

# Une brève introduction à l'Intelligence Artificielle

## Agents reflexes - Apprentissage

September 5, 2023



CentraleSupélec

# Sommaire

## 1. Agents reflexes - Apprentissage

- Apprentissage automatique

## 2. Evaluation de modèles

- Métriques de performance
- Ré-échantillonnage
- Choix des paramètres

# Agents reflexes - Apprentissage

# Les agents reflexes

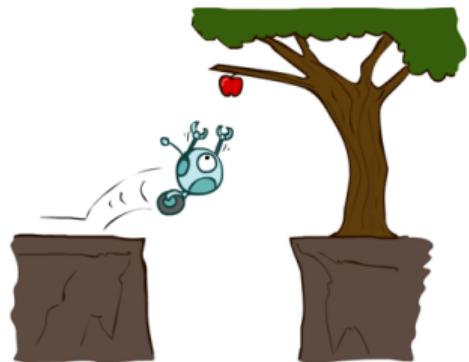


Figure: Source : CS188 Intro to AI - Berkeley

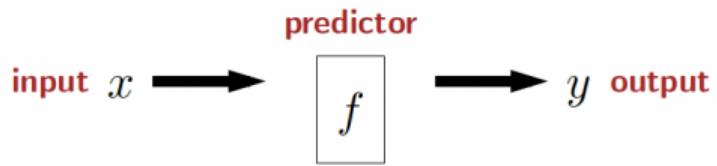


Figure: Source : CS221 Intro to AI - Stanford

# Les agents reflexes

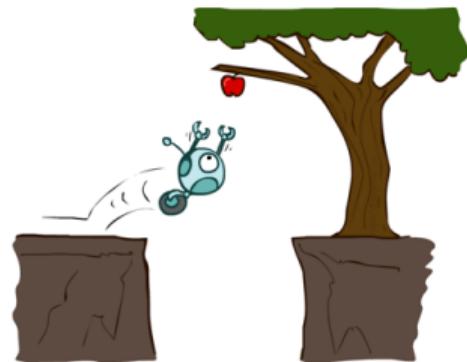


Figure: Source : CS188 Intro to AI - Berkeley

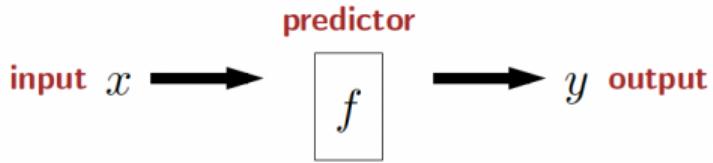


Figure: Source : CS221 Intro to AI - Stanford

- Sortie : choix de l'action (une simple action) à faire (la décision) en se basant sur la perception courante.

# Les agents reflexes

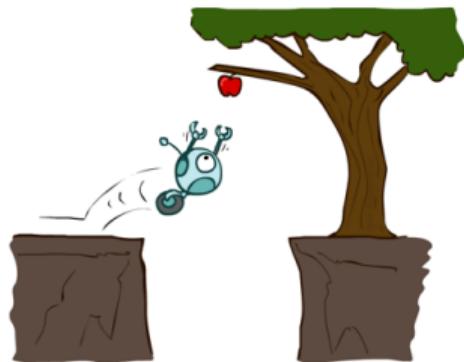


Figure: Source : CS188 Intro to AI - Berkeley

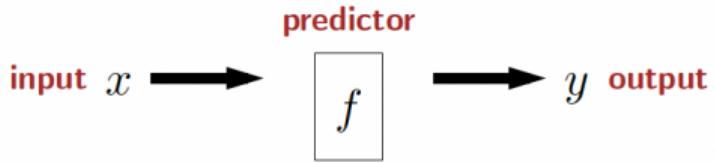


Figure: Source : CS221 Intro to AI - Stanford

- Sortie : choix de l'action (une simple action) à faire (la décision) en se basant sur la perception courante.
- Considère le monde comme il est sans prendre en compte les conséquences de ses actions sur le monde.

# Les agents reflexes

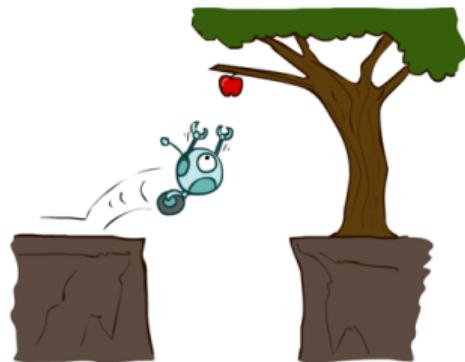


Figure: Source : CS188 Intro to AI - Berkeley

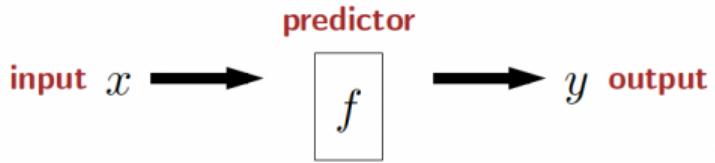


Figure: Source : CS221 Intro to AI - Stanford

- Sortie : choix de l'action (une simple action) à faire (la décision) en se basant sur la perception courante.
- Considère le monde comme il est sans prendre en compte les conséquences de ses actions sur le monde.
- Effectue une séquence déterminée de calculs sur une entrée donnée (*feedforward*)

# Les agents reflexes

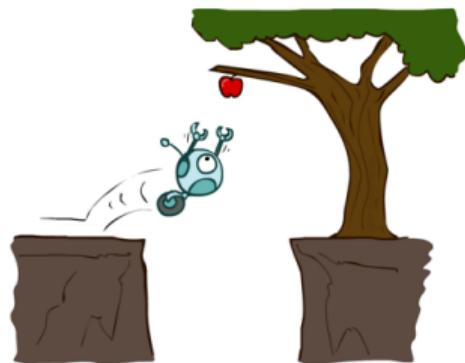


Figure: Source : CS188 Intro to AI - Berkeley

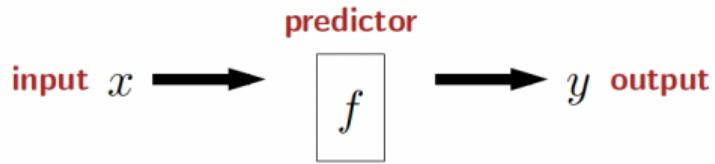


Figure: Source : CS221 Intro to AI - Stanford

- Sortie : choix de l'action (une simple action) à faire (la décision) en se basant sur la perception courante.
- Considère le monde comme il est sans prendre en compte les conséquences de ses actions sur le monde.
- **Effectue une séquence déterminée de calculs sur une entrée donnée (feedforward)**
- Exemples de modèles : la plupart des modèles d'apprentissage automatique : des classifieurs linéaires simples aux réseaux neuronaux profonds.

# Les agents reflexes

Modèles par apprentissage

# Les agents reflexes

## Modèles par apprentissage

- Point de départ :
  - Un ensemble d'**exemples** décrivant partiellement le comportement attendu du système : les données.
  - Un programme simple dont on ne connaît pas les paramètres : choix du type de modèle, l'espace d'hypothèses.

# Les agents reflexes

## Modèles par apprentissage

- Point de départ :
  - Un ensemble d'**exemples** décrivant partiellement le comportement attendu du système : les données.
  - Un programme simple dont on ne connaît pas les paramètres : choix du type de modèle, l'espace d'hypothèses.
- L'algorithme d'apprentissage apprend les paramètres du programme à partir des exemples de manière à reproduire au mieux le comportement du système.

# Les agents reflexes

## Modèles par apprentissage

- Point de départ :
  - Un ensemble d'**exemples** décrivant partiellement le comportement attendu du système : les données.
  - Un programme simple dont on ne connaît pas les paramètres : choix du type de modèle, l'espace d'hypothèses.
- L'algorithme d'apprentissage apprend les paramètres du programme à partir des exemples de manière à reproduire au mieux le comportement du système.
- La complexité n'est pas dans la conception du programme en lui-même.

# Les agents reflexes

## Modèles par apprentissage

- Point de départ :
  - Un ensemble d'**exemples** décrivant partiellement le comportement attendu du système : les données.
  - Un programme simple dont on ne connaît pas les paramètres : choix du type de modèle, l'espace d'hypothèses.
- L'algorithme d'apprentissage apprend les paramètres du programme à partir des exemples de manière à reproduire au mieux le comportement du système.
- La complexité n'est pas dans la conception du programme en lui-même.
- **Point important : la généralisation**, i.e. la capacité du programme à fonctionner sur de nouveaux exemples.

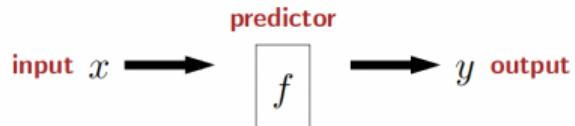


Figure: Source : cs221 course Stanford - Liang

# Qu'est-ce que l'apprentissage supervisé ? (plus formellement)

On considère un espace d'entrée  $\mathcal{X} \subseteq \mathbb{R}^d$  et un espace de sortie  $\mathcal{Y} \subset \mathbb{R}$ .

