



京都大学人工知能研究会
KaiRA
Kyoto Univ. AI Research Association

2023

KaiRA

JOURNAL vol.7

Kyoto Univ. November Festival 2023.11

はじめに

2023年は生成AIが世間一般に広く認知された年となり、今年の新語・流行語大賞には「生成AI」「チャットGPT」がノミネートされました。さらに「観る将」という言葉も新語・流行語大賞にノミネートされました。最近では将棋のライブ配信でAIの評価値が表示されることが多くなり、将棋の知識がなくてもどちらが勝勢なのかが容易に分かるようになったことが「観る将」の流行につながりました。

このようにさまざまなAIに一般の人々が触れる機会が増え、AIがより身近な存在になったと言えるでしょう。本会誌では、今年注目を集めたChatGPT（第Ⅱ部）、将棋AI（第Ⅲ部）、画像生成AI（第Ⅳ部）についての記事を掲載しています。

第Ⅰ部では、AIの歴史と未来について紹介しています。もしかすると皆さんの中には、AIはつい最近登場したものだと思っている方がいるかもしれません、実はAIは1950年代から研究されており、幾度かのAIブームを経て今に至っています。

第Ⅲ部では、私が独自に開発した”引き分けを目指す”オセロAIについての解説を行っています。NFの展示会場でも体験できるようになっていますので、ぜひ遊んでみてください。

第Ⅴ部では、AIの判断理由を説明する手法であるXAIについて紹介しています。実はAIは「なぜその予測をしたのか」を説明することが苦手です。研究者たちは何とかその問題を解決しようと様々な解釈手法を考案しており、中でも代表的な3つの手法を紹介しています。

本会誌は、KaiRAの会員たちが協力して作成してくれました。KaiRAでは毎週木曜日に勉強会を開催しており、活発な議論が行われています。本会誌を読んでAIに興味を持っていただけたら、ぜひあなたもKaiRAで私たちと一緒にAIを学んでみませんか。

京都大学人工知能研究会 KaiRA 会長
理学部(地球物理学)3回生
松田拓巳

目次

第Ⅰ部 AI の歴史と未来	7
第1章 AI はどこから来たのか AI は何者か AI はどこへ行くのか	8
1.1 成長速度の爆発	8
1.2 AI はどこから来たのか そして何者か	10
1.2.1 黎明期	10
1.2.2 第一次 AI ブーム 「推論と探索の時代」	10
1.2.3 第二次 AI ブーム 「知識」の時代	12
1.2.4 第三次 AI ブーム 「機械学習」の時代	13
1.2.5 第四次 AI ブーム 「生成系」の時代	15
第Ⅱ部 ChatGPT	16
第2章 自然言語処理の基本アイデア	17
2.1 始めに	17
2.2 単語の意味	17
2.3 言語モデル	18
2.4 終わりに	20
第3章 ChatGPT のモデルの仕組み	21
3.1 はじめに	21
3.2 モデルの構造	22
3.3 モデルの学習方法	22
3.4 終わりに	25
第4章 言語モデルのプロンプトエンジニアリング	26
4.1 プロンプトとは	26
4.2 Few-shot prompting	26
4.3 Chain of Thoughts	27
4.3.1 Few-shot Chain of Thoughts	27
4.3.2 Zero-shot Chain of Thoughts (通称: step by step)	30
4.4 Self-translate	32
4.5 Take a Deep Breath	32

4.6	Echo Prompt	33
4.7	まとめ	34
第5章	生成系AIを巡る法的・倫理的課題	36
5.1	開発企業に関する課題	36
5.1.1	個人情報保護	36
5.1.2	プライバシーのリスクとセキュリティ	36
5.1.3	バイアス	36
5.2	ユーザーに関する課題	37
5.2.1	著作権や知財	37
5.2.2	プライバシーと個人情報保護	37
5.2.3	業法との関係	37
5.2.4	出力されたデータの不適切性	37
5.3	AI自体に関わる課題	38
5.3.1	仕事を奪う	38
5.3.2	環境上の課題	38
5.4	法規制やガイドラインに関する動向	38
第6章	最近のLLMの動向	40
6.1	LLM開発の流れ	40
6.1.1	OpenAI	40
6.1.2	Google	41
6.1.3	Microsoft	41
6.1.4	Meta	41
6.1.5	中国の動向	42
6.1.6	日本の動向	42
第III部	ゲームAI	43
第7章	引き分けを目指す「忖度オセロAI」	44
7.1	忖度オセロAIを作った理由	44
7.2	精度	44
7.3	忖度オセロAIのしくみ	44
7.4	評価関数の学習	45
7.5	探索アルゴリズム	46
7.6	おわりに	47
第8章	将棋AIの仕組み	49
8.1	将棋AIの入力と出力	49
8.1.1	局面の表現	49
8.1.2	指し手の表現	49
8.2	方策ネットワークと価値ネットワーク	50
8.2.1	方策ネットワーク	51

8.2.2	価値ネットワーク	51
8.2.3	マルチタスク学習	51
8.3	モンテカルロ木探索	52
8.3.1	探索ノード	53
8.3.2	多腕バンディット問題	53
8.3.3	探索のシミュレーション	53
8.3.4	アーク評価値と勝率の更新	54
8.4	まとめ	54
第 9 章	将棋 AI の思考プロセスを解き明かす：局面評価の可視化と考察	55
9.1	はじめに	55
9.2	将棋 AI の局面評価を可視化について検討する	55
9.2.1	Grad-CAM (Gradient-weighted Class Activation Mapping)	56
9.2.2	「dlshogi」開発者が提案する将棋 AI の可視化手法	56
9.3	将棋 AI の思考プロセスについて考察する	56
9.3.1	入力層から将棋の特徴について考察する	56
9.3.2	出力された局面の駒の価値を可視化する	58
9.3.3	将棋 AI の局面評価について考察 1	58
9.3.4	将棋 AI の局面評価について考察 2	58
9.3.5	将棋 AI の局面評価について考察 3	59
9.3.6	将棋 AI の評価値の注意点	61
9.4	まとめ	61
第 IV 部	画像生成	62
第 10 章	生成モデルの基本	63
10.1	生成モデルとは	63
10.1.1	生成モデルの目的	63
10.1.2	生成モデルによるデータの生成	63
10.2	拡散モデルの基礎	63
10.2.1	DDPM	64
10.2.2	拡散モデルの課題	68
10.3	GAN の基礎	69
10.3.1	基本的な GAN	69
10.3.2	WGAN とは	71
10.3.3	モード崩壊	73
第 11 章	生成モデルの応用	75
11.1	Diffusion Model の応用	75
11.1.1	ガイダンス	75
11.1.2	サンプリング手法	77
11.1.3	Stable Diffusion の活用	79

11.2 GAN の応用	81
第 V 部 説明可能な AI(XAI)	83
第 12 章 LIME	84
12.1 LIME による説明の例	84
12.2 LIME の仕組みを 3 段階のレベルで説明	85
第 13 章 SHAP	89
13.1 シャープレイ値	89
13.1.1 シャープレイ値のアイデア	89
13.1.2 シャープレイ値の具体的な計算方法	90
13.1.3 シャープレイ値の性質	91
第 14 章 タラレバを機械で実現する方法	93
14.1 反実仮想説明とは	93
14.2 反実仮想説明の仕組み	94
14.2.1 理想的な反実仮想の条件	94
14.2.2 反実仮想のモデリング	94
参考文献	96

