Formation Machine Learning

Apprentissage Supervisé

Bassem Ben Hamed

ENET'Com, Université de Sfax

17 février 2026

Plan du Cours

Plan de la section

Cadre Formel de l'Apprentissage Supervisé

Définition

L'apprentissage supervisé consiste à apprendre une fonction de prédiction à partir de données étiquetées.

- $\mathcal{X}$: espace des entrées (features)
- $\mathcal{Y}$: espace des sorties (labels)
- $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$: ensemble d'apprentissage

où $x_i \in \mathcal{X} \subseteq \mathbb{R}^d$ et $y_i \in \mathcal{Y}$

Objectif

Apprendre une fonction $h : \mathcal{X} \to \mathcal{Y}$ qui minimise l'erreur de prédiction sur de nouvelles données.

Risque Empirique

Definition (Risque Empirique)

Le risque empirique est défini par :

$$\hat{R}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(h(x_i), y_i)$$

où $\mathcal{L}$ est la fonction de perte mesurant l'écart entre prédiction et vérité.

Principe ERM (Empirical Risk Minimization)

$$h^* = \arg\min_{h \in \mathcal{H}} \hat{R}(h)$$

où $\mathcal{H}$ est l'espace des hypothèses.

Risque Réel vs. Risque Empirique

Risque Réel

$$R(h) = \mathbb{E}_{(x,y)\sim P}[\mathcal{L}(h(x), y)]$$

- Inconnue en pratique
- Distribution P inconnue
- Objectif véritable

Risque Empirique

$$\hat{R}(h) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(h(x_i), y_i)$$

- Calculable
- Approximation de $R(h)$
- Base de l'apprentissage

Loi des Grands Nombres

Sous conditions de régularité : $\hat{R}(h) \xrightarrow{n\to\infty} R(h)$ presque sûrement

Décomposition Biais-Variance

Theorem (Décomposition de l'Erreur)

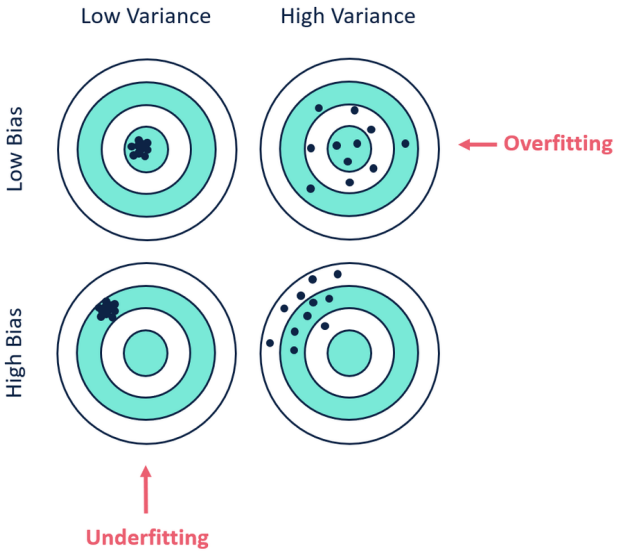
Pour une fonction apprise h et un point fixé x :

$$\mathbb{E}[(h(x) - y)^2] = \underbrace{\text{Bias}^2(h(x))}_{\text{Biais}^2} + \underbrace{\text{Var}(h(x))}_{\text{Variance}} + \underbrace{\sigma^2}_{\text{Bruit irréductible}}$$

où :

- $\text{Bias}(h(x)) = \mathbb{E}[h(x)] - f(x)$: écart entre prédiction moyenne et vraie fonction
- $\text{Var}(h(x)) = \mathbb{E}[(h(x) - \mathbb{E}[h(x)])^2]$: sensibilité aux variations des données
- σ^2 : variance du bruit

Trade-off Biais-Variance



Classification vs. Régression

Classification

Espace de sortie discret

$$\mathcal{Y} = \{c_1, c_2, \ldots, c_K\}$$

Exemples :

- Détection de spam
- Diagnostic médical
- Reconnaissance d'images

Perte 0-1 :

$$\mathcal{L}_{0-1}(\hat{y}, y) = \mathbb{1}\{\hat{y} \neq y\}$$

Régression

Espace de sortie continu

$$\mathcal{Y} = \mathbb{R} \text{ ou } \mathcal{Y} \subseteq \mathbb{R}^p$$

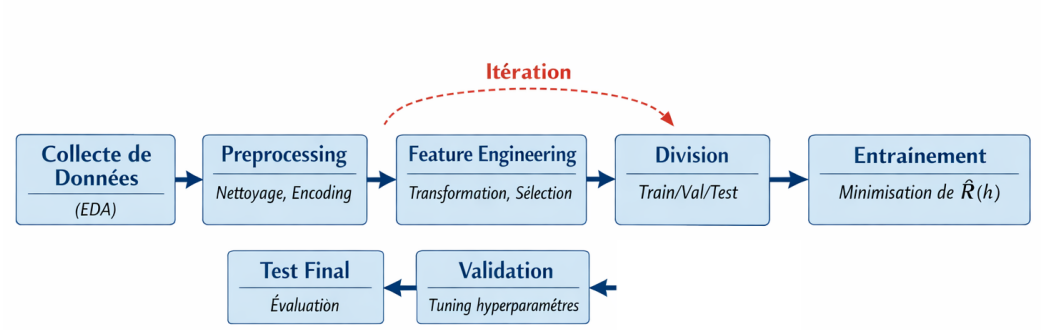
Exemples :

