公開

# 研 究 終 了 報 告 書

## 「試行錯誤を効率化する Jupyter Notebook 拡張」

研究期間: 2021 年 10 月～2024 年 3 月
研 究 者: 中丸 智貴

### 1. 研究のねらい

本研究で目指すのは、データ可視化や AI 構築などに取り組む際に必要不可欠な試行錯誤を効率化し、データを活用した社会や学問の発展を加速させることである。データを活用した社会や学問の発展は、既に至る所で試みられている。日常に関わる例であれば、商品推薦や自動運転などが挙げられる。学術領域では、超新星爆発の分類や石器の分類のような事例が挙げられる。このようなデータ活用という大きな流れは、今後もさらに広がっていくことが予想される。

データ可視化や AI 構築では、より良い可視化(図表)や予測性能を得るために、プログラムの修正と実行結果の確認を繰り返すという試行錯誤が行われている。これは、可視化/予測の手法やハイパーパラメータ値を事前に確定させることが困難であることに起因する。統計的機械学習技術の発展に伴いプログラミングを自動化する様々な道具が登場したものの、前述したような試行錯誤が完全に解消されるには至っていない。機械学習技術については今後も発展が期待されるが、それでも試行錯誤の完全な解消は困難であると予想される。必要な図表や AI は個別の状況ごとに異なり、それぞれの状況に応じた適切なチューニングが本質的に避けられないからである。

本研究で解消を目指すのは、愚直にプログラムの修正と実行が繰り返した時に生じる、試行間で共通する部分の無駄な再計算である。図1は、機械学習モデルを訓練するプログラムにおいて、データ前処理を変更する試行を行った際に発生する無駄な再計算の例である。この例において、試行1と試行2の冒頭2行は完全に同一の処理であり、再計算は無駄である。このような再計算は電力を浪費し、環境的にも経済的にも望ましくない。さらに、プログラマが再計算の終了まで待つ必要もあり、労働生産性の悪化にもつながる。

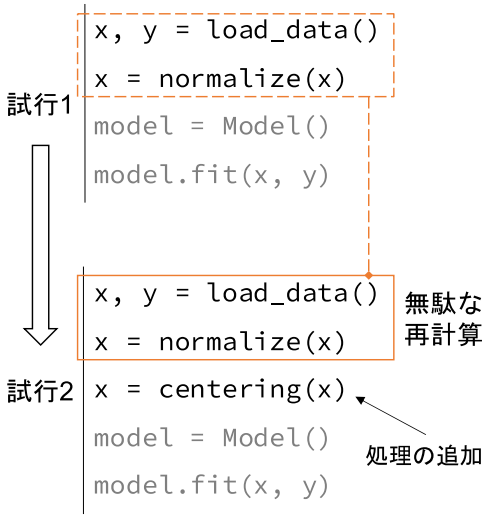


図 1. 無駄な再計算

本研究は、効率的に試行錯誤を進められる新しいノートブックプログラミング環境 Multiverse Notebook の実現を通して、データを活用した社会/学問の発展加速を目指した。提案する Multiverse Notebook は、データ可視化や AI 構築において広く利用されている Jupyter Notebook に似たプログラミング環境であるが、Jupyter Notebook にはない「分岐」という拡張操作を備えている点に特徴がある。図2は、Multiverse Notebook のスクリーンショットである。図に示したように、コード片の系列は途中で「分岐」し、木構造を成している。