第Ⅰ部

AIの歴史と未来

第1章

AIはどこから来たのか AIは何者か AIはどこへ行くのか

人間が文字を理解することができる原因是、人の脳に言語処理に特化する特殊な領域があるからだとされています [1]。ブローカ領域と呼ばれるこの場所は、言語の理解と生成に寄与して人間に独自の言語能力を与えてくれます。われわれ人類が地球上で、ある種の支配的な地位を得ることができたのも、この言語機能のおかげといえるでしょう。

たしかに人間は、爪の鋭さや筋肉の力強さなどの多くの面で他の生物種よりも劣っています。しかしこれ頭脳においては現存する他のどの生物種と比べても優れています。この優位性によって、我々は言語を発達させ、技術を発展させ、複雑な社会組織を構築することができたのです。そして、われわれは先行する時代の成果の上に、自分たちの時代の成果を積み重ねていき、世代を経るごとに優位性を増幅させてきました。

ここ最近、特にChatGPTなどの生成系AIの実現などは、技術的特異点^{*1}を連想させ、人類はいつの日か、人間の頭脳を凌駕する”超知能”を構築し、われわれ人類の運命をそのスーパーインテリジェンスに委ねることになるのではないかという言説が、ややディストピア思想や陰謀論の怪しげな雰囲気をはらみながら広がりつつあります。

この章では、人工知能がどのような歴史をたどって現在の技術にたどり着いたのか、また現在の技術レベルはいかほどのものかなどを歴史に沿って理解することを主眼に論説していきます。

1.1 成長速度の爆発

数百万年前、われわれ人類はアフリカ大陸で樹上生活をしていました [2]。その頃のヒト科の種は、主に狩猟採集生活を営んでおり、動物の狩りと野生の植物や果物の採集で生計を立てていました。加えて簡素な道具や火の利用、社会的な構造、言語を用いた知識の蓄積など、ある程度発達した文化体系を持っていましたとされています。

その後、今から約20万年前にホモサピエンスという新たな種が出現しました。ここで重要なのは、脳容量と神経組織の進化が認知能力の大躍進をもたらしたということです。その結果、人類は物事を抽象的に考えることができるようになり、複雑な意思疎通を行える能力を獲得しました。そして、これによって場所や世代を超えて知識を蓄積でき、人類は地球上である種の支配的な地位を築くことができています。

^{*1} シンギュラリティ、2045年問題とも呼ばれる

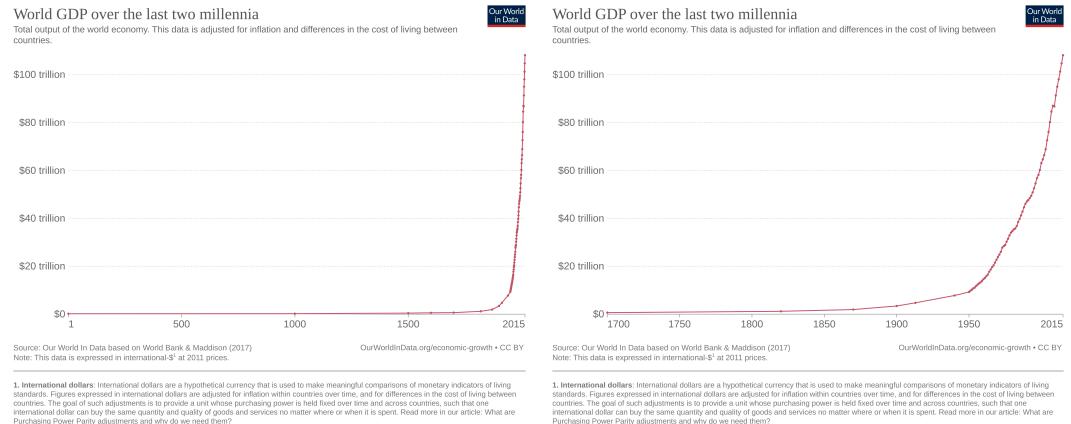


図 1.1: 世界の GDP の推移 (紀元後: 左)(1970 年以降: 右)

出典：<https://ourworldindata.org/grapher/world-gdp-over-the-last-two-millennia>

これらの能力を獲得したお陰で、われわれの祖先は効率的な生産技術を徐々に身につけ、アフリカの熱帯雨林からサバンナ地帯まで世界の様々な地域に拡散することができるようになりました。それ以降、総人口の増加も伴い、経済的生産性や技術的能力を増加させていきます。

ホモサピエンス誕生以降の経済の発展において、いくつかのブレークスルーがあるとされています。

まず初めに農業革命^{*2}です。紀元前 7000 年のメソポタミア文明をはじめとする古代文明では、従来の狩猟採集生活からなる獲得経済にとどまっていた人類が、はじめて穀類や根菜類を栽培し、家畜を飼育して食糧を生産する農耕・牧畜という生産経済に移行しました。

次に挙げられるのが産業革命です。産業革命は、18 世紀後半のイギリスで起こった、綿工業や蒸気機関などの工場製機械工業の出現という生産技術の革新とエネルギーの変革のことをいいます。このような生産様式の大幅な変化は、人間の成長速度のギアを数段高くなる効果があったとされています [3]。

この革命は、人類の文化技術に重大な進展をもたらしました。例えば、先史の狩猟採集生活をしていた時代では、世界の経済規模が 2 倍になるのに 22 万 4000 年の年月が必要だったのに対し、農耕革命以降では 990 年、産業革命以降の産業化社会では 6.3 年と成長速度の劇的なスピード上昇があります [4]。つまり、このような推論から農耕革命や産業革命に匹敵する成長速度のギアアップが今後起これば、以前の革命と同様に指數関数の底が変わるので似た GDP の増加が起こることが予想できます [図 1.1]。

さらに 20 世紀に中頃には、コンピュータ、インターネット、通信技術の発展によって情報の流通が格段に向上了し、グローバルな経済とつながりが強化されました。またこれらは、アイデアを瞬時に世界中に伝えることができ成長の速度上昇に寄与しています。さらに 21 世紀に入り、その情報技術の発展に伴い、機械学習、ディープラーニング、自然言語処理などの AI 技術が発達していきます。ここ数年の単位では ChatGPT などの大規模言語モデルによって、さらなるブレークスルーが起きようとしています。

そこで出てくる議題が、このような発展しつつある人工知能の分野が農業革命や産業革命に匹敵す

^{*2} ここでは 18 世紀のイギリスにおける農業革命ではなく、紀元前 7000 年のメソポタミア文明に始まったとされる農耕牧畜への経済様式の変容を指している。

るような次なる革命になりうるのか、また先述のように世界経済の成長曲線がある時点で跳ね上がるようなことが起きうるのかというものです。そのような問題を適切に議論するためには、これまで人工知能がどのような道筋を辿ってきたのか、また現在どのような技術を持ち、どのようなことが可能にならぬかということを知ることは必須です。

