toy model到AI數學討論：自動證明的模型化

1. 標題

- 課程： 機器學習
- 作業： 期末專案（第10次作業的延伸）
- 主題： AI數學定理自動證明
- 作者： 潘煒翔(112950009)

2. 摘要

本報告旨在探討AI在數學定理自動證明領域的研究路徑，主要想像**20年後AI能夠作為合作者，在證明輔助系統（如Lean）中自動完成並發現新數學證明**。

詳細闡述了一個基於**監督學習**的「證明步驟預測器」toy model，該模型在一個受限的**代數表達式**簡化任務中，學習從當前證明狀態預測下一步應應用的推理規則

此模型為理解將證明過程建模為搜尋問題或馬可夫決策過程，並處理更複雜系統（如結合強化學習與神經符號方法）

3. 問題設計

我的最終目標是開發能在**證明輔助系統中自動完成證明的AI**，簡化問題於自動證明的環節：根據當前證明狀態，選擇下一步的推理規則，這對應到真實證明環境中，給定目前的證明狀態（代數表達式），選擇下一步的推理規則，模型需要從數十種tactic（如rewrite、apply、intro）中做出選擇。

- 定義：
    - 狀態空間 S：所有合法的代數表達式（以樹狀結構與字串表示）
    - 行動空間 A：四條推理規則的編號 \({0,1,2,3\}\)
    - 轉移函數 $\mathrm{P}(\mathrm{s}' \mid \mathrm{s}, \mathrm{a})$：在 state s 上套用規則 a 所得到的新表達式 s'（在 toy model 中，轉移是 deterministic 的）
    - 把證明簡化過程視為一條條 **Markov 軌跡**：

$$(\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_1), (\mathrm{s}_1, \mathrm{a}_1, \mathrm{s}_2), \ldots, (\mathrm{s}_\mathrm{T}, \mathrm{a}_\mathrm{T}, \mathrm{s}_{\mathrm{T}+1})$$

- s_t：第 t 步的表達式字串（例如 $(x + 0) + 1$、$0 + (x + 1)$）
- a_t：在 s_t 上實際套用的規則編號0–3
- s_{t+1}：套用規則之後得到的新表達式

資料形式:

1. 先用手寫的「簡化器」對隨機生成的表達式做多步簡化，
   為每個起始式產生一條長度至多為 K 的軌跡：

$$(s_0, a_0, s_1), (s_1, a_1, s_2), \ldots$$

2. 將所有軌跡展平，得到一個 triple 集合：

$$D = \{(s_t, a_t, s_{t+1})\}_t$$

3. 之後在訓練 policy 時，使用 (s_t, a_t) 當作 **supervised learning** 的訓練樣本，而 s_{t+1} 保留於「馬可夫模式推演」

4. 換句話說，資料不再只是獨立的「(表達式字串, 規則標籤)」pair，而是滿足馬可夫性質的序列資料，可以同時討論 **MLP policy** 與 **MDP 上的前向推演（rollout）**。

4. 模型與方法

4.1 選擇模型：雙層全連接神經網路（MLP）

理由：

1. 結構簡單：
   MLP能學習輸入特徵（表達式編碼）與輸出類別（規則）之間的非線性映射，適合這個多分類問題

2. 可解釋性相對好：
   與更複雜的Transformer或RNN相比，MLP的參數和行為更容易分析

3. 計算效率高：
   適合作為toy model的起點

4. 輸入特徵固定：
   我們將變長字串編碼為固定維度向量，MLP能很好地處理這種輸入

模型架構：

$$\text{輸入層}\ (\mathrm{R}^{\mathrm{d}})\ \rightarrow\ \text{隱藏層}\ (64\ \text{維},\ \mathrm{ReLU})\ \rightarrow\ \text{輸出層}\ (4\ \text{維},\ \mathrm{Softmax})$$

其中 $\mathrm{d} = \max\_\mathrm{len} \times |\mathrm{vocab}|$， vocab 只包含 $\{\mathrm{x}, 0, 1, +, (,)\}$。

4.2 監督式學習：從 $(\mathrm{s}_t, \mathrm{a}_t)$ 學 π_θ

訓練階段，我將從 **Markov** 軌跡中抽取的樣本$(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1})$ 視為 **supervised dataset**:

- 輸入 : state 字串 s_t 經過 one-hot 展平得到的向量 $\mathrm{v}_t \in \mathrm{R}^{\mathrm{d}}$
- 實作 : 損失函數採用 cross-entropy :

$$\mathrm{L}(\theta) = -\sum_{\mathrm{t}} \log \pi_\theta(\mathrm{a}_t \mid \mathrm{s}_t)$$

- Adam optimizer 最小化 : L，訓練若干 **epoch** 後，可以在訓練集與測試集上取得接近 90% 以上 的準確率

4.3 馬可夫模式推演：用 policy 做 forward rollout

在模型訓練好之後，我將它當成一個 policy，在 **MDP** 上進行「馬可夫式的前向模擬」：

1. 從隨機起始表達式 s_0 出發；
2. 在每一步 t：
    - 根據目前的 state s_t，用 MLP 計算 $\pi_\theta(\mathrm{a} \mid \mathrm{s}_t)$
    - 取 $\mathrm{a}_t = \arg\max_{\mathrm{a}} \pi_\theta(\mathrm{a} \mid \mathrm{s}_t)$（隨機從分布中抽樣）
    - 套用 transition 函數 $\mathrm{s}_{t+1} = \mathrm{f}(\mathrm{s}_t, \mathrm{a}_t)$，得到新表達式

