代理モデル(プロキシ・サロゲートモデル)構築の実践

実践 貯留層工学 データ駆動型シミュレータ編 第2回(詳細版)

本日のアジェンダ

- パートA: 前回(第1回)の振り返り ―プロキシモデルの分類―
- パートB: 応答曲面法(RSM)・回帰モデルによる代理モデル構築実践
- パートC: 機械学習モデル(ランダムフォレスト・ニューラルネットワーク)による構築実践
- パートD: 先端手法(ROM・Fourier Neural Operator等)の概要と実装のポイント
- パートE: 精度検証の考え方

パートA

前回(第1回)の振り返り

› 本パートのねらい: 前回学んだ『なぜプロキシモデルが必要か』『どう分類されるか』を短時間で振り返り、本日の実践パートへの橋渡しとする

› 前回は、従来型シミュレーションの計算負荷・不確実性評価の課題(パートA)から出発し、プロキシモデル・サロゲートモデルの位置づけと分類(パートB)を整理した

› 本日は、この分類に沿って『実際に手を動かして構築する』段階に進む

A1: 前回の振り返り(1) なぜプロキシモデルが必要か

■ グリッド・タイムステップ増大、ヒストリーマッチング、モンテカルロ的不確実性評価、井戸配置最適化はいずれも『シミュレーションを何度も繰り返す』ことがボトルネック

■ プロキシモデルはこの反復計算を高速な近似モデルで代替する発想

■ 計算コスト削減・不確実性定量化・最適化への応用が主な目的

出典(社内資料): 調査部門/成果物『データ駆動型シミュレータ編_第1回_詳細版.pptx』パートA・B

A2: 前回の振り返り(2) 3分類の全体像

■ (1)統計的手法: 応答曲面法(RSM)・多項式回帰・クリギング(Kriging)

■ (2)機械学習手法: ランダムフォレスト・サポートベクターマシン・ニューラルネットワーク(MLP)

■ (3)先端手法: Reduced Order Model(ROM/POD)、Fourier Neural Operator、深層学習サロゲート(CNN/LSTM等)

出典: ResearchGate, 'Advancements in Machine Learning-Driven Proxy Modelling Techniques for Reservoir Engineering: A Systematic Review'
https://www.researchgate.net/publication/399879336_Advancements_in_Machine_Learning-Driven_Proxy_Modelling_Techniques_for_Reservoir_Engineering_A_Systematic_Review

プロキシモデルの分類マップ(復習)

統計的手法	機械学習手法	先端手法	ハイブリッド
応答曲面法(RSM)	ランダムフォレスト	ROM/POD	dual-driven
多項式回帰	SVM	Fourier Neural Operator	Physics-informed NN
クリギング	ニューラルネット	深層学習サロゲート	

A3: 前回の振り返り(3) 利点と限界

- 利点: 計算コストの大幅削減、多数回評価が前提の最適化・不確実性評価との親和性
- 限界: 学習データ範囲外への外挿での精度低下、データ量・品質への依存、モデルの説明可能性の課題
- 手法選択(統計的/機械学習/先端)は、データ量・非線形性の強さ・解釈性要求に応じて使い分ける

出典: Academia.edu, 'Development of proxy models for petroleum reservoir simulation: a systematic literature review and state-of-the-art'
https://www.academia.edu/44346638/Development_of_proxy_models_for_petroleum_reservoir_simulation_a_systematic_literature_review_and_state-of-the-art

A4: 本日『実際に構築する』フェーズへ

- 本日はパートB(統計的手法)→パートC(機械学習手法)→パートD(先端手法)の順に、具体的な実装手順を扱う
- パートEで精度検証の考え方を整理

パートB

応答曲面法(RSM)・回帰モデルによる代理モデル構築実践

› 本パートのねらい: 統計的手法(RSM・多項式回帰・クリギング)による代理モデル構築の具体的な手順とツールを理解する

› 実験計画法(DOE)によるサンプリング設計 → モデルフィッティング → 検証、という一連の流れを扱う

› 使用するPythonライブラリ(pyDOE2・scikit-learn・SMT等)も紹介する

B1: RSMによる代理モデル構築の全体設計

■ 応答曲面法(Response Surface Methodology, RSM)は、少数の入力パラメータ組み合わせでシミュレーションを実行し、その結果に関数(多項式等)をフィッティングして応答曲面を構築する統計的手法

■ 入力(物性・操業条件等の設計変数)と出力(生産量・回収率等の応答)の関係を、少ない実行回数で近似することを目的とする

■ ヒストリーマッチング・不確実性評価・最適化における古典的かつ実務で広く使われる手法

出典: Penn State University, STAT 503, 'Lesson 11: Response Surface Methods and Designs' https://online.stat.psu.edu/stat503/lesson/11

