La dernière réunion...

Projet M1 – SciML

Semaine 4

Rapide

Synthétique

Par Williams & Robin

Ce qu'on a fait la semaine dernière
Compendium & pré-traitement

Compteur de commits : 91

$$\dot{a}_k(t) = \begin{pmatrix} \dot{a}_1(t) \\ \dot{a}_2(t) \\ \vdots \\ \dot{a}_1(t) \end{pmatrix} = f_\theta(a_1(t), a_2(t), \ldots, a_k(t))$$

5 notebooks :

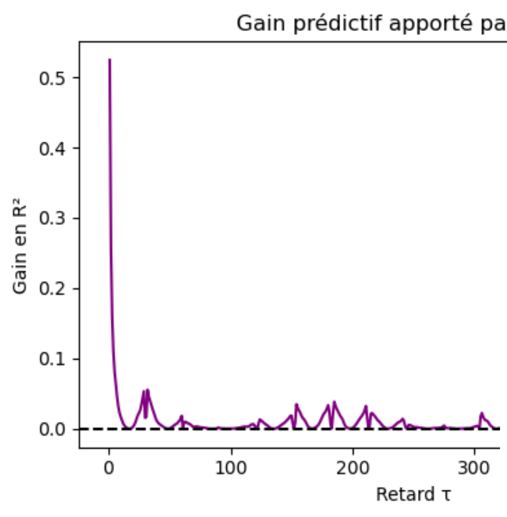
14 - Construction de l'état réduit

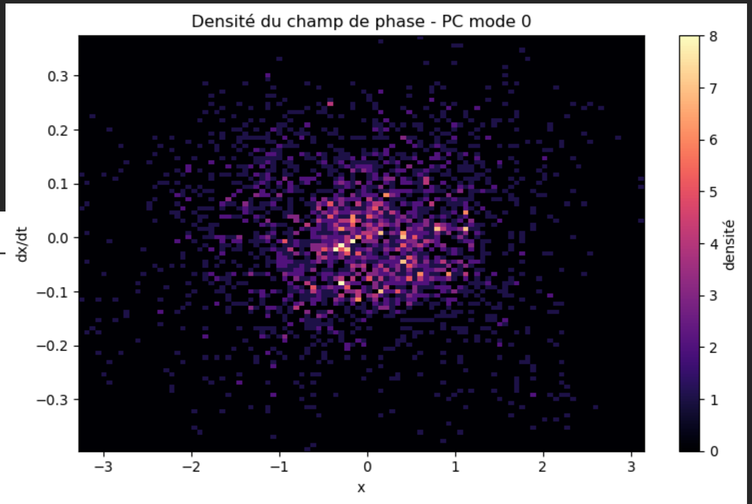
15 - Pré-traitement numérique

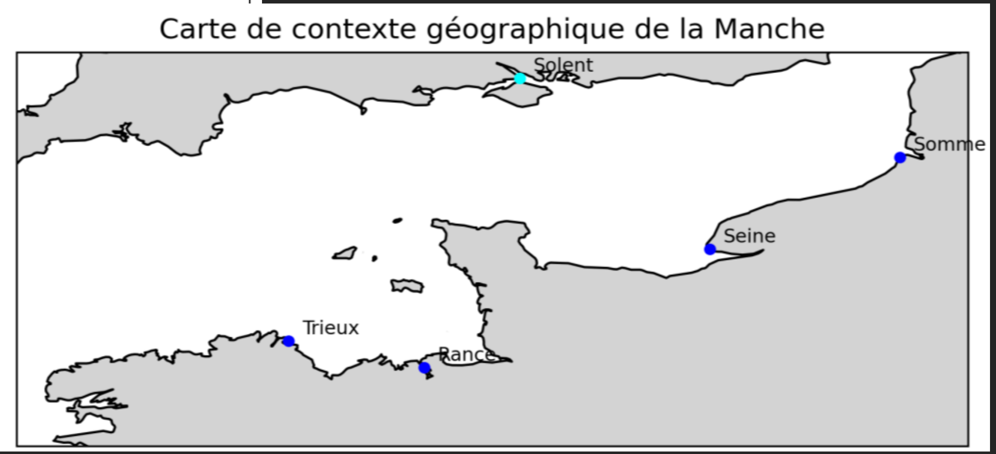
16 - Diagnostique dynamique

17 - Markovianité

BONUS – Tracé d'une carte de référence







Avancé du COMPENDIUM

Ce qu'on a fait en 5 jours ...

Fin pré-traitement & ML - SciML

Compteur de commits : 106

4 notebooks :

10 - Analyse multivariée (modification)
14 Bis - Réduction d'état par sélection
17 - Mémoire markovienne
18 - Modélisation ML classique

Qu'est-ce que la régularisation et pourquoi est-elle nécessaire ?

Dans les systèmes physiques à haute dimension, on observe souvent que le nombre de variables N est grand, le nombre d'observations T est limité et que les variables sont fortement corrélées. Ce contexte pose un problème de régression classique ($y = X\beta + \varepsilon$) qui mène le plus souvent à des approches erronées (solutions instables, sur-apprentissage et/ou faible capacité à généraliser). Lesquelles, dans notre cas, sont :

- un cas critique avec $N \geq T$, car la solution $\hat{\beta}$ n'existe pas ou instable
- une multicolinéarité structurelle majeure et problématique, car le modèle n'arrive pas à "choisir" entre plusieurs variables redondantes (points spatiaux voisins fortement corrélés)

La régularisation est une méthode solutionnant cette problématique. Celle-ci consiste à rajouter une contrainte (ou pénalité) supplémentaire à la régression classique dans le but de stabiliser la solution, comme suit :

$$\min_{\beta} ||y - X\beta||_2^2 + \lambda \mathcal{R}(\beta)$$

Il existe 3 principaux types de régressions classiques :

- Régularisation L2 dite *Ridge* :

$$\mathcal{R}(\beta) = ||\beta||_2^2$$

→ réduit l'amplitude des coefficients et stabilise la solution (utile pour la prédiction).

- Régularisation L1 dite *LASSO* :

$$\mathcal{R}(\beta) = ||\beta||_1 = \sum_j \beta_j$$

→ pousse certains coefficients à zéro et induit donc une sélection de variables (transformant le problème de régression en problème de sélection, notre cas).

- Régularisation L1 + L2 dite *Elastic Net* :

$$\mathcal{R}(\beta) = \alpha||\beta||_1 + (1-\alpha)||\beta||_2^2$$

→ compromission entre stabilité et parcimonie.

```python
from sklearn.linear_model import Ridge
from sklearn.multioutput import MultiOutputRegressor
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import TimeSeriesSplit, GridSearchCV
from sklearn.metrics import mean_squared_error
import numpy as np

# Modèle de base
ridge_base = MultiOutputRegressor(Ridge())

# Pipeline de prétraitement et de modèle
pipe_ridge = Pipeline([
    ("scaler", StandardScaler()),
    ("ridge", ridge_base)
])

# Hyperparamètres
param_grid_ridge = {
    "ridge__estimator__alpha": [0.01, 0.1, 1, 10, 100]
}

# Apprentissage passé puis on valide sur le futur
tscv = TimeSeriesSplit(n_splits=5)

grid_ridge = GridSearchCV(
    pipe_ridge,
    param_grid_ridge,
    cv=tscv,
    scoring="neg_mean_squared_error",
    n_jobs=-1
)

grid_ridge.fit(X_train, y_train)

ridge_best = grid_ridge.best_estimator_

print("Meilleurs hyperparamètres Ridge :", grid_ridge.best_params_)
print("MSE CV Ridge :", -grid_ridge.best_score_)
```

```
Meilleurs hyperparamètres Ridge : {'ridge__estimator__alpha': 100}
MSE CV Ridge : 0.718700647354126
```

Relus, uniformes et lisibles

Résultats de Williams

Le ML classique

L'objectif de cette partie est de savoir jusqu'où des modèles de Machine Learning classiques peuvent prédire l'évolution de la SST réduite

On utilise le one-step forecast pour répondre au problème : Prédire l'état réduit en t+1 à partir de l'état présent.

On utilise les modèles suivants :
- Ridge : régression linéaire régularisée
- MLP : Multi-Layer Perception
- LSTM : Réseau récurrent à mémoire longue
- GRU : Réseau récurrent simplifié

Résultats RMSE globale :
- 1.20 pour le LSTM
- 1.17 pour le GRU
- 1.16 pour le MLP
- 0.96 pour le Ridge

Résultats de Robin

Pré-traitement

Extraction de l'état réduit par sélection (LASSO)

$$\min_{\beta} ||y - X\beta||_2^2 + \alpha||\beta||_1$$

$$\min_{\beta} ||y - X\beta||_2^2 + \lambda\mathcal{R}(\beta)$$

Explication de la notion de régularisation

Comparaison avec la règle de Kaiser

```
# Comparison with Kaiser criterion
# Method where we keep all modes with singular values greater than the average singular value (or 1)
avgSingularValue = np.mean(singularValues)
KOptKaiser = np.sum(singularValues > avgSing  (variable) KOptKaiser: numpy.bool[builtins.bool]
print("Nombre de modes retenus (Kaiser) :", KOptKaiser)
print("Avec un thereshold à 1, on aurait :", np.sum(singularValues > 1.0))

Nombre de modes retenus (delta) : 49 pour un delta d'erreur < 0.001
Nombre de modes retenus (delta) : 90 pour un delta d'erreur < 0.0005
Nombre de modes retenus (delta) : 381 pour un delta d'erreur < 0.0001
Nombre de modes retenus (Kaiser) : 655
Avec un thereshold à 1, on aurait : 2531
```

Ce qu'on va faire pendant les 7 prochains jours ...

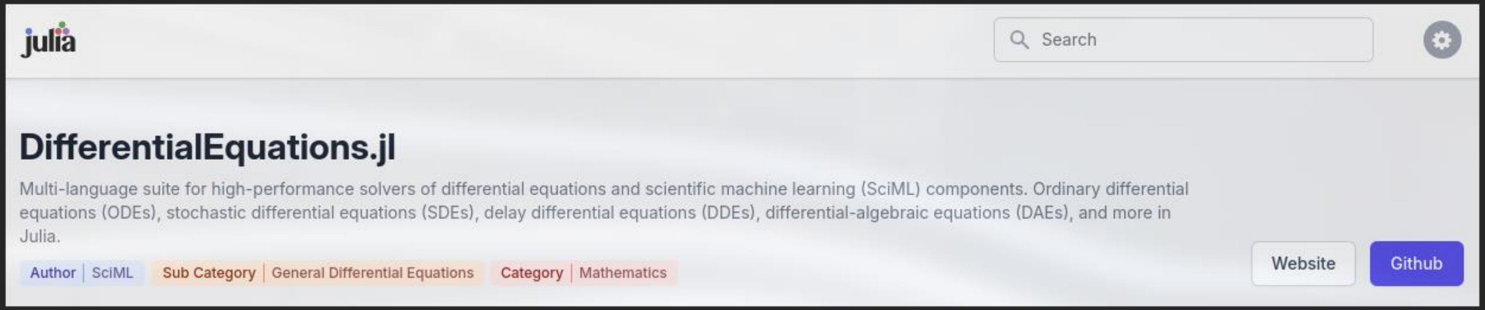
SciML

Tutoriel installation + utilisation de Julia

Rapport de présentation de Julia

Guide théorique de la modélisation SciML + choix modèles à implémenter

Implémentation de premiers modèles SciML



Merci de votre écoute !

Template inspirée de **Noé** aka **SkohTV**