符合馬可夫性質：

$$\mathrm{P}(\mathrm{s}_{t+1} \mid \mathrm{s}_0, \mathrm{a}_0, \ldots, \mathrm{s}_t, \mathrm{a}_t) = \mathrm{P}(\mathrm{s}_{t+1} \mid \mathrm{s}_t, \mathrm{a}_t)$$

每個下一步只依賴當前的狀態與選擇的規則，透過這樣的 rollout ，可以直觀地展示「MLP policy + Markov 過程

5. 實作與結果

5.1 分類準確率

在約 3,000 條隨機生成的 Markov 軌跡上，取得 7,250 筆$(\mathrm{s}_t, \mathrm{a}_t, \mathrm{s}_{t+1})$ 樣本，並以 8:2 切分為訓練與測試集，分別約為 5,800 與 1,450 筆

使用**MLP**（輸入為 174 維的 one-hot 向量，輸出為 4 類規則 ID）
進行 20 個 epoch 的訓練，訓練 log ：

- train_acc : 0.52 上升至約 0.66

- test_acc : 約 0.52 上升後穩定在 0.55–0.56

1. 相較於隨機猜測四條規則（理論上約 0.25），模型明顯有學到結構性的資訊
2. 同一個 state 可能存在多個合法的規則應用位置，在資料生成時是「隨機從可用規則中選一條作為 label」，造成 state → action 的 mapping 並非單一，帶有相當程度的 label noise
3. 在這樣的設定下，MLP 的表現被壓在約 0.6 左右的自然上限

5.2 Markov 模式推演

在訓練完成後，將學到的 MLP 視為 policy $\hat{\pi}_\theta(a \mid s)$，在同一個 MDP 上進行前向推演

從隨機起始 state 出發，每步根據 $\hat{\pi}_\theta$ 選取機率最高的規則 ID，再透過 deterministic transition $s_{t+1} = f(s_t, a_t)$ 得到新 state。

> 觀察到的行為包括(by 數個AI:chatgpt,deepseek,gemini得到):
>
> - 對於早期就沒有合法規則可套用的簡單 state，例如 `x` 或 `1`，
> 因為資料集中從未出現「在這些 state 上應該採取哪個 action」的樣本，
> MLP 仍會輸出某個規則 ID，但實際上在該 state 上不可用，
> 因此 rollout 立即終止。
> 這說明目前的模型只被訓練在「有規則可用」的局部空間，
> 並未學到「何時應該停下來」的終止條件。
>
> - 對於較複雜的表達式，例如 `((x+(x+x))+0)`，
> 模型會頻繁選擇結合律相關的規則（rule 2 / 3），
> 在不同括號結構之間來回切換；
> 偶爾也會選擇加零規則（rule 0），但不一定套在真正的 `+0` 上。
> 由於原始 "oracle" policy 在資料生成時就是隨機選擇所有可用規則之一，
> 整體行為更像是「模仿一個隨機 re-bracketing 的決策者」，而非穩定朝向最簡型前進。
>
> 這些現象一方面說明了：
>
> - **MLP policy 確實學到某些局部結構的偏好**（例如優先使用結合律調整括號）；
>
> 但另一方面也凸顯：
>
> - 在沒有 reward / 目標導向設計的情況下，
> 單純用 supervised learning 模仿隨機策略，
> 很難得到一個會自動導向「化簡」目標的全域規劃能力。
>
> 因此，這個 toy model 的結果既展示了
> 「如何在 MDP 設定下訓練與使用一個 MLP policy」，

> 也自然引出下一步研究的方向：
> 需要引入更有結構的 state 表示（例如 AST / GNN）、
> 更合理的目標策略設計，
> 甚至結合 reward 與強化學習，才能逼近真正的「自動證明」系統。

5.3 實做紀錄

> 紀錄在code裡

6.討論

6.1 toy model

1. supervised policy 能力有限

這在學一個將 state 過去最常被選的 rule 對應的 mapping，沒有reward，所以 policy 會模仿「隨機 re-write 的習慣」

2. 資料來源

同一個 state 可能有多個合法 rule，資料生成時又是「隨機挑一個當 label」，在同一個 s_t 在 dataset 裡有不同的 a_t，產生 label noise

3. state 的表粗糙

我用的是 one-hot，較為扁平，沒有把「哪裡有 +0」、「哪一側是子樹」標出來，即使 MLP 有學到一些偏好（像是常用結合律），它其實看不到真正的代數結構

4. 標準策略

要做 theorem proving，就需要好的自動策略或人類示範，或用 RL 去逐步逼近成功導向的分佈，而不是模仿 noisy behavior

6.2 參考資料

1. Trinh, T. H., Luong, T. et al. (2024). “AlphaGeometry: An Olympiad-level AI system for geometry.” *Nature*.
2. Polu, S., & Sutskever, I. (2020). “Generative language modeling for automated theorem proving.” arXiv preprint arXiv:2009.03393.
3. Yang, K., Swope, A. M., Gu, A., Chalamala, R., Song, P., Yu, S., ... Anandkumar, A. (2023). “LeanDojo: Theorem Proving with Retrieval-Augmented Language Models.” arXiv preprint arXiv:2306.15626.

4. Yin, D. S., & Gao, J. (2025). “Generating Millions of Lean Theorems With Proofs By Exploring State Transition Graphs.” arXiv preprint arXiv:2503.04772.
5. Liu, C., Shen, J., Xin, H., Liu, Z., Yuan, Y., Wang, H., … & Qiu, L. (2023). “FIMO: A Challenge Formal Dataset for Automated Theorem Proving.” arXiv preprint arXiv:2309.04295.
6. Li, Z., Sun, J., Murphy, L., Su, Q., Li, Z., Zhang, X., Yang, K., & Si, X. (2024). “A Survey on Deep Learning for Theorem Proving.” *COLM 2024*. (via DL4TP)
`