B2: 実験計画法(DOE)の目的と全体像

■ DOEは、限られたシミュレーション実行回数で入力空間をできるだけ効率的にカバーするためのサンプリング設計手法

■ 古典的要因計画(Full/Fractional Factorial)、応答曲面計画(Box-Behnken・Central Composite)、空間充填計画(Latin Hypercube Sampling等)に大別される

■ 目的(スクリーニング/応答曲面推定/空間全体の近似)に応じて設計手法を選ぶ

出典: SPE Annual Technical Conference and Exhibition, 'Experimental Design or Monte Carlo Simulation? Strategies for Building Robust Surrogate Models'
https://onepetro.org/SPEATCE/proceedings-abstract/15ATCE/15ATCE/D031S030R005/180367

B3: 具体的手順(1) 古典的要因計画・応答曲面計画

■ 全要因計画(Full Factorial)は各因子の全水準組み合わせを網羅するが、因子数増加で実行回数が指数的に増加する

■ 一部実施要因計画(Fractional Factorial)・Plackett-Burman計画は、スクリーニング段階で影響の大きい因子を絞り込むのに使われる

■ Box-Behnken計画・中心複合計画(Central Composite Design)は、2次(曲面)の応答をフィッティングするための代表的な応答曲面計画

出典: GitHub, clicumu/pyDOE2, 'Design of experiments for Python' https://github.com/clicumu/pyDOE2

B4: 具体的手順(2) Latin Hypercube Sampling(LHS)

■ LHSは、各入力変数の値域を等確率の区間に分割し、各区間から重複なく1点ずつサンプリングする空間充填型の設計手法

■ 因子数が多い場合でも、少ない実行回数で入力空間全体をバランスよくカバーできる

出典: Viana, F.A.C., 'A Tutorial on Latin Hypercube Design of Experiments', Quality and Reliability Engineering International, Vol.32, No.5(2016)
https://onlinelibrary.wiley.com/doi/10.1002/qre.1924(※購読が必要 purchase_list参照)

B5: Pythonツール(1) pyDOE2によるサンプリング設計

■ pyDOE2は、全要因計画(fullfact)・2水準要因計画(ff2n)・一部実施要因計画(fracfact)・Plackett-Burman(pbdesign)・Box-Behnken(bbdesign)・中心複合(ccdesign)・Latin Hypercube(lhs)を提供するオープンソースライブラリ

■ `from pyDOE2 import lhs` のように呼び出し、因子数とサンプル数を指定するだけで設計行列を生成できる

■ 生成した設計点をもとに、シミュレータ(またはベンチマークデータ)を実行して応答値を得る

出典: GitHub, clicumu/pyDOE2, 'Design of experiments for Python' https://github.com/clicumu/pyDOE2

B6: 多項式回帰のフィッティング手順

■ 手順1: 入力変数の中心化・コーディング(規格化)により多重共線性を軽減する

■ 手順2: 1次モデル(主効果のみ)→2次モデル(交互作用・2乗項を含む)の順に当てはめを試み、必要な次数を判断する

■ 手順3: 最小二乗法により回帰係数を推定する(Pythonでは`numpy.polyfit`や`statsmodels.OLS`等を利用)

出典: MDPI, 'Importance of Using Modern Regression Analysis for Response Surface Models in Science and Technology' https://www.mdpi.com/2076-3417/15/13/7206

B7: RSMモデルの検証(ANOVA・Lack-of-Fit検定)

- 分散分析(ANOVA)のF検定により、モデル全体が統計的に有意かどうかを確認する
- Lack-of-Fit検定により、選択した多項式の次数が実際のデータの曲がり方を十分に捉えているかを確認する
- 決定係数(R2)・残差プロットもあわせて確認し、モデルの妥当性を総合的に判断する

出典: Penn State University, STAT 503, 'Lesson 11: Response Surface Methods and Designs' https://online.stat.psu.edu/stat503/lesson/11

B8: クリギング(Gaussian Process回帰)の基礎

■ クリギングは、空間統計(地球統計学)に由来する手法で、既知点からの距離に応じた共分散関数(カーネル)を用いて未知点の値を予測する

■ 予測値だけでなく予測の不確実性(分散)も同時に得られる点が、多項式回帰にない特徴

■ 機械学習分野では『ガウス過程回帰(Gaussian Process Regression)』としてほぼ同じ理論的枠組みで扱われる

出典: Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P., 'Design and Analysis of Computer Experiments', Statistical Science, Vol.4, No.4(1989)(論文・購読が必要 ※purchase_list参照)

B9: 実装ツール クリギング・応答曲面のPythonライブラリ

■ scikit-learnの`GaussianProcessRegressor`は、カーネル(RBF等)を指定してガウス過程回帰を手軽に実装できる。カーネルのハイパーパラメータは対数周辺尤度最大化により自動最適化される(`n_restarts_optimizer`で局所解対策)

