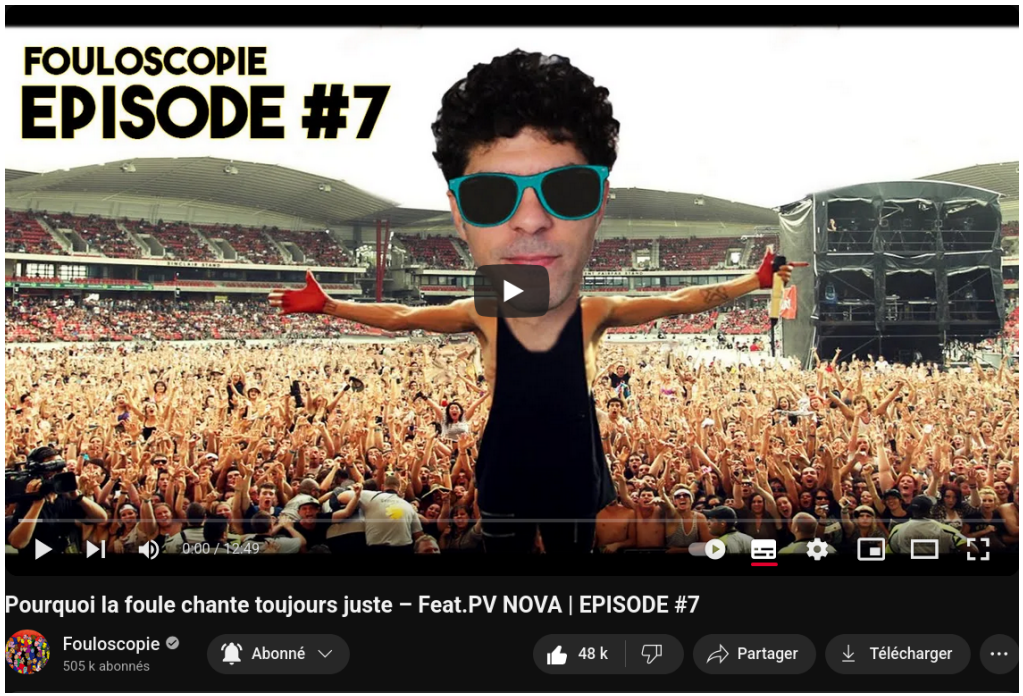
Parallélisme du Random Forest

Machine Learning

Béryl HOUESSOU

Institut de Mathématiques et de Sciences Physiques
Université d'Abomey-Calavi, Bénin

1er juillet 2025



Pourquoi lors des concerts, la foule chante-t-elle toujours juste ?

De la foule au Random Forest

Dans un concert :

- Chaque chanteur fait des erreurs

Dans Random Forest :

- Chaque arbre fait des erreurs

De la foule au Random Forest

Dans un concert :

- Chaque chanteur fait des erreurs
- Collectivement : voix juste

Dans Random Forest :

- Chaque arbre fait des erreurs
- Collectivement : prédiction plus ou moins précise

De la foule au Random Forest

Dans un concert :

- Chaque chanteur fait des erreurs
- Collectivement : voix juste
- Correction mutuelle

Dans Random Forest :

- Chaque arbre fait des erreurs
- Collectivement : prédiction plus ou moins précise
- Agrégation des résultats

De la foule au Random Forest

Dans un concert :

- Chaque chanteur fait des erreurs
- Collectivement : voix juste
- Correction mutuelle

Dans Random Forest :

- Chaque arbre fait des erreurs
- Collectivement : prédiction plus ou moins précise
- Agrégation des résultats

Intelligence collective = Modèles ensemblistes

Qu'est-ce que le Random Forest ?

Définition

Algorithme d'apprentissage automatique ensembliste qui :

- Construit plusieurs arbres sur des **échantillons aléatoires**

Qu'est-ce que le Random Forest ?

Définition

Algorithme d'apprentissage automatique ensembliste qui :

- Construit plusieurs arbres sur des **échantillons aléatoires**
- **Combine leurs résultats** pour une prédiction finale

Qu'est-ce que le Random Forest ?

Définition

Algorithme d'apprentissage automatique ensembliste qui :

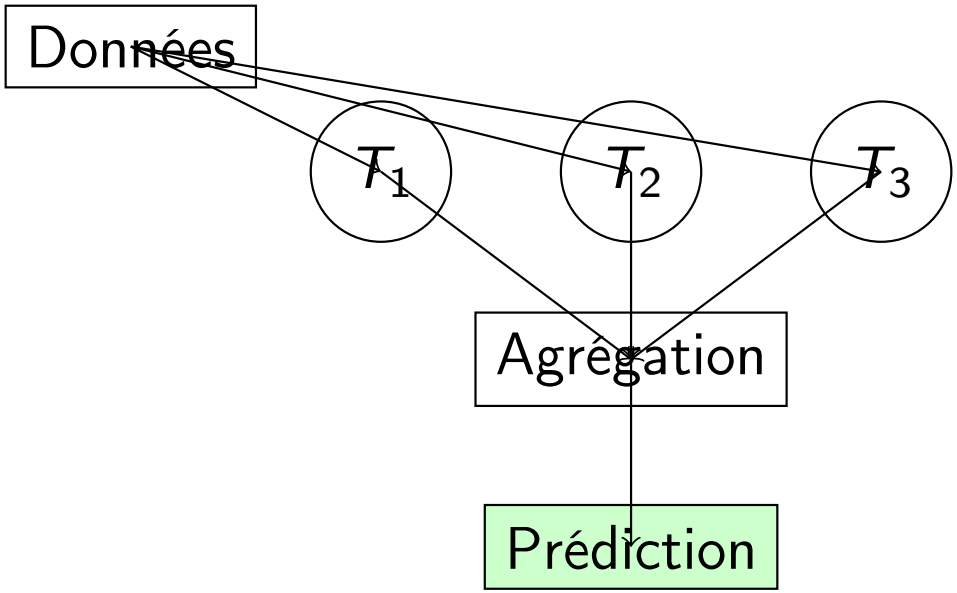
- Construit plusieurs arbres sur des **échantillons aléatoires**
- **Combine leurs résultats** pour une prédiction finale
- Réduit le surapprentissage et améliore la robustesse

Qu'est-ce que le Random Forest ?

Définition

Algorithme d'apprentissage automatique ensembliste qui :

- Construit plusieurs arbres sur des **échantillons aléatoires**
- **Combine leurs résultats** pour une prédiction finale
- Réduit le surapprentissage et améliore la robustesse



Algorithme Random Forest

Algorithm 1 Random Forest

Require: Ensemble d'entraînement D, nombre d'arbres n

Ensure: Forêt de décision F

1: Initialiser $F \leftarrow \emptyset$
2: **for** $i = 1$ à n **do**
3: $\quad D_i \leftarrow$ échantillon bootstrap de D
4: $\quad T_i \leftarrow$ construire un arbre de décision sur D_i
5: $\quad$ Ajouter T_i à F
6: **end for**
7: **return** F

Construction des arbres : Bagging

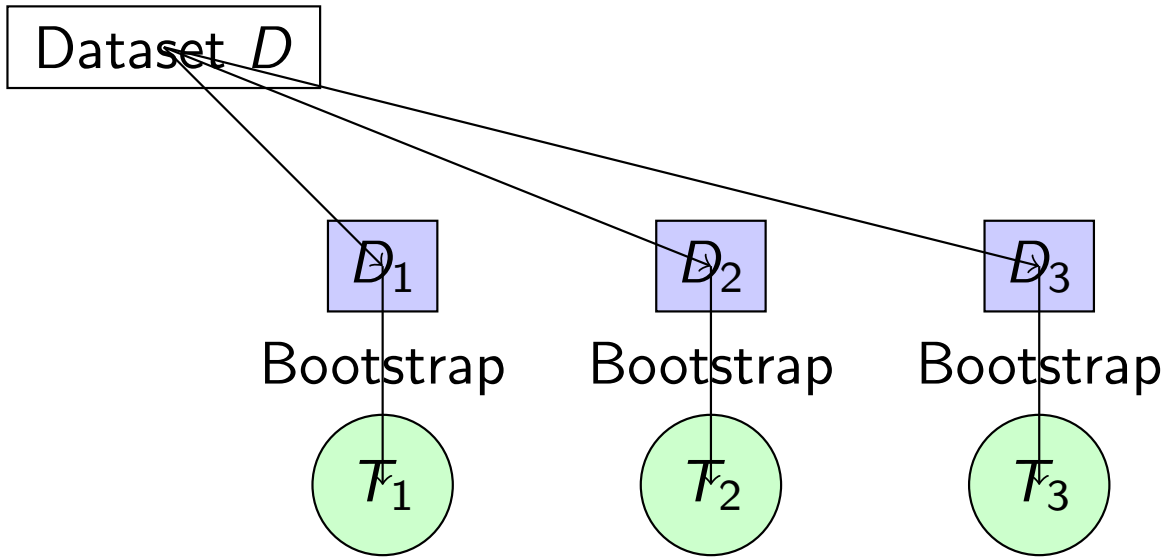
Bootstrap Aggregating

Principe : Créer plusieurs sous-ensembles par échantillonnage avec remise

Construction des arbres : Bagging

Bootstrap Aggregating

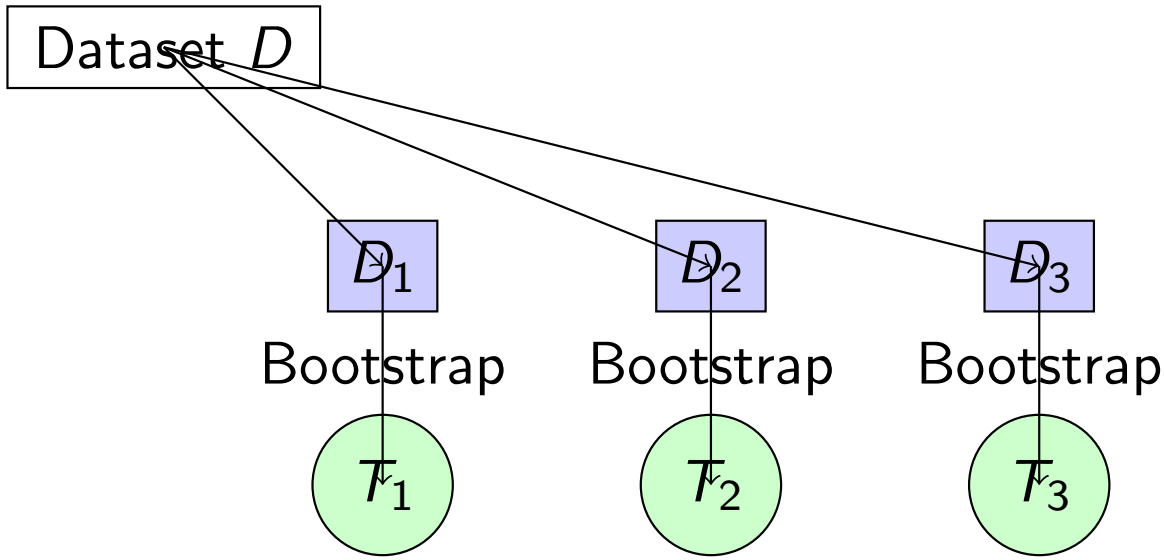
Principe : Créer plusieurs sous-ensembles par échantillonnage avec remise



Construction des arbres : Bagging

Bootstrap Aggregating

Principe : Créer plusieurs sous-ensembles par échantillonnage avec remise



Avantage : Rajouter de la diversité entre les arbres, ce qui réduit le surapprentissage et améliore la généralisation.

Mesures de pureté

Entropie

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1
- 0 = distribution homogène

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1
- 0 = distribution homogène
- 1 = distribution hétérogène

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1
- 0 = distribution homogène
- 1 = distribution hétérogène

Indice de Gini

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1
- 0 = distribution homogène
- 1 = distribution hétérogène

Indice de Gini

$$Gini(X) = 1 - \sum_{i=1}^{n} p_i^2$$

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1
- 0 = distribution homogène
- 1 = distribution hétérogène

Indice de Gini

$$Gini(X) = 1 - \sum_{i=1}^{n} p_i^2$$

- Varie entre 0 et 0,5

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1
- 0 = distribution homogène
- 1 = distribution hétérogène

Indice de Gini

$$Gini(X) = 1 - \sum_{i=1}^{n} p_i^2$$

- Varie entre 0 et 0,5
- 0 = pureté parfaite

Mesures de pureté

Entropie

$$H(X) = -\sum_{i=1}^{n} p_i \log_2(p_i)$$

- Varie entre 0 et 1
- 0 = distribution homogène
- 1 = distribution hétérogène

Indice de Gini

$$Gini(X) = 1 - \sum_{i=1}^{n} p_i^2$$

- Varie entre 0 et 0,5
- 0 = pureté parfaite
- 0,5 = désordre maximal

Construction d'un arbre

Algorithm 2 Construction d'un arbre de décision

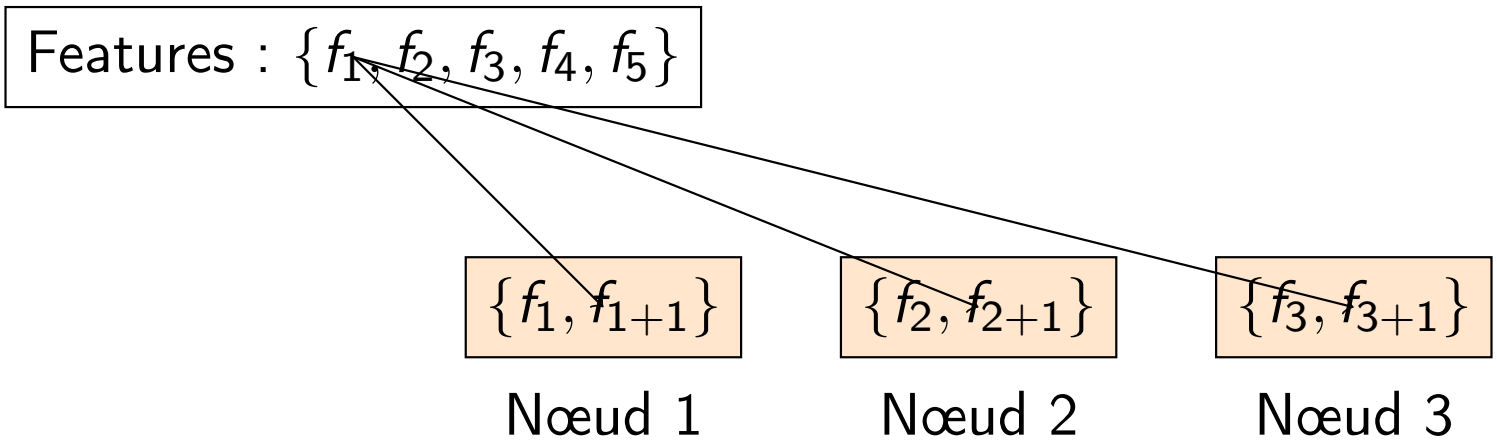
Require: Ensemble D, profondeur max d, échantillons min m
Ensure: Arbre de décision T

1: **if** profondeur max atteinte OU $|D| < m$ **then**
2: **return** feuille (classe majoritaire/moyenne)
3: **end if**
4: Sélectionner sous-ensemble aléatoire de caractéristiques
5: **for** chaque caractéristique **do**
6: Calculer gain d'impureté pour chaque seuil
7: **end for**
8: Choisir meilleure caractéristique et seuil
9: Créer D_{gauche} et D_{droite}
10: $T_{gauche} \leftarrow$ récursion sur D_{gauche}
11: $T_{droite} \leftarrow$ récursion sur D_{droite}
12: **return** T

Sélection aléatoire de caractéristiques : Feature bagging

Objectif

Réduire la corrélation entre arbres et améliorer la diversité



Avantages :

- Évite la domination par quelques caractéristiques
- Particulièrement utile avec beaucoup de features

Agrégation des résultats

Classification

- Vote majoritaire
- Classe la plus fréquente

T_1 : A

T_2 : B

T_3 : A

Résultat : A

Régression

- Moyenne des prédictions
- Réduction de la variance

T_1 : 5.2

T_2 : 4.8

T_3 : 5.1

Résultat : 5.03

Avantages et inconvénients du Random Forest

Avantages

- **Robustesse :** Réduit le surapprentissage
- **Précision :** Souvent supérieure aux arbres individuels
- **Versatilité :** Fonctionne sur différents types de données
- **Gestion du bruit :** Résistant aux données aberrantes
- **Pas de normalisation :** Invariant aux transformations monotones

Avantages et inconvénients du Random Forest

Avantages

- **Robustesse :** Réduit le surapprentissage
- **Précision :** Souvent supérieure aux arbres individuels
- **Versatilité :** Fonctionne sur différents types de données
- **Gestion du bruit :** Résistant aux données aberrantes
- **Pas de normalisation :** Invariant aux transformations monotones

Inconvénients

- **Coût computationnel :** Temps de calcul élevé
- **Mémoire :** Consommation importante
- **Interprétabilité :** Moins lisible qu'un arbre unique
- **Surapprentissage :** Possible avec trop d'arbres
- **Biais :** Peut favoriser les features avec plus de modalités

Avantages et inconvénients du Random Forest

Avantages

- **Robustesse :** Réduit le surapprentissage
- **Précision :** Souvent supérieure aux arbres individuels
- **Versatilité :** Fonctionne sur différents types de données
- **Gestion du bruit :** Résistant aux données aberrantes
- **Pas de normalisation :** Invariant aux transformations monotones

Inconvénients

- **Coût computationnel :** Temps de calcul élevé
- **Mémoire :** Consommation importante
- **Interprétabilité :** Moins lisible qu'un arbre unique
- **Surapprentissage :** Possible avec trop d'arbres
- **Biais :** Peut favoriser les features avec plus de modalités

Solution aux problèmes de performance : La parallélisation !

Parallélisation : Pourquoi ?

Limitations du Random Forest

- Coûteux en temps de calcul
- Consommation mémoire importante
- Lent avec beaucoup d'arbres

Avantages de la parallélisation

Parallélisation : Pourquoi ?

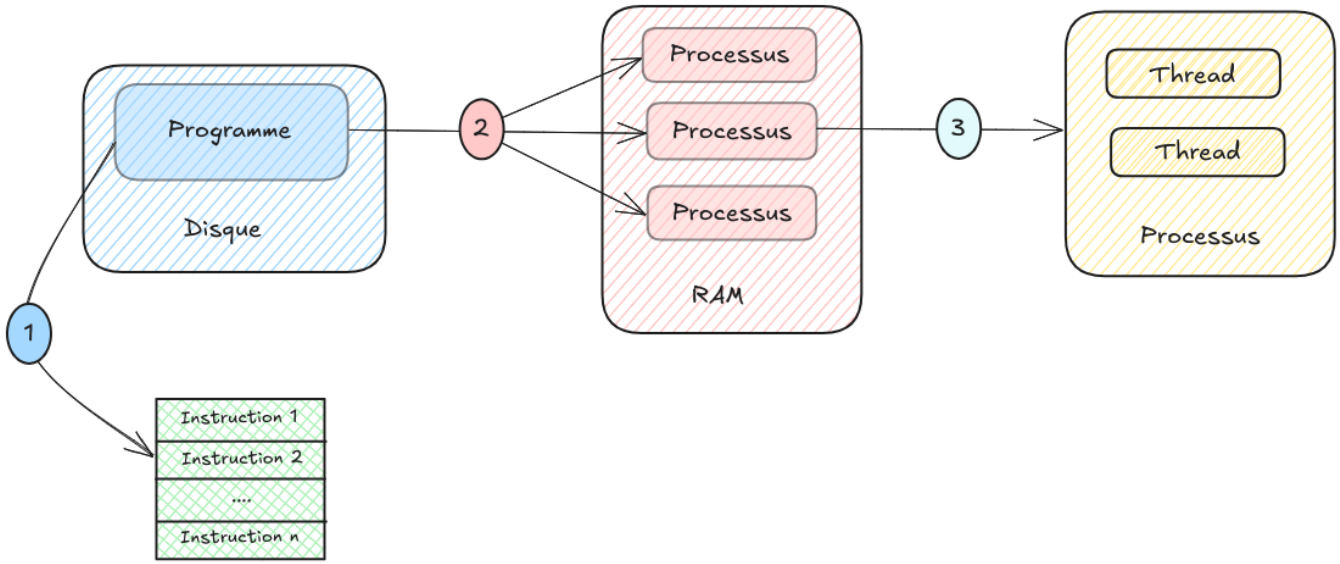
Limitations du Random Forest

- Coûteux en temps de calcul
- Consommation mémoire importante
- Lent avec beaucoup d'arbres

Avantages de la parallélisation

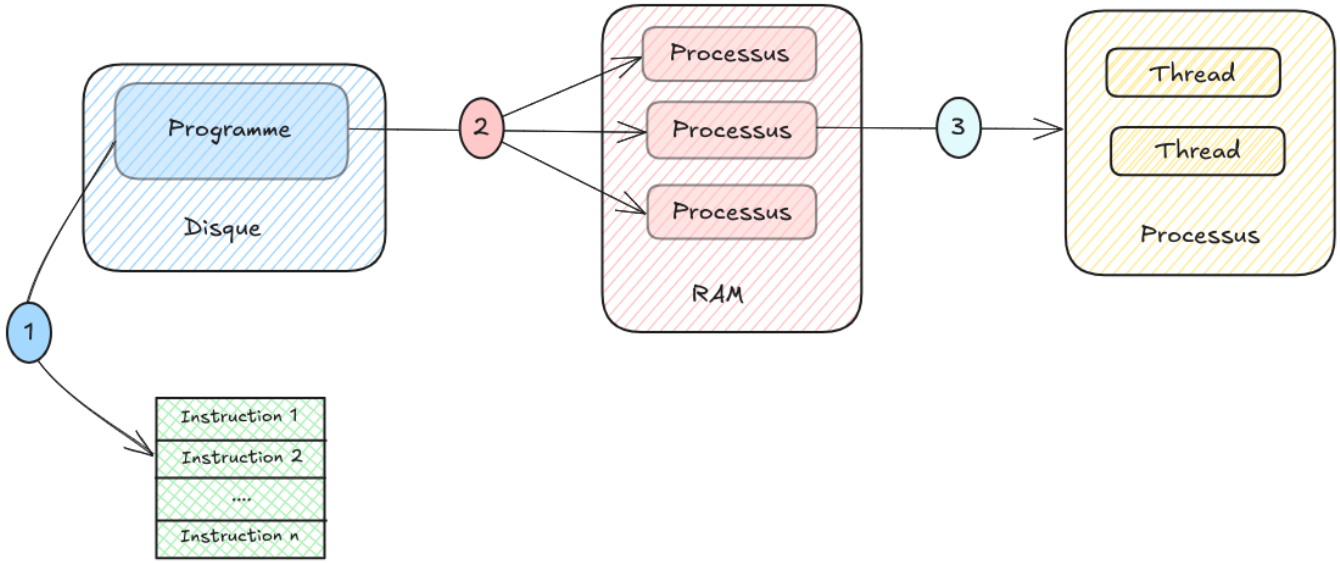
- **Efficacité :** Réduit le temps d'entraînement
- **Scalabilité :** Gère de gros volumes de données
- **Indépendance :** Chaque arbre peut être construit séparément

Thread vs Processus



Thread

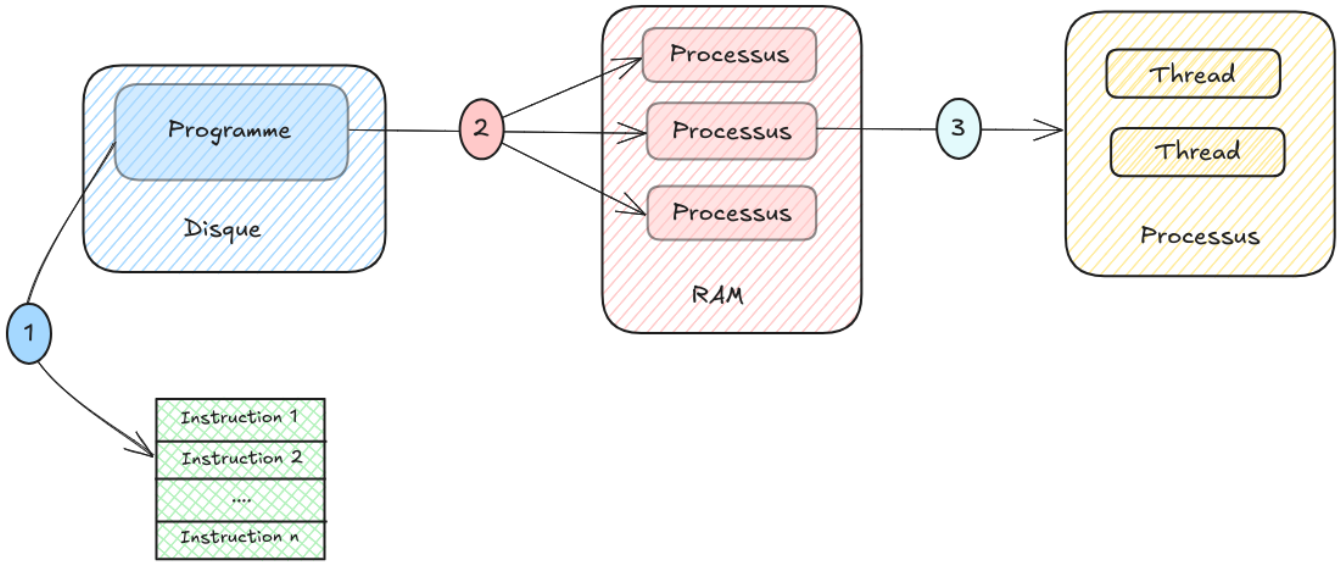
Thread vs Processus



Thread

- Unité d'exécution indépendante dans un processus

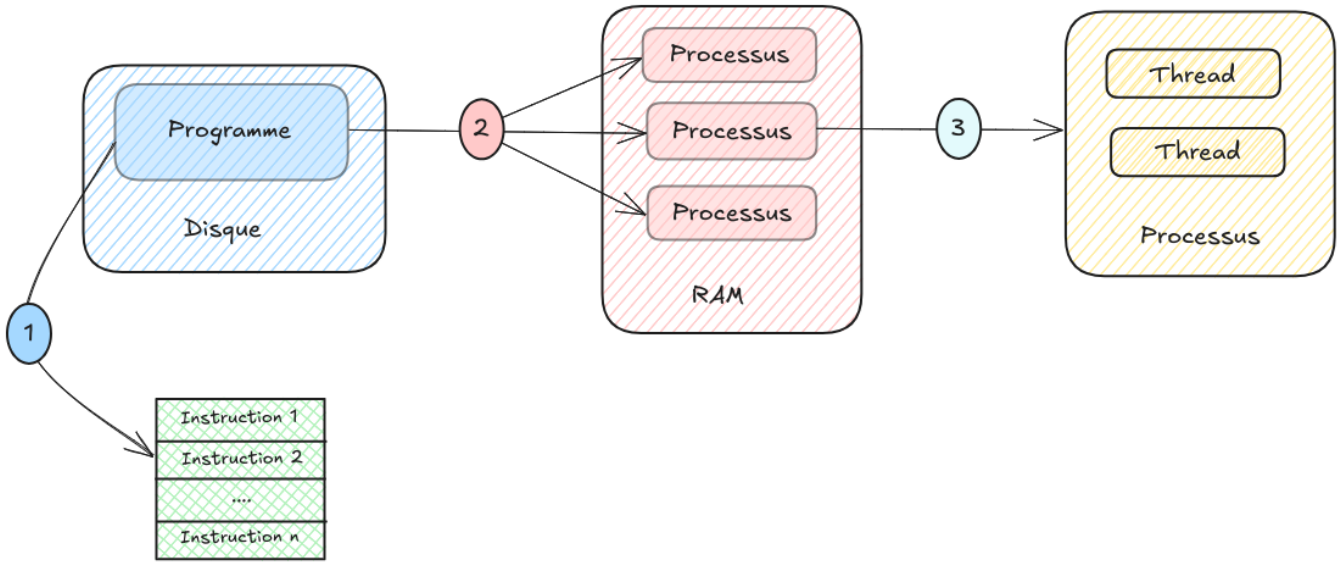
Thread vs Processus



Thread

- Unité d'exécution indépendante dans un processus
- Partage mémoire et ressources

Thread vs Processus



Thread

- Unité d'exécution indépendante dans un processus
- Partage mémoire et ressources
- Communication rapide mais synchronisation nécessaire

Parallélisation avec std : :thread

```cpp
#include <vector>
#include <thread>

void build_tree_for_index(int i, std::vector<Tree>& trees,
const std::vector<Data>& data_samples) {
  trees[i] = build_tree(data_samples[i]);
}

void build_forest_parallel(std::vector<Tree>& trees,
const std::vector<Data>& data_samples) {
  std::vector<std::thread> threads;
  int n_trees = trees.size();

  // Creation des threads
  for (int i = 0; i < n_trees; ++i) {
    threads.emplace_back(build_tree_for_index, i,
    std::ref(trees), std::cref(data_samples));
  }

  // Synchronisation
  for (auto& t : threads) t.join();
}
```

Étapes de la parallélisation

1. **Création des threads**
   - Un thread par arbre à construire
   - Fonction de construction + données bootstrap

Étapes de la parallélisation

1. **Création des threads**
   - Un thread par arbre à construire
   - Fonction de construction + données bootstrap
2. **Synchronisation**
   - Attendre tous les threads (`join()`)

Étapes de la parallélisation

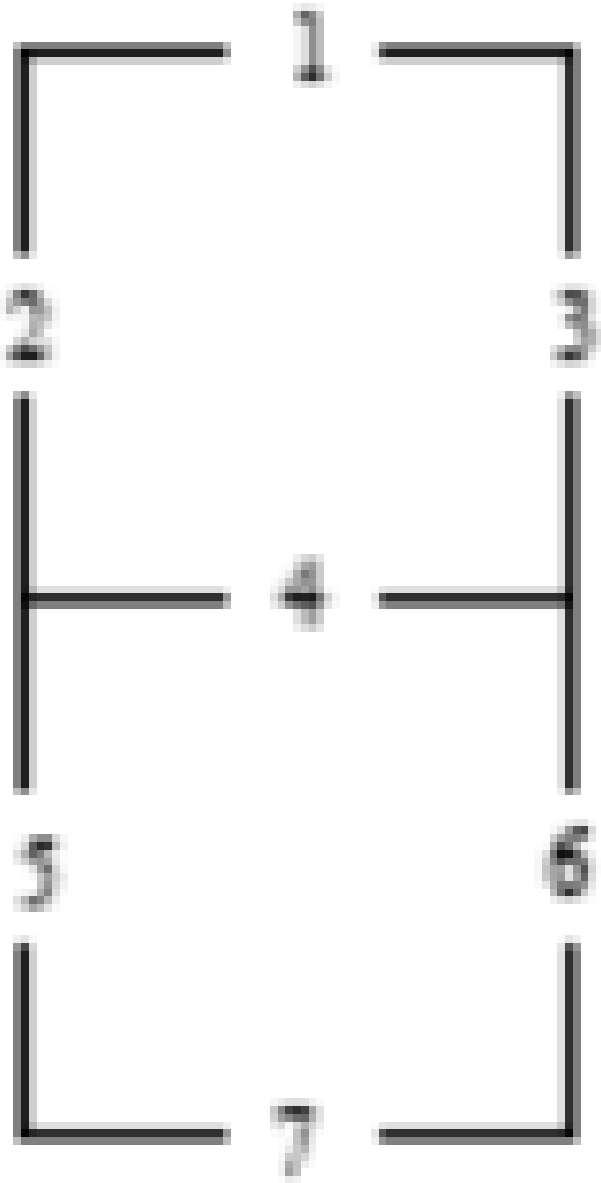
1. **Création des threads**
   - Un thread par arbre à construire
   - Fonction de construction + données bootstrap
2. **Synchronisation**
   - Attendre tous les threads (`join()`)
3. **Gestion des résultats**
   - Chaque thread écrit dans une zone mémoire distincte
   - Pas de partage de données modifiables

Exercice pratique : Reconnaissance des chiffres

Problème : Reconnaître les chiffres 0-9 à partir de 7 segments

Représentation :

- Vecteur binaire de 7 éléments
- 1 = segment allumé, 0 = éteint
- Exemple : "8" = [1,1,1,1,1,1,1]



Conclusion

Points clés

- **Random Forest** : Intelligence collective appliquée au ML
- **Robustesse** : Réduction du surapprentissage par agrégation
- **Parallélisation** : Accélération grâce à l'indépendance des arbres

Conclusion

Points clés

- **Random Forest :** Intelligence collective appliquée au ML
- **Robustesse :** Réduction du surapprentissage par agrégation
- **Parallélisation :** Accélération grâce à l'indépendance des arbres

Perspectives

- Autres algorithmes ensemblistes (XGBoost, LightGBM)

Merci, pour votre attention !