ニューラル数式ソルバーにおける途中結果の追跡と操作

松本悠太 [1]　Benjamin Heinzerling[2,1]　吉川将司 [1]　乾健太郎 [1,2]
[1] 東北大学　[2] 理化学研究所
yuta.matsumoto.q8@dc.tohoku.ac.jp　benjamin.heinzerling@riken.jp
yoshikawa@tohoku.ac.jp　kentaro.inui@tohoku.ac.jp

概要

言語モデルのより深い理解のためには、「モデル内部でどのような処理が行われているか」という観点も重要である。我々は単純な数式とその途中結果に着目することで、Transformer モデルが複数ステップに及ぶ処理を行っているかを検証する。途中結果の情報が符号化されている箇所を特定する Tracing と、符号化されている箇所の状態を操作してモデルに対して因果的介入を行う Manipulation の二つの実験を行なった結果、内部表現の特定の方向が線形に近い形で途中結果を符号化していること、そしてそのような方向がモデルの推論結果に対して因果的にも関係していることを示す。

www.github.com/cl-tohoku/trace-manipulate

1　はじめに

近年の言語モデルは数量推論のような複雑な処理が必要だと思われる問題においてもある程度高い性能を出せることが知られている [1]。また、複数の先行研究では線形代数や初等数学のような数値計算のタスクを通じて言語モデルが持つ潜在的な能力を測っており [2, 3]、現在のモデルによって解ける問題や解けない問題の性質が分かりつつある。

一方で、モデルがこのような問題を解く際、その内部で何が行われているかについての研究はほとんど行われていない。しかし、モデルの能力をよりよく理解するためには、入出力の結果だけでなくその内部での処理を分析することも非常に重要だと考えられる。我々は過去にモデルが途中結果を介した複数ステップの推論を行なっているかを調べるために、四則演算を学習した Transformer [4] の隠れ層を分析し、四則演算の途中結果[1)]がモデル内部の特定の方向に保存 (符号化) されていることを示した [5]。

本研究ではこの研究を拡張し、途中結果とモデル内部状態の相関関係だけでなく、**因果関係**も調査する。具体的には、四則演算を学習した Transformer の内部状態から主成分分析を用いて途中結果と相関の高い、すなわち途中結果を符号化しているような方向を抽出したのち、その部分のアクティベーションを操作してモデルの予測結果の変化を観察する。これによって、途中結果を符号化している部分がモデルの推論に与える因果的な影響を調査する。その結果、途中結果を符号化しているような方向の中には実際にモデルの推論時にも使用されているものがあることを確認できた。また、相関が高くてもモデルの推論には使用されていない方向も確認されたため、モデルの内部を分析する際に因果的な関係を調査することの重要性を示唆する結果となった。

本論文の貢献は、Transformer モデルの内部でどのような処理が行われているのかを因果推論まで含めて詳細に分析できた点である。

2　関連研究

言語モデルと数値　複数の先行研究は、近年の言語モデルが数学的な問題をある程度解けることを示している。Geva ら [1] は大規模言語モデルを使用することで、数量推論タスクでほぼ最先端の性能を達成できることを示した。また、数値計算に特化させた Transformer は初等数学や線形代数といった記号推論の問題をある程度解けることも示されている [2, 3]。彼らの知見に基づき本研究では、モデルに解ける問題として四則演算を取り上げる。

モデル内部の分析　モデルの内部表現にどのような情報が符号化されているかについては、近年高い関心が集まっている。このような研究の多くはプロービングの文脈で行われているが [6, 7]、Shibata ら [8] は我々と似た手法で実験を行っている。彼らは形式言語で訓練した言語モデルを分析して、形式言語の深さと高い相関を持つアクティベーションの