■ SMT(Surrogate Modeling Toolbox)は、クリギング・radial basis function等の代理モデル手法とサンプリング手法をまとめたPython専用ライブラリで、勾配(微分)対応が特徴

■ 両者とも`pip install`で導入可能。第2回演習ではまずscikit-learnでの実装を扱う

出典: scikit-learn, 'Gaussian Processes' https://scikit-learn.org/stable/modules/gaussian_process.html ; SMT: Surrogate Modeling Toolbox documentation https://smt.readthedocs.io/en/latest/

B10: RSM・クリギングの使い分けと限界

- 少数データ・低次元(因子数が少ない)問題では、多項式回帰・RSMが解釈しやすく実務で扱いやすい
- 非線形性が強い・因子間相互作用が複雑な系では、クリギングの方が精度が高くなりやすいが、計算コスト(共分散行列の逆行列計算)は因子数・データ数の増加に伴い増大する
- いずれも、学習データの範囲外(外挿領域)では精度が大きく低下する点に注意が必要

出典: ResearchGate, 'Response Surface Methodology Approach for History Matching and Uncertainty Assessment of Reservoir Simulation Models' https://www.researchgate.net/publication/254527939_Response_Surface_Methodology_Approach_for_History_Matching_and_Uncertainty_Assessment_of_Reservoir_Simulation_Models

B11: パートBまとめ

■ RSM・回帰・クリギングは、DOEによるサンプリング設計 → モデルフィッティング → ANOVA/Lack-of-Fit等による検証、という流れで構築する

■ pyDOE2(サンプリング)、scikit-learn/SMT(フィッティング)という組み合わせが、Pythonでの実装の基本形

■ 次パートでは、より非線形性・データ量に強い機械学習手法(ランダムフォレスト・ニューラルネットワーク)を扱う

RSM構築の3ステップ

パートC

機械学習モデル(ランダムフォレスト・ニューラルネットワーク)による構築実践

› 本パートのねらい: scikit-learn・PyTorch等を用いた具体的な実装ステップと、ハイパーパラメータ調整の考え方を理解する

› パートBの統計的手法に比べ、非線形性の強い系・データ量が多い場合に精度が出やすい一方、過学習リスクへの対策が必要になる

› ランダムフォレスト(決定木系アンサンブル)とニューラルネットワーク(MLP)の2手法を扱う

C1: 機械学習モデルによる代理モデル構築の全体設計

- 手順は概ね共通: データ収集・前処理 → 訓練/検証/テストデータ分割 → モデル学習 → ハイパーパラメータ調整 → 精度検証(パートEで詳説)
- 入力(物性・操業条件等)の正規化・標準化(特にニューラルネットワークでは学習の安定性に直結)
- 特徴量エンジニアリング(交互作用項の追加、無関係な特徴量の除外等)が精度向上に寄与する

出典: ResearchGate, 'Advancements in Machine Learning-Driven Proxy Modelling Techniques for Reservoir Engineering: A Systematic Review'
https://www.researchgate.net/publication/399879336_Advancements_in_Machine_Learning-Driven_Proxy_Modelling_Techniques_for_Reservoir_Engineering_A_Systematic_Review

C2: ランダムフォレストの実装手順(1) 基本形

■ scikit-learnでは`from sklearn.ensemble import RandomForestRegressor`の後、`.fit(X_train, y_train)`で学習、`.predict(X_test)`で予測する数行のコードで実装できる

■ 決定木を多数(`n_estimators`)組み合わせたアンサンブル学習で、各木は特徴量・データのランダムサブサンプルで学習する(バギング)

■ 貯留層の生産量予測・プロキシモデル構築での適用例が複数報告されている

出典: SBC, 'Predicting oil field production using the Random Forest algorithm' https://sol.sbc.org.br/index.php/sibgrapi_estendido/article/download/23277/23104/

C3: ランダムフォレストの実装手順(2) 特徴量重要度・OOB評価

- `feature_importances_`属性により、どの入力パラメータが予測結果に効いているかを定量的に把握できる(解釈性の高さがRFの利点)
- Out-of-Bag(OOB)スコア(`oob_score=True`)により、別途検証データを用意せずにモデル精度の目安を得られる
- Volveデータセットを用いた坑井検層値予測でも、ランダムフォレスト回帰の実装例が報告されている

出典: Towards Data Science, 'Random Forest Regression for Continuous Well Log Prediction' https://towardsdatascience.com/random-forest-regression-for-continuous-well-log-prediction-61d3ec1c683a/

