPÉ de Nantes

Ada Lovelace en **1843** traduit un article sur la machine analytique de Babbage et le complète de commentaires (notes) qui formalisent ses idées.

- Dans sa note A, elle entrevoit que de telles machines pourraient même composer de la musique.
- Dans sa Note G elle affirme cependant que, malgré ses pouvoirs impressionnants, on ne peut pas vraiment dire qu'elle « pense » ; ce que Alan Turing appellera plus tard « l'objection de Lady Lovelace ».

L'article fondateur est celui de **Alan Turing** en **1950** *Computing machinery and intelligence*. Il y propose de considérer la question suivante :
« Est-ce que les machines peuvent penser ? ».

- « Au lieu de produire un programme qui simule l'esprit d'un adulte, pourquoi ne pas plutôt essayer d'en produire un qui simule celui de l'enfant ? S'il était alors soumis à une éducation appropriée, on aboutirait au cerveau humain. » [...]
- « On peut aussi soutenir qu'il vaut mieux équiper la machine avec les meilleurs organes sensoriels que l'on puisse acheter, puis lui apprendre à comprendre et à parler anglais. Ce processus pourrait se conformer à l'enseignement normal d'un enfant. On lui montrerait et nommerait des objets, etc. »

John McCarthy en **1956** propose l'expression « intelligence artificielle » ce qui fait date de ce domaine de recherche.

1. Jeu de Nim

- Un jeu de Nim se joue à 2 joueurs.
- Il nécessite un (ou plusieurs) tas d'objets (jetons, billes, allumettes…).
- Chaque joueur, à son tour, peut enlever un ou plusieurs objets d'un tas.
- Un joueur gagne s'il parvient à retirer le dernier objet

Les variantes consiste à choisir

- le nombre de tas et d'objets par tas, ← 1 tas 7 feutres
- le nombre maximum d'objet pouvant être retirés en une fois, ← max 2
- si le joueur retirant le dernier objet gagne ou perd ← dernier perd.

a) Exemple de partie

- 7 objets dans un tas
- prise de 1 ou 2
- dernier perd

← Le joueur 1 prend 1 objet, il en reste 6

← Le joueur 2 prend 2 objets, il en reste 4

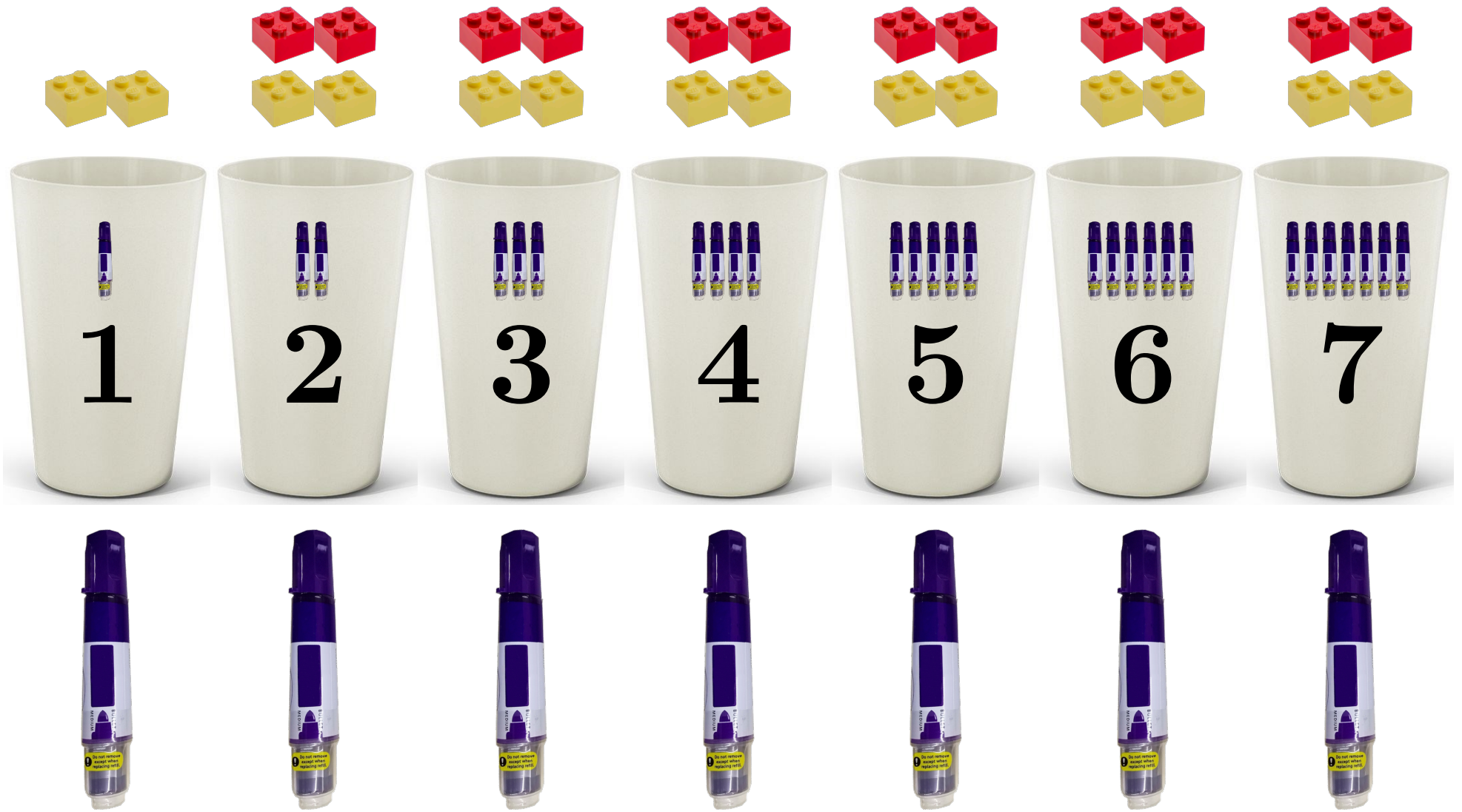
← Le joueur 1 prend 1 objet, il en reste 3

