Chapitre 3 : Apprentissage Automatique (Machine Learning)

Ce chapitre est le cœur de votre formation. Il fait le pont entre les mathématiques du chapitre précédent et les modèles complexes (ANN, LSTM, SVR) que vous utilisez dans vos recherches.

1. Concepts clés

Pour comprendre le Machine Learning, il faut maîtriser son vocabulaire spécifique :

1.1 Données (Data)

Les **données** sont la matière première du Machine Learning. Sans données de qualité, le modèle ne peut rien apprendre (*Garbage in, Garbage out*). Elles peuvent être :

- Numériques (température, concentration, âge…)
- Catégorielles (couleur, type de matériau…)
- Images, Signaux, Texte

On représente souvent les données sous forme d'une **matrice** :

$$X = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}$$

n : nombre d'échantillons, **p** : nombre de variables (features)

1.2 Features (Caractéristiques)

Les **features** (variables explicatives) sont les informations utilisées pour prédire la sortie. Ce sont vos variables d'entrée (notées **X**). Par exemple : température, pression, temps. La qualité des features influence directement la performance du modèle.

1.3 Étiquettes (Labels)

Les **étiquettes** sont les valeurs que l'on cherche à prédire (noté **y**). C'est la **« vérité terrain »**. On note généralement : $y = f(X$

- En régression : valeur continue (ex : capacité d'adsorption)
- En classification : classe (ex : bon/mauvais)

1.4 Modèle

C'est l'algorithme « entraîné ». On peut le voir comme une boîte mathématique qui transforme une entrée en une prédiction. Un **modèle** est une fonction mathématique qui relie les features aux étiquettes :

$$\hat{y} = f_\theta(X)$$

où :

- θ = paramètres du modèle
- $\hat{y}$= prédiction

Exemples de modèles :Linear Regression, Support Vector Machine (SVM), Artificial Neural Network (ANN), Long Short-Term Memory (LSTM), …, etc.

1.5 Généralisation

La **généralisation** est la capacité du modèle à bien prédire des données de test (**jamais vues)**. C'est l'objectif ultime. Un bon modèle doit être capable de donner une réponse correcte sur des données qu'il n'a **jamais vues** pendant l'entraînement. Deux problèmes possibles :

Underfitting : modèle trop simple, **Overfitting** : modèle trop complexe (apprend le bruit). On mesure l'erreur avec des métriques comme :

$$MSE = \frac{1}{n}(y_i - \hat{y}_i)^2 \quad , \quad \text{RMSE} = \sqrt{MSE}$$

MSE(Mean Square Error)= Erreur Quadratique Moyenne, RMSE(Root Mean Square Error)

2. Phases d'un pipeline d'apprentissage

Un **pipeline ML** est une suite d'étapes structurées. On ne donne jamais toutes les données au modèle en une seule fois. On divise le jeu de données en trois parties distinctes :

2.1.Entraînement (Training) : Le modèle regarde les données et ajuste ses paramètres internes (poids) pour minimiser l'erreur (Le modèle apprend les paramètres, Minimisation d'une fonction de coût (ex : MSE), Ajustement des poids (ex : descente de gradient)).

2.2.Validation : Utilisée pour régler les **« hyperparamètres »** (ex: choisir le nombre de neurones, le taux d'apprentissage, Paramètre C pour SVM) et détecter le sur-apprentissage.

2.3 Test : C'est l'examen final. On évalue le modèle sur des données totalement inconnues pour mesurer sa performance réelle (RMSE, R^2).(Données totalement nouvelles, Évalue la performance réelle, Ne doit jamais influencer l'entraînement.)