C4: ハイパーパラメータ調整の考え方(1) 主要パラメータ

- `n_estimators`(木の本数): 多いほど安定するが計算コストが増える
- `max_depth`・`max_leaf_nodes`(木の深さ): 大きすぎると過学習、小さすぎると未学習(underfitting)につながる
- `min_samples_leaf`・`min_samples_split`: 葉ノードの最小サンプル数を制御し、過学習を抑える

出典: scikit-learn, 'Tuning the hyper-parameters of an estimator' https://scikit-learn.org/stable/modules/grid_search.html

C5: ハイパーパラメータ調整の考え方(2) 探索手法

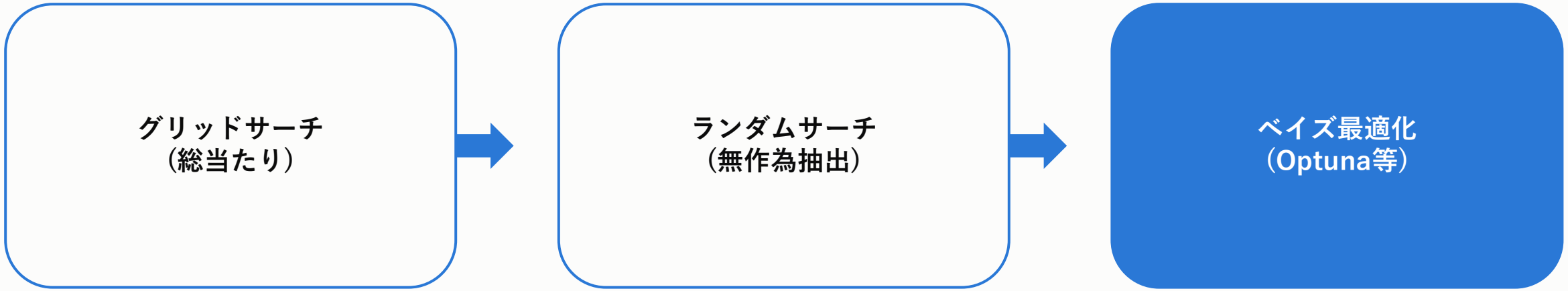
■ グリッドサーチ(`GridSearchCV`): 候補パラメータの全組み合わせを総当たりで評価。網羅的だが組み合わせ数が多いと計算コストが高い

■ ランダムサーチ(`RandomizedSearchCV`): パラメータ空間からランダムにサンプリングして評価。次元が高い場合に効率的

■ ベイズ最適化(Optuna等): 過去の評価結果を踏まえて次に評価するパラメータを賢く選び、少ない試行回数で良い解に到達しやすい

出典: Akiba, T. et al., 'Optuna: A Next-generation Hyperparameter Optimization Framework', KDD 2019, arXiv:1907.10902 https://arxiv.org/abs/1907.10902

ハイパーパラメータ探索手法の発展



C6: ニューラルネットワークの実装手順(1) モデル構築

■ PyTorchでは`torch.nn.Module`を継承したクラスに全結合層(`nn.Linear`)と活性化関数(ReLU等)を積み重ねてMLPを定義する

■ 入力層のノード数=特徴量数、出力層のノード数=予測したい応答変数の数(生産量・圧力等)に対応させる

■ 隠れ層の数・ノード数は、データ量や問題の複雑さに応じて調整する(パートC8で詳説)

出典: PyTorch公式ドキュメント https://pytorch.org/docs/stable/index.html

C7: ニューラルネットワークの実装手順(2) 学習ループ

■ 損失関数(平均二乗誤差MSEや平均絶対誤差MAE等)を定義し、Adam等の最適化アルゴリズムで重みを更新する

■ 順伝播(forward)→損失計算→逆伝播(backward)→パラメータ更新、をエポック数だけ繰り返す

■ ミニバッチ(`DataLoader`)を用いることで、大規模データでも安定した学習が可能になる

出典: PyTorch公式ドキュメント https://pytorch.org/docs/stable/index.html

C8: ハイパーパラメータ調整の考え方(NN)

- 主な調整対象: 層数・各層のノード数、学習率(learning rate)、バッチサイズ、エポック数
- 学習率が大きすぎると発散し、小さすぎると収束が遅くなる。学習率スケジューラの活用も有効
- OptunaのようなフレームワークとPyTorchを組み合わせ、これらを自動探索する事例が報告されている

出典: Towards Data Science, 'Hyperparameter Tuning of Neural Networks with Optuna and PyTorch' https://towardsdatascience.com/hyperparameter-tuning-of-neural-networks-with-optuna-and-pytorch-22e179efc837/

C9: 過学習・データ量不足への対策

■ 正則化(L2正則化・weight decay)、ドロップアウト(Dropout層)により、モデルが訓練データに過剰適合するのを防ぐ

■ Early stopping(検証データの損失が改善しなくなった時点で学習を打ち切る)も広く使われる対策

■ データ量が少ない場合、パートB(統計的手法)への切替や、データ拡張・転移学習の検討も選択肢となる(プレースホルダー: 具体的なデータ拡張手法は教材開発時に確定)