- **Hypothèse fondamentale** : Les paires d'exemples  $(x, y) \in \mathcal{X} \times \mathcal{Y}$  sont *identiquement et indépendamment distribuées* suivant une distribution de probabilité jointe fixe mais inconnue  $\mathbb{P}(X, Y)$ .
  - i.e. **Il existe une relation entre les entrées et les sorties.**
- **Echantillons** : On observe une séquence de  $m$  paires  $(x_i, y_i)$  générées suivant  $\mathbb{P}$  :  $\mathcal{D} = (x_i, y_i)_{i=1}^m \in (\mathcal{X} \times \mathcal{Y})^m$ .
  - $\mathcal{D}$  est l'ensemble d'apprentissage
- **But** : Construire une fonction  $f : \mathcal{X} \rightarrow \mathcal{Y}$  qui prédit la sortie  $y$  d'une nouvelle observation  $x$  avec une probabilité d'erreur minimale.

Théorie de l'apprentissage [Vapnik88]

Borner la probabilité d'erreur  $R(f)$

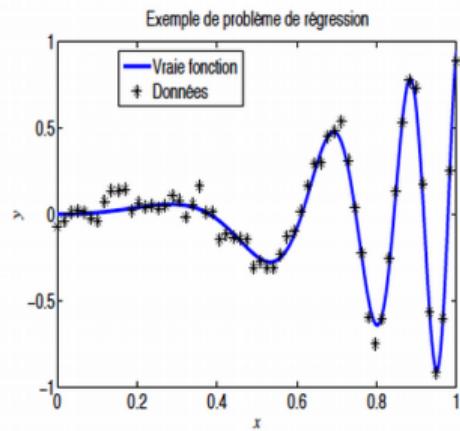
$$R(f) \leq \text{erreur empirique de } f + \text{Complexité de la classe de fonctions} + \text{terme résiduel}$$

# Remarques

- ➊ La seule observation dont on dispose est l'ensemble des couples  $(x, y)$  :  $\mathbb{P}$  est inconnue.
- ➋ Cet ensemble  $\mathcal{D} = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$  est l'ensemble d'apprentissage.
- ➌  $x \in \mathcal{X}$ . En général  $\mathcal{X} \subseteq \mathbb{R}^d$ .
  - $x$  peut être une image, du texte, une table de transactions, des séries temporelles...
  - Une étape importante est l'étape de description des données d'entrées pour les transformer en un vecteur de caractéristiques  $\Phi(x)$ .
  - Cette étape peut être apprise (apprentissage profond, apprentissage de représentations) ou construite à la main (feature-engineering).
- ➍  $y \in \mathcal{Y}$ , selon le type de  $\mathcal{Y}$ , différents problèmes d'apprentissage.

# Apprentissage supervisé : régression

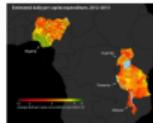
On parle de regression si  $y$  est à valeurs réelles, i.e.  $\mathcal{Y} = \mathbb{R}^m$ .



Source : G.Gasso

On ne dispose que des points  $(x_i, y_i)$  en noir et il faut estimer une fonction  $f(x)$  qui donne une bonne estimation de la vraie fonction en bleu.

# Apprentissage supervisé : régression



Poverty mapping: satellite image → asset wealth index



Housing: information about house → price

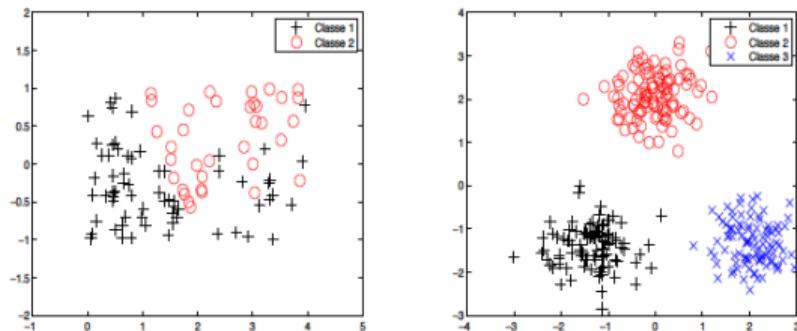


Arrival times: destination, weather, time → time of arrival

Figure: Source : CS221 Stanford

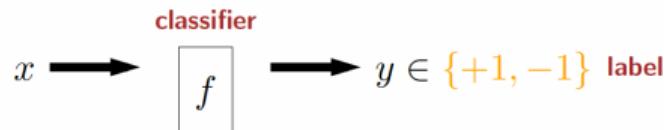
# Apprentissage supervisé : classification

On parle de classification si  $x_i$  est associé à une sortie catégorielle  $y_i \in \{1, \dots, K\}$  ou encore  $y_i \in \{\mathcal{C}_1, \dots, \mathcal{C}_K\}$



Source : G.Gasso

# Apprentissage supervisé : régression



Fraud detection: credit card transaction → fraud or no fraud



Toxic comments: online comment → toxic or not toxic



Higgs boson: measurements of event → decay event or background

Figure: Source : CS221 Stanford

# Objectifs de l'apprentissage (supervisé)

On cherche à construire une fonction  $f(x)$  à partir des  $n$  données d'apprentissage

$$\begin{array}{rcl} f : & \mathcal{X} \rightarrow & \mathcal{Y} \\ & x \mapsto & \hat{y} = f(x) \end{array}$$

Propriétés de  $f(x)$

- $\forall (x_i, y_i) \in \mathcal{D}$ , on veut que  $f(x_i)$  prédise la bonne valeur de  $y_i$  ( $\hat{y}_i = y_i$ ).
- $f$  doit pouvoir prédire les bonnes sorties pour des exemples futurs  $x_j$  (**généralisation**).
- $f$  appartient à un espace  $\mathcal{H}$  appelé **espace d'hypothèses**.

# Qualités attendues d'une technique d'apprentissage

- **Précision** : le taux d'erreur doit être le plus bas possible.
- **Robustesse** : le modèle doit dépendre aussi peu que possible de l'échantillon d'apprentissage et se généraliser à d'autres échantillons.
- **Concision, parcimonie** : les règles du modèles doivent être aussi simples et aussi peu nombreuses que possible.
- **Diversité des types de données utilisées** : données qualitatives, discrètes, continues et manquantes.
- **Rapidité de calcul du modèle** : apprentissage rapide pour affinement du modèle.
- **Paramétrage** : pouvoir pondérer les erreurs.

Comment mesurer la qualité d'une fonction apprise ?

# Mesure de la qualité d'une fonction apprise

## Fonction de coût (loss fonction)

Une fonction de coût permet d'évaluer la pertinence de la prédiction et de pénaliser les erreurs, i.e. lorsque la prédiction  $\hat{y} = f(x)$  est différente de  $y$ .

$$L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+ \text{ telle que } L(y, y') > 0 \text{ pour } y \neq y'$$

Le coût d'une bonne prédiction ( $y = f(x)$ ) est plus faible que le coût d'une erreur ( $y \neq f(x)$ ).

## Exemples

- Exemple de fonction coût pour la classification binaire :  
Coût 0-1  $L(y_i, f(x_i)) = 0$  si  $y_i = f(x_i)$ , 1 sinon.  
Mesure le nombre de points mal classés.
- Exemple de fonction coût pour la regression :  
Coût quadratique  $L(y_i, f(x_i)) = (y_i - f(x_i))^2$

# Mesure de la qualité d'une fonction apprise

Notion de **risque** : On s'intéresse au comportement moyen de la fonction de perte (ou de coût).

## Notion de risque

- ➊ Risque, risque fonctionnel, erreur de généralisation : espérance de  $L(y, f(x))$  par rapport à  $(x, y)$ .

$$R(f) = E_{P(x,y)} L(y, f(x))$$

**Représente en moyenne le coût pour toutes les données possibles** : **risque moyen** ou **erreur de généralisation**  
Il n'est pas possible de le calculer.

- ➋ Risque empirique:

$$R_{emp}(f, \mathcal{D}) = \frac{1}{n} \sum_{i=1}^n L(y_i, f(x_i))$$

**Représente en moyenne le coût pour toutes les données de l'ensemble d'apprentissage**  
Il est possible de le calculer.

# Comment apprendre la fonction $f$ ?

On cherche à construire une fonction  $f(x)$  à partir de  $n$  données.  $f(x)$  doit donner de bonnes approximations sur ces  $n$  données mais aussi sur les données futures.

- On aimerait minimiser le risque fonctionnel mais ce n'est pas possible.
- On va donc apprendre  $f$  en minimisant le risque empirique ([Empirical Risk Minimization](#))

Apprentissage : un problème d'optimisation

On cherche une fonction  $f \in \mathcal{H}$  qui minimise le risque empirique.

Machine Learning

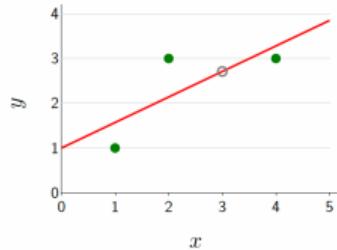
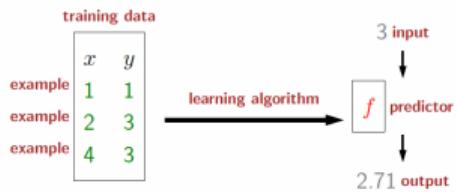
$$u^* = \operatorname{argmin}_{\mathbf{u}} \frac{1}{N} \sum_{i=1}^N L(\nu_i, y_i, \mathbf{u})$$

Data-driven ←

Optimization problem

# Petit exemple avec un problème de regression linéaire

[Source CS 221 Stanford]

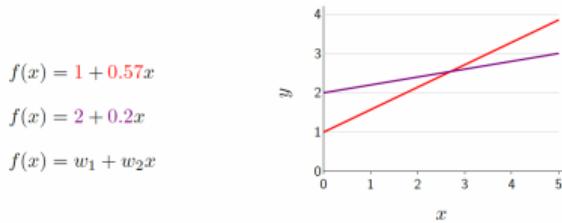


## Choix de conception

- Quels types de fonctions de prédiction? [Espace d'hypothèses](#)
- Comment mesurer la qualité de la fonction de prédiction ? [Fonction de perte ou de coût](#)
- Comment calculer la meilleure fonction de prédiction ? [Algorithme d'optimisation](#)

# Petit exemple avec un problème de regression linéaire

## Choix de l'espace d'hypothèses



On note le vector de poids  $\mathbf{w} = [w_1, w_2]$  et  $\Phi$  l'extracteur de caractéristiques avec donc  $\Phi(x) = [1, x]$  le vecteur caractéristique.

