UPEC
Connaissance - Action

UNIVERSITÉ
PARIS-EST CRÉTEIL
VAL DE MARNE

TP3: ATDN2

Optimisation Bayésienne et Modèles Bayésiens à Noyau

GAKOU YOUSSOUF

31/03/2025

Partie 1 : Optimisation Bayésienne

1.Expliquez le principe de l'optimisation bayésienne. Décrivez comment elle permet de gérer les fonctions coûteuses à évaluer.

L'optimisation bayesienne cest une méthode pour trouver le maximum ou le minimum d'une fonction en limitant les essais aleatoires ca permet de predire les résultats et leur incertitude elle dirige les tests vers les zones les plus intéressantes surtout quand les evaluations sont coûteuses ou prennent du temps.

2.Définissez et expliquez les processus gaussiens. Pourquoi sont-ils utilisés pour modéliser la fonction objective ?

Un processus gaussien estime tout les point de la fonction en utilisant une loi normale ca donne ensuite une moyenne et une mesure d'incertitude ici dans loptimisation bayesienne cela aide a decider ensuite en trouvant un equilibre entre lexploration de nouvelles zones et lutilisation des informations déjà connues

3.Décrivez les principales fonctions d'acquisition (Expected Improvement, Upper Confidence Bound, etc.). Expliquez leur rôle dans le compromis exploration/exploitation.

Une fonction d'acquisition indique le prochain point a tester il ya deux principales methodes

Expected Improvement: Favorise les points qui pourraient améliorer le meilleur score actuel.
Upper Confidence Bound: ca equilibre la moyenne prédite et l'incertitude pour explorer des zones moins connues.

Lobjectif des deux méthodes cest de mélanger l'exploration de nouvelles zones et l'amélioration des résultats qui sont deja bons

Partie 1 : Optimisation Bayésienne

1.Expliquez le principe de l'optimisation bayésienne. Décrivez comment elle permet de gérer les fonctions coûteuses à évaluer.

```python
# Je charge les données
data = pd.read_csv('tp2_atdn_donnees.csv')
data.rename(columns={
    'Humidité (%)': 'Humidite',
    'Température (°C)': 'Temperature',
    'Rendement agricole (t/ha)': 'Rendement',
    'Type de sol': 'Type_sol'
}, inplace=True)

# Je prends que ce qu'il faut
X = data[['Humidite', 'Temperature']]
y = data['Rendement']

# J'entraîne un modèle pour simuler
gp = GaussianProcessRegressor(kernel=RBF()).fit(X, y)
```

```python
# Fonction objectif
def objectif(params):
    humidite, temperature = params
    return -gp.predict([[humidite, temperature]])[0]

space = [
    Real(X['Humidite'].min(), X['Humidite'].max()),
    Real(X['Temperature'].min(), X['Temperature'].max())
]

res = gp_minimize(objectif, space, n_calls=30, random_state=0)

print("Meilleure humidité:", res.x[0])
print("Meilleure température:", res.x[1])
print("Rendement prédit:", -res.fun)
```

Dans ce code j'ai choisi gp_minimize pour itérer et trouver les meilleurs paramètres humidité, température j'utilise plot_convergence pour voir comment la fonction objectif évolue comme on veut maximiser le rendement je mets un signe négatif pour qu'il minimise l'algo essaye plusieurs valeurs et se concentre sur celles qui semblent les plus bons

```
Meilleure humidité: 68.73206959026872
Meilleure température: 19.57221011097964
Rendement prédit: 9.50818183287833
```

Dans les résultats on obtient une humidité optimale de 68.73 et une température optimale de 19.57 et un rendement prédit de 9.51

ca veut dire que pour avoir le meilleur rendement environ 9,51 tonnes par hectare il faut maintenir l'humidité entre 68 et 69 % et la température autour de 20degres selon les données disponibles ces conditions sont les plus favorables

Partie 1 : Optimisation Bayésienne

5.Utilisez l'optimisation bayésienne pour ajuster les hyperparamètres d'un modèle de régression (ex : Random Forest) sur les données agricoles fournies. Comparez les résultats avec Grid Search et Random Search.