- Prix immobiliers
- Prévision météo
- Estimation d'âge

MSE :

$$\mathcal{L}_{\text{MSE}}(\hat{y}, y) = (\hat{y} - y)^2$$

Pipeline de Machine Learning

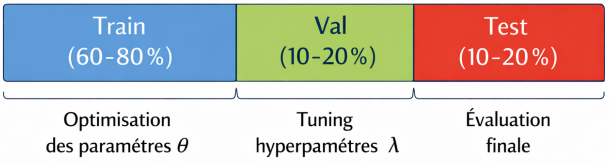


Division Train/Val/Test

Principe Fondamental

$$\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{val}} \cup \mathcal{D}_{\text{test}}$$

avec $\mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{val}} = \mathcal{D}_{\text{train}} \cap \mathcal{D}_{\text{test}} = \mathcal{D}_{\text{val}} \cap \mathcal{D}_{\text{test}} = \emptyset$



Règle d'Or

Ne **JAMAIS** toucher à $\mathcal{D}_{\text{test}}$ avant l'évaluation finale !

Plan de la section

Gestion des Valeurs Manquantes

Stratégies d'Imputation

Statistiques simples :

- Moyenne : $\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$
- Médiane : valeur centrale
- Mode : valeur la plus fréquente

Méthodes avancées :

- KNN Imputation
- Régression
- MICE (Multiple Imputation)

Types de Données Manquantes

MCAR : Missing Completely At Random

$$P(M \mid X_{\text{obs}}, X_{\text{mis}}) = P(M)$$

MAR : Missing At Random

$$P(M \mid X_{\text{obs}}, X_{\text{mis}}) = P(M \mid X_{\text{obs}})$$

MNAR : Missing Not At Random

- Dépend de la valeur manquante elle-même

Normalisation et Standardisation

Min-Max Scaling (Normalisation)

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \in [0, 1]$$

- Borne les valeurs entre 0 et 1
- Sensible aux outliers

Z-score Standardisation

$$x' = \frac{x - \mu}{\sigma} \sim \mathcal{N}(0, 1), \quad \mu = \mathbb{E}[x],\ \sigma^2 = \mathrm{Var}(x)$$

- Centre les données autour de 0 avec variance unitaire
- Plus robuste aux outliers que Min-Max
- Nécessaire pour algorithmes basés sur distance (KNN, SVM)

Encodage des Variables Catégorielles

One-Hot Encoding

Pour une variable avec K catégories : $\{c_1, c_2, \ldots, c_K\}$

$$x_{\text{cat}} = c_k \Rightarrow \boldsymbol{x}_{\text{enc}} = [0, \ldots, 0, \underbrace{1}_{k\text{-ième}}, 0, \ldots, 0] \in \{0, 1\}^K$$

Exemple : Couleur = {Rouge, Vert, Bleu}

$$\begin{aligned} \text{Rouge} &\rightarrow [1, 0, 0] \\ \text{Vert} &\rightarrow [0, 1, 0] \\ \text{Bleu} &\rightarrow [0, 0, 1] \end{aligned}$$

Attention

Peut créer des features très dimensionnelles si K est grand ! $\Rightarrow$ Curse of dimensionality

Feature Selection

Méthodes de Filtrage

Corrélation de Pearson :

$$\rho_{X,Y} = \frac{\text{Cov}(X,Y)}{\sigma_X \sigma_Y}$$

Information mutuelle :

$$I(X;Y) = \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

Chi-deux (χ^2) :

$$\chi^2 = \sum_{i,j} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Méthodes Embarquées

Lasso (L1) :

$$\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|_1$$

Force certains coefficients à 0

Tree-based :

- Feature importance de Random Forest
- Gain de XGBoost

Plan de la section

Modèle de Régression Linéaire

Definition (Régression Linéaire)

Le modèle de régression linéaire suppose une relation affine :

$$y = \boldsymbol{w}^\top \boldsymbol{x} + b + \varepsilon = \sum_{j=1}^{d} w_j x_j + b + \varepsilon$$

- $\boldsymbol{w} \in \mathbb{R}^d$: vecteur de poids
- $b \in \mathbb{R}$: biais (intercept)
- $\varepsilon \sim \mathcal{N}(0, \sigma^2)$: bruit gaussien

Forme Matricielle

Pour n observations :

