大規模言語モデルと幾何制約チェックを活用した言語入力から図形描画を行う幾何作図ソフトウェアの提案

明治大学大学院・先端数理科学研究科　齋藤　裕樹 (Hiroki Saito)
Graduate School of Advanced Mathematical Sciences, Meiji University
明治大学・総合数理学部　阿原　一志 (Kazushi Ahara)
School of Interdisciplinary Mathematical Sciences, Meiji University

1　はじめに

数学教育を目的にしたソフトウエアのなかでも、幾何作図ソフトウェアは一定の評価を得ており、GeoGebra[1] のように世界規模で利用されているソフトウエアもある。幾何作図ソフトウエアを用いた、学習者による図形の表示や操作体験は数学教育の観点から見て効果が評価されている。このような幾何作図ソフトの一つの可能性として、平面幾何の図に関する文章や言語情報をもとに、ソフトウエア上で図形を描画することがテクノロジーとして可能であるかという問題について考えてみた。この問題を 3 つに分割してみよう。第 1 に、文章は自然言語で書かれることとし、その中から図形の構造、頂点などの名称、図形のもつ幾何的性質、構成要素間の関係性などの情報を把握する必要がある。第 2 に、そうして得られた幾何的な状況や関係性を列挙できたときに、実際に図を描画することができるかどうか、というのは状況把握とは別のテクノロジーが必要である。第 3 に、平面幾何の図をソフトウエアが描画できたとして、それを定規とコンパス（もしくは定規とコンパスに類するような、幾何ソフトの機能）を使って作図する手順を生成するアルゴリズムはありうるのだろうか。この問題の第 1 の部分に対応するための方法の一つとして、第 1 著者は、平面幾何の図について日本語で説明した文章に対して、大規模言語モデルを用いて、これを幾何の要素や関係性の羅列のレベルまで書き下すことが可能であるかどうかという実験を試みた。ここで、要素や関係性の羅列は XML で出力するものとし、阿原研で開発中のソフトウエアである PointLine[2] による図の構築に対応できるようにした。この試みを通して、平面幾何の図を説明した文章から、PointLine 上の図形描画を生成する道筋が示せたものと考えている。

2　研究概要

2.1　提案システムについて

本研究では、図 1 のように言語情報の入力から PointLine 上への図形の表示を一貫して行うシステムを提案する。このシステムは主に言語情報から決められた形式の構造化データを LLM(大規模言語モデル) によって出力する段階とその構造化データに矛盾がないかを判定し、矛盾がなければ表示する図形の位置を決める段階の 2 つに分けられる。

1 つ目の段階で用いる LLM は深層学習技術によって構築されるモデルである。リクエストとなる文章を入力すると、その応答として適切と思われる文章を推論する。そして、その LLM の一部のパラメータを再学習して、その学習内容に特化したモデルを構築するものとして、ファインチューニングと呼ばれる手法がある。本研究ではそのファインチューニングを用いて、図形に関する構造化データを文章から出力することに特化したモデルを構築する。学習に用いるデータセットは幾何学的な用語や言い回し、それに対する構造化データの記述テクニックに関して学習させることを目標に準備した。

2 つめの段階では LLM により出力された構造化データが正しく出力されているかを確認する処理を行う。具体的には図形間での参照情報が正しいか、形式通りの入力が与えられているかどうか、対象のデータが幾何学的に描画可能かどうかなどの確認項目がある。幾何学的に矛盾があるかの確認は非線形最適化による近似解の有無によって判定する。もし、矛盾がないと判定された場合、最適化により得られた解をもとに図形の位置を決め画面上に表示する。

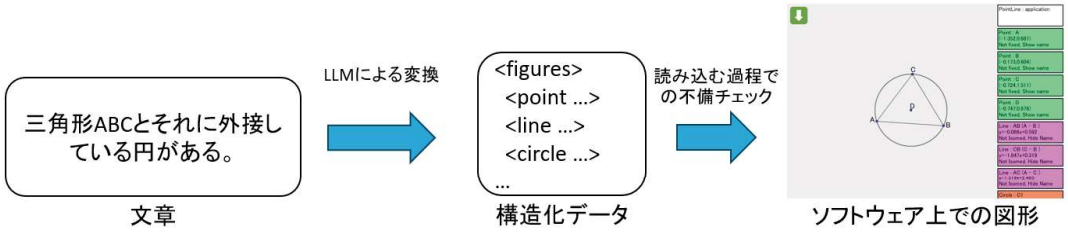

図 1: 言語情報から図形表示までの流れ

2.2 構造化データ

構造化データは PointLine のソフトウェア上で読み込める XML 形式のものを用いる。各要素はそれぞれオブジェクトとモジュールの 2 つに分けることができる。オブジェクトは図形（点や線など）、モジュールは図形間の関係性（接する、平行など）を表す要素である。オブジェクトは必ず一意に決まる id を持ち、モジュールはそのオブジェクトの id を参照する形で構成される。これにより、図形間の関係性を表現することができる。また、モジュールはオブジェクトの id を参照する形で構成されるため、モジュールの参照先が存在しない場合は不適切な形式として検出される。本システムで使用するオブジェクトとモジュールは表 1 のとおりである。