次の章では、人工知能の発展の歴史を簡単に紹介しそのような機械が開発されてきたかを順番に追ってみようと思います。

1.2 AIはどこから来たのか そして何者か

1.2.1 黎明期

1940年代後半から1950年代にかけて、戦争の影響を受けてコンピュータの開発が進んでいた頃。当時の計算速度や記憶容量が限られたコンピュータ黎明期においてチェスやチェックなどのゲームが応用の対象となり、ゲーム理論をはじめとする理論が組み立てられました[5]。ノイマンとモルゲンシュテルンの1944年の「ゲーム理論と経済行動」[6]や、シャノンのチェスに関する最初の論文[7]などが発表されました。

また同時にコンピュータの開発によって、人工的に作られた知能という夢の機械について人間機械論や頭脳と計算機の関係など、大きな議論が沸き起こりました。

その中で特に顕著だったのがノーバート・ウィーナーというMITの数学教授が著した「サイバнетิกス」という書籍[8][9]です。その中で、著者はマシンと生物の脳や神経系との類似点を並べてAI関連の多くのアイデアにふれています。この本は本来数学に熟達した人でないと理解不能であつたはずですが、社会から多くの注目を集めました。その後、マシンは考えることができるのかといった問題提起が出版物やラジオショーで真剣に議論されるようになったそうです[10]。

そのような社会的な期待が高まった中で第一次AIブームが始ることとなります。

1.2.2 第一次AIブーム「推論と探索の時代」

人工知能研究の発端は諸説ありますが、1956年に開かれたダートマス会議が人工知能研究の夜明けだと言われることが多いです。世界最初のコンピュータからわずか10年後に開かれたこの会議は、この分野の先駆者が多くが参加しました。彼らが開催にあたってロックフェラー財団に提出した提案書[11]には、以下のような内容が書かれてあります。

We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it. An attempt will be made to find how to make machines use language, form abstractions and concepts, solve kinds of problems now reserved for humans, and improve themselves. We think that a significant advance can be made in one or more of these problems if a carefully selected group of scientists work on it together for a summer.

(出典：<http://jmc.stanford.edu/articles/dartmouth/dartmouth.pdf>)

「1956年の夏にダートマス大学で人工知能に関するワークショップの実施を提案します。このワークショップでは、人間のあらゆる知的活動は、それを詳細にかつ正確に表すことができれば機械によってシミュレートすることができる、という前提を元に議論を進めていきます。厳選された科学者が一緒に取り組めば、抽象的な議論や今は人間にしか解決できないようなことを機械に解かせるという問題に対して大きな進歩を生むことになるでしょう。」

この提案書に現れているように、知能を人工的に作ることに対して楽観的な見解を持っていたようです。この会議の中での研究者の意見の多くは「人間の知的活動は機械で代行でき、その最良の道具はデジタル計算機である」というものでした。このような人工知能研究の出発から現在に至るまで70年近く経過しており、その間、大きな進展と期待の時代と反対に挫折と落胆を味わう冬の時代を交互に経験しながら今日に至ります。

とにかく、最初の人工知能の盛り上がりはダートマス会議の直後にやってきて、これが第一次AIブームと呼ばれています。

この時期の、人工知能の可能性に懐疑的であった人々はアラン・チューリングの停止問題^{*3}から派生した「いかなるマシンによってもXを行うことは不可能である」(Xには推論などが当てはまる)という言葉を中心とした人工知能に対する反論を行っていました。

それに対して人工知能の研究者たちは、例えば簡単な数学の問題、論理記号問題、大学一般教養レベルの微積分問題、IQテストの視覚類推の典型問題、単純な代数文章題などの、厳密に定義された特定の条件下において推論を行いうる小規模システムを開発し、懐疑論者の主張が誤りであることを証明してみせました。

他にも1965年にはELIZAとして名付けられたシステムが発表され衝撃を与えました[12]。これは簡単な構文的操作をするだけで自然な会話をするプログラムで、ELIZAが心理的療法のセラピストになりますとして、患者役のユーザに対応することができることを示してみせました。相手側の文章の一部をたくみに借りることによって、どのような会話でもこなすことができたようです。ELIZAは、最初のチャットボットとしてChatGPTの文脈で紹介されることも多く、YouTubeなどにも動画が上がっているので調べてChatGPTと比較してみると面白いのではないかでしょうか。

1960年代の終わりになると、SHRDLUというシステムが開発されました。このシステムはユーザーがコンピュータに英語で指示を行い、それに従ってコンピュータが画面上の円錐や直方体、球などの積み木みたいな物体を動かすというシステムです[13]。その他にも、クラシック調の音楽を作曲できるシステム、自動運転システム、簡単な臨床診断ができるシステムなど多くのプログラムが開発されました。

しかし、順調そうに見えた人工知能研究の分野にも問題解決の対象分野が広がったり、さらなる高度で難解な問題の解決を求められたりすると、その挑戦に多くの場合挫折することとなりました。そのような事態から、トイプロブレム（おもちゃの問題）という言葉が生まれ、簡単なルールとゴールが決まっている問題しかAIが解くことはできず、複雑な現実問題を解くことはできないというふうに揶揄されるようになります。

当時の人工知能が壁に突き当たった理由の一つとして、網羅的探索法のようなアプローチを採用していたという点が挙げられるようです[3]。網羅的探索法とは、一言で表すならばしらみつぶしに考えていく手法のことで、例えば京都大学から金閣寺まで行く最適なルートを考えたときに最短距離の

^{*3} 計算可能性理論における停止性問題とは、「どんなチューリングマシン、あるいは同様な計算機構についても、それが有限時間で停止するかを判定できるアルゴリズム」は可能か、という問題 [出典: Wikipedia]。チューリング自身によって停止問題を解くアルゴリズムは存在しないことが証明されている。

Talk to Eliza by typing your questions and answers in the input box.

```
> Hello, I am Eliza. I'll be your therapist today.  
* Hello  
> It is nice to be chatting with you.  
* what can you do?  
> Is it that you would like to be able to do?  
* I want to be able to speak Japanese fluently  
> What would it mean to you if you got to be able to speak Japanese  
fluently?  
* I want to talk to Japanese people and understand their culture.  
> I sometimes also want to talk to Japanese people and understand their  
culture.
```

TYPE HERE

図1.2: ELIZAとの対話

出典：<https://web.njit.edu/~ronkowitz/eliza.html>

経路から地球を一周してたどり着くルートまですべての経路を数え上げてその中で最適なものを選ぶという方法です。当然京都大学から金閣寺まですべてのルートを考慮すると無数に考えることができます。このような組み合わせ爆発に対応するには、抽象的な解釈や事前知識を導入するアルゴリズムが必要とされます。初期の人工知能開発の時代に、このようなアルゴリズムの有用性や、不確実性や記号表現の理解が進まなかったようです。他にもメモリ容量やプロセッサ演算能力などのハードウェアの制約も存在していました。

1970年代なかばになるとこれらの問題の存在が知られるようになり、当初期待されていたような成果を上げるのは無理なのではないかとの見解が人々の間で広がり始めます。それを契機に人工知能関連の研究予算が削られ、最初の冬の時代が到来します。