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{w} + \boldsymbol{\varepsilon}$$

où $\boldsymbol{X} \in \mathbb{R}^{n \times d}$, $\boldsymbol{y} \in \mathbb{R}^n$, $\boldsymbol{w} \in \mathbb{R}^d$

Moindres Carrés Ordinaires (OLS)

Critère d'Optimisation

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 = \arg\min_{\boldsymbol{w}} \sum_{i=1}^{n} (y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2$$

Theorem (Solution Analytique)

Si $\boldsymbol{X}^\top \boldsymbol{X}$ est inversible, la solution des moindres carrés est :

$$\boxed{\boldsymbol{w}^* = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}}$$

Dérivation : Gradient nul $\nabla_{\boldsymbol{w}} \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 = -2\boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = 0$

Régularisation Ridge (L2)

Problème du Surajustement

OLS peut donner des poids $\boldsymbol{w}$ très grands si multicolinéarité ou d proche de n.

Definition (Ridge Regression)

Ajouter un terme de régularisation L2 :

$$\boldsymbol{w}^*_{\text{Ridge}} = \arg\min_{\boldsymbol{w}} \left\{ \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|_2^2 \right\}$$

où $\lambda > 0$ est le paramètre de régularisation.

Solution Analytique

$$\boxed{\boldsymbol{w}^*_{\text{Ridge}} = (\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}}$$

Régularisation Lasso (L1)

Definition (Lasso Regression)

Régularisation L1 pour sélection de features :

$$\boldsymbol{w}^*_{\text{Lasso}} = \arg\min_{\boldsymbol{w}} \left\{ \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \|\boldsymbol{w}\|_1 \right\}$$

où $\|\boldsymbol{w}\|_1 = \sum_{j=1}^{d} |w_j|$

Avantages

- Force certains w_j à être exactement 0
- Sélection automatique de features
- Modèle interprétable

Limitation

- Pas de solution analytique
- Nécessite optimisation itérative
- Moins stable que Ridge

Elastic Net

Definition (Elastic Net)

Combinaison de régularisations L1 et L2 :

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left\{ \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda_1 \|\boldsymbol{w}\|_1 + \lambda_2 \|\boldsymbol{w}\|_2^2 \right\}$$

ou de manière équivalente :

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left\{ \|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2 + \lambda \left[\alpha \|\boldsymbol{w}\|_1 + (1-\alpha) \|\boldsymbol{w}\|_2^2 \right] \right\}$$

avec $\alpha \in [0, 1]$ contrôlant le mélange L1/L2.

Propriétés

- $\alpha = 1$: Lasso pur
- $\alpha = 0$: Ridge pur
- $0 < \alpha < 1$: compromis entre sélection et stabilité

Plan de la section

Definition (Modèle Logistique)

Pour classification binaire $y \in \{0, 1\}$, modéliser la probabilité :

$$P(y = 1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{w}^\top \boldsymbol{x} + b) = \frac{1}{1 + e^{-(\boldsymbol{w}^\top \boldsymbol{x} + b)}}$$

où $\sigma(z) = \frac{1}{1+e^{-z}}$ est la fonction sigmoïde.

Maximum de Vraisemblance

Fonction de Vraisemblance

Pour n observations indépendantes :

$$\mathcal{L}(\boldsymbol{w}) = \prod_{i=1}^{n} P(y_i \mid \boldsymbol{x}_i) = \prod_{i=1}^{n} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)^{y_i} (1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i))^{1-y_i}$$

Log-Vraisemblance (à maximiser)

$$\ell(\boldsymbol{w}) = \log \mathcal{L}(\boldsymbol{w}) = \sum_{i=1}^{n} \Big[y_i \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i) + (1 - y_i) \log(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)) \Big]$$

Cross-Entropy Loss (à minimiser)

$$J(\boldsymbol{w}) = -\frac{1}{n} \ell(\boldsymbol{w})$$

Optimisation par Gradient

Gradient de la Log-Vraisemblance

$$\nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}) = \sum_{i=1}^{n} (y_i - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_i)) \boldsymbol{x}_i = \boldsymbol{X}^\top (\boldsymbol{y} - \boldsymbol{\sigma})$$

où $\boldsymbol{\sigma} = [\sigma(\boldsymbol{w}^\top \boldsymbol{x}_1), \ldots, \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)]^\top$

Gradient Descent

Mise à jour itérative :

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta \nabla_{\boldsymbol{w}} \ell(\boldsymbol{w}^{(t)})$$

où $\eta > 0$ est le learning rate.

Alternative : Newton-Raphson (convergence plus rapide mais coût en $O(d^3)$)

Régression Logistique Multi-Classes

Definition (Softmax)

Pour K classes, généralisation avec softmax :

$$P(y = k \mid \boldsymbol{x}) = \frac{e^{\boldsymbol{w}_k^\top \boldsymbol{x}}}{\sum_{j=1}^{K} e^{\boldsymbol{w}_j^\top \boldsymbol{x}}} = \mathrm{softmax}_k(\boldsymbol{w}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{w}_K^\top \boldsymbol{x})$$

Cross-Entropy Loss Multi-Classes

$$J(\boldsymbol{W}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log P(y = k \mid \boldsymbol{x}_i)$$

où $y_{ik} = \mathbb{1}\{y_i = k\}$ (one-hot encoding).