1)　例えば、(154 − 38) − (290 − 67) という数式中の途中結果は 154, 290, 116, 223 などである。

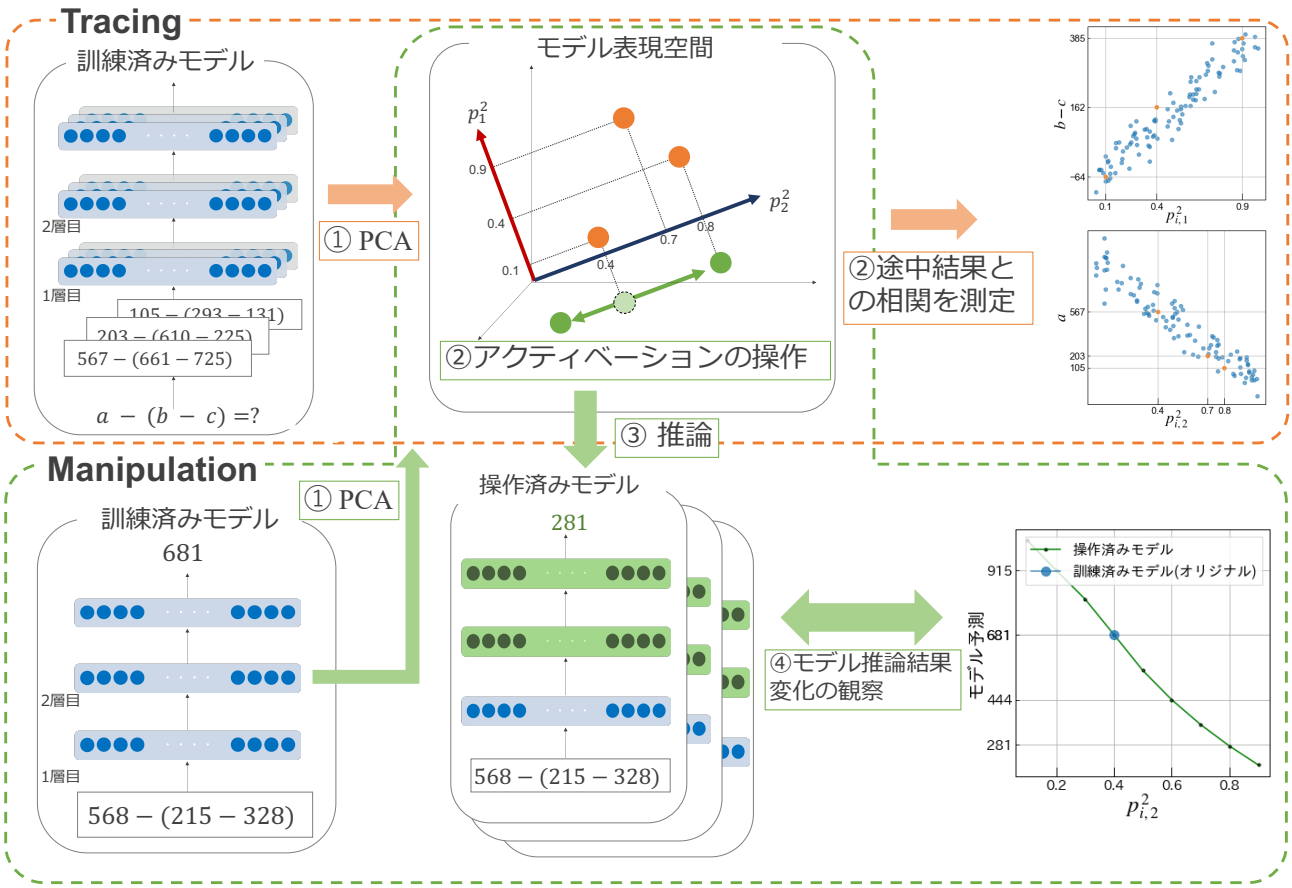


図 1: 途中結果の追跡 (Tracing) と操作 (Manipulation) の概要。Tracing では PCA を用いて、数式中の途中結果と相関が高いモデル内の方向を追跡する。Manipulation では、Tracing で追跡された方向に沿ってモデルのアクティベーションを操作した際のモデル予測の変化を観察する。