表 1: オブジェクトとモジュール一覧

	要素名	概要	必要な情報
オブジェクト	point	点	なし
	line-segment	線分	点の id × 2
	circle	円	点の id(中心点), 半径
	angle	角度	点の id × 3
モジュール	middle-point	中点	点の id × 3
	point-on-line	線上の点	点の id, 線分の id
	point-on-circle	円周上の点	点の id, 円の id
	line-tangent-circle	円に接する線	線分の id, 円の id
	circle-tangent-circle	2 つの円が接する	円の id × 2
	vertical	垂直	線分の id × 2
	parallel	平行	線分の id × 2
	isometry	長さが等しい	線分の id × 2
	bisector	角度が等しい	角度の id × 2

3　研究の目的

本研究では、様々な学習手法や学習データの構築を試行錯誤することで、言語情報から図形の表示がどの程度可能になるかを調査することが目的である。また、構築したモデルでの推論において、どのような文章を入力として与えれば、適切に図形を出力できるかについても調べる。これらの調査結果を基に、LLM が適切な結果を返せるようになるための条件を明らかにし、それに必要な学習手法やデータセット、学習時の要点などを示す。

4　大規模言語モデルを用いた推論

4.1　ファインチューニングモデルによる推論

本システムで用いる LLM(大規模言語モデル) は作図手順を推論するのではなく、描画したい図形とその関連性について推論を行うことを想定している。例えば、内接円を作図することを考えた場合、通常の作図方法であれば、三角形の各頂点から二等分線を引き、それらの直線の交点 (内接円の中心点) を求める。さらに辺に対して垂直二等分線を引くことによって内接円の半径がわかり、これをもって内接円の作図ができたことになる。LLM にこの作図手順の一つ一つの意味を学習させ、内接円やその他の図形の作図に応用させるのは非常に難しいだろう。しかし、PointLine では作図手順による入力ではなく、描画したい図形とその関連性の情報のみから描画することができる [3]。このことから本研究で用いる LLM には作図手順を学習させずに、用語の定義や数学的な表現を学習すれば十分であると考えられる。

本システムで用いられるような数学的な知識を学習させるためには、LLM がこれらを追加で学習してその領域に特化させることが望ましい。そこで、LLM による推論は OpenAI が提供する GPT-4o[4] のファインチューニングモデル API を利用する。具体的な流れは図 2 のようになっており、独自で作成したデータセットでファインチューニングをし、そのモデルを用いて推論を行う。

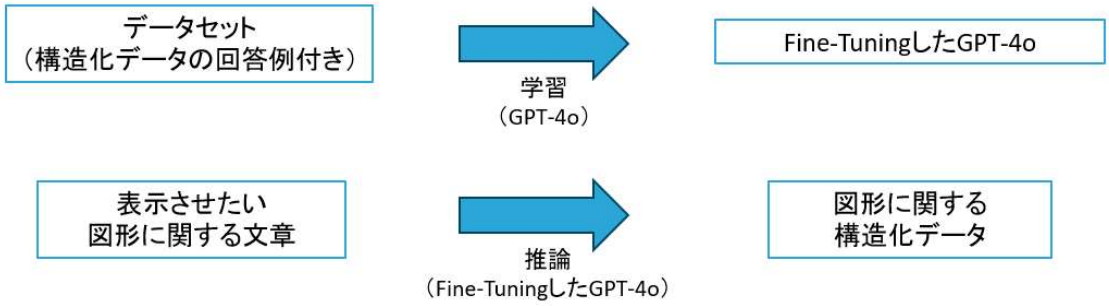

図 2: ファインチューニングによる推論

4.2 データセットの作成

LLM は事前に用意したデータセットを用いて、構造化データを文章から適切に出力出来るようにファインチューニングを行う。これにより、すでに学習された LLM からさらに図形推論に特化したモデルへと再学習でき、より正確な図形描画を行うことができる。データセットは幾何学的な状況の説明文とその状況に対応する構造化データをペアとして複数用意する。本研究で用いたデータセットは主に以下の点に注意して作成した。