1.2.3 第二次AIブーム「知識」の時代

1980年代になると、全世界的に第二次AIブームが始まります。このブームは研究や産業だけではなく社会現象にまで広がりました。

日本ではこの時期に「第5世代コンピュータプロジェクト」という官民一体のプロジェクトが開始されます。このプロジェクトは、人工知能の開発の基盤になるような最先端の並列推論マシンの開発という目標を掲げ、Prologという言語で進められていました。計算機の高性能化や知識工学の理論の提唱などの背景もありながら、高度経済成長期の日本の恩恵にあやかりたい他国もそれにつづき人工知能の研究開発に資金が投じられるようになりました。

第二次AIブームの一番の特徴は、エキスパートシステムです。1977年ごろから知識工学が提唱されはじめ、豊富な記憶容量を獲得したコンピュータで知識処理を行うという研究が勃興しました。エキスパートシステムは、人間の専門家から収集された知識ベースのデータと、その高度な利用を行う推論技術を組み合わせて推論をおこなうというプログラムです。医療、教育、評価支援、診断、設計などの分野を対象としたエキスパートシステムが数百単位で構築されて、小規模なシステムはそれなりに得るべきことがあることが実証されました。しかし、大規模なエキスパートシステムとなると、その一つの大量のプログラムのために大量の知識を保有するコンピュータを用意しなければならず、

財政面や使い勝手の点で現実的でないことが段々と判明していきました。

第5世代コンピュータプロジェクトや、それに追随するアメリカやヨーロッパのプロジェクトはそれなりに成果を上げたものの喧伝された目標を達成することではなく、人工知能の分野に二回目の冬の時代が訪れることとなりました。

1980年代に絶頂期を迎えた従来のAIは少し揶揄的なニュアンスを込めて、現在ではGOFAI(Good Old-Fashioned Artificial Intelligence)と呼ばれるようになります。

1.2.4 第三次AIブーム「機械学習」の時代

第二次AIブームの失敗にもめげずに、研究者の間では技術的課題を克服するための努力が継続されました。そして1990年代になると冬の時代は緩やかに回復はじめ、人工知能の研究分野に風が吹き始めます。この契機となったのが、ニューラルネットワークと言われています。

ニューラルネットワークとは、現在でも用いられている機械学習や深層学習の核とも言える理論で、端的にいえば生物の脳神経回路を模して作られたアルゴリズムのことをいいます。

ニューラルネットワークのはしりは1940年代にさかのぼります。1943年に、人工知能を作るにあたって、生物の脳の仕組みを解明しようという目的で、MCPニューロンと呼ばれる神経細胞を模した簡単なアルゴリズムが発表されます。その数年後、1950年代にMCPニューロンを基にしたペーセプトロンという学習規則が発表されました。ペーセプトロンの理論が発表されてから数十年の間、ニューロン（神経細胞）を基にした学習規則の良い方法が誰からも提案されない空白期間が存在しました。しかし1986年、誤差逆伝播法と呼ばれる多層ニューラルネットワークを学習する際に用いられる非常に重要なアルゴリズムが考案され、ニューラルネットワークの研究が注目されるようになりました。

その後、1990年代、2000年代のいくつかのブレークスルーによって現在のディープラーニングアルゴリズムが開発されました。深層学習とは、神経細胞のネットワークをいくつか組み合わせることで非常に厚い神経細胞の層を構成して学習を行うというものです。深層学習は、ハードウェアの高性能化も寄与し、2010年代から圧倒的な性能を誇るようになっていきます。特に顕著な事例は、国際的な画像認識アルゴリズム性能のコンペティションとして知られるILSVRCの2012年大会において、トロント大学のHintonらのチームが深層学習を用いて史上最高スコアを大幅に塗り替えて優勝しました。この時をもってディープラーニングの勃興とみなす資料は多いです。

以後のディープラーニングの快進撃は凄まじく、画像認識分野を皮切りに、自然言語処理や時系列データなどにも理論が適用されました。

2012年にGoogleから発表された、猫の画像認識を行うAIも大変話題になりました。GoogleはYouTubeやWebなどからランダムに抽出した大量の猫の画像を使ってディープラーニングを行ったところ、人間の顔や猫の顔に反応するニューロンが生成されたというのです。つまり人工知能が人間の顔や猫の顔を認識できるようになったということです。

2015年にはDeep Q-Network(DQN^{*4})がDeep Mindから発表されました。DQNはわずか数時間でコンピュータゲームのルールを見つけ、高得点を獲得する方法を身につけたという内容です。その例として、ブロック崩しを600回プレイしたところ、DQNはブロックの一部分を集中的に破壊してトンネルを作り、上部にボールを入れることで効率的にブロックを消していく必勝法を自ら発見するまでに至りました。

^{*4} ヤバい人という意味ではない



図 1.3: Google の猫

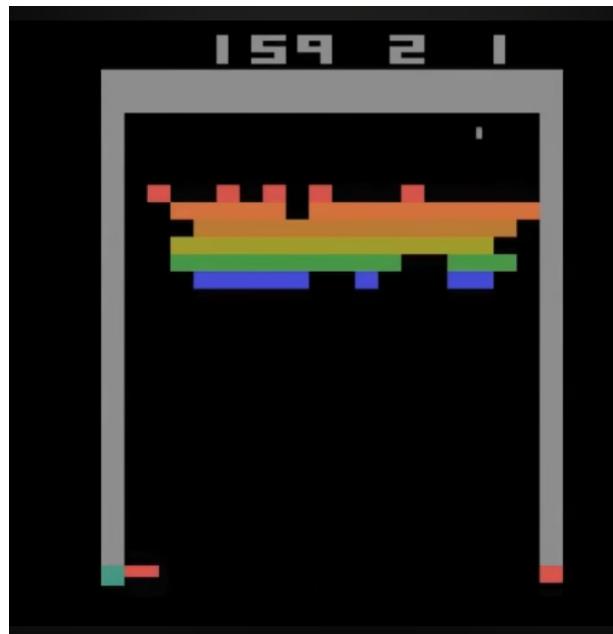
出典：<https://blog.google/technology/ai/using-large-scale-brain-simulations-for/>

図 1.4: ブロック崩しプレイする DQN

出典：<https://youtu.be/TmPfTpjtdgg?si=eiOHkg29PCDOKwUJ>

ゲームの分野でも快進撃が続きます。非常に場の組み合わせが多く、人間に勝つことは難しいとされていた将棋で 2015 年に AI がプロ棋士に勝利し、囲碁でも 2016 年に Google の AlphaGo が当時のチャンピオンに勝利しました。ここで重要なのは、人間が決めたロジックに従って AI は手を選んでいるわけではないという点です。実際に AlphaGo の打つ手はプロ棋士にも不可解で、なぜこの手を選んだかをあとから人間が理解することは難しいです。ルールを教えなくてもディープラーニングによって学習ができるという証明がなされ、ボードゲームの世界において AI と人間の勝負はついたといえるでしょう。

上で見てきたように、第一次 AI ブームから第三次 AI ブームまで必ずしも一直線ではない歴史を歩んできたことが見て取れると思います。人工知能実現への道は、コンピュータの開発、ダートマス会議からいくつもの波を受けながら今も進んでいる最中です。