Plan de la section

Principe du KNN

Definition (K-Nearest Neighbors)

Classification non-paramétrique basée sur la similarité :

1. Trouver les k voisins les plus proches de $\boldsymbol{x}$ dans $\mathcal{D}_{\text{train}}$
2. Classification : vote majoritaire parmi ces k voisins
3. Régression : moyenne des valeurs des k voisins

Prédiction

Classification :

$$\hat{y} = \arg\max_{c \in \mathcal{Y}} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} \mathbb{1}\{y_i = c\}$$

Régression :

$$\hat{y} = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\boldsymbol{x})} y_i$$

Métriques de Distance

Distance Euclidienne (L2)

$$d_2(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{\sum_{j=1}^{d} (x_j - x'_j)^2} = \|\boldsymbol{x} - \boldsymbol{x}'\|_2$$

Distance de Manhattan (L1)

$$d_1(\boldsymbol{x}, \boldsymbol{x}') = \sum_{j=1}^{d} |x_j - x'_j| = \|\boldsymbol{x} - \boldsymbol{x}'\|_1$$

Distance de Minkowski (Lp)

$$d_p(\boldsymbol{x}, \boldsymbol{x}') = \left(\sum_{j=1}^{d} |x_j - x'_j|^p \right)^{1/p}$$

Choix de k et Propriétés

k petit

- Frontières complexes
- Sensible au bruit
- **Variance élevée**
- Risque d'overfitting

k grand

- Frontières lisses
- Robuste au bruit
- **Biais élevé**
- Risque d'underfitting

Complexité

Entraînement : $O(1)$
Aucun calcul !

Prédiction : $O(n \cdot d)$
Calcul de distances pour tous les points

Optimisation :

- KD-Tree
- Ball Tree
- LSH (Locality Sensitive Hashing)

Avantages et Limitations

Avantages

- Simple et intuitif
- Pas d'hypothèse sur les données
- Non-paramétrique
- Naturellement multi-classes
- Pas de phase d'entraînement

Limitations

- Prédiction coûteuse en $O(nd)$
- Sensible aux features non pertinentes
- **Curse of dimensionality**
- Nécessite normalisation
- Stockage de toutes les données

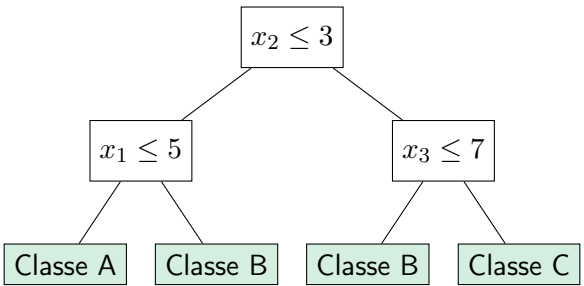
Curse of Dimensionality

En haute dimension, tous les points deviennent équidistants :

$$\lim_{d \to \infty} \frac{d_{\max}(\boldsymbol{x}, \mathcal{D}) - d_{\min}(\boldsymbol{x}, \mathcal{D})}{d_{\min}(\boldsymbol{x}, \mathcal{D})} \to 0$$

Plan de la section

Structure d'un Arbre de Décision



- **Nœuds internes** : tests sur une feature
- **Branches** : résultats du test
- **Feuilles** : prédictions (classe ou valeur)

Critères de Division - Classification

Impureté de Gini

$$\text{Gini}(S) = 1 - \sum_{k=1}^{K} p_k^2$$

où $p_k = \frac{1}{|S|} \sum_{i \in S} \mathbb{1}\{y_i = k\}$ est la proportion de classe k dans S.

Entropie (Information Gain)

$$H(S) = -\sum_{k=1}^{K} p_k \log_2 p_k$$

Gain d'Information

$$\text{Gain}(S, A) = H(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} H(S_v)$$

Critères de Division - Régression

Variance Reduction (MSE)

Pour un nœud S, variance :

$$\text{Var}(S) = \frac{1}{|S|} \sum_{i \in S} (y_i - \bar{y}_S)^2$$

où $\bar{y}_S = \frac{1}{|S|} \sum_{i \in S} y_i$

Réduction de Variance

Choisir la division qui maximise :

$$\Delta\text{Var}(S, A, t) = \text{Var}(S) - \left(\frac{|S_L|}{|S|} \text{Var}(S_L) + \frac{|S_R|}{|S|} \text{Var}(S_R) \right)$$

où $S_L = \{\boldsymbol{x} \in S : x_A \leq t\}$ et $S_R = \{\boldsymbol{x} \in S : x_A > t\}$