公開

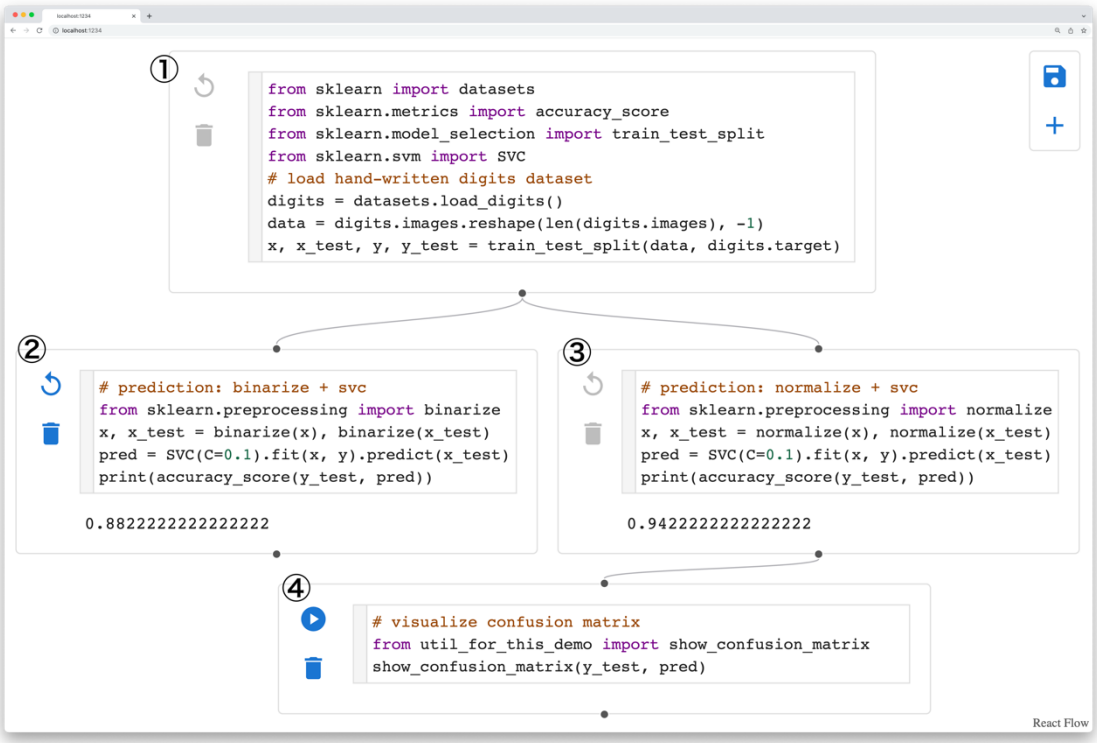


図 2. Multiverse Notebook のスクリーンショット

## 2. 研究成果

### (1)概要

提案ノートブック環境 Multiverse Notebook の実現に必要な2つの課題に取り組んだ。1つは「効率的な分岐を実現する言語処理系実装技術の開発」である。これは Multiverse Notebook のバックエンド部分のための技術開発である。そしてもう一方は「既存のノートブック環境でのセル操作体系の調査」である。これは、Multiverse Notebook のフロントエンド実装のための調査研究である。

### (2)詳細

**テーマ A「効率的な分岐を実現する言語処理系実装技術の開発」**

提案する分岐機能は、視覚的なセルの配置だけでなく、コード実行環境の分岐も行う。つまり、図2において②と③は完全に独立し、一方のコード実行が他方のコード実行には影響しない。このような実行環境の独立性は、試行錯誤において重要である。ある試行で行なった操作が別の試行に影響すると、誤った結果や再現が困難な結果を得ることにつながるからである。

上述したような実行環境の分岐を実現するには、各コード片の実行直後で言語処理系の状態のスナップショットを作成する必要がある。スナップショットを作成することで、Multiverse Notebook の利用者は、任意の時点で任意のセルの直後から新たな試行を開始することができるようになる。ノートブック利用者が記述できるコードを制限すれば、分岐の際に言語処理系の状態を巻き戻すという手法もあり得る。しかし本研究は現実のデータ可視化や AI 構築を支援することを想定しているため、そのような方針は採用しなかった。