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \Phi(x)$$

e.g.  $f_{\mathbf{w}}(3) = [1, 0.57] \cdot [1, 3] = 2.71$

## Classe d'hypothèses

$$\mathcal{F} = \{f_{\mathbf{w}} : \mathbf{w} \in \mathbb{R}^2\}$$

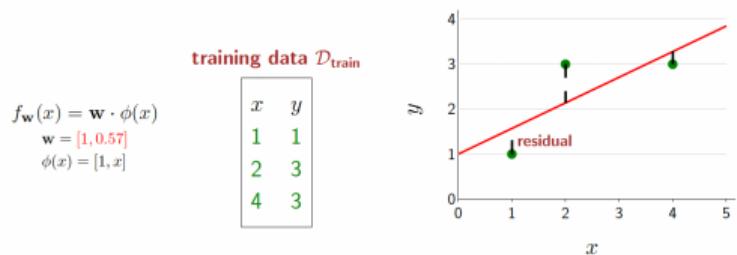
# Espace des hypothèses

- Pour poser un problème d'apprentissage supervisé, il faut décider du type de fonctions de modélisation que nous allons considérer (en fonction de nos convictions par rapport au problème).
- Etant donné un jeu de  $n$  observations étiquetées  $\mathcal{D} = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$  et un espace d'hypothèses  $\mathcal{F}$ , la tâche d'apprentissage supervisé consiste à trouver une hypothèse  $f \in \mathcal{F}$  qui approche au mieux la vraie fonction de mise en correspondance entre les données et leurs étiquettes.

## Besoin de deux outils

- Quantifier la qualité d'une hypothèse : notion de **fonction de coût**.
- Chercher une hypothèse optimale dans  $\mathcal{F}$  : apprentissage par optimisation (hypothèse maximale au sens de la fonction de coût)

# Petit exemple avec un problème de regression linéaire



Fonction de perte : Comment mesurer la qualité de la fonction de prédiction ?

Erreur au sens des moindres carrés.

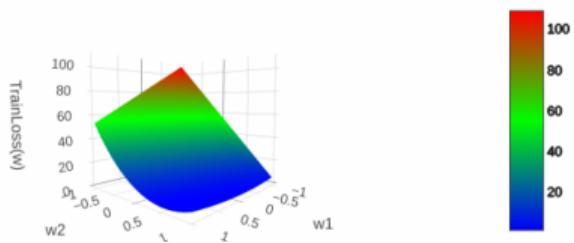
$$\text{Loss}(x, y, \mathbf{w}) = (f_{\mathbf{w}}(\mathbf{w}) - y)^2$$

Risque empirique

$$R_{\text{emp}}(\mathbf{w}) = \text{TrainLoss}(\mathbf{w}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{w})$$

# Petit exemple avec un problème de regression linéaire

L'apprentissage comme un problème d'optimisation



La meilleure fonction de prédiction est celle qui a le plus petit risque empirique. Il s'agit donc de résoudre un problème d'optimisation : la minimisation du risque empirique.

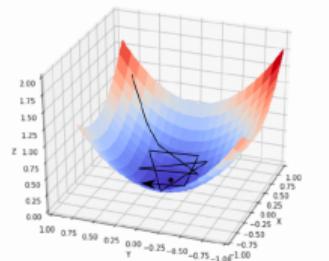
# Petit exemple avec un problème de regression linéaire

## Algorithme d'apprentissage, optimisation

But :  $\min_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$

## Gradient

Le gradient  $\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})$  est la direction dans laquelle le risque empirique croît le plus : on va donc utiliser un algorithme de descente de gradient.



### Algorithm: gradient descent

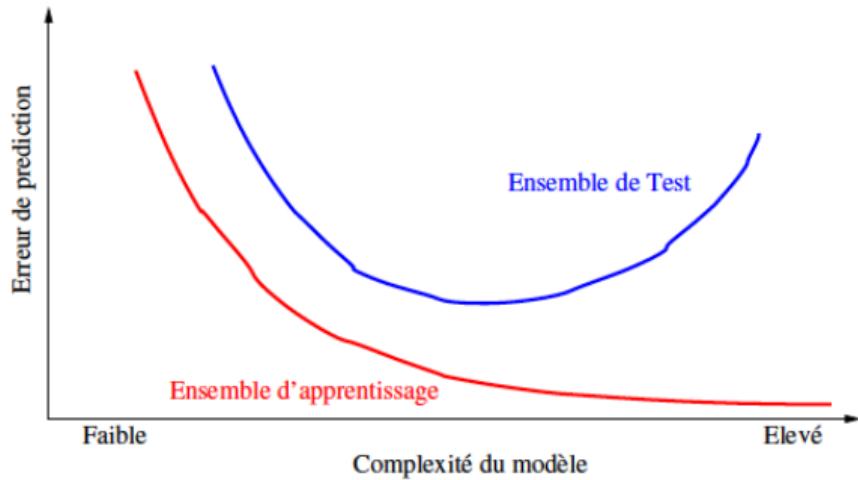
Initialize  $\mathbf{w} = [0, \dots, 0]$

For  $t = 1, \dots, T$ : epochs

$$\mathbf{w} \leftarrow \mathbf{w} - \underbrace{\eta}_{\text{step size}} \underbrace{\nabla_{\mathbf{w}} \text{TrainLoss}(\mathbf{w})}_{\text{gradient}}$$

# Mais...

Le risque empirique ne permet pas d'évaluer la pertinence du modèle car il est possible de choisir  $f$  telle que le risque empirique soit nul mais que l'erreur de généralisation soit élevée. On parle de **sur-apprentissage**.



# Sur-apprentissage : exemple



## Algorithm: rote learning

Training: just store  $\mathcal{D}_{\text{train}}$ .

Predictor  $f(x)$ :

If  $(x, y) \in \mathcal{D}_{\text{train}}$ : return  $y$ .

Else: **segfault**.

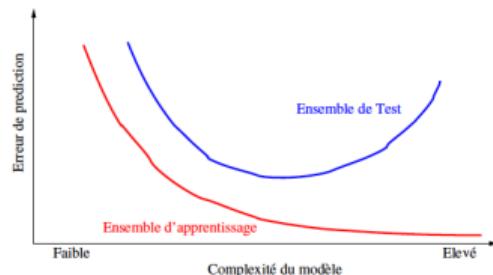
Le risque empirique sera nul mais le modèle de prédiction sera très mauvais sur de nouvelles données.

# Notion de généralisation

- Soit  $f$  une fonction de décision construite sur  $\mathcal{D}_n = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$
- $R_{emp}(\mathcal{D}_n, f)$  : performance du modèle évalué sur  $\mathcal{D}_n$ .
- $R_{moy}(\mathcal{D}_\infty, f)$  : performance théorique de  $f$  sur toutes les données futures possibles.

## Capacité de généralisation

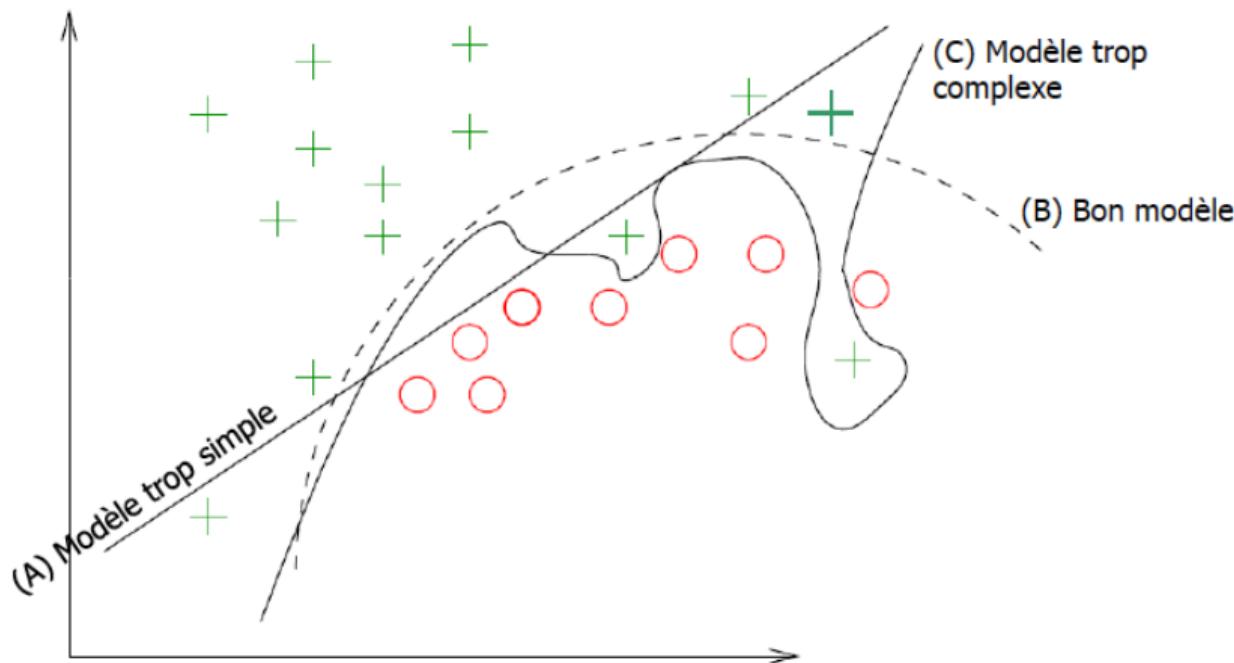
Capacité de  $f$  de donner de bonnes performances lorsqu'elle est testée sur des données autres que celles qui ont servi à l'apprentissage.