存在を示した。また、Elzar ら [9] はプロービングによって抽出できる特徴は必ずしもモデルの推論には使われていないことを明らかにし、内部表現への因果的介入を行う重要性を示した。本研究ではモデルの隠れ層を分析対象として、途中結果の情報を符号化している部分を抽出し、アクティベーションを連続的に変化させることで抽出された情報がモデルの推論に使用されているかを調査する。

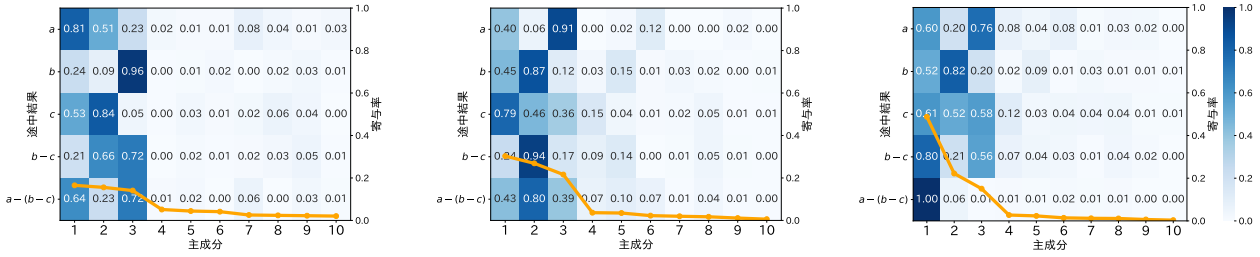
3 手法

本節では図 1 に示した本研究の提案手法について記述する。まず四則演算のデータで Transformer を訓練した後、下記に示す 2 つの方法でモデルの内部状態と数式の途中結果の関係を調査する。

3.1 途中結果の追跡 (Tracing)

数式 (入力) を変化させると、出力と同時に隠れ層の表現も変化する。では、この表現は数式中の途中結果とどのような関係があるのだろうか。汎化したモデルの表現空間はタスクの特徴を捉えたような構造になる [10] ことを考慮すると、数式中の途中結果は表現空間になんらかの形で符号化されているはずである。我々は一番単純な符号化の形として、表現空間中の特定の方向と途中結果の関係を調べる。まず、モデルの各層に対してその表現空間中で支配的な方向を見つける。モデルの l 層目, j 番目のトークンにおける表現 h_j^l を全て連結した表現 $H^l = h_1^l \oplus h_2^l \oplus \cdots \oplus h_n^l$ に対して主成分分析 (PCA) を行うことで、主成分 $p_k^l, k \in [1, ..., K]$ を得ることができる。n は入力列の長さを表す[2)]。データセット中のあるインスタンス i に対してこの PCA モデルを適用すると、各主成分の重み $p_{i,k}^l$ を得ることができる。次に、数式中の途中結果 R_i^j と各主成分 p_k^l の重みの相関 $\mathbf{corr}(R_i^j, p_{i,k}^l)$ を測る。これを各層における全ての途中結果と主成分の直積に対して行

2) 本研究では $K = 10, n = 50$ とした。



(a) 1 層目　(b) 3 層目　(c) 4 層目

図 2: 層ごとの各主成分重みと数式の途中結果の相関関係のヒートマップ。各セルは第 k 主成分の重み（列）とある途中結果（行）との相関係数の絶対値を表す。

うことで、ある途中結果と最も相関の高い主成分である**最大相関方向** $\hat{p}^l_k(R^j) := \mathbf{argmax}_k(\mathbf{corr}(R^j_i, p^l_{i,k}))$ を得る。仮に最大相関方向の相関が十分高ければ、対応する途中結果を符号化していると考えることができる。例えば、図 1 のイメージは $a-(b-c)$ という数式における途中結果 a や $b-c$ の値がある層のある主成分に符号化されていることを表す。