But :

- Train → ajustement des paramètres
- Validation → choix des hyperparamètres
- Test → estimation finale

Division typique :70% entraînement, 15% validation, 15% test

3. Types d'apprentissage

3.1 Apprentissage supervisé

- Le modèle apprend à partir d'exemples annotés. On dispose de couples **(X, y)**. L'apprentissage supervisé (*supervised learning*) s'intéresse aux données étiquetées. L'objectif est d'apprendre la relation X → y et de prédire l'étiquette (inconnue) *y* associée à une nouvelle observation $\boldsymbol{x}$, à partir de la connaissance fournie par les *N* observations étiquetées du jeu de données $(\boldsymbol{x_n}, \boldsymbol{y_n})_{\boldsymbol{1 \le n \le N}}$.
- Pour ce faire, la quasi-totalité des méthodes vues dans ce cours considèrent une fonction de prédiction $\boldsymbol{f_w}$ appartenant à une famille paramétrée par $\boldsymbol{w}$. Le vecteur $\boldsymbol{w}$ représente un ensemble de paramètres. Le nombre de paramètres peut être petit (cas de la régression linéaire) ou très grand (cas de l'apprentissage profond).
- La prédiction de l'étiquette d'une nouvelle observation $\boldsymbol{x}$ ne faisant pas partie du jeu de données sera alors $\boldsymbol{f_w(x)}$.

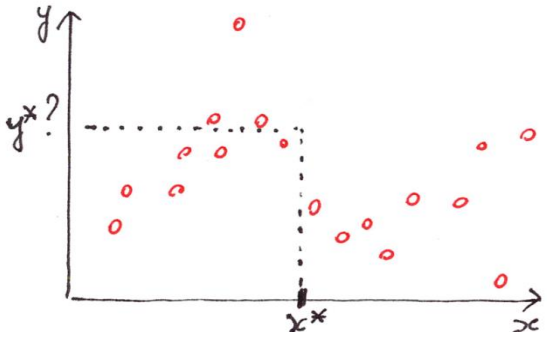
- Dans une phase d'apprentissage (*learning*), aussi appelée entraînement (*training*), les paramètres $\boldsymbol{w}$ sont adaptés de manière à optimiser les performances de prédiction sur le jeu de données $(\boldsymbol{x_n}, \boldsymbol{y_n})_{1 \leq n \leq N}$, appelé ici base d'apprentissage ou d'entraînement.
- L'apprentissage est basé sur la mesure de l'écart entre les « vraies » étiquettes $\boldsymbol{y_n}$ et les étiquettes prédites $\boldsymbol{f_w(x_n)}$. Comme on le verra, il ne suffit pas de choisir $\boldsymbol{w}$ minimisant cet écart pour obtenir des performances de prédiction optimales.
- Dans un second temps (phase de test ou d'inférence), on prédit l'étiquette d'une nouvelle observation *x* comme la valeur de $\boldsymbol{f_{\tilde{w}}(x)}$, à l'aide de l'ensemble de paramètres $\tilde{\boldsymbol{w}}$ obtenu par la phase d'apprentissage.
- Les problèmes à résoudre sont : le choix de la famille à laquelle appartient la fonction de prédiction $\boldsymbol{f_w}$, la manière de mesurer l'écart entre vraie étiquette et étiquette prédite, ou le choix du critère d'optimisation permettant de fixer $\boldsymbol{w}$.

  On distingue deux grandes familles de problèmes de l'apprentissage supervisé : **classification supervisée** (*y* désigne une classe) et **régression** (*y* est une valeur scalaire ou vectorielle). La distinction est justifiée par les méthodes de résolution sensiblement différentes.

3.1.1. Régression

Prédire une valeur numérique continue (ex: puissance d'un panneau solaire).

Figure 1 : *Régression*. *Les observations sont des scalaires x ; elles sont étiquetées par des scalaires y, les couples* (*x, y*) *étant représentés dans le plan. L'objectif est de prédire la valeur de l'étiquette* $\boldsymbol{y*}$ *associée à un scalaire* $\boldsymbol{x*}$*, à partir de la base d'apprentissage* $(\boldsymbol{x_n}, \boldsymbol{y_n})$ *représentée par les ronds. La difficulté est que certaines observations semblent entachées de perturbations aléatoires, certaines observations semblent même aberrantes.*