C10: ランダムフォレスト vs ニューラルネットワークの使い分け

■ ランダムフォレスト: 実装・調整が比較的容易、特徴量重要度による解釈性が高い、中規模データで頑健

■ ニューラルネットワーク: 大規模データ・高次元入力・複雑な非線形関係に強いが、データ量・計算資源への要求が高く、ハイパーパラメータ調整の手間も大きい

■ 実務では、まずランダムフォレストで基準精度を把握し、必要に応じてニューラルネットワークで精度向上を試す進め方が現実的

出典: ResearchGate, 'Advancements in Machine Learning-Driven Proxy Modelling Techniques for Reservoir Engineering: A Systematic Review' https://www.researchgate.net/publication/399879336_Advancements_in_Machine_Learning-Driven_Proxy_Modelling_Techniques_for_Reservoir_Engineering_A_Systematic_Review

ランダムフォレスト vs ニューラルネットワーク

ランダムフォレスト		ニューラルネットワーク
実装・調整が容易	使い分け	大規模・高次元データに強い
特徴量重要度で解釈しやすい		調整の手間が大きい

C11: パートCまとめ

■ ランダムフォレスト・ニューラルネットワークとも、scikit-learn/PyTorchを使えば数十行程度のコードで基本形を実装できる

■ ハイパーパラメータ調整(グリッドサーチ→ランダムサーチ→ベイズ最適化)と過学習対策(正則化・early stopping)が精度を左右する

■ 次パートでは、さらにデータ量・計算コストの制約が厳しい場合に有効な先端手法(ROM・FNO)を扱う

パートD

先端手法(ROM・Fourier Neural Operator等)の概要と実装のポイント

› 本パートのねらい: 第1回で紹介したReduced Order Model(ROM)・Fourier Neural Operator(FNO)等について、実装面での留意点を整理する

› 本講座では概論紹介にとどめ、詳細な数式展開は扱わない(第1回たたき台からの方針を踏襲)

› データ量・計算資源・専門知識のハードルが、統計的手法・機械学習手法より高い点に注意

D1: 先端手法の位置づけ(復習)

■ ROM(Reduced Order Model)は、高次元シミュレーション結果を少数の代表変数(低次元空間)に圧縮して計算する手法

■ Fourier Neural Operator(FNO)は、関数空間から関数空間への写像(オペレータ)を学習する『ニューラルオペレータ』の一種

■ いずれも、統計的手法・従来型機械学習よりも大規模・高次元なシミュレーション出力(空間分布・時系列全体)を扱うのに適している

出典(社内資料): 調査部門/成果物『データ駆動型シミュレータ編_第1回_詳細版.pptx』B9-B10

D2: ROM実装ステップ(1) スナップショット収集とSVD

■ 手順1: 物理ベースシミュレータを複数の入力条件(DOE等で設計)で実行し、各時刻・各条件での状態量(圧力・飽和率分布)を『スナップショット』として収集する

■ 手順2: スナップショット行列に特異値分解(SVD)を適用し、Proper Orthogonal Decomposition(POD)の基底(主要モード)を得る

■ 基底の数(次元圧縮の度合い)は、特異値の減衰の様子を見て決定する

出典: Cardoso, M.A., 'Development and Application of Reduced-Order Modeling for the Solution of Oil Reservoir Simulation Problems', Stanford University PhD Dissertation(SUPRI-B) https://pangea.stanford.edu/ERE/pdf/pereports/PhD/Cardoso09.pdf

D3: ROM実装ステップ(2) 低次元空間での時間発展・再構成

■ 得られた低次元基底に、支配方程式(圧力・飽和率の保存則)を射影(Galerkin射影)し、低次元空間で時間発展を計算する

■ Trajectory Piecewise Linearization(TPWL)等、非線形性を扱うための追加手法と組み合わせる例が報告されている

■ 低次元空間での解を、基底を使って元の高次元空間(グリッド全体)に再構成し、圧力・飽和率分布を得る

出典: Stanford SUETRI-B, 'Reduced-Order Modeling for Compositional Simulation by Use of Trajectory Piecewise Linearization' https://suetri-b.stanford.edu/publications/journal-articles/reduced-order-modeling-compositional-simulation-use-trajectory

D4: ROM実装の留意点・Pythonツール

■ POD自体はnumpy/scipyの特異値分解(SVD)機能だけでも実装可能だが、EZyRB等のPython向けROM専用ライブラリも存在する

■ シミュレータ本体との連携(スナップショット取得のための複数回実行環境)が必要な点が、統計的手法・機械学習手法と異なる実装上のハードル

■ (プレースホルダー: 本講座で使用する具体的なROM実装ライブラリ・演習データは教材開発時に確定)

