Projet de Master 2 : Implémentation de l'algorithme FCI (Fast Causal Inference) avec pyAgrum

Contexte et motivation

L'inférence causale est un domaine clé de l'intelligence artificielle et de la statistique moderne. Elle vise à identifier des **relations causales** à partir de données observationnelles, au-delà des simples corrélations.
L'algorithme **FCI (Fast Causal Inference)**, proposé par *Spirtes, Glymour et Scheines (1999)*, permet d'inférer des structures causales même en présence de **variables latentes** ou de **biais de sélection**.

Ce projet a pour objectif d'**implémenter et d'expérimenter** l'algorithme FCI à l'aide de la bibliothèque Python **pyAgrum**, dédiée aux modèles graphiques probabilistes. Vous comparerez ensuite les résultats obtenus avec d'autres algorithmes de découverte causale déjà implémentés dans pyAgrum comme MIIC.

Description du travail

1. Étude théorique et revue bibliographique

Avant le développement, le groupe devra :

- Étudier les bases de l'apprentissage de structure causale :
  - PC Algorithm (Peter-Clark)
  - FCI et RFCI
- Proposer une texte de synthèse sur FCI et ses variantes à partir de la littérature, par exemple :
  - Chapitre 6 de *Causation, Prediction, and Search* (2nd ed.). Spirtes, P., Glymour, C., & Scheines, R. (1999). MIT Press.
  - *On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. Artificial Intelligence*, Zhang, J. (2008). 172(16–17), 1873–1896.
  - *Learning high-dimensional directed acyclic graphs with latent and selection variables. Annals of Statistics*, Colombo, D., Maathuis, M. H., Kalisch, M., & Richardson, T. S. (2012). 40(1), 294–321.

2. Implémentation de FCI avec pyAgrum

Implémentez les principales étapes de l'algorithme :

1. **Découverte du squelette** : détection des arêtes via des tests d'indépendance conditionnelle.
2. **Identification des v-structures** : orientation des arcs basés sur les séparateurs trouvés.
3. **Application des règles d'orientation** : gestion des flèches et des cercles pour produire un **PAG** (Partial Ancestral Graph).

Utilisez les structures offertes par **pyAgrum** :

- `gum.BayesNet()` pour représenter les graphes.
- `gum.BNLearner` pour manipuler les données (calculer des χ^2, G^2, etc.).
- `gum.PDAG` pour représenter le graphe causal final.

3. Évaluation et comparaison

- Testez votre implémentation sur :
  - des données **simulées** (par exemple à partir d'un réseau bayésien généré par `pyAgrum.generateBN()`) ;
- Comparez les structures obtenues à celles produites par :
  - `miic` ou `GHC` dans pyAgrum ;
- Analysez les différences, par exemple en termes de :
  - structure (arêtes, orientations, présence d'ambiguïtés) ;
  - complexité computationnelle

Livrables attendus

1. **Code Python** documenté et exécutable à partir d'unscript `.py` . Un fichier notebook `.ipynb` est envisageable mais uniquement pour la présentation des résultats : les algorithmes doivent se trouvent dans des fichiers python. On pourra par exemple structure le code ainsi :
   - une classe ou un ensemble de fonctions réalisant FCI ;
   - un exemple d'exécution sur un jeu de données simulé et démontrant les avantages et critiques que vous voulez mettre en avant.
2. Le mini rapport de synthèse (quelques pages) sur l'algorithme
3. **Présentation orale** (10–15 min) avec démonstration.

Ressources utiles

- **Documentation pyAgrum**
  https://pyagrum.readthedocs.io

Indications pratiques

- Travail en binôme ou trinôme (si problème de parité).

- Soumission attendue avant la fin du semestre via la plateforme Moodle.:
  – code source sur GitHub/GitLab,
  – rapport PDF,
  – notebook ou slides de présentation comme support pour la présentation orale.