公開

本研究で提案したのは、高速かつ空間効率の良い差分スナップショットの作成手法である。より具体的には、fork システムコールを利用したスナップショットの作成手法である。fork システムコールは、通常 Copy-on-Write 方式でのメモリ複製を採用している。これは、メモリへの書き込みが発生した場合にのみ該当部分（より具体的にはメモリページ）の複製を初めて行い、プロセス間のメモリ共有を促進、不必要な複製を抑える方式である。このような特徴を持つ fork の採用は、ノートブックの各セルにおいて言語処理系が抱えるデータの大部分を書き換えるという処理は珍しく、Copy-on-Write 方式が有効に機能すると予想したためである。

研究における主要な技術的貢献は、fork と参照カウント方式の Garbage Collection（GC）の相性の悪さを解消する新しいメモリ管理手法を提案した点である。参照カウント方式は、各オブジェクトが参照された際に、参照数を増減させてオブジェクトを生存させる必要があるか否かを判定する方式である。つまり、参照カウント方式では、高水準な視点ではデータの書き換えが起こっていないコードでも、低水準ではメモリ書き込みを引き起こす。しかしこの現象は、fork によるメモリ共有の促進を阻害する。高水準な視点では書き換えを行っていないコードでも低水準では書き込みが発生し、Copy-on-write が発火、結果的に本来必要ないメモリページの複製を発生させてしまうという相性の悪さがあった。提案したメモリ管理手法は、セルごとに特定のアドレス範囲にデータが確保されるようにメモリアロケータを改変し、さらにデータのアドレスに基づいて参照カウントの増減の対象か否かを判定する（部分的に Tenuring を行う）手法である。

提案したスナップショットの作成手法とメモリ管理手法の性能については、現実世界のノートブックから収集した試行錯誤の情報を用いて実験評価を行った。より具体的には、Kaggle という世界的に知られているデータサイエンス競技プラットフォーム上で公開されているノートブックから試行錯誤の情報を抽出し、実験評価を行った。この実験により、提案手法がデータ可視化や AI 構築のコードに対して大きな効果があることが実験的に確認された。より具体的には、過去にノートブック環境での利用が試みられていた他のスナップショット作成手法に比べて、数倍から数十倍の時間・空間効率の改善が確認された。

表 1 は fork を用いたスナップショットの効果を測定した実験の結果である。最左列は実験に用いたノートブックを示す識別子であり、それ以降の列はスナップショットなしの場合、既知のスナップショット手法を用いた場合、提案手法を用いた場合の実行時間（単位：秒）をである。表に示した通り、既存のスナップショット作成手法に比べると最大 21 倍の高速化が実現できていることが分かる。

| Notebook | No Checkpointing | Checkpointing by Dill | Checkpointing by Fork |
|---|---|---|---|
| allunia | $4.96 \times 10^1$ (-12%) | $1.17 \times 10^3$ (+1971%) | $5.66 \times 10^1$ |
| artgor | $7.98 \times 10^0$ (-18%) | $7.89 \times 10^1$ (+714%) | $9.69 \times 10^0$ |
| markpeng | $2.55 \times 10^3$ (+30%) | $2.12 \times 10^3$ (+8%) | $1.96 \times 10^3$ |
| namanj27 | $2.55 \times 10^2$ (+34%) | Error | $1.90 \times 10^2$ |
| reighns | $1.30 \times 10^2$ (+47%) | $1.40 \times 10^2$ (+58%) | $8.85 \times 10^1$ |
| robikscube-ieee | $1.22 \times 10^3$ (+17%) | Timeout (> +243%) | $1.05 \times 10^3$ |
| robikscube-covid | $2.23 \times 10^2$ (+1%) | $1.01 \times 10^3$ (+359%) | $2.21 \times 10^2$ |
| umutto | $1.85 \times 10^1$ (+89%) | $6.36 \times 10^1$ (+549%) | $9.80 \times 10^0$ |
| vbmokin | $2.70 \times 10^2$ (-3%) | $3.18 \times 10^2$ (+14%) | $2.78 \times 10^2$ |
| xhlulu | $2.41 \times 10^2$ (+8%) | $6.09 \times 10^2$ (+172%) | $2.24 \times 10^2$ |