1.2.5 第四次 AI ブーム 「生成系」の時代

2022 年終わり頃、その年の人工知能の話題をかっさらった存在がいます。それは ChatGPT です。2022 年 11 月 30 日に一般公開された ChatGPT は公開一週間後に利用者が 100 万人を突破し、2 ヶ月で 1 億人を突破し、今までとは異次元の”人間っぽさ”によって大きな話題を呼んでいます。さらに ChatGPT 以外にも画像生成、動画生成、プログラミングコード生成などをまとめて Generative AI (生成系 AI) と呼びこれらの勃興で、第三次 AI ブームから継続して第四次 AI ブームに入ったという見方もあります。

ChatGPT の内容については本書でも取り上げているので、詳しくはそちらに譲るとして、ここでは ChatGPT を含む生成系 AI が話題を呼ぶまでの簡単な経緯を解説します。

現在の Generative AI の勃興は、第三次 AI ブームの深層学習の流れを汲んでいます。2014 年には画像系の Generative AI につながる GAN(Generative Adversarial Network) が開発されました。GAN は 2 つのモデルを競わせるように戦わせて、新しい画像データを生成する深層学習の一種です。このブレークスルーによってリアルな画像や動画作成が可能になりました。GAN の理論はこの会誌でも取り上げています。

2017 年に Google が深層学習モデル Transformer を発表しました。Transformer はもともと機械翻訳用のモデルとして提案され、入力された単語や文字を Attention という機構によって特定の情報に重点を置きながら学習するのが特徴です。これによって、自然言語処理と呼ばれる人間が自然に話す言語のマシンが急激に精度を向上させていきました。

その後、Transformer を基に様々なモデルが開発されました。特に注目されたのが 2018 年に Google が発表した BERT と OpenAI が発表した GPT です。BERT は自然言語処理のベンチマークの多くのタスクにおいて最高スコアを更新し、BERT の改良が 2019 年と 2020 年に立て続けに発表されました。一方 GPT は 2020 年に長くまとまりのあるテキストを生成する GPT-2、2020 年 7 月には GPT-3 が、2023 年 3 月に GPT-4 が発表されて更にバージョンアップを重ねています。

そのような流れで 2023 年の現在では、自然言語処理に限らず、画像生成、音声生成が同様の経緯で発達し、まとめて Generative AI として急速に普及し始めています。

第 II 部

ChatGPT

第2章

自然言語処理の基本アイデア

2.1 始めに

自然言語とは私たちが普段話している言葉のこと、プログラミング言語などの機械語に対置される概念です。自然言語処理の技術というのは、その自然言語をコンピュータに処理させる技術のこと、近年話題の ChatGPT もこれを活用しています。我々が普段何気なく使っている自然言語は、いざコンピュータに扱わせようすると中々難しいものです。本稿では、どのようにして複雑な自然言語をコンピュータで扱っているのかというところに焦点を当てて、自然言語処理の仕組みを簡単に解説してみたいと思います。

2.2 単語の意味

コンピュータに自然言語を教える際に難しい点はなんでしょうか？当然問題は一つではないですが、単語の意味を如何にして理解させるか、ということが大きな困難となるのは想像に難くないと思います。誰もが直ぐに思いつきそうな方法として、私たちが外国語を学習するときのように、辞書を作成するというアイデアが考えられます。実際にコンピュータ用に作成された特別な辞書「シソーラス」が開発されました。シソーラスでは、同義語や概念の包含関係などの言葉の関係が示されています。しかし、シソーラスで表現可能な意味は図式的に示せるものが主であり、単語の細かなニュアンスなどの表現は難しいものでした。もっと言うと、シソーラスの作成にはかなりの手間がかかります。辞書を編纂しているようなものですから、かなり効率は悪そうですよね。そこで、単語の意味をいくつかの数値の集まりであるベクトルとして表現するというアイデアが発案されました。平面上の位置を座標で表すことは中学校で習いますし、コンピュータでは色を赤・緑・青の3成分の比率で表しているということを知っている人も多いでしょう。その考え方を拡張して、異なるカテゴリーの単語も一つのベクトルとして表すことが試みられたのです。ちなみに、そのような単語をベクトルで表したものを、「単語の分散表現」と言います。さて、単語の分散表現を得る上で重要な仮説があります。それが「分布仮説」です。分布仮説とは、「単語の意味は周囲の単語によって決定される」というものです。単語の意味はそれが用いられる文脈に依存するということですね。これは、単語そのものには意味がないと言っているようなもので、疑問をもつ人もいるかと思いますが、実のところかなりの精度で単語の意味を表現できます。さらに素晴らしいことに、この仮説に基づけば、比較的容易に単語の分散表現が得られるのです。単語の意味がその周りにある単語で決まるなら、単純にその単語の周りにどの単語が現れやすいかを数えればよいですね。そうすれば、ある単語の周りに他の各単語がそれぞれどのくらいの頻度で現れたかをベクトルとして表すことができ、単語の分散表現が得

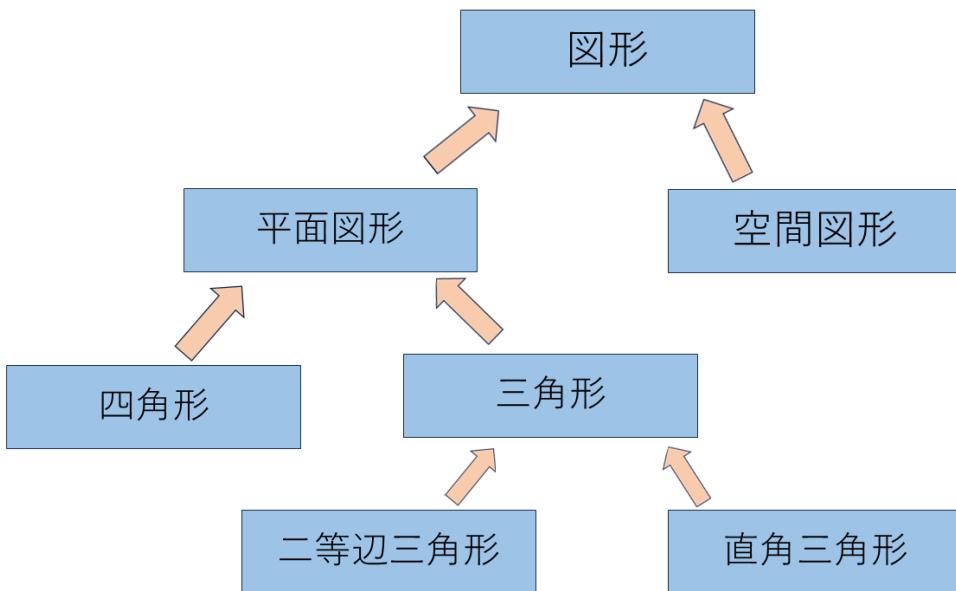


図 2.1: シソーラスのイメージ

られます。このような手法を「カウントベースの手法」と言います。

単語の分散表現を得るためにもう一つのアプローチとして、「推論ベースの手法」というものがあります。カウントベースの手法では、単語の周りにどんな単語が現れるかを数えましたが、推論ベースの手法では周囲の単語が与えられた時にどのような単語が現れるかを推測します。ここで、ニューラルネットワークが登場します。この記事では詳しい解説はしませんが（少なくともこの記事における）ニューラルネットワークとは、入力に対して出力を最適化するための仕組み、と言うことができます。例えば文字認識をする場合は、画像データという入力と読み取った文字という出力がありますが、ニューラルネットワークを用いた最適化を行うことで、正しく文字を判別することができるようになります。今回は、一単語が欠落した文という入力に対して、欠落部分の単語という出力を推測するというタスクにおいてニューラルネットワークを用います。適切な学習が行えればかなりの精度で空欄部分の単語を推測できるようになるのですが、このときの副産物として単語の分散表現が得られます。実は、ニューラルネットワークにおいては入力と出力はベクトルで与えられ、内部には行列値のパラメータが存在します。そしてこのパラメータを学習によって最適化しているのですが、この最適化された行列は単語の分散表現としても解釈できるのです。