- 幾何学的な用語 (交わる、接する、平行など) や特徴的な図形の名前（平行四辺形、正五角形、内接円など）を含む文章を用いる。専門的な用語に関しては説明文の後に補足として、その用語の意味を記述する。これにより LLM が専門用語を学習して、より適切な結果を返せるようになることが期待できる。
- 同じ説明文でも別の言い回しや少し特殊な表現、記号を用いた文章を用意する。これにより、LLM が多様な言い回しや同義語を学習することができる。
- 説明文の長いものや全ての要素が多くなるようにデータセットを用意する。LLM が多くの情報量を正確に構造化データに出力できるようにすることが目的である。

本研究で用いたデータセットは以下のリンクで公開している。
(https://github.com/saityyy/PointLine-LLM.git)

5　非線形最適化による幾何制約チェック

LLM による出力は必ずしも形式通りの構造化データになっているとは限らない。そのため、本システムには構造化データが形式通りに構成されているかと幾何学的に矛盾がないかどうかを確認する機能が必要である。前者は主に id の参照エラーや定義されていない要素や属性の使用などがあり、これは PointLine が構造化データを読み込むときに検出する仕組みになっている。後者においては、本研究ではいくつかの非線形方程式を満たしながら与えられた変数の近似解の探索を行う非線形最適化をし、その解の有無で幾何制約チェックを行う手法をとる。また、もし解がある場合にはその解をそのまま点の座標や円の半径として設定し、画面上に図形を表示する。幾何制約チェックや座標決めは PointLine でも可能であるが、本研究では、非線形最適化でも PointLine と同様に幾何制約チェックと座標決めを適切に行えるかを調査する。非線形最適化は Python の SciPy ライブラリ [5] の minimize 関数を用い、使用するアルゴリズムはいくつかのケースを試して一番うまく判定できた SLSQP[6] を用いた。求めるべき解を点の座標、図形の関連性をそれらの点で構成された方程式と考える。各変数の初期値は最初にランダムに決められ、そこから目的関数が最小になるように、つまりはモジュールの制約が満たされているような解を探索していく。幾何学的に矛盾があるケースはないケースと比べて目的関数が期待通りに最小化されないため、その間にしきい値を設定することによって矛盾があるかどうかを判定する。表 2 は各モジュールそれぞれが満たすべき方程式の一覧である。与えられる点の座標や円の半径などをもとに方程式に必要なパラメータ (線の傾き、切片、ある始点と終点からなるベクトルなど) を求め、方程式を立てる。

表 2: 各モジュールでの満たすべき方程式

要素名	満たすべき方程式
middle-point	$(x_1 + x_2) - 2x_3 = 0$ $(y_1 + y_2) - 2y_3 = 0$
point-on-line	$y_1 - (ax_1 + b) = 0$
point-on-circle	$(x_1 - c_x)^2 + (y_1 - c_y) - r^2 = 0$
line-tangent-circle	$\frac{\lvert ac_x - c_y + b\rvert}{\sqrt{a^2+1}} - r = 0$
circle-tangent-circle	$\sqrt{(p_1 - p_2)^2 + (q_1 - q_2)^2} - (r_1 + r_2) = 0$
vertical	$a_1 a_2 + 1 = 0$
parallel	$a_1 - a_2 = 0$
isometry	$(x_1 - x_2)^2 + (y_1 - y_2)^2 = (x_3 - x_4)^2 + (y_3 - y_4)^2$
bisector	$\frac{(v_1, v_2)}{\lvert v_1\rvert\lvert v_2\rvert} = \frac{(v_3, v_4)}{\lvert v_3\rvert\lvert v_4\rvert}$ (v はベクトル)

6　システムの全体的な評価

6.1　推論の精度

本システムの性能を評価するために、図３のようにソフトウェア上で様々な文章を入力として与え、その出力結果がどの程度正確であるかを調査した。まず、三角形や円などの基本的な図形や平行四辺形や正三角形などの一部の特徴的な図形はどのような言い方でも適切に出力できることが確認できた。また、独自の記法や言い回しに関してもいくつものパターンに対応できるということがわかった。一方で記述量や情報量の多い入力に関しては一部の条件が抜け落ちているものや全くでたらめな出力を出してしまうことが多かった。特に後者に関しては、学習に用いたデータセットの内容からそのまま取ってきたものが出力されていることが多かった。

他の興味深い事例として、同じ出力を想定した文章であっても、説明の仕方が異なるだけで結果が変わるということが多く見られた。例えば、「正六角形がある」という説明だけでは適切な出力ができないが、「各辺が等しく、内角もそれぞれ等しい六角形ABCDEF」という説明であれば適切な出力ができる。このことから、入力文の説明の細かさが推論結果の正確さに反映すると考えられる。これとは別に、２つの入力文でそれぞれ違う出力を想定したはずが、それぞれ全く同じ結果を返してしまうというケースもあった。例えば、「三角形 ABC とそれに内接する円がある」という入力に対する出力が「三角形 ABC とそれに外接する円がある」の解答と同じになることがあった。これはお互いの文章が非常に似ているために、LLM 内での確率計算により両者は同じものであると判断されてしまうためだと思われる。これに関しても前述と同様に細かく説明をしてやれば正しく出力できるケースが多かった。

