

Introduction à l'Intelligence Artificielle

INF103

Romain COUILLET, Valentin GIRARD (IMA-04), Achille BAUCHER
(IMA-05), Alexandre DONZE (IMA-09)

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques

Le cours

Organisation générale:

- 6 cours (1.5h chacun), à partir de S39
- 6 TPs (3h) en fil rouge, vu comme un “projet découverte de l’IA”: (à partir de S41)

Le cours

Organisation générale:

- 6 cours (1.5h chacun), à partir de S39
- 6 TPs (3h) en fil rouge, vu comme un “projet découverte de l’IA”: (à partir de S41)

Outils:

- implémentations en Python, apprentissage des packages `sklearn`, `skimage`, `matplotlib`
- plateforme VSCode et/ou Jupyter-Notebook
- rapports en `markdown`, `Jupyter-Notebook` (ou `LateX` pour les ambitieux!)

Le cours

Organisation générale:

- 6 cours (1.5h chacun), à partir de S39
- 6 TPs (3h) en fil rouge, vu comme un “projet découverte de l’IA”: (à partir de S41)

Outils:

- implémentations en Python, apprentissage des packages `sklearn`, `skimage`, `matplotlib`
- plateforme VSCode et/ou Jupyter-Notebook
- rapports en `markdown`, `Jupyter-Notebook` (ou `LateX` pour les ambitieux!)

Communication:

- tout se passe sur Caseine:
<https://moodle.caseine.org/course/view.php?id=803>
- **inscrivez vous au plus vite!** (chercher INF103 sur Caseine, puis “M’inscrire”)

Le cours

Modalités d'évaluation:

- **CC1:** compte-rendu de projet à mi-parcours (TP3)
- **CC2:** compte-rendu final du projet (TP6)
- **Exam:** QCM, questions ouvertes et questions liées au projet.

Le cours

Modalités d'évaluation:

- **CC1:** compte-rendu de projet à mi-parcours (TP3)
- **CC2:** compte-rendu final du projet (TP6)
- **Exam:** QCM, questions ouvertes et questions liées au projet.

Critères d'évaluation:

- auto-évaluation en lien avec les facilitateurs
- valorisation du travail fourni par rapport à votre propre**contrat**.

Le cours

Les TP sont effectués à distance, soit (i) sur Jupyterhub (le plus simple), soit (ii) sur le serveur turing en utilisant l'interface VSCode et le plugin Jupyter.

Logistique pour les TP via Jupyterhub:

- se connecter à <https://jupyterhub.univ-grenoble-alpes.fr/>
- la première fois, créer un projet par l'onglet Nouveau et Python 3
- il est possible qu'il vous soit nécessaire d'installer des paquets supplémentaires en local, par exemple comme suit:
`!python -m pip install -U scikit-image`

Logistique pour les TP via VSCode:

- en salle de TP, VSCode déjà installé
- si vous utilisez vos ordis personnels: installer VSCode et (si sous Windows) WSL

Le cours

Première exécution de VSCode:

- lancez Visual Studio Code
- clic en bas à gauche (symbole \geq vert) et choisir “Connect to Host”
- “Add new Host”
- ssh -X [your_agalan_name]@turing.e.ujf-grenoble.fr
- Password.

puis...

- dans “Extensions” (5e icône), installez Python et Jupyter
- dans “Explorer” (1er icône), créer un dossier INF103
- dans INF103, créer un fichier projet.ipynb et l'ouvrir
- **vous êtes prêt!**

Le cours

Première exécution de VSCode:

- lancez Visual Studio Code
- clic en bas à gauche (symbole \geq vert) et choisir “Connect to Host”
- “Add new Host”
- ssh -X [your_agalan_name]@turing.e.ujf-grenoble.fr
- Password.

puis...

- dans “Extensions” (5e icône), installez Python et Jupyter
- dans “Explorer” (1er icône), créer un dossier INF103
- dans INF103, créer un fichier projet.ipynb et l'ouvrir
- vous êtes prêt!

À la fin de chaque TP:

- exportez votre travail en HTML [Export → HTML]
- récupérer le grâce à l'outil “FTP turing” (icône du bureau) et postez le sur Caseine

1 Logistique du cours

2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases

- Bases historiques: de l'homme à la machine
- Bases scientifiques: de l'IA à l'apprentissage machine
- Acte 1: Les réseaux de neurones: de la fiction à la réalité?
- Acte 2: Le retour en force des mathématiques
- Acte 3: La révolution des réseaux profonds
- Aujourd'hui: où cela nous mène-t-il?

3 Des données aux vecteurs: séparabilité, représentations et kNN

4 Les réseaux de neurones

5 L'IA moderne: révolution informatique et retour des mathématiques

1 Logistique du cours

2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases

- **Bases historiques:** de l'homme à la machine
- **Bases scientifiques:** de l'IA à l'apprentissage machine
- Acte 1: Les réseaux de neurones: de la fiction à la réalité?
- Acte 2: Le retour en force des mathématiques
- Acte 3: La révolution des réseaux profonds
- Aujourd'hui: où cela nous mène-t-il?

3 Des données aux vecteurs: séparabilité, représentations et kNN

4 Les réseaux de neurones

5 L'IA moderne: révolution informatique et retour des mathématiques

Fondements historiques: de la fiction à la science

L'“extension” historique d'Homo Sapiens

- -300 000 à -40 000: Homo Sapiens tire ses produits de sa propre énergie, de la nature (alimentation), et de collaborations (sécurité, chasse).

Fondements historiques: de la fiction à la science

L'“extension” historique d'Homo Sapiens

- -300 000 à -40 000: Homo Sapiens tire ses produits de sa propre énergie, de la nature (alimentation), et de collaborations (sécurité, chasse).
- -40 000 à 1800: agriculture et force animale, l'âge des civilisations: Homo Sapiens gagne des “nouveaux muscles” (cultures, armées).

Fondements historiques: de la fiction à la science

L'“extension” historique d'Homo Sapiens

- -300 000 à -40 000: Homo Sapiens tire ses produits de sa propre énergie, de la nature (alimentation), et de collaborations (sécurité, chasse).
- -40 000 à 1800: agriculture et force animale, l'âge des civilisations: Homo Sapiens gagne des “nouveaux muscles” (cultures, armées).
- 1800 à 1950: révolution industrielle, Homo Sapiens tire une énergie gigantesque des ressources de carbone (charbon, pétrole), équivalentes à des centaines de bras humains...
au point de bouleverser l'équilibre planétaire!



Fondements historiques: de la fiction à la science

Les années 1950: la révolution informatique

- Le transistor étend **le cerveau**: la machine “pense” plus vite que 1 000 Homo Sapiens.

Fondements historiques: de la fiction à la science

Les années 1950: la révolution informatique

- Le transistor étend **le cerveau**: la machine “pense” plus vite que 1 000 Homo Sapiens.
- Le mythe des robots se développe!

Fondements historiques: de la fiction à la science

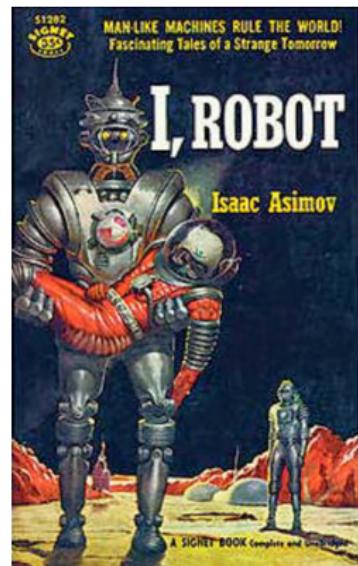
Les années 1950: la révolution informatique

- Le transistor étend **le cerveau**: la machine “pense” plus vite que 1 000 Homo Sapiens.
- Le mythe des robots se développe!
- . . . plus vite. . . **mais moins bien**: la recherche en **intelligence artificielle** émerge.

Fondements historiques: de la fiction à la science

Les années 1950: la révolution informatique

- Le transistor étend **le cerveau**: la machine “pense” plus vite que 1 000 Homo Sapiens.
- Le mythe des robots se développe!
- ... plus vite... **mais moins bien**: la recherche en **intelligence artificielle** émerge.
- mais progresse très très lentement! **Domaine scientifique très difficile**.



Isaac Asimov:
I, Robot. (1950)

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

- pas de définition claire, mais en gros, c'est:

“l'ensemble des algorithmes qui, à partir de stimuli, (i) prennent des décisions et (ii) s'améliorent en apprenant seuls.”

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

- pas de définition claire, mais en gros, c'est:
“l'ensemble des algorithmes qui, à partir de stimuli, (i) prennent des décisions et (ii) s'améliorent en apprenant seuls.”
- des exemples concrets:
 - ▶ détection et classification d'objets (images, vidéos), de sons (flux audio), d'odeurs (capteurs chimiques): on mime ici les sens animaux

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

- pas de définition claire, mais en gros, c'est:
“l'ensemble des algorithmes qui, à partir de stimuli, (i) prennent des décisions et (ii) s'améliorent en apprenant seuls.”
- des exemples concrets:
 - ▶ détection et classification d'objets (images, vidéos), de sons (flux audio), d'odeurs (capteurs chimiques): on mime ici les sens animaux
 - ▶ apprentissage de jeux, de compétitions par entraînements (ex: échecs, Go): ce qu'on appelle **apprentissage par renforcement**

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

- pas de définition claire, mais en gros, c'est:
“l'ensemble des algorithmes qui, à partir de stimuli, (i) prennent des décisions et (ii) s'améliorent en apprenant seuls.”
- des exemples concrets:
 - ▶ détection et classification d'objets (images, vidéos), de sons (flux audio), d'odeurs (capteurs chimiques): on mime ici les sens animaux
 - ▶ apprentissage de jeux, de compétitions par entraînements (ex: échecs, Go): ce qu'on appelle **apprentissage par renforcement**
 - ▶ génération d'images, sons réalistes (le côté “artistique” de l'IA)

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

- pas de définition claire, mais en gros, c'est:
“l'ensemble des algorithmes qui, à partir de stimuli, (i) prennent des décisions et (ii) s'améliorent en apprenant seuls.”
- des exemples concrets:
 - ▶ détection et classification d'objets (images, vidéos), de sons (flux audio), d'odeurs (capteurs chimiques): on mime ici les sens animaux
 - ▶ apprentissage de jeux, de compétitions par entraînements (ex: échecs, Go): ce qu'on appelle **apprentissage par renforcement**
 - ▶ génération d'images, sons réalistes (le côté “artistique” de l'IA)

IA et “apprentissage automatique”: (machine learning)

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

- pas de définition claire, mais en gros, c'est:
“l'ensemble des algorithmes qui, à partir de stimuli, (i) prennent des décisions et (ii) s'améliorent en apprenant seuls.”
- des exemples concrets:
 - ▶ détection et classification d'objets (images, vidéos), de sons (flux audio), d'odeurs (capteurs chimiques): on mime ici les sens animaux
 - ▶ apprentissage de jeux, de compétitions par entraînements (ex: échecs, Go): ce qu'on appelle **apprentissage par renforcement**
 - ▶ génération d'images, sons réalistes (le côté “artistique” de l'IA)

IA et “apprentissage automatique”: (**machine learning**)

- le “machine learning” est le domaine des **mathématiques appliquées et de l'informatique** qui étudie, développe, évalue et améliore sans cesse les outils à la base des algorithmes de l'IA.

Tentative de définition de l'IA

Finalement, l'intelligence artificielle (IA), c'est quoi?

- pas de définition claire, mais en gros, c'est:
“l'ensemble des algorithmes qui, à partir de stimuli, (i) prennent des décisions et (ii) s'améliorent en apprenant seuls.”
- des exemples concrets:
 - ▶ détection et classification d'objets (images, vidéos), de sons (flux audio), d'odeurs (capteurs chimiques): on mime ici les sens animaux
 - ▶ apprentissage de jeux, de compétitions par entraînements (ex: échecs, Go): ce qu'on appelle **apprentissage par renforcement**
 - ▶ génération d'images, sons réalistes (le côté “artistique” de l'IA)

IA et “apprentissage automatique”: (machine learning)

- le “machine learning” est le domaine des **mathématiques appliquées et de l'informatique** qui étudie, développe, évalue et améliore sans cesse les outils à la base des algorithmes de l'IA.
- c'est ce que vous allez étudier dans ce cours!

1 Logistique du cours

2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases

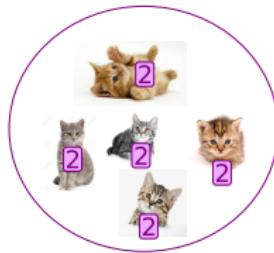
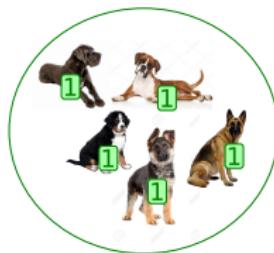
- Bases historiques: de l'homme à la machine
- **Bases scientifiques: de l'IA à l'apprentissage machine**
- Acte 1: Les réseaux de neurones: de la fiction à la réalité?
- Acte 2: Le retour en force des mathématiques
- Acte 3: La révolution des réseaux profonds
- Aujourd'hui: où cela nous mène-t-il?

3 Des données aux vecteurs: séparabilité, représentations et kNN

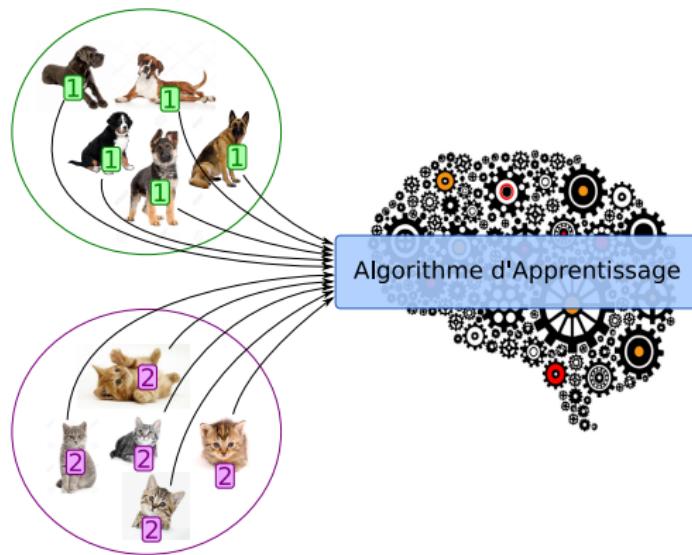
4 Les réseaux de neurones

5 L'IA moderne: révolution informatique et retour des mathématiques

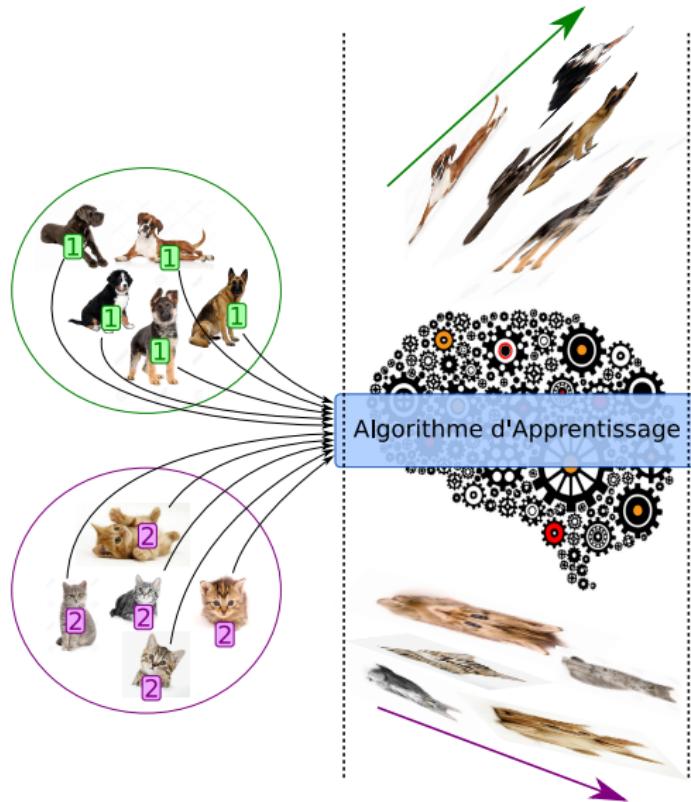
Apprentissage machine: le problème de classification



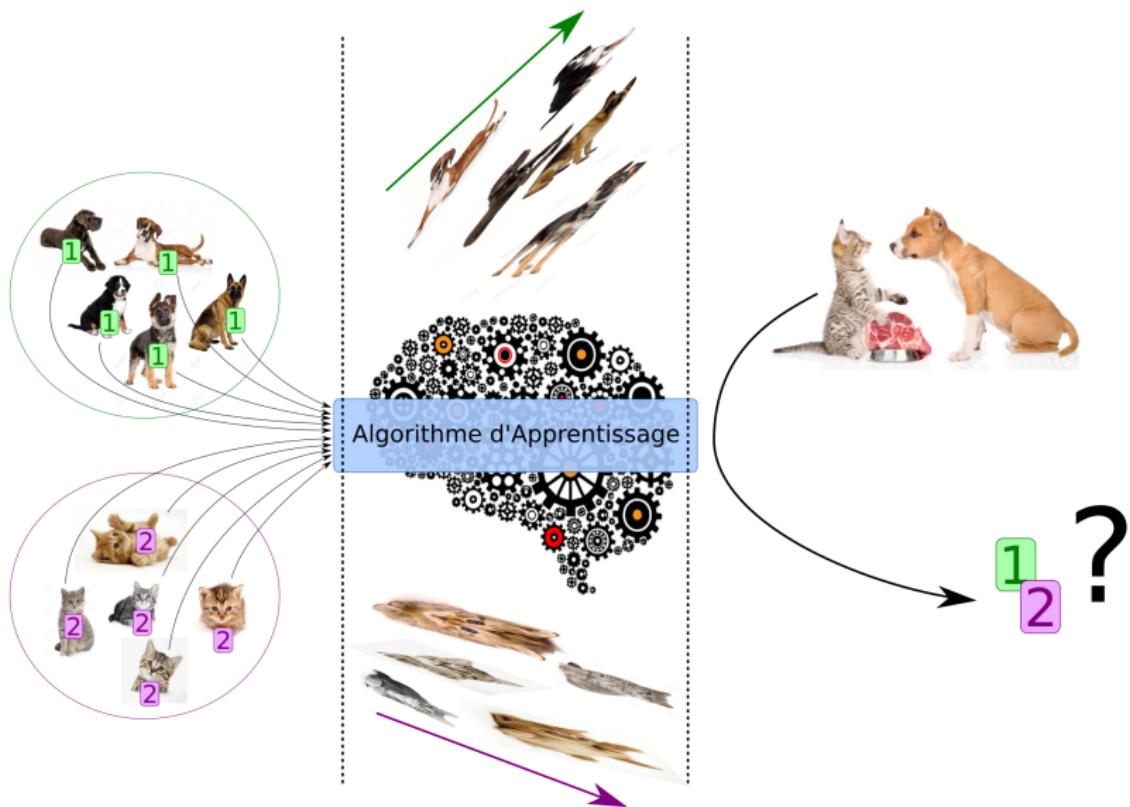
Apprentissage machine: le problème de classification



Apprentissage machine: le problème de classification



Apprentissage machine: le problème de classification



La réponse

La réponse

La version courte:

La réponse

La version courte: Personne ne sait vraiment le résoudre.

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

- **D'après le mathématicien:**

✓ C'est un problème **d'optimisation**

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

- **D'après le mathématicien:**

- C'est un problème **d'optimisation**
- OK, mais quelle est la **fonction objectif**?

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

- **D'après le mathématicien:**

- ✓ C'est un problème d'optimisation**
- ✗ OK, mais quelle est la fonction objectif?**
- ✗ et comment modélise-t-on les données mathématiquement?**

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

- **D'après le mathématicien:**

- ✓ C'est un problème **d'optimisation**
- ✗ OK, mais quelle est la **fonction objectif**?
- ✗ et comment modélise-t-on les données mathématiquement?
- ✗ Et, quand bien même on sait écrire le problème, sait-on le résoudre?

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

- **D'après le mathématicien:**

- C'est un problème **d'optimisation**
- OK, mais quelle est la **fonction objectif**?
- et comment modélise-t-on les données mathématiquement?
- Et, quand bien même on sait écrire le problème, sait-on le résoudre?

- **D'après l'informaticien:**

- Notre cerveau a des neurones et des axones, mon ordinateur des câbles et des transistors, apprenons aux machines ce qu'on apprend aux enfants.

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

- **D'après le mathématicien:**

- C'est un problème **d'optimisation**
- ✗ OK, mais quelle est la **fonction objectif**?
- ✗ et comment modélise-t-on les données mathématiquement?
- ✗ Et, quand bien même on sait écrire le problème, sait-on le résoudre?

- **D'après l'informaticien:**

- Notre cerveau a des neurones et des axones, mon ordinateur des câbles et des transistors, apprenons aux machines ce qu'on apprend aux enfants.
- ✗ Très bien, mais comment le cerveau apprend-t-il?

La réponse

La version courte: Personne ne sait vraiment le résoudre.

et c'est un peu pour ça qu'on a besoin de vous!

Les réponses “scientifiquement correctes”:

- **D'après le mathématicien:**

- ✓ C'est un problème **d'optimisation****
- ✗ OK, mais quelle est la **fonction objectif**?**
- ✗ et comment modélise-t-on les données mathématiquement?**
- ✗ Et, quand bien même on sait écrire le problème, sait-on le résoudre?**

- **D'après l'informaticien:**

- ✓ Notre cerveau a des neurones et des axones, mon ordinateur des câbles et des transistors, apprenons aux machines ce qu'on apprend aux enfants.**
- ✗ Très bien, mais comment le cerveau apprend-t-il?**
- ✗ Et, en supposant qu'on puisse, quelles sont nos **garanties de performances et contrôles**?**

1 Logistique du cours

2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases

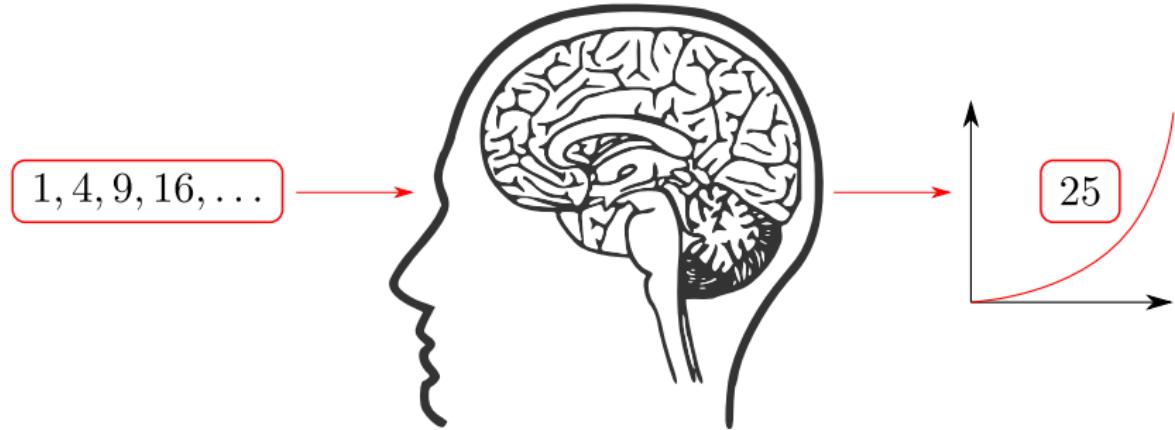
- Bases historiques: de l'homme à la machine
- Bases scientifiques: de l'IA à l'apprentissage machine
- **Acte 1: Les réseaux de neurones: de la fiction à la réalité?**
- Acte 2: Le retour en force des mathématiques
- Acte 3: La révolution des réseaux profonds
- Aujourd'hui: où cela nous mène-t-il?

3 Des données aux vecteurs: séparabilité, représentations et kNN

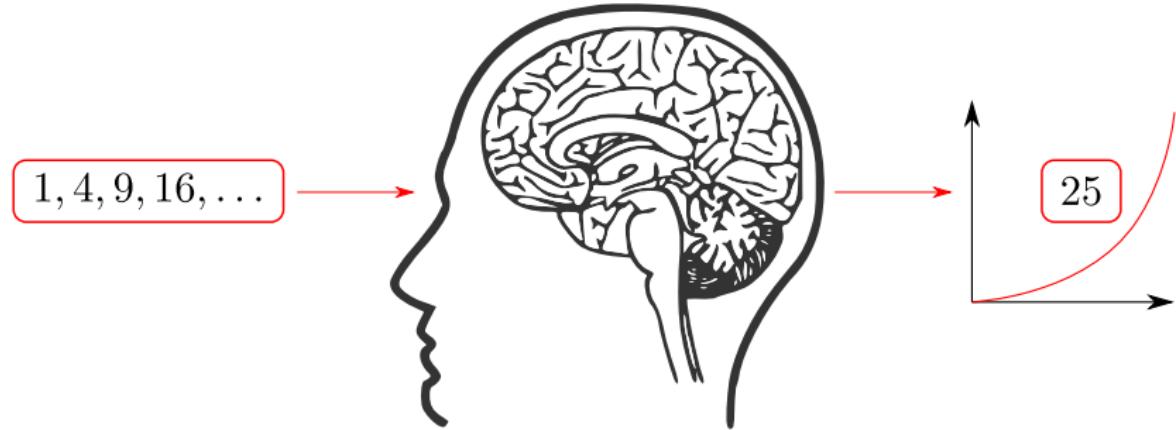
4 Les réseaux de neurones

5 L'IA moderne: révolution informatique et retour des mathématiques

Le cerveau comme “générateur de fonctions”

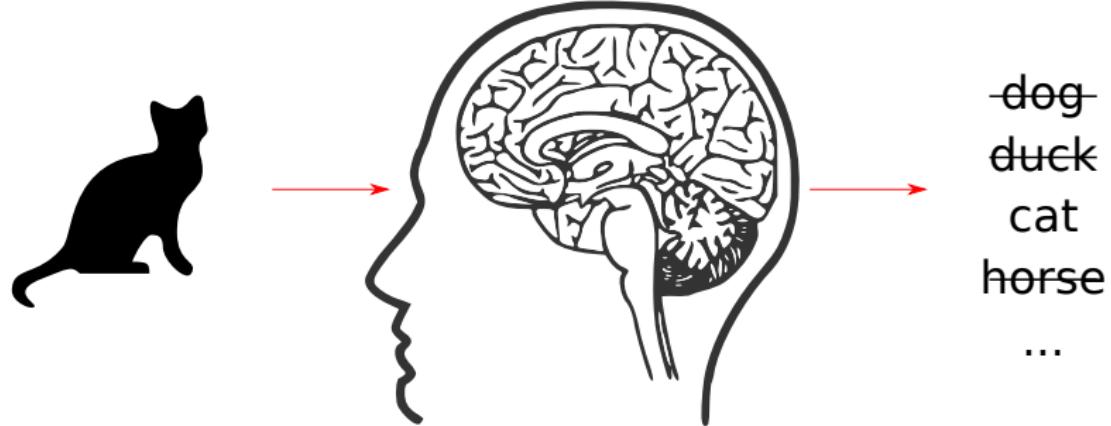


Le cerveau comme “générateur de fonctions”

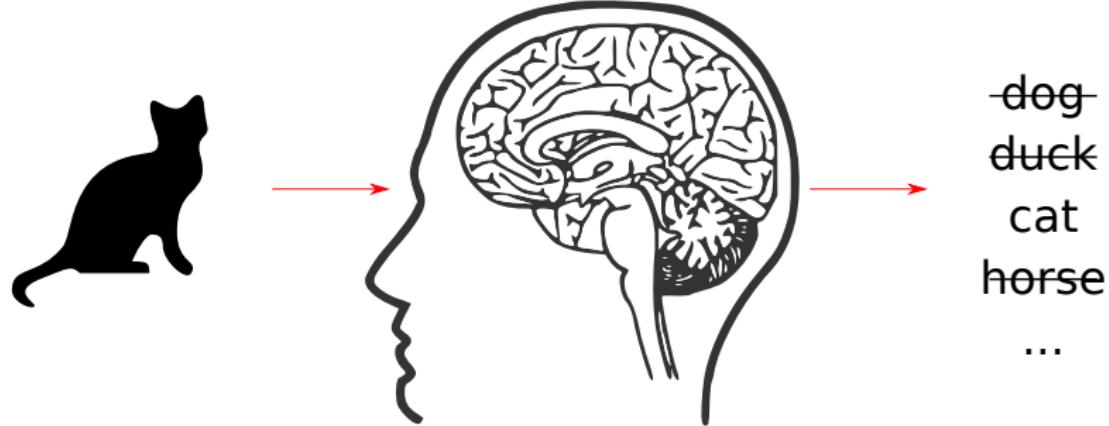


Le cerveau peut inférer la fonction $y = x^2$.

Le cerveau comme “générateur de fonctions”



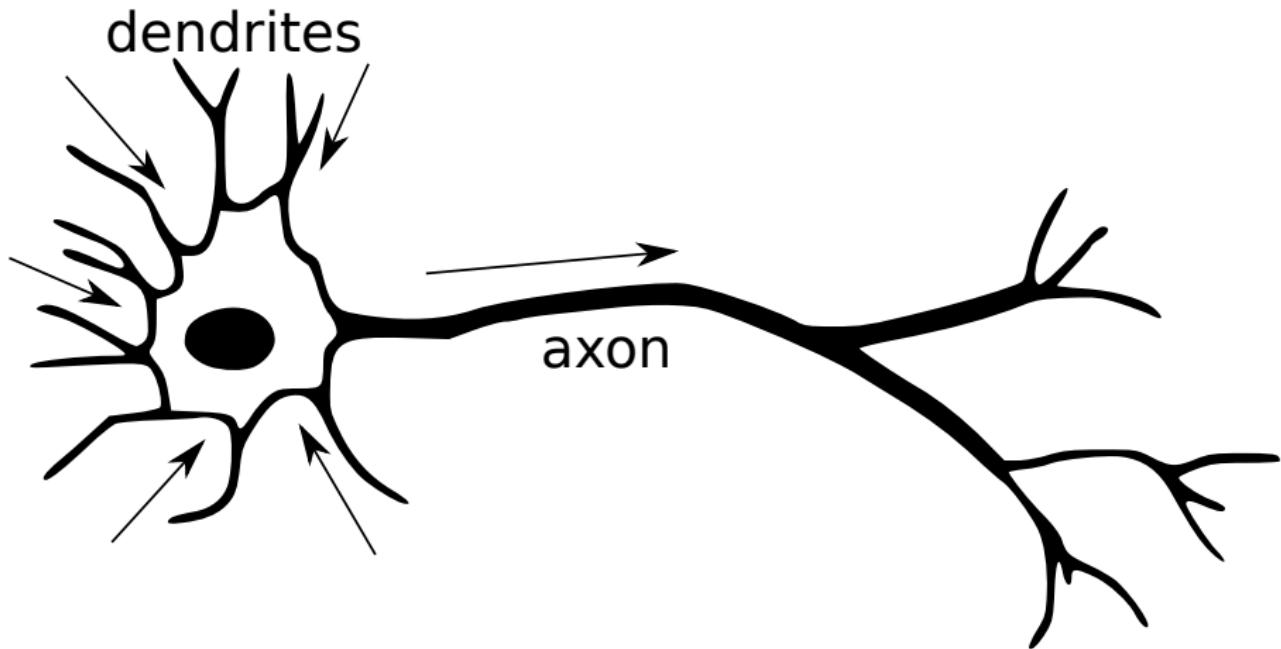
Le cerveau comme “générateur de fonctions”



Le cerveau peut inférer la fonction **difficile** $y = 1_{x \in \{\text{cat}\}}$

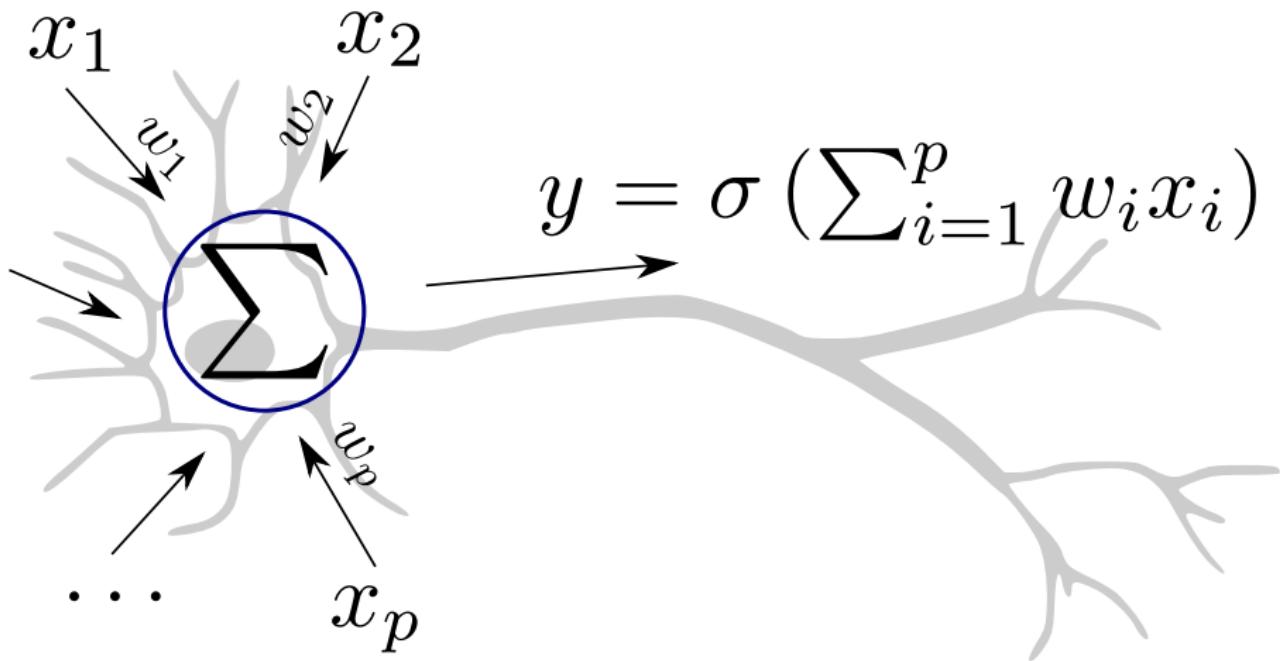
Les neurones biologiques en équations

Le signal des *dendrites* dépolarise l'*axone* **si et seulement si** la somme des signaux **dépasse un seuil**.

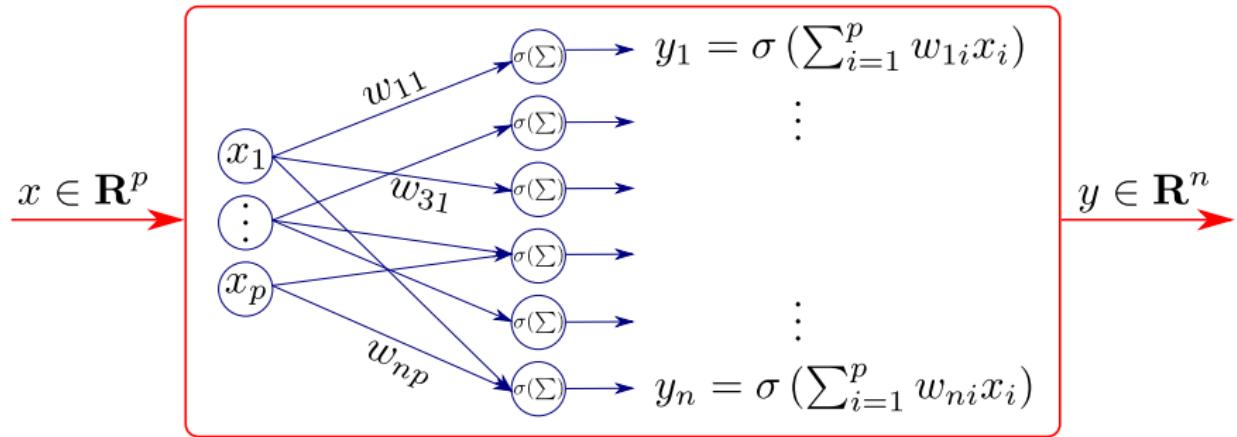


Les neurones biologiques en équations

Le signal des *dendrites* dépolarise l'*axone* **si et seulement si** la somme des signaux **dépasse un seuil**.

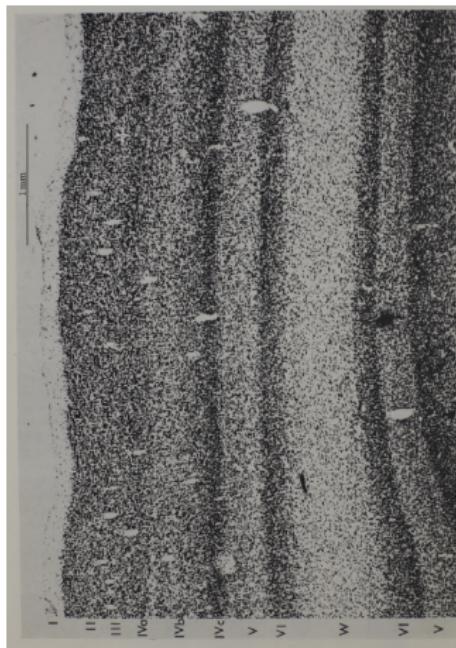


Le modèle du “perceptron”

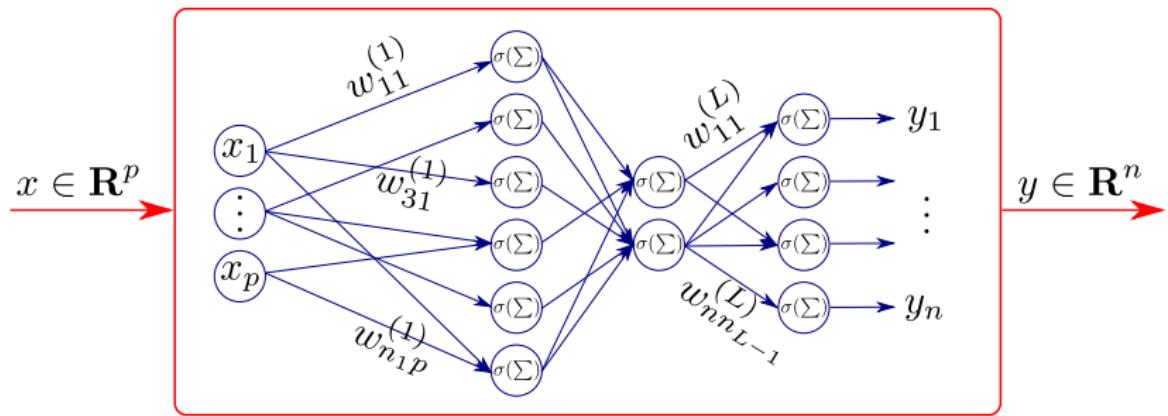


Des perceptrons aux réseaux “profonds”

Hubel, Wiesel, "Functional architecture of macaque monkey visual cortex", 1977.



Des perceptrons aux réseaux “profonds”



Mais...

Construire un réseau de neurones fonctionnel:

Mais...

Construire un réseau de neurones fonctionnel:

- demande énormément de données d'entraînement

Mais...

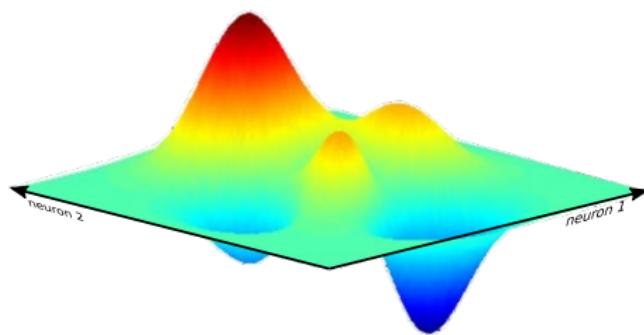
Construire un réseau de neurones fonctionnel:

- demande énormément de données d'entraînement
- est fortement consommateur de ressources

Mais...

Construire un réseau de neurones fonctionnel:

- demande énormément de données d'entraînement
- est fortement consommateur de ressources
- **plus important:** n'est pas un "problème convexe": conduit en général à de mauvais résultats!



1 Logistique du cours

2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases

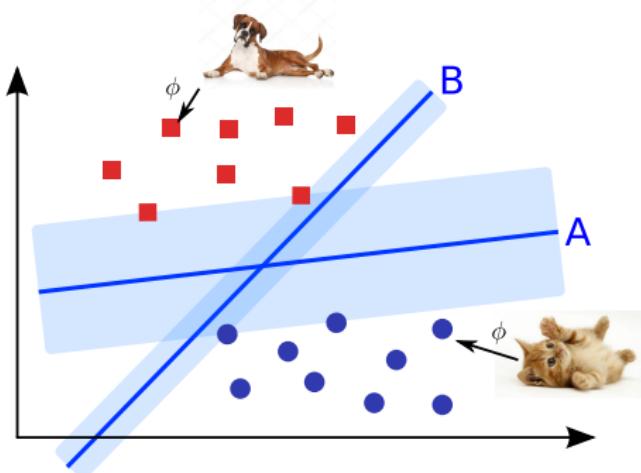
- Bases historiques: de l'homme à la machine
- Bases scientifiques: de l'IA à l'apprentissage machine
- Acte 1: Les réseaux de neurones: de la fiction à la réalité?
- **Acte 2: Le retour en force des mathématiques**
- Acte 3: La révolution des réseaux profonds
- Aujourd'hui: où cela nous mène-t-il?

3 Des données aux vecteurs: séparabilité, représentations et kNN

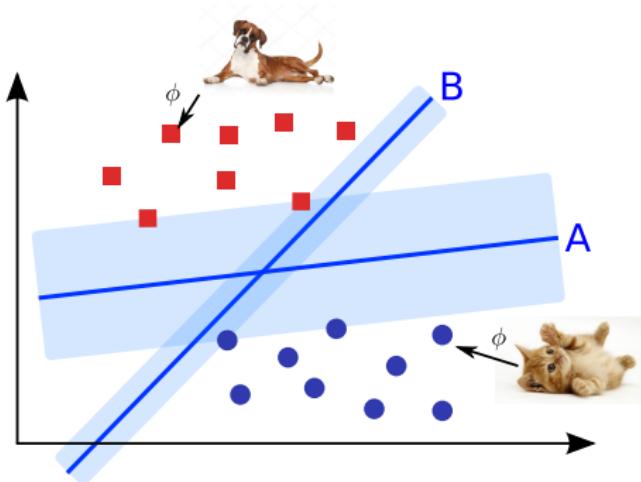
4 Les réseaux de neurones

5 L'IA moderne: révolution informatique et retour des mathématiques

Des données aux vecteurs: séparabilité linéaire



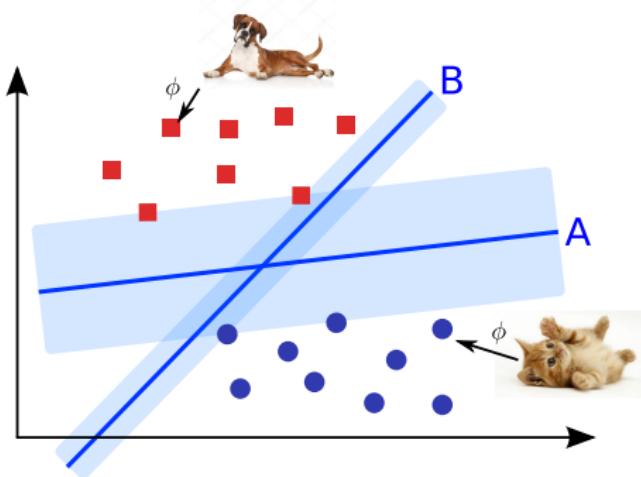
Des données aux vecteurs: séparabilité linéaire



Question centrale: trouver une **représentation** φ telle que:

φ : “images” \mapsto “**représentations** séparables”.

Des données aux vecteurs: séparabilité linéaire



Question centrale: trouver une **représentation** φ telle que:

φ : “images” \mapsto “**représentations** séparables”.

En général créées: à la main (histogramme des gradients orientés: **HOG**)
ou statistiquement (analyse en composantes principales: **ACP**) .

Mais...

Construire des séparateurs linéaires (ex: machine à vecteurs support: SVM):

Mais...

Construire des séparateurs linéaires (ex: machine à vecteurs support: SVM):

- demande des fonctions φ puissantes (donc de l'ingénierie)

Mais...

Construire des séparateurs linéaires (ex: machine à vecteurs support: SVM):

- demande des fonctions φ puissantes (donc de l'ingénierie)
- a des performances moyennes (**bien inférieures à l'humain**)

Mais...

Construire des séparateurs linéaires (ex: machine à vecteurs support: SVM):

- demande des fonctions φ puissantes (donc de l'ingénierie)
- a des performances moyennes (**bien inférieures à l'humain**)
- mathématiquement accessible, mais pas entièrement.

Et le vainqueur est...

Et le vainqueur est...

Les séparateurs linéaires! et de loin! ...

Et le vainqueur est...

Les séparateurs linéaires! et de loin! ... jusqu'à récemment ...

1 Logistique du cours

2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases

- Bases historiques: de l'homme à la machine
- Bases scientifiques: de l'IA à l'apprentissage machine
- Acte 1: Les réseaux de neurones: de la fiction à la réalité?
- Acte 2: Le retour en force des mathématiques
- Acte 3: La révolution des réseaux profonds**
- Aujourd'hui: où cela nous mène-t-il?

3 Des données aux vecteurs: séparabilité, représentations et kNN

4 Les réseaux de neurones

5 L'IA moderne: révolution informatique et retour des mathématiques

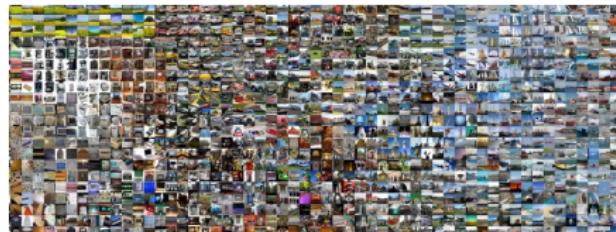
L'avènement des réseaux profonds (DNN)

Oubliez ce que j'ai dit sur l'instabilité des réseaux de neurones... Étrangement,



L'avènement des réseaux profonds (DNN)

Oubliez ce que j'ai dit sur l'instabilité des réseaux de neurones... Étrangement,



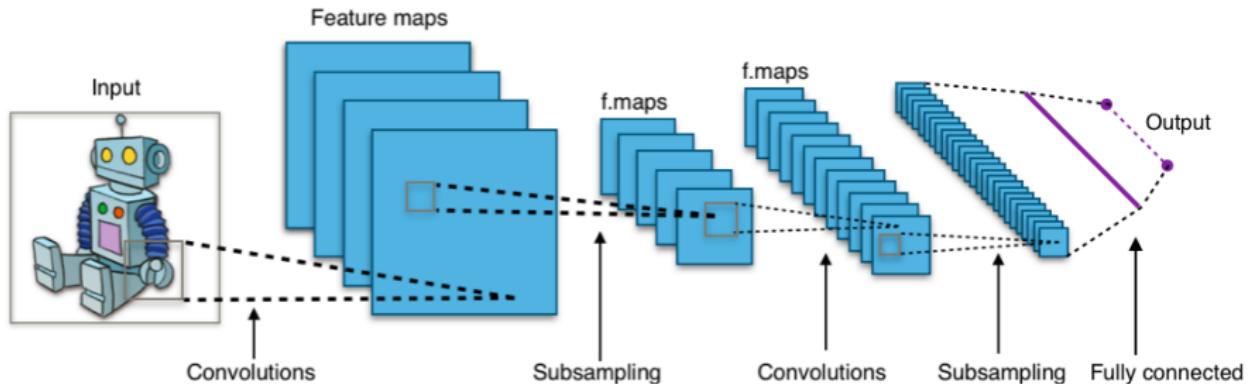
+



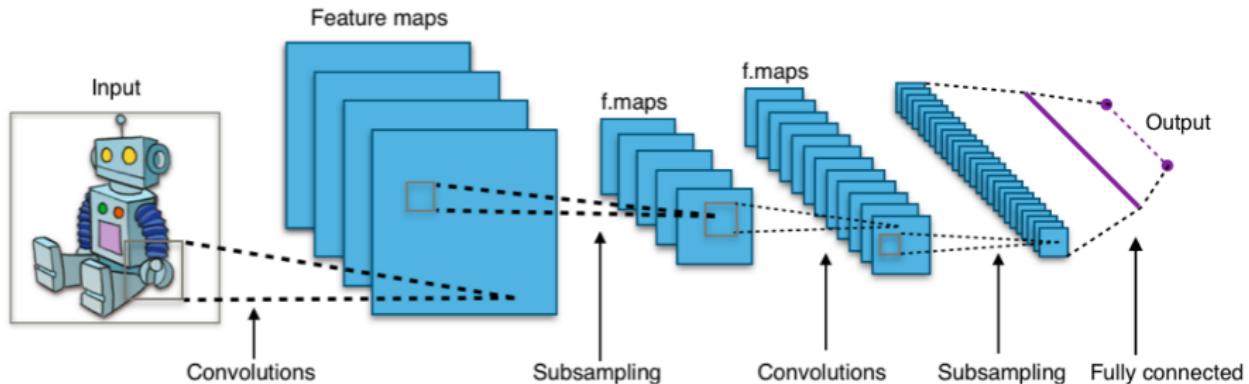
=



Bref coup d'oeil sur un DNN



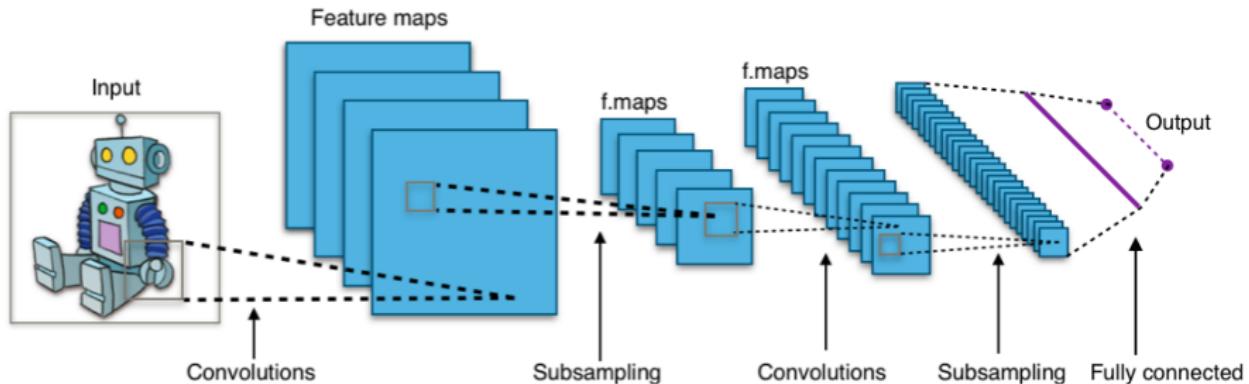
Bref coup d'oeil sur un DNN



Les Deep Neural Networks:

- sont “juste” des grands réseaux de neurones

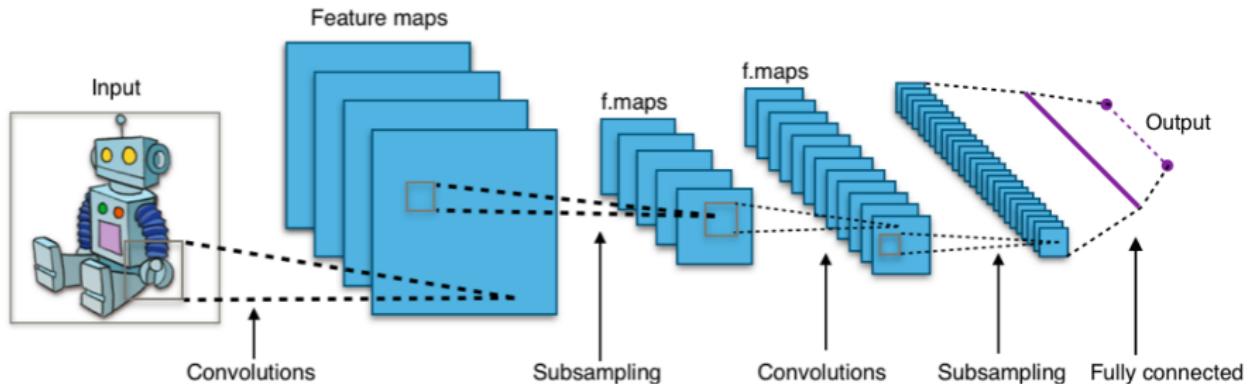
Bref coup d'oeil sur un DNN



Les Deep Neural Networks:

- sont “juste” des grands réseaux de neurones
- qui sont **surprenants de stabilité** (ils convergent vers de bons minima)

Bref coup d'oeil sur un DNN



Les Deep Neural Networks:

- sont “juste” des grands réseaux de neurones
- qui sont **surprenants de stabilité** (ils convergent vers de bons minima)
- et parfois atteignent des **performances sur-humaines!**

Mais...

- on ne sait toujours pas pourquoi et comment ils fonctionnent!

Mais...

- on ne sait toujours pas pourquoi et comment ils fonctionnent!
- sont extrêmement énergivores

Mais...

- on ne sait toujours pas pourquoi et comment ils fonctionnent!
- sont extrêmement énergivores
- ont besoin de gigantesques bases de données étiquetées (ex: millions d'images)

1 Logistique du cours

2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases

- Bases historiques: de l'homme à la machine
- Bases scientifiques: de l'IA à l'apprentissage machine
- Acte 1: Les réseaux de neurones: de la fiction à la réalité?
- Acte 2: Le retour en force des mathématiques
- Acte 3: La révolution des réseaux profonds
- Aujourd'hui: où cela nous mène-t-il?

3 Des données aux vecteurs: séparabilité, représentations et kNN

4 Les réseaux de neurones

5 L'IA moderne: révolution informatique et retour des mathématiques

Retour nécessaire des mathématiques

Ce qu'on commence à comprendre:

Retour nécessaire des mathématiques

Ce qu'on commence à comprendre:

- les données **grandes et nombreuses** apportent stabilité et “concentration” (loi des grands nombres).

Retour nécessaire des mathématiques

Ce qu'on commence à comprendre:

- les données **grandes et nombreuses** apportent stabilité et “concentration” (loi des grands nombres).
- les grandes données se modélisent très bien par des **modèles probabilistes élémentaires**

Retour nécessaire des mathématiques

Ce qu'on commence à comprendre:

- les données **grandes et nombreuses** apportent stabilité et “concentration” (loi des grands nombres).
- les grandes données se modélisent très bien par des **modèles probabilistes élémentaires**
- plus crucial: les **grandes données se comportent très différemment des petites.**

Retour nécessaire des mathématiques

Ce qu'on commence à comprendre:

- les données **grandes et nombreuses** apportent stabilité et “concentration” (loi des grands nombres).
- les grandes données se modélisent très bien par des **modèles probabilistes élémentaires**
- plus crucial: les **grandes données se comportent très différemment des petites**.
- de nombreuses intuitions passées sont fausses: **il y beaucoup à refaire.**

En L1 INF, que fait-on de tout ça?

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
 - Des données aux vecteurs
 - Approche par séparabilité linéaire: l'algorithme SVM
 - Approche par distances dans l'espace des représentations: l'algorithme kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
 - Des données aux vecteurs
 - Approche par séparabilité linéaire: l'algorithme SVM
 - Approche par distances dans l'espace des représentations: l'algorithme kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques

Un langage commun: l'espace vectoriel \mathbb{R}^p

Ce que l'on appelle "données" peut être un peu tout:

- des images de résolutions/tailles différentes (= des ensembles de pixels)
- des séries temporelles de sons (encodées ou non dans des formats différents)
- du texte en **langage naturel**
- des séquences ADN (ATGC), des graphes de molécules, etc.

Un langage commun: l'espace vectoriel \mathbb{R}^p

Ce que l'on appelle "données" peut être un peu tout:

- des images de résolutions/tailles différentes (= des ensembles de pixels)
- des séries temporelles de sons (encodées ou non dans des formats différents)
- du texte en **langage naturel**
- des séquences ADN (ATGC), des graphes de molécules, etc.

Mais un seul langage en informatique pour les représenter: des séquences binaires (01001011101)

- chaque donnée est "transformée", **représentée** en une suite de chiffres.
- divers types: binaire (info booléenne), catégorielle (classe d'un attribut), mais le plus souvent réel (échelle, valeur).
- souvent, chaque donnée sera donc représentée par un point dans

$$\mathbb{R}^p = \{(x_1, \dots, x_p), \forall i, x_i \in \mathbb{R}\}.$$

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **Rappel de l'objectif:** classer des données de mêmes catégories.
- **Images** (déttection d'objet, classification, génération d'images virtuelles):

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **Rappel de l'objectif:** classer des données de mêmes catégories.
- **Images** (détection d'objet, classification, génération d'images virtuelles):
 - ▶ ensemble des pixels (de gauche à droite, de haut en bas)

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **Rappel de l'objectif:** classer des données de mêmes catégories.
- **Images** (détection d'objet, classification, génération d'images virtuelles):
 - ▶ ensemble des pixels (de gauche à droite, de haut en bas)
 - ★ idée naïve: très vite trop grands vecteurs pour des grandes images ($1024 \times 768 = 786\,432$)
 - ★ pas du tout robuste à un déplacement, rotation de l'objet dans l'image!
 - ★ contournement possible: ajouter des images déplacées, tournées dans la base de données (idée à garder en tête pour le projet!)

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **Rappel de l'objectif:** classer des données de mêmes catégories.
- **Images** (détection d'objet, classification, génération d'images virtuelles):
 - ▶ ensemble des pixels (de gauche à droite, de haut en bas)
 - ★ idée naïve: très vite trop grands vecteurs pour des grandes images ($1024 \times 768 = 786\,432$)
 - ★ pas du tout robuste à un déplacement, rotation de l'objet dans l'image!
 - ★ contournement possible: ajouter des images déplacées, tournées dans la base de données (idée à garder en tête pour le projet!)
 - ▶ représentations adaptées aux images: SURF, SIFT, HOG...

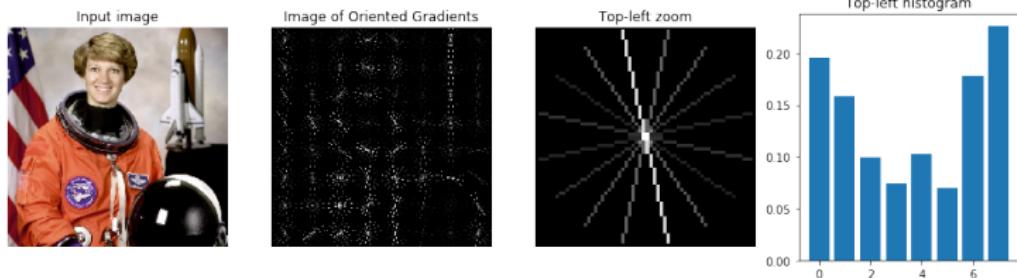
Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **Rappel de l'objectif:** classer des données de mêmes catégories.
- **Images** (détection d'objet, classification, génération d'images virtuelles):
 - ▶ ensemble des pixels (de gauche à droite, de haut en bas)
 - ★ idée naïve: très vite trop grands vecteurs pour des grandes images ($1024 \times 768 = 786\,432$)
 - ★ pas du tout robuste à un déplacement, rotation de l'objet dans l'image!
 - ★ contournement possible: ajouter des images déplacées, tournées dans la base de données (idée à garder en tête pour le projet!)
 - ▶ représentations adaptées aux images: SURF, SIFT, HOG...
 - ★ l'exemple de HOG: **histogramme des gradients orientés**
 - ★ plus flexible, plus "discriminant", taille adaptable des représentations
 - ★ limite le problème de décalage, rotation, échelle, mais pas entièrement
 - ★ n'apporte pas d'info "sémantique" (animal vs. objet, situation...)

Un langage commun: l'espace vectoriel \mathbb{R}^p

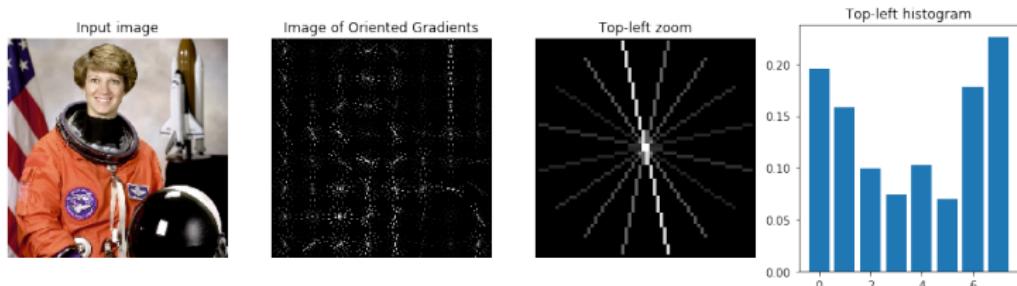
```
1 from skimage import data  
2 my_image = data.astronaut()  
3  
4 from skimage.feature import hog  
5 x, hog_image = hog(my_image, orientations=8, pixels_per_cell=(64,64), cells_per_block  
6 = (3,3), visualize=True)
```



```
x[:8]= [0.19564552 0.15833915 0.09890324 0.07407019 0.10239067 0.07030556 0.17850777 0.2264639 ]
```

Un langage commun: l'espace vectoriel \mathbb{R}^p

```
1 from skimage import data  
2 my_image = data.astronaut()  
3  
4 from skimage.feature import hog  
5 x, hog_image = hog(my_image, orientations=8, pixels_per_cell=(64,64), cells_per_block  
6 =(3,3), visualize=True)
```



```
x[:8] = [0.19564552 0.15833915 0.09890324 0.07407019 0.10239067 0.07030556 0.17850777 0.2264639 ]
```

- ▶ aujourd'hui: représentation par **couche intermédiaire** (souvent avant-dernière) d'un réseau de neurones profond!
 - ★ résilient aux variations d'images présentes dans la base de données
 - ★ ajoutent un contenu sémantique (découvert par le réseau)
 - ★ mais que signifie cette représentation créée par le réseau?

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **texte en langage naturel** (classification, traduction, générateur de réponses)
 - ▶ domaine de recherche entier: **traitement naturel du langage** (NLP)
 - ▶ granularité variée: mots, phrases, paragraphes, ouvrage complet

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **texte en langage naturel** (classification, traduction, générateur de réponses)
 - ▶ domaine de recherche entier: **traitement naturel du langage** (NLP)
 - ▶ granularité variée: mots, phrases, paragraphes, ouvrage complet
 - ▶ représentation par fréquences dans un dictionnaire: **bag of words, tf*idf**

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemples de **représentations** de données:

- **texte en langage naturel** (classification, traduction, générateur de réponses)
 - ▶ domaine de recherche entier: **traitement naturel du langage** (NLP)
 - ▶ granularité variée: mots, phrases, paragraphes, ouvrage complet
 - ▶ représentation par fréquences dans un dictionnaire: **bag of words, tf*idf**
 - ★ vecteur de taille 26 pour les mots, mais de taille $\sim 10\,000$ pour le texte!
 - ★ pas d'info sémantique, pas de notion d'ordre, juste des occurrences

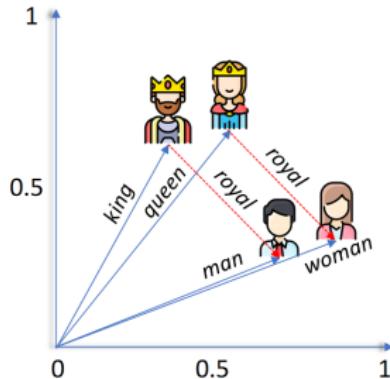
Un langage commun: l'espace vectoriel \mathbb{R}^p

- ▶ représentations modernes: par réseaux profonds (**word2vec**, **BERT**, **transformeurs**)

Un langage commun: l'espace vectoriel \mathbb{R}^p

- ▶ représentations modernes: par réseaux profonds (**word2vec, BERT, transformeurs**)
 - ★ très puissant, espace vectoriel “sémantique”: ex.,

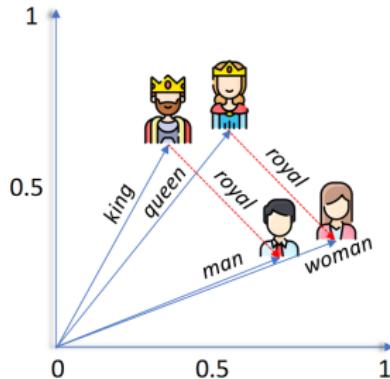
$$\text{vec(roi)} + \{\text{vec(femme)} - \text{vec(homme)}\} = \text{vec(reine)}$$



Un langage commun: l'espace vectoriel \mathbb{R}^p

- ▶ représentations modernes: par réseaux profonds (**word2vec, BERT, transformeurs**)
 - ★ très puissant, espace vectoriel “sémantique”: ex.,

$$\text{vec(roi)} + \{\text{vec(femme)} - \text{vec(homme)}\} = \text{vec(reine)}$$

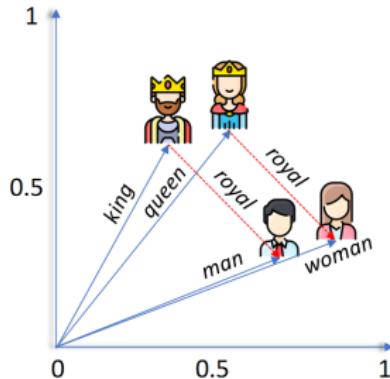


- ★ mais incontrôlable: issu de l'apprentissage du réseau

Un langage commun: l'espace vectoriel \mathbb{R}^p

- ▶ représentations modernes: par réseaux profonds (**word2vec, BERT, transformeurs**)
 - ★ très puissant, espace vectoriel “sémantique”: ex.,

$$\text{vec(roi)} + \{\text{vec(femme)} - \text{vec(homme)}\} = \text{vec(reine)}$$



- ★ mais incontrôlable: issu de l'apprentissage du réseau
- ★ extrêmement consommateur à la génération: eqCO₂ = 3 SUVs pour le dernier réseau en date!

Un langage commun: l'espace vectoriel \mathbb{R}^p

Un langage commun: l'espace vectoriel \mathbb{R}^p

La base de données:

- données “brutes” non traitables par la machine

$$s_1, \dots, s_n \in \mathcal{E}$$

- transformées en p représentations vectorielles x_i dans \mathbb{R}^p

$$\varphi : \mathcal{E} \rightarrow \mathbb{R}^p$$

$$s_i \mapsto x_i = \varphi(s_i)$$

Un langage commun: l'espace vectoriel \mathbb{R}^p

La base de données:

- données “brutes” non traitables par la machine

$$s_1, \dots, s_n \in \mathcal{E}$$

- transformées en p représentations vectorielles x_i dans \mathbb{R}^p

$$\varphi : \mathcal{E} \rightarrow \mathbb{R}^p$$

$$s_i \mapsto x_i = \varphi(s_i)$$

- on oublie alors les s_i et on travaille avec la base de données

$$x_1, \dots, x_n \in \mathbb{R}^p, \quad x_i = [x_{i1}, \dots, x_{ip}]$$

Un langage commun: l'espace vectoriel \mathbb{R}^p

La base de données:

- données “brutes” non traitables par la machine

$$s_1, \dots, s_n \in \mathcal{E}$$

- transformées en p représentations vectorielles x_i dans \mathbb{R}^p

$$\varphi : \mathcal{E} \rightarrow \mathbb{R}^p$$

$$s_i \mapsto x_i = \varphi(s_i)$$

- on oublie alors les s_i et on travaille avec la **base de données**

$$x_1, \dots, x_n \in \mathbb{R}^p, \quad x_i = [x_{i1}, \dots, x_{ip}]$$

aussi représentable en une matrice (ensemble de lignes)

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n \times p}$$

Un langage commun: l'espace vectoriel \mathbb{R}^p

La base de données:

- pour la classification, on ajoute des **étiquettes** (ou labels)

$$y_1, \dots, y_n \in \mathcal{Y} = \{\text{classe 1}, \text{classe 2}, \dots, \text{classe } k\}$$

connus ou inconnus!

Un langage commun: l'espace vectoriel \mathbb{R}^p

La base de données:

- pour la classification, on ajoute des **étiquettes** (ou labels)

$$y_1, \dots, y_n \in \mathcal{Y} = \{\text{classe 1}, \text{classe 2}, \dots, \text{classe } k\}$$

connus ou inconnus!

- la **base de données** devient alors un couple

$$(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^p \times \mathcal{Y}$$

Un langage commun: l'espace vectoriel \mathbb{R}^p

La base de données:

- pour la classification, on ajoute des **étiquettes** (ou labels)

$$y_1, \dots, y_n \in \mathcal{Y} = \{\text{classe 1, classe 2, \dots, classe } k\}$$

connus ou inconnus!

- la **base de données** devient alors un couple

$$(x_1, y_1), \dots, (x_n, y_n) \in \mathbb{R}^p \times \mathcal{Y}$$

aussi représentable en un couple (matrice,vecteur)

$$(X, y), \quad y = [y_1, \dots, y_n] \in \mathcal{Y}^n$$

Un langage commun: l'espace vectoriel \mathbb{R}^p

Point central: entraînement et test

Un langage commun: l'espace vectoriel \mathbb{R}^p

Point central: entraînement et test

- pour vérifier qu'un algorithme d'apprentissage fonctionne, il faut pouvoir le tester sur **d'autres données étiquetées que celles utilisées pendant l'apprentissage**

Un langage commun: l'espace vectoriel \mathbb{R}^p

Point central: entraînement et test

- pour vérifier qu'un algorithme d'apprentissage fonctionne, il faut pouvoir le tester sur **d'autres données étiquetées que celles utilisées pendant l'apprentissage**
- on va donc créer en fait deux jeux de données: une **base d'entraînement** (utilisée pour construire l'algorithme) et une **base de test** (utilisée pour vérifier que l'algorithme fonctionne sur d'autres données)

Un langage commun: l'espace vectoriel \mathbb{R}^p

Point central: entraînement et test

- pour vérifier qu'un algorithme d'apprentissage fonctionne, il faut pouvoir le tester sur **d'autres données étiquetées que celles utilisées pendant l'apprentissage**
- on va donc créer en fait deux jeux de données: une **base d'entraînement** (utilisée pour construire l'algorithme) et une **base de test** (utilisée pour vérifier que l'algorithme fonctionne sur d'autres données)
- en général, on n'a accès qu'à une seule base de données: on va donc la diviser en **base d'entraînement** et **base de test**

Un langage commun: l'espace vectoriel \mathbb{R}^p

Point central: entraînement et test

- pour vérifier qu'un algorithme d'apprentissage fonctionne, il faut pouvoir le tester sur **d'autres données étiquetées que celles utilisées pendant l'apprentissage**
- on va donc créer en fait deux jeux de données: une **base d'entraînement** (utilisée pour construire l'algorithme) et une **base de test** (utilisée pour vérifier que l'algorithme fonctionne sur d'autres données)
- en général, on n'a accès qu'à une seule base de données: on va donc la diviser en **base d'entraînement** et **base de test**
- **DANGER!** Le **surapprentissage** = apprendre "trop bien" la base d'entraînement mais ne pas bien **généraliser** à la base test.

Un langage commun: l'espace vectoriel \mathbb{R}^p

Point central: entraînement et test

- pour vérifier qu'un algorithme d'apprentissage fonctionne, il faut pouvoir le tester sur **d'autres données étiquetées que celles utilisées pendant l'apprentissage**
- on va donc créer en fait deux jeux de données: une **base d'entraînement** (utilisée pour construire l'algorithme) et une **base de test** (utilisée pour vérifier que l'algorithme fonctionne sur d'autres données)
- en général, on n'a accès qu'à une seule base de données: on va donc la diviser en **base d'entraînement** et **base de test**
- **DANGER!** Le **surapprentissage** = apprendre "trop bien" la base d'entraînement mais ne pas bien **généraliser** à la base test.
→ *on peut prouver que si $p > n$, on peut obtenir une erreur nulle sur la base d'entraînement, mais potentiellement énorme sur la base test.*

Un langage commun: l'espace vectoriel \mathbb{R}^p

Point central: entraînement et test

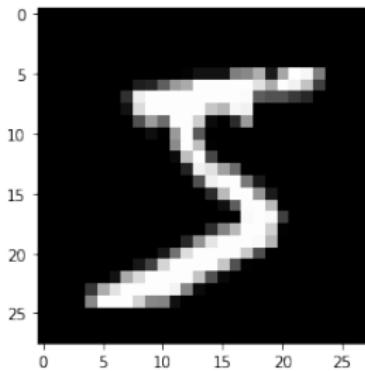
- pour vérifier qu'un algorithme d'apprentissage fonctionne, il faut pouvoir le tester sur **d'autres données étiquetées que celles utilisées pendant l'apprentissage**
- on va donc créer en fait deux jeux de données: une **base d'entraînement** (utilisée pour construire l'algorithme) et une **base de test** (utilisée pour vérifier que l'algorithme fonctionne sur d'autres données)
- en général, on n'a accès qu'à une seule base de données: on va donc la diviser en **base d'entraînement** et **base de test**
- **DANGER!** Le **surapprentissage** = apprendre "trop bien" la base d'entraînement mais ne pas bien **généraliser** à la base test.
→ *on peut prouver que si $p > n$, on peut obtenir une erreur nulle sur la base d'entraînement, mais potentiellement énorme sur la base test.*
- trouver la bonne division est une question difficile: en général, on prendra 90% de la base pour l'entraînement, 10% pour le test.

Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemple: la base **MNIST** (chiffres manuscrits $28 \times 28 (= 784)$ pixels):

```
1 from sklearn.datasets import fetch_openml
2 X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
3
4 print('Taille de la base de données:', np.shape(X))
5 _ = plt.imshow(X[0].reshape(28,28), cmap=plt.cm.gray)
6
```

Taille de la base de données: (70000, 784)

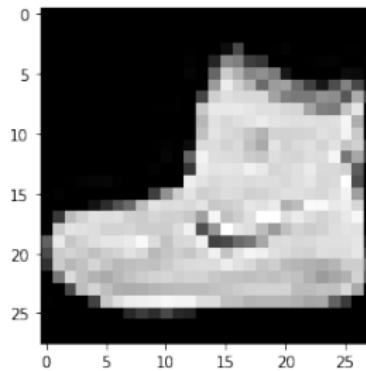


Un langage commun: l'espace vectoriel \mathbb{R}^p

Exemple: la base **Fashion-MNIST** (vêtements):

```
1 from sklearn.datasets import fetch_openml
2 X, y = fetch_openml('Fashion-MNIST', version=1, return_X_y=True)
3
4 print('Taille de la base de données:', np.shape(X))
5 _ = plt.imshow(X[0].reshape(28,28), cmap=plt.cm.gray)
6
```

Taille de la base de données: (70000, 784)

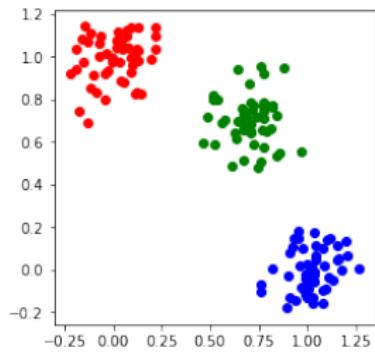


- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
 - Des données aux vecteurs
 - Approche par séparabilité linéaire: l'algorithme SVM
 - Approche par distances dans l'espace des représentations: l'algorithme kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques

Vecteurs dans \mathbb{R}^p et séparabilité linéaire

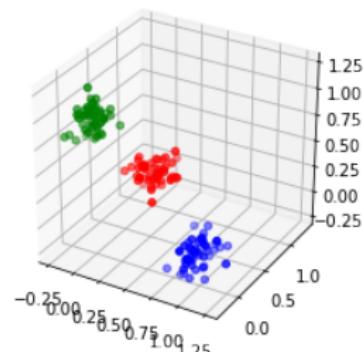
Représentation des données x_i linéairement séparables:

2D ($p = 2$)



ex: $x_i = [.02, .81]$

3D ($p = 3$)

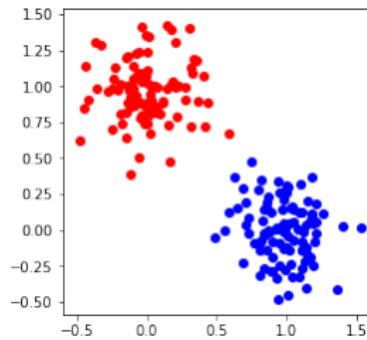


ex: $x_i = [.02, .81, .23]$

Vecteurs dans \mathbb{R}^p et séparabilité linéaire

Exemple: Séparabilité et non-séparabilité linéaire:

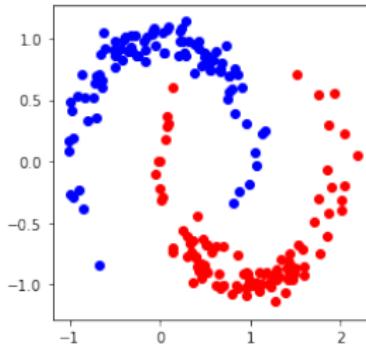
```
1 n=100
2
3 X1 = np.array([1,0])+.2*np.random.randn( n,2 )
4 X2 = np.array([0,1])+.2*np.random.randn( n,2 )
5
6 plt.figure(figsize=(4,4))
7 plt.scatter(X1[:,0],X1[:,1],color='blue')
8 plt.scatter(X2[:,0],X2[:,1],color='red')
```



Vecteurs dans \mathbb{R}^p et séparabilité linéaire

Exemple: Séparabilité et non-séparabilité linéaire:

```
1 n=100
2
3 Z = np.random.randn( n )*np.pi*.3
4 X1 = np.array([np.sin(Z),np.cos(Z)]).T+.1*np.random.randn( n,2 )
5
6 Z = np.random.randn( n )*np.pi*.3 + np.pi
7 X2 = np.array([1,0]).T+np.array([np.sin(Z),np.cos(Z)]).T+.1*np.random.randn( n,2 )
8
9 plt.figure(figsize=(4,4))
10 plt.scatter(X1[:,0],X1[:,1],color='blue')
11 plt.scatter(X2[:,0],X2[:,1],color='red')
```



L'algorithme SVM

Idée centrale: trouver un **hyperplan séparateur optimal**

L'algorithme SVM

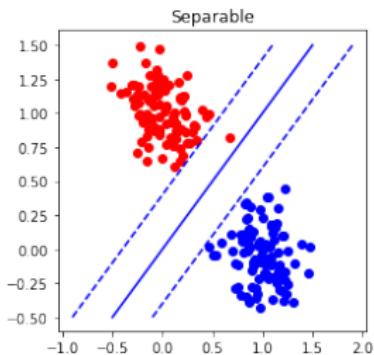
Idée centrale: trouver un **hyperplan séparateur optimal**

- suppose (**parfois à tort**) les données linéairement séparables

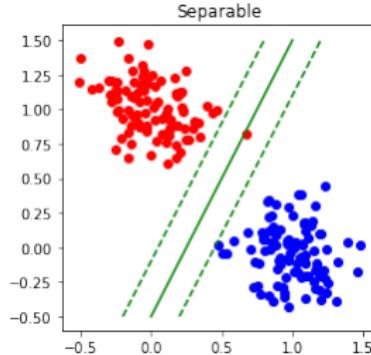
L'algorithme SVM

Idée centrale: trouver un **hyperplan séparateur optimal**

- suppose (**parfois à tort**) les données linéairement séparables
- parmi tous les hyperplans séparateurs, trouve celui de **marge maximale**



Hyperplan de marge optimale



Hyperplan sous-optimal

L'algorithme SVM

L'approche SVM: un problème d'**optimisation sous contraintes**

Hyperplan $\mathcal{H}^* = \arg \max_{\mathcal{H}} \{\text{marge maximale} \mid \forall i, \mathbf{x}_i \text{ du bon côté}\}.$

Ensuite, pour un nouveau \mathbf{x} d'étiquette inconnue, on associe l'étiquette selon le côté de l'hyperplan.

L'algorithme SVM

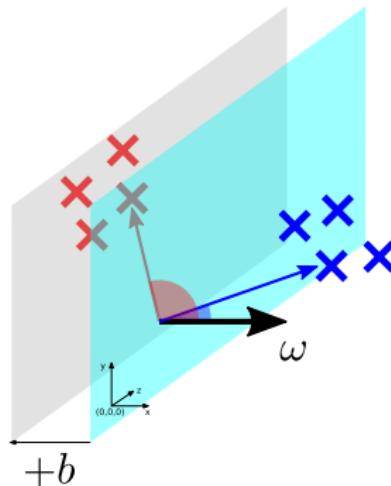
Mathématiquement: (cas binaire, $y_i \in \{+1, -1\}$)

$$\mathcal{H} = \mathcal{H}(\omega, b) = \{x \mid \omega^T x + b = 0\},$$

$$(\omega^*, b^*) = \arg \min_{\omega, b} \left\{ \|\omega\|^2 \mid \forall i, \textcolor{violet}{y}_i (\omega^T \textcolor{violet}{x}_i + b) \geq 0 \right\}.$$

Ce problème se résoud par des outils d'**optimisation convexe**.

Pour une nouvelle donnée x , l'étiquette vaut $\text{sign}(\omega^{*\top} x + b^*)$

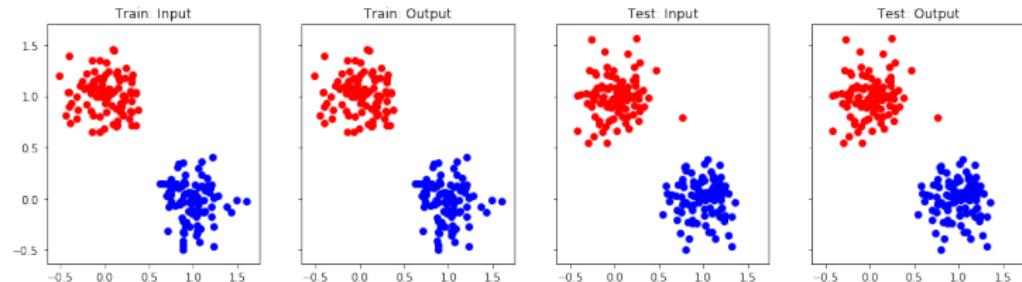


$$(a^T b = \sum_i a_i b_i = \|a\| \|b\| \cos(\angle(a, b))).$$

L'algorithme SVM

Exemple: Séparabilité linéaire:

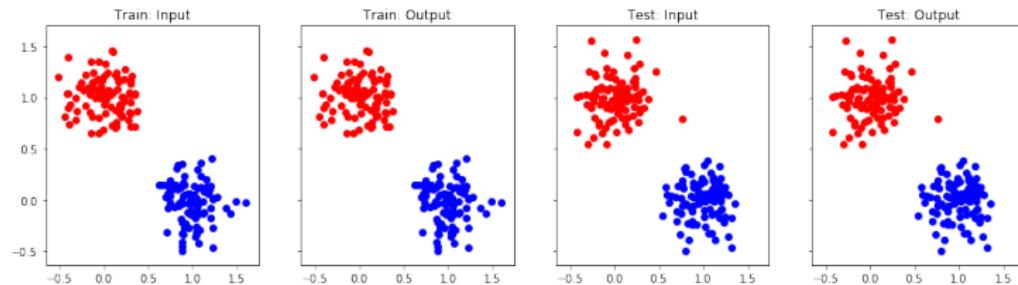
```
1 X_train = np.concatenate([X1_train,X2_train]) # built as X
2 y_train = np.concatenate([np.ones(n),-np.ones(n)])
3
4 X_test = np.concatenate([X1_test,X2_test]) # built as X
5 y_test = np.concatenate([np.ones(n_test),-np.ones(n_test)])
6
7 from sklearn import svm
8
9 classifier = svm.SVC(kernel='linear')
10 classifier.fit(X_train,y_train)
11
12 yest = classifier.predict(X_train) # prediction of training data!
13 y_est = classifier.predict(X_test) # prediction on test data!
14
15 plt.scatter(X_train[yest==1,0],X_train[yest==1,1],color='blue')      # Train: Output
16 plt.scatter(X_train[yest==-1,0],X_train[yest==-1,1],color='red')
17
18 plt.scatter(X_test[y_est==1,0],X_test[y_est==1,1],color='blue')      # Test: Output
19 plt.scatter(X_test[y_est==-1,0],X_test[y_est==-1,1],color='red')
20
```



L'algorithme SVM

Exemple: Séparabilité linéaire, les paramètres de l'hyperplan

```
1 w,b = classifier.coef_[0],classifier.intercept_
2 print('w=',w,'    b=',b)
3
```

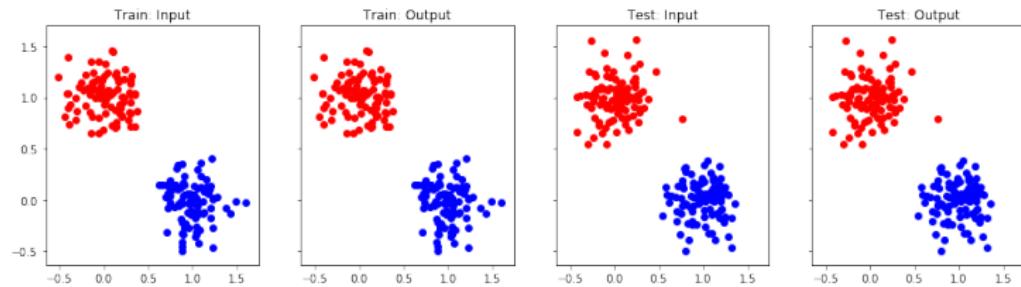


w= [1.95440136 -1.85647763] b= [-0.09901789]

L'algorithme SVM

Exemple: Séparabilité linéaire, les paramètres de l'hyperplan

```
1 w,b = classifier.coef_[0],classifier.intercept_
2 print('w=',w,' b=',b)
3
```



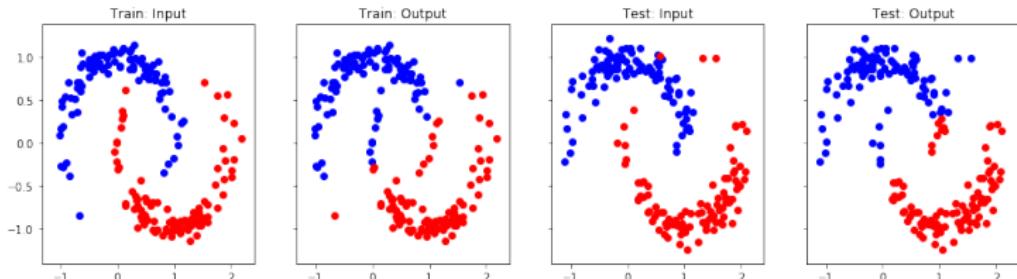
```
w= [ 1.95440136 -1.85647763] b= [-0.09901789]
```

w est le vecteur normal (orthogonal) à l'hyperplan: l'hyperplan pointe donc dans la direction (1.85, 1.95) avec pour ordonnée à l'origine -0.09

L'algorithme SVM

Exemple: Non séparabilité linéaire:

```
1 X_train = np.concatenate([X1_train,X2_train]) # built as X
2 y_train = np.concatenate([np.ones(n),-np.ones(n)])
3
4 X_test = np.concatenate([X1_test,X2_test]) # built as X
5 y_test = np.concatenate([np.ones(n_test),-np.ones(n_test)])
6
7 from sklearn import svm
8
9 classifier = svm.SVC(kernel='linear')
10 classifier.fit(X_train,y_train)
11
12 yest = classifier.predict(X_train) # prediction of training data!
13 y_est = classifier.predict(X_test) # prediction on test data!
14
15 plt.scatter(X_train[yest==1,0],X_train[yest==1,1],color='blue')      # Train: Output
16 plt.scatter(X_train[yest==-1,0],X_train[yest==-1,1],color='red')
17
18 plt.scatter(X_test[y_est==1,0],X_test[y_est==1,1],color='blue')      # Test: Output
19 plt.scatter(X_test[y_est==-1,0],X_test[y_est==-1,1],color='red')
20
```



L'algorithme SVM

Exemple: Non séparabilité linéaire:

- Correction par linéarisation des données

L'algorithme SVM

Exemple: Non séparabilité linéaire:

- Correction par linéarisation des données

$$\forall i, \quad \varphi_i = \varphi(\textcolor{violet}{x}_i) \in \mathbb{R}^{p'}$$

(p' peut être différent de p) ou sous forme matricielle

$$\Phi = \begin{bmatrix} \varphi(\textcolor{violet}{x}_1) \\ \vdots \\ \varphi(\textcolor{violet}{x}_n) \end{bmatrix} \in \mathbb{R}^{n \times p'}$$

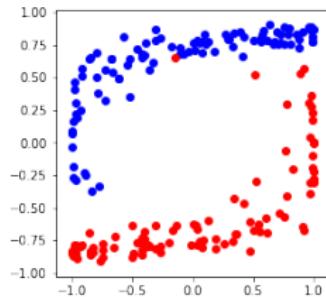
L'algorithme SVM

- Mais comment choisir φ ?
 - ▶ soit intelligemment (structure des données): **approche “ingénieur”**

L'algorithme SVM

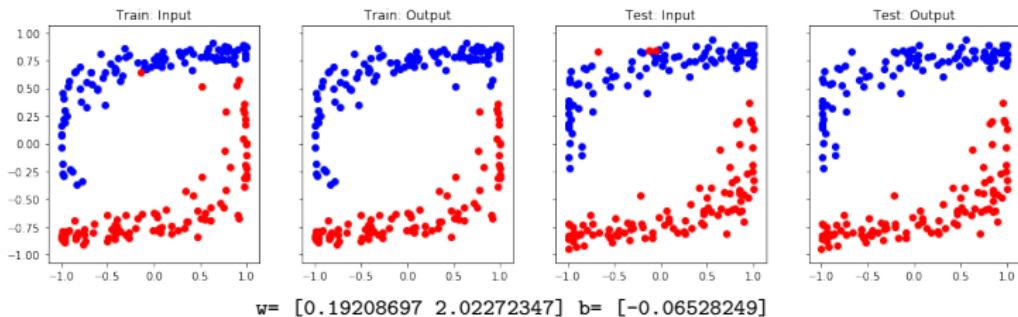
- Mais comment choisir φ ?
 - ▶ soit intelligemment (structure des données): **approche “ingénieur”**

```
1 phi = lambda x:np.array([np.cos(3*x[0]),np.sin(x[1])])
2 Phi_train = phi(X_train.T).T
3 Phi_test = phi(X_test.T).T
4
5 plt.scatter(Phi_train[y==1,0],Phi_train[y==1,1],color='blue')
6 plt.scatter(Phi_train[y==-1,0],Phi_train[y==-1,1],color='red')
7
```



L'algorithme SVM

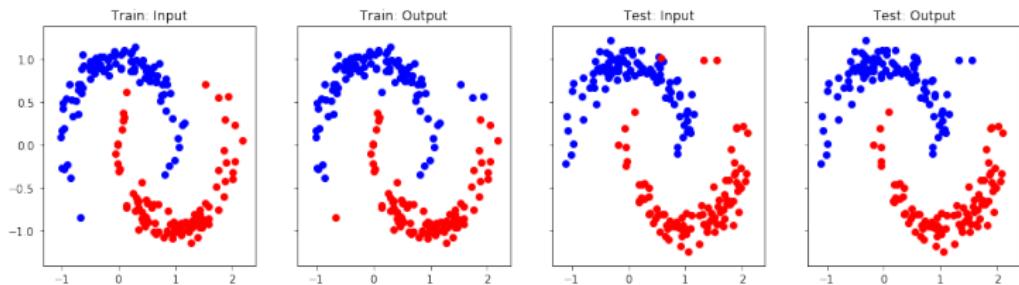
```
1 classifier = svm.SVC(kernel='linear')
2 classifier.fit(Phi_train,y)
3
4 yest  = classifier.predict(Phi_train)      # prediction of training data
5 y_est = classifier.predict(Phi_test) # prediction of test data
6
7 plt.scatter(Phi_train[yest==1,0],Phi_train[yest==1,1],color='blue')
8 plt.scatter(Phi_train[yest== -1,0],Phi_train[yest== -1,1],color='red')
9
10 plt.scatter(Phi_test[y_est==1,0],Phi_test[y_est==1,1],color='blue')
11 plt.scatter(Phi_test[y_est== -1,0],Phi_test[y_est== -1,1],color='red')
12
13 w,b = classifier.coef_[0],classifier.intercept_
14 print('w=',w,'    b=',b)
15
```



L'algorithme SVM

Ou, dans l'espace originel des x_i :

```
1 plt.scatter(X_train[yest==1,0],X_train[yest==1,1],color='blue')
2 plt.scatter(X_train[yest== -1,0],X_train[yest== -1,1],color='red')
3
4 plt.scatter(X_test[y_est==1,0],X_test[y_est==1,1],color='blue')
5 plt.scatter(X_test[y_est== -1,0],X_test[y_est== -1,1],color='red')
6
```



L'algorithme SVM

- Mais comment choisir φ ?
 - ▶ soit aléatoirement avec p' grand: représentations (noyaux) aléatoires

L'algorithme SVM

- Mais comment choisir φ ?

- ▶ soit aléatoirement avec p' grand: **représentations (noyaux) aléatoires**
- ▶ exemple très populaire du **noyau gaussien (ou rbf)**

$$\varphi_i = \varphi(\mathbf{x}_i) = [\cos(2\pi w_1^T \mathbf{x}_i), \cos(2\pi w_2^T \mathbf{x}_i), \dots] \in \mathbb{R}^{\infty}$$

avec $w_1, w_2, \dots \in \mathbb{R}^p$ suite aléatoire infinie!

L'algorithme SVM

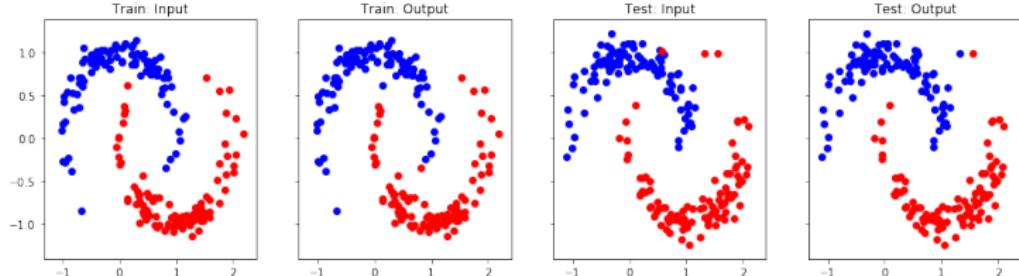
- Mais comment choisir φ ?

- ▶ soit aléatoirement avec p' grand: **représentations (noyaux) aléatoires**
- ▶ exemple très populaire du **noyau gaussien (ou rbf)**

$$\varphi_i = \varphi(\mathbf{x}_i) = [\cos(2\pi w_1^T \mathbf{x}_i), \cos(2\pi w_2^T \mathbf{x}_i), \dots] \in \mathbb{R}^\infty$$

avec $w_1, w_2, \dots \in \mathbb{R}^p$ suite aléatoire infinie!

```
1 classifier = svm.SVC(kernel='rbf')
2 classifier.fit(X_train, y_train)
3
4 yest = classifier.predict(X_train)      # prediction of training data
5 y_est = classifier.predict(X_test) # prediction of test data
6
7 plt.scatter(X_train[yest==1,0],X_train[yest==1,1],color='blue')
8 plt.scatter(X_train[yest== -1,0],X_train[yest== -1,1],color='red')
9
10 plt.scatter(X_test[y_est==1,0],X_test[y_est==1,1],color='blue')
11 plt.scatter(X_test[y_est== -1,0],X_test[y_est== -1,1],color='red')
12
```



- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
 - Des données aux vecteurs
 - Approche par séparabilité linéaire: l'algorithme SVM
 - Approche par distances dans l'espace des représentations:
l'algorithme kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques

Notion de distances and kNN

Limitations des approches par hyperplan:

- impose de **forcer la séparabilité linéaire**

Notion de distances and kNN

Limitations des approches par hyperplan:

- impose de **forcer la séparabilité linéaire**
- difficile en manuel, pas clair en aléatoire!

Notion de distances and kNN

Limitations des approches par hyperplan:

- impose de **forcer la séparabilité linéaire**
- difficile en manuel, pas clair en aléatoire!

L'alternative de la distance:

- **Idée de base:** associer chaque nouveau point à ses **plus proches voisins étiquetés**

Notion de distances and kNN

Limitations des approches par hyperplan:

- impose de **forcer la séparabilité linéaire**
- difficile en manuel, pas clair en aléatoire!

L'alternative de la distance:

- **Idée de base:** associer chaque nouveau point à ses **plus proches voisins étiquetés**
- **L'algorithme des k plus proches voisins kNN (k -nearest neighbors)** implémente:

Etiquette \hat{y} de \hat{x} = Etiquette majoritaire des k plus proches x_i de \hat{x} .

Notion de distances and kNN

Limitations des approches par hyperplan:

- impose de **forcer la séparabilité linéaire**
- difficile en manuel, pas clair en aléatoire!

L'alternative de la distance:

- **Idée de base:** associer chaque nouveau point à ses **plus proches voisins étiquetés**
- **L'algorithme des k plus proches voisins kNN (k -nearest neighbors)** implémente:

Etiquette \hat{y} de \hat{x} = Etiquette majoritaire des k plus proches x_i de \hat{x} .

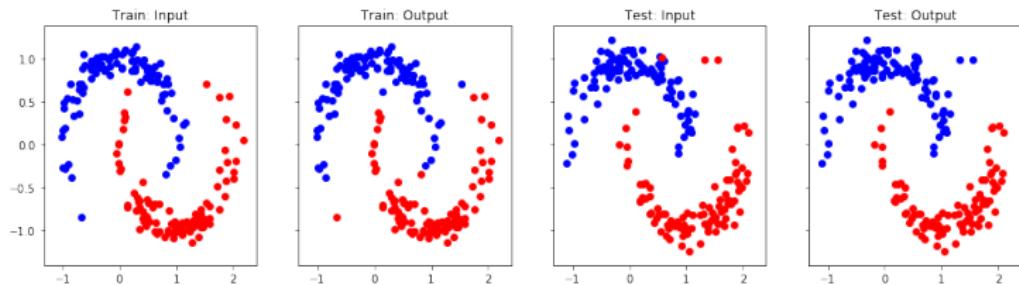
- **Mathématiquement:** pour $k = 1$ (plus proche voisin)

$$\hat{y} = \arg \min_{y_i} \{\|x_i - \hat{x}\|\}.$$

Notion de distances et kNN

Implémentation en Python:

```
1 from sklearn.neighbors import KNeighborsClassifier  
2 classifier = KNeighborsClassifier(n_neighbors=11)  
3 classifier.fit(X_train,y_train)  
4  
5 yest = classifier.predict(X_train)      # prediction of training data  
6 y_est = classifier.predict(X_test) # prediction of test data  
7  
8 plt.scatter(X_train[yest==1,0],X_train[yest==1,1],color='blue')  
9 plt.scatter(X_train[yest==-1,0],X_train[yest==-1,1],color='red')  
10  
11 plt.scatter(X_test[y_est==1,0],X_test[y_est==1,1],color='blue')  
12 plt.scatter(X_test[y_est==-1,0],X_test[y_est==-1,1],color='red')  
13
```



Notion de distances et kNN

De multiples avantages:

- évite de chercher une représentation “linéaire” compliquée

Notion de distances et kNN

De multiples avantages:

- évite de chercher une représentation “linéaire” compliquée
- intuition claire et simple

Notion de distances et kNN

De multiples avantages:

- évite de chercher une représentation “linéaire” compliquée
- intuition claire et simple

Malheureusement, de grosses limites!:

- quand la dimension p augmente (mais le nombre de données n est constant), les vecteurs x_i s'éloignent dans l'espace \mathbb{R}^p !

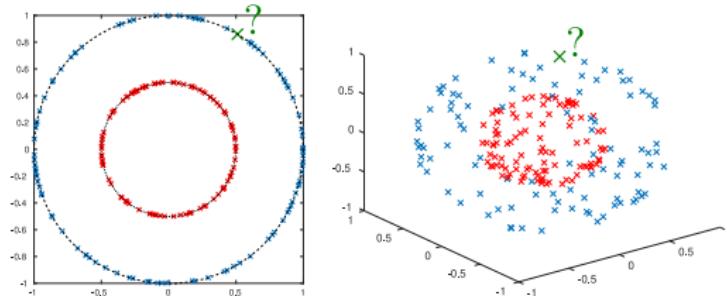
Notion de distances et kNN

De multiples avantages:

- évite de chercher une représentation “linéaire” compliquée
- intuition claire et simple

Malheureusement, de grosses limites!:

- quand la dimension p augmente (mais le nombre de données n est constant), les vecteurs x_i s'éloignent dans l'espace \mathbb{R}^p !
- conséquence majeure: le plus proche voisin dans \mathbb{R}^p n'est plus dans la bonne classe quand $p \rightarrow \infty$



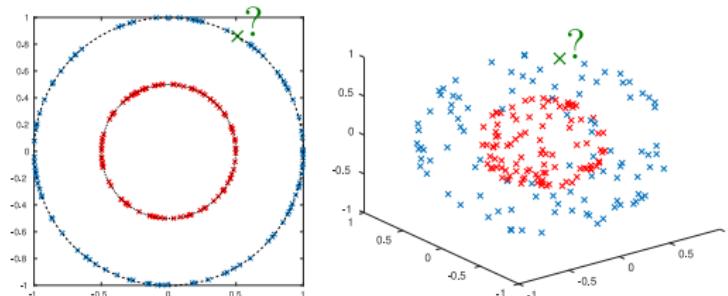
Notion de distances et kNN

De multiples avantages:

- évite de chercher une représentation “linéaire” compliquée
- intuition claire et simple

Malheureusement, de grosses limites!:

- quand la dimension p augmente (mais le nombre de données n est constant), les vecteurs x_i s'éloignent dans l'espace \mathbb{R}^p !
- conséquence majeure: le plus proche voisin dans \mathbb{R}^p n'est plus dans la bonne classe quand $p \rightarrow \infty$



- de plus, coût calculatoire explose pour n, p grands: **inutilisable sur les données modernes.**

Résumé: séparation, distance... SVM ou kNN ?

kNN et algorithmes basés sur les distances:

- adapté aux **données de petites dimensions, nombreuses mais pas trop!**
(denses en 2D, 3D)

Résumé: séparation, distance... SVM ou kNN ?

kNN et algorithmes basés sur les distances:

- adapté aux **données de petites dimensions, nombreuses mais pas trop!**
(denses en 2D, 3D)
- très simple et pratique sur des ensembles de **classes de vecteurs "connexes"**

Résumé: séparation, distance... SVM ou kNN ?

kNN et algorithmes basés sur les distances:

- adapté aux **données de petites dimensions, nombreuses mais pas trop!** (denses en 2D, 3D)
- très simple et pratique sur des ensembles de **classes de vecteurs "connexes"**
- peu utilisable pour des données modernes (grandes images, représentations de textes): alternative = réduire les dimensions (**compression**), la taille de la base de données (**échantillonnage**).

Résumé: séparation, distance... SVM ou kNN ?

kNN et algorithmes basés sur les distances:

- adapté aux **données de petites dimensions, nombreuses mais pas trop!** (denses en 2D, 3D)
- très simple et pratique sur des ensembles de **classes de vecteurs "connexes"**
- peu utilisable pour des données modernes (grandes images, représentations de textes): alternative = réduire les dimensions (**compression**), la taille de la base de données (**échantillonnage**).

SVM et algorithmes “linéariseurs” :

- plutôt flexible sur les dimensions des données (compression par noyaux, bon passage à l'échelle)

Résumé: séparation, distance... SVM ou kNN ?

kNN et algorithmes basés sur les distances:

- adapté aux **données de petites dimensions, nombreuses mais pas trop!** (denses en 2D, 3D)
- très simple et pratique sur des ensembles de **classes de vecteurs "connexes"**
- peu utilisable pour des données modernes (grandes images, représentations de textes): alternative = réduire les dimensions (**compression**), la taille de la base de données (**échantillonnage**).

SVM et algorithmes “linéariseurs” :

- plutôt flexible sur les dimensions des données (compression par noyaux, bon passage à l'échelle)
- **puissance des noyaux**: créent implicitement des représentants riches de grandes dimensions (parfois infinie!)

Résumé: séparation, distance... SVM ou kNN ?

kNN et algorithmes basés sur les distances:

- adapté aux **données de petites dimensions, nombreuses mais pas trop!** (denses en 2D, 3D)
- très simple et pratique sur des ensembles de **classes de vecteurs "connexes"**
- peu utilisable pour des données modernes (grandes images, représentations de textes): alternative = réduire les dimensions (**compression**), la taille de la base de données (**échantillonnage**).

SVM et algorithmes "linéariseurs":

- plutôt flexible sur les dimensions des données (compression par noyaux, bon passage à l'échelle)
- **puissance des noyaux**: créent implicitement des représentants riches de grandes dimensions (parfois infinie!)
- parfois difficile à manipuler, notamment en petite dimension.