2.3 言語モデル

さて、ここまででは単語の意味をどう表現するかという話題について述べてきましたが、ここからは別の問題についても考えていきたいと思います。もし仮に単語の意味を正確にコンピュータに与えることができたとして、果たしてコンピュータは私たちと同じように会話ができるようになるでしょうか？恐らくそれは難しいでしょう。言語には単語の他に文法という重要な要素があります。そして、文法とは単語の配置の規則、つまり単語の並べ方のことだと言うことができます。とは言え、自然言語においては、文法にもある程度の曖昧さが存在します。倒置はその代表格ですし、文学的な表現のために敢えて不自然に単語を並べることもあります。ある単語の並びがあったときに、その並びが正

カウントベースの手法

You say good and I say hello.

※Sayの周りの単語を調べる

推論ベースの手法

You say good and I [?] hello.

※周りの単語からsayを推測する

図 2.2: カウントベースと推論ベースの比較

しいのか間違っているのか判断することが難しいものが存在するのです。でも、いい方法があります。単語の並びを正誤の二択で判断するのではなく、その並びがありえそうな確率を求めればよいのです。そして、そのような単語の並びに対して確率を与えるモデルを「言語モデル」と言います。言語モデルは実に様々な用途に応用可能です。例えば、音声認識や文章生成の場でも言語モデルは活躍します。音声認識では、音声データを元に発声された言葉を推測しますが、似た発音の言葉は判別できることあります。そんなときでも、言語モデルを用いてどちらの方が単語の並びとして自然かを評価することで、認識の精度を上げることができます。さらに、言語モデルは文章生成にも活用できます。ある途中までの文章が与えられたとき、言語モデルを用いて、次にどの単語が来るのが一番自然かを判断することで、文章を紡ぎだすことすらできるのです。では、どのようにして言語モデルを構築するのでしょうか。本稿では実装の具体的な手法は省略しますが、言語モデル構築の有力手法である RNN を紹介したいと思います。RNN とは Recurrent Neural Network の略です。ニューラルネットワークは既に紹介しましたが、Recurrent（循環する）とはどういう意味でしょうか？ RNN はその最大の特徴として入力から出力までの流れの中に閉じた経路を持ちます。入力データはその経路の中で循環（Recurrent）するのです。この循環はニューラルネットワークに驚くべき機能を与えます。それは、時系列データをニューラルネットワークで扱えるようにするというものです。閉じた経路の中で循環するデータは過去の入力値を保持しています。そして、新しく与えられた入力は保持されていた過去のデータとともに処理されます。そのようにして、過去のデータを保ったまま新しい入力を受け取っていくことで、RNN は時系列のデータをその時間的順序を崩すことなく上手く扱うことができます。これがとても優れたモデルだということは、日常会話においても文脈という時系列データが大きな意味を持つことからもわかるかと思います。実のところ、单なるニューラルネットワークでは時系列的な並びは圧縮されてしまい、無視されてしまうことが多いのです。

例文1：
「彼はその鳥を捕まえて食べてしまいました」

可能性：高

例文2：
「その鳥は彼を捕まえて食べてしまいました」

可能性：中

例文3：
「捕まえてその食べては彼しました鳥」

可能性：低

図 2.3: 言語モデルの動作イメージ

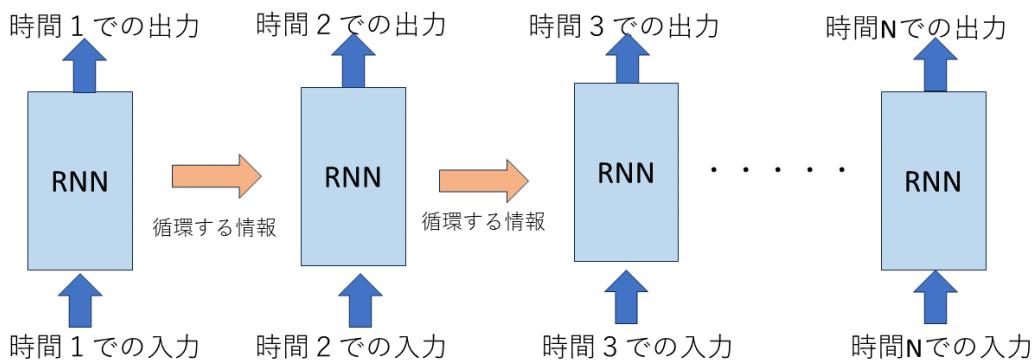


図 2.4: RNN のイメージ

2.4 終わりに

本稿では、自然言語処理の初歩的な考え方を具体的な数式やコードを用いずに解説しました。できる限り分かりやすくイメージしやすい記述を心がけましたが、著者もまだ勉強中の身ですので、分かりにくい箇所や不正確な言葉遣いはご容赦ください。もし、本稿を読んで自然言語処理に興味を持つていただければ幸いです。

第3章

ChatGPT のモデルの仕組み

3.1 はじめに

最近、世間で ChatGPT による AI チャットサービスが話題になっています。AI チャットサービスとしてはとても人間らしい文章を返してくれるのが特徴的ですよね。簡単な内容であればプログラミング言語を書くことができたりもします。素晴らしい面もある一方、説明が冗長的であったりとか、もっともらしい説明だけど論理的にはおかしいことを言っていたりするなどの欠点もあったりしますよね。このように、世間で話題になっている ChatGPT ですが、皆さんも ChatGPT がどのような仕組みなのかが気になったりしてませんか？ここでは、ChatGPT では、どのようなモデルが使われており、モデルをどのように学習しているのかを紹介したいと思います。

まず、ChatGPT がどのようなモデルなのかを説明していきます。ChatGPT は GPT(Generative Pre-trained Transformer) シリーズと言われる LLM(大規模言語モデル) のモデルの構造を用いています。ChatGPT では普通の教師あり学習に加えて、強化学習を行うことで良くない文を出力しないようにモデルを学習しています。ちなみに、ChatGPT を使っていたら普段よく見かけるのは GPT-3.5 や GPT-4 というモデルではないでしょうか。

LLM(大規模言語モデル) というのは言語モデルというもののパラメーター数を増やし、多くのデータで学習したものです。LLM(大規模言語モデル) いうものが現れた大きな理由として Transformer というモデルの登場があります。Transformer は当時のモデルよりも良い精度を示しました。さらに、Transformer というモデルは、パラメーター数を増やせば増やすほどモデルの予測精度が上がるということが知られてるようになったのです。（図 3.1 の一番右の図）（OpenAI のスケーリング則に関する論文ではパラメーター数以外のことにも主張をしていますがその説明は省略します。）このため、どんどんパラメーターの数を増やしていくという動きがありました。