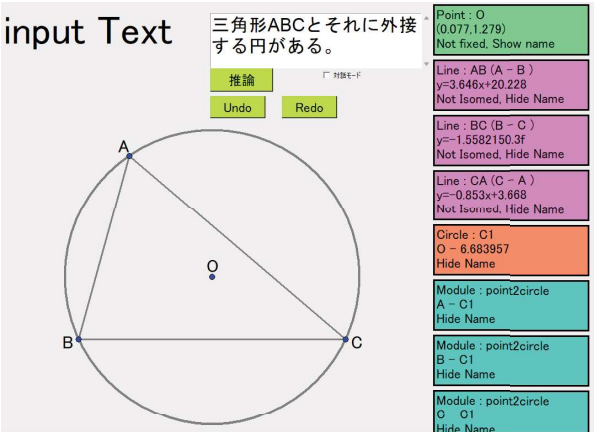

図 3: PointLine 上での文章入力から図形を出力する

6.2　幾何制約チェックと座標決めの精度

幾何制約チェックの精度に関しては、非線形最適化による近似解の収束度をもとに判定を行った。この判定は、目的関数が収束したかどうかをしきい値を設定することによって行い、収束した場合は矛盾がないと判断し、収束しなかった場合は矛盾があると判断する。この判定方法により、矛盾があるケースをほぼ検出することができた。ただし、オブジェクトやモジュール数が多い場合に処理時間が長くかかることがある。　非線形最適化で収束した場合にその解をそのまま座標として用いるとき、表示した図形が想定通りの配置であるかどうかも調べた。その結果、解そのものは間違ってはいないが、想定しているような図形が表示されないことが多かった。例えば、各辺が等しい五角形の点座標の候補を求めるとき、５つの点全てが同じ座標に配置されてしまうケースがあった。この結果は方程式を満たすという観点から言えば正解であるが、PointLine 上での表示を考えると適切な結果とはいえない。これとは別に「五角形を描け」という入力に対する出力を考えると、本研究で使った制約だけでは点の配置は完全にランダムとなってしまう。こうなると図４のような普通は想定しないような図形が表示されることがある。本研究で用いた制約だけでなく、適切な図形を表示するための制約についても考える必要がある。

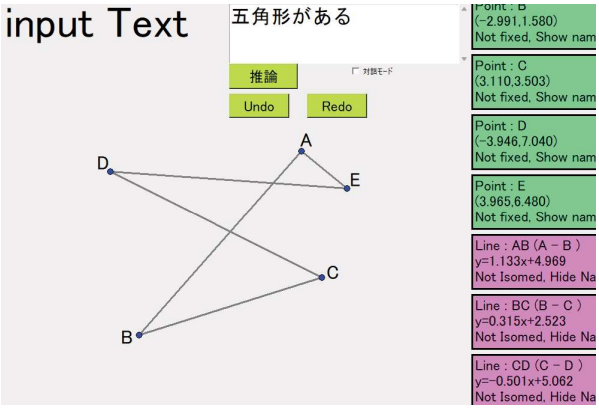

図 4: 一般的に想定しないような五角形の配置

7　まとめ

本研究では LLM とエラー検出機能による言語入力からソフトウェア上に図形を表示するようなシステムを提案した。LLM による図形推論は短い文章や単純な図形であればほぼ正しく出力できるが、情報量の多い長い文章の場合は出力できないケースがあった。その中でのほとんどは説明文の情報が省略されていたり、文章が簡潔すぎるケースが多く見られた。そのため、言語情報から適切な図形表示を行うためには幾何学的な状況に関する詳細な説明が必要である。

幾何学的な矛盾の判定は点座標の近似解を最適化で求めることによって行った。これ

により近似解がどれだけ収束したかで矛盾をほとんどのケースで検知することが可能であるが、一方でオブジェクトやモジュールの多さに依存した処理時間の長さが課題となっている。また、求めた近似解における図形の配置は必ずしも適切なものとは限らないことがわかり、モジュールとは別の制約をつけるか、非線形最適化を用いる手法自体を見直すかなどの改善が必要であると考える。

参考文献

[1] GeoGebra: https://www.geogebra.org/

[2] PointLine: https://aharalab.sakura.ne.jp/PointLine/

[3] 阿原一志: PointLine：作図手順の概念のない作図ソフトウエアの提案, 数理解析研究所講究録 2105, (2019)

[4] GPT-4o: https://openai.com/index/hello-gpt-4o/

[5] SciPy: https://scipy.org/

[6] SLSQP: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-slsqp.html