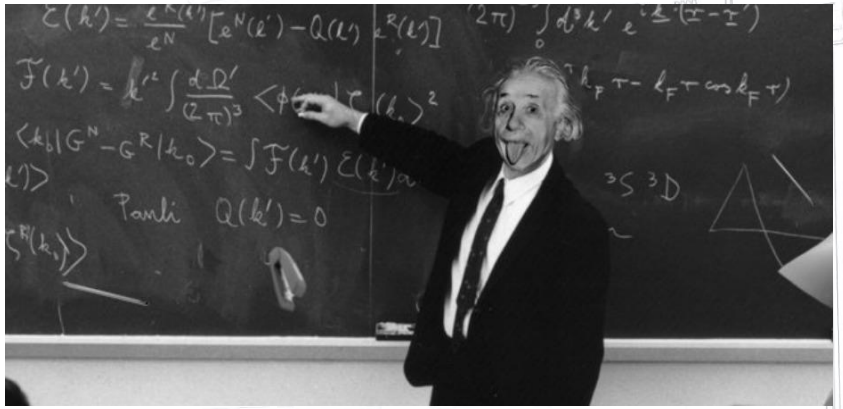
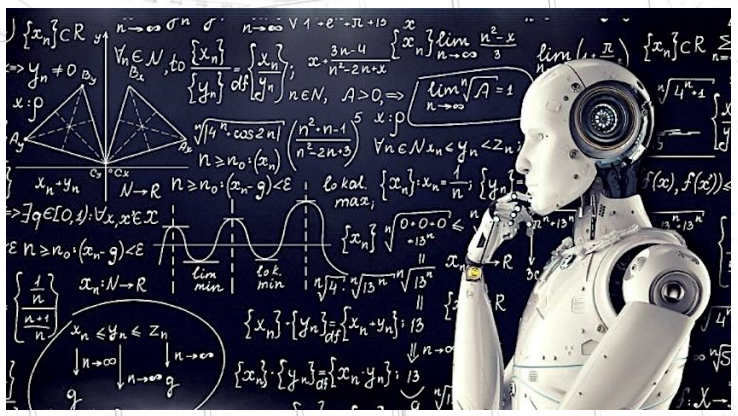
L'intelligence artificielle : un nouvel outil pour la physique ?

Hubert Baty - Observatoire astronomique de Strasbourg

hubert.baty@unistra.fr





Plan général des cours

I. Introduction aux méthodes numériques classiques pour résoudre les lois de la physique

II. C'est quoi l'intelligence artificielle (IA) ? - Exemples

III. Utilisation de l'IA (réseaux de neurones) pour résoudre les lois de la physique

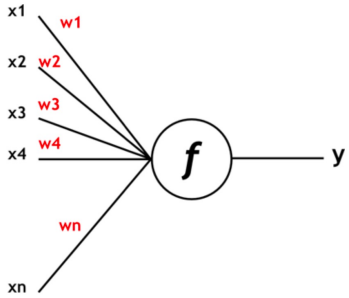
. Retour sur le dernier cours : reconnaissance de chiffres manuscrits

Perceptron ou neurone formel

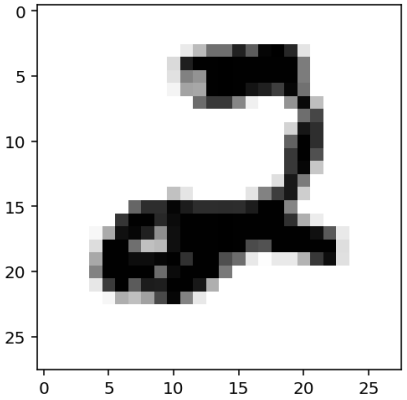
$$y = (w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) + b$$

w_i = poids b = biais

Reproduction du fonctionnement biologique



Le perceptron (Rosenblatt, 1958)

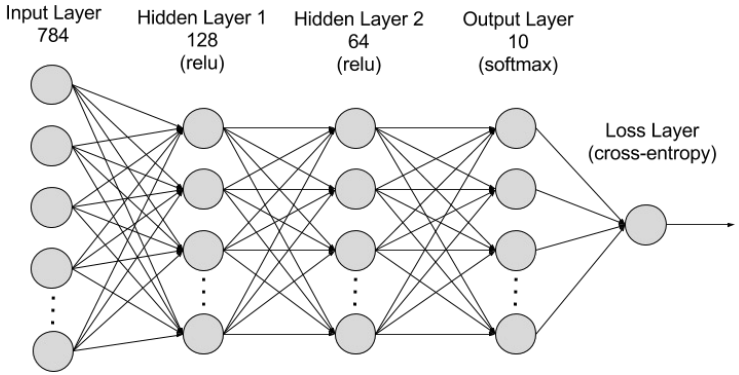


28 par 28 pixels = 784 entrées

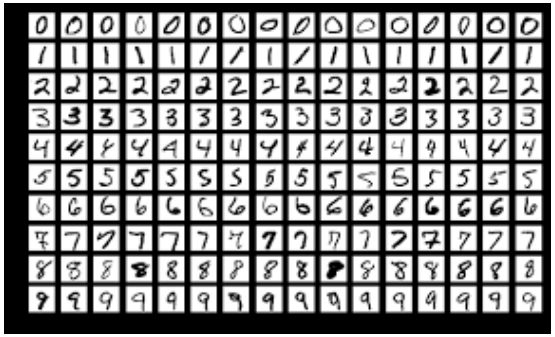
Pixel gris => valeurs intermédiaires entre 0 et 1 (niveaux de gris)
« Valeur » en entrée => ranger dans une classe (de 0 à 9)

. Retour sur le dernier cours : reconnaissance de chiffres manuscrits

1er jeu de données

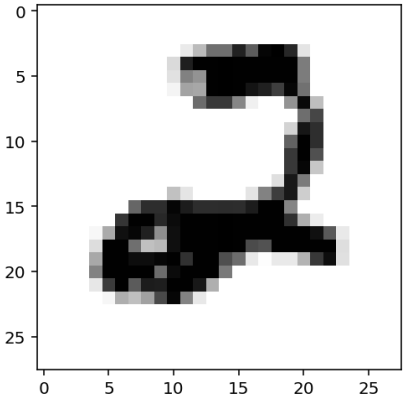


10 classes



Apprentissage (training ou entraînement) supervisé

classification

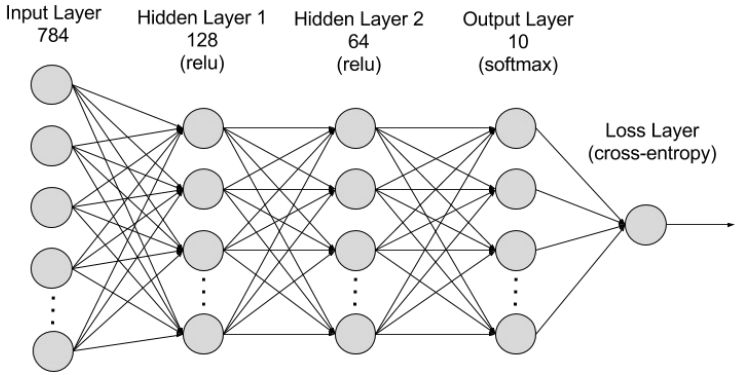


28 par 28 pixels = 784 entrées

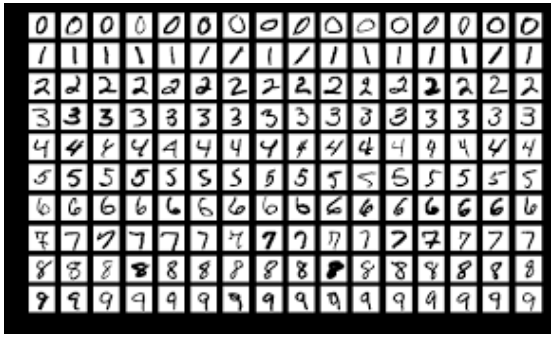
Pixel gris => valeurs intermédiaires entre 0 et 1 (niveaux de gris)
« Valeur » en entrée => ranger dans une classe (de 0 à 9)

. Retour sur le dernier cours : reconnaissance de chiffres manuscrits

1er jeu de données



10 classes



Apprentissage (training) supervisé

classification

Apprentissage <=> Régler de façon itérative tous les poids w_i et les biais b de chaque neurone

Une fois l'apprentissage terminé, une ***relation mathématique*** complexe est obtenue entre l'entrée (valeurs associées aux pixels) et la sortie (les classes)
- Cette relation est l'équivalent d'une formule analytique !

-> On « propage » une information de l'entrée vers la sortie avec le réseau ainsi réglé

. Retour sur le dernier cours : reconnaissance de chiffres manuscrits

Apprentissage

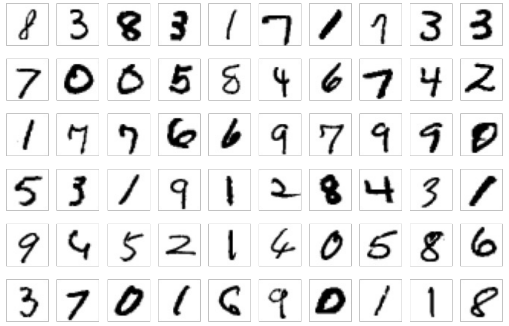
peut prendre du temps …

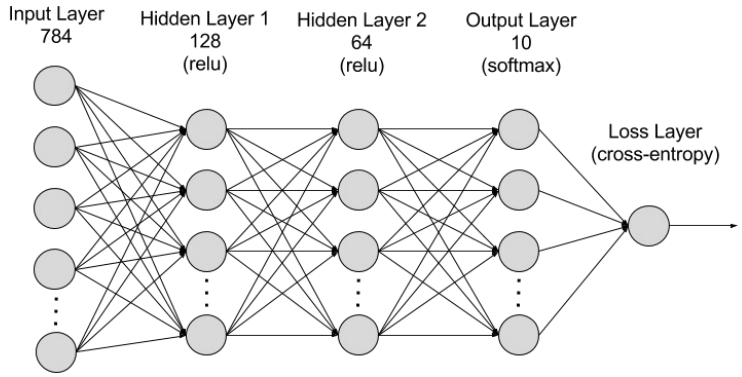
(beaucoup de données et de poids …)

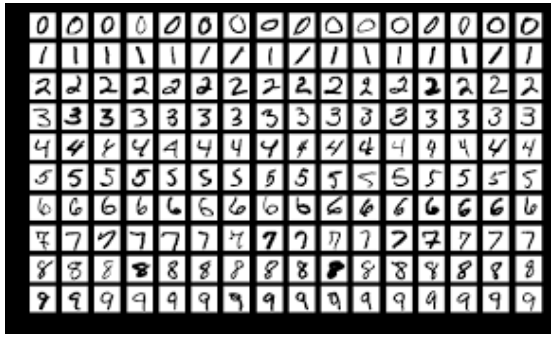
```
Epoch 0 - Training loss: 0.657679927636629
Epoch 1 - Training loss: 0.2827098820565034
Epoch 2 - Training loss: 0.2225405108660205 3
Epoch 3 - Training loss: 0.1791361747964867 2
Epoch 4 - Training loss: 0.1508991499064065 2
Epoch 5 - Training loss: 0.1296386897528190 6
Epoch 6 - Training loss: 0.1150803639323138 8
Epoch 7 - Training loss: 0.100977225213258
Epoch 8 - Training loss: 0.0922013519675349 7
Epoch 9 - Training loss: 0.082864141280351 8
Epoch 10 - Training loss: 0.07524079854077877
Epoch 11 - Training loss: 0.06968157741078325
Epoch 12 - Training loss: 0.064615222324591 4
Epoch 13 - Training loss: 0.05817643330315315
Epoch 14 - Training loss: 0.05458003892324992
```

. Retour sur le dernier cours : reconnaissance de chiffres manuscrits

1er jeu de données





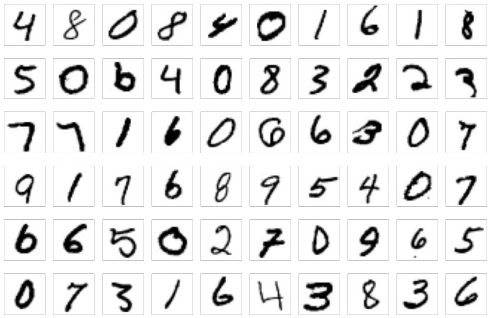


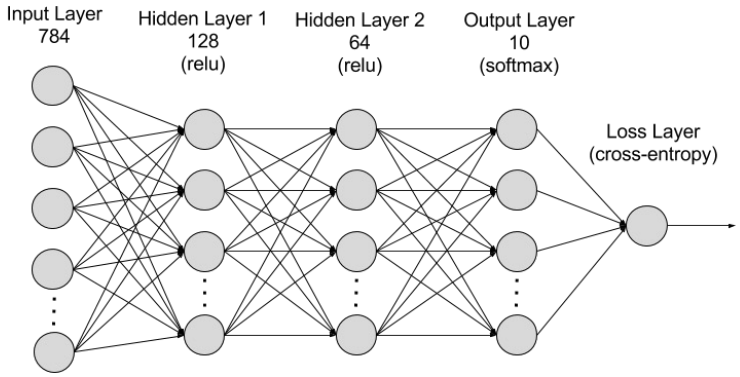
Apprentissage (training) supervisé

classification

Apprentissage <=> Régler de façon itérative tous les poids w_i et les biais b de chaque neurone

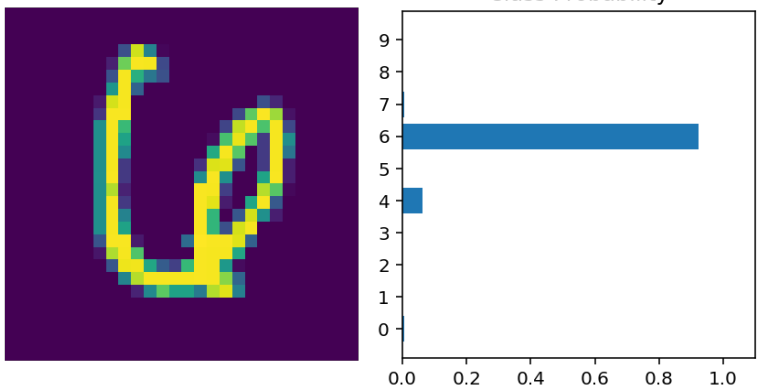
2nd jeu de données

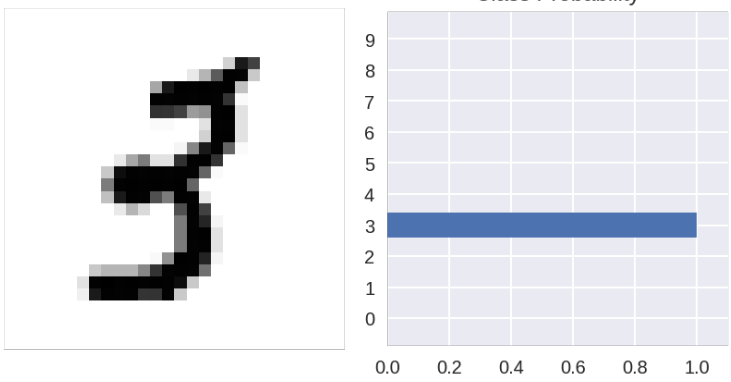




Test (second jeu de données)
=> Fiabilité à x pour cent !

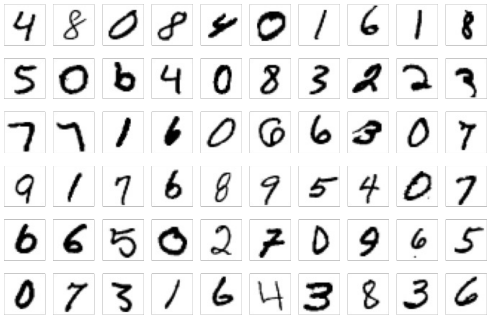
. Retour sur le dernier cours : reconnaissance de chiffres manuscrits

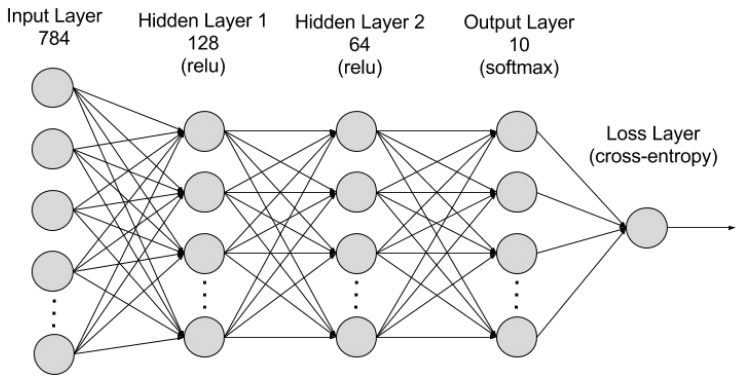




Number Of Images Tested = 10000 Model Accuracy = 0.9742

2nd jeu de données





Test (second jeu de données) => Fiabilité à 97,42 pour cent

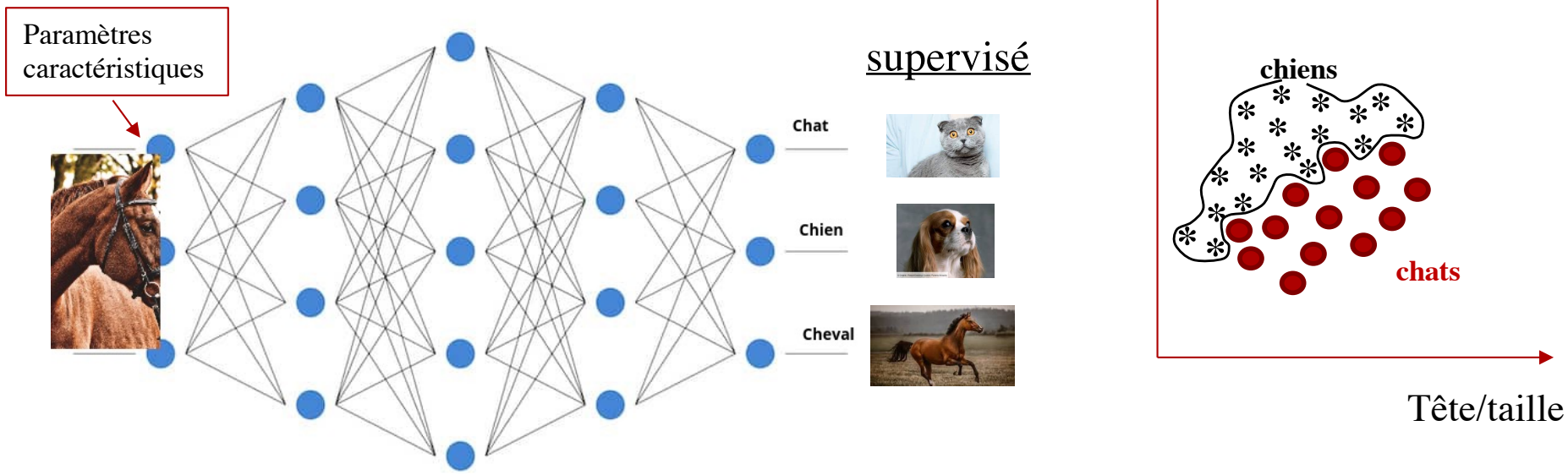
1. L'apprentissage pour la classification avec les réseaux neuronaux

2. Réseaux neuronaux classiques pour approximer des solutions

3. Réseaux neuronaux capables de trouver les solutions avec pas ou très peu de données (aidés par la physique)

1. L'apprentissage pour la classification

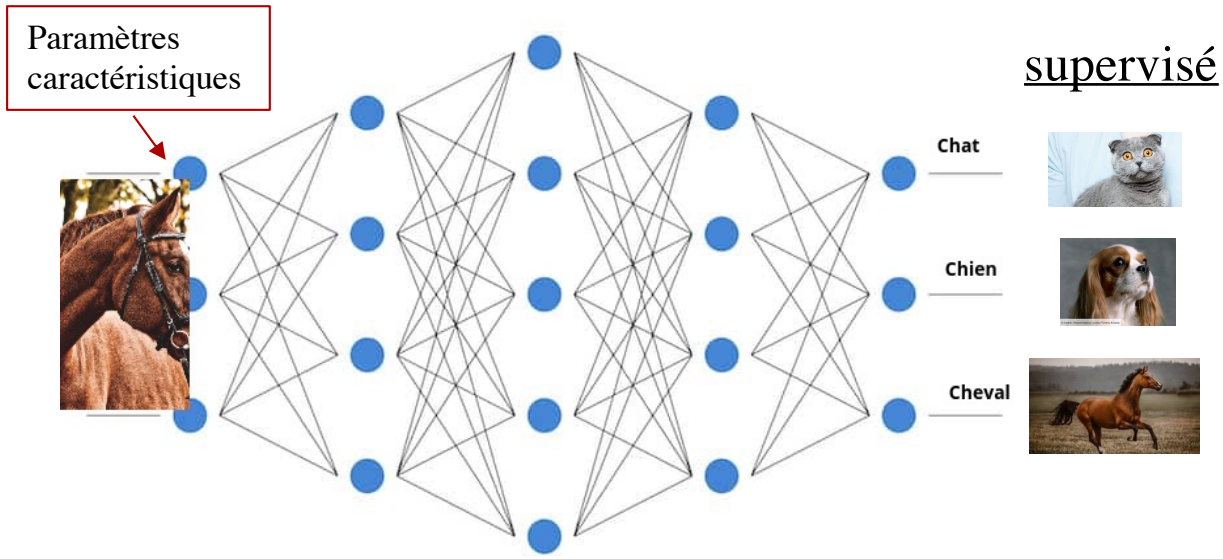
Réseaux de neurones et apprentissage profond -> <u>classification</u>



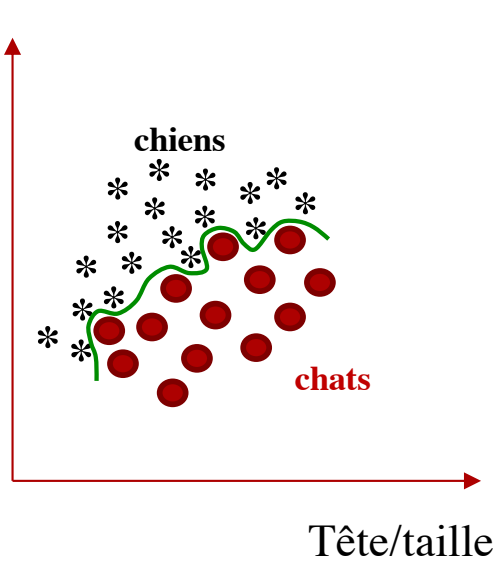
. Définir les classes ⇔définir les régions les caractérisant dans l'espace des paramètres
. **Statistique** => formules de position **moyenne** et des **écarts types** pour chaque classe

1. L'apprentissage pour la classification

Réseaux de neurones et apprentissage profond -> <u>classification</u>



<u>supervisé</u>



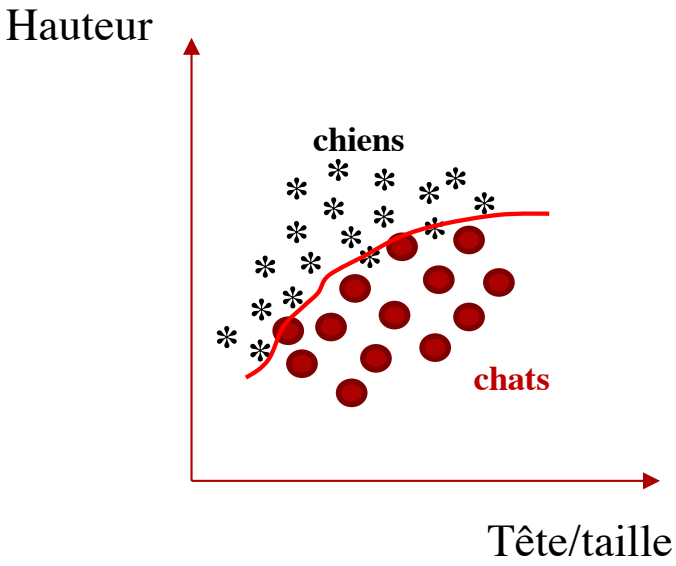
. Définir les classes ⇔définir des courbes séparant les classes

1. L'apprentissage pour la classification

Réseaux de neurones et apprentissage profond -> classification

Méthode itérative : correction à chaque tentative => améliorer la qualité du résultat

1ère tentative non satisfaisante



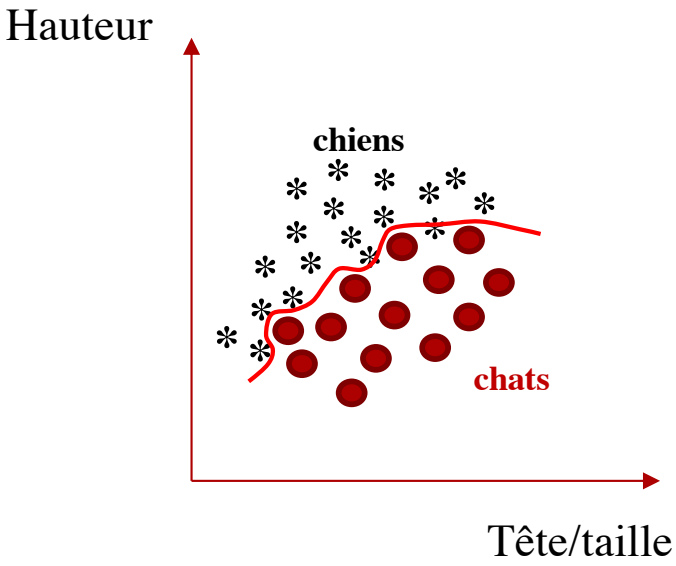
. Définir les classes ⇔définir des courbes séparant les classes

1. L'apprentissage pour la classification

Réseaux de neurones et apprentissage profond -> classification

Méthode itérative : correction à chaque tentative => améliorer la qualité du résultat

2nde tentative non satisfaisante



. Définir les classes ⇔définir des courbes séparant les classes

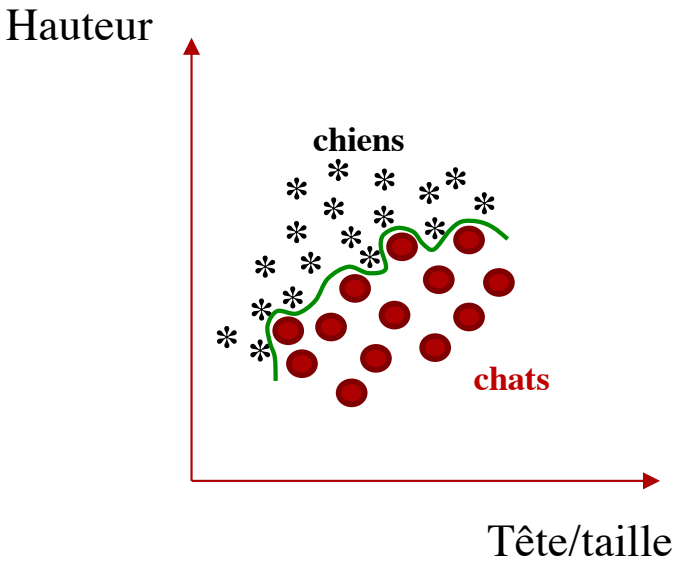
1. L'apprentissage pour la classification

Réseaux de neurones et apprentissage profond -> classification

Méthode itérative : correction à chaque tentative => améliorer la qualité du résultat

3ème tentative satisfaisante

En pratique, il en faut un grand nombre …



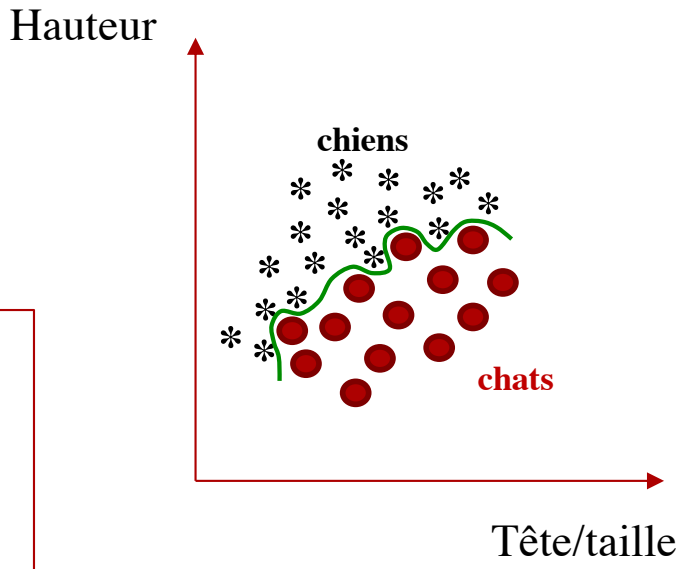
. Définir les classes ⇔définir des courbes séparant les classes

1. L'apprentissage pour la classification

Réseaux de neurones et apprentissage profond -> classification

Méthode itérative : correction à chaque tentative => améliorer la qualité du résultat

3ème tentative satisfaisante



1. Il faut un critère qui détermine la qualité
⇒ La fonction de Loss -> le côut !

2. Il faut une procédure qui corrige les poids de façon à avoir une meilleure Loss (-> plus petite)

. Définir les classes ⇔définir des courbes séparant les classes

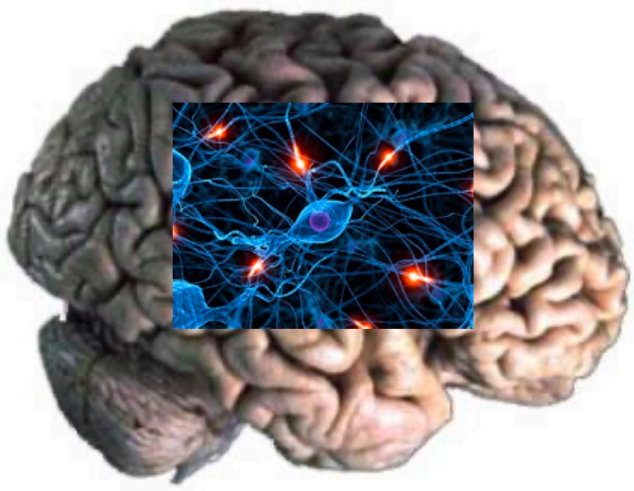
III. Utilisation de l'IA (réseaux de neurones) pour résoudre les lois de la physique

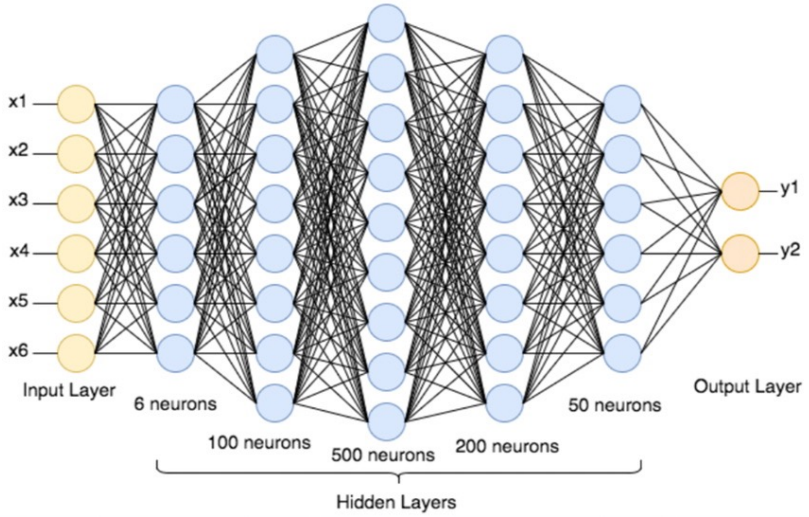
1. L'apprentissage pour la classification avec les réseaux neuronaux

2. Réseaux neuronaux classiques pour approximer des solutions

3. Réseaux neuronaux capables de trouver les solutions avec pas ou très peu de données (aidés par la physique)

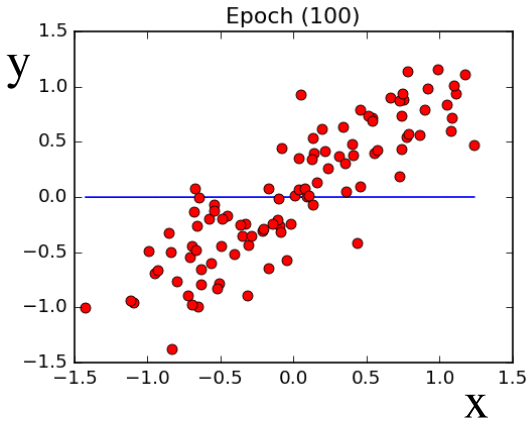
2. Réseaux neuronaux pour l'approximation

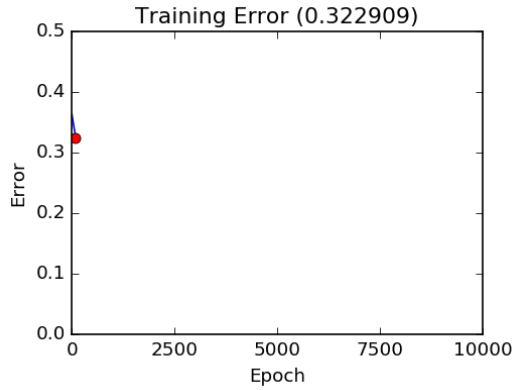




2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression linéaire</u>





Trouver la droite qui passe <u>au plus près</u> des points (de façon itérative)

⇔ **Trouver a et b** tels que y = a x + b

-> 2 coefficients a et b

a = pente

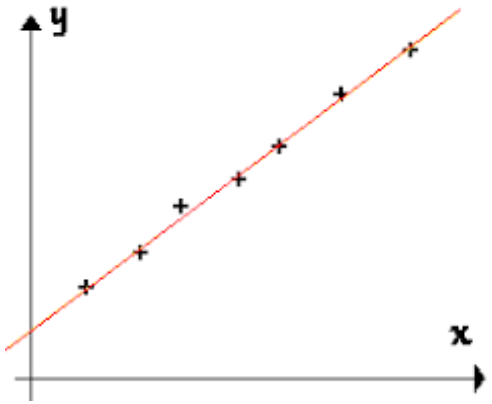
- Utile pour trouver une loi à partir de mesures entachées d'imprécision !

- méthodes connues depuis longtemps sans utiliser les réseaux de neurones (méthodes de moindres carrés)

https://larevueia.fr/regression-lineaire-fonctionnement-et-exemple-avec-python/

2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression linéaire</u>



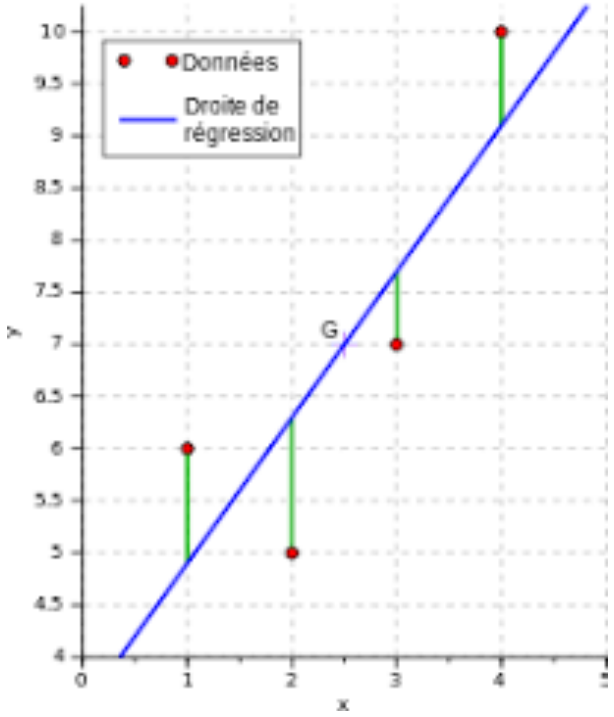
Trouver la droite qui passe au plus près des points

⇔ **Trouver a et b** tels que $y = a\,x + b$

Si ces points sont les positions d'une trajectoire rectiligne
=> Équation de la trajectoire obtenue à partir de quelques points

2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression linéaire</u>



Trouver la droite qui passe au plus près des points : méthodologie ?

⇔ **Trouver a et b** tels que $y = a\,x + b$

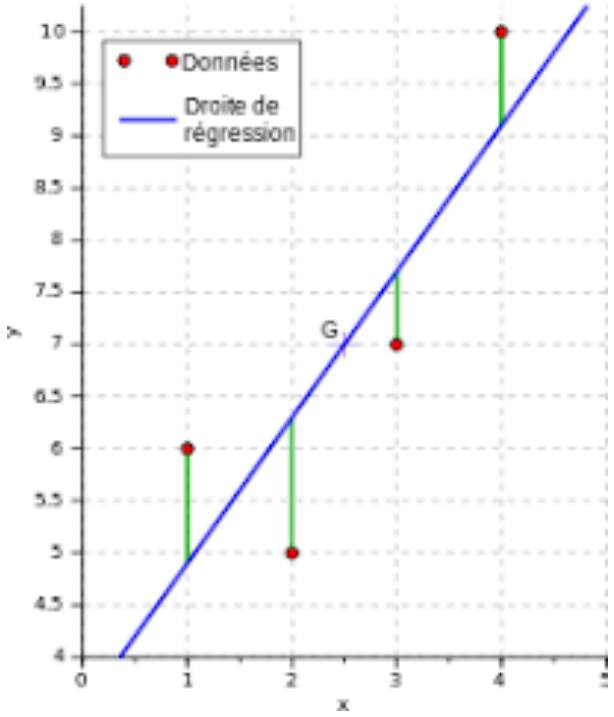
Distances des points à la droite (segments verts)
N : nombre de points (au moins 2 pour une droite)

Loss = (Somme des distance^2) / N
On cherche la Loss la plus petite -> minimum

- Pour la classification on a aussi une autre Loss

2. Réseaux neuronaux classiques pour approximer des solutions

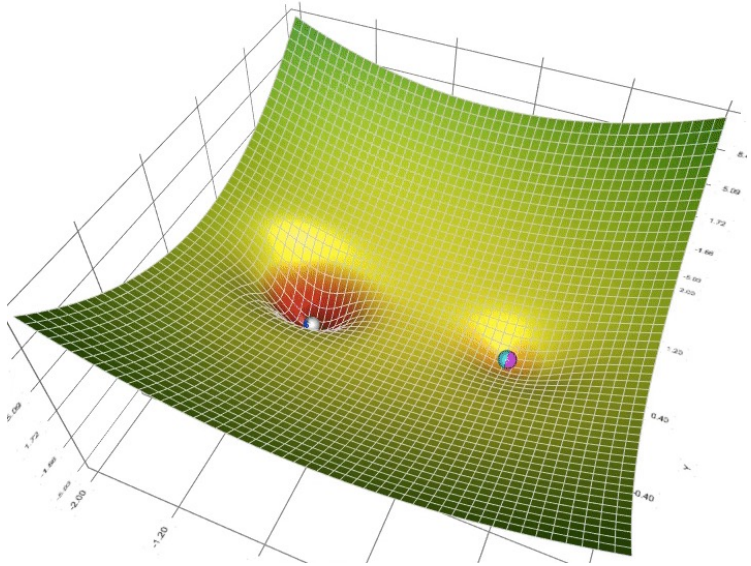
Approximation par <u>régression linéaire</u>



Représentation de la **Loss** en fonction des écarts pour chaque point de donnée = > **hyper-surface**

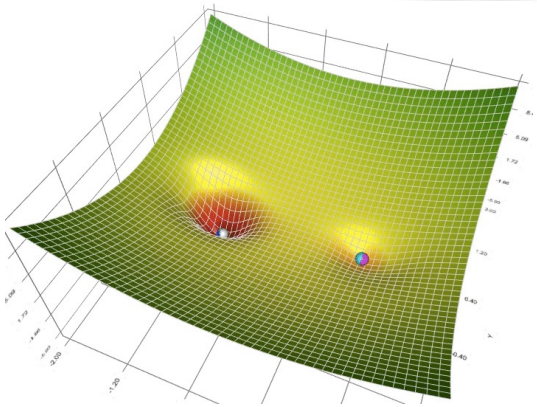
Chaque point ->
<- droite

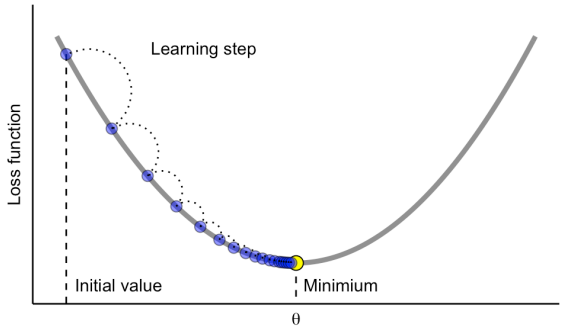
Minimum ->
<- droite optimale



2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression linéaire</u>





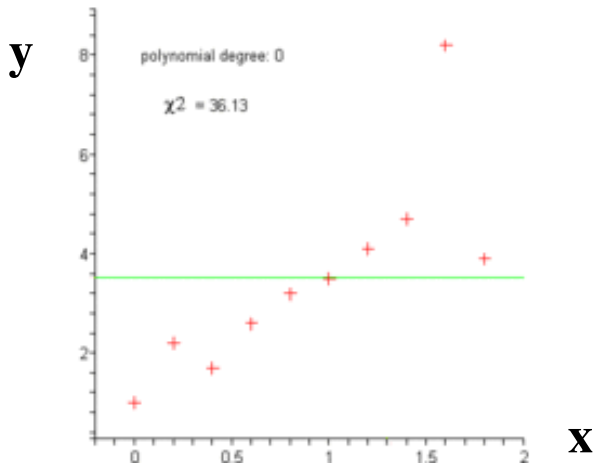
<u>Méthodologie</u> : **Descente de gradient** pour trouver le minimum !

On peut ainsi trouver les coefficients a et b optimaux de façon itérative ….

<u>Formules mathématiques</u> permettant à partir d'une situation initiale de converger vers le minium sont connues …

2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression non linéaire</u>



https://boowiki.info/art/analyse-de-regression/la-regression-non-lineaire.html

<u>Méthodologie</u> : **Descente de gradient** pour trouver le minimum reste valable !

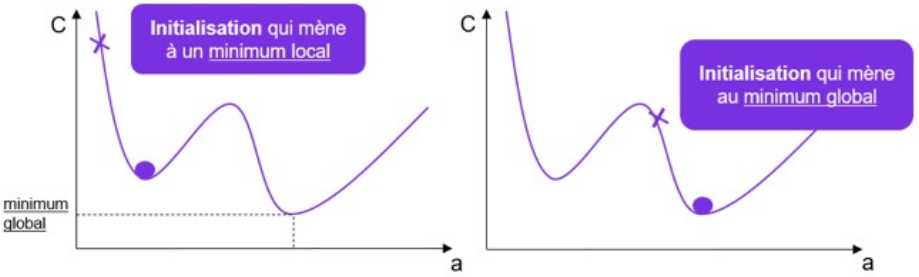
Trouver la courbe qui passe au plus près des points paramétrée par <u>un certain nombre de coefficients</u> …
(il y en a plus que 2, par exemple 3 pour une parabole)

Si ces points sont les positions d'une trajectoire non rectiligne
=> Équation de la trajectoire (mais formule assez complexe)

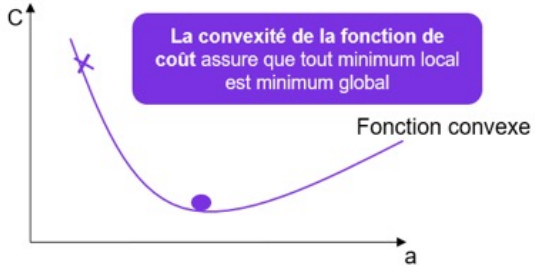
2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression non linéaire</u>

- **Cas d'une fonction de coût non convexe**



- **Cas d'une fonction de coût convexe**



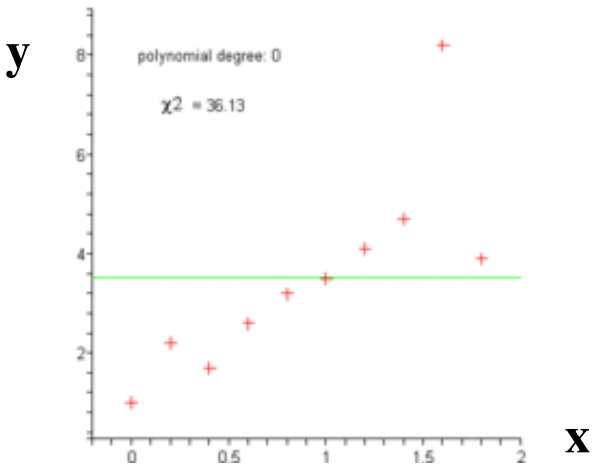
La descente de gradient est plus « périlleuse » …

⇒ c'est un « art » !
Il faut bien choisir la vitesse de descente

- Le point de départ est déterminé de façon aléatoire …

2. Réseaux neuronaux classiques pour approximer des solutions

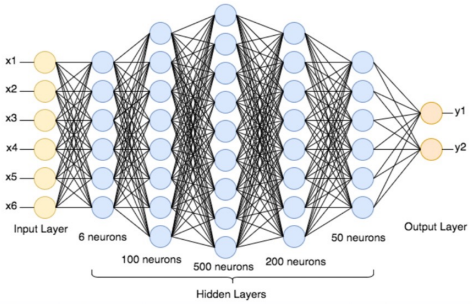
Approximation par <u>régression non linéaire</u>



<u>Méthodologie</u> : **Descente de gradient** avec un réseau de neurones

La descente dépend des poids w_i

Entrées (points)



Sortie (fonction)

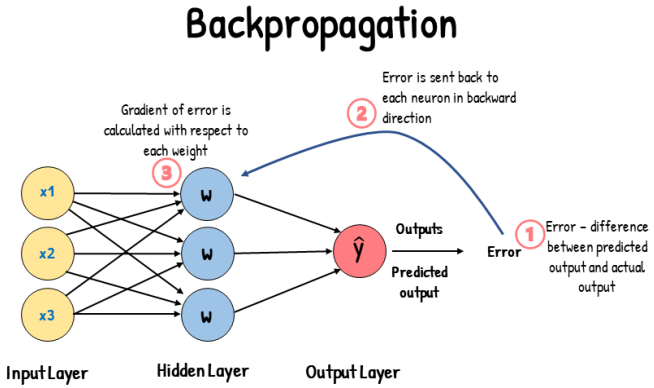
Training
Loss minimale

Le réseau de neurones ne vas pas donner des coefficients (a,b, …) mais un **ajustement de poids** => <u>Relation entre les entrées et la sortie</u> !

2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression non linéaire</u>

La rétro-propagation du gradient



<u>Méthodologie</u> : **Descente de gradient** avec un réseau de neurones

Avant de faire la descente :

- On cherche les contributions respectives des neurones dans la <u>Loss précédente</u> (la formule de descente en tient compte)

- **La descente permet de modifier les poids w_i**

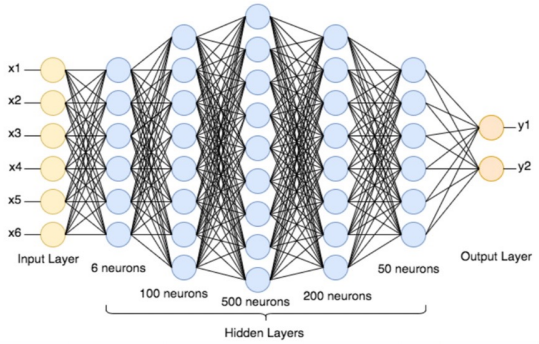
2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>régression non linéaire</u>

<u>Méthodologie</u> : **Descente de gradient** avec un réseau de neurones

Un réseau de neurones est un approximateur non linéaire universel

⇒ Équivalent à une fonction analytique reliant des entrées à une sortie et déterminée par les poids et biais du réseau choisi !

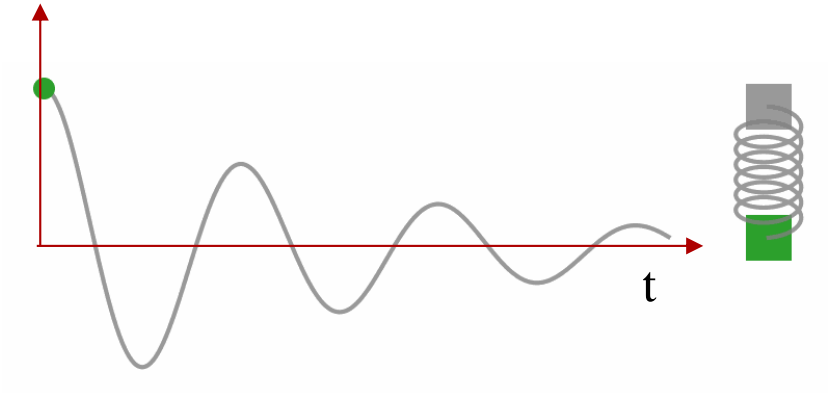


L' apprentissage est <u>supervisé</u> (réseaux neuronaux classique)

2. Réseaux neuronaux classiques pour approximer des solutions

Exemples de solutions -> <u>oscillation d'un ressort</u>

Amplitude en fonction du temps



<u>Solution analytique</u>

-> connue

Cas avec frottements

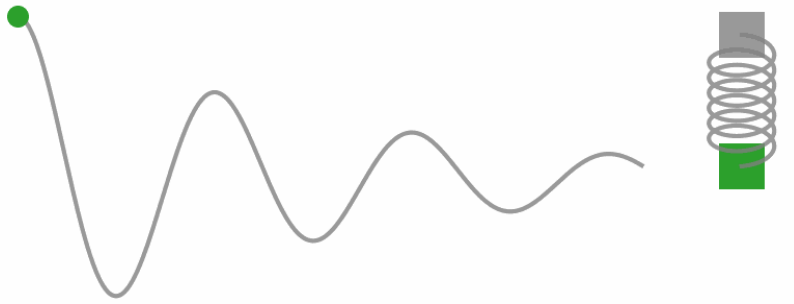
2. Réseaux neuronaux classiques pour approximer des solutions

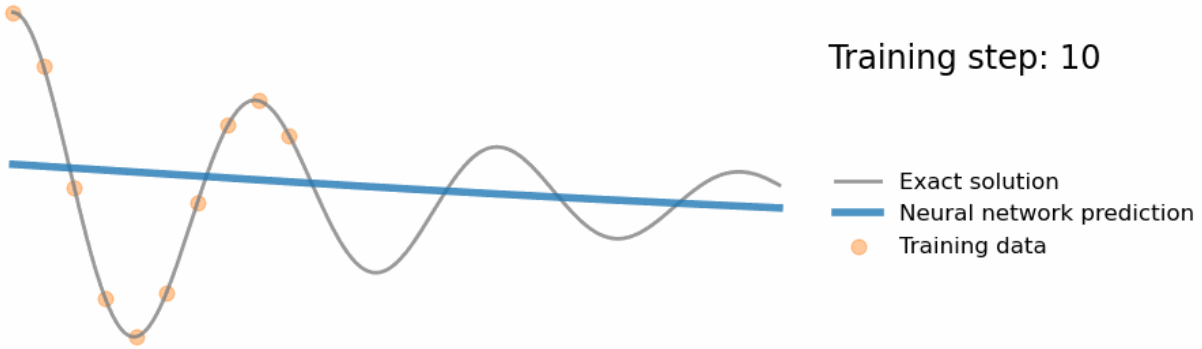
Exemples de solutions -> <u>oscillation d'un ressort</u>

<u>Réseau classique</u> :

A partir de quelques points connues de solutions :

-> approximer la solution

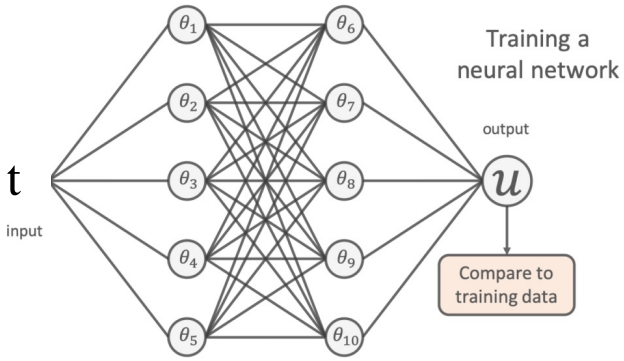


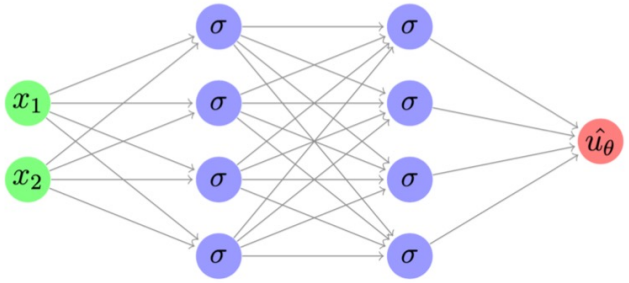


https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/

2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par réseaux neuronaux classiques





Réseau entraîné sur quelques données d'une solution (analytique ou numérique classique)

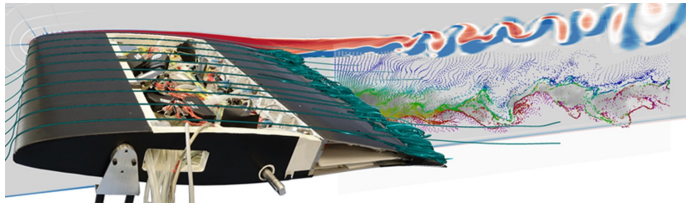
⇒ Permettra de 'prédire' la solution à n'importe quel t mais seulement dans l'intervalle d'entraînement

Réseau entraîné sur les données de plusieurs solutions (par exemple pour plusieurs conditions initiales)

⇒ Permettra de prédire la solution à n'importe quel temps pour différentes conditions initiales
- utile pour éviter de faire de nouvelles simulations (approche classique)

2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>réseaux neuronaux classiques</u>

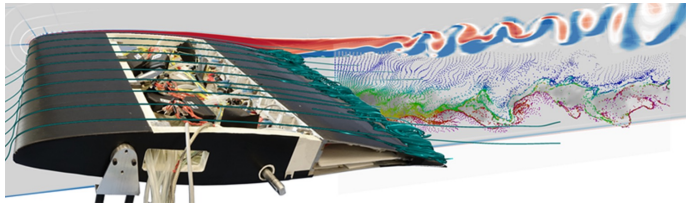


Réseau entraîné sur les <u>données d'une solution</u>
(analytique ou numérique classique)

Pour trouver les formes optimales => nombreuses simulations numériques
(réduire la turbulence autour d'une aile d'avion par exemple)

2. Réseaux neuronaux classiques pour approximer des solutions

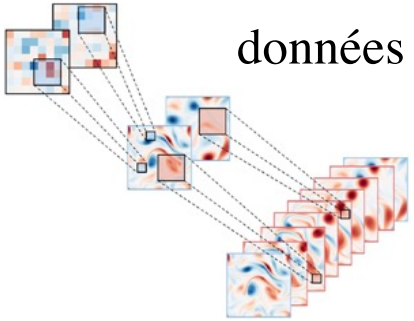
Approximation par <u>réseaux neuronaux classiques</u>



Réseau entraîné sur les <u>données d'une solution</u>
(analytique ou numérique classique)

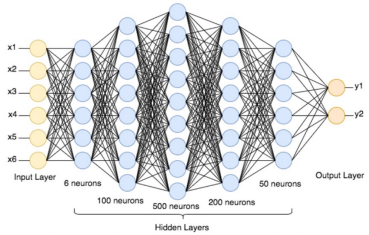
Pour trouver les formes optimales => nombreuses simulations numériques
(réduire la turbulence autour d'une aile d'avion par exemple)

Méthodologie en plein développement dans la recherche industrielle



données

+



⇒ Prédire des solutions sans faire de nouvelles simulations

2. Réseaux neuronaux classiques pour approximer des solutions

Approximation par <u>réseaux neuronaux classiques</u>

<u>Inconvénients</u> => très gourmand en données pour l'apprentissage !

⇒ Devient prohibitif dans certains cas !

<u>Alternative</u> => Réseau peut <u>apprendre à créer lui-même des données supplémentaires</u> d'entraînement à l'aide de la physique ou du PFD (Newton) ?

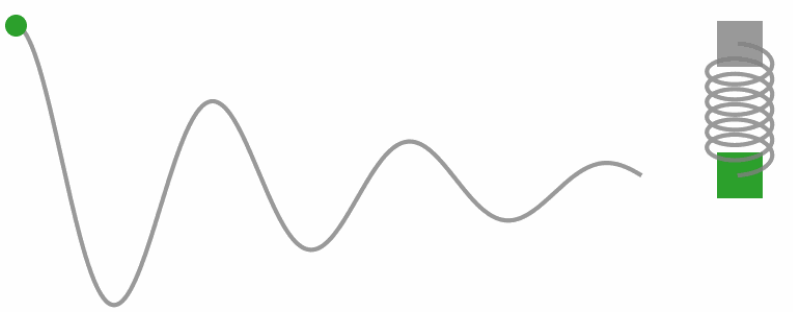
=> <u>Apprentissage non supervisé</u>

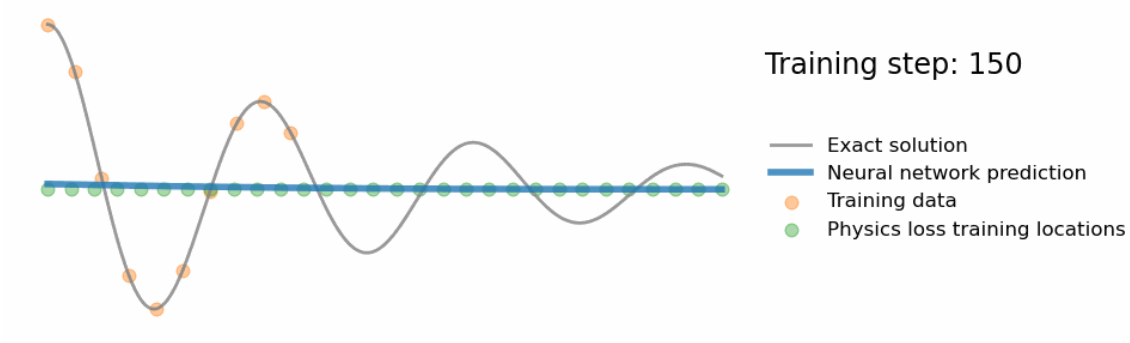
3. Réseaux neuronaux aidés par la physique

Exemples de solutions -> <u>oscillations d'un ressort</u>

<u>Réseau modifié informé (aidé) par la physique</u>

PINNs
Physics Informed Neural Networks





Aux instants dits points de collocations, le réseau peut aussi apprendre à résoudre le PFD !

https://benmoseley.blog/my-research/so-what-is-a-physics-informed-neural-network/

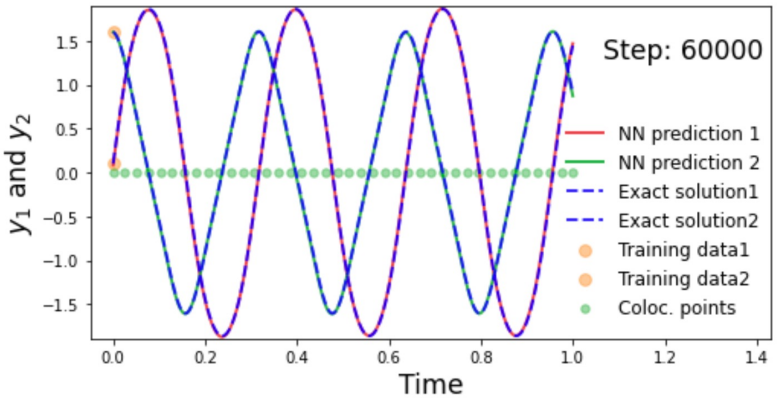
3. Réseaux neuronaux aidés par la physique

Exemples de solutions -> <u>oscillations d'un ressort</u>

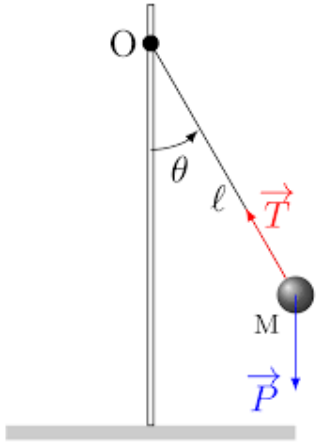
<u>Réseau modifié informé (aidé) par la physique</u>

PINNs
Physics Informed Neural Networks

Cas du pendule sans frottement (amplitude et vitesse)



Les données initiales peuvent même suffire …



Voir Baty & Baty -> https://arxiv.org/pdf/2302.12260.pdf

3. Réseaux neuronaux aidés par la physique

Exemples de solutions -> <u>oscillations d'un ressort</u>

<u>Réseau modifié informé (aidé) par la physique</u>

PINNs
Physics Informed Neural Networks

Comment ça marche ?

Les mêmes outils mathématiques qui permettent de mettre à jour les poids du réseau avec la descente de gradient

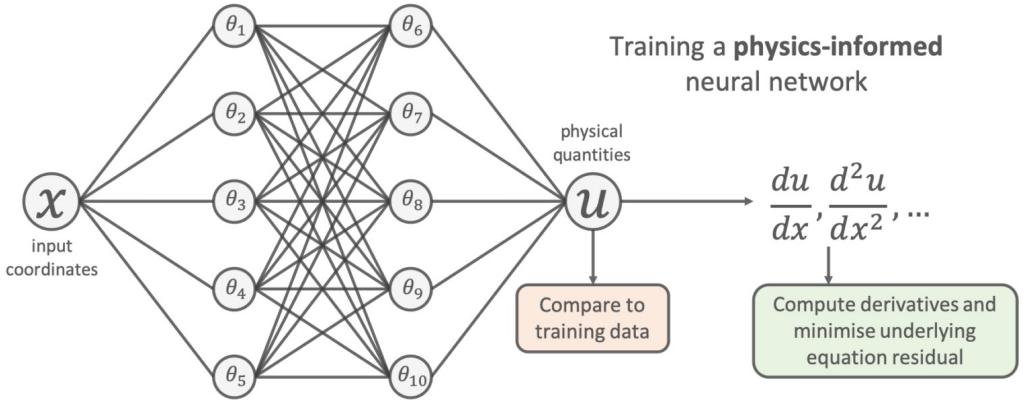
=> Ajoutant une seconde Loss qui estime <u>l'écart à la résolution exacte du PFD</u>

PFD -> somme des **forces = masse * accélération**

3. Réseaux neuronaux aidés par la physique

Exemples de solutions -> <u>oscillations d'un ressort</u>

Réseau modifié informé (aidé) par la physique



PINNs
Physics Informed Neural Networks

Loss1 -> mesure l'écart entre les données connues et celles évaluées

Loss2 -> mesure l'écart entre 0 et solution évaluée

Loss = Loss1 + Loss2

PFD -> somme des **forces - masse * accélération = 0**

3. Réseaux neuronaux aidés par la physique

Exemples de solutions -> <u>oscillations d'un ressort</u>

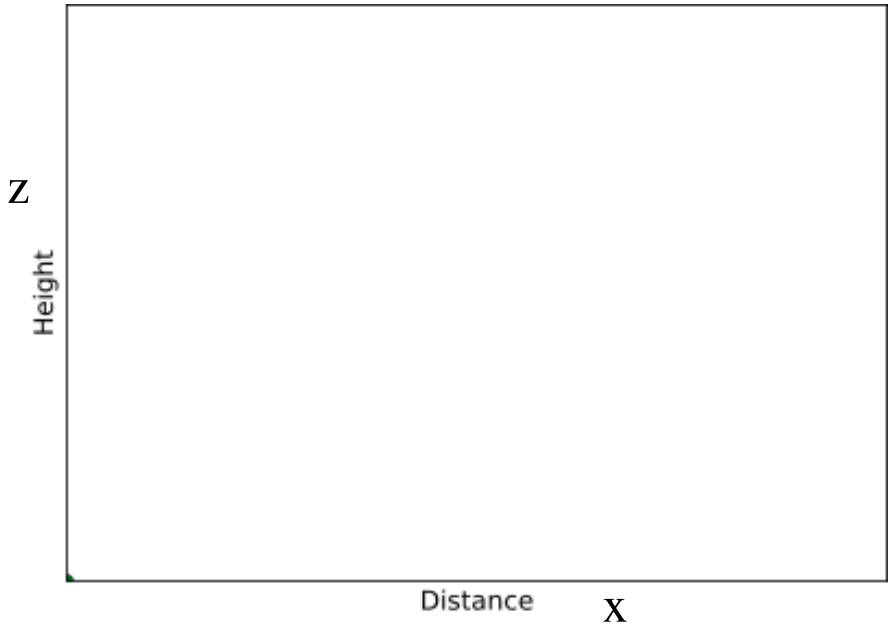
<u>Réseau modifié informé (aidé) par la physique</u>

PINNs
Physics Informed Neural Networks

Place aux DEMOS

3. Réseaux neuronaux aidés par la physique

Exemples de problèmes -> <u>trajectoires dans le champ de pesanteur</u>



ici $z_0 = 0$

Tiré de wikipédia

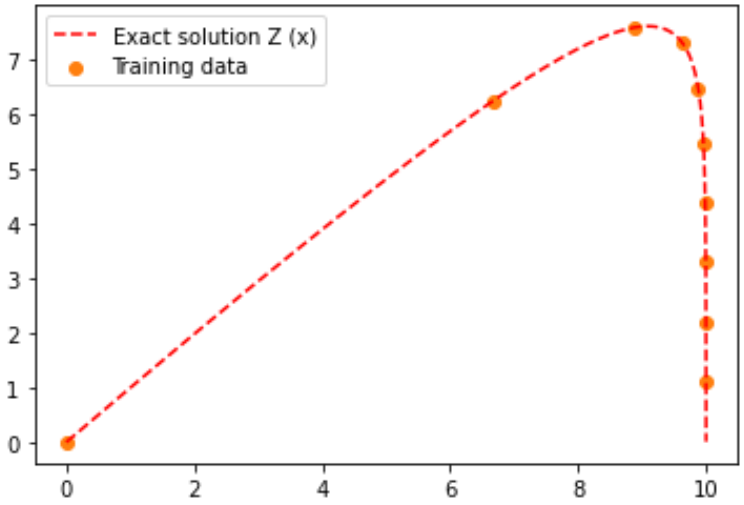
Sans frottements versus
avec frottements (force supplémentaire)

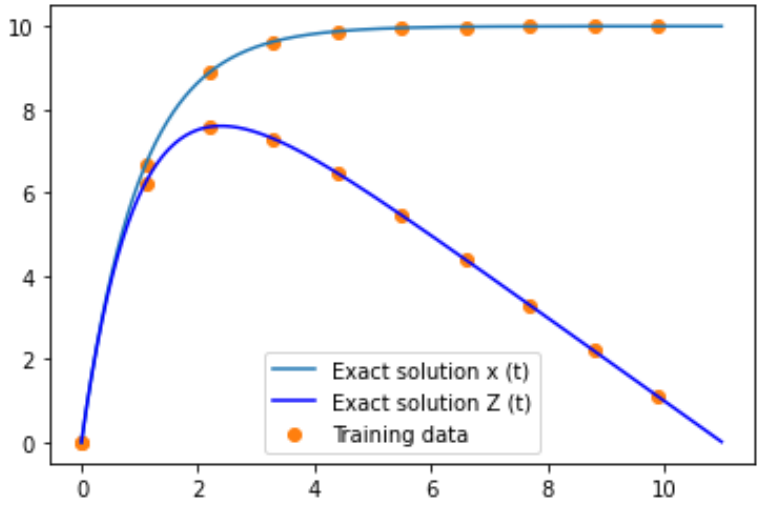
- solutions analytiques existent à base de fonctions exponentielles

- solutions numériques classiques aussi

3. Réseaux neuronaux aidés par la physique

Exemples de problèmes -> <u>trajectoires dans le champ de pesanteur</u>

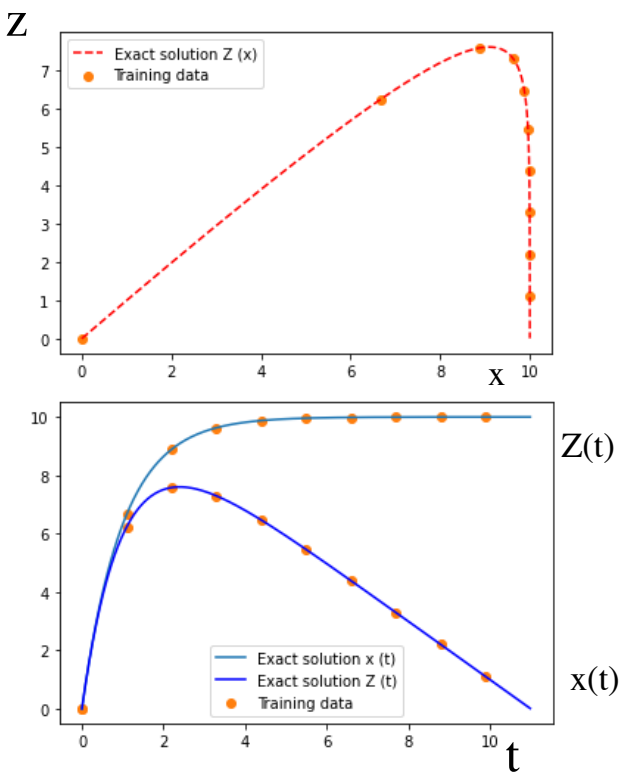




Avec Force frottement = - k V

3. Réseaux neuronaux aidés par la physique

Exemples de problèmes -> <u>trajectoires dans le champ de pesanteur</u>

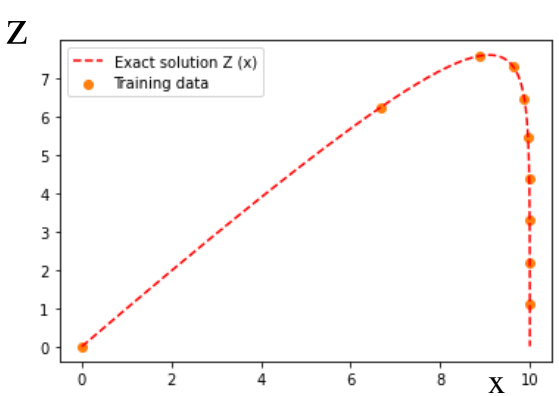


<u>Réseau classique</u> (supervisé)

Peu de données d'entraînement

3. Réseaux neuronaux aidés par la physique

Exemples de problèmes -> <u>trajectoires dans le champ de pesanteur</u>

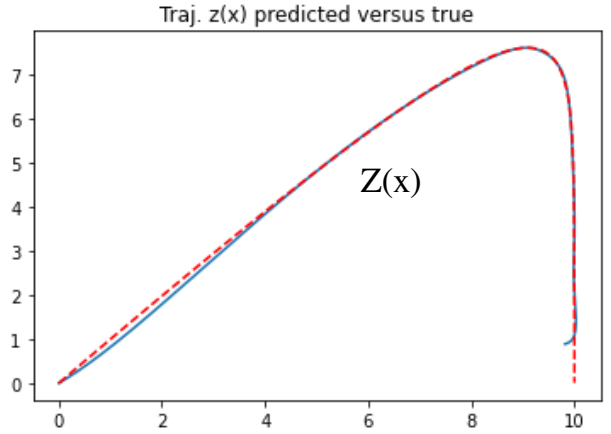


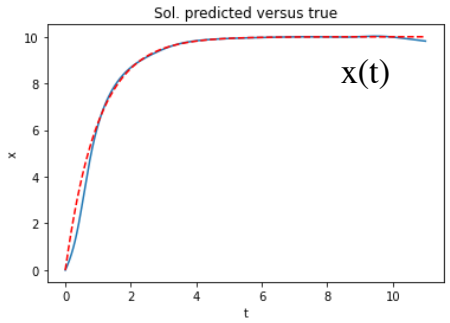
<u>Réseau classique</u>

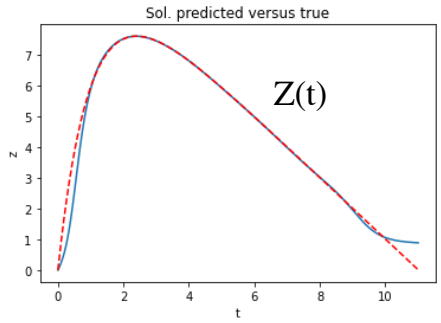
Peu de données d'entraînement

=>

Peu satisfaisant

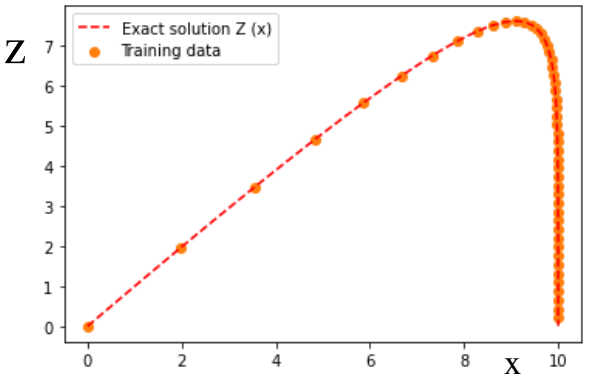


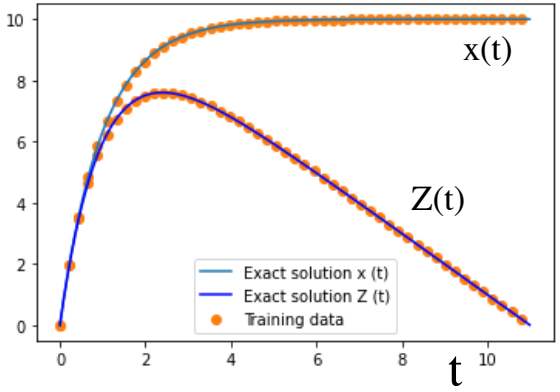




3. Réseaux neuronaux aidés par la physique

Exemples de solutions -> trajectoires dans le champ de pesanteur



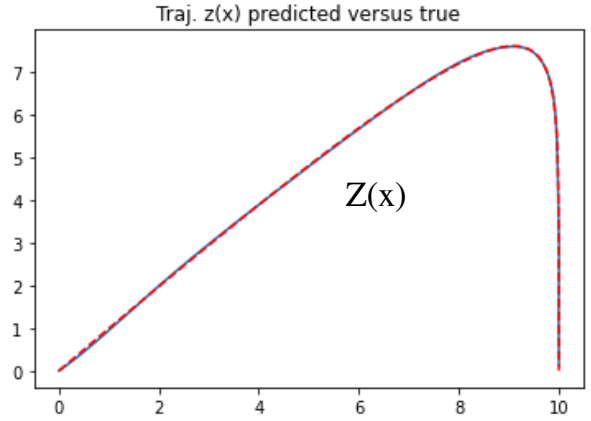


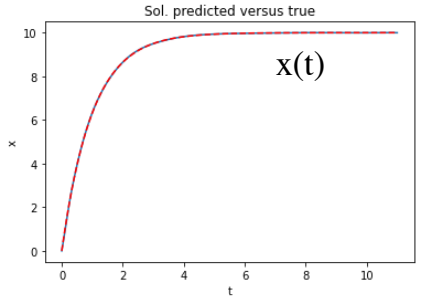
Réseau classique

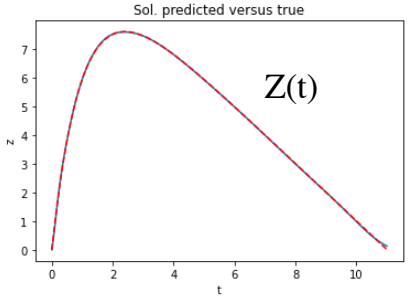
Beaucoup de données d'entraînement

=>

Satisfaisant

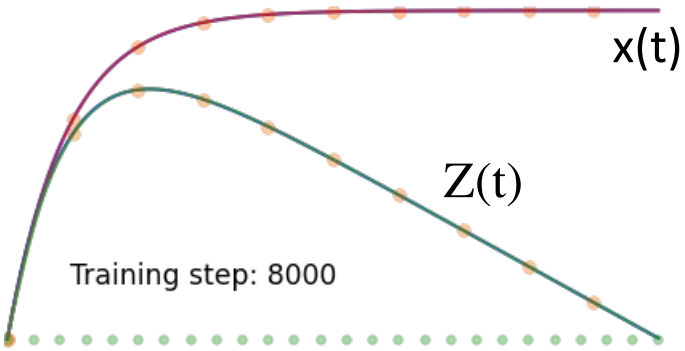






3. Réseaux neuronaux aidés par la physique

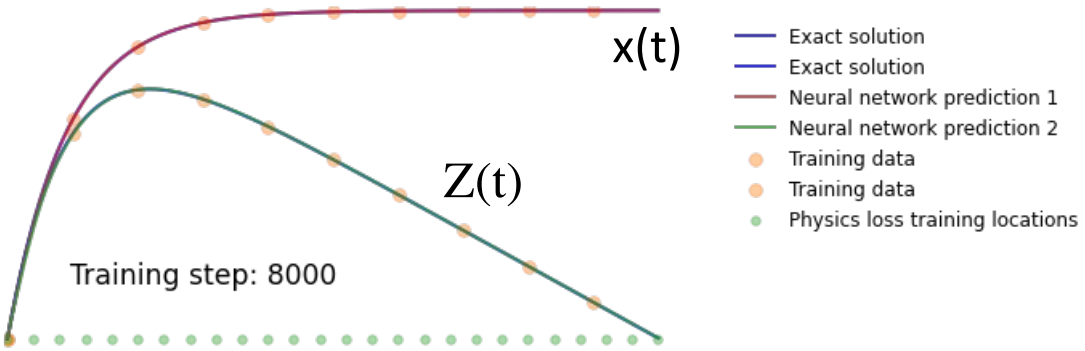
Approximation par <u>réseaux neuronaux informés par la physique</u>



Apprentissage peu supervisé avec PINNs

3. Réseaux neuronaux aidés par la physique

Approximation par <u>réseaux neuronaux informés par la physique</u>

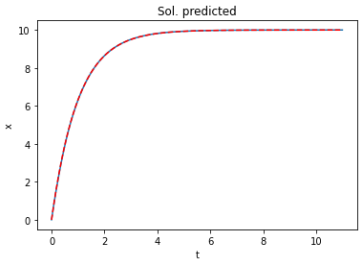


Apprentissage peu supervisé
avec PINNs

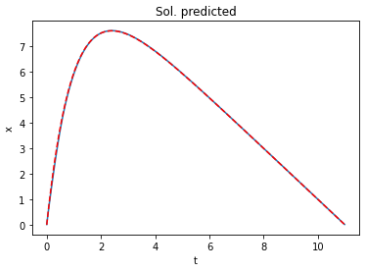
Très satisfaisant et peu
de points de collocation suffisent !

Demo ?

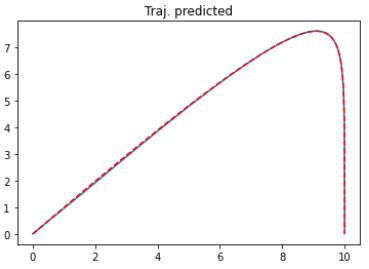
x(t)



Z(t)



Z(x)

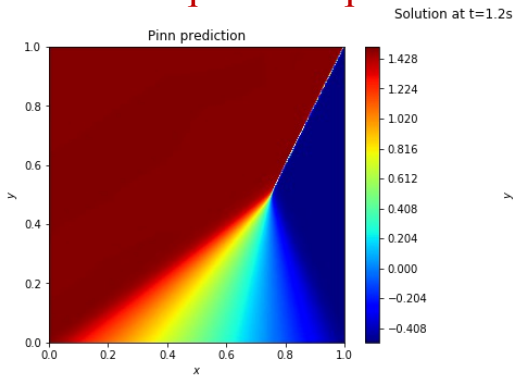


3. Réseaux neuronaux aidés par la physique

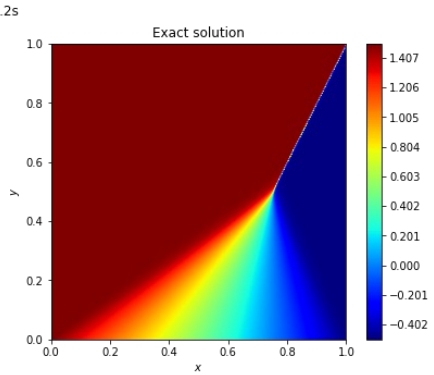
Approximation par <u>réseaux neuronaux informés par la physique</u>

- Les PINNs fonctionnent t'ils sur des problèmes plus compliqués ?

Numérique classique



PINNs



1. Problème numéro 1

Données seulement pour les conditions initiales et aux bords :
-> <u>aucune donnée ailleurs n'est nécessaire</u>

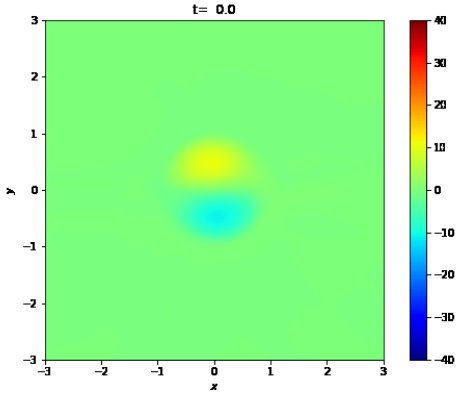
M2 internship of Vincent Italiano -> E. Franck, V. Michel-Dansac & V. Vigon
https://irma.math.unistra.fr/~franck/cours/supervision/rapport/Italiano-Vincent.

3. Réseaux neuronaux aidés par la physique

Approximation par <u>réseaux neuronaux informés par la physique</u>

- Les PINNs fonctionnent t'ils sur des problèmes plus compliqués ?

PINNs



1. Problème numéro 2 : instabilité MHD physique

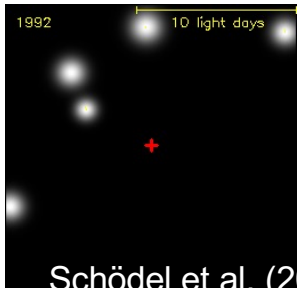
Pour reproduire la solution, quelques données issues d'une simulation numérique classique ont été hélas nécessaires …

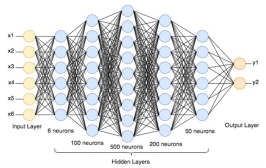
M2 internship of Vincent Italiano -> E. Franck, V. Michel-Dansac & V. Vigon
https://irma.math.unistra.fr/~franck/cours/supervision/rapport/Italiano-Vincent.

3. Réseaux neuronaux aidés par la physique

De nombreuses applications intéressantes possibles en physique

Données d'observations





Réseau

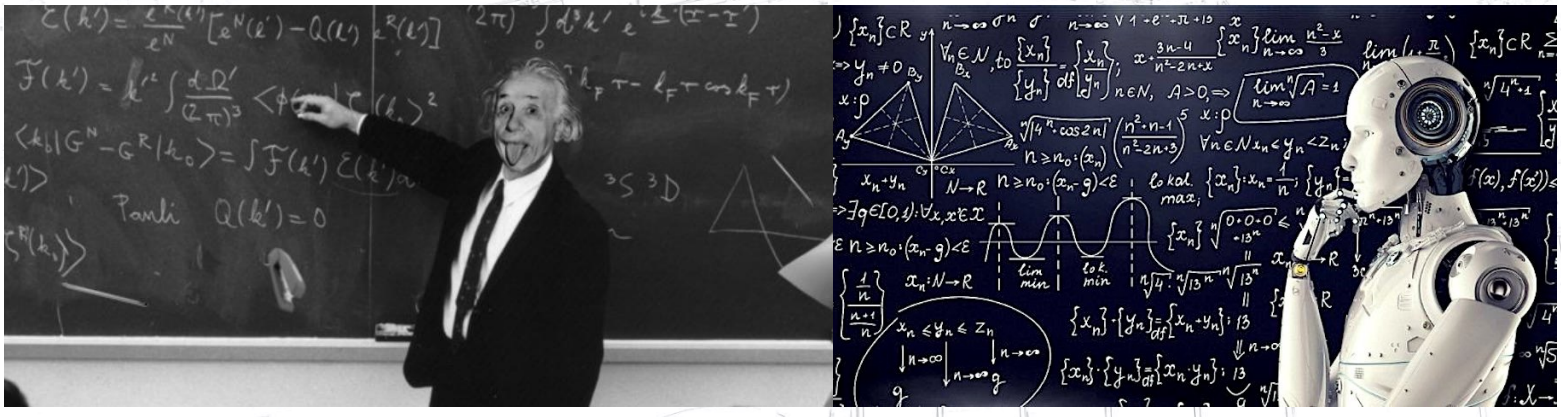
PINNs
Physics Informed Neural Networks

Réseaux neuronaux peuvent aussi 'découvrir' des lois physiques (déduire les forces)

- à partir de données d'observations
- et résoudre des lois PFD

⇒ Trouver la force de gravitation ?
- théorie MOND (Newton modifiée) ou masse manquante ?

L'intelligence artificielle : un nouvel outil pour la physique ?



- **Résoudre les équations à la façon d'un humain => oui mais peut mieux faire !**
- **Trouver des lois => oui mais besoin quand même de la forme générale …**

=> Pour l'instant c'est une alternative intéressante au calcul numérique traditionnel

MERCI C'EST FINI !

Merci à Thierry pour l'organisation