Algorithme de Construction (CART)

Algorithm 1 Construction d'un Arbre de Décision

Require: Ensemble S, profondeur maximale $d_{\max}$, taille minimale $n_{\min}$
Ensure: Arbre de décision T

if $|S| < n_{\min}$ **or** profondeur $= d_{\max}$ **or** S pur **then**
  return Feuille avec prédiction $\hat{y} =$ majoritaire(S) ou $\bar{y}_S$
end if
Trouver meilleure division (A^*, t^*) maximisant le gain
Créer nœud de décision $x_A \leq t$
$S_L \leftarrow \{\boldsymbol{x} \in S : x_{A^*} \leq t^*\}$
$S_R \leftarrow \{\boldsymbol{x} \in S : x_{A^*} > t^*\}$
$T_L \leftarrow$ **construire**$(S_L, d_{\max}, n_{\min})$
$T_R \leftarrow$ **construire**$(S_R, d_{\max}, n_{\min})$
return Nœud(A^*, t^*, T_L, T_R)

Pruning (Élagage)

Problème du Surajustement

Un arbre complet s'ajuste parfaitement aux données d'entraînement mais généralise mal.

Pre-Pruning (Early Stopping)

Arrêter la croissance selon des critères :

- Profondeur maximale : depth $\leq d_{\max}$
- Taille minimale de nœud : $|S| \geq n_{\min}$
- Gain minimum : $\Delta\text{Gain} \geq \epsilon$

Post-Pruning (Cost Complexity Pruning)

Pénaliser la complexité de l'arbre :

$$\text{Cost}(T) = \text{Error}(T) + \alpha|T|$$

Avantages et Limitations

Avantages

- Très interprétable
- Gère features numériques et catégorielles
- Pas de normalisation nécessaire
- Capture non-linéarités
- Feature selection implicite
- Robuste aux outliers

Limitations

- Instable (variance élevée)
- Surajustement facile
- Frontières en escalier
- Biais vers features à nombreuses valeurs
- Performances limitées seul

Solution

Utiliser des **ensembles d'arbres** : Random Forest, Gradient Boosting

Plan de la section

Principe du Bagging

Definition (Bootstrap Aggregating (Bagging))

1. Créer B échantillons bootstrap de $\mathcal{D}$:

$$\mathcal{D}_b = \{(\boldsymbol{x}_{i_1}, y_{i_1}), \ldots, (\boldsymbol{x}_{i_n}, y_{i_n})\} \text{ avec } i_j \sim \text{Uniform}(\{1, \ldots, n\})$$

2. Entraîner un modèle h_b sur chaque $\mathcal{D}_b$
3. Agréger les prédictions :

Agrégation

Classification : Vote majoritaire

$$\hat{y} = \text{mode}(\{h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_B(\boldsymbol{x})\})$$

Régression : Moyenne

$$\hat{y} = \frac{1}{B} \sum_{b}^{B} h_b(\boldsymbol{x})$$

Random Forest

Definition (Random Forest)

Bagging + randomisation des features :

- À chaque nœud, considérer seulement m features aléatoires parmi d
- Typiquement : $m = \sqrt{d}$ (classification) ou $m = d/3$ (régression)
- Entraîner B arbres profonds (sans pruning)

Pourquoi Randomiser les Features ?

- Réduit la corrélation entre arbres
- Empêche une feature dominante d'être toujours choisie
- Augmente la diversité des arbres
- **Réduit la variance** sans augmenter le biais

Réduction de Variance

Theorem (Réduction de Variance par Bagging)

Soit $h_1, \ldots, h_B$ des modèles identiquement distribués avec variance σ^2 et corrélation ρ.
L'agrégation a pour variance :

$$\mathrm{Var}\left(\frac{1}{B}\sum_{b=1}^{B} h_b\right) = \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2$$

Interprétation :

- Si $\rho = 0$ (indépendance) : $\mathrm{Var} = \sigma^2/B$ (division par B)
- Si $\rho = 1$ (parfaite corrélation) : $\mathrm{Var} = \sigma^2$ (aucun gain)
- Random Forest vise à réduire ρ via randomisation

Out-of-Bag (OOB) Error

Estimation OOB

Pour chaque arbre b, environ 37% des observations ne sont pas dans $\mathcal{D}_b$ (bootstrap).

Pour chaque $(\boldsymbol{x}_i, y_i)$:

1. Trouver les arbres où $(\boldsymbol{x}_i, y_i)$ est OOB
2. Prédire $\hat{y}_i^{\text{OOB}}$ en moyennant ces arbres
3. Calculer l'erreur OOB

Avantage

$$\text{OOB Error} \approx \text{Validation Error}$$

Pas besoin de set de validation séparé !

Feature Importance

Mean Decrease Impurity (MDI)

Pour chaque feature j :

$$\text{Importance}_j = \frac{1}{B} \sum_{b=1}^{B} \sum_{t \in T_b : \text{split sur } j} \Delta\text{Impurity}(t)$$

où $\Delta\text{Impurity}(t)$ est la réduction d'impureté au nœud t.

Mean Decrease Accuracy (MDA)

1. Calculer l'OOB error de référence
2. Permuter aléatoirement la feature j dans les données OOB
3. Calculer le nouvel OOB error
4. Importance = différence d'erreur

$$\text{Importance}_j = \text{OOB Error}_{\text{permuted } j} - \text{OOB Error}_{\text{original}}$$

Principaux Hyperparamètres

- B : nombre d'arbres
  - Plus grand = meilleur (convergence)
  - Typique : 100-500
  - Pas de surajustement (au pire plateau)
- m : nombre de features par split
  - Classification : $m = \sqrt{d}$
  - Régression : $m = d/3$
- `max_depth` : profondeur maximale
  - Souvent : aucune limite (arbres complets)
  - Peut limiter pour réduire temps de calcul
- `min_samples_split` : taille minimale pour diviser
  - Typique : 2-10

Plan de la section

Boosting vs. Bagging

Bagging (Random Forest)

- Arbres **indépendants** en parallèle
- Bootstrap + randomisation
- Réduit la **variance**
- Vote majoritaire / moyenne

Boosting (XGBoost)

- Arbres **séquentiels**
- Chaque arbre corrige les erreurs précédentes
- Réduit le **biais**
- Somme pondérée

Formulation Additive

$$\hat{y}_i = \sum_{t=1}^{T} f_t(\boldsymbol{x}_i)$$

où f_t est un arbre ajouté à l'itération t.

Definition (Gradient Boosting Machine (GBM))

À chaque itération t :

1. Calculer les résidus (erreurs) :

$$r_{it} = -\frac{\partial \mathcal{L}(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$$

2. Entraîner un arbre f_t pour prédire r_{it}
3. Mettre à jour :

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(\boldsymbol{x}_i)$$

où $\eta \in (0, 1]$ est le learning rate (shrinkage).

XGBoost : Objectif

Definition (Fonction Objectif de XGBoost)

$$\mathcal{J}^{(t)} = \sum_{i=1}^{n} \mathcal{L}(y_i, \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i)) + \Omega(f_t)$$

où :

- $\mathcal{L}$: fonction de perte (MSE, log-loss, etc.)
- $\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$: régularisation

Régularisation

- T : nombre de feuilles dans l'arbre f_t
- w_j : poids de la feuille j
- γ : pénalité sur le nombre de feuilles (complexité)
- λ : régularisation L2 sur les poids

Approximation de Second Ordre

Développement de Taylor

Approximation d'ordre 2 de la perte :

$$\mathcal{L}(y_i, \hat{y}_i^{(t-1)} + f_t(\boldsymbol{x}_i)) \approx \mathcal{L}(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i)$$

où :

- $g_i = \frac{\partial \mathcal{L}(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}}$: gradient (premier ordre)
- $h_i = \frac{\partial^2 \mathcal{L}(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2}$: hessienne (second ordre)

Objectif Simplifié

$$\tilde{\mathcal{J}}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(\boldsymbol{x}_i) + \frac{1}{2} h_i f_t^2(\boldsymbol{x}_i) \right] + \Omega(f_t)$$

Poids Optimal d'une Feuille

Pour une feuille j contenant $I_j = \{i : f_t(\boldsymbol{x}_i) = w_j\}$:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$$

Gain de Split

Pour une division créant I_L et I_R :

$$\text{Gain} = \frac{1}{2}\left[\frac{(\sum_{i \in I_L} g_i)^2}{\sum_{i \in I_L} h_i + \lambda} + \frac{(\sum_{i \in I_R} g_i)^2}{\sum_{i \in I_R} h_i + \lambda} - \frac{(\sum_{i \in I} g_i)^2}{\sum_{i \in I} h_i + \lambda}\right] - \gamma$$

Choix de split : Maximiser ce gain.

Hyperparamètres Clés de XGBoost

Paramètres de Boosting

- `n_estimators` : nombre d'arbres T
- `learning_rate` (η) : shrinkage, typique 0.01-0.3
- `subsample` : fraction d'échantillons par arbre (0.5-1.0)
- `colsample_bytree` : fraction de features par arbre

Paramètres d'Arbre

- `max_depth` : profondeur maximale (3-10)
- `min_child_weight` : somme minimale de h_i dans une feuille
- `gamma` (γ) : réduction minimale de perte pour split

Régularisation

- `reg_alpha` : régularisation L1 sur poids
- `reg_lambda` (λ) : régularisation L2 sur poids

Avantages de XGBoost

Forces

- **Performances exceptionnelles** : gagnant de nombreuses compétitions Kaggle
- Régularisation intégrée (γ, λ) : prévient l'overfitting
- Gestion native des valeurs manquantes
- Optimisations algorithmiques (parallélisation, cache-aware)
- Approximation de second ordre : convergence plus rapide
- Pruning intelligent (max-depth avec pénalité)
- Support de fonctions de perte personnalisées

Limitations

- Tuning d'hyperparamètres complexe
- Sensible au surajustement si mal configuré
- Moins interprétable qu'un arbre unique

Plan de la section

Validation Croisée (Cross-Validation)

Definition (K-Fold Cross-Validation)

1. Diviser $\mathcal{D}$ en K folds $F_1, \ldots, F_K$ de taille égale
2. Pour $k = 1, \ldots, K$:
   - Entraîner sur $\mathcal{D} \setminus F_k$
   - Valider sur F_k
   - Enregistrer Score_k
3. Score CV :

$$\text{CV Score} = \frac{1}{K} \sum_{k=1}^{K} \text{Score}_k$$

Choix typique : $K = 5$ ou $K = 10$

Cas limite : $K = n$ (Leave-One-Out CV) - très coûteux

Grid Search

Definition (Grid Search)

Recherche exhaustive dans une grille d'hyperparamètres :

$$\lambda^* = \underset{\lambda \in \Lambda_{\text{grid}}}{\arg\min}\ \text{CV Score}(\lambda)$$

où $\Lambda_{\text{grid}} = \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_p$

Exemple

Pour XGBoost :

- `n_estimators` $\in \{50, 100, 200, 500\}$
- `learning_rate` $\in \{0.01, 0.05, 0.1, 0.3\}$
- `max_depth` $\in \{3, 5, 7, 10\}$

Nombre total de combinaisons : $4 \times 4 \times 4 = 64$

Randomized Search

Definition (Randomized Search)

Au lieu de grille exhaustive, échantillonner aléatoirement :

1. Définir distributions pour chaque hyperparamètre
2. Échantillonner N configurations aléatoires
3. Évaluer chaque configuration par CV
4. Retenir la meilleure

Avantages

- Plus rapide que Grid Search
- Explore mieux l'espace
- Budget contrôlable (N)

Distributions Typiques

- Uniforme : learning_rate $\sim \mathcal{U}(0.01, 0.3)$
- Log-uniforme : $\lambda \sim 10^{\mathcal{U}(-3,1)}$
- Entiers : max_depth $\sim$ Unif$\{3, \ldots, 10\}$

Bayesian Optimization

Definition (Optimisation Bayésienne)

Approche probabiliste pour trouver $\lambda^* = \arg\min_\lambda f(\lambda)$ où f est coûteux à évaluer.

Principe

1. Maintenir un modèle probabiliste de f (typiquement : Gaussian Process)

$$f(\lambda) \sim \mathcal{GP}(\mu(\lambda), k(\lambda, \lambda'))$$

2. Choisir λ_{next} optimisant une **fonction d'acquisition** :
   - **Expected Improvement (EI)**
   - **Upper Confidence Bound (UCB)**
   - **Probability of Improvement (PI)**
3. Évaluer $f(\lambda_{\text{next}})$, mettre à jour le GP
4. Répéter

Expected Improvement

Definition (Expected Improvement)

$$\mathrm{EI}(\lambda) = \mathbb{E}\left[\max(f_{\min} - f(\lambda), 0)\right]$$

où $f_{\min}$ est la meilleure valeur observée.

Forme Analytique (GP)

Soit $\mu(\lambda)$ et $\sigma(\lambda)$ la moyenne et écart-type prédits par le GP :

$$\mathrm{EI}(\lambda) = \begin{cases} (f_{\min} - \mu(\lambda))\Phi(Z) + \sigma(\lambda)\phi(Z) & \text{si } \sigma(\lambda) > 0 \\ 0 & \text{si } \sigma(\lambda) = 0 \end{cases}$$

- $Z = \frac{f_{\min} - \mu(\lambda)}{\sigma(\lambda)}$
- Φ : CDF de $\mathcal{N}(0,1)$
- ϕ : PDF de $\mathcal{N}(0,1)$

Comparaison des Méthodes

Méthode	Complexité	Efficacité	Garanties
Grid Search	$O(\|\Lambda_1\| \times \cdots \times \|\Lambda_p\|)$	Faible	Exhaustive
Random Search	$O(N)$	Moyenne	Probabiliste
Bayesian Opt.	$O(N)$	Élevée	Convergence

Recommandations

- **Grid Search** : espace petit (< 100 configs), exploration exhaustive
- **Random Search** : budget limité, bonne baseline
- **Bayesian Opt.** : fonction coûteuse, optimisation fine (compétitions)

Plan de la section

Métriques de Régression

Mean Squared Error (MSE)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE)

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

Avantage : même unité que y

Mean Absolute Error (MAE)

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Coefficient de Détermination R^2

Definition (R-squared)

$$R^2 = 1 - \frac{\text{SS}_{\text{res}}}{\text{SS}_{\text{tot}}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

où :

- $\text{SS}_{\text{res}} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$: somme des carrés résiduels
- $\text{SS}_{\text{tot}} = \sum_{i=1}^{n}(y_i - \bar{y})^2$: somme des carrés totale
- $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$: moyenne

Interprétation

- $R^2 = 1$: prédiction parfaite
- $R^2 = 0$: modèle aussi bon que la moyenne
- $R^2 < 0$: modèle pire que la moyenne (possible en test)

Matrice de Confusion

	Prédite Positive	Prédite Négative
Vraie Positive	TP (True Positive)	FN (False Negative)
Vraie Negative	FP (False Positive)	TN (True Negative)

- **TP** : Vrais Positifs (correctement prédits positifs)
- **TN** : Vrais Négatifs (correctement prédits négatifs)
- **FP** : Faux Positifs (erreur de type I)
- **FN** : Faux Négatifs (erreur de type II)

Métriques de Classification

Accuracy (Exactitude)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = \frac{\text{Correct}}{n}$$

Precision (Précision)

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

"Parmi les prédictions positives, combien sont vraies ?"

Recall (Rappel / Sensibilité)

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

"Parmi les vrais positifs, combien ai-je détectés ?"

F1-Score et F_β-Score

Definition (F1-Score)

Moyenne harmonique de Precision et Recall :

$$F_1 = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}}$$

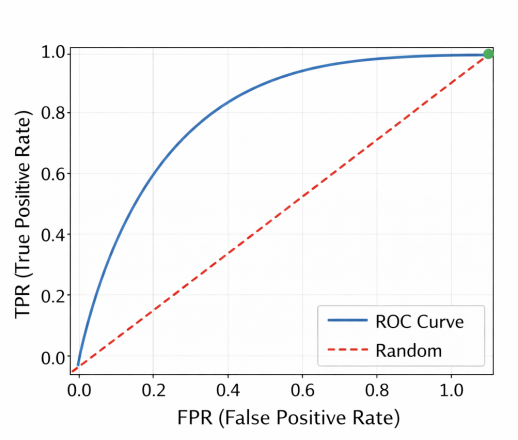
Definition (F_β-Score)

Généralisation pondérant Recall β fois plus que Precision :

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{Precision} \times \text{Recall}}{\beta^2 \cdot \text{Precision} + \text{Recall}}$$

- $\beta = 1$: F1-Score (équilibre)
- $\beta = 2$: F2-Score (favorise Recall)
- $\beta = 0.5$: F0.5-Score (favorise Precision)

Courbe ROC et AUC



ROC Curve

Trace TPR vs FPR pour tous seuils τ :

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

AUC (Area Under Curve)

$$\text{AUC} \in [0, 1]$$

- AUC = 1 : parfait
- AUC = 0.5 : hasard
- AUC > 0.8 : très bon

Métriques pour Classes Déséquilibrées

Problème

Avec 99% de classe 0 et 1% de classe 1 :

- Prédire toujours 0 $\Rightarrow$ Accuracy = 99% mais Recall(classe 1) = 0%

Métriques Adaptées

Balanced Accuracy :

$$\text{Balanced Acc} = \frac{1}{K}\sum_{k=1}^{K}\text{Recall}_k$$

Cohen's Kappa :

$$\kappa = \frac{p_0 - p_e}{1 - p_e}$$

où p_0 est l'accuracy observée et p_e l'accuracy attendue par hasard.

Autres : F1-Score, AUC-ROC, AUC-PR

Plan de la section

Récapitulatif des Algorithmes

Algorithme	Type	Interprétabilité	Performance
Régression Linéaire	Régression	+++++	+
Régression Logistique	Classification	++++	++
KNN	Les deux	+++	++
Decision Tree	Les deux	++++	++
Random Forest	Les deux	++	++++
XGBoost	Les deux	++	+++++

Principes Clés

- Partir simple, complexifier si nécessaire
- Toujours faire train/val/test
- Choisir métriques adaptées au problème
- Random Forest et XGBoost : excellent compromis
- Tuning crucial pour performances optimales

Pour Aller Plus Loin

Sujets Avancés

- **Deep Learning** : réseaux de neurones profonds (CNN, RNN, Transformers)
- **Apprentissage Non-Supervisé** : clustering (K-means, DBSCAN), réduction de dimensionnalité (PCA, t-SNE)
- **Apprentissage par Renforcement** : Q-learning, Policy Gradient
- **Time Series Forecasting** : ARIMA, Prophet, LSTM
- **Natural Language Processing** : BERT, GPT, word embeddings
- **Computer Vision** : ResNet, YOLO, Semantic Segmentation

Ressources

- *The Elements of Statistical Learning* - Hastie, Tibshirani, Friedman
- *Pattern Recognition and Machine Learning* - Bishop
- *Deep Learning* - Goodfellow, Bengio, Courville

Merci de votre attention !

Questions ?

Bonne continuation dans votre apprentissage du Machine Learning !