表 1. 提案したスナップショット作成手法の性能

公開

表2は、提案したメモリ管理手法に焦点を当てて計測を行った実験の結果である。最左列は実験に用いたノートブックを示す識別子であり、それ以降の列は特殊なメモリ管理を行はない場合、循環参照 GC のみを停止させた場合、循環参照 GC と参照カウントの両方を停止させた場合の実行時間とメモリ消費量の変化である。最右列が提案手法であり、右から2番目は既知の手法である。表から分かる通り、提案手法は特殊なメモリ管理を一切行わない場合と比べて、最大で 57%のメモリ使用量削減を実現している。

| | No Tenuring | | Lightweight | | Aggressive | |
|---|---|---|---|---|---|---|
| | Time [s] | PSSs [MB] | Rel. Time | Rel. PSSs | Rel. Time | Rel. PSSs |
| allunia | $5.99 \times 10^1$ | $5.97 \times 10^3$ | -0% | -6% | -6% | -57% |
| artgor | $9.29 \times 10^0$ | $8.43 \times 10^2$ | -1% | -5% | +4% | -39% |
| markpeng | $1.91 \times 10^3$ | $1.03 \times 10^4$ | +0% | -6% | +2% | -5% |
| namanj27 | $1.85 \times 10^2$ | $1.17 \times 10^4$ | -0% | -0% | +3% | -1% |
| reighns | $8.46 \times 10^1$ | $2.91 \times 10^3$ | +0% | -10% | +5% | -26% |
| robikscube-ieee | $1.01 \times 10^3$ | $6.03 \times 10^4$ | -5% | -5% | +4% | -8% |
| robikscube-covid | $2.18 \times 10^2$ | $1.27 \times 10^4$ | -3% | -15% | +2% | -46% |
| umutto | $9.72 \times 10^0$ | $3.70 \times 10^3$ | -0% | +0% | +1% | -2% |
| vbmokin | $2.73 \times 10^2$ | $3.19 \times 10^3$ | +0% | -5% | +2% | -3% |
| xhlulu | $2.14 \times 10^2$ | $3.14 \times 10^4$ | -0% | -1% | +5% | -4% |
| Geo. mean | | | -1% | -5% | +2% | -22% |

表 2. 提案したメモリ管理手法の性能

**テーマ B「既存のノートブック環境でのセル操作体系の調査」**

提案ノートブック環境 Multiverse Notebook は、GitHub 上で Open-Source Software（OSS）として公開されている[1]。しかしながら、現在は必要最小限の操作（セルの追加、実行、削除など）しか提供しておらず、Jupyter Notebook が提供するような多様なセル操作を提供していない。Multiverse Notebook をより利用しやすいノートブック環境とし、社会へ広めていくためには、Jupyter Notebook が提供するような様々な操作への対応が必要になる。

しかし、Jupyter Notebook で提供されている操作には、不可解な挙動を示すものも多く存在する。最も顕著な例は、実行中のセルの移動である。最新の Jupyter Notebook v7 においても、実行中のセルを移動させると実行中である表示が消え、実行結果も出力されなくなる。しかし、背後にある言語処理系ではコードが実行され続けるという不可解な挙動を示す。このような不可解な挙動は、事前に行った簡単な調査においても容易に発見することができた。

このような不可解な操作群を単に模倣して Multiverse Notebook に実装することは、提案環境の利用者に悪影響である。そこで本研究では、Multiverse Notebook において一貫性あるセル操作体系を提供することを目指し、Jupyter Notebook のセル操作に関する調査を行った。この調査では単に各操作の挙動を確かめるだけでなく、対話的なプログラミング環境において重要なセル操作がどういう挙動を示すべきかを整理することにも取り組んだ。より具体的には、Jupyter Notebook v7 の各操作の挙動（意味）を定式化に取り組み、不可解である操作や再現性を毀損する操作について網羅的かつ体系的な分析を行った。これに加え、部分的にソースコードの解析にも取り組み、実装ミスと考えられる箇所の指摘にも取り組んだ。