Lorsque les étiquettes $\boldsymbol{y_n}$ prennent des valeurs scalaires ou vectorielles, on parle de problème de régression. La figure 1 illustre un problème de régression d'une variable scalaire. Dans ce cadre, les étiquettes $\boldsymbol{y}$ sont scalaires, et les fonctions $\boldsymbol{f_w}$ sont affines sur l'espace $\mathbf{R}^{\boldsymbol{d}}$ des observations. Ces fonctions s'écrivent sous la forme :

$$\boldsymbol{f_w(x) = w_0 + w_1 \cdot x}$$

où $\boldsymbol{w_0} \in R$, $\boldsymbol{w_1} \in R^d$, et $\boldsymbol{w_1 \cdot x}$ désigne le produit scalaire entre $\boldsymbol{w_1}$ et $\boldsymbol{x}$.

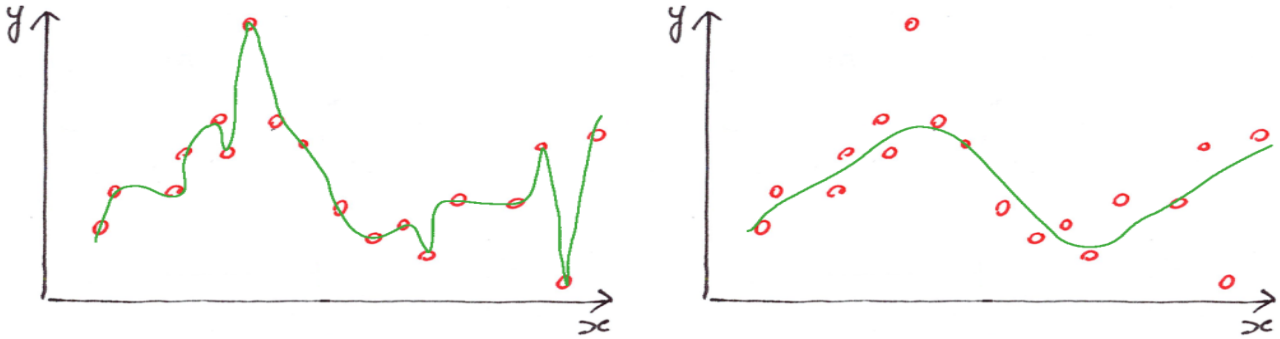


FIGURE 2 : *Deux exemples de courbes de régression sur la même base d'apprentissage. Faut-il mieux une courbe passant très près des observations d'apprentissage (à gauche), ou une courbe plus régulière évitant les observations aberrantes et robuste à des perturbations aléatoires des observations, voire robuste à des observations aberrantes (à droite) ?*

Régression linéaire : Les paramètres $\boldsymbol{w_0}$ et $\boldsymbol{w_1}$ sont obtenus par la méthode des moindres carrés, c'est-à-dire en minimisant la somme des carrés des résidus du modèle sur la base d'apprentissage $\left(\boldsymbol{x_n}, \boldsymbol{y_n}\right)_{\boldsymbol{1 \le n \le N}}$:

$$\sum_{n=1}^{N} |y_n - w_0 - \boldsymbol{w}_1 \cdot \boldsymbol{x}|^2$$

De manière générale, on appelle « résidu » l'écart entre valeur prédite $\boldsymbol{f_w}(\boldsymbol{x})$ et « vraie » valeur y. Si x est un scalaire et $\boldsymbol{x} = (x, x^2, \ldots, x^p)$ est constitué de puissances de x (jusqu'au degré p > 0), on voit que la régression polynomiale apparaît comme un cas particulier de la régression linéaire multivariée. En effet, dans ce cas, $\boldsymbol{f_w}(\boldsymbol{x}) = \boldsymbol{w_0} + \sum_{i=1}^{p} \boldsymbol{w_{1,i}} \cdot \boldsymbol{x^i}$, où $\boldsymbol{w_1} = (\boldsymbol{w_{1,1}}, \cdots, \boldsymbol{w_{1,p}})$