言語モデルというものについても説明しておくと、言語モデルは今までに現れた文章の次に現れる可能性の高い文字を予測することを繰り返すモデルになっています。例えば「今日は」という文を言語モデルへ入力すると、「今日は」のあとに続く可能性が高い文字が出力されます。ここでは「晴」が続く可能性が高いとしましょう。すると文は「今日は晴」となります。さらにこの文を言語モデルへ入力し次の文字を出力させます。このように文の入出力を繰り返すことで、最終的には「今日は晴れです。」といった文が生成されるわけです。実際にチャットの用途としてモデルを用いる場合は入力として人が入力した文章が与えられ、先ほど説明したような文の入出力を繰り返して返答がされます。

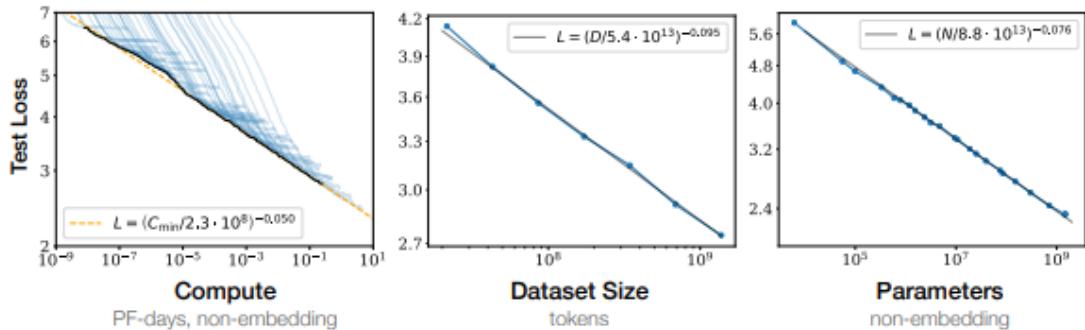


Figure 1 Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute² used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.

図 3.1: Transformer のスケーリング則の図 ([14] の引用)

3.2 モデルの構造

ここまででは ChatGPT についての前座ということで、ここからより詳細なところに触れていくましょう。まずは、ChatGPT のベースとして使われている GPT モデルの構造についてです。Transformer は Encoder と呼ばれる部分と Decoder と呼ばれる部分に分かれていますが、GPT シリーズでは、Transformer というモデルの Decoder 部分だけを使ったようなモデルになっています。(図 3.2。真ん中の青い四角部分が Transformer の Decoder) ざっくり Transformer についても説明しておくと、Transformer はある単語が別の単語とどれくらい関係性があるかを調べる Attention 機構というのを用いているモデルです(図 3.3)。Transformer は入力を次元の低いベクトル(潜在変数)にまとめる Encoder と、その潜在変数を用いて出力を出す Decoder という二つの部分から構成されているのですが、GPT シリーズではそのうち Decoder しか用いていません。ちなみに、Transformer の Encoder しか用いないモデルも存在しています。その例としては BERT とかが有名です。

3.3 モデルの学習方法

それではモデルの学習方法について説明しましょう。学習方法は InstructGPT と呼ばれるモデルとほぼ同じです。ChatGPT と InstructGPT では、使っている GPT シリーズが違うことと、ChatGPT の学習データには InstructGPT の学習データに加えて対話形式の学習データが加わっているということしか大きな違いがありません。学習方法の特徴はモデルの出力を人が評価してモデルの出力がより人から見て良い出力になるように学習する Reinforcement Learning from Human Feedback (RLHF) という手法を用いていることです。ChatGPT におけるモデルの学習のステップは以下のステップに分かれています。(図 3.4 もご参考に。強化学習のところについてより詳細に書いてくれている図 3.5 もご参考に。)

1. ラベルを出力するように教師あり学習をする
2. 複数個モデルの出力を無作為に選んできて、その出力を人間が良い順番に並べる
3. 順番に並べたラベルから、報酬モデルという出力の良さを評価するモデルを学習する

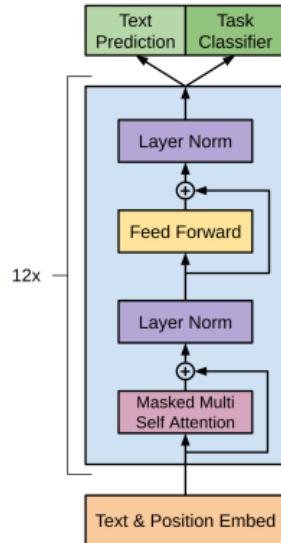


図 3.2: GPT-1 のアーキテクチャの図 ([15] の引用)

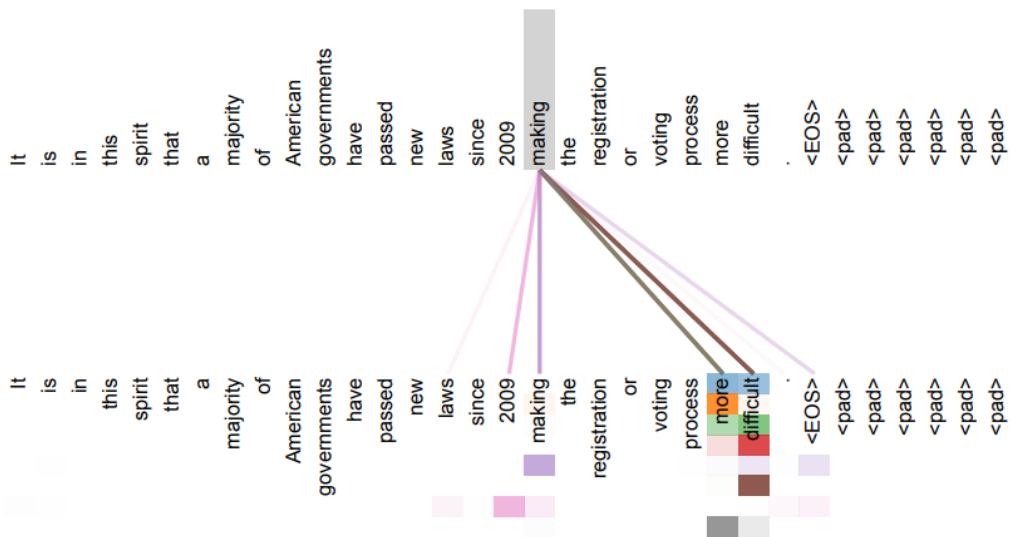


Figure 3: An example of the attention mechanism following long-distance dependencies in the encoder self-attention in layer 5 of 6. Many of the attention heads attend to a distant dependency of the verb ‘making’, completing the phrase ‘making...more difficult’. Attentions here shown only for the word ‘making’. Different colors represent different heads. Best viewed in color.