[1] https://github.com/mvnb-research/mvnb

公開

3. 今後の展開

冒頭で述べた通り、本研究は、データ可視化や AI 構築などに取り組む際に必要不可欠な試行錯誤を効率化することを目指している。ここまでで研究成果として述べた通り、必要な（A）言語処理系実装技術の開発と（B）ユーザインターフェイスの実装のための調査は完了している。（A）については提案手法の実装が存在する一方、（B）に対応する実装は未着手である。

今後まず取り組むべきは、（B）の調査結果を参考にした Multiverse Notebook のフロントエンド（グラフィカルユーザインターフェイス）部分の実装の改善である。この作業には、最低でも1年程度の時間を要すると予想される。これは、（B）を参考にした追加実装だけでなく、公開中の Multiverse Notebook の利用を第三者へ促し、そこで得られたバグ報告やフィードバックを取り入れて実装を改善、より良いソフトウェアとして成長させる必要があるからである。但し、上述の期間は研究提案者本人が単独で作業を進めた場合にかかる時間である。そのため、何らかの方法で作業人員を増やすことができれば、期間短縮できることが期待される。

4. 自己評価

本研究の期間内において、実用的な技術の開発と実装を行えたことは、大きな成果であった。また、Jupyter Notebook に関する調査において、定式化などの手法に取り組んだことは、提案者の経験としては大きな意味があることであった。しかしながら、研究期間内に大きな学術的成果（国際学会での発表など）を残せなかった点は、今後の大きな課題である。研究成果として報告した内容は、期間終了後1年以内を目処に発表することを目指している。

研究の進め方（実施体制や研究費執行など）については、研究成果以上の大きな成果を得ることができたと強く感じている。より具体的に言えば、本プロジェクトをきっかけとした様々な研究者との議論や交流によって、自身の今後の方向性や指針を形成することができたと感じている。ここで得られた経験をもとに、今後より良い研究を行い、社会が直面する問題の解決に貢献していきたいと考えている。

本研究の社会への波及効果について明言することは難しいが、本研究が目指した試行錯誤における非効率の改善は、データを活用した価値創造の促進とそれを担う人材不足の緩和のために、プログラミングの研究者が解決しなければならない重要な課題であると現在も強く考えている。今後はこの研究での成果を起点として、より良い俯瞰的な視点から探索的プログラミング環境の実現に貢献していきたい。

5. 主な研究成果リスト

（1）代表的な論文（原著論文）発表

研究期間累積件数：1件

| |
|---|
| 1. 中丸智貴, 佐藤重幸. “ノートブックプログラミングにおける手戻りの調査と分析.” 第 25 回プログラミングおよびプログラミング言語ワークショップ（PPL 2023）. 2023 年 3 月. |
| 本研究は、テーマ A で述べた技術開発に向けた定量的な根拠と、開発技術の仮評価の結果をまとめた論文である。具体的には、Kaggle 上にあるノートブックの改訂過程で生じた手戻りを定量的に分析し、さらにその改訂過程を Multiverse Notebook で実行する際のオーバーヘッドを実験的に評価した。 |

公開

（2）特許出願

研究期間全出願件数：0 件（特許公開前のものも含む）

（3）その他の成果（主要な学会発表、受賞、著作物、プレスリリース等）

1. T. Nakamaru and S. Sato. “Multiverse Notebook: A Notebook Environment for Safe and Efficient Exploration.” Poster Abstract in Companion Proceedings of SPLASH 2022. December 2022.
2. 中丸智貴, 佐藤重幸. “セル単位の実行状態分離を備えたノートブックプログラミング環境.” 第 139 回プログラミング研究発表会 (PRO). 2022 年 6 月.