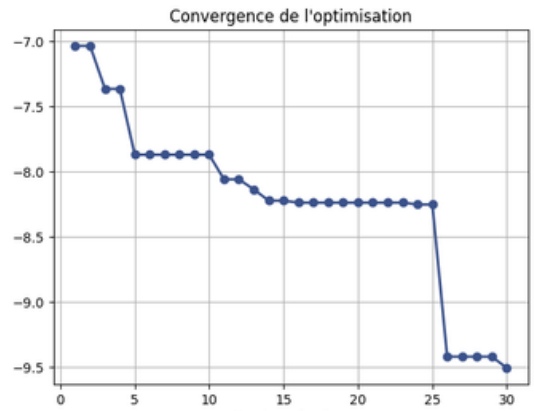
```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

param_grid = {'n_estimators': [50, 100, 150], 'max_depth': [5, 10, 15]}
gs = GridSearchCV(RandomForestRegressor(random_state=0), param_grid, cv=5)
gs.fit(X, y)

param_dist = {'n_estimators': np.arange(50, 200), 'max_depth': np.arange(3, 16)}
rs = RandomizedSearchCV(RandomForestRegressor(random_state=0), param_dist, n_iter=20, cv=5, random_state=0)
rs.fit(X, y)

print("Grid Search:", gs.best_params_)
print("Random Search:", rs.best_params_)
print("Bayésienne:", {'n_estimators': res_rf.x[0], 'max_depth': res_rf.x[1]})
```

ici gp_minimize permet de trouver de bons hyperparamètres sans faire trop d'essais GridSearchCV essaye chaque combinaison et RandomizedSearchCV en essaye au hasard je compare ensuite les trois résultats:

```
Grid Search: {'max_depth': 5, 'n_estimators': 150}
Random Search: {'n_estimators': np.int64(87), 'max_depth': np.int64(4)}
Bayésienne: {'n_estimators': np.int64(50), 'max_depth': np.int64(4)}
```

le bayésien a trouvé qu'un petit nombre d'arbres 50 avec une profondeur de 4 le grid search a trouvé une forêt plus grande 150 arbres avec une profondeur de 5 en testant toutes les combinaisons.
le random search a trouvé un compromis 87 arbres profondeur 4 en testant des valeurs au hasard
ici le bayésien est plutôt efficace un modele simple est suffisant pour les données

6.Visualisez le processus d'optimisation (courbe de convergence, choix des points). Commentez la manière dont le modèle explore l'espace de recherche.




```
from skopt.plots import plot_convergence
plot_convergence(res)
plt.title("Convergence de l'optimisation")
plt.show()
```

On voit que l'erreur descend petit à petit parcque lalgorithme fait des essais aleatoires puis affine ses choix Il explore l'espace de recherche en essayant plusieurs valeurs puis se focalise sur celles qui semblent bon

Partie 1 : Optimisation Bayésienne

7.Analysez les avantages et limites de l'optimisation bayésienne face aux méthodes classiques.

l'optimisation bayésienne est bien pour tester moins de combinaisons cest a dire que elle est rapide pour les modèles longs à entrainer mais elle peut être lente à démarrer et plus compliquée à mettre en place que grid search elle fonctionne mieux quand ya pas beaucoup dessais a faire

Partie 2 : Modèles Bayésiens à Noyau

8. Expliquez le concept d'inférence bayésienne. Comment met-on à jour les croyances avec de nouvelles données ?

L'inférence bayésienne commence avec une croyance initiale et l'ajuste avec de nouvelles données et plus il ya de la de données mieux le modèle s'adapte

9. Décrivez la théorie des méthodes à noyau et leur lien avec les processus gaussiens. Pourquoi utiliser un noyau dans un modèle bayésien ?