← Le joueur 2 prend 1 objet, il en reste 2

← Le joueur 1 prend 1 objet

Le joueur 2 perd en prenant le dernier objet.

b) Mise en place de l'I.A.



Par tirage au sort, dans le gobelet correspondant à la situation (nombre d'objets):

= L'IA choisit d'enlever un objet = L'IA choisit d'enlever deux objets

c) Apprentissage par renforcement

Si l'IA perd, par exemple : J-1, IA-2, J-1, IA-1, J-1, IA-1

Les  tirés ne sont pas remis dans les gobelets et on recommence avec ceux qui restent. (Si un gobelet est vide, on le ré-initialise).

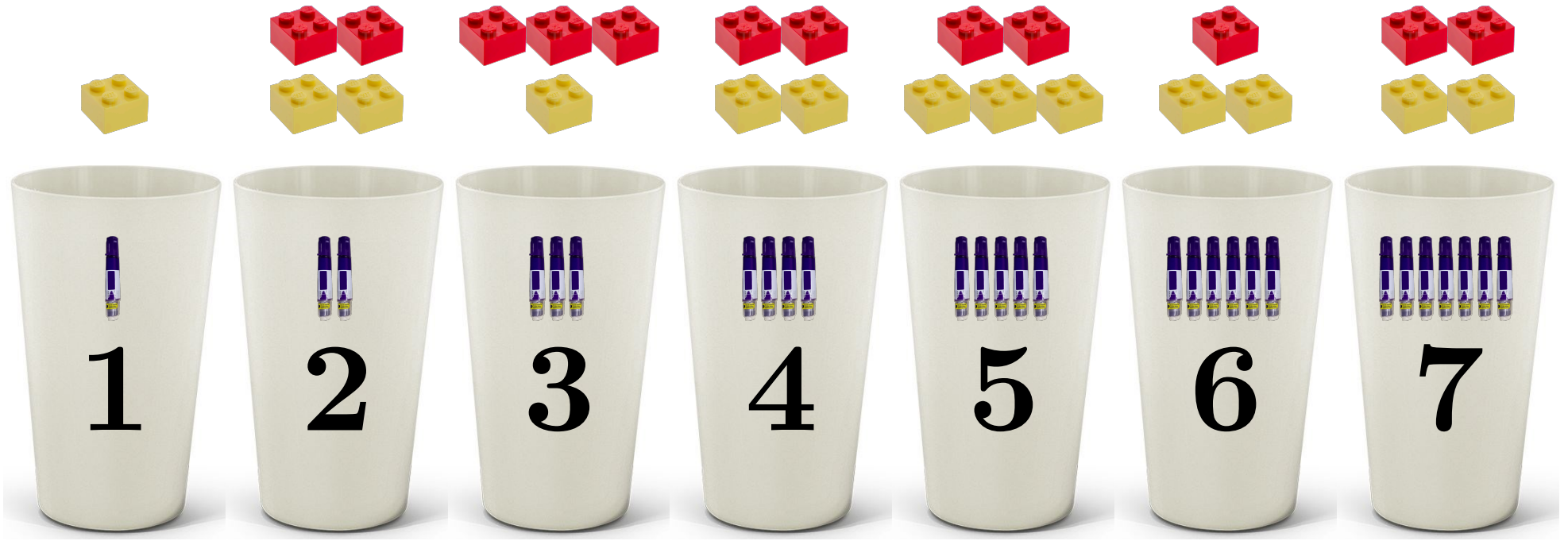
Ainsi, aux prochaines parties, l'IA aura moins de chance de refaire ces coups qui l'ont fait perdre.

Si, ensuite, l'IA gagne, par exemple : J-2, IA-1, J-1, IA-2, J-1

Les   tirés sont remis en double dans les gobelets et on recommence.

