DU Python Big Data

Machine Learning

Partie 2

gilles.michel@sudintralog.com
@unimes.fr

Principe

Mettre en œuvre des algorithmes améliorant **automatiquement** les performances d'un système

Plan

I. Régressions
II. Arbres de décision
III. Forêt aléatoire et apprentissage d'ensemble
IV. SVM (Support Vector Machine)

Descente de gradient

- A chaque étape de l'apprentissage, on veut minimiser l'erreur globale *J*

$$J(paramètres) = \frac{1}{2} \sum_{BDD} (S_{calculée} - S_{attendue})^2$$

- Somme est calculée sur la BDD complète
- Bonne qualité mais temps de calcul important

Descente de gradient stochastique (SGD)

- A chaque étape de l'apprentissage, on veut minimiser l'erreur globale *J*

$$J(paramètres) = \frac{1}{2}\sum_{LOT}(S_{calculée} - S_{attendue})^2$$

- Somme est calculée sur un lot = échantillon aléatoire (mini-batch) de la BDD souvent de taille entre 10 et 1 000
- Très rapide à calculer
- On peut changer de lot régulièrement durant l'apprentissage

Learning rate adaptatif

On fait varier le PAS d'apprentissage

- PAS élevé au départ de l'apprentissage pour converger rapidement

- PAS faible vers la fin de l'apprentissage pour améliorer la précision

Exemple de pas adaptatif :

$$PAS = \frac{PAS_0}{t^{0,25}}$$

Plan

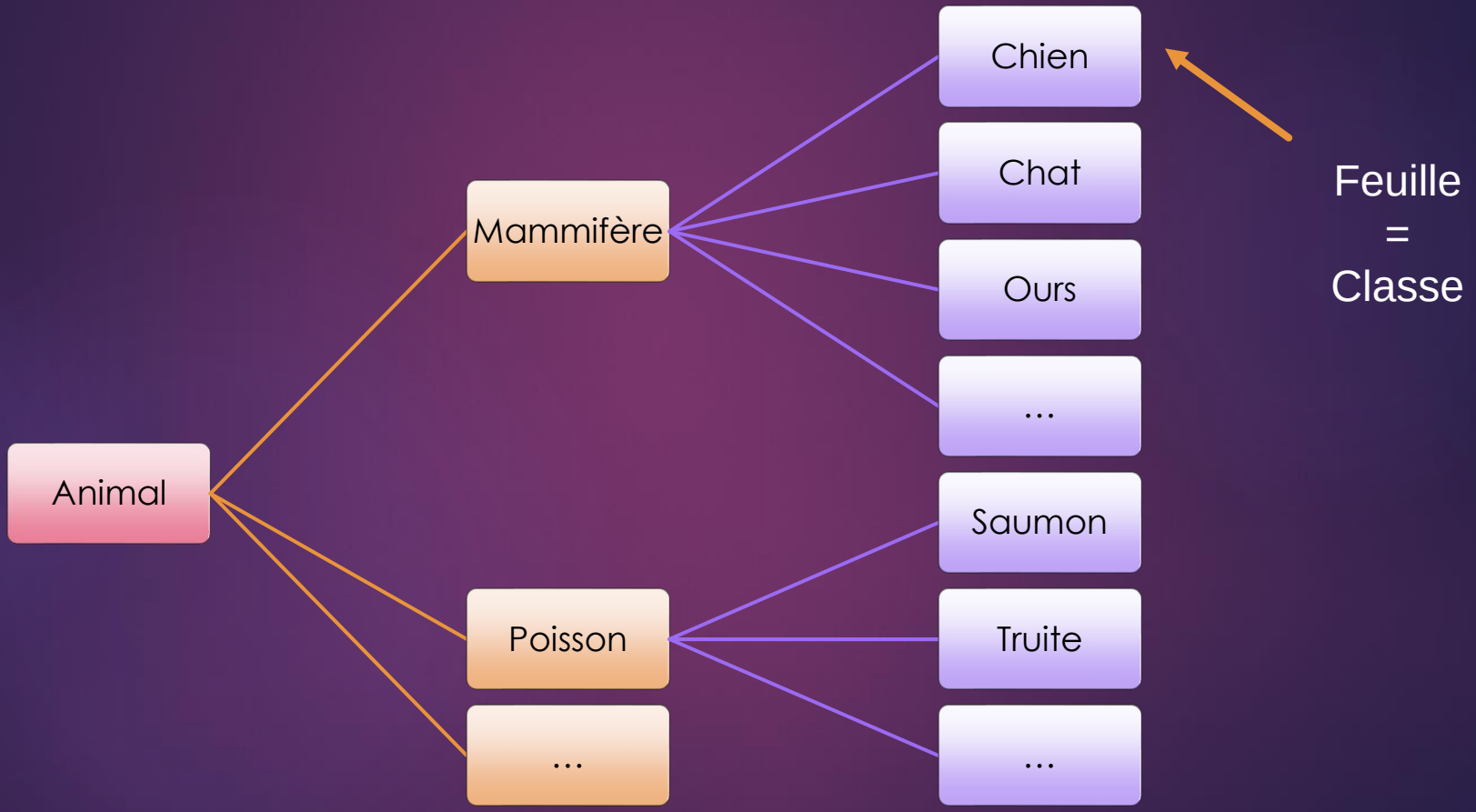
I. Régressions
II. Arbres de décision
III. Forêt aléatoire et apprentissage d'ensemble
IV. SVM (Support Vector Machine)

II. Arbres de décision

Ensemble de choix représentés symboliquement sous la forme d'un arbre, permettant de classer une population.

→ **Recherche de variables discriminantes**

Exemple d'arbre de décision



Avantages des arbres de décision

- Calculable automatiquement par apprentissage supervisé
- Lisible
- Exécutable rapidement (application à un nouvel individu pour le classer)
- Sortie = une ou plusieurs classes ou un nombre

Objectif visé

Objectifs :

- Obtenir les critères les plus discriminants
- Obtenir des classes les plus homogènes possible
- Obtenir l'erreur la plus faible possible

Visualisation d'un graphe

- Utilisation du module graphviz de Python

Utilisation module graphviz

- Installation du module

  pip install graphviz

(Vérification des dépendances dans Anaconda Navigator)

- Importation de la classe Digraph de graphviz pour les arbres directionnels

  from graphviz import Digraph

- Langage DOT pour décrire un arbre

Langage DOT

Création de l'objet arbre

arbre = Digraph(comment='Mon arbre')

Création d'un noeud

arbre.node('A', 'Femme')

nom du noeud

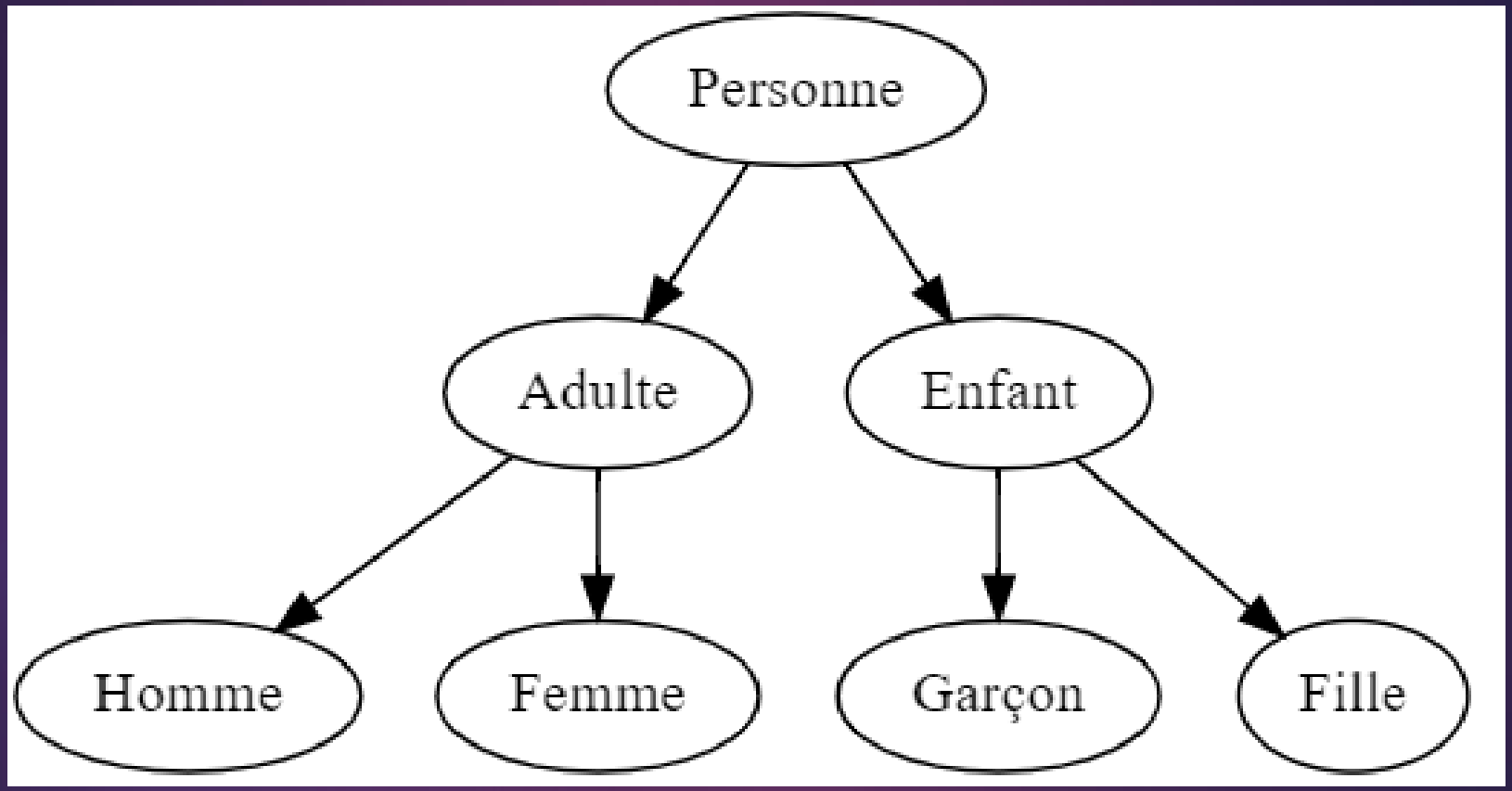
texte affiché sur le noeud

Création des branches

arbre.edges(['AB', 'AC',...]) ou arbre.edge('A','B')

Apparence → arbre.node_attr={"shape":"box"}

TP1 : Représenter l'arbre



Module scikit-learn

- Calcule variables discriminantes d'une BDD
  ⟶ une valeur d'une variable discrim. = nœud

- Crée un arbre décrit dans le langage DOT

Module sklearn


```
from sklearn import tree
X = [[0, 0], [1, 1]]
Y = ['H', 'F']
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)
clf.predict([[2., 2.]])
clf.predict_proba([[2., 2.]])
tree.plot_tree(clf.fit(X, Y))
```

→ BDD d'apprentissage

→ classe de chaque individu

→ démarre l'apprentissage

→ classe un nouvel individu

→ proba des classes

→ représente l'arbre

Critères d'évaluation d'arbres de décision

On suppose que l'arbre donne des classes C1, …, Cn avec des probabilités P1, …, Pn

Ci* = la classe la plus fréquente de probabilité Pi*

Mesure de l'hétérogénéité des classes :

Taux d'erreur = 1 – Pi* ➔ peu utilisé

Critère de Gini = $\sum_k P_k (1 - P_k)$ (utilisé dans méthode CART)
➔indicateur synthétique permettant de rendre compte du niveau d'inégalité pour une variable et sur une population donnée

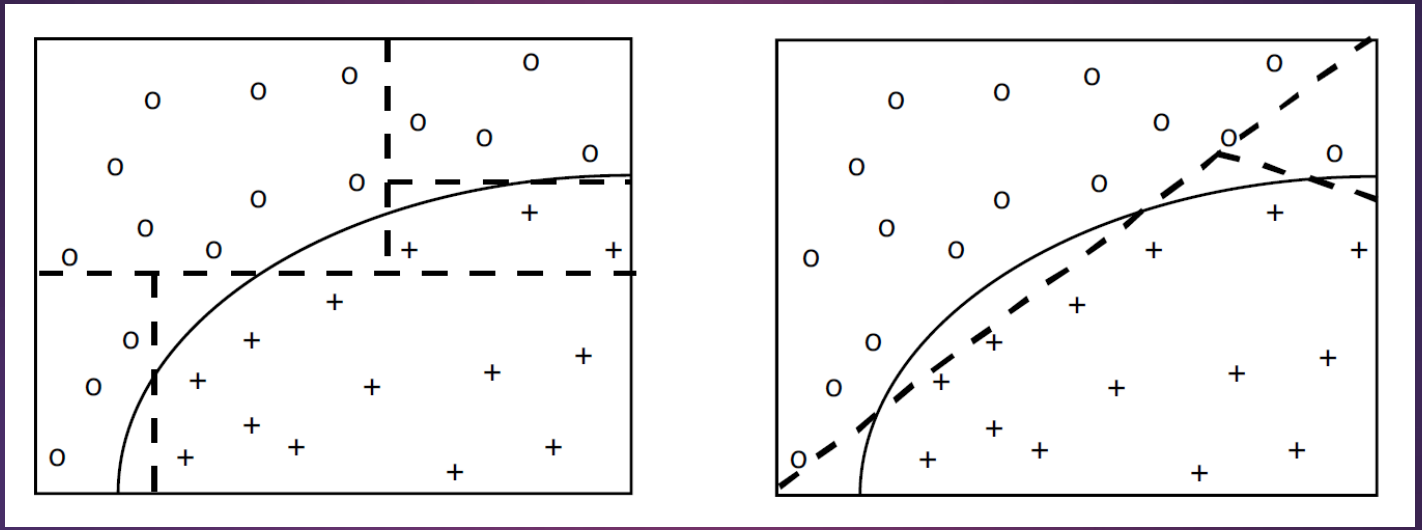
Entropie = $-\sum_k P_k \log(P_k)$ (utilisé dans méthode ID3)
➔quantité moyenne d'information nécessaire pour identifier la classe d'un exemple

Construction de l'arbre de décision

Une fois le critère de mesure choisi, il y a 2 phases

- Phase 1 : Construction = processus récursif de division (toutes les modalités pour variable qualitative, souvent binaire (seuil), notamment pour les variables quantitatives avec détermination d'un seuil)
  - Séparation parallèle aux axes (suivant 1 variable)
  - Séparation oblique (combinaison de variables)
- On minimise le mélange des classes dans les feuilles

Séparation parallèle vs oblique (données quantitatives)



- Moins de tests pour une séparation oblique mais plus complexes
- Problème d'échelle (ordre de grandeur) pour oblique ➔ normaliser
- Cross-validation (on divise la base d'apprentissage en K échantillons utilisés pour le calcul ou la validation)

- Phase 2 : Élagage (pruning) = moins de feuilles (nœuds terminaux) ➔ on garde les plus pertinentes
  - Pré-élagage : on arrête de subdiviser l'arbre quand un test est vérifié par une quantité acceptable d'individus
  - Post-élagage : on supprime un test si le taux d'erreur s'améliore

TP2 : Calcul d'arbres de décision

Pour chaque série, calculer l'arbre de décision et l'afficher

Série 1

- $(0.1, 0.5) \rightarrow 1$
- $(0.2, 0.9) \rightarrow 1$
- $(0.6, 0.5) \rightarrow 0$
- $(0.7, 0.9) \rightarrow 0$
- $(0.3, 0.7) \rightarrow 1$
- $(0.6, 0.5) \rightarrow$ valeur ?

Série 2

- $(0.4, 0.9) \rightarrow 0$
- $(0.0, 0.2) \rightarrow 1$
- $(0.3, 0.6) \rightarrow 0$
- $(0.1, 0.4) \rightarrow 1$
- $(0.2, 0.0) \rightarrow 1$
- $(0.2, 0.7) \rightarrow$ valeur ?

Sous Python avec Sklearn

- Par défaut, Scikit-Learn utilise l'algorithme CART (basé sur l'indice de Gini)

- Pour utiliser l'algorithme ID3 (basé sur l'indice d'entropie) : DecisionTreeClassifier(criterion="entropy")

TP3 : Arbre de décision pour prêt bancaire

- Utiliser la base de données de prêts bancaires pour générer 2 arbres de décision (méthodes CART par défaut ou ID3) puis comparer les taux d'erreur

Plan

I. Régressions
II. Arbres de décision
III. Forêt aléatoire et apprentissage d'ensemble
IV. SVM (Support Vector Machine)

Forêt aléatoire (random forest)

- Algorithme proposé en 1995 par Ho et détaillé en 2001 par Breiman et Cutler

Constat :

- les arbres de décision sont sensibles à l'ordre des prédicteurs (variables dans les tests)

➔ On calcule différents arbres basés uniquement sur une partie des variables (en général moins que $\sqrt{nb\_variables}$)

➔ Algorithme très efficace quand *nb_variables* est grand

Principe de bagging

Bagging = agrégation de modèles

- Les forêts aléatoires fournissent plusieurs arbres de décision ➔ plusieurs prédictions différentes pour chaque individu
- Nécessité de regrouper toutes ces prédictions en une seul ➔ agrégation (bagging)

Méthodes de bagging

Comment obtenir la prédiction finale pour un individu à partir de plusieurs arbres de décision ?

- Pour une classification : on choisit la catégorie la plus fréquente
- Pour une régression : on fait la moyenne des valeurs prédites