$R_{emp}(\mathcal{D}_n, f)$  n'est pas un bon estimateur de la capacité de généralisation.

# Sur-apprentissage

Taux d'erreur en fonction de la complexité du modèle.



# Juste un peu plus de théorie

- Pour obtenir une modèle décisionnel à partir des données de  $\mathcal{D}_n$ , il est nécessaire de choisir une famille paramétrique de modèles  $\mathcal{F}(w)$  (e.g. modèles linéaires) et d'appliquer un algorithme d'optimisation pour déterminer le vecteur de paramètre optimal  $w^*$  qui définit le modèle qui présente la meilleure généralisation.
- Idée : chercher le paramètre optimal  $w^*$  qui minimise le risque empirique ([minimisation du risque empirique \(MRE\)](#)).
- Cela permet-il d'obtenir un modèle qui a un risque théorique minimal ? On examine deux aspects :
  - La cohérence (consistency) : quand  $n \rightarrow \infty$  le risque empirique converge-t-il vers le risque espéré ? MRE est cohérente ssi la VC-dimension de la famille  $\mathcal{F}$  est finie.
  - Si le nombre de données  $n$  est fini, est-il possible de borner l'écart entre le risque espéré et le risque empirique pour un modèle  $f$ .  
De telles bornes existent : elles dépendent de  $\mathcal{F}$  et de  $n$ .

# Juste un peu plus de théorie

## Compromis biais-variance

Comparaison de  $R(f)$  avec l'erreur minimale  $R^*$  qui peut être atteinte par n'importe quelle fonction mesurable de  $\mathcal{X}$  dans  $\mathcal{Y}$

$$R(f) - R^* = (R(f) - \min_{h \in \mathcal{F}}) + (\min_{h \in \mathcal{F}} - R^*)$$

- Premier terme : distance entre le modèle  $f$  et le modèle optimal : [erreur d'estimation](#)
- Second terme : quantifie la qualité du modèle optimal sur  $\mathcal{F}$ , i.e. la qualité du choix de l'espace de fonctions mesurables : [erreur d'approximation](#)

Compromis biais-variance : un espace d'hypothèses plus large (modèles plus complexes) permet de réduire l'erreur d'approximation mais la solution optimale est plus difficile à trouver : l'erreur d'estimation augmente.

# Régularisation et pénalisation

En pratique,

- minimisation du risque empirique régularisé (MRER) qui cherche à réduire le risque empirique tout en pénalisant la complexité du modèle.

$$w^* = \operatorname{argmin}_w (R_{\text{emp}}(f(w), \mathcal{D}_n) + \text{Reg}(f(w)))$$

- ou minimisation du risque structurel (MRS) : pénalisation de la capacité de la famille (cardinal du plus grand ensemble de points que l'algorithme peut pulvériser) en considérant une séquence  $\mathcal{F}_d, d \in \mathbb{N}$  de modèles de capacité croissante.

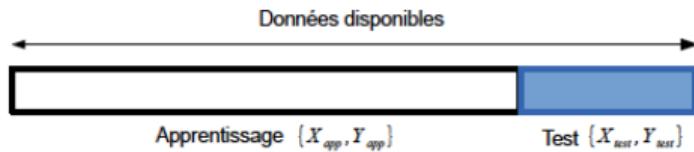
$$w^* = \operatorname{argmin}_w (R_{\text{emp}}(f(w), \mathcal{D}_n) + \text{Pen}(n, d))$$

Pour minimiser le risque espéré, il faut chercher un bon compromis entre la minimisation de la capacité de la famille de modèles et la minimisation de l'erreur d'apprentissage.

Apprentissage : fonction objectif

$$\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{w}) + \text{Reg}(\mathbf{w})$$

# Paradigme ensemble de test et ensemble d'apprentissage



Découper aléatoirement  $\mathcal{D}_n$  en deux ensembles disjoints  $\mathcal{D}_{app}$  et  $\mathcal{D}_{test}$

- $\mathcal{D}_{app} = \{(x_i, y_i)\}_{i=1..n_{app}}$  : données servant à l'apprentissage de  $f$ .
- $\mathcal{D}_{test} = \{(x_i, y_i)\}_{i=1..n_{test}}$  : données servant à évaluer la capacité de généralisation du modèle  $f$ .

# Apprentissage automatique : les principaux ingrédients

- Espace d'hypothèses:  $f_{\mathbf{w}}(x) = \mathbf{w} \cdot \Phi(x)$ 
  - Extracteur de caractéristiques  $\Phi(x)$
  - Architecture, nombre de couches.
- Fonction objectif pour l'apprentissage :  $\frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \mathbf{w}) + \text{Reg}(\mathbf{w})$ 
  - Fonction de perte
  - Régularisation
- Algorithme d'optimisation : descente de gradient (stochastique)
  - Nombre d'éPOCHS ( $T$ )
  - Pas d'apprentissage ( $\nu$ )
  - Initialisation

# Plan

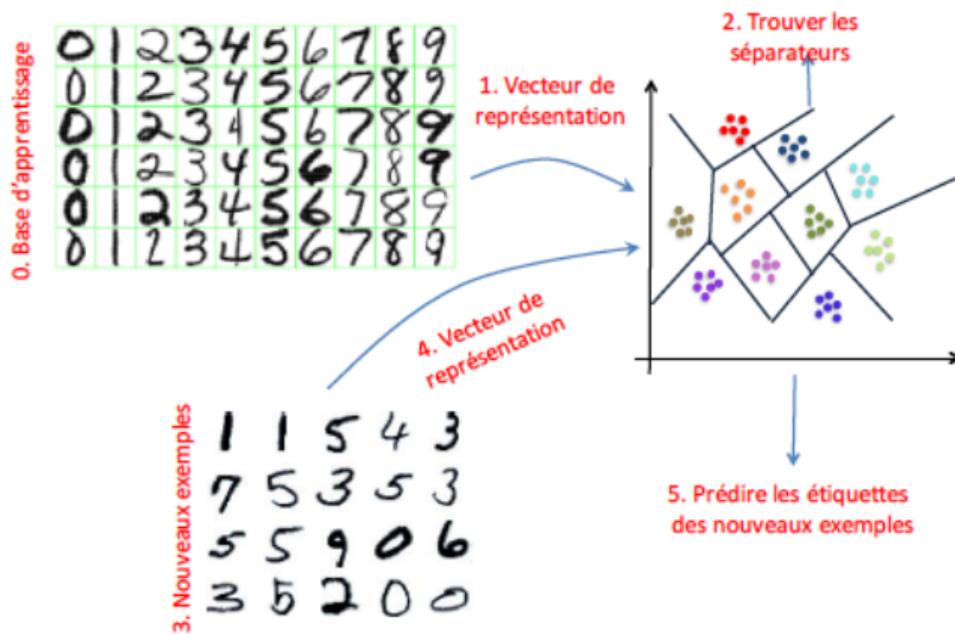
## 1 Agents reflexes - Apprentissage

- Apprentissage automatique

## 2 Evaluation de modèles

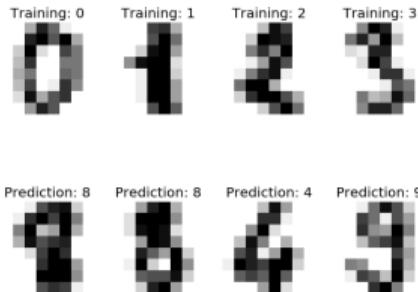
- Métriques de performance
- Ré-échantillonnage
- Choix des paramètres

# Exemple : reconnaissance de caractères



De nombreux travaux autour de l'apprentissage de représentation en utilisant les réseaux de neurones profonds.

# Exemple : reconnaissance de caractères



De nombreuses approches avec de nombreux hyper-paramètres à régler

- K-NN : choix de K
- SVM : choix de  $C$ , le paramètre de régularisation
- ...

Nécessité de comparer et de sélectionner le bon modèle.

# Evaluation de la qualité d'un classifieur

- Panoplie de méthodes de classification.
- Laquelle choisir ? Y-a-t-il une méthode supérieure aux autres quelque soit le problème ? Comment comparer les méthodes de classification entre elles ? (**selection de modèles**)
- Y-a-t-il un ensemble de caractéristiques meilleur qu'un autre ?
- Comment évaluer une méthode de classification ? Quelles métriques ? Quelles méthodes ? (**evaluation de modèles**)

# Métriques pour l'évaluation : classification

## Matrice de confusion

Pour la classification binaire

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	a	b
	Class>No	c	d

- a: TP (true positive)
- b: FN (false negative)
- c: FP (false positive)
- d: TN (true negative)

- $TP$  et  $TN$  : bonnes prédictions.
- $FP$  et  $FN$  : erreurs.