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
 - Notions de base et garanties théoriques
 - Le perceptron multi-couche (MLP)
- 5 L'IA moderne: révolution informatique et retour des mathématiques

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
 - Notions de base et garanties théoriques
 - Le perceptron multi-couche (MLP)
- 5 L'IA moderne: révolution informatique et retour des mathématiques

Idée générale

Rappel:

- pour les données $x \in \mathbb{R}^p$ modernes (p grand),

Représentation puissante de x : $\varphi = \varphi(x)$ linéairement séparable

Idée générale

Rappel:

- pour les données $x \in \mathbb{R}^p$ modernes (p grand),
Représentation puissante de x : $\varphi = \varphi(x)$ linéairement séparable
- solution des SVMs restreinte (représentations manuelles ou noyaux aléatoires arbitraires)

Idée générale

Rappel:

- pour les données $x \in \mathbb{R}^p$ modernes (p grand),
Représentation puissante de x : $\varphi = \varphi(x)$ linéairement séparable
- solution des SVMs restreinte (représentations manuelles ou noyaux aléatoires arbitraires)
- **problèmes majeurs:**
 - ▶ représentations des données modernes **trop complexes à gérer manuellement** (même par ingénierie fine)
 - ▶ noyaux aléatoires **fixes, indépendants des données!**, donc faibles

Idée générale

Rappel:

- pour les données $x \in \mathbb{R}^p$ modernes (p grand),
Représentation puissante de x : $\varphi = \varphi(x)$ linéairement séparable
- solution des SVMs restreinte (représentations manuelles ou noyaux aléatoires arbitraires)
- **problèmes majeurs:**
 - ▶ représentations des données modernes **trop complexes à gérer manuellement** (même par ingénierie fine)
 - ▶ noyaux aléatoires **fixes, indépendants des données!**, donc faibles

Solution possible: les réseaux de neurones

Idée générale

Rappel:

- pour les données $x \in \mathbb{R}^p$ modernes (p grand),
Représentation puissante de x : $\varphi = \varphi(x)$ linéairement séparable
- solution des SVMs restreinte (représentations manuelles ou noyaux aléatoires arbitraires)
- **problèmes majeurs:**
 - ▶ représentations des données modernes **trop complexes à gérer manuellement** (même par ingénierie fine)
 - ▶ noyaux aléatoires **fixes, indépendants des données!**, donc faibles

Solution possible: les réseaux de neurones

- **apprennent automatiquement** une représentation sur la base des données d'apprentissage

Idée générale

Rappel:

- pour les données $x \in \mathbb{R}^p$ modernes (p grand),
Représentation puissante de x : $\varphi = \varphi(x)$ linéairement séparable
- solution des SVMs restreinte (représentations manuelles ou noyaux aléatoires arbitraires)
- **problèmes majeurs:**
 - ▶ représentations des données modernes **trop complexes à gérer manuellement** (même par ingénierie fine)
 - ▶ noyaux aléatoires **fixes, indépendants des données!**, donc faibles

Solution possible: les réseaux de neurones

- **apprennent automatiquement** une représentation sur la base des données d'apprentissage
- basés sur un concept très simple, mais à forte garantie mathématique!

Idée générale

Rappel:

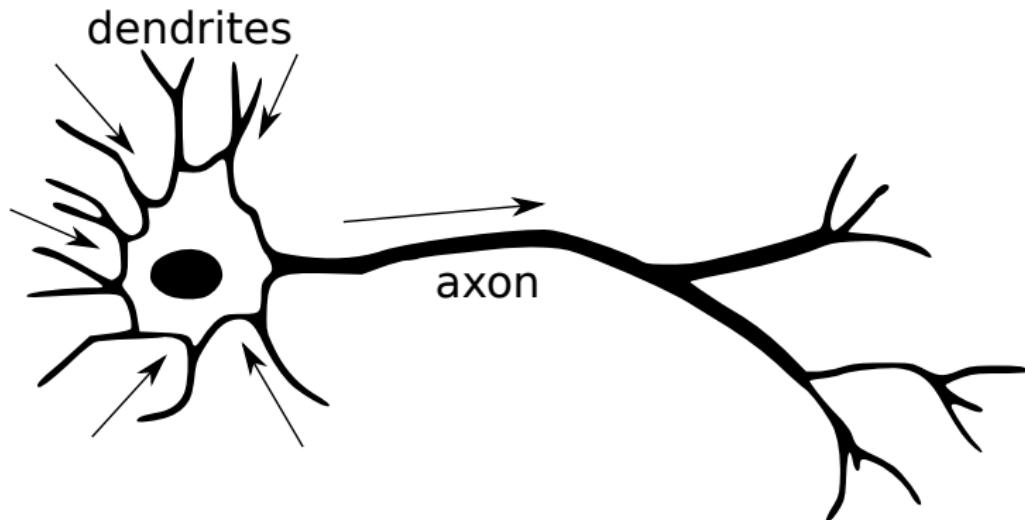
- pour les données $x \in \mathbb{R}^p$ modernes (p grand),
Représentation puissante de x : $\varphi = \varphi(x)$ linéairement séparable
- solution des SVMs restreinte (représentations manuelles ou noyaux aléatoires arbitraires)
- **problèmes majeurs:**
 - ▶ représentations des données modernes **trop complexes à gérer manuellement** (même par ingénierie fine)
 - ▶ noyaux aléatoires **fixes, indépendants des données!**, donc faibles

Solution possible: les réseaux de neurones

- **apprennent automatiquement** une représentation sur la base des données d'apprentissage
- basés sur un concept très simple, mais à forte garantie mathématique!
- grande (trop grande?) flexibilité et peu coûteux à utiliser (**mais peuvent être coûteux à entraîner!**)

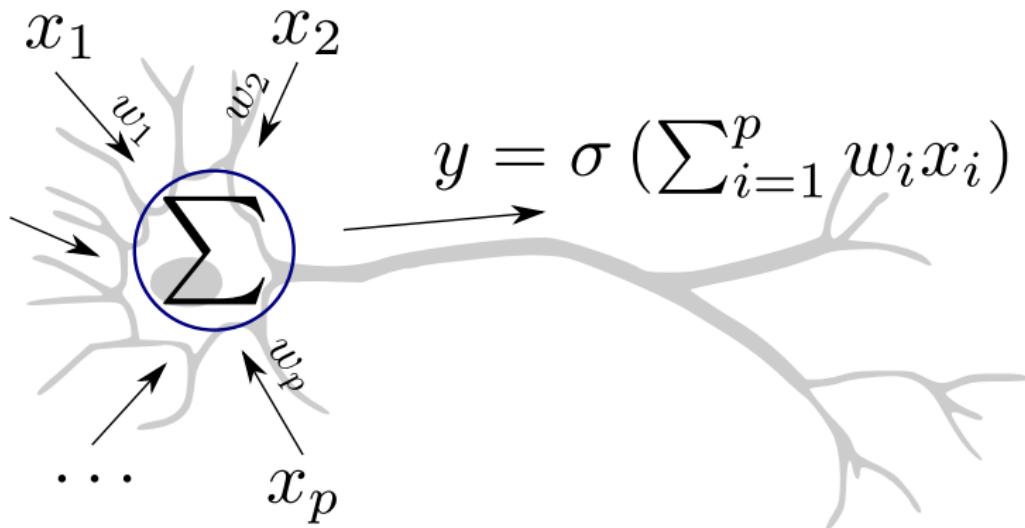
Intuition biologique

Le signal des *dendrites* dépolarise l'axone **si et seulement si** la somme des signaux **dépasse un seuil**.



Intuition biologique

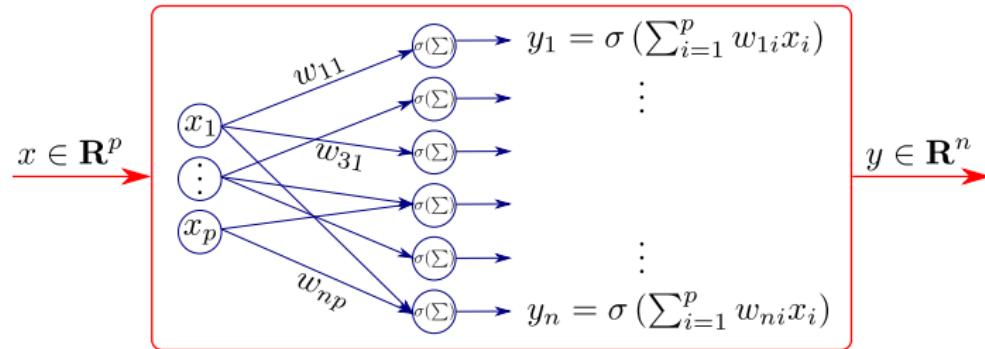
Le signal des *dendrites* dépolarise l'*axone* **si et seulement si** la somme des signaux **dépasse un seuil**.



$$\sigma(t) = \begin{cases} 0, & t < T \\ 1, & t \geq T \end{cases} \quad \text{ou} \quad \sigma(t) = \begin{cases} 0, & t < T \\ t, & t \geq T \end{cases} \quad \text{ou} \quad \sigma(t) = \tanh(t) \dots$$

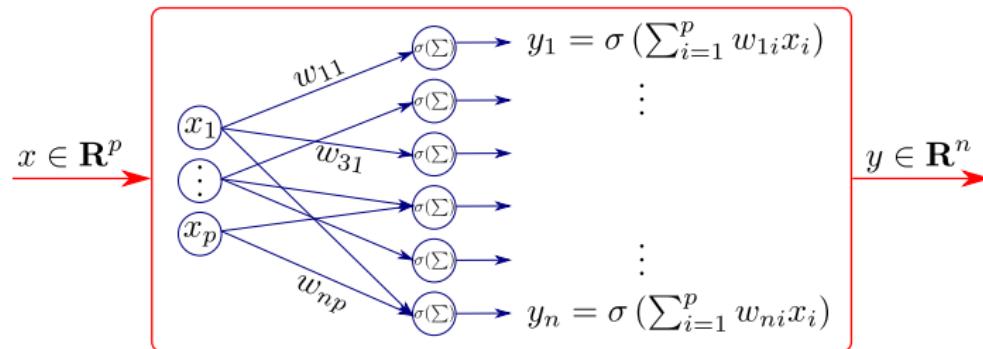
Le modèle du “perceptron”

Généralisation à plusieurs neurones



Le modèle du “perceptron”

Généralisation à plusieurs neurones



Définition [le perceptron à n neurones de McCulloch et Pitts (1943)]

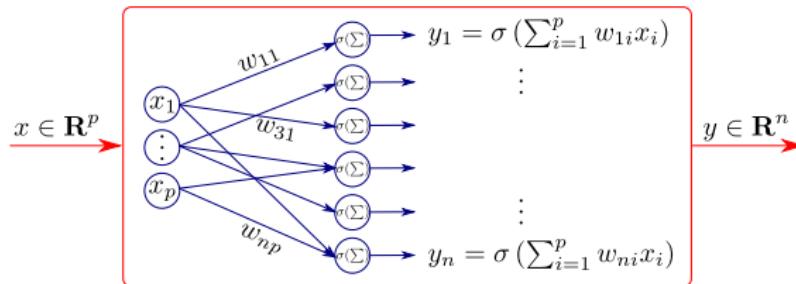
Pour des **vecteurs de poids** $w_1, \dots, w_n \in \mathbb{R}^p$ et une fonction $\sigma : \mathbb{R} \rightarrow \mathbb{R}$,

$$y_k = \sigma(w_k^\top x) = \sigma \left(\sum_{i=1}^p w_{ki} x_i \right).$$

(ici on a défini, pour $a, b \in \mathbb{R}^p$, le **produit scalaire** $a^\top b = \sum_{i=1}^p a_i b_i$)

Le modèle du “perceptron”

Représentation matricielle:

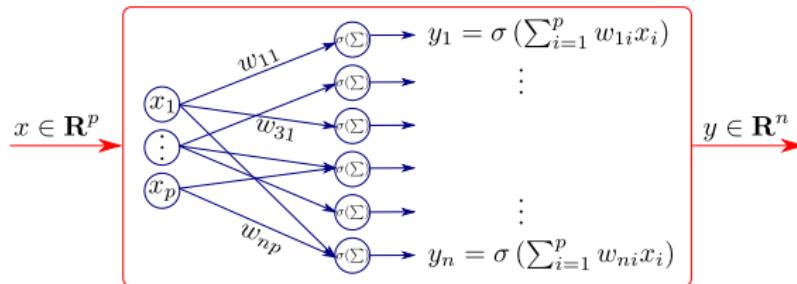


Exemple du cas 3×2 : on a la notation (d'algèbre linéaire) suivante

$$\begin{aligned} z_1 &= w_{11}x_1 + w_{12}x_2 \\ z_2 &= w_{21}x_1 + w_{22}x_2 \\ z_3 &= w_{31}x_1 + w_{32}x_2 \end{aligned} \Leftrightarrow \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}}_z = \underbrace{\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}}_W \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_x \Leftrightarrow z = Wx$$

Le modèle du “perceptron”

Représentation matricielle:



Exemple du cas 3×2 : on a la notation (d'algèbre linéaire) suivante

$$\begin{aligned} z_1 &= w_{11}x_1 + w_{12}x_2 \\ z_2 &= w_{21}x_1 + w_{22}x_2 \\ z_3 &= w_{31}x_1 + w_{32}x_2 \end{aligned} \Leftrightarrow \underbrace{\begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix}}_z = \underbrace{\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix}}_W \underbrace{\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}}_x \Leftrightarrow z = Wx$$

et finalement

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \sigma(z_1) \\ \sigma(z_2) \\ \sigma(z_3) \end{bmatrix} \equiv \sigma(z) = \sigma(Wx)$$

Le modèle du “perceptron”

Conventions d'appellations:

- x est le vecteur de la **couche d'entrée**
- y est le vecteur de la **couche de sortie**
- W est la **matrice de connexion** de la couche de neurones cachée
- $\sigma(\cdot)$ est la **fonction d'activation**

Le modèle du “perceptron”

Conventions d'appellations:

- x est le vecteur de la **couche d'entrée**
- y est le vecteur de la **couche de sortie**
- W est la **matrice de connexion** de la couche de neurones cachée
- $\sigma(\cdot)$ est la **fonction d'activation**

Connexion avec les **représentations non linéaires**:

- pour une donnée $x \in \mathbb{R}^p$, si on définit

$$\varphi : \mathbb{R}^p \rightarrow \mathbb{R}^{p'}$$

$$x \mapsto \varphi(x) = \sigma(Wx)$$

on obtient une **représentation non-linéaire de x** .

Le modèle du “perceptron”

Conventions d'appellations:

- x est le vecteur de la **couche d'entrée**
- y est le vecteur de la **couche de sortie**
- W est la **matrice de connexion** de la couche de neurones cachée
- $\sigma(\cdot)$ est la **fonction d'activation**

Connexion avec les **représentations non linéaires**:

- pour une donnée $x \in \mathbb{R}^p$, si on définit

$$\varphi : \mathbb{R}^p \rightarrow \mathbb{R}^{p'}$$

$$x \mapsto \varphi(x) = \sigma(Wx)$$

on obtient une **représentation non-linéaire de x** .

- nombreux degrés de libertés:

- ▶ la taille p' de la représentation $\varphi(x)$
- ▶ les $p' \times p$ éléments de la matrice de connexion $W \in \mathbb{R}^{p' \times p}$
- ▶ la fonction d'activation $\sigma(\cdot)$.

Le modèle du “perceptron”

Résultat fondateur: puissantes garanties théoriques

Theorem (Approximation universelle)

Si $\sigma(\cdot)$ est une **sigmoïde** (fonction en “S”), alors toute fonction (mesurable compacte)

$$\begin{aligned} f : \mathbb{R}^p &\rightarrow \mathbb{R} \\ x &\mapsto f(x) \end{aligned}$$

peut être approximée **arbitrairement près** par une fonction g de la famille

$$\mathcal{G} \equiv \left\{ g : x \mapsto a^T \sigma(Wx + b), \ a, b \in \mathbb{R}^{p'}, \ W \in \mathbb{R}^{p' \times p}, \ p' \in \mathbb{N} \right\}$$

Le modèle du “perceptron”

Résultat fondateur: puissantes garanties théoriques

Theorem (Approximation universelle)

Si $\sigma(\cdot)$ est une **sigmoïde** (fonction en “S”), alors toute fonction (mesurable compacte)

$$\begin{aligned} f : \mathbb{R}^p &\rightarrow \mathbb{R} \\ x &\mapsto f(x) \end{aligned}$$

peut être approximée **arbitrairement près** par une fonction g de la famille

$$\mathcal{G} \equiv \left\{ g : x \mapsto a^T \sigma(Wx + b), \ a, b \in \mathbb{R}^{p'}, \ W \in \mathbb{R}^{p' \times p}, \ p' \in \mathbb{N} \right\}$$

À quoi cela sert-il ?

Le modèle du “perceptron”

Approximation universelle:

- si x est l'ensemble des p pixels d'une photo et f est la fonction (très compliquée!)

$$f : \mathbb{R}^p \rightarrow \{+1, -1\}$$

$$x \in \{\text{chien}\} \mapsto +1$$

$$x \in \{\text{chat}\} \mapsto -1$$

Le modèle du “perceptron”

Approximation universelle:

- si x est l'ensemble des p pixels d'une photo et f est la fonction (très compliquée!)

$$f : \mathbb{R}^p \rightarrow \{+1, -1\}$$

$$x \in \{\text{chien}\} \mapsto +1$$

$$x \in \{\text{chat}\} \mapsto -1$$

alors, il existe un **perceptron** $x \mapsto \varphi(x) = \sigma(Wx)$ et un vecteur a tels que

$$f(x) \simeq g(x) = a^T \varphi(x) = a^T \sigma(Wx).$$

Le modèle du “perceptron”

Approximation universelle:

- si x est l'ensemble des p pixels d'une photo et f est la fonction (très compliquée!)

$$f : \mathbb{R}^p \rightarrow \{+1, -1\}$$

$$x \in \{\text{chien}\} \mapsto +1$$

$$x \in \{\text{chat}\} \mapsto -1$$

alors, il existe un **perceptron** $x \mapsto \varphi(x) = \sigma(Wx)$ et un vecteur a tels que

$$f(x) \simeq g(x) = a^T \varphi(x) = a^T \sigma(Wx).$$

- le perceptron permet de réaliser des classifications potentiellement dures! et c'est **mathématiquement prouvé!** (magnifique preuve!)

Le modèle du “perceptron”

Approximation universelle:

- si x est l'ensemble des p pixels d'une photo et f est la fonction (très compliquée!)

$$f : \mathbb{R}^p \rightarrow \{+1, -1\}$$

$$x \in \{\text{chien}\} \mapsto +1$$

$$x \in \{\text{chat}\} \mapsto -1$$

alors, il existe un **perceptron** $x \mapsto \varphi(x) = \sigma(Wx)$ et un vecteur a tels que

$$f(x) \simeq g(x) = a^T \varphi(x) = a^T \sigma(Wx).$$

- le perceptron permet de réaliser des classifications potentiellement dures! et c'est **mathématiquement prouvé!** (magnifique preuve!)
- **mais...** le théorème n'est pas *constructif*: il ne dit pas comment construire le réseau!

Le modèle du “perceptron”

Approximation universelle:

- si x est l'ensemble des p pixels d'une photo et f est la fonction (très compliquée!)

$$f : \mathbb{R}^p \rightarrow \{+1, -1\}$$

$$x \in \{\text{chien}\} \mapsto +1$$

$$x \in \{\text{chat}\} \mapsto -1$$

alors, il existe un **perceptron** $x \mapsto \varphi(x) = \sigma(Wx)$ et un vecteur a tels que

$$f(x) \simeq g(x) = a^T \varphi(x) = a^T \sigma(Wx).$$

- le perceptron permet de réaliser des classifications potentiellement dures! et c'est **mathématiquement prouvé!** (magnifique preuve!)
- **mais...** le théorème n'est pas *constructif*: il ne dit pas comment construire le réseau!
- **difficulté majeure**: trouver p' , a et surtout W .

Le modèle du “perceptron”

Idée de construction:

- on commence par fixer un p' et un σ , et on laisse W, a libres

Le modèle du “perceptron”

Idée de construction:

- on commence par fixer un p' et un σ , et on laisse W, a libres
- on va alors **entrainer le réseau** à apprendre W, a à partir d'échantillons d'une base de donnée connue

$$(X, y) = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

Le modèle du “perceptron”

Idée de construction:

- on commence par fixer un p' et un σ , et on laisse W, a libres
- on va alors entrainer le réseau à apprendre W, a à partir d'échantillons d'une base de donnée connue

$$(X, y) = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

- on obtient alors pour chaque échantillon d'entraînement

$$y_i = f(x_i) \simeq a^T \sigma(W x_i)$$

Le modèle du “perceptron”

Idée de construction:

- on commence par fixer un p' et un σ , et on laisse W, a libres
- on va alors entrainer le réseau à apprendre W, a à partir d'échantillons d'une base de donnée connue

$$(X, y) = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$$

- on obtient alors pour chaque échantillon d'entraînement

$$y_i = f(x_i) \simeq a^T \sigma(W x_i)$$

- et on espère (mais on ne sait a priori pas!) que, pour tout couple arbitraire (\hat{x}, \hat{y}) ,

$$\hat{y} = f(\hat{x}) \simeq a^T \sigma(W \hat{x})$$

Le modèle du “perceptron”

Méthode d'apprentissage:

- il suffit de trouver W, a qui **minimise le coût**

$$\text{error}(W) \equiv \text{error}(W; X, y) = \sum_{i=1}^n \left| f(x_i) - a^\top \sigma(W x_i) \right|^2$$

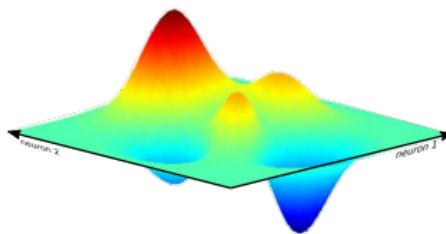
Le modèle du “perceptron”

Méthode d'apprentissage:

- il suffit de trouver W, a qui **minimise le coût**

$$\text{error}(\mathbf{W}) \equiv \text{error}(\mathbf{W}; \mathbf{X}, \mathbf{y}) = \sum_{i=1}^n \left| f(\mathbf{x}_i) - \mathbf{a}^\top \sigma(\mathbf{W} \mathbf{x}_i) \right|^2$$

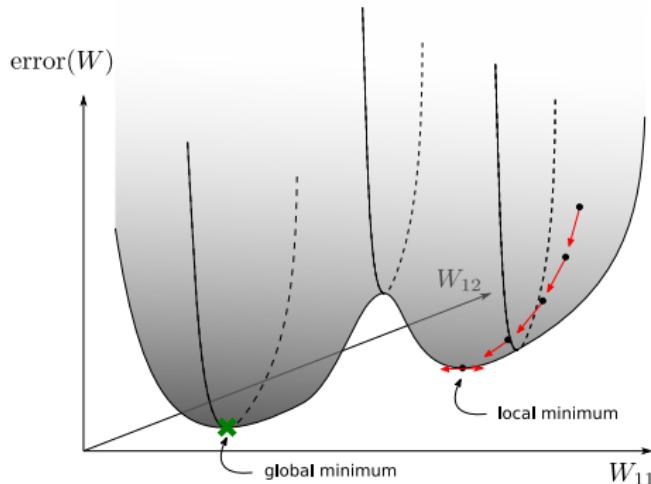
- mais beaucoup d'embûches (déjà en 1950 et toujours aujourd'hui!)
 - ▶ problème d'optimisation **non convexe**



conduit à des **solutions sous-optimales**, voire mauvaises!

Le modèle du “perceptron”

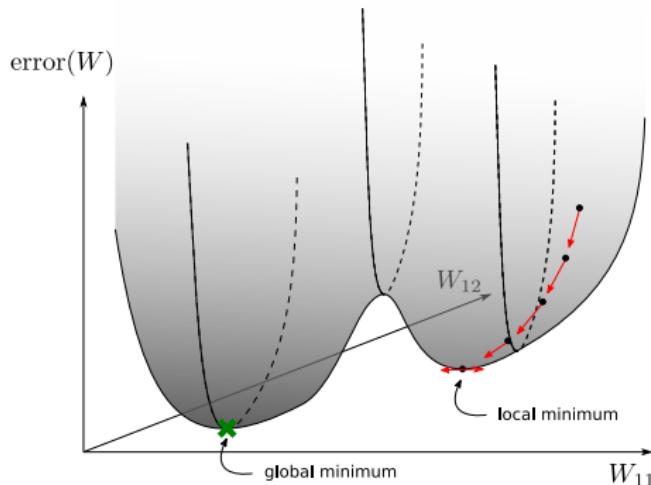
Optimisation par descente de gradient:



- consiste à partir d'un point $(W_{11}, W_{12}, W_{13}, \dots)$ au hasard et à “descendre” en suivant la pente (le gradient) de la fonction erreur.

Le modèle du “perceptron”

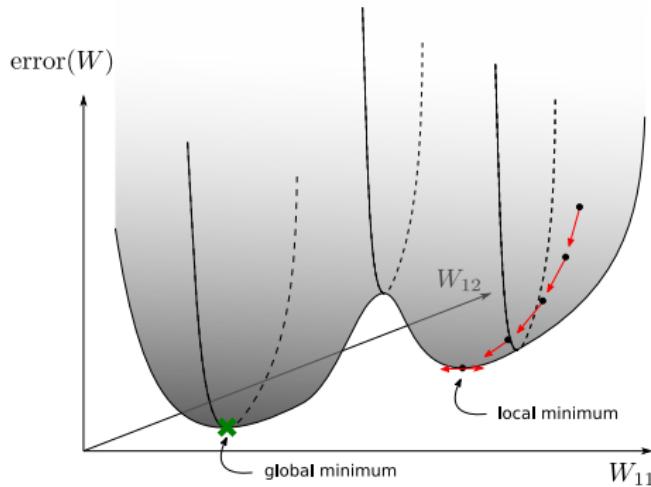
Optimisation par descente de gradient:



- consiste à partir d'un point $(W_{11}, W_{12}, W_{13}, \dots)$ au hasard et à “descendre” en suivant la pente (le gradient) de la fonction erreur.
- pas simple: il faut contrôler la taille des “sauts” et la condition d'arrêt

Le modèle du “perceptron”

Optimisation par descente de gradient:



- consiste à partir d'un point ($W_{11}, W_{12}, W_{13}, \dots$) au hasard et à “descendre” en suivant la pente (le gradient) de la fonction erreur.
- pas simple: il faut contrôler la taille des “sauts” et la condition d'arrêt
- **risques:** ne jamais s'arrêter, remonter trop haut, changer de vallée, être trop lent et ne pas progresser...

Le modèle du “perceptron”

Remarques:

- pour être précis, il faut énormément de données étiquetées (x_i, y_i)

Le modèle du “perceptron”

Remarques:

- pour être précis, il faut énormément de données étiquetées (x_i, y_i)
- compromis délicat entre “beaucoup de neurones” pour construire des fonctions élaborées et “peu de neurones” pour éviter le problème du **surapprentissage**:

Le modèle du “perceptron”

Remarques:

- pour être précis, il faut énormément de données étiquetées (x_i, y_i)
- compromis délicat entre “beaucoup de neurones” pour construire des fonctions élaborées et “peu de neurones” pour éviter le problème du **surapprentissage**:
 - ▶ **surapprentissage** = algorithme très précis sur (x_i, y_i) de la base d’entraînement $(X_{\text{train}}, y_{\text{train}})$ mais mauvais sur des nouvelles données (\hat{x}, \hat{y})

Le modèle du “perceptron”

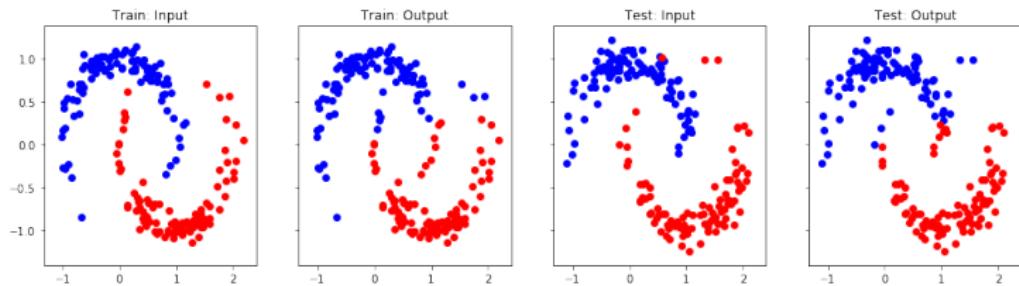
Remarques:

- pour être précis, il faut énormément de données étiquetées (x_i, y_i)
- compromis délicat entre “beaucoup de neurones” pour construire des fonctions élaborées et “peu de neurones” pour éviter le problème du **surapprentissage**:
 - ▶ **surapprentissage** = algorithme très précis sur (x_i, y_i) de la base d’entraînement $(X_{\text{train}}, y_{\text{train}})$ mais mauvais sur des nouvelles données (\hat{x}, \hat{y})
 - ▶ d’où l’intérêt de toujours avoir une base de test $(X_{\text{test}}, y_{\text{test}})$

Le modèle du “perceptron”

Implémentation Python: $p' = 4$ neurones

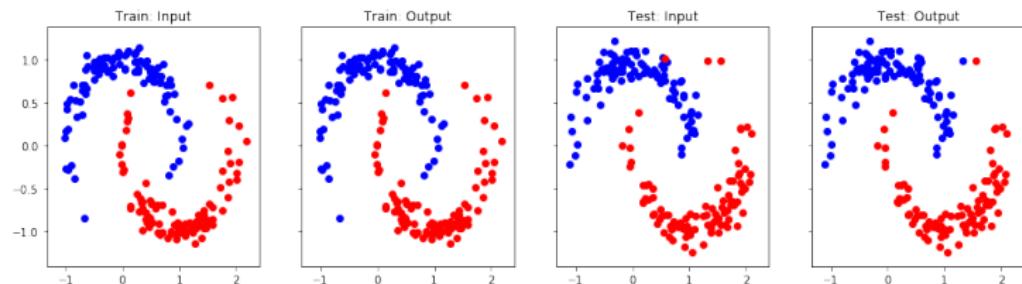
```
1 from sklearn.neural_network import MLPClassifier
2 classifier = MLPClassifier(hidden_layer_sizes=(4,), learning_rate_init=0.1)
3 classifier.fit(X_train,y_train)
4
5 yest = classifier.predict(X_train)      # prediction of training data
6 y_est = classifier.predict(X_test) # prediction of test data
7
8 plt.scatter(X_train[yest==1,0],X_train[yest==1,1],color='blue')
9 plt.scatter(X_train[yest==-1,0],X_train[yest==-1,1],color='red')
10
11 plt.scatter(X_test[y_est==1,0],X_test[y_est==1,1],color='blue')
12 plt.scatter(X_test[y_est==-1,0],X_test[y_est==-1,1],color='red')
13
```



Le modèle du “perceptron”

Implémentation Python: $p' = 10$ neurones

```
1 from sklearn.neural_network import MLPClassifier
2 classifier = MLPClassifier(hidden_layer_sizes=(10,), learning_rate_init=0.1)
3 classifier.fit(X_train, y_train)
4
5 yest = classifier.predict(X_train)      # prediction of training data
6 y_est = classifier.predict(X_test) # prediction of test data
7
8 plt.scatter(X_train[yest==1,0],X_train[yest==1,1],color='blue')
9 plt.scatter(X_train[yest==-1,0],X_train[yest==-1,1],color='red')
10
11 plt.scatter(X_test[y_est==1,0],X_test[y_est==1,1],color='blue')
12 plt.scatter(X_test[y_est==-1,0],X_test[y_est==-1,1],color='red')
13
```

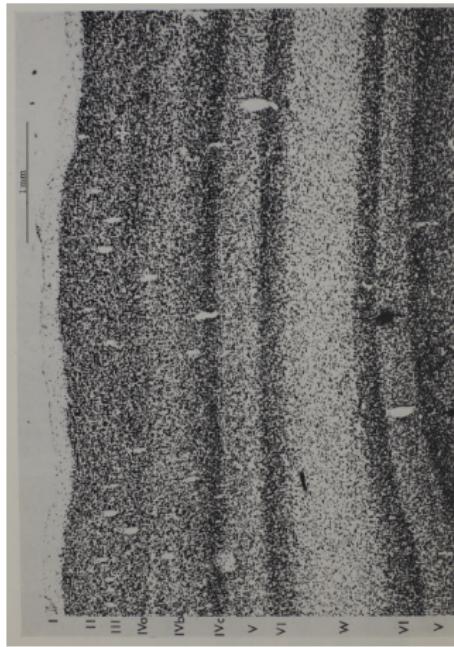


Notez le très haut niveau de performance!

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
 - Notions de base et garanties théoriques
 - Le perceptron multi-couche (MLP)
- 5 L'IA moderne: révolution informatique et retour des mathématiques

Du perceptron au “perceptron multi-couches”

Rappel: le cortex visuel humain (et animal évidemment) est organisé en couches successives de neurones.



Hubel, Wiesel, "Functional architecture of macaque monkey visual cortex", 1977.

Du perceptron au “perceptron multi-couches”

Pourquoi plusieurs couches?

Du perceptron au “perceptron multi-couches”

Pourquoi plusieurs couches?

- a priori inutile d'après le théorème d'approximation universelle?

Du perceptron au “perceptron multi-couches”

Pourquoi plusieurs couches?

- a priori inutile d'après le **théorème d'approximation universelle?**
- **oui mais... :**
 - ▶ rappelons qu'il y a une distance entre théorie (existence d'un réseau) et pratique (construction d'un réseau)

Du perceptron au “perceptron multi-couches”

Pourquoi plusieurs couches?

- a priori inutile d'après le **théorème d'approximation universelle?**
- **oui mais... :**
 - ▶ rappelons qu'il y a une distance entre théorie (existence d'un réseau) et pratique (construction d'un réseau)
 - ▶ enchaîner les fonctions d'activation non-linéaires crée des **fonctions plus riches!**

Du perceptron au “perceptron multi-couches”

Pourquoi plusieurs couches?

- a priori inutile d'après le **théorème d'approximation universelle?**
- **oui mais... :**
 - ▶ rappelons qu'il y a une distance entre théorie (existence d'un réseau) et pratique (construction d'un réseau)
 - ▶ enchaîner les fonctions d'activation non-linéaires crée des **fonctions plus riches!**

Vision “représentations” des données:

Du perceptron au “perceptron multi-couches”

Pourquoi plusieurs couches?

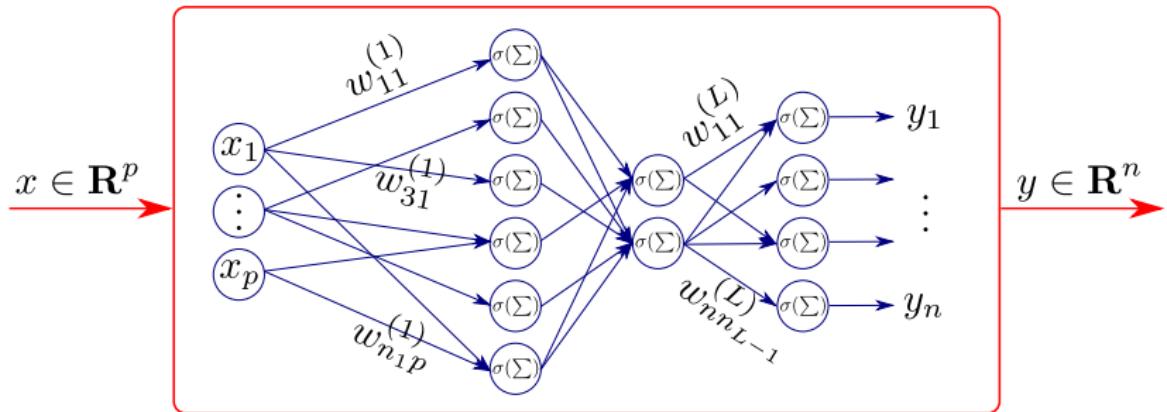
- a priori inutile d'après le **théorème d'approximation universelle?**
- **oui mais... :**
 - ▶ rappelons qu'il y a une distance entre théorie (existence d'un réseau) et pratique (construction d'un réseau)
 - ▶ enchaîner les fonctions d'activation non-linéaires crée des **fonctions plus riches!**

Vision “représentations” des données:

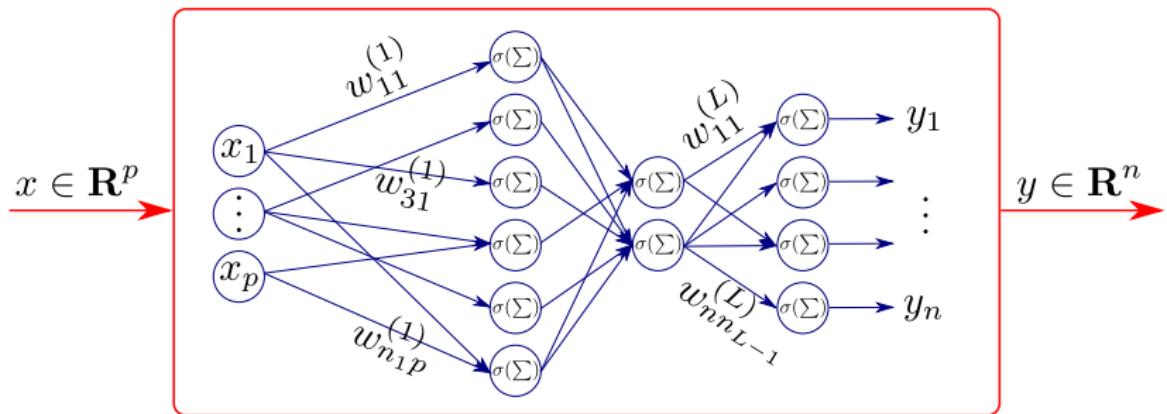
- la sortie de chaque couche génère une **nouvelle représentation de la couche précédente**:

$$x^{(\ell+1)} = \sigma(W^{(\ell)}x^{(\ell)})$$

Du perceptron au “perceptron multi-couches”



Du perceptron au “perceptron multi-couches”

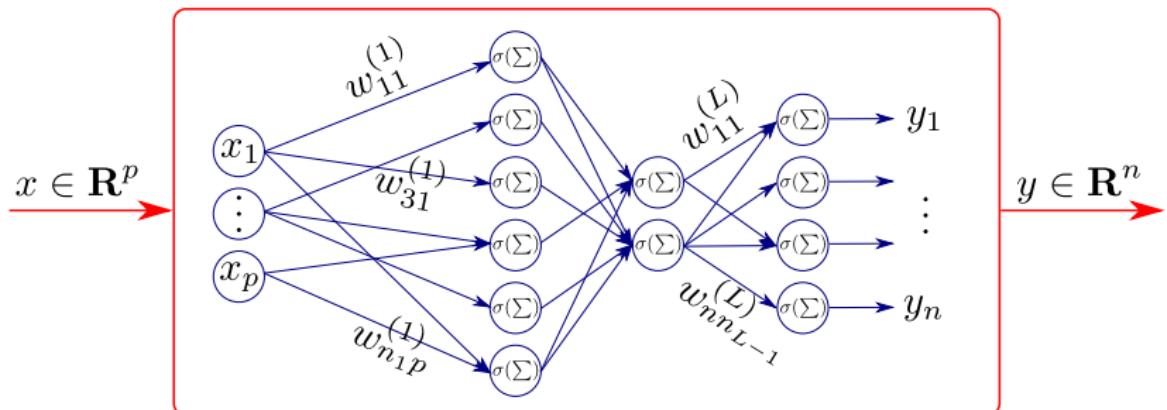


Définition [Perceptron multi-couches]

Pour $W^{(1)}, \dots, W^{(L)}$ des matrices de connexion entre couches et $\sigma_1, \dots, \sigma_L$ des fonctions d'activation,

$$y = \sigma_L \left(W^{(L)} \sigma_{L-1} \left(\cdots \sigma \left(W^{(2)} \sigma_1 \left(W^{(1)} x \right) \right) \cdots \right) \right).$$

Du perceptron au “perceptron multi-couches”



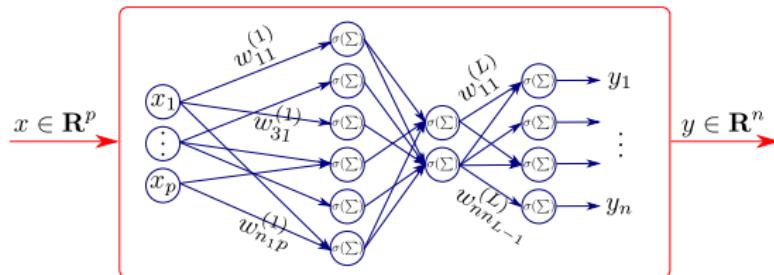
Définition [Perceptron multi-couches]

Pour $W^{(1)}, \dots, W^{(L)}$ des matrices de connexion entre couches et $\sigma_1, \dots, \sigma_L$ des fonctions d'activation,

$$y = \sigma_L \left(W^{(L)} \sigma_{L-1} \left(\cdots \sigma \left(W^{(2)} \sigma_1 \left(W^{(1)} x \right) \right) \cdots \right) \right).$$

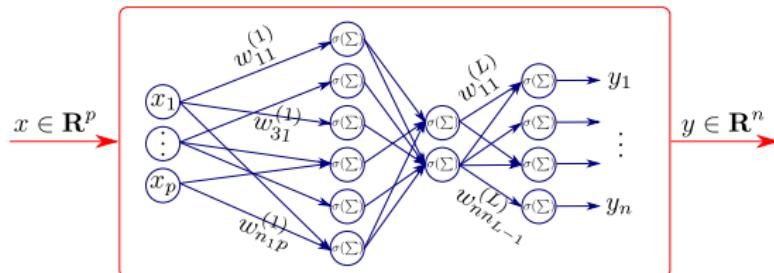
Il “suffit” d'**apprendre** $W^{(1)}, \dots, W^{(L)}$ à partir de (beaucoup de) données d'entraînement.

Du perceptron au “perceptron multi-couches”



Quelques remarques:

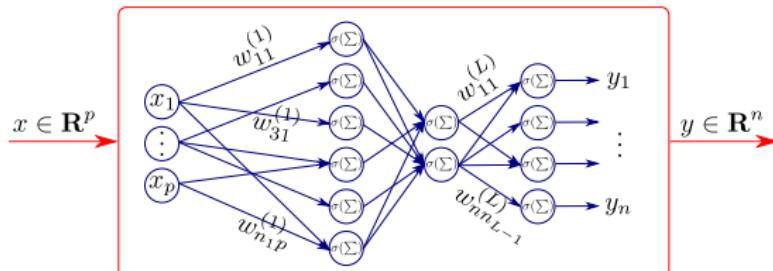
Du perceptron au “perceptron multi-couches”



Quelques remarques:

- en général, σ_i identiques pour toutes les couches, **sauf la dernière** (qui sert à la classification finale)

Du perceptron au “perceptron multi-couches”

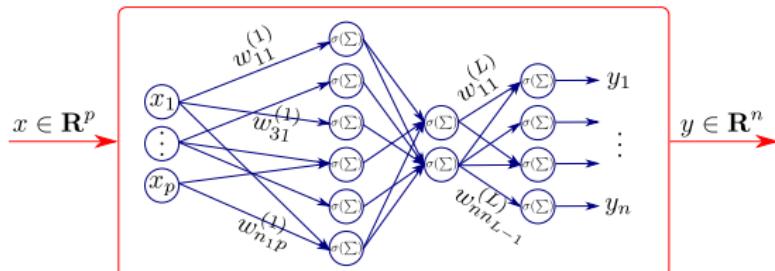


Quelques remarques:

- en général, σ_i identiques pour toutes les couches, **sauf la dernière** (qui sert à la classification finale)
- pour une classification à $k > 2$ classes, on prend une dernière couche de taille k et pour (x_i, y_i) de classe j , étiquette “**hot-bit**” :

$$y_i = [0, \dots, 0, \underbrace{1}_{\text{position } j}, 0, \dots, 0]$$

Du perceptron au “perceptron multi-couches”



Quelques remarques:

- en général, σ_i identiques pour toutes les couches, **sauf la dernière** (qui sert à la classification finale)
- pour une classification à $k > 2$ classes, on prend une dernière couche de taille k et pour (x_i, y_i) de classe j , étiquette “**hot-bit**” :

$$y_i = [0, \dots, 0, \underbrace{1}_{\text{position } j}, 0, \dots, 0]$$

- pour nouvelle donnée \hat{x} , on a en général à la sortie du réseau:

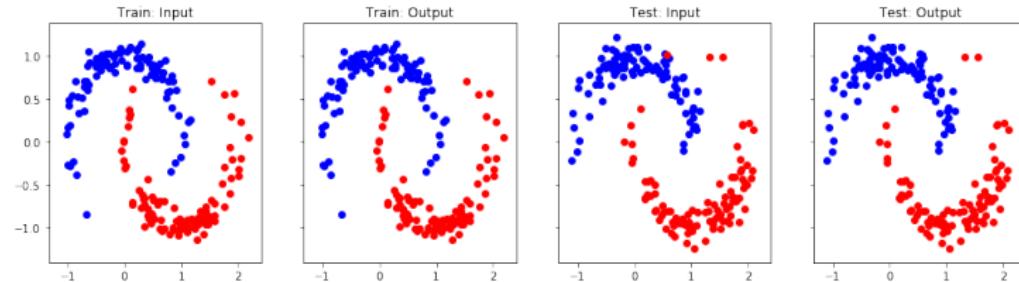
$$\hat{y} = [p_1, \dots, p_k]$$

avec p_j la “probabilité” d’être en classe j .

Du perceptron au “perceptron multi-couches”

Implémentation Python:

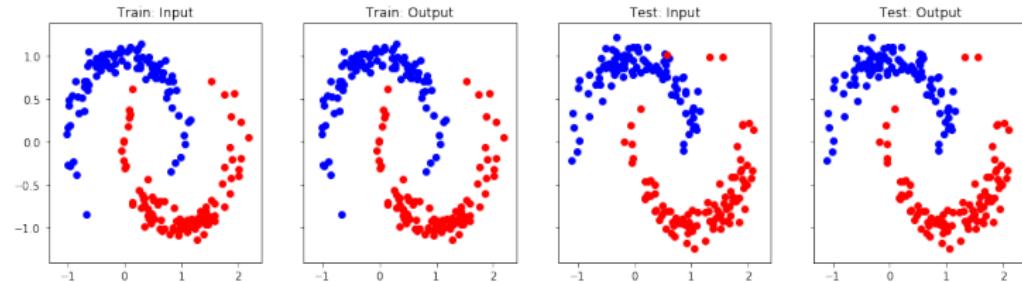
```
1 from sklearn.neural_network import MLPClassifier  
2 classifier = MLPClassifier(hidden_layer_sizes=(4,5,4),learning_rate_init=0.1)  
3 classifier.fit(X_train,y_train)
```



Du perceptron au “perceptron multi-couches”

Implémentation Python:

```
1 from sklearn.neural_network import MLPClassifier  
2 classifier = MLPClassifier(hidden_layer_sizes=(4,5,4), learning_rate_init=0.1)  
3 classifier.fit(X_train, y_train)
```



Beaucoup de paramètres réglables (à découvrir en projet)

```
1 classifier.get_params()  
  
{'activation': 'relu',  
 'alpha': 0.0001,  
 'batch_size': 'auto',  
 'beta_1': 0.9,  
 'beta_2': 0.999,  
 'early_stopping': False,  
 'epsilon': 1e-08,  
 'hidden_layer_sizes': (5, 10, 5, 3),  
 'learning_rate': 'constant',  
 'learning_rate_init': 0.1,  
 'max_fun': 15000,  
 'max_iter': 200,  
 'momentum': 0.9,  
 'n_iter_no_change': 10,  
 'nesterovs_momentum': True,  
 'power_t': 0.5,  
 'random_state': None,  
 'shuffle': True,  
 'solver': 'adam',  
 'tol': 0.0001,  
 'validation_fraction': 0.1,  
 'verbose': False,  
 'warm_start': False}
```

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques
 - La magie des réseaux convolutionnels et antagonistes
 - L'IA de demain: sobriété numérique et éthique
 - Quid des mathématiques?

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques
 - La magie des réseaux convolutionnels et antagonistes
 - L'IA de demain: sobriété numérique et éthique
 - Quid des mathématiques?

État de l'art jusqu'en 2010

Prédominance des SVMs et de l'ingénierie “manuelle”

État de l'art jusqu'en 2010

Prédominance des SVMs et de l'ingénierie “manuelle”

- les SVMs servent de classifieur, la difficulté est d'extraire de bonnes représentations (= linéairement séparables)
- ex: en traitement d'images, on découvre la **décomposition en ondelettes** (\simeq décomposition en fréquences (JPEG) mais améliorée)

État de l'art jusqu'en 2010

Prédominance des SVMs et de l'ingénierie “manuelle”

- les SVMs servent de classifieur, la difficulté est d'extraire de bonnes représentations (= linéairement séparables)
- ex: en traitement d'images, on découvre la **décomposition en ondelettes** (\simeq décomposition en fréquences (JPEG) mais améliorée)
- les performances sont correctes mais pas suffisantes (ex: insuffisant pour un véhicule autonome)
- les progrès du domaine sont lents (quelques 0.1% de performance gagnés ici et là)

État de l'art jusqu'en 2010

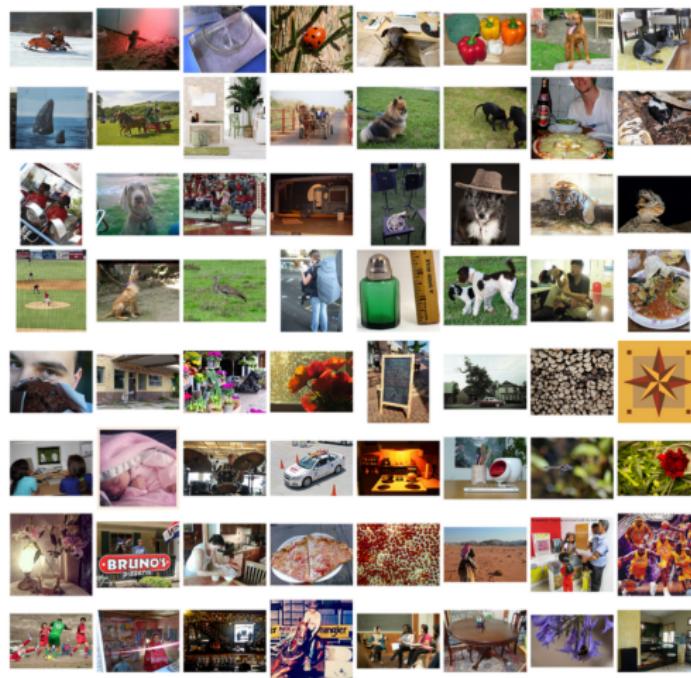
Prédominance des SVMs et de l'ingénierie “manuelle”

- les SVMs servent de classifieur, la difficulté est d'extraire de bonnes représentations (= linéairement séparables)
- ex: en traitement d'images, on découvre la **décomposition en ondelettes** (\simeq décomposition en fréquences (JPEG) mais améliorée)
- les performances sont correctes mais pas suffisantes (ex: insuffisant pour un véhicule autonome)
- les progrès du domaine sont lents (quelques 0.1% de performance gagnés ici et là)
- les réseaux de neurones ont depuis longtemps été abandonnés: **trop instables** (performances chaotiques), **trop longs à entraîner**.

État de l'art jusqu'en 2010

En 2012, tout change!

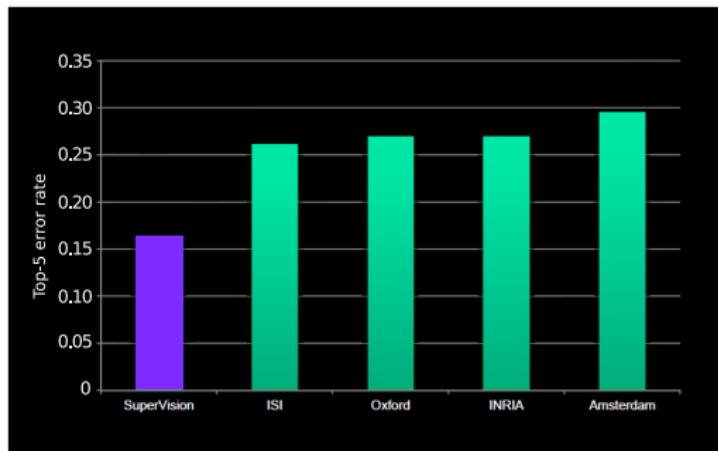
- **Contexte:** compétition de classification sur l'**énorme base de données ImageNet** (1 400 000 images, 1 000 classes!)



État de l'art jusqu'en 2010

En 2012, tout change!

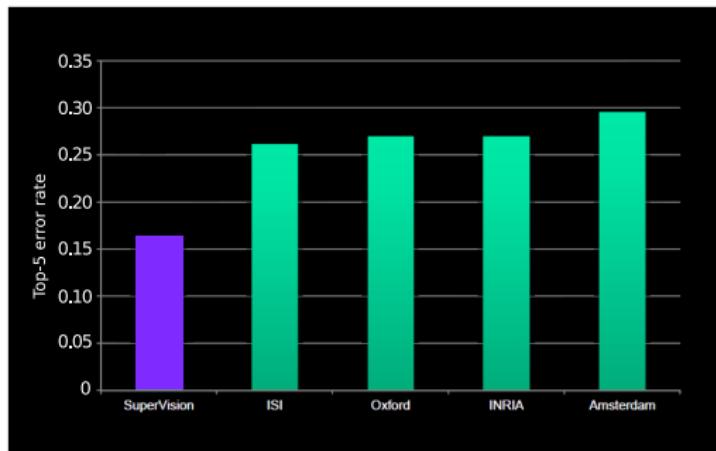
- **Résultats en 2012:** arrivée de l'algorithme “Supervision” basé sur un réseau de neurones **profond**



État de l'art jusqu'en 2010

En 2012, tout change!

- **Résultats en 2012:** arrivée de l'algorithme “Supervision” basé sur un réseau de neurones **profond**

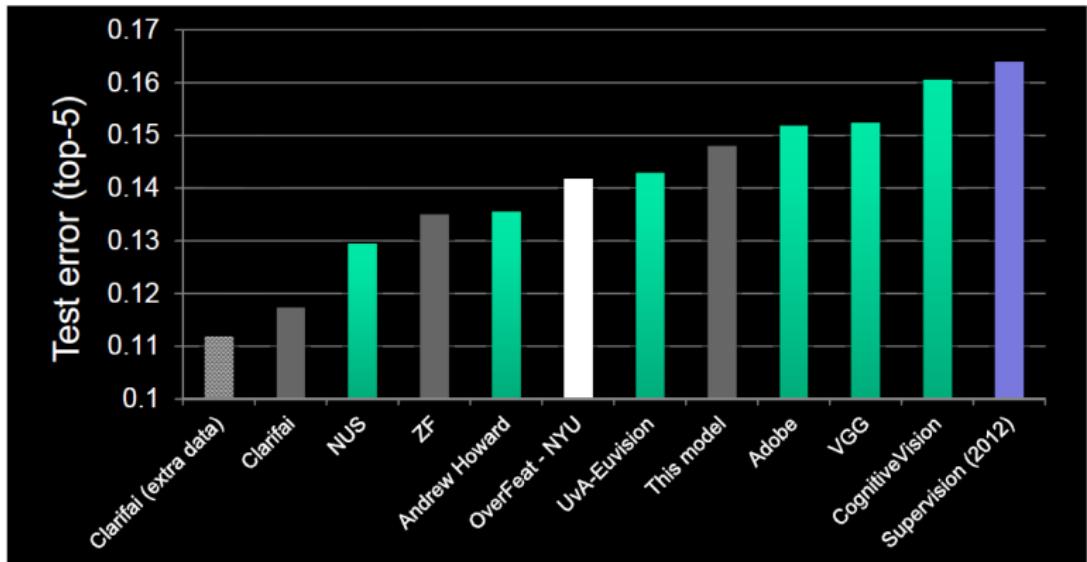


- les performances soudain sautent de –10% d'erreur!

État de l'art jusqu'en 2010

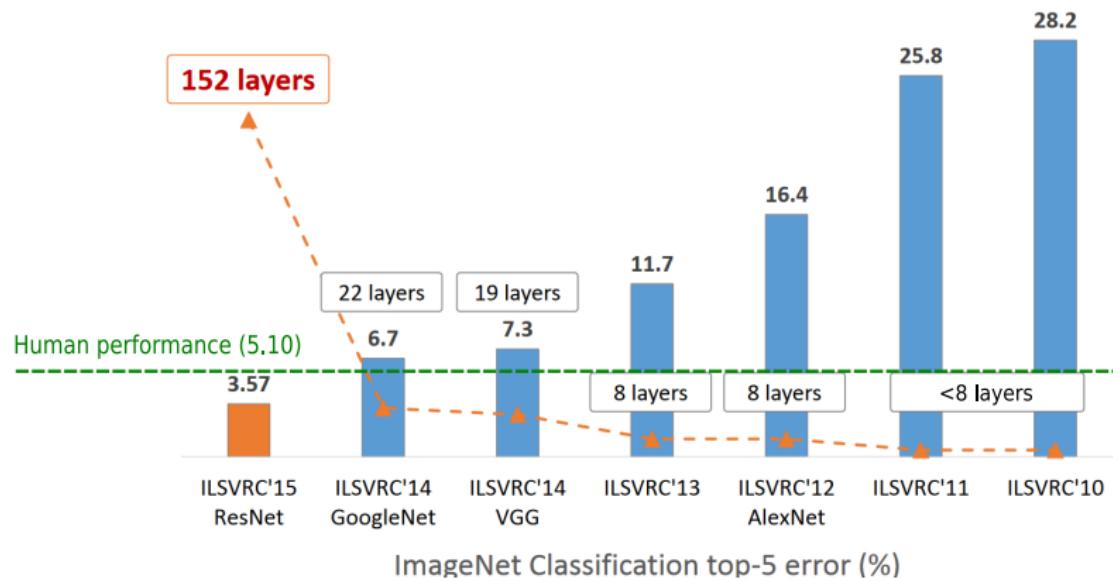
En 2012, tout change!

- **Résultats en 2013:** la chute des taux d'erreur continue!



L'après 2012, l'ère des réseaux profonds

En 2015, la machine bat l'être humain!



L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- cela change tout pour **l'apprentissage machine**: on envisage des applications réelles (reconnaissance faciale, vidéosurveillance, véhicules autonomes, etc.)

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- cela change tout pour **l'apprentissage machine**: on envisage des applications réelles (reconnaissance faciale, vidéosurveillance, véhicules autonomes, etc.)
- techniquement, les performances augmentent grâce à:
 - ▶ pas beaucoup plus de neurones par couche, mais **bien plus de couches!**

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- cela change tout pour **l'apprentissage machine**: on envisage des applications réelles (reconnaissance faciale, vidéosurveillance, véhicules autonomes, etc.)
- techniquement, les performances augmentent grâce à:
 - ▶ pas beaucoup plus de neurones par couche, mais **bien plus de couches!**
 - ▶ la stabilisation du réseau par le **gigantisme des données et du réseau** (loi des grands nombres)

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- cela change tout pour **l'apprentissage machine**: on envisage des applications réelles (reconnaissance faciale, vidéosurveillance, véhicules autonomes, etc.)
- techniquement, les performances augmentent grâce à:
 - ▶ pas beaucoup plus de neurones par couche, mais **bien plus de couches!**
 - ▶ la stabilisation du réseau par le **gigantisme des données et du réseau** (loi des grands nombres)
 - ▶ une capacité de calcul accrue grâce aux **GPU... développés pour les jeux vidéos!** (calcul matriciel massivement parallèle)

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- cela change tout pour **l'apprentissage machine**: on envisage des applications réelles (reconnaissance faciale, vidéosurveillance, véhicules autonomes, etc.)
- techniquement, les performances augmentent grâce à:
 - ▶ pas beaucoup plus de neurones par couche, mais **bien plus de couches!**
 - ▶ la stabilisation du réseau par le **gigantisme des données et du réseau** (loi des grands nombres)
 - ▶ une capacité de calcul accrue grâce aux **GPU... développés pour les jeux vidéos!** (calcul matriciel massivement parallèle)
 - ▶ le développement de serveurs de calculs massif: **milliers de GPUs qui tournent en parallèle!**

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- cela change tout pour **l'apprentissage machine**: on envisage des applications réelles (reconnaissance faciale, vidéosurveillance, véhicules autonomes, etc.)
- techniquement, les performances augmentent grâce à:
 - ▶ pas beaucoup plus de neurones par couche, mais **bien plus de couches!**
 - ▶ la stabilisation du réseau par le **gigantisme des données et du réseau** (loi des grands nombres)
 - ▶ une capacité de calcul accrue grâce aux **GPU... développés pour les jeux vidéos!** (calcul matriciel massivement parallèle)
 - ▶ le développement de serveurs de calculs massif: **milliers de GPUs qui tournent en parallèle!**
 - ▶ l'utilisation d'une structure **convolutionnelle et hiérarchique** dans les matrices $W^{(\ell)}$: proche du traitement par l'œil

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- cela change tout pour **l'apprentissage machine**: on envisage des applications réelles (reconnaissance faciale, vidéosurveillance, véhicules autonomes, etc.)
- techniquement, les performances augmentent grâce à:
 - ▶ pas beaucoup plus de neurones par couche, mais **bien plus de couches!**
 - ▶ la stabilisation du réseau par le **gigantisme des données et du réseau** (loi des grands nombres)
 - ▶ une capacité de calcul accrue grâce aux **GPU... développés pour les jeux vidéos!** (calcul matriciel massivement parallèle)
 - ▶ le développement de serveurs de calculs massif: **milliers de GPUs qui tournent en parallèle!**
 - ▶ l'utilisation d'une structure **convolutionnelle et hiérarchique** dans les matrices $W^{(\ell)}$: proche du traitement par l'œil
 - ▶ malgré tout, il faut **plusieurs semaines pour entraîner le réseau!**: taille totale de $W = \{W^{(1)}, \dots, W^{(L)}\} = \text{centaines de millions!!!}$

$$[w_{11}, w_{12}, w_{13}, \dots] \in \mathbb{R}^{100\,000\,000}$$

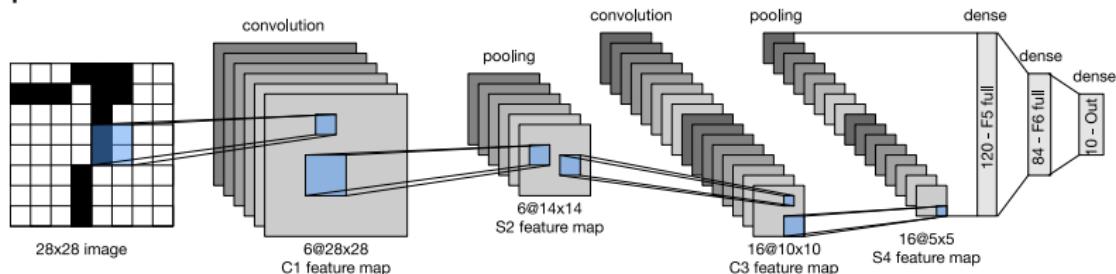
L'après 2012, l'ère des réseaux profonds

Aparté technique sur les réseaux profonds: (deep neural nets: DNN)

L'après 2012, l'ère des réseaux profonds

Aparté technique sur les réseaux profonds: (deep neural nets: DNN)

- plusieurs “astuces” très utiles:

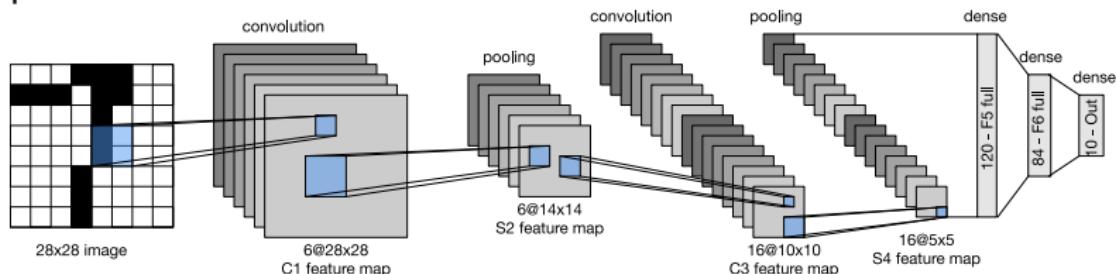


- matrices $W^{(\ell)}$ très **creuses**: $w_{ij} = 0$ si $|i - j|$ grand (pixels éloignés non comparés)

L'après 2012, l'ère des réseaux profonds

Aparté technique sur les réseaux profonds: (deep neural nets: DNN)

- plusieurs “astuces” très utiles:

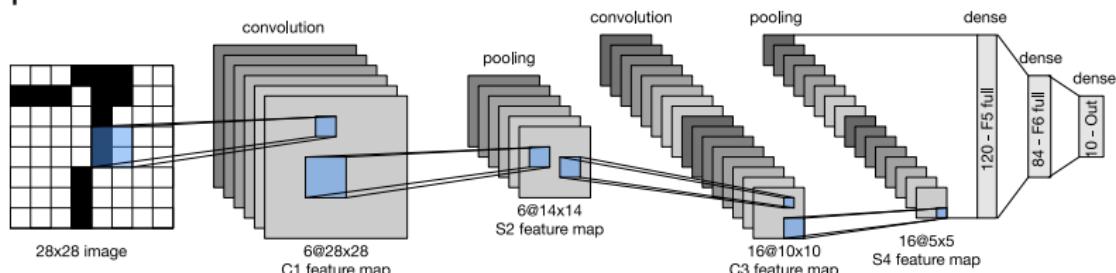


- matrices $W^{(\ell)}$ très **creuses**: $w_{ij} = 0$ si $|i - j|$ grand (pixels éloignés non comparés)
- matrices $W^{(\ell)}$ “**périodiques**”: $w_{ij} = w_{kl}$ si $i - j = k - l$ (on traite les pixels voisins de la même manière partout dans l'image)

L'après 2012, l'ère des réseaux profonds

Aparté technique sur les réseaux profonds: (deep neural nets: DNN)

- plusieurs “astuces” très utiles:

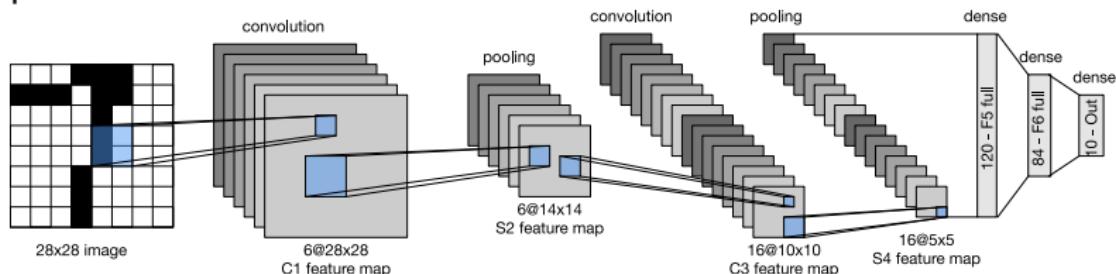


- matrices $W^{(\ell)}$ très **creuses**: $w_{ij} = 0$ si $|i - j|$ grand (pixels éloignés non comparés)
- matrices $W^{(\ell)}$ “**périodiques**”: $w_{ij} = w_{kl}$ si $i - j = k - l$ (on traite les pixels voisins de la même manière partout dans l'image)
- le nombre de neurones réduit au fil des couches: synthèse hiérarchique des informations (du local au global!)

L'après 2012, l'ère des réseaux profonds

Aparté technique sur les réseaux profonds: (deep neural nets: DNN)

- plusieurs “astuces” très utiles:



- matrices $W^{(\ell)}$ très **creuses**: $w_{ij} = 0$ si $|i - j|$ grand (pixels éloignés non comparés)
- matrices $W^{(\ell)}$ “**périodiques**”: $w_{ij} = w_{kl}$ si $i - j = k - l$ (on traite les pixels voisins de la même manière partout dans l'image)
- le nombre de neurones réduit au fil des couches: synthèse hiérarchique des informations (du local au global!)
- dernières couches entièrement connectées “à l'ancienne”.

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- aujourd'hui, cela prend des dimensions réellement démesurées: coût en "équivalent CO₂" intenable (entraînement du réseau "transformers" en traitement du langage = EqCO₂ de la **durée de vie de 3 SUVs**)

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- aujourd'hui, cela prend des dimensions réellement démesurées: coût en "équivalent CO₂" intenable (entraînement du réseau "transformers" en traitement du langage = EqCO₂ de la **durée de vie de 3 SUVs**)
- paradoxalement, théoriquement on est toujours bloqué en 1950! (théorème d'approximation universelle): combien de couches? de neurones? que fait le réseau? **toujours aucune réponse...**

L'après 2012, l'ère des réseaux profonds

Quelques remarques importantes:

- aujourd'hui, cela prend des dimensions réellement démesurées: coût en “équivalent CO₂” intenable (entraînement du réseau “transformers” en traitement du langage = EqCO₂ de la **durée de vie de 3 SUVs**)
- paradoxalement, théoriquement on est toujours bloqué en 1950! (théorème d’approximation universelle): combien de couches? de neurones? que fait le réseau? **toujours aucune réponse...**
- **conséquence:** l’IA est devenu “un art” plus qu’une science!... mais un art qui coûte absurdement cher!

Vision pour l’homme possible en mangeant une pomme par jour



Vision pour la machine possible en “mangeant” des tonnes de pétrole.

L'exemple frappant des “GAN”

La génération de fausses données réalistes: (images, sons, textes)

L'exemple frappant des “GAN”

La génération de fausses données réalistes: (images, sons, textes)

- encore considéré récemment comme un problème impossible à résoudre avant des dizaines d'années

L'exemple frappant des “GAN”

La génération de fausses données réalistes: (images, sons, textes)

- encore considéré récemment comme un problème impossible à résoudre avant des dizaines d'années
- soudainement résolu en 2016 (*I. Goodfellow, “Generative adversarial networks”*)

L'exemple frappant des “GAN”

La génération de fausses données réalistes: (images, sons, textes)

- encore considéré récemment comme un problème impossible à résoudre avant des dizaines d'années
- soudainement résolu en 2016 (*I. Goodfellow, “Generative adversarial networks”*)
- grâce à... deux réseaux de neurones en compétition l'un contre l'autre!

L'exemple frappant des “GAN”

La génération de fausses données réalistes: (images, sons, textes)

- encore considéré récemment comme un problème impossible à résoudre avant des dizaines d'années
- soudainement résolu en 2016 (*I. Goodfellow, “Generative adversarial networks”*)
- grâce à... deux réseaux de neurones en compétition l'un contre l'autre!

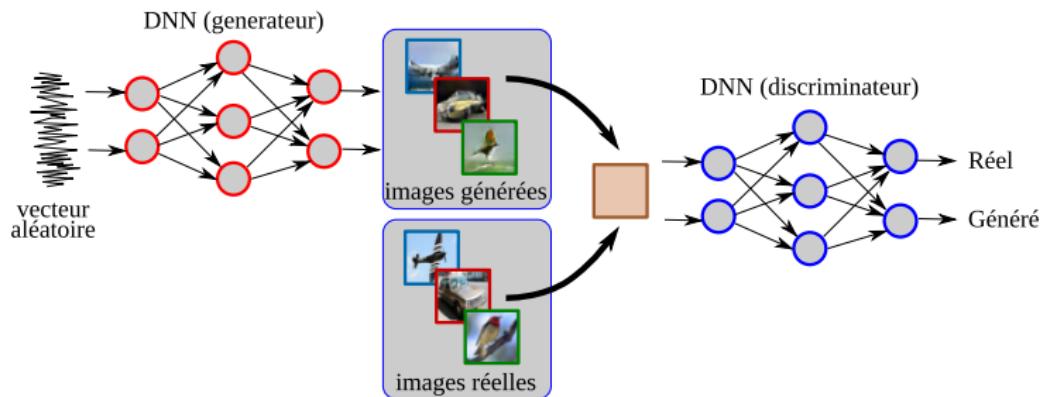
Les réseaux antagonistes: (GAN)

L'exemple frappant des "GAN"

La génération de fausses données réalistes: (images, sons, textes)

- encore considéré récemment comme un problème impossible à résoudre avant des dizaines d'années
- soudainement résolu en 2016 (*I. Goodfellow, "Generative adversarial networks"*)
- grâce à... deux réseaux de neurones en compétition l'un contre l'autre!

Les réseaux antagonistes: (GAN)



L'exemple frappant des “GAN”

Fonctionnement:

L'exemple frappant des “GAN”

Fonctionnement:

- le DNN discriminateur a pour objectif de différencier des images vraies et fausses: au début de l'entraînement, sa tâche est facile!

L'exemple frappant des “GAN”

Fonctionnement:

- le DNN discriminateur a pour objectif de différencier des images vraies et fausses: au début de l'entraînement, sa tâche est facile!
- le DNN générateur transforme un vecteur de bruit en une sortie, et **a pour objectif de tromper le discriminateur** (augmenter son erreur)

L'exemple frappant des “GAN”

Fonctionnement:

- le DNN discriminateur a pour objectif de différencier des images vraies et fausses: au début de l'entraînement, sa tâche est facile!
- le DNN générateur transforme un vecteur de bruit en une sortie, et **a pour objectif de tromper le discriminateur** (augmenter son erreur)
- le générateur va donc (malgré lui) générer des “images” **purement aléatoires** mais proches des images réelles

L'exemple frappant des “GAN”

Fonctionnement:

- le DNN discriminateur a pour objectif de différencier des images vraies et fausses: au début de l'entraînement, sa tâche est facile!
- le DNN générateur transforme un vecteur de bruit en une sortie, et **a pour objectif de tromper le discriminateur** (augmenter son erreur)
- le générateur va donc (malgré lui) générer des “images” **purement aléatoires** mais proches des images réelles

Quelques remarques:

- théoriquement, on ne sait pas pourquoi cela fonctionne (aussi bien)!

L'exemple frappant des “GAN”

Fonctionnement:

- le DNN discriminateur a pour objectif de différencier des images vraies et fausses: au début de l'entraînement, sa tâche est facile!
- le DNN générateur transforme un vecteur de bruit en une sortie, et **a pour objectif de tromper le discriminateur** (augmenter son erreur)
- le générateur va donc (malgré lui) générer des “images” **purement aléatoires** mais proches des images réelles

Quelques remarques:

- théoriquement, on ne sait pas pourquoi cela fonctionne (aussi bien)!
- mais le résultat est puissant: le DNN générateur a “recréé” **la variété (la déformation φ de l'espace) qui plonge n'importe quel vecteur aléatoire en une image**: ex., il sait construire **la variété “chien”** dans le monde des “tas de pixels possibles”

L'exemple frappant des “GAN”

Et voilà le résultat:



- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques
 - La magie des réseaux convolutionnels et antagonistes
 - L'IA de demain: sobriété numérique et éthique
 - Quid des mathématiques?

Le problème de l'éthique en IA

L'inquiétude de la “boîte noire”:

Le problème de l'éthique en IA

L'inquiétude de la “boîte noire” :

- les algorithmes deviennent puissants **mais** aussi incontrôlables! (on ne sait pas ce qu'ils font)

Le problème de l'éthique en IA

L'inquiétude de la “boîte noire”:

- les algorithmes deviennent puissants **mais** aussi incontrôlables! (on ne sait pas ce qu'ils font)
- utilisés dans la société, cela pose des **questions éthiques graves**:
 - ▶ **biais et discriminations**: l'algorithme favorise la majorité, et peut défavoriser les minorités

Le problème de l'éthique en IA

L'inquiétude de la “boîte noire” :

- les algorithmes deviennent puissants **mais** aussi incontrôlables! (on ne sait pas ce qu'ils font)
- utilisés dans la société, cela pose des **questions éthiques graves**:
 - ▶ **biais et discriminations**: l'algorithme favorise la majorité, et peut défavoriser les minorités
 - ▶ **renforcement des discriminations**: l'algorithme auto-apprend ou se sert des décisions humaines basées sur ses propres règles précédentes (cercle vicieux!)

Le problème de l'éthique en IA

L'inquiétude de la “boîte noire” :

- les algorithmes deviennent puissants **mais** aussi incontrôlables! (on ne sait pas ce qu'ils font)
- utilisés dans la société, cela pose des **questions éthiques graves**:
 - ▶ **biais et discriminations**: l'algorithme favorise la majorité, et peut défavoriser les minorités
 - ▶ **renforcement des discriminations**: l'algorithme auto-apprend ou se sert des décisions humaines basées sur ses propres règles précédentes (cercle vicieux!)
 - ▶ **le problème des données**: biais de sélection des données accessibles (exclut certaines populations)

Le problème de l'éthique en IA

Pourquoi l'algorithme n'est en fait **pas objectif**:

Le problème de l'éthique en IA

Pourquoi l'algorithme n'est en fait **pas objectif**:

- l'algorithme fait ce qu'on lui demande: en ce sens, il **est** objectif (contrairement parfois à l'humain)

Le problème de l'éthique en IA

Pourquoi l'algorithme n'est en fait **pas objectif**:

- l'algorithme fait ce qu'on lui demande: en ce sens, il **est** objectif (contrairement parfois à l'humain)
- mais les algorithmes sont **écrits par des humains**, qui transfèrent donc leurs propres biais (volontairement ou non)

Le problème de l'éthique en IA

Pourquoi l'algorithme n'est en fait **pas objectif**:

- l'algorithme fait ce qu'on lui demande: en ce sens, il **est** objectif (contrairement parfois à l'humain)
- mais les algorithmes sont **écrits par des humains**, qui transfèrent donc leurs propres biais (volontairement ou non)
- les données d'apprentissage sont **étiquetées par des humains**: toute la connaissance "de base" (et les erreurs apprises!) sont humaines

Le problème de l'éthique en IA

Pourquoi l'algorithme n'est en fait pas objectif:

- l'algorithme fait ce qu'on lui demande: en ce sens, il **est** objectif (contrairement parfois à l'humain)
- mais les algorithmes sont **écrits par des humains**, qui transfèrent donc leurs propres biais (volontairement ou non)
- les données d'apprentissage sont **étiquetées par des humains**: toute la connaissance "de base" (et les erreurs apprises!) sont humaines
- effet domino: les **nouvelles décisions de l'algorithme alimentent d'autres algorithmes.**

Le problème de l'éthique en IA

Quelles sont les solutions?

Le problème de l'éthique en IA

Quelles sont les solutions?

- problème délicat car mal défini: qu'est-ce qui est "discriminant" ?

Le problème de l'éthique en IA

Quelles sont les solutions?

- problème délicat car mal défini: qu'est-ce qui est "discriminant"?
 - ▶ peut-on utiliser une info discriminante (âge, sexe, religion, salaire) dans la **représentation des données** de sorte à corriger les biais?

Le problème de l'éthique en IA

Quelles sont les solutions?

- problème délicat car mal défini: qu'est-ce qui est "discriminant"?
 - ▶ peut-on utiliser une info discriminante (âge, sexe, religion, salaire) dans la **représentation des données** de sorte à corriger les biais?
 - ▶ si c'est interdit, peut-on altérer les données d'entraînement (ou leur traitement) pour infléchir les décisions?

Le problème de l'éthique en IA

Quelles sont les solutions?

- problème délicat car mal défini: qu'est-ce qui est "discriminant"?
 - ▶ peut-on utiliser une info discriminante (âge, sexe, religion, salaire) dans la **représentation des données** de sorte à corriger les biais?
 - ▶ si c'est interdit, peut-on altérer les données d'entraînement (ou leur traitement) pour infléchir les décisions?
 - ▶ ex: un algorithme lie des CV et choisit un candidat pour un poste:
 - ★ on remarque qu'il favorise les hommes aux femmes: c'est discriminant pour les femmes
 - ★ on utilise alors l'information "sexe" dans l'algorithme pour augmenter les chances des femmes: mais c'est alors discriminant pour les hommes.

Le problème de l'éthique en IA

Quelles sont les solutions?

- problème délicat car mal défini: qu'est-ce qui est "discriminant"?
 - ▶ peut-on utiliser une info discriminante (âge, sexe, religion, salaire) dans la **représentation des données** de sorte à corriger les biais?
 - ▶ si c'est interdit, peut-on altérer les données d'entraînement (ou leur traitement) pour infléchir les décisions?
 - ▶ ex: un algorithme lie des CV et choisit un candidat pour un poste:
 - ★ on remarque qu'il favorise les hommes aux femmes: c'est discriminant pour les femmes
 - ★ on utilise alors l'information "sexe" dans l'algorithme pour augmenter les chances des femmes: mais c'est alors discriminant pour les hommes.
- on doit donc **fixer une définition mathématique claire** de l'équité (en termes de probabilités) selon la situation

Le problème de l'éthique en IA

Quelles sont les solutions?

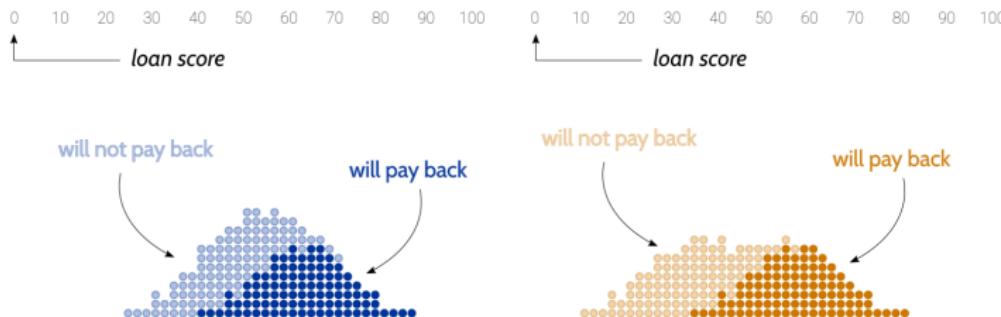
- problème délicat car mal défini: qu'est-ce qui est "discriminant"?
 - ▶ peut-on utiliser une info discriminante (âge, sexe, religion, salaire) dans la **représentation des données** de sorte à corriger les biais?
 - ▶ si c'est interdit, peut-on altérer les données d'entraînement (ou leur traitement) pour infléchir les décisions?
 - ▶ ex: un algorithme lie des CV et choisit un candidat pour un poste:
 - ★ on remarque qu'il favorise les hommes aux femmes: c'est discriminant pour les femmes
 - ★ on utilise alors l'information "sexe" dans l'algorithme pour augmenter les chances des femmes: mais c'est alors discriminant pour les hommes.
- on doit donc **fixer une définition mathématique claire** de l'équité (en termes de probabilités) selon la situation
- et alors développer des algorithmes **solutions du problème sous contrainte** d'équité.

Exemple d'application: prêt bancaire (loan grant)

Emprunté de:

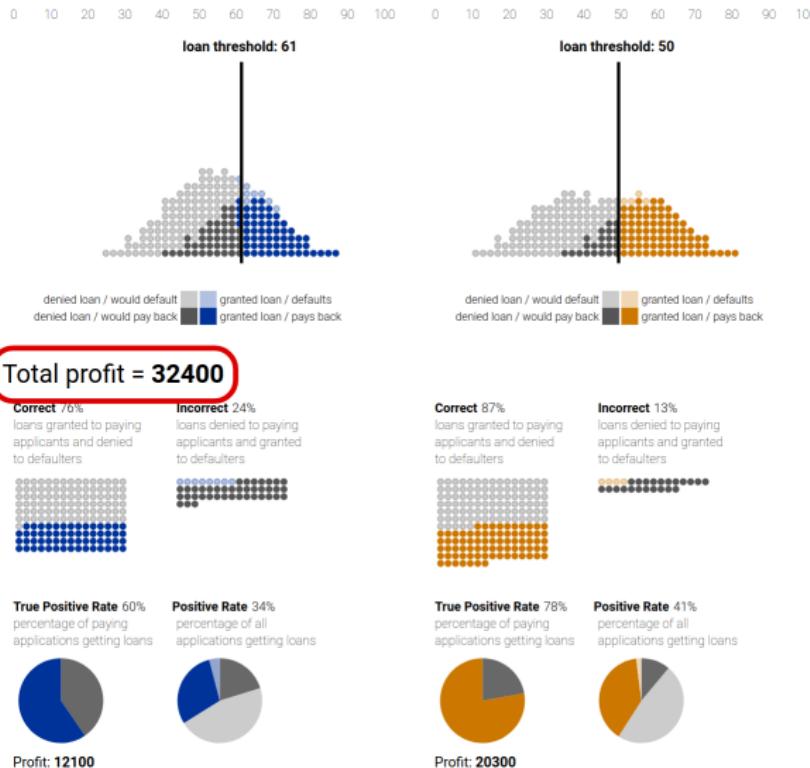
<https://research.google.com/bigpicture/attacking-discrimination-in-ml/>

Populations:



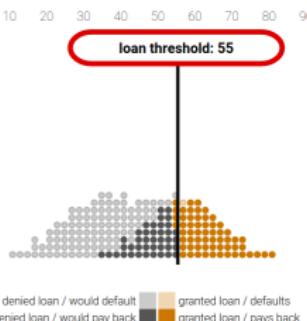
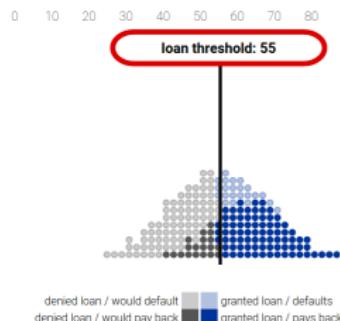
Exemple d'application: prêt bancaire (loan grant)

Aucune équité: profit maximal pour la banque



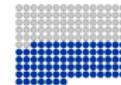
Exemple d'application: prêt bancaire (loan grant)

Équité par omission des groupes: profit maximal comme s'il n'y avait qu'un seul groupe



Total profit = 25600

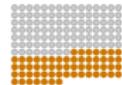
Correct 79%
loans granted to paying applicants and denied to defaulters



Incorrect 21%
loans denied to paying applicants and granted to defaulters



Correct 79%
loans granted to paying applicants and denied to defaulters



Incorrect 21%
loans denied to paying applicants and granted to defaulters



True Positive Rate 81%
percentage of paying applications getting loans



Positive Rate 52%
percentage of all applications getting loans



True Positive Rate 60%
percentage of paying applications getting loans

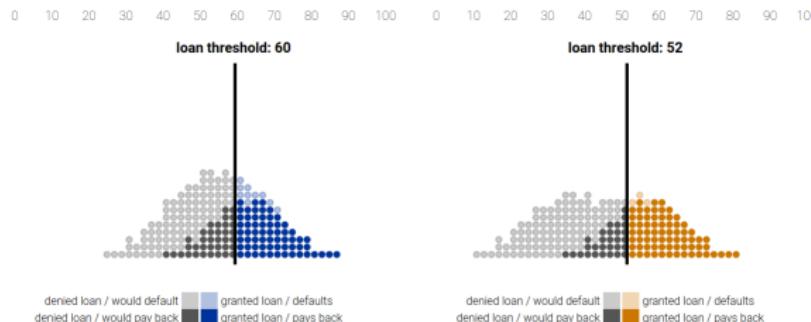


Positive Rate 30%
percentage of all applications getting loans



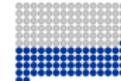
Exemple d'application: prêt bancaire (loan grant)

Équité par équilibrage démographique: (indépendance entre décision et groupe)



Total profit = 30800

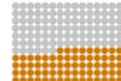
Correct 77%
loans granted to paying applicants and denied to defaulters



Incorrect 23%
loans denied to paying applicants and granted to defaulters



Correct 84%
loans granted to paying applicants and denied to defaulters



Incorrect 16%
loans denied to paying applicants and granted to defaulters



True Positive Rate 64%
percentage of paying applications getting loans



Positive Rate 37%
percentage of all applications getting loans



Profit: 11900

True Positive Rate 71%
percentage of paying applications getting loans



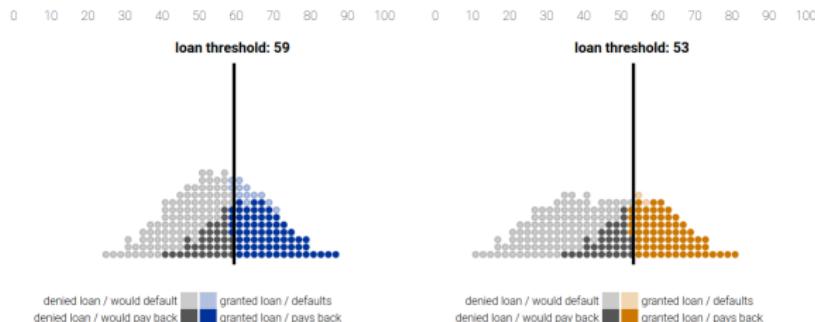
Positive Rate 37%
percentage of all applications getting loans



Profit: 18900

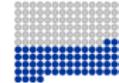
Exemple d'application: prêt bancaire (loan grant)

Équité par équilibre des mérites: (indépendance entre décision et groupe **si la solution cherchée était connue**)



Total profit = 30400

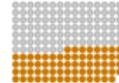
Correct 78%
loans granted to paying
applicants and denied
to defaulters



Incorrect 22%
loans denied to paying
applicants and granted
to defaulters



Correct 83%
loans granted to paying
applicants and denied
to defaulters



Incorrect 17%
loans denied to paying
applicants and granted
to defaulters



True Positive Rate 68%
percentage of paying
applications getting loans



Profit: 11700

Positive Rate 40%
percentage of all
applications getting loans



True Positive Rate 68%
percentage of paying
applications getting loans



Profit: 18700

Positive Rate 35%
percentage of all
applications getting loans



Le problème de l'éthique en IA

Quels liens avec le reste du cours?

Le problème de l'éthique en IA

Quels liens avec le reste du cours?

- les algorithmes vus en cours (kNN, SVM, réseaux de neurones) sont des **solutions de problèmes d'optimisation** (ils minimisent un “coût”)

Le problème de l'éthique en IA

Quels liens avec le reste du cours?

- les algorithmes vus en cours (kNN, SVM, réseaux de neurones) sont des **solutions de problèmes d'optimisation** (ils minimisent un “coût”)
- ces coûts ne prennent pas en compte l'éthique: cela induit des biais, de la discrimination

Le problème de l'éthique en IA

Quels liens avec le reste du cours?

- les algorithmes vus en cours (kNN, SVM, réseaux de neurones) sont des **solutions de problèmes d'optimisation** (ils minimisent un “coût”)
- ces coûts ne prennent pas en compte l'éthique: cela induit des biais, de la discrimination
- il faut donc **changer le problème d'optimisation en ajoutant des contraintes** (interdire aux solutions trouvées d'être biaisées: on sait l'écrire mathématiquement!)

La question de la sobriété numérique

Le problème du compromis performances-besoins-énergie:

La question de la sobriété numérique

Le problème du compromis performances-besoins-énergie:

- on l'a vu, les algorithmes modernes sont puissants, de haute performance, et boostés par les technologies modernes (ex: GPU)

La question de la sobriété numérique

Le problème du compromis performances-besoins-énergie:

- on l'a vu, les algorithmes modernes sont puissants, de haute performance, et boostés par les technologies modernes (ex: GPU)
- cela crée un **effet rebond** classique en écologie-technologie:
 - ▶ quand une techno est plus efficace, on gagne en consommation: **effet positif!**

La question de la sobriété numérique

Le problème du compromis performances-besoins-énergie:

- on l'a vu, les algorithmes modernes sont puissants, de haute performance, et boostés par les technologies modernes (ex: GPU)
- cela crée un **effet rebond** classique en écologie-technologie:
 - ▶ quand une techno est plus efficace, on gagne en consommation: **effet positif!**
 - ▶ mais alors on multiplie la technologie partout! (ex: dans tous les téléphones au lieu d'un seul ordinateur): **effet négatif!**

La question de la sobriété numérique

Le problème du compromis performances-besoins-énergie:

- on l'a vu, les algorithmes modernes sont puissants, de haute performance, et boostés par les technologies modernes (ex: GPU)
- cela crée un **effet rebond** classique en écologie-technologie:
 - ▶ quand une techno est plus efficace, on gagne en consommation: **effet positif!**
 - ▶ mais alors on multiplie la technologie partout! (ex: dans tous les téléphones au lieu d'un seul ordinateur): **effet négatif!**
 - ▶ au final, très souvent, **la somme totale des coûts démultipliés est bien pire qu'avant!**

$\text{CO}_2 = \text{"population"} \times \text{"usages par individu"} \times \text{"inefficacité technologique"}$

La question de la sobriété numérique

Le problème du compromis performances-besoins-énergie:

- on l'a vu, les algorithmes modernes sont puissants, de haute performance, et boostés par les technologies modernes (ex: GPU)
- cela crée un **effet rebond** classique en écologie-technologie:
 - ▶ quand une techno est plus efficace, on gagne en consommation: **effet positif!**
 - ▶ mais alors on multiplie la technologie partout! (ex: dans tous les téléphones au lieu d'un seul ordinateur): **effet négatif!**
 - ▶ au final, très souvent, **la somme totale des coûts démultipliés est bien pire qu'avant!**

$$\text{CO}_2 = \text{"population"} \times \text{"usages par individu"} \times \text{"inefficacité technologique"}$$

- la puissance de l'IA nous mène au **développement de technologies "gadget"** (visio avec oreilles de chats!, deep fakes)

La question de la sobriété numérique

Tout se passe (encore!) comme si les ressources étaient infinies et l'impact planétaire était nul:

La question de la sobriété numérique

Tout se passe (encore!) comme si les ressources étaient infinies et l'impact planétaire était nul:

- le bilan carbone IA explose: opposition frontale aux accords de Paris

La question de la sobriété numérique

Tout se passe (encore!) comme si les ressources étaient infinies et l'impact planétaire était nul:

- le bilan carbone IA explose: opposition frontale aux accords de Paris
- l'énergie nécessaire (électricité) augmente drastiquement et est produite à 65% au charbon/pétrole/gaz (sans transition court-terme viable)

Source	2018	Vol. 2000-2018
charbon	38%	×1.7
gaz/pétrole	26%	×2.2
hydro	16%	×1.6
nucléaire	10%	×1.05
solaire PV	2%	×70k
éolien	5%	×5k

La question de la sobriété numérique

Tout se passe (encore!) comme si les ressources étaient infinies et l'impact planétaire était nul:

- le bilan carbone IA explose: opposition frontale aux accords de Paris
- l'énergie nécessaire (électricité) augmente drastiquement et est produite à 65% au charbon/pétrole/gaz (sans transition court-terme viable)

Source	2018	Vol. 2000-2018
charbon	38%	×1.7
gaz/pétrole	26%	×2.2
hydro	16%	×1.6
nucléaire	10%	×1.05
solaire PV	2%	×70k
éolien	5%	×5k

- l'impact planétaire (biodiversité, températures, famines, sécheresses et feux de forêts) s'aggrave

La question de la sobriété numérique

Les (tentatives de) solutions:

La question de la sobriété numérique

Les (tentatives de) solutions:

- évaluer le bilan carbone de chaque technologie et **conserver les technologies:**
 - ▶ de **bilan carbone négatif** (celles dont les gains surcompensent les coûts): régulation thermique automatique, prévention sociale (médicale, climatique)

La question de la sobriété numérique

Les (tentatives de) solutions:

- évaluer le bilan carbone de chaque technologie et **conserver les technologies**:
 - ▶ de **bilan carbone négatif** (celles dont les gains surcompensent les coûts): régulation thermique automatique, prévention sociale (médicale, climatique)
 - ▶ de **bilan carbone positif mais de nécessité planétaire** (lutte contre déforestation, transition végétale)

La question de la sobriété numérique

Les (tentatives de) solutions:

- évaluer le bilan carbone de chaque technologie et **conserver les technologies**:
 - ▶ de **bilan carbone négatif** (celles dont les gains surcompensent les coûts): régulation thermique automatique, prévention sociale (médicale, climatique)
 - ▶ de **bilan carbone positif mais de nécessité planétaire** (lutte contre déforestation, transition végétale)
 - ▶ refuser toute technologie au bilan désastreux ou à l'intérêt illusoire (réseaux de neurones inutilement surpuissants, jeux vidéos, technologies de pseudo-confort)

La question de la sobriété numérique

Les (tentatives de) solutions:

- évaluer le bilan carbone de chaque technologie et **conserver les technologies**:
 - ▶ de **bilan carbone négatif** (celles dont les gains surcompensent les coûts): régulation thermique automatique, prévention sociale (médicale, climatique)
 - ▶ de **bilan carbone positif mais de nécessité planétaire** (lutte contre déforestation, transition végétale)
 - ▶ refuser toute technologie au bilan désastreux ou à l'intérêt illusoire (réseaux de neurones inutilement surpuissants, jeux vidéos, technologies de pseudo-confort)
- plus de théorie de l'IA pour atteindre des rapports performance/coût meilleurs: **développer les mathématiques pour l'IA.**

La question de la sobriété numérique

Les (tentatives de) solutions:

- évaluer le bilan carbone de chaque technologie et **conserver les technologies**:
 - ▶ de **bilan carbone négatif** (celles dont les gains surcompensent les coûts): régulation thermique automatique, prévention sociale (médicale, climatique)
 - ▶ de **bilan carbone positif mais de nécessité planétaire** (lutte contre déforestation, transition végétale)
 - ▶ refuser toute technologie au bilan désastreux ou à l'intérêt illusoire (réseaux de neurones inutilement surpuissants, jeux vidéos, technologies de pseudo-confort)
- plus de théorie de l'IA pour atteindre des rapports performance/coût meilleurs: **développer les mathématiques pour l'IA**.
- concentrer la R&D sur les besoins primaires (alimentation, sécurité sanitaire, environnement)

La question de la sobriété numérique

Les (tentatives de) solutions:

- évaluer le bilan carbone de chaque technologie et **conserver les technologies:**
 - ▶ de **bilan carbone négatif** (celles dont les gains surcompensent les coûts): régulation thermique automatique, prévention sociale (médicale, climatique)
 - ▶ de **bilan carbone positif mais de nécessité planétaire** (lutte contre déforestation, transition végétale)
 - ▶ refuser toute technologie au bilan désastreux ou à l'intérêt illusoire (réseaux de neurones inutilement surpuissants, jeux vidéos, technologies de pseudo-confort)
- plus de théorie de l'IA pour atteindre des rapports performance/coût meilleurs: **développer les mathématiques pour l'IA.**
- concentrer la R&D sur les besoins primaires (alimentation, sécurité sanitaire, environnement)
- **rendre le leitmotiv “faire plus avec beaucoup moins” sexy, ringardiser la futilité technologique!**

La question de la sobriété numérique

Quelques chiffres : (source ADEME 2021)

L'activité numérique d'un seul Français occasionne, chaque année:

- un impact sur le changement climatique similaire à un trajet de **2740 km en voiture**;
- la production de **370 kg de déchets** sur l'ensemble des étapes du cycle de vie;
- le **déplacement de 1160 kg de matériaux**.

- 1 Logistique du cours
- 2 Qu'est-ce que l'intelligence artificielle? Histoire et notions de bases
- 3 Des données aux vecteurs: séparabilité, représentations et kNN
- 4 Les réseaux de neurones
- 5 L'IA moderne: révolution informatique et retour des mathématiques
 - La magie des réseaux convolutionnels et antagonistes
 - L'IA de demain: sobriété numérique et éthique
 - Quid des mathématiques?

Ce que l'on sait, ce que l'on sent...

La singularité “IA”:

Ce que l'on sait, ce que l'on sent...

La singularité “IA”:

- les mathématiques appliquées sont très puissantes en ingénierie:
télécoms, mécanique, optique...

Ce que l'on sait, ce que l'on sent...

La singularité “IA”:

- les mathématiques appliquées sont très puissantes en ingénierie: télécoms, mécanique, optique...
 - ▶ s'explique par les modèles physiques simples: **systèmes linéaires** (= produits et sommes)

Ce que l'on sait, ce que l'on sent...

La singularité “IA”:

- les mathématiques appliquées sont très puissantes en ingénierie: télécoms, mécanique, optique...
 - ▶ s'explique par les modèles physiques simples: **systèmes linéaires** (= produits et sommes)
 - ▶ les modèles **non-linéaires** sont beaucoup plus durs à traiter (= sin, cos, exp, sign)

Ce que l'on sait, ce que l'on sent...

La singularité “IA”:

- les mathématiques appliquées sont très puissantes en ingénierie: télécoms, mécanique, optique...
 - ▶ s'explique par les modèles physiques simples: **systèmes linéaires** (= produits et sommes)
 - ▶ les modèles **non-linéaires** sont beaucoup plus durs à traiter (= sin, cos, exp, sign)
- l'IA moderne **doit transformer les données par des opérations non-linéaires complexes**: c'est “le pire scénario possible”

Ce que l'on sait, ce que l'on sent...

La singularité “IA”:

- les mathématiques appliquées sont très puissantes en ingénierie: télécoms, mécanique, optique...
 - ▶ s'explique par les modèles physiques simples: **systèmes linéaires** (= produits et sommes)
 - ▶ les modèles **non-linéaires** sont beaucoup plus durs à traiter (= sin, cos, exp, sign)
- l'IA moderne **doit transformer les données par des opérations non-linéaires complexes**: c'est “le pire scénario possible”
- cela conduit les informaticiens aujourd’hui à penser que l'IA n'est plus une affaire de mathématiques, mais d'informatique

Ce que l'on sait, ce que l'on sent...

La singularité “IA”:

- les mathématiques appliquées sont très puissantes en ingénierie: télécoms, mécanique, optique...
 - ▶ s'explique par les modèles physiques simples: **systèmes linéaires** (= produits et sommes)
 - ▶ les modèles **non-linéaires** sont beaucoup plus durs à traiter (= sin, cos, exp, sign)
- l'IA moderne **doit transformer les données par des opérations non-linéaires complexes**: c'est “le pire scénario possible”
- cela conduit les informaticiens aujourd’hui à penser que l'IA n'est plus une affaire de mathématiques, mais d'informatique
- ce qui pose de nombreux problèmes: effet “boîte noire”, évaluation des performances, contrôle des biais, réduction du bilan carbone!?

Ce que l'on sait, ce que l'on sent...

Les lueurs d'espoir:

Ce que l'on sait, ce que l'on sent...

Les lueurs d'espoir:

- le retour en force des réseaux de neurones **instables** jusqu'en 2010 et soudain **redevenus stables** n'est pas un hasard:

Ce que l'on sait, ce que l'on sent...

Les fléurs d'espoir:

- le retour en force des réseaux de neurones **instables** jusqu'en 2010 et soudain **redevenus stables** n'est pas un hasard:
 - ▶ c'est une conséquence du gigantisme des données et du modèle: le **gigantisme des degrés de liberté** (ou aléa).

Ce que l'on sait, ce que l'on sent...

Les lueurs d'espoir:

- le retour en force des réseaux de neurones **instables** jusqu'en 2010 et soudain **redevenus stables** n'est pas un hasard:
 - ▶ c'est une conséquence du gigantisme des données et du modèle: le **gigantisme des degrés de liberté** (ou aléa).
 - ▶ en probabilités, **la loi des grands nombres** prédit que
“la moyenne de très nombreuses variables aléatoires devient **déterministe, prédictible, stable**”

Ce que l'on sait, ce que l'on sent...

Les auteurs d'espoir:

- le retour en force des réseaux de neurones **instables** jusqu'en 2010 et soudain **redevenus stables** n'est pas un hasard:
 - ▶ c'est une conséquence du gigantisme des données et du modèle: le **gigantisme des degrés de liberté** (ou aléa).
 - ▶ en probabilités, **la loi des grands nombres** prédit que
“la moyenne de très nombreuses variables aléatoires devient **déterministe, prédictible, stable**”
- de nouvelles mathématiques émergent: **statistiques en grandes dimensions, théorie des matrices aléatoires, physique statistique**

Ce que l'on sait, ce que l'on sent...

Les auteurs d'espoir:

- le retour en force des réseaux de neurones **instables** jusqu'en 2010 et soudain **redevenus stables** n'est pas un hasard:
 - ▶ c'est une conséquence du gigantisme des données et du modèle: le **gigantisme des degrés de liberté** (ou aléa).
 - ▶ en probabilités, **la loi des grands nombres** prédit que

"la moyenne de très nombreuses variables aléatoires devient **déterministe, prédictible, stable**"
- de nouvelles mathématiques émergent: **statistiques en grandes dimensions, théorie des matrices aléatoires, physique statistique**
- ces théories permettent de:
 - ▶ casser la difficulté des non-linéarités
 - ▶ transformer l'aléatoire insaisissable en des lois déterministes simples

Ce que l'on sait, ce que l'on sent...

Les auteurs d'espoir:

- le retour en force des réseaux de neurones **instables** jusqu'en 2010 et soudain **redevenus stables** n'est pas un hasard:
 - ▶ c'est une conséquence du gigantisme des données et du modèle: le **gigantisme des degrés de liberté** (ou aléa).
 - ▶ en probabilités, **la loi des grands nombres** prédit que

"la moyenne de très nombreuses variables aléatoires devient **déterministe, prédictible, stable**"
- de nouvelles mathématiques émergent: **statistiques en grandes dimensions, théorie des matrices aléatoires, physique statistique**
- ces théories permettent de:
 - ▶ casser la difficulté des non-linéarités
 - ▶ transformer l'aléatoire insaisissable en des lois déterministes simples
 - ▶ **conséquences**: prédire, comprendre, débiaiser, améliorer la performance, réduire le coût carbone des algorithmes

Ce que l'on sait, ce que l'on sent...

Les lieux d'espoir:

- le retour en force des réseaux de neurones **instables** jusqu'en 2010 et soudain **redevenus stables** n'est pas un hasard:
 - ▶ c'est une conséquence du gigantisme des données et du modèle: le **gigantisme des degrés de liberté** (ou aléa).
 - ▶ en probabilités, **la loi des grands nombres** prédit que

"la moyenne de très nombreuses variables aléatoires devient **déterministe, prédictible, stable**"
- de nouvelles mathématiques émergent: **statistiques en grandes dimensions, théorie des matrices aléatoires, physique statistique**
- ces théories permettent de:
 - ▶ casser la difficulté des non-linéarités
 - ▶ transformer l'aléatoire insaisissable en des lois déterministes simples
 - ▶ **conséquences**: prédire, comprendre, débiaiser, améliorer la performance, réduire le coût carbone des algorithmes
 - ▶ **difficultés restantes**: pas pour tout! les DNN demeurent très compliqués à étudier, ce sont toujours aujourd'hui des boîtes noires!

Pourquoi pas une IA “low-tech” ?

- Dr Valérie d'Acremont, *“Technologies et santé: Quels compromis entre éthique, environnement et climat? Analyse réflexive et expérience de terrain”*

À vous d'inventer la suite!