Rappelons que les cours de statistique nous donnent l'expression de w0 et w1 en fonction de la base d'apprentissage. Notons W le vecteur-colonne des paramètres, Y le vecteur colonne des étiquettes, et X la matrice des observations, c'est-à-dire :

$$W = \begin{pmatrix} w_0 \\ \boldsymbol{w}_1 \end{pmatrix}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}, \quad X = \begin{pmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1d} \\ 1 & x_{21} & x_{22} & \ldots & x_{2d} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N1} & x_{N2} & \ldots & x_{Nd} \end{pmatrix}$$

Avec ces notations, on obtient l'estimation de W au sens des moindres carrés par les équations dites normales :

$$\boldsymbol{W} = (\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X})^{-1} . \boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{Y}$$

à la condition que $\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}$ soit une matrice inversible.

Démonstration : En effet, la quantité à minimiser en fonction des composantes de W est :

$$\sum_{n=1}^{N} |y_n - w_0 - \boldsymbol{w}_1 \cdot \boldsymbol{x}|^2 = ||Y - XW||_2^2$$

et on développe :

$$||Y - XW||_2^2 = (Y - XW)^T (Y - XW) = Y^T Y - 2W^T X^T Y + W^T X^T XW$$

Il s'agit d'une fonction quadratique convexe en , car $\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}$ est symétrique positive. L'unique minimum est donc atteint en W où le gradient s'annule; on déduit en calculant les dérivées partielles par rapport à chaque composante :

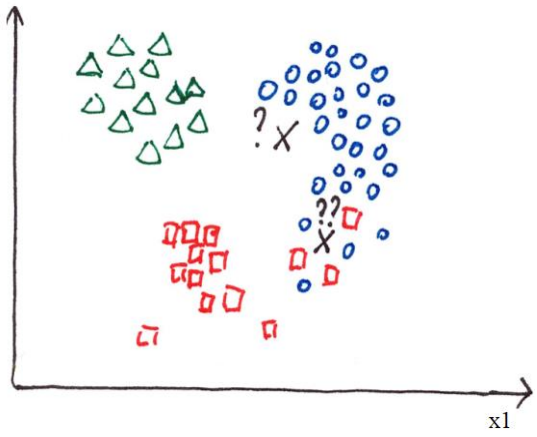
$$-2X^T Y + X^T XW = 0$$

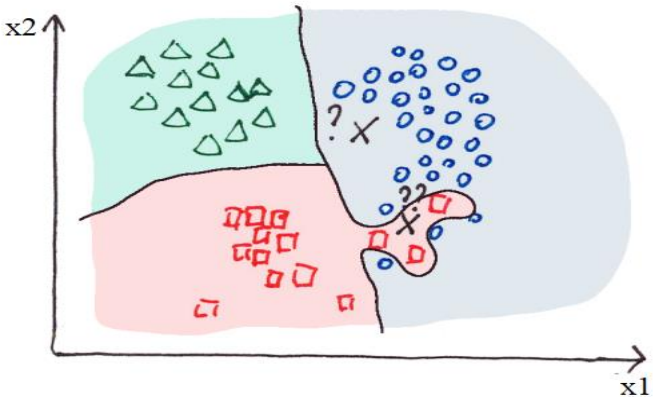
d'où les équations normales. Les cours de statistique discutent l'effet des observations influentes, qui sont telles que leur variation peut entraîner une forte variation des paramètres $\boldsymbol{w_0}$ et $\boldsymbol{w_1}$, et des caractéristiques colinéaires (ou fortement corrélées) qui peuvent entraîner la singularité (ou des problèmes de conditionnement) de la matrice $\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}$. Notons également que si le nombre d'observations N est inférieur à la dimension d, alors la matrice $\boldsymbol{X}^{\boldsymbol{T}} \boldsymbol{X}$ n'est pas inversible. En effet, si $N \le d$, les colonnes de X sont nécessairement liées

3.1.2. Classification : Prédire une catégorie (ex: diagnostiquer si une pièce est défectueuse ou non).

Lorsque les étiquettes y_n prennent leurs valeurs dans un ensemble fini dont les éléments correspondent à des catégories (ou classes) à identifier, on parle de classification supervisée. La fonction f_w associe alors une nouvelle observation x à une des classes. Elle définit donc une partition de l'ensemble des observations : chaque élément de cet ensemble