Plusieurs critères peuvent être construits à partir de cette matrice

# Métriques pour l'évaluation

## Accuracy

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

$$\text{Accuracy} = \frac{a + d}{a + b + c + d} = \frac{TP + TN}{TP + TN + FP + FN}$$

Plus ce critère est grand, meilleur est le modèle. Il représente le taux de bonne classification

# Métriques pour l'évaluation

Taux d'erreur

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

$$\text{Error\_rate} = \frac{FP + FN}{TP + TN + FP + FN}$$

Plus ce critère est petit, meilleur est le modèle.

# Métriques pour l'évaluation

Matrice de cout

		PREDICTED CLASS		
		C(i j)	<b>Class=Yes</b>	<b>Class&gt;No</b>
ACTUAL CLASS	<b>Class=Yes</b>	C(Yes Yes)	C(No Yes)	
	<b>Class&gt;No</b>	C(Yes No)	C(No No)	

$C(i|j)$  : cout de mal classifier une instance de la classe  $j$  en  $i$ .

# Métriques pour l'évaluation

## Autres métriques

		PREDICTED CLASS	
		Class=Yes	Class>No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

$$\text{Precision} = \frac{TP}{TP + FP}$$

Proportion de positifs de la population parmi tous les positifs prédits (un modèle de précision 1 ne fait aucune erreur)

$$\text{Rappel} = \frac{TP}{TP + FN}$$

Proportions de positifs prédits parmi tous les positifs de la population (un modèle parfait de rappel 1 prédit la totalité des positifs)

$$F_{\text{mesure}} = 2 \frac{\text{Precision} \times \text{Rappel}}{\text{Precision} + \text{Rappel}}$$

# Métriques pour l'évaluation

## Autres métriques

		PREDICTED CLASS	
ACTUAL CLASS		Class=Yes	Class>No
	Class=Yes	a (TP)	b (FN)
	Class>No	c (FP)	d (TN)

$$\text{Sensibility} = \frac{TP}{TP + FN}$$

$$1 - \text{Specificity} = \frac{FP}{TN + FP} = 1 - \frac{TN}{TN + FP}$$

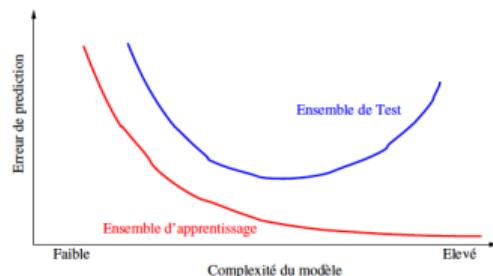
La spécificité mesure la proportion de négatifs prédis parmi tous les négatifs de la population.  
Permet de construire la courbe ROC

# Notion de généralisation

- Soit  $f$  une fonction de décision construite sur  $\mathcal{D}_n = ((x_1, y_1), (x_2, y_2), \dots, (x_n, y_n))$
- $R_{emp}(\mathcal{D}_n, f)$  : performance du modèle évalué sur  $\mathcal{D}_n$ .
- $R_{moy}(\mathcal{D}_\infty, f)$  : performance théorique de  $f$  sur toutes les données futures possibles.

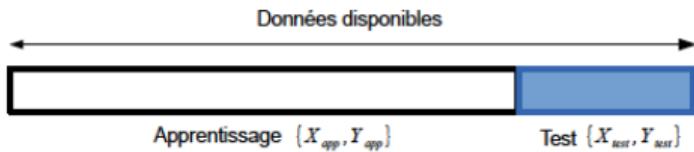
## Capacité de généralisation

Capacité de  $f$  de donner de bonnes performances lorsqu'elle est testée sur des données autres que celles qui ont servi à l'apprentissage.



$R_{emp}(\mathcal{D}_n, f)$  n'est pas un bon estimateur de la capacité de généralisation.

# Paradigme ensemble de test et ensemble d'apprentissage

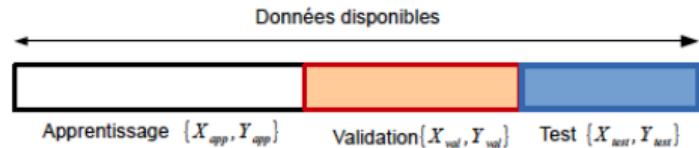


Découper aléatoirement  $\mathcal{D}_n$  en deux ensembles disjoints  $\mathcal{D}_{app}$  et  $\mathcal{D}_{test}$

- $\mathcal{D}_{app} = \{(x_i, y_i)\}_{i=1..n_{app}}$  : données servant à l'apprentissage de  $f$ .
- $\mathcal{D}_{test} = \{(x_i, y_i)\}_{i=1..n_{test}}$  : données servant à évaluer la capacité de généralisation du modèle  $f$ .

# Ensemble de validation - Sélection de modèles

Comment sélectionner le bon modèle sans toucher à  $\mathcal{D}_{test}$ ? Cas idéal =  $n$  est grand !



Découper aléatoirement  $\mathcal{D}_n$  en trois ensembles disjoints  $\mathcal{D}_{app}$ ,  $\mathcal{D}_{test}$  et  $\mathcal{D}_{val}$

## Procédure de sélection de modèle

- ① Apprendre chaque modèle possible sur  $\mathcal{D}_{app}$
- ② Evaluer sa performance en généralisation sur  $\mathcal{D}_{val}$
- ③ Sélectionner le modèle qui donne la meilleure performance sur  $\mathcal{D}_{val}$
- ④ Tester le modèle retenu sur  $\mathcal{D}_{test}$  : on teste la capacité de généralisation du modèle retenu.

# Evaluation et comparaison des modèles

- Qu'arrive-t-il si on dispose de très peu d'échantillons ?
- Comment savoir si le taux d'erreur est précis ou si on est pas tombé par hasard sur une situation particulière en coupant l'ensemble  $\mathcal{D}$  ?
- Si pour un ensemble de données  $\mathcal{D}$ , 2 modèles  $C_1$  et  $C_2$  ont 80% et 85% de précision, est-ce que  $C_2 > C_1$ ?
- Solution : le ré-échantillonnage.

# Plan

## 1 Agents reflexes - Apprentissage

- Apprentissage automatique

## 2 Evaluation de modèles

- Métriques de performance
- Ré-échantillonnage**
- Choix des paramètres

# Techniques de ré-échantillonnage

Le ré-échantillonnage génère différents sous-ensembles de données à partir de l'ensemble initial  $\mathcal{D}$ .

- Pour comparer des classifieurs :
  - Validation croisée.
  - Bootstrap
- Pour améliorer un classifieur :
  - Bagging
  - Boosting

# La validation croisée

Approche plus exhaustive pour tirer parti de l'ensemble des données.

## Principe

Plusieurs partitionnements de  $\mathcal{D}_n$  sont effectués :  $\mathcal{D}_n = \mathcal{D}_{app}^i \cup \mathcal{D}_{val}^i, i \in 1,..k$  avec  $\mathcal{D}_{app}^i \cap \mathcal{D}_{val}^i = \emptyset$

- On apprend à chaque fois un modèle  $f$  sur  $\mathcal{D}_{app}^i$ .
- On calcule son erreur empirique  $R_{emp}$  sur  $\mathcal{D}_{val}^i$
- Le risque espéré est estimé par la moyenne des risques  $\frac{1}{k} \sum_{i=1}^k R_{emp}(f, \mathcal{D}_{val}^i)$ .

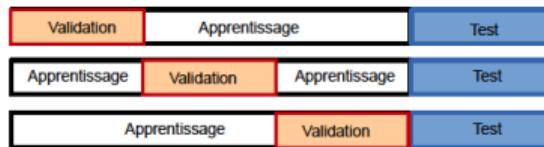
# La validation croisée

Plusieurs stratégies selon le partitionnement

- **Exhaustives** : tous les partitionnements possibles respectant certains effectifs sont utilisés
  - Leave p out.
  - Leave one out.
- **Non-Exhaustives**
  - k-fold.
  - Echantillonnage répété.

# Evaluation et comparaison des modèles

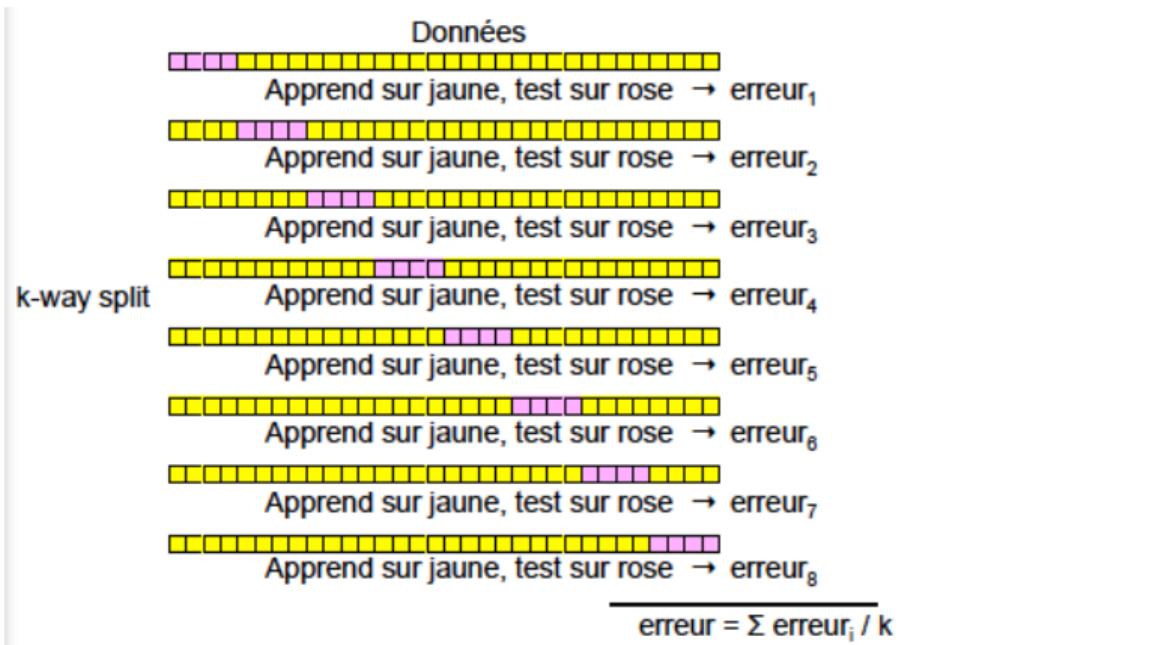
## Validation croisée $K$ -blocs



- ➊ Découper aléatoirement  $\mathcal{D}_n = \mathcal{D}_{app} \cup \mathcal{D}_{test}$
- ➋ On prend  $K$  ensemble disjoints de  $\frac{|\mathcal{D}_{app}|}{K}$  échantillons dans  $\mathcal{D}_{app}$  ( $\mathcal{D}_{app} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \dots \cup \mathcal{D}_K$ ).
- ➌ Pour  $k = 1..K$ 
  - ➊ Mettre de côté  $\mathcal{D}_k$
  - ➋ Apprendre le modèle  $f$  sur les  $K - 1$  ensembles restants.
  - ➌ Estimer sa performance  $R_k$  en généralisation sur  $\mathcal{D}_k$
- ➍ Moyenner les  $K$  mesures de performance  $R_k$ .

# Evaluation et comparaison des modèles

## Validation croisée K-blocs



# Evaluation et comparaison des modèles

## Validation croisée $n$ -blocs : leave one out

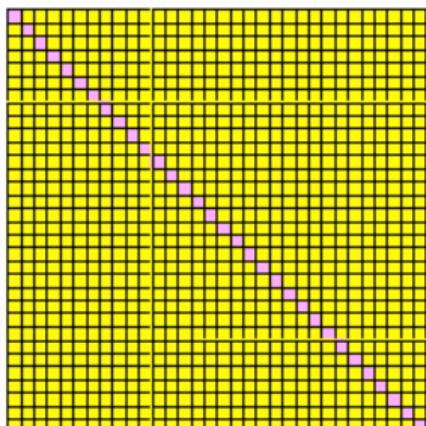
- On prend un seul échantillon pour tester
- On teste avec l'un d'entre eux.
- Taux d'erreur = moyenne des  $n$  expériences.
- Avantage : utile quand  $\mathcal{D}$  petit

Leave  $p$  out :  $N - p$  données sont utilisées pour l'apprentissage et  $p$  pour la validation.

# Evaluation et comparaison des modèles

Validation croisée  $n$ -blocs : leave one out

Données



- Faible biais
- Haute variance
- Tend à sous-estimer l'erreur si les données ne sont pas vraiment i.i.d.

[Guyon & Elisseeff, jMLR, 03]

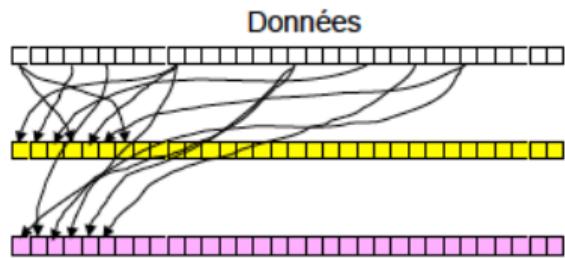
# Evaluation et comparaison des modèles

## Validation croisée aléatoire

- On prend aléatoirement  $k$  échantillons dans l'ensemble  $\mathcal{D}$  (sans remise) pour chaque expérience.
- Le taux d'erreur est la moyenne des taux de chacune des expériences.

# Evaluation et comparaison des modèles

Validation croisée aléatoire : bootstrap



On apprend sur jaune, on teste sur rose  
Répéter et faire la moyenne.

# Plan

## 1 Agents reflexes - Apprentissage

- Apprentissage automatique

## 2 Evaluation de modèles

- Métriques de performance
- Ré-échantillonnage
- Choix des paramètres

# Choix des paramètres

## Principe

- Définition d'intervalles de variation pour les paramètres et d'une procédure d'exploration de cet espace.
- Exploration de l'espace suivant la procédure et évaluation des modèles résultant par application de la validation croisée.
- Comparaison des résultats de validation croisée, sélection des valeurs des paramètres qui mènent aux meilleures performances.

# Choix des paramètres : procédure Grid Search

## Principe

- Exploration = recherche dans une grille
- Si on dispose de  $m$  paramètres, on définit pour chaque paramètre un intervalle et un pas de variation : grille de dimension  $m$  dont les cellules sont les valeurs à tester.
- Pour chaque cellule, on applique la validation croisée pour le modèle correspondant.
- Comparaison des résultats de validation croisée, sélection des valeurs des paramètres qui mènent aux meilleures performances.

# Conclusion

## Schéma typique d'une démarche d'apprentissage supervisé

Modélisation décisionnelle à partir de données.

- Importation, récupération et préparation des données.
- Choix d'une fonction de perte permettant de qualifier les réponses d'un modèle.
- Choix des familles paramétriques dans lesquelles les modèles seront recherchés.
- Subdivision en échantillons d'apprentissage et de test. (validation croisée).
- Construction du modèle sur l'échantillon d'apprentissage (avec la recherche des paramètres optimaux des algorithmes : estimation du meilleur modèle intra-famille et choix du meilleur modèle entre familles).
- Prédiction et évaluation sur l'échantillon test : performances de généralisation du modèle retenu.

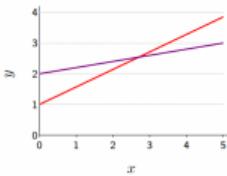
De l'apprentissage automatique à l'apprentissage profond

# Les réseaux de neurones

Pouvoir construire des fonctions de prédictions non-linéaires.

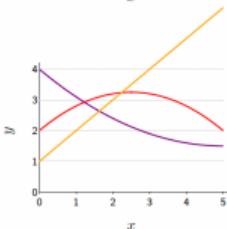
Linear predictors:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x), \phi(x) = [1, x]$$



Non-linear (quadratic) predictors:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x), \phi(x) = [1, x, x^2]$$



Non-linear neural networks:

$$f_{\mathbf{w}}(x) = \mathbf{w} \cdot \sigma(\mathbf{V}\phi(x)), \phi(x) = [1, x]$$

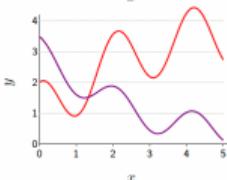


Figure: Source : CS221 Stanford

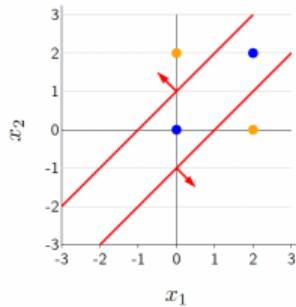
# Intuition avec un exemple simple

## Prédire la collision de deux voitures

- Entrées : positions des deux voitures (en terme de distance d'un côté de la route) :  $x_1$  et  $x_2$ .
- Sorties : si les voitures sont en sécurité ( $y = 1$ ) ou si elles sont en collision ( $y = -1$ ). On considère que les voitures sont en sécurité si elles sont séparées par une distance d'au moins 1.

$$y = \text{sign}(|x_1 - x_2| - 1)$$

$x_1$	$x_2$	$y$
0	2	1
2	0	1
0	0	-1
2	2	-1



Frontière de décision: tous les points situés à l'intérieur de la région entre les deux lignes rouges sont négatifs et tous les points situés à l'extérieur (de part et d'autre) sont positifs. l'extérieur (de chaque côté) comme positif.

# Intuition avec un exemple simple

## Décomposition du problème

Test if car 1 is far right of car 2:

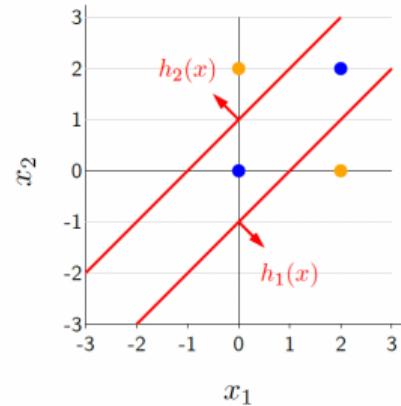
$$h_1(x) = \mathbf{1}[x_1 - x_2 \geq 1]$$

Test if car 2 is far right of car 1:

$$h_2(x) = \mathbf{1}[x_2 - x_1 \geq 1]$$

Safe if at least one is true:

$$f(x) = \text{sign}(h_1(x) + h_2(x))$$



$x$	$h_1(x)$	$h_2(x)$	$f(x)$
$[0, 2]$	0	1	+1
$[2, 0]$	1	0	+1
$[0, 0]$	0	0	-1
$[2, 2]$	0	0	-1

# Intuition avec un exemple simple

Réécriture avec une notation vectorielle

Intermediate subproblems:

$$h_1(x) = \mathbf{1}[x_1 - x_2 \geq 1] = \mathbf{1}[[\textcolor{red}{-1, +1, -1}] \cdot [1, x_1, x_2] \geq 0]$$

$$h_2(x) = \mathbf{1}[x_2 - x_1 \geq 1] = \mathbf{1}[[\textcolor{red}{-1, -1, +1}] \cdot [1, x_1, x_2] \geq 0]$$

$$\mathbf{h}(x) = \mathbf{1} \left[ \begin{bmatrix} \textcolor{red}{-1} & \textcolor{red}{+1} & \textcolor{red}{-1} \\ \textcolor{red}{-1} & \textcolor{red}{-1} & \textcolor{red}{+1} \end{bmatrix} \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \geq 0 \right]$$

Predictor:

$$f(x) = \text{sign}(h_1(x) + h_2(x)) = \text{sign}([\textcolor{red}{1, 1}] \cdot \mathbf{h}(x))$$

L'objectif est d'apprendre les poids en rouge.

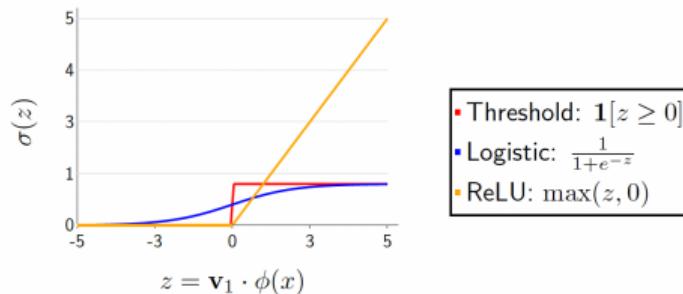
# Intuition avec un exemple simple

## Eviter les gradients nuls

- **Problème:** Le gradient de  $h_1(x)$  par rapport à  $\mathbf{v}_1$  est 0.

$$h_1(x) = 1[\mathbf{v}_1 \cdot \phi(x) \geq 0]$$

- **Solution :** remplacer la fonction indicatrice par une **fonction d'activation**  $\sigma$  avec des gradients non nuls.



$$h_1(x) = \sigma(\mathbf{v}_1 \cdot \phi(x))$$

# Intuition avec un exemple simple

Réseaux de neurones à 2 couches.

Intermediate subproblems:

$$\mathbf{h}(x) = \sigma(\mathbf{V} \phi(x))$$

Predictor (classification):

$$f_{\mathbf{V}, \mathbf{w}}(x) = \text{sign}(\mathbf{w}^\top \mathbf{h}(x))$$

Interpret  $\mathbf{h}(x)$  as a learned feature representation!

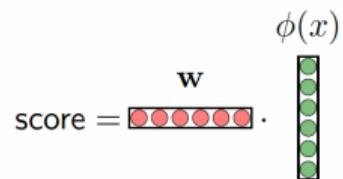
Hypothesis class:

$$\mathcal{F} = \{f_{\mathbf{V}, \mathbf{w}} : \mathbf{V} \in \mathbb{R}^{k \times d}, \mathbf{w} \in \mathbb{R}^k\}$$

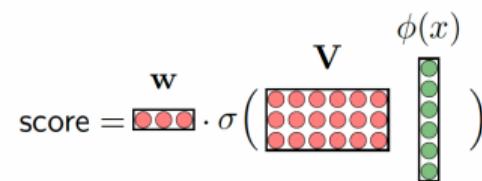
matrice de poids  $\mathbf{V}$  : vecteurs de poids des  $k$  sous-problèmes.

# Réseaux de neurones profonds

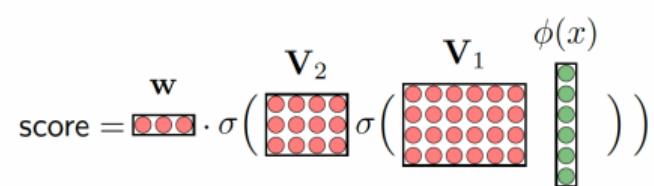
1-layer neural network:



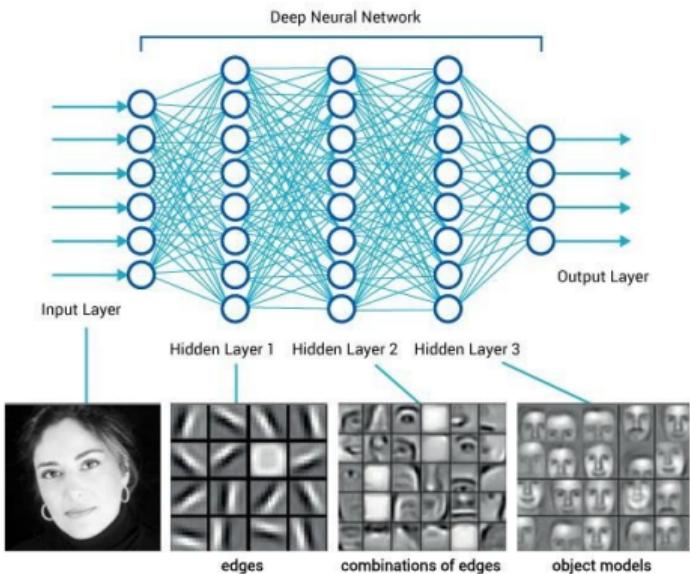
2-layer neural network:



3-layer neural network:

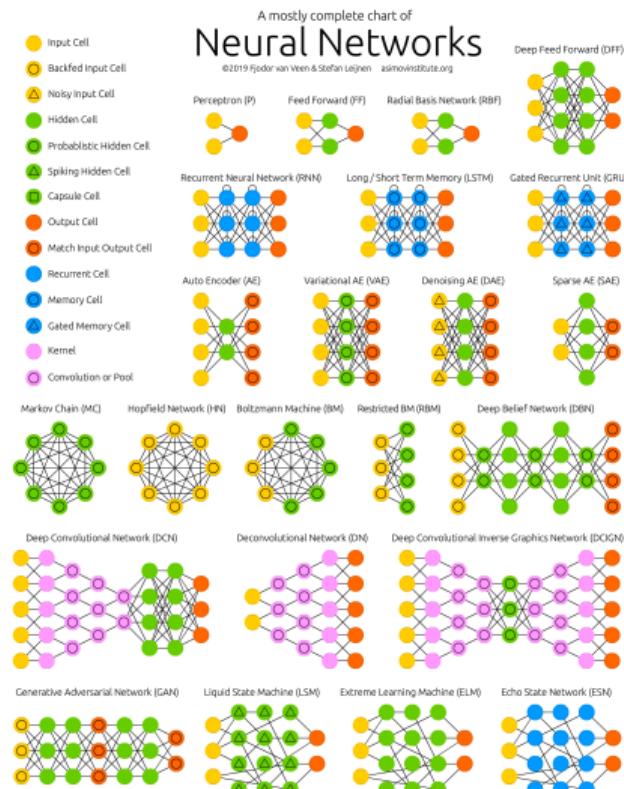


# Réseaux de neurones profonds



# Réseaux de neurones profonds

Un zoo de modèles (<https://www.asimovinstitute.org/neural-network-zoo/>)



## Apprentissage automatique et apprentissage profond : sur quelles données?

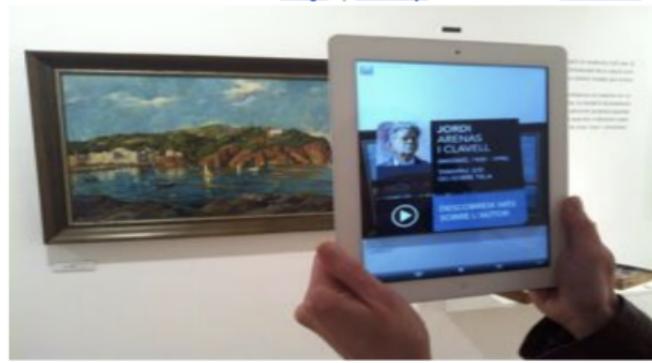
# Données visuelles - Capteurs visuels



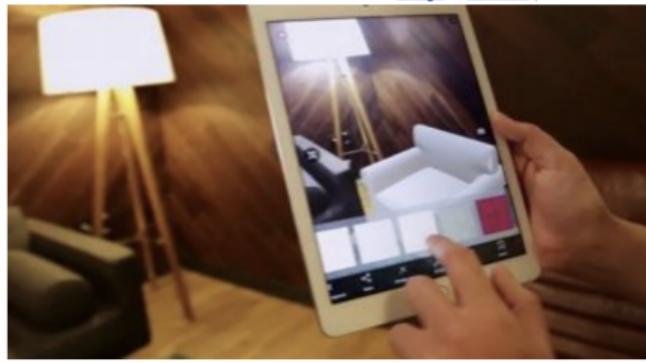
[Image](#) by [US Army](#) is licensed under [CC BY 2.0](#)



[Image](#) is [CC0 1.0](#) public domain



[Image](#) by [Kippelboy](#) is licensed under [CC BY-SA 3.0](#)