出典: GitHub, vassilikitsios/snapshot_pod_rom_py, 'Python code to calculate proper orthogonal decomposition modes'
https://github.com/vassilikitsios/snapshot_pod_rom_py

D5: FNO実装ステップ(1) データ形式・前処理

- FNOは、入力・出力を規則的なグリッド(格子)上の関数として扱う。浸透率場・圧力分布等の空間データをテンソル(多次元配列)として準備する
- 解像度非依存性(訓練時と異なる解像度のグリッドでも推論可能)が理論上の特徴とされる
- 井戸条件・浸透率場が変化する場合のプロキシモデルとしての応用が報告されている

出典: Li, Z. et al., 'Fourier Neural Operator for Parametric Partial Differential Equations', arXiv:2010.08895(2020) https://arxiv.org/abs/2010.08895

D6: FNO実装ステップ(2) neuraloperatorライブラリでの構築

■ 公式実装ライブラリ`neuraloperator`(PyTorchエコシステムの一部)を用いると、`from neuralop.models import FNO`のように数行でモデルを定義できる

■ Fourier空間での積分カーネルのパラメータ化により、畳み込みニューラルネットワークより効率的に長距離の空間相関を学習できるとされる

■ 学習には、通常のニューラルネットワーク同様、PyTorchの学習ループ(損失関数・Adam等)を利用する

出典: neuraloperator(GitHub公式), 'Learning in infinite dimension with neural operators' https://github.com/neuraloperator/neuraloperator ; neuraloperator documentation, 'Fourier Neural Operators' https://neuraloperator.github.io/dev/theory_guide/fno.html

D7: 先端手法導入のハードル

■ データ量: ROM・FNOとも、十分な数のシミュレーション実行(スナップショット・訓練サンプル)が前提となり、統計的手法より多くの学習データを要することが多い

■ 計算資源: 深層学習ベースの手法(FNO等)は、GPU環境での学習が現実的な選択肢になる場合が多い

■ 専門知識: 実装・チューニングには、偏微分方程式・信号処理(フーリエ変換)等の背景知識が必要になる場面がある

出典(社内資料): 調査部門/参考資料/新規講習企画書_データ駆動型シミュレータ編.md(1章)

D8: パートDまとめ

- ROM(POD)はSVDによる次元圧縮+低次元空間での時間発展、FNOはFourier空間での演算子学習という、それぞれ異なるアプローチで高次元・大規模シミュレーションを近似する
- いずれも実装のハードルは統計的手法・機械学習手法より高く、本講座では概論・実装の考え方の紹介にとどめる
- 次パートでは、これまで扱った全手法に共通する『精度検証』の考え方を整理する

ROM vs Fourier Neural Operator

ROM(POD)		Fourier Neural Operator
SVDによる次元圧縮	先端手法	Fourier空間で演算子学習
低次元空間で時間発展		解像度非依存

パートE

精度検証の考え方

› 本パートのねらい: どの手法(RSM・機械学習・先端手法)を使う場合にも共通する、代理モデルの精度検証の考え方を整理する

› 交差検証・ホールドアウト法という2つの基本的な検証アプローチと、代表的な評価指標(RMSE・R2・MAE)を扱う

› パートFの演習(Ex.2)で、実際にこれらを計算する

E1: ホールドアウト法(訓練/検証/テストデータ分割)

- データを訓練データ・検証データ・テストデータに分割し、訓練データでモデルを学習、検証データでハイパーパラメータ調整、テストデータで最終的な精度を評価する
- scikit-learnの`train_test_split`関数で簡単に分割できる(`test_size`引数で比率を指定)
- データ量が少ない場合、単純な1回の分割では評価結果が分割の仕方に左右されやすい点に注意

出典: scikit-learn, 'Cross-validation: evaluating estimator performance' https://scikit-learn.org/stable/modules/cross_validation.html

E2: 交差検証(k-fold CV)

■ データをk個のグループ(fold)に分割し、そのうち1つを検証用、残りを訓練用として、k回繰り返して評価を平均する手法

■ 実務上はk=5〜10が選ばれることが多い。データ量が少ない代理モデル構築では特に有効な検証手段とされる

■ scikit-learnの`cross_val_score`・`KFold`・`cross_validate`関数で実装できる

出典: scikit-learn, 'Cross-validation: evaluating estimator performance' https://scikit-learn.org/stable/modules/cross_validation.html

E3: 評価指標(1) RMSE・MAE

■ RMSE(Root Mean Squared Error、二乗平均平方根誤差): 予測値と実測値の差を二乗して平均し平方根を取った指標。大きな誤差を強く罰する