Ainsi, aux prochaines parties, l'IA aura plus de chances de refaire ces coups qui l'ont fait gagner.

Bilan des deux parties :



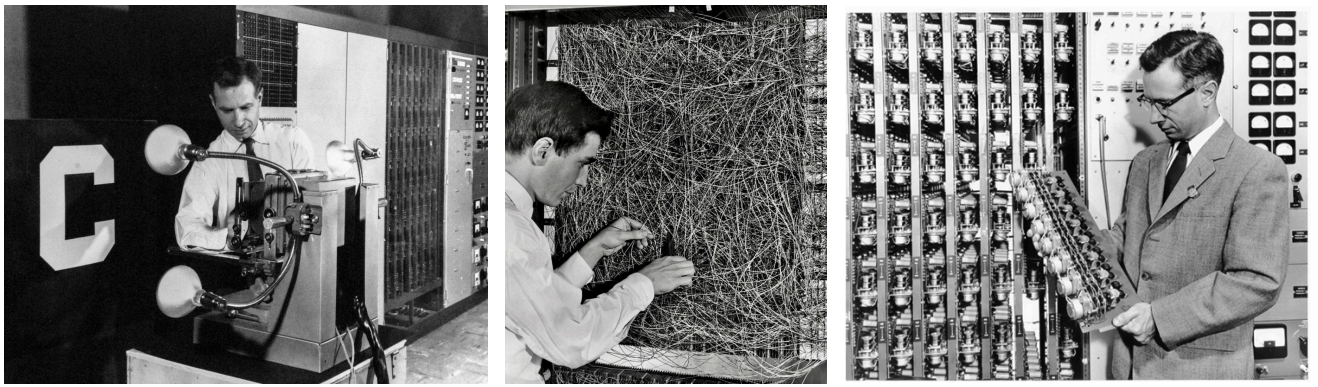
L'IA a maintenant 75% de chance de gagner dans la situation a trois objets.

Pour aller plus loin : https://projet.liris.cnrs.fr/~mam/machine/

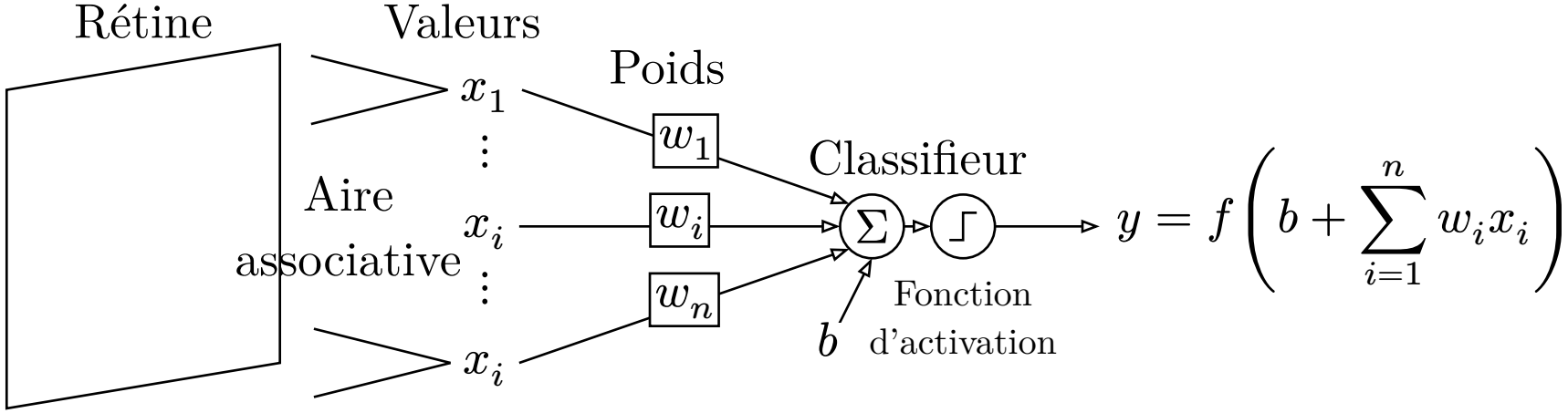
L'apprentissage par renforcement est l'un des principes de base que l'on retrouve dans de nombreux systèmes d'intelligence artificielle. Ces systèmes contiennent beaucoup de paramètres que l'on peut ajuster automatiquement grâce à l'expérience.

Dans un modèle de langages large (LLM) par exemple il y en a de 10^9 à 10^{12}.