3.2 途中結果の操作 (Manipulation)

本節では、Tracing によって得られた最大相関方向がモデルの推論に使用されているのかを検証するため、モデルに対して因果的介入を行う。具体的には図 1 下部に示すように、主成分に沿ってモデルのアクティベーションを操作し、それによるモデルの予測結果の変化を観察する。定式化して表すと、l 層目の表現 H^l に対して次の式に基づいて主成分 p^l_k の重みを r 倍することで、操作後の表現 $H^{l'}$ を得る。

$$H^{l'} \leftarrow H^l + (r-1)\left(p^{l\top}_k H^l\right) p^l_k \tag{1}$$

直観的には r を変化させることで主成分 p^l_k に沿ってアクティベーション H^l が移動し、それに伴ってモデルの予測結果も変化する。

もし最大相関方向 $\hat{p}^l_k(R^j)$ が因果的にも途中結果 R^j を符号化しているのであれば、p^l_k の重みを動かすことは数式の途中結果 R^j を動かすことに対応していると考えられる。すなわち、$R^j_i = f(p^l_{i,k})$ となるような関数 f を定義できる。この時、逆関数を取ることで途中結果から主成分の重みを予測する関数 $f_p = f^{-1}$ を定義することができる。「$\hat{p}^l_k(R^j)$ が因果的にも途中結果 R^j を符号化している」という仮説を検証するためには、この f_p が正しいかを確かめれば良い。すなわち、実際に入力項を変更した際の最大相関方向 $\hat{p}^l_k(R^j)$ の重みを $p^l_{i,k} = f_a(R^j_i)$ のように表せる実測関数 f_a と f_p を比較して確かめる。

4 実験設定

ニューラル数式ソルバーとして、Sajjad ら [11] を参考に 6 層の Transformer エンコーダを用意し、[CLS] トークンから線形回帰を行う。足し算と引き算、括弧を含む四則演算のデータをテンプレートから 20 万個自動生成し、[3] そのうち 19 万個のデータでモデルを学習する。先行研究 [1] に従い、入力中の数字は 1 桁ごとにトークン化する。例えば、“123” という数字は 1, ##2, ##3 にトークン化される。訓練後、1 万個の検証データにおいてこのモデルを評価したところ決定係数 $R^2 = 0.9988$ で、異なるパターンの数式を並列に学習させてもほぼ正しい答えを出力できることが確認された。

5 結果

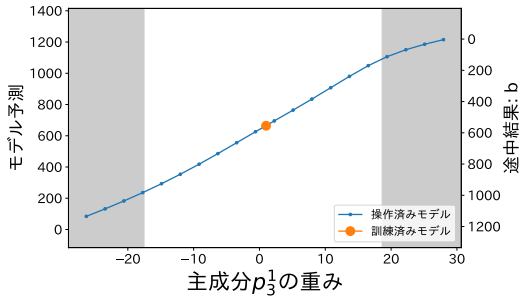
5.1 途中結果の追跡

今回は $a-(b-c)$ という数式を分析の対象とする。各途中結果と上位 10 主成分の重みの関係を測定した結果を図 2 に示す。図 2(a), 2(b) から、中間層において数式中の途中結果と相関が非常に高い最大相関方向が複数存在することがわかる。例えば、1 層目の第 3 主成分の重みと b の値の相関係数は 0.96、3 層目の第 2 主成分の重みと $b-c$ の値の相関係数は 0.94 である。

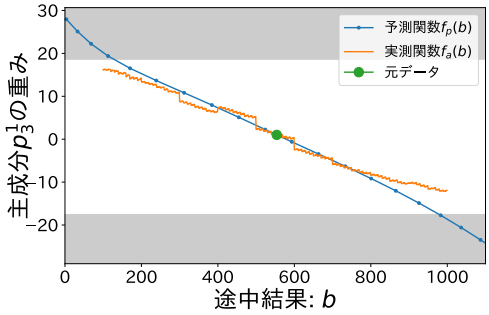
ここから、数式の途中結果はモデル内部の表現空間における特定の方向に線形的、局所的に符号化されていると考えることができる。

一方で図 2(c) からは p^4_1 が最終結果とほぼ 1 の相関を持っていることがわかり、4 層目の時点で数値計算処理が終了していると推察できる。

3) $(154-38)-(290-67)$, $(621-(821-430))-131$, など。方程式の種類は全部で 87797 種類である。



(a) 重み操作によるモデル予測とそこから導いた途中結果の変化。

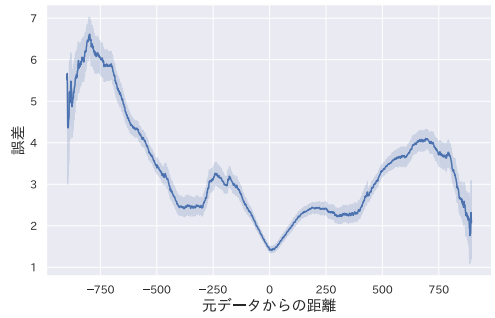


(b) 途中結果と主成分の重みに関する実測関数 $f_a(b)$ と予測関数 $f_p(b)$。

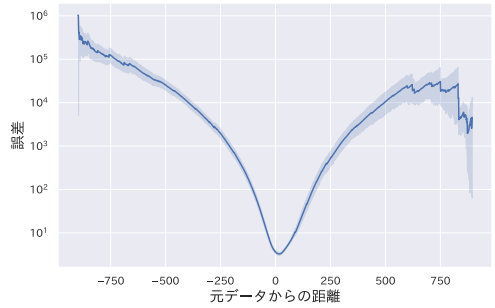
図 3: $\hat{p}_3^1(b)$ に対する Manipulation の結果。影の部分はデータセット中に現れない重みの範囲を表す。

5.2 途中結果の操作

数式 617 – (555 – 602) とその途中結果 $b = 555$ を例に取り、Manipulation を行う。5.1 節から、数式 $a - (b - c)$ において、最大相関方向 $\hat{p}_3^1(b)$ は 0.96 という高い相関を持つ。ここで、p_3^1 に沿ってモデルのアクティベーションを操作すると、モデルの予測結果は図 3(a) に示されるように、元の予測結果 (664) から約 100–1200 まで変化することがわかる。この時、予測結果の変化をモデル内の途中結果 b の表現を変えたものによるものだと仮定すると、図 3(a) の右軸を得ることができ、軸を反転することで $p_{i,3}^1 = f_p(b)$ となるような関数 f_p が得られる。この予測関数 f_p を、実際に b の値を変化させた時の重みを $p_{i,3}^1 = f_a(b)$ と表せるような実測関数 f_a と比較した結果を図 3(b) に示す。図 3(b) から、予測関数 f_p と実測関数 f_a は概して一致しており、特に元データの周りでは一致度が高いことがわかる。これは、$\hat{p}_3^1(b)$ が b の値を一定の範囲では因果的にも符号化している証拠だと考えることができる。同じ実験を異なる値を持つ $a - (b - c)$ の 1000 インスタンスで行った時の $|f_p(b) - f_a(b)|$ の中央値を図 4(a) に示す。図 4(a) から、元データの表現から離れると必ずしも $\hat{p}_3^1(b)$ が b の値を因果的に符号化しているとは言えないことが定量的にも観察される。逆に、Manipulation によって因果的には途中結果を符号化していない最大相関方向を特定することもできる。例えば、$\hat{p}_2^4(b)$ は b と 0.81 の高い相関を持つが、この方向に対して重みの操作を行なった際の誤差中央値は図 4(b) のようになり、誤差が非常に大きいことが観測される。これは図 2 から推測される「4 層目までで途中結果を使用した数値計算処理はほぼ完了している」という仮説と対応するものである。



(a) $\hat{p}_3^1(b)$ に対する Manipulation の結果。



(b) $\hat{p}_2^4(b)$ に対する Manipulation の結果。

図 4: 1000 個のデータに対して重みの操作を行なった際の誤差 $|f_p(b) - f_a(b)|$ の中央値。

6 おわりに

本研究では言語モデルが数値計算を行う時に内部で行なっている処理を明らかにするため、数式中の途中結果に着目し追跡と操作を行なった。その結果、「数値計算ができるモデル」の内部では途中結果の情報が存在し、かつその情報はモデルの推論時にも使われていることを解明した。本研究の手法はアクティベーションを連続的に変化させることでモデルに対して詳細な因果推論を行うことができる。今後はこの手法をより一般的な数量推論モデルにも適用し、分析道具としての有用性を示したい。

謝辞

本研究は JST CREST JPMJCR20D2、JSPS 科研費 21K17814 の助成を受けたものです。

参考文献

[1] Mor Geva, Ankit Gupta, and Jonathan Berant. Injecting numerical reasoning skills into language models. In **Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics**, pp. 946–958, Online, July 2020. Association for Computational Linguistics.

[2] David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. Analysing mathematical reasoning abilities of neural models. In **International Conference on Learning Representations**, 2019.

[3] François Charton. Linear algebra with transformers, 2021.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, L ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, **Advances in Neural Information Processing Systems**, Vol. 30. Curran Associates, Inc., 2017.

[5] 松本悠太, 吉川将司, Benjamin Heinzerling, 乾健太郎. 四則演算を用いた transformer の再帰的構造把握能力の調査. 言語処理学会年次大会, 2022.

[6] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does BERT learn about the structure of language? In **Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics**, pp. 3651–3657, Florence, Italy, July 2019. Association for Computational Linguistics.

[7] Alessandro Raganato and Jörg Tiedemann. An analysis of encoder representations in transformer-based machine translation. In **Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP**, pp. 287–297, Brussels, Belgium, November 2018. Association for Computational Linguistics.

[8] Chihiro Shibata, Kei Uchiumi, and Daichi Mochihashi. How LSTM encodes syntax: Exploring context vectors and semi-quantization on natural text. In **Proceedings of the 28th International Conference on Computational Linguistics**, pp. 4033–4043, Barcelona, Spain (Online), December 2020. International Committee on Computational Linguistics.

[9] Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral explanation with amnesic counterfactuals. **Transactions of the Association for Computational Linguistics**, Vol. 9, pp. 160–175, 2021.

[10] Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, **Advances in Neural Information Processing Systems**, 2022.

[11] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. On the effect of dropping layers of pre-trained transformer models, 2021.

A　四則演算データセットの詳細

四則演算データは、最大で 5 段の深さを持つ。各項 x の大きさは、訓練、検証データ中では $0 < x < 1000$、5.1, 5.2 節での分析に使用したデータでは桁数を揃えるために $100 <= x < 1000$ とした。訓練に使用した数式の例を表 1 に示す。

表 1: 四則演算データセットの例

数式	値	深さ
426	426	1
(746 – (894 – 355))	207	3
(355 + (((534 + 384) – 678) + 314))	909	5

B　ニューラル数式ソルバーの詳細

ニューラル数式ソルバーは表 2 に示す設定で訓練を行った。

表 2: ハイパーパラメータの一覧

設定	値
モデルアーキテクチャ	Transformer (エンコーダのみ)
層	6
初期値	bert-base-uncased[4)]
最適化手法	BertAdam[5)]
バッチサイズ	64
学習率	5.0×10^{-5}
訓練エポック	20
ドロップアウト	0.4
最大トークン数	100

4) https://huggingface.co/bert-base-uncased
5) https://pypi.org/project/pytorch-pretrained-bert