Les méthodes a noyau sert a mesurent la similarité entre les points sans calculer tout dans l'espace d'origine dans le processus gaussienn le noyau indique comment deux points sont lies cest utile pour modéliser des relations complexes et estimer une fonction avec peu de donné

10. Qu'est-ce qu'une distribution a priori et une distribution a posteriori ? Donnez un exemple appliqué à la prédiction de rendement agricole.

Une distribution a priori c'est ce qu'on croit avant d'avoir vu les données et a posteriori c'est après par exemple on pense que le rendement est autour de 6t/ha la cest a priori puis on voit les vraies données, et on change notre davis la cest a posteriori.

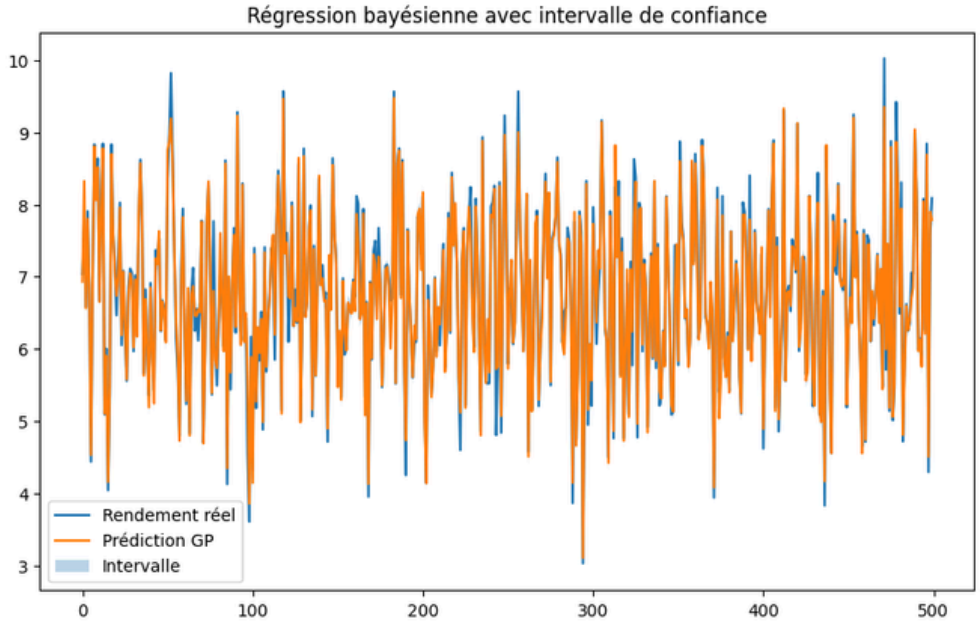
Partie 2 : Modèles Bayésiens à Noyau

11. Implémentez une régression bayésienne à noyau sur les données agricoles fournies. Visualisez les prédictions et les intervalles de confiance.

```python
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF

kernel = RBF()
model = GaussianProcessRegressor(kernel=kernel, alpha=1e-2)
model.fit(X, y)

y_pred, y_std = model.predict(X, return_std=True)

plt.figure(figsize=(10, 6))
plt.plot(y.values, label="Rendement réel")
plt.plot(y_pred, label="Prédiction GP")
plt.fill_between(np.arange(len(y_pred)), y_pred - y_std, y_pred + y_std, alpha=0.3, label="Intervall
plt.legend()
plt.title("Régression bayésienne avec intervalle de confiance")
plt.show()
```

J'ai choisi GaussianProcessRegressor avec un noyau RBF car il s'adapte bien à des formes variées. Le paramètre alpha sert à gérer le bruit. J'affiche la prédiction et l'intervalle de confiance pour voir où le modèle est plus ou moins sûr

on voit que la courbe de prédiction suit assez bien les valeurs réelles. Les zones avec moins de données ou plus de variabilité ont un intervalle plus large.



12. Réalisez une classification bayésienne à noyau pour prédire le type de sol (argileux, sableux, limoneux) en fonction des données climatiques. Comparez les résultats avec un SVM classique.

