UNIVERSITE CHEIKH ANTA DIOP DE DAKAR

ECOLE SUPÉRIEURE POLYTECHNIQUE

DEPARTEMENT GENIE INFORMATIQUE

CLASSE : DIC1 - Informatique

MODULE : Introduction à l'intelligence Artificielle

Année Universitaire : 2024 - 2025

Compte Rendu TP Introduction à l'intelligence Artificielle

Étudiant :

Cheikh Ahmed Tidiane THIANDOUM

Introduction

Les algorithmes **GridSearch**, **RandomizedSearch**, et **Optuna** sont utilisés pour l'optimisation d'hyperparamètres, une étape clé en apprentissage automatique pour améliorer les performances d'un modèle. Voici une explication détaillée de chacun :

---

1. GridSearch

Principe :

- **GridSearch** explore systématiquement toutes les combinaisons possibles des hyperparamètres définis dans une grille. Chaque combinaison est évaluée pour trouver la meilleure configuration.

Fonctionnement :

1. **Définition d'une grille** : L'utilisateur spécifie des valeurs discrètes pour chaque hyperparamètre.

   Exemple :
   ```
   param_grid = {
      'learning_rate': [0.01, 0.1, 1],
      'n_estimators': [50, 100, 200]
   }
   ```

2. **Itération exhaustive** : GridSearch teste toutes les combinaisons possibles. Pour l'exemple ci-dessus, il y aurait $3 \times 3 = 9$ combinaisons.
3. **Évaluation** : Chaque modèle est entraîné et évalué avec une métrique choisie, par exemple, la précision ou l'erreur quadratique moyenne.
4. **Choix de la meilleure combinaison** : La configuration qui optimise la métrique est retenue.

Avantages :

- Simple à comprendre et à implémenter.
- Garantit d'explorer toutes les combinaisons possibles.

Inconvénients :

- **Coût computationnel élevé** : Peu efficace lorsque la grille est grande ou que le modèle est lent à entraîner.
- Ne tient pas compte des dépendances ou de la corrélation entre les hyperparamètres.

2. RandomizedSearch

Principe :

- Contrairement à GridSearch, **RandomizedSearch** explore un sous-ensemble aléatoire des combinaisons possibles. Cela permet une exploration plus rapide tout en gardant une bonne probabilité de trouver une configuration proche de l'optimum.

Fonctionnement :

1. **Définition d'une distribution** :
    - Les hyperparamètres sont spécifiés sous forme de distributions (ou listes d'échantillons).

    Exemple :
    ```
    param_distributions = {
      'learning_rate': [0.01, 0.1, 1],
      'n_estimators': [50, 100, 200]
    }
    ```

    -
2. **Échantillonnage aléatoire** :
    - RandomizedSearch choisit un nombre limité de combinaisons (par exemple, 10 combinaisons aléatoires).
3. **Évaluation** :
    - Chaque combinaison est évaluée sur le modèle.
4. **Choix de la meilleure combinaison** :

    - Comme avec GridSearch, la configuration qui optimise la métrique est retenue.

Avantages :

- Plus rapide que GridSearch, surtout pour de larges espaces d'hyperparamètres.
- Évite de tester des combinaisons redondantes ou peu probables.

Inconvénients :

- Ne garantit pas de trouver l'optimum global, car il ne teste qu'un sous-ensemble aléatoire.

3. Optuna

Principe :

- **Optuna** est une bibliothèque moderne pour l'optimisation d'hyperparamètres. Elle utilise des techniques avancées comme les algorithmes bayésiens pour **diriger la recherche** vers les zones prometteuses de l'espace d'hyperparamètres.

Fonctionnement :

1. **Définition de l'espace de recherche** :
    - L'utilisateur définit les hyperparamètres avec des plages ou des distributions probables.

    Exemple :
    ```
    import optuna

    def objective(trial):
        learning_rate = trial.suggest_loguniform('learning_rate', 1e-4, 1e-1)
        n_estimators = trial.suggest_int('n_estimators', 50, 500)
        …
    ```

2. **Optimisation adaptative** :

    - Contrairement aux méthodes exhaustives ou aléatoires, Optuna **apprend** des évaluations précédentes pour guider les essais futurs.
    - Utilise des techniques comme **Tree-structured Parzen Estimator (TPE)** pour prédire les zones intéressantes.
3. **Évaluation itérative** :
    - À chaque itération, un ensemble de valeurs d'hyperparamètres est testé, et les résultats sont utilisés pour ajuster la recherche.
4. **Arrêt intelligent** :
    - Optuna peut arrêter les configurations qui semblent peu prometteuses avant qu'elles soient entièrement évaluées (pruning).

Avantages :

- Efficace pour les espaces d'hyperparamètres complexes et continus.
- Réduit le nombre d'essais nécessaires pour trouver un bon ensemble d'hyperparamètres.
- Intègre des outils de **pruning** pour arrêter les configurations sous-optimales.

Inconvénients :

- Plus complexe à mettre en œuvre.
- Peut être sensible à une mauvaise définition de l'espace de recherche.

Comparaison

Critères	GridSearch	RandomizedSearch	Optuna
Exploration	Exhaustive	Aléatoire	Dirigée (Bayésienne)
Efficacité	Lente pour de grands espaces	Rapide	Très rapide (adaptatif)
Facilité d'usage	Simple	Simple	Plus complexe
Garanties	Trouve l'optimum global	Peut manquer l'optimum	Dirige la recherche, mais pas garanti
Support pruning	Non	Non	Oui

Quand utiliser quoi ?

- **GridSearch** : Lorsque l'espace d'hyperparamètres est petit et que le temps de calcul n'est pas un problème.
- **RandomizedSearch** : Lorsque le temps est limité ou pour une recherche rapide de bonnes configurations.
- **Optuna** : Pour des projets complexes nécessitant une optimisation avancée et efficace.