■ MAE(Mean Absolute Error、平均絶対誤差): 予測値と実測値の差の絶対値の平均。外れ値の影響を受けにくい

■ どちらも値が小さいほど精度が高いことを示すが、単位は元の応答変数(生産量等)と同じになる点に注意

出典: MachineLearningMastery.com, 'A Gentle Introduction to k-fold Cross-Validation'(評価指標の一般的解説として参照) https://machinelearningmastery.com/k-fold-cross-validation/

E4: 評価指標(2) 決定係数R2・使い分け

■ R2(決定係数): モデルが応答の分散をどれだけ説明できているかを示す指標(1に近いほど当てはまりが良い)

■ RMSE・MAEは『誤差の大きさ』、R2は『説明力』を示すため、両方を併用して評価するのが一般的

■ 複数手法(RSM・ランダムフォレスト・ニューラルネットワーク等)を比較する際は、同じ評価指標・同じテストデータで揃えて比較することが重要

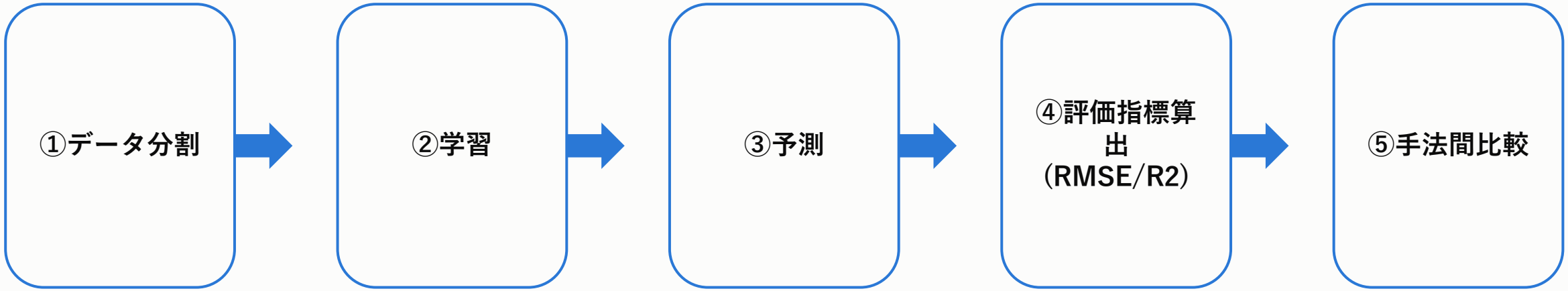
出典: scikit-learn, 'Cross-validation: evaluating estimator performance' https://scikit-learn.org/stable/modules/cross_validation.html

E5: パートEまとめ — 精度検証のワークフロー

■ 手順: データ分割(ホールドアウトまたはk-fold)→学習→予測→評価指標(RMSE・MAE・R2)の算出→手法間の比較

■ 適用範囲(学習データがカバーする入力パラメータの範囲)を明示し、範囲外への適用(外挿)には注意を促すことも重要

精度検証のワークフロー



参考資料一覧(出典) (1/4)

■ Li, Z. et al., 'Fourier Neural Operator for Parametric Partial Differential Equations'(arXiv:2010.08895, 2020) https://arxiv.org/abs/2010.08895

■ ResearchGate, 'Advancements in Machine Learning-Driven Proxy Modelling Techniques for Reservoir Engineering: A Systematic Review' https://www.researchgate.net/publication/399879336_Advancements_in_Machine_Learning-Driven_Proxy_Modelling_Techniques_for_Reservoir_Engineering_A_Systematic_Review

■ ResearchGate, 'Response Surface Methodology Approach for History Matching and Uncertainty Assessment of Reservoir Simulation Models' https://www.researchgate.net/publication/254527939_Response_Surface_Methodology_Approach_for_History_Matching_and_Uncertainty_Assessment_of_Reservoir_Simulation_Models

■ Academia.edu, 'Development of proxy models for petroleum reservoir simulation: a systematic literature review and state-of-the-art' https://www.academia.edu/44346638/Development_of_proxy_models_for_petroleum_reservoir_simulation_a_systematic_literature_review_and_state-of-the-art

■ SBC, 'Predicting oil field production using the Random Forest algorithm' https://sol.sbc.org.br/index.php/sibgrapi_estendido/article/download/23277/23104/

参考資料一覧(出典) (2/4)

■ GitHub, vassilikitsios/snapshot_pod_rom_py, 'Python code to calculate proper orthogonal decomposition modes' https://github.com/vassilikitsios/snapshot_pod_rom_py

■ ScienceDirect, 'Neural operator-based proxy for reservoir simulations considering varying well settings, locations, and permeability fields' https://www.sciencedirect.com/science/article/abs/pii/S0098300424003091

■ neuraloperator(GitHub公式), 'Learning in infinite dimension with neural operators' https://github.com/neuraloperator/neuraloperator