Je fais un GaussianProcessClassifier et un SVM et ensuite je compare leurs scores

```python
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report

X_clf = data[['Humidite', 'Temperature']]
y_clf = data['Type_sol']

# GP Classifier
gp_clf = GaussianProcessClassifier().fit(X_clf, y_clf)
y_pred_gp = gp_clf.predict(X_clf)

# SVM Classifier
svm = SVC().fit(X_clf, y_clf)
y_pred_svm = svm.predict(X_clf)

print("Résultats GP :\n", classification_report(y_clf, y_pred_gp))
print("Résultats SVM :\n", classification_report(y_clf, y_pred_svm))
```

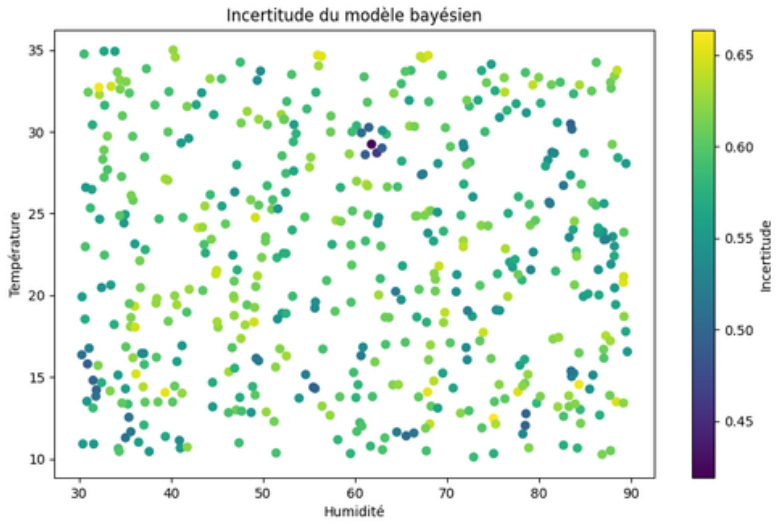
Partie 2 : Modèles Bayésiens à Noyau

```
Résultats GP :
              precision    recall  f1-score   support

    Argileux       0.84      0.85      0.84       177
    Limoneux       0.88      0.87      0.87       165
     Sableux       0.86      0.87      0.86       158

    accuracy                           0.86       500
   macro avg       0.86      0.86      0.86       500
weighted avg       0.86      0.86      0.86       500

Résultats SVM :
              precision    recall  f1-score   support

    Argileux       0.35      1.00      0.52       177
    Limoneux       0.00      0.00      0.00       165
     Sableux       0.00      0.00      0.00       158

    accuracy                           0.35       500
   macro avg       0.12      0.33      0.17       500
weighted avg       0.13      0.35      0.19       500
```

La classification bayésienne atteint une précision de 0,86 et distingue bien les types de sol (Argileux Limoneux Sableux) avec des scores autour de 0,84–0,88 il sadapt bien aux donnes climatiques Le SVM avec ses reglages par défaut a une précision de 0,35 et echoue a différencier Limoneux et Sableux classant presque tout en Argileux ce qui crée un déséquilibre donc la classification bayésienne est plus efficace sans réglages supplémentaires

13. Analysez l'incertitude dans les prédictions. Commentez les zones où le modèle est moins confiant.

```python
proba = gp_clf.predict_proba(X_clf)
incertitude = 1 - np.max(proba, axis=1)

plt.figure(figsize=(10, 6))
plt.scatter(X_clf['Humidite'], X_clf['Temperature'], c=incertitude, cmap='viridis')
plt.colorbar(label='Incertitude')
plt.xlabel('Humidité')
plt.ylabel('Température')
plt.title('Incertitude du modèle bayésien')
plt.show()
```



Les points en jaune sont ceux où le modèle est le plus incertain alors que les points en bleu indiquent plus de certitude

Visuellement on constate que certaines zones de l'espace par exemple autour de 60 % d'humidité / 30 °C) présentent des valeurs plus extrêmes