Figure 3 : *Classification supervisée. Ici, l'espace des observations est le plan bidimensionnel. Les observations ont des coordonnées* ($x1$, $x2$) *et sont étiquetées par trois catégories (triangle ,rond, carré). L'objectif est de déterminer quelle catégorie associer à une nouvelle observation non-étiquetée (représentée par une croix), à partir des observations étiquetées formant la base d'apprentissage. Si, pour l'observation marquée «***?** *», la tâche peut sembler facile (la classe est vraisemblablement « rond »), la classe de l'observation «***??** *» est plus discutable (« rond » ou « carré »?)*





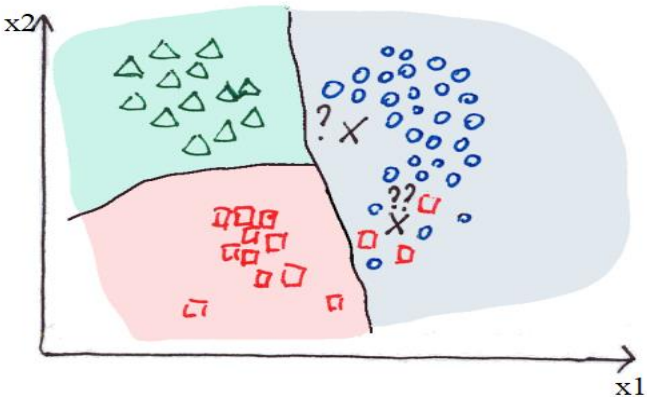


Figure 4 : *Frontières de séparation sur deux exemples. Quelle frontière de séparation est-elle la plus adaptée : une frontière complexe, permettant de classer correctement toutes les observations de la base d'apprentissage (à gauche), ou une frontière plus régulière, quitte à mal classer certaines de ces observations (à droite) ?*

3.2 Apprentissage non supervisé

Le modèle cherche des structures cachées par lui-même, car les données n'ont pas d'étiquettes.

- **Clustering :** Regrouper des données similaires (ex: segmenter des types de sols ou de matériaux sans étiquettes préalables, ex :Algorithme **K-means)**
- Réduction de dimension : **Principal Component Analysis (PCA)**

Sur la figure 5, les données non-étiquetées sont des points dans un espace bidimensionnel de composantes ($x1$, $x2$). *On peut chercher :*

- *une densité de probabilité ayant permis de générer les observations représentées sur la figure 5.*
- *A identifier des groupes.*

Dans cet exemple, il semble naturel d'identifier trois groupes. Notons qu'ils ne sont pas nécessairement isotropes (la « forme » d'un groupe n'est pas sphérique), n'ont pas le même nombre d'éléments, et que l'appartenance de certains points à un groupe ou l'autre est ambiguë.

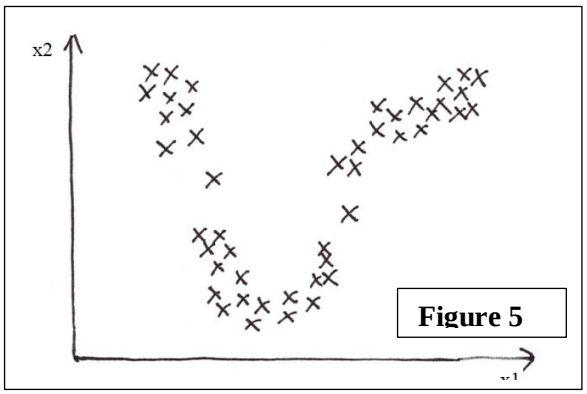


Figure 5

3.3 Apprentissage par renforcement (aperçu)

Un agent apprend par interaction avec un environnement

(L'agent apprend par l'expérience. Il effectue des actions

dans un environnement et reçoit des **récompenses** ou

des **punitions)**.

C'est le principe utilisé pour les robots autonomes ou les jeux (échecs, Go).

Éléments (Agent, Action, Récompense, Politique)

Applications (Robotique, Jeux, Optimisation dynamique)

NB :

- L'**apprentissage automatique** repose sur : Des données de qualité, Une bonne sélection de features, Une séparation correcte entraînement/validation/test, Une évaluation rigoureuse.
- Il constitue aujourd'hui un outil puissant en : Sciences des matériaux, Énergie, Médecine, Agrivoltaïque, Traitement du signal.

4) Exercices d'application

Exercice 1 : Identification des éléments

On donne : Température, pH, Temps → capacité d'adsorption

1. Identifier les features
2. Identifier la variable cible
3. Type d'apprentissage ? Justifier.

Réponses :

1. **Features (X) :** Température, pH, Temps

2. **Variable cible (y)** : Capacité d'adsorption

3. **Type d'apprentissage➞ Apprentissage supervisé – Régression**

   On dispose des couples (X, y) et la sortie est une valeur continue

Exercice 2 : Calcul de MSE

On a :

y réel	5	7	10
y prédit	4.5	6.8	9.5

1. Calculer le MSE
2. Calculer le RMSE

Réponse

y réel	5	7	10
y prédit	4.5	6.8	9.5
$e_i^2 = (y_i - \hat{y}_i)^2$	$(5-4.5)^2 = 0.25$	$(7-6.8)^2 = 0.04$	$(10-9.5)^2 = 0.25$

Somme = 0.54

$\boldsymbol{MSE = \frac{0.54}{3} = 0.18}$ **,** $\boldsymbol{RMSE = \sqrt{0.18} \approx 0.424}$, En moyenne, l'erreur est d'environ **0.42 unités**

Exercice 3 : Détection du sur-apprentissage

Un modèle donne : Erreur training = 0.001, Erreur validation = 0.45

1. Quel est le problème ?
2. Proposer deux solutions.

Solution

1) **Diagnostic➞** Overfitting sévère. **Pourquoi ? Le modèle :**

- Apprend parfaitement les données d'entraînement
- Ne généralise pas sur les données nouvelles

2) Régularisation (L1, L2), Réduction de la complexité du modèle, Augmenter la taille du dataset

Early stopping, Validation croisée (k-fold)

Modèles sensible au surapprentissage : Artificial Neural Network, Long Short-Term Memory

Exercice 4 : Petit jeu de données (34 échantillons)

On dispose de 34 expériences d'adsorption.

1. Peut-on utiliser ANN ?
2. Peut-on utiliser LSTM ?
3. Justifier scientifiquement

Réponse

1) Oui, mais : Risque élevé d'overfitting. Conditions : Architecture simple (1 couche cachée), Peu de neurones, Régularisation, Validation croisée
2) **En général : Non recommandé.** Car Long Short-Term Memory **est conçu pour : Données séquentielles, Séries temporelles**

Dépendances temporelles : Si vos 34 expériences sont indépendantes (conditions expérimentales discrètes), alors LSTM n'est pas conceptuellement approprié.

Recommandation scientifique Pour petit dataset :

- **Support Vector Regression**
- Régression linéaire régularisée
- Random Forest

Exercice 2 : Mise en œuvre du Pipeline (Python)

Complétez ce code pour séparer vos données avant d'entraîner un modèle :

```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
import numpy as np

# 1. Création de données fictives
X = np.random.rand(100, 3) # 100 échantillons, 3 caractéristiques
y = np.random.rand(100)    # 100 cibles

# 2. SEPARATION DES DONNÉES (Étape cruciale du pipeline)
# On garde 20% pour le test final
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# 3. Création du modèle
ann = MLPRegressor(hidden_layer_sizes=(10,), max_iter=1000)

# 4. Entraînement uniquement sur le TRAIN
ann.fit(X_train, y_train)

# 5. Évaluation sur le TEST
score = ann.score(X_test, y_test)
print(f"Précision (R²) sur le test : {score:.2f}")
```