図 3.3: アテンションの図 ([16] の引用)

4. 報酬モデルで良いと評価されるようにモデルをファインチューニング(微調整)する(ここでPPO(Proximal Policy Optimization)という強化学習手法を使っています。)

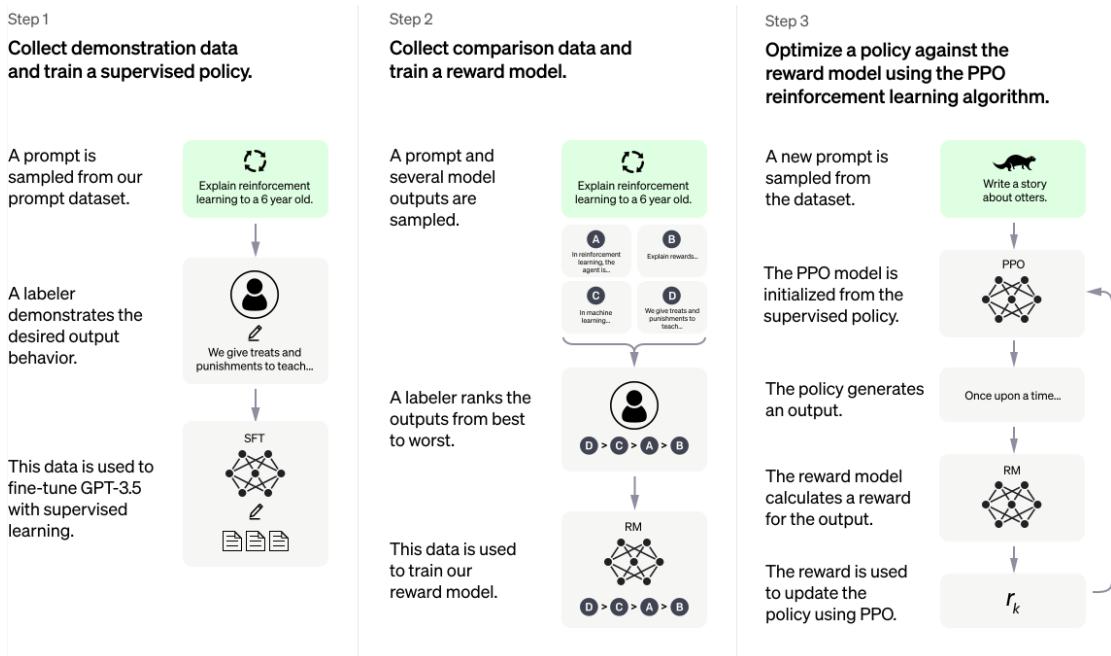


図3.4: ChatGPTの学習ステップの図([17]の引用)

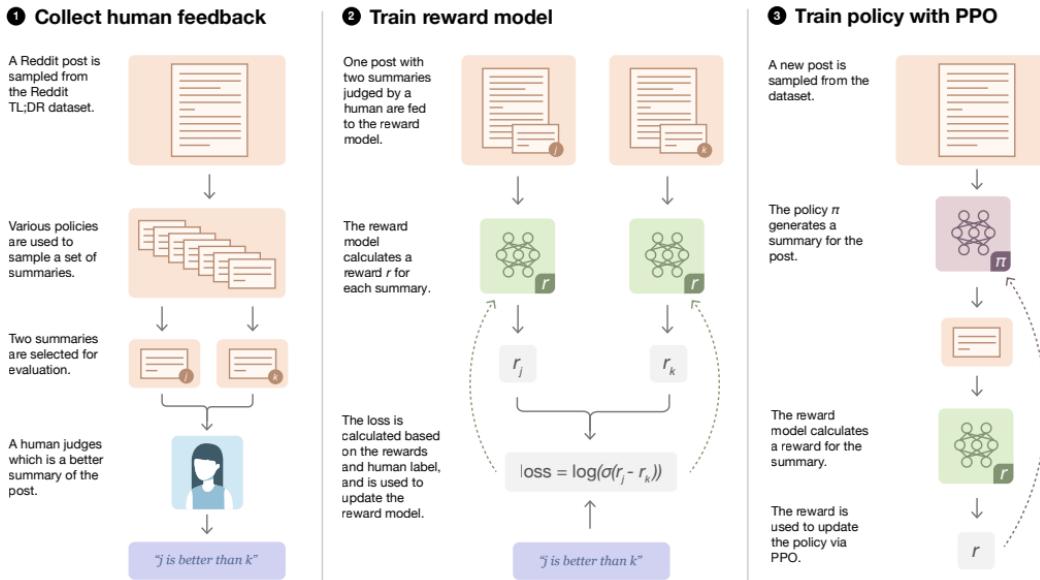


Figure 2: Diagram of our human feedback, reward model training, and policy training procedure.

図3.5: ChatGPTの学習ステップの二つ目の図([18]の引用)

ステップ1では教師あり学習をしていると書きましたが、1種類のラベルを使っているのではなく、以下の3種類のラベルを用いています。対話を意識したようなラベルであることが分かりますね。

- Plain: 十分に多様性のある任意の質問
- Few-Shot: 複数の質問と回答を含む指示
- User-based: OpenAI の API のユースケース

ステップ 4 で強化学習を使っていると言いましたが、強化学習がどのような手法なのかについてざっくり説明しておくと、エージェントと呼ばれるもの（ゲームのキャラクターみたいに思っておいてください。）が環境内（ゲームの画面みたいに思っておいてください。）を行動するというふうに問題を捉え、エージェントがより良い行動をするように最適化をするような手法です。

ここで強化学習の説明と ChatGPT の強化学習によるモデルの学習を対応付けると、モデルのパラメータが環境に対応しており、エージェントが行動するということがモデルのパラメータを変更することに対応しています。そして、報酬モデルでの出力結果でより人間らしい解答だと判断されるようなパラメータの変更というのと良いエージェントの行動というのが対応づいているわけです。

強化学習で PPO を使っているということで、PPO がどのような強化学習なのかが気になる方いるかもしれませんので、ざっくり説明しておきます。PPO の細かい仕組みの話も大事だとは思うのですが、今回はそこまで踏み込まずに方策的勾配法の改良版で学習を安定させるために最適化の方法を工夫したもの、とだけ説明しておいて方策的勾配法がどのようなものなのかだけ説明をしておきます。

方策的勾配法の説明をする前に少し強化学習の用語の導入をします。強化学習で行動がどれだけ良いのかを表す値のことを報酬と言い、ある戦略に基づいてある状態において行う行動に関する確率分布を方策と言います。また、ある状態から i 回目の行動で得られる報酬を表す確率変数を R_i として、その状態における収益は $\mathbb{E}(R_1 + \gamma R_2 + \gamma^2 R_3 + \dots)$ (\mathbb{E} は期待値を表しています。 γ は $0 < \gamma < 1$ を満たす定数です) と定義します。収益とはある状態から行動を繰り返すことによってこれから先に得られる報酬の和のようなものです。ただし、より未来に行う行動に対する報酬は軽くみなします。これらの用語を用いて方策的勾配法を説明すると、方策的勾配法とは、その方策に従うと収益が最大になるような方策を直接モデリングする手法になります。方策的勾配法は比較的学習が安定しにくいのですが、PPO では工夫を行うことによってより安定した学習がしやすくなっています。

3.4 終わりに

ChatGPT の仕組みについてざっくり説明してきましたがいかがだったでしょうか。今回はできるだけ前提知識を要求せず、かつできるだけ数式を出さずに ChatGPT がどのような仕組みなのかについて説明を試みました。もし、興味を持った方がいれば是非ご自身で調べて頂けたら幸いです。読んでいただきありがとうございました。

第4章

言語モデルのプロンプトエンジニアリング

4.1 プロンプトