

1. はじめに

修士研究のテーマについて z

Adjusting Input Coefficients and Visualising Spillover Paths
in an Input-Output Analysis from Financial and Economic Data

蒲田 涼馬 (Ryoma Gamada)
u2455007@st.pu-toyama.ac.jp

富山県立大学 工学研究科
電子・情報工学専攻 情報基盤工学講座

May 13, 2025

1. 金融経済データからの産業連関分析における 投入係数の調整と波及経路の可視化

2/10

背景

近年、金融工学は計算機性能の向上やデータサイエンス手法の進化、公的機関によるオープンデータの提供に伴い、飛躍的な発展を遂げている、将来予測などの分野は成長しているにも関わらず、多様な要素の相互関係や経済の動向を可視化するような研究は多くない。



将来予測: 資源配分の最適化やリスク管理の向上



発展

影響分析: 政策分析や経営における戦略分析などに使われる



少ない

Figure 1: 研究の背景

1. はじめに

目的

AI, あるいは別の手法を用いて過去の金融データから自動的に法則を発見するシステムを作成する.

なるべく金融に特化したものを考え, 最終的には他の手法との比較を行う.
発見された数式の可読性を評価と金融市場の理解に貢献できる知見を得る.

1. はじめに

シンボリック回帰 (Symbolic Regression)

シンボリック回帰はデータセットに適合する数学的表現を自動的に探索する機械学習の手法.

従来の回帰分析がモデル構造を事前に仮定するのに対し, 記号的回帰はモデル構造自体を発見する.

PySR

PySR では損失関数を以下のように定義している.

$$\mathcal{L}(E) = \mathcal{L}_{pred}(E) + \lambda \cdot C(E) \quad (1)$$

ここで E は候補となる数式, $\mathcal{L}_{pred}(E)$ は予測誤差, $C(E)$ は式の複雑度, λ は複雑性に対するペナルティ係数を意味する.

これによって可読性, 精度の両方が高い式を生存させる.

一般的な進化アルゴリズム

1. 準備: 多くの数式の集まりを用意する. 個体の良し悪しを測る基準と個体を変化させるルールを設定する.
2. トーナメントによる個体の厳選: 集団の中からランダムに数個の個体 (サブセット) を選ぶ. サブセット内で適合度を評価し, 競争させる.
3. 新しい個体の生成: トーナメントで選ばれた個体をコピーし, そのコピーにランダムに突然変異を加えて少し変化させる.
4. 集団の更新: 新しく生成した個体とサブセット内で最も劣っていた個体を入れ替える.

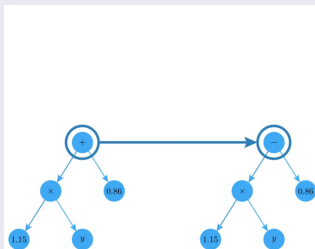


図3. 突然変異動作

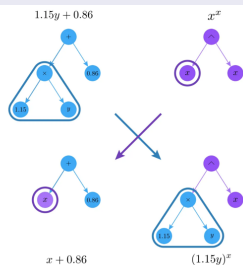


図4. 交叉動作

PySR

1. 準備: ランダムな「数式」を多く生成. PySR ではこの集団をさらに islands に分けてそれぞれで進化を進める.
2. 数式の評価: 損失関数の計算を行い各数式がどれだけの優劣なのかを評価する.
3. 数式の選択: 点数の高い数式が次の世代の親として選ばれやすくなる.
4. 新しい数式の作成: 選ばれた親の数式を元に新たな数式を生成する. ここで交叉, 突然変異, 定数最適化を行う.
5. 数式集団の入れ替え: 新しくできた数式と元の集団の劣っている数式を入れ替える.
6. islands 間の交流: 定期的に各 islands から優秀な数式が他の islands へ移動する.
7. 繰り返しと最良の数式の選択

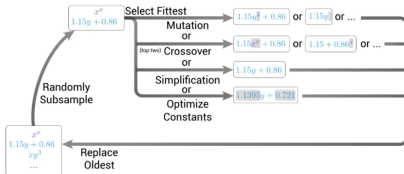


図3. PySRの内部ループ

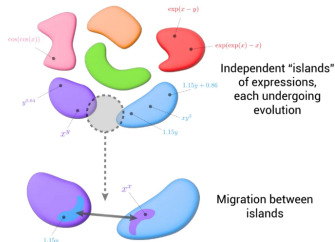


図4. PySRの外部ループ

1. はじめに

実装について

PySR はオープンソースのライブラリであるためコードの実装，修正を比較的容易に行える．

Julia と Python によってプログラムは作成されているがその両方が Github に存在する．

今回はデモデータを対象とした法則発見と実データ (金融データ) を対象とした式生成を行った．

対象としたデータについて

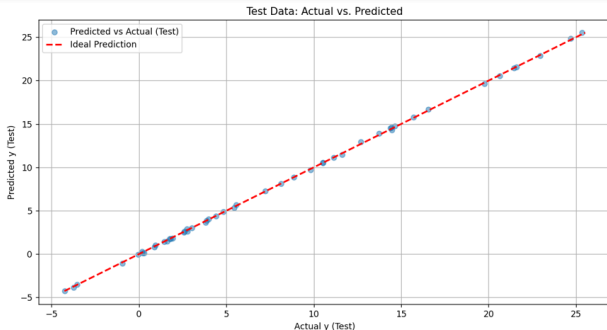
デモデータ:

$$y = 2.5 * \cos(x_0) + x_1^2 - 0.5x^2 \quad (2)$$

実データ: 目的変数がガスの日足株価 (終値)，説明変数が日足その他変数 (40 個) の計 41 のデータ．データ数は 500 程度．

結果 1

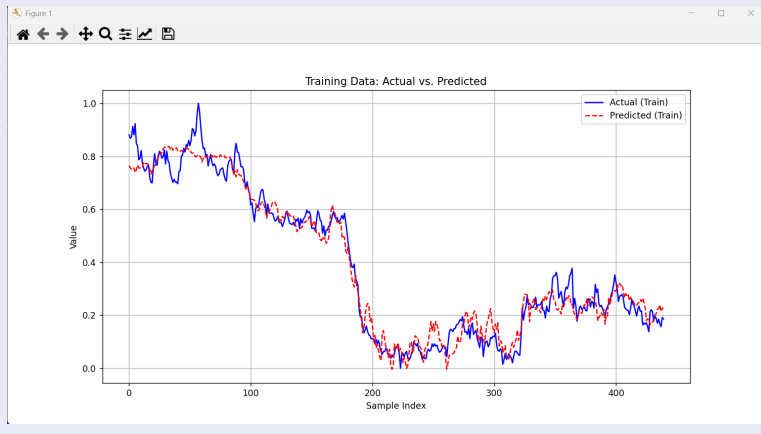
デモデータについて実行した結果を以下に示す。



```
Equation String: (square(x1) + (cos(x0) * 2.5080454)) - (x2 * 0.49848592)
SymPy Format:    x1**2 - 0.49848592*x2 + cos(x0)*2.5080454
LaTeX Format:    x_{1}^{2} - x_{2} \cdot 0.49848592 + \cos{\left(x_{0} \right)} 2.5080454
Score:          2.4121
Loss (MSE):     0.0089
Complexity:     11
```


結果 2

実データについて実行した結果を以下に示す。



1. はじめに

今回のまとめ

研究の方針を変えた.

手法の調査とデモデータと実データに対しての式生成を行った.

今後 1

デモデータについてはかなり正解に近い式が得られた.

しかし実データについてはフィッティングはそこそこできていそうではあったがテストデータで式の検証をしてみたところかなり渋い結果担っていた.

今回はデータ数が少なく十分に学習できていなかった可能性, またデータの処理が不十分だったことも考えられるので実験と検証を繰り返す必要がある.

また新規性についても考え, それをプログラムに組み込んで実験を行う必要がある.