2. Le perceptron (1957)

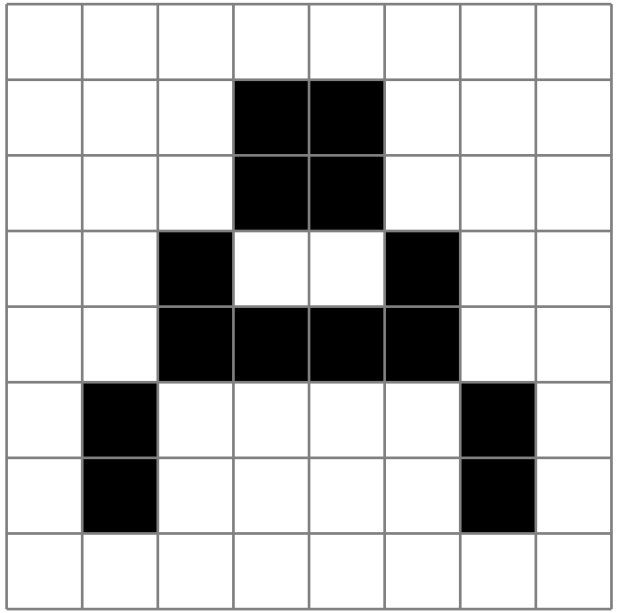


La machine est constituée d'une rétine pour percevoir une image simple, d'une aire associative fournissant les valeurs d'entrée, le tout relié à un classifieur, qui fait la somme de ces valeurs après les avoir multipliées par des poids (potentiomètres motorisés) et la soumet à une fonction d'activation, donnant, en sortie, la classe de l'image perçue.

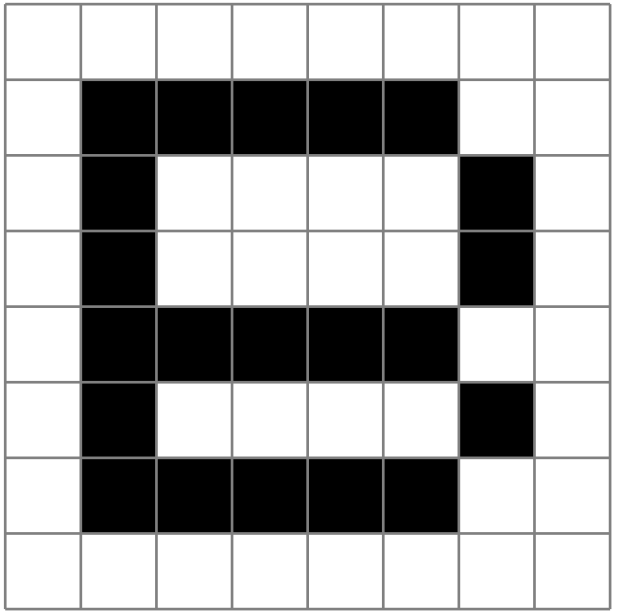


a) Apprentissage supervisé

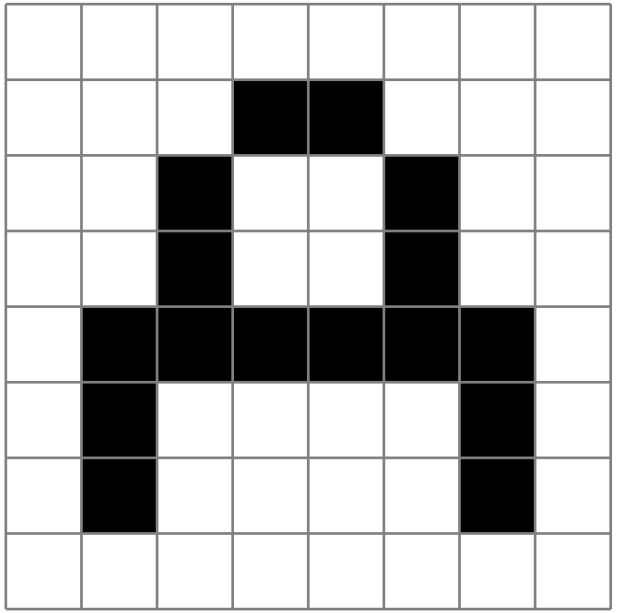
On suppose disposer d'un jeu de test. C'est à dire ici d'images représentant des A et des B avec une étiquette indiquant qu'il s'agit effectivement d'un A ou d'un B :



Étiquette : A



Étiquette : B



Étiquette : A

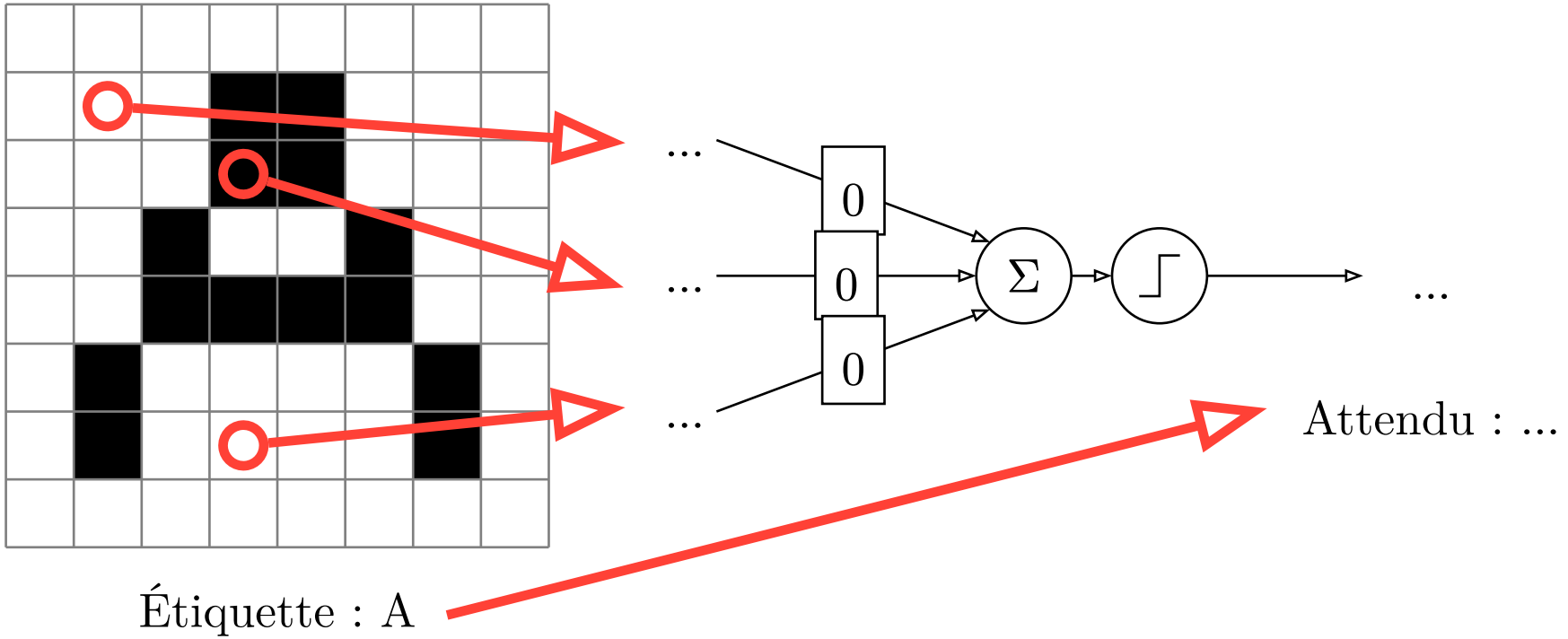
Comme l'apprentissage est numérique, il faut numériser les informations :

- Les entrées :
  - 1 pour noir
  - -1 pour blanc
- Les sorties :
  - 1 pour A
  - -1 pour B

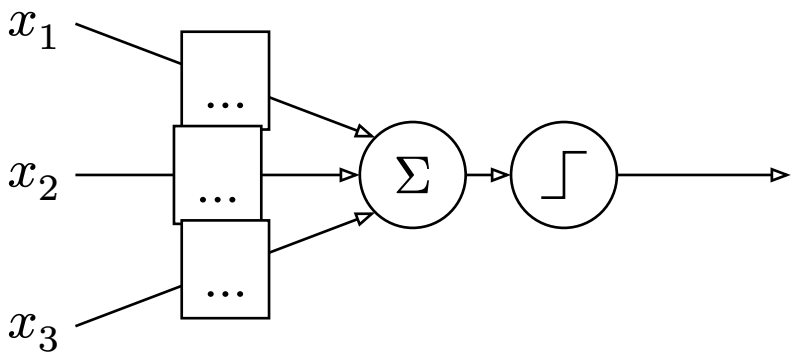
L'apprentissage consiste à modifier automatiquement les poids selon la réponse.

Apprentissage : Présentation d'un A

À cette étape, on peut mettre des poids aléatoires... ici tout à 0 (Fonction d'activation : signe(x) = 1 si $x > 0$, sinon 0)

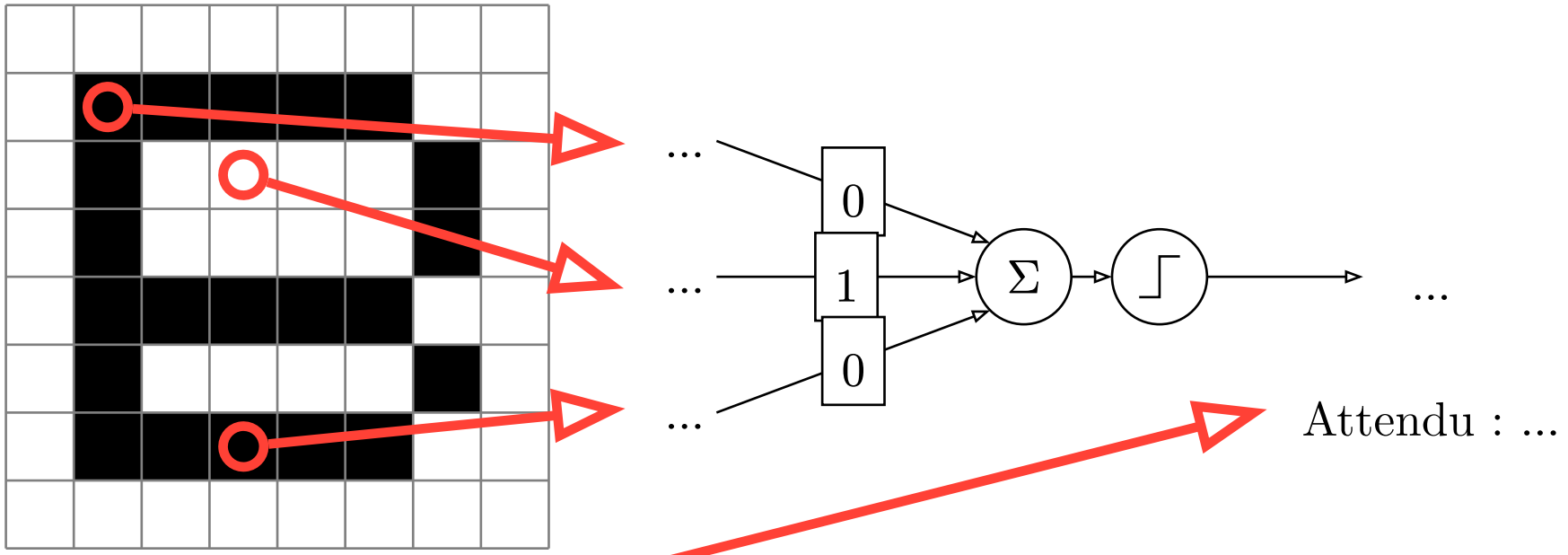


Donne comme nouveau modèle :



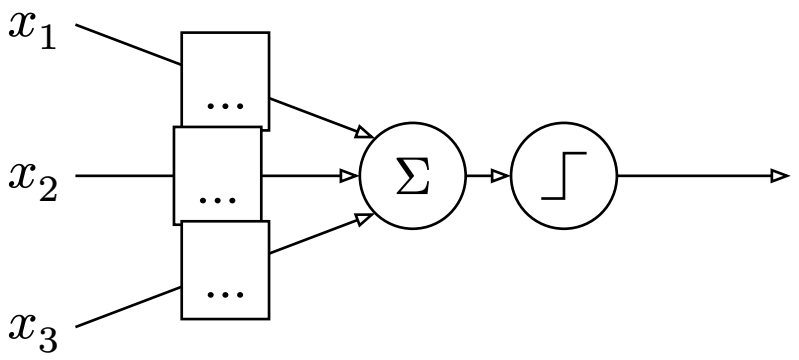
Apprentissage : Présentation d'un B

A chaque étape on peut affiner les poids



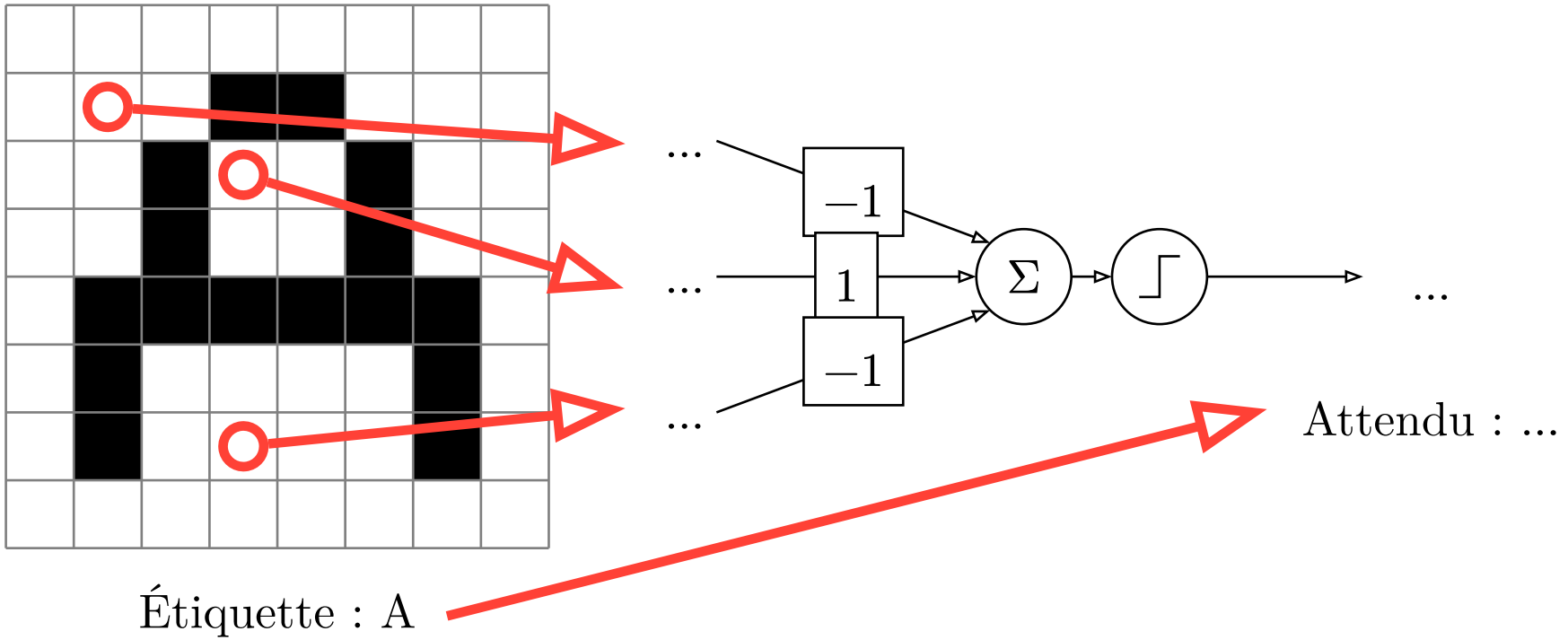
Étiquette : B

Donne comme nouveau modèle :

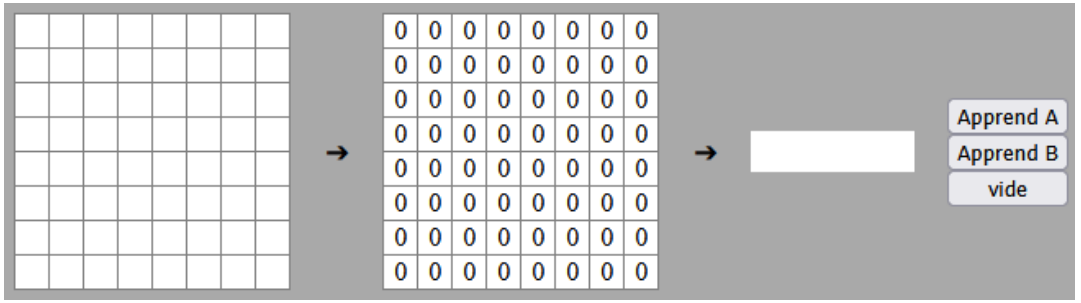


Test du modèle : Présentation d'un A

On a conservé une partie des données pour tester le modèle



Pour obtenir un meilleurs modèle, il faut augmenter le nombre d'entrées, donc le nombre de paramètres et la taille des données étiquetées...

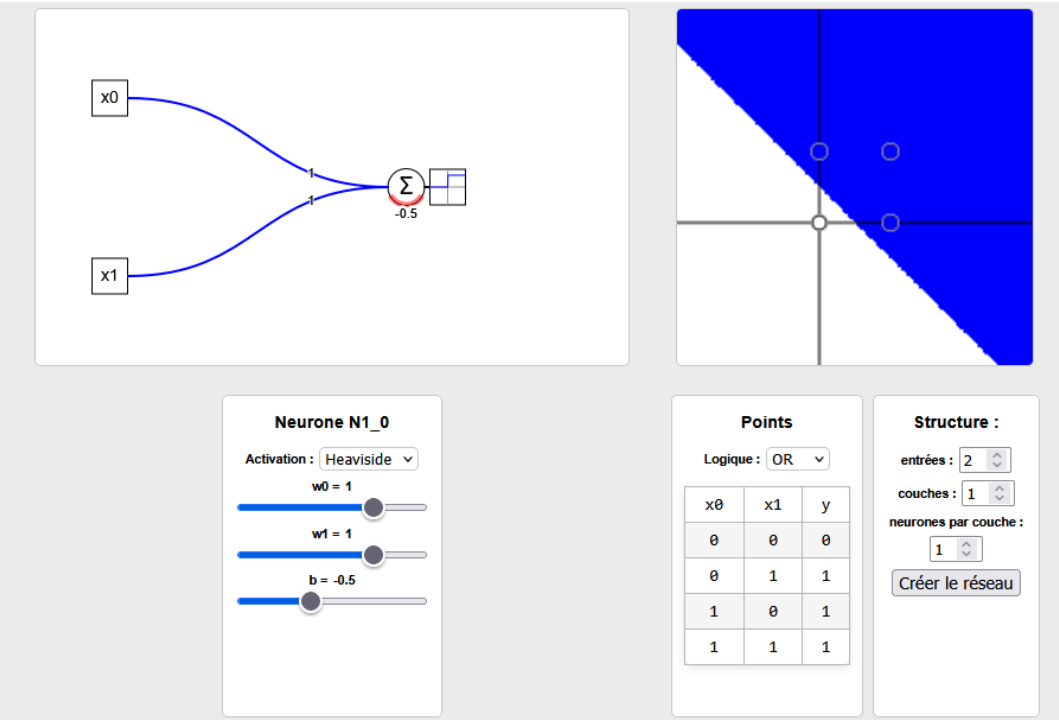


3. Un premier réseau de neurones

Question : Est-ce qu'un perceptron suffit ?

Idée : Testons les portes logiques, constitutives des processeurs de nos ordinateurs :

https://nsi-delattre.fr/outils?r=neurones

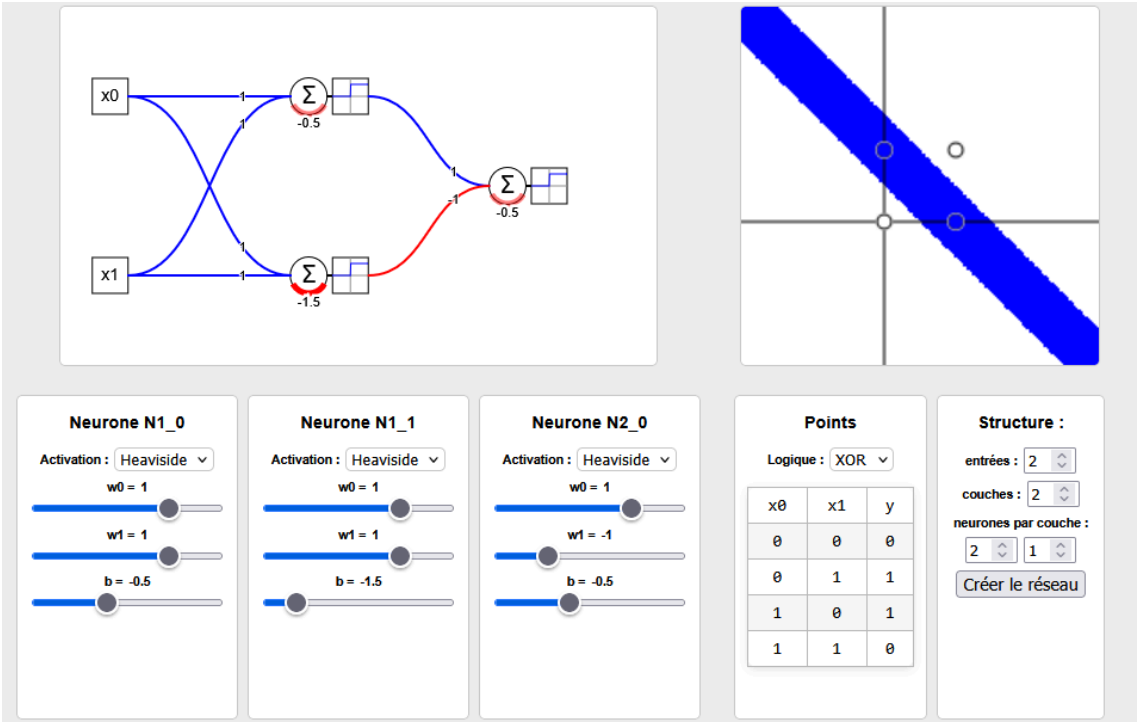


Un perceptron (un neurone seul) ne permet de séparer l'espace des entrés que par un hyperplan. (Le plan par la droite $w_0x_0 + w_1x_1 + b = 0$)

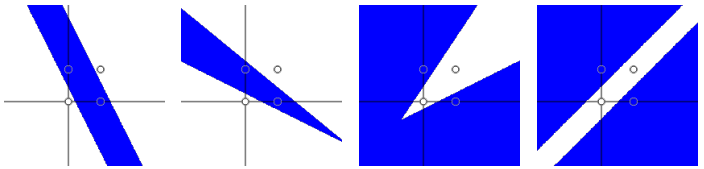
Cela suffit pour la porte AND et la porte OR, mais pas pour la porte XOR

Une solution ?

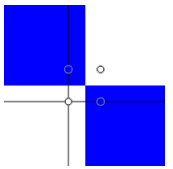
Ajouter des neurones en couches :



Il n'y a bien sûr pas unicité des coefficients possibles :



Ni de l'architecture :



Pour aller plus loin : https://playground.tensorflow.org

4. Conclusion

Pour un LLM le principe de base est aussi un réseau de neurones multicouche, avec de (nombreuses) améliorations.

Les deux apprentissages abordées sont valables, et on peut préciser :

1. Pré-entraînement (Pretraining)
   - Le modèle lit énormément de textes.
   - Apprentissage auto-supervisé de prédiction du prochain token.
2. Ajustement (Fine-Tuning) à une tâche spécifique
   - On lui montre des exemples de questions et de bonnes réponses.
   - Apprentissage supervisé.
3. Amélioration par retour humain
   - On évalue les réponses du modèle.
   - Apprentissage par renforcement.

Au final, le modèle ne fait que des prédictions sur des distributions statistiques. Les variations dans ses réponses sont liées aux choix probabilistes faits pour chaque token.