■ neuraloperator documentation, 'Fourier Neural Operators' https://neuraloperator.github.io/dev/theory_guide/fno.html

■ Viana, F.A.C., 'A Tutorial on Latin Hypercube Design of Experiments', Quality and Reliability Engineering International, Vol.32, No.5(2016) https://onlinelibrary.wiley.com/doi/10.1002/qre.1924

■ GitHub, clicumu/pyDOE2, 'Design of experiments for Python' https://github.com/clicumu/pyDOE2

■ SPE Annual Technical Conference and Exhibition, 'Experimental Design or Monte Carlo Simulation? Strategies for Building Robust Surrogate Models' https://onepetro.org/SPEATCE/proceedings-abstract/15ATCE/15ATCE/D031S030R005/180367

参考資料一覧(出典) (3/4)

■ scikit-learn, 'Gaussian Processes' https://scikit-learn.org/stable/modules/gaussian_process.html

■ scikit-learn, 'Cross-validation: evaluating estimator performance' https://scikit-learn.org/stable/modules/cross_validation.html

■ scikit-learn, 'Tuning the hyper-parameters of an estimator' https://scikit-learn.org/stable/modules/grid_search.html

■ MDPI, 'Importance of Using Modern Regression Analysis for Response Surface Models in Science and Technology' https://www.mdpi.com/2076-3417/15/13/7206

■ Penn State University, STAT 503, 'Lesson 11: Response Surface Methods and Designs' https://online.stat.psu.edu/stat503/lesson/11

■ Akiba, T. et al., 'Optuna: A Next-generation Hyperparameter Optimization Framework', KDD 2019, arXiv:1907.10902 https://arxiv.org/abs/1907.10902

■ Towards Data Science, 'Hyperparameter Tuning of Neural Networks with Optuna and PyTorch' https://towardsdatascience.com/hyperparameter-tuning-of-neural-networks-with-optuna-and-pytorch-22e179efc837/

参考資料一覧(出典) (4/4)

■ MachineLearningMastery.com, 'A Gentle Introduction to k-fold Cross-Validation' https://machinelearningmastery.com/k-fold-cross-validation/

■ Equinor, 'Volve field data set' https://www.equinor.com/energy/volve-data-sharing

■ SINTEF, 'spe10: Access to the SPE10 benchmark case' https://www.sintef.no/contentassets/2551f5f85547478590ceca14bc13ad51/spe10.html

■ Myers, R.H., Montgomery, D.C., Anderson-Cook, C.M., 'Response Surface Methodology: Process and Product Optimization Using Designed Experiments', 4th ed., Wiley(2016)(書籍・社内未所蔵)

■ Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P., 'Design and Analysis of Computer Experiments', Statistical Science, Vol.4, No.4(1989)(論文・社内未所蔵)

■ Mohaghegh, S.D., 'Data-Driven Reservoir Modeling', SPE/PennWell(2017, ISBN 9781613995600) https://store.spe.org/Data-Driven-Reservoir-Modeling-P1054.aspx

■ 企画部門(社内資料)『新規講習企画書_データ駆動型シミュレータ編.md』(2026年7月)

■ 調査部門(社内資料)『データ駆動型シミュレータ編_第1回_詳細版.pptx』(2026年7月)

購入が必要な参考資料一覧

■ Myers, R.H., Montgomery, D.C., Anderson-Cook, C.M., 'Response Surface Methodology: Process and Product Optimization Using Designed Experiments'(4th ed., Wiley, 2016、書籍) — 必要箇所: パートB『B6: 多項式回帰のフィッティング手順』『B7: RSMモデルの検証(ANOVA・Lack-of-Fit検定)』の数式的詳細・具体的な検定手順を一次資料として裏付けるために必要

■ Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P., 'Design and Analysis of Computer Experiments', Statistical Science, Vol.4, No.4(1989)(論文・有料) — 必要箇所: パートB『B8: クリギング(Gaussian Process回帰)の基礎』の理論的基礎付け(共分散関数・予測分散の数式)の引用に必要

■ Viana, F.A.C., 'A Tutorial on Latin Hypercube Design of Experiments', Quality and Reliability Engineering International(Wiley, 2016)(論文・購読が必要) — 必要箇所: パートB『B4: 具体的手順(2) Latin Hypercube Sampling(LHS)』のアルゴリズム詳細・具体的な事例の引用に必要(現状は要旨レベルの紹介に留まる)

■ Mohaghegh, S.D., 'Data-Driven Reservoir Modeling'(SPE/PennWell, 2017, ISBN 9781613995600、書籍) — 必要箇所: パートC・パートFの機械学習モデル構築ワークフロー全体および演習設計を、体系的な実務書と突き合わせて具体化する際の裏付けに必要