[Image](#) by Christina C. is licensed under [CC BY-SA 4.0](#)

# Données visuelles - Capteurs visuels

Object detection  
car



Action recognition  
bicycling



Scene graph prediction  
<person - holding - hammer>

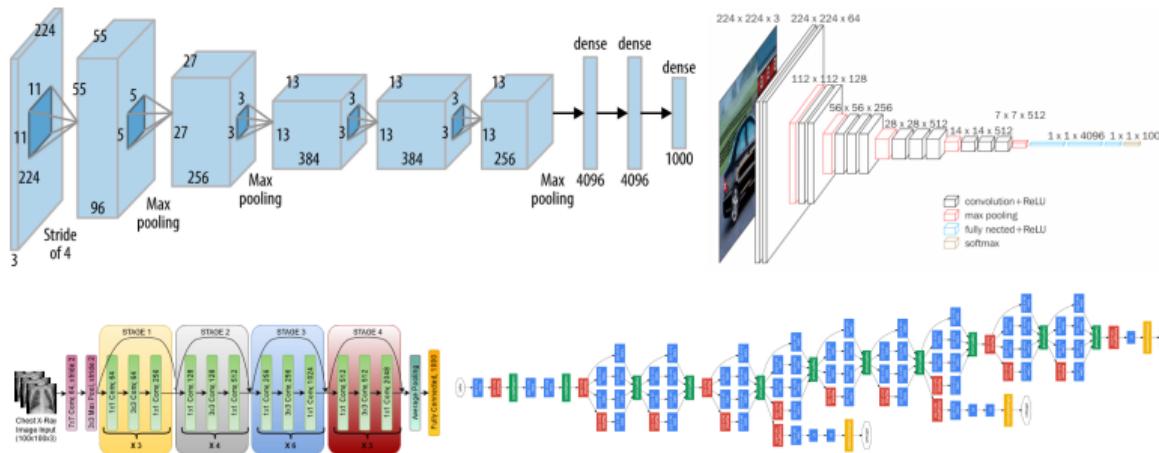
Captioning:  
*a person holding a hammer*



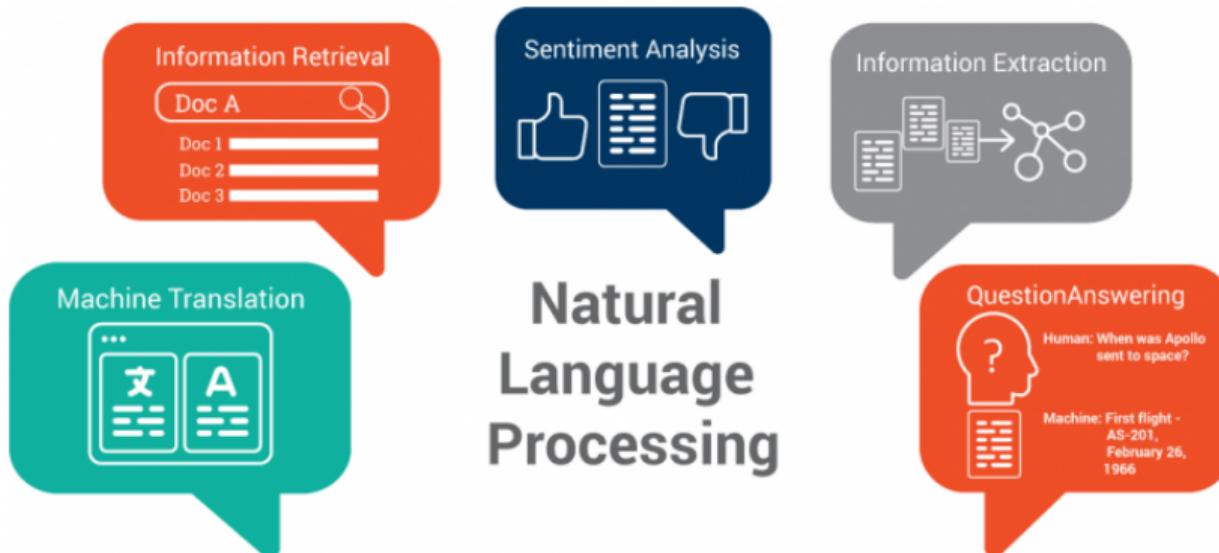
# Données visuelles - Capteurs visuels

## Les modèles stars

AlexNet, VGGNet, ResNet, Inception...

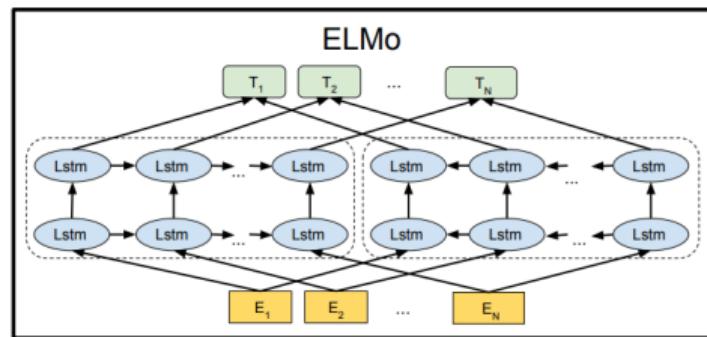
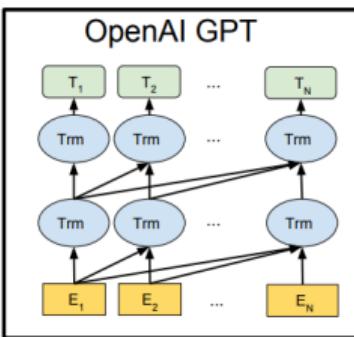
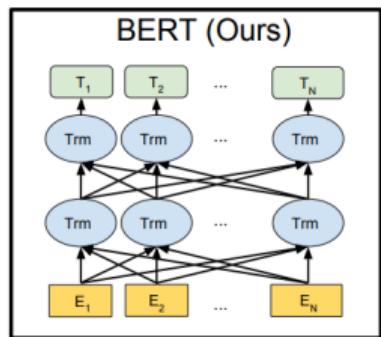


# Données textuelles

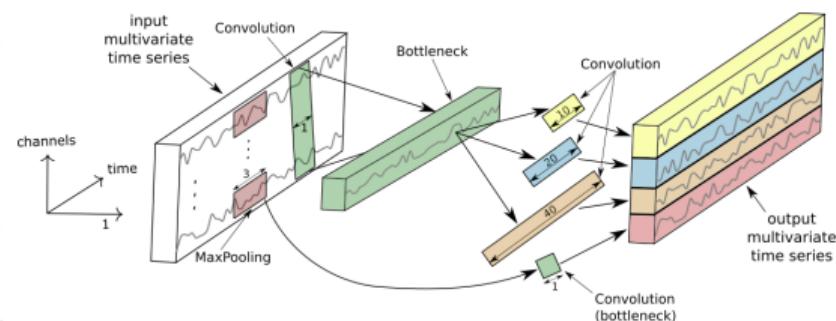
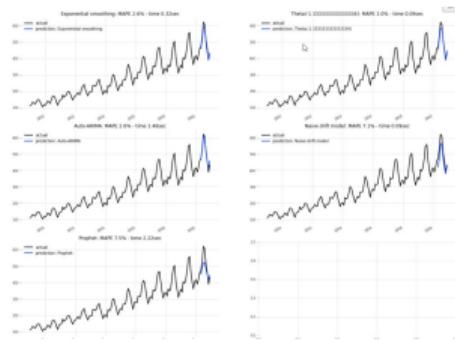


# Données textuelles

## Les modèles stars

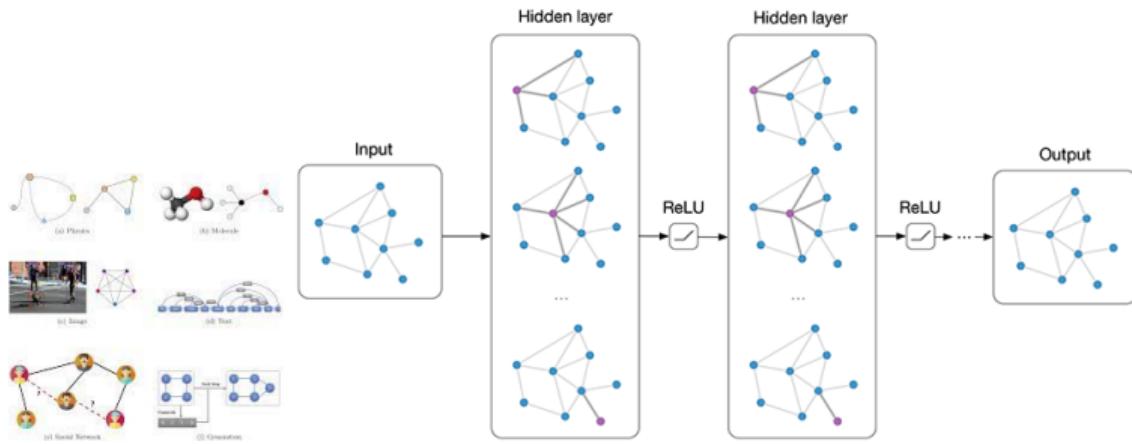


# Séries temporelles



Les stars : N-Beats, DeepAR, Spacetimeformer, TFT....

# Sur des données de type graphes



Graph neural networks, graph convolutional network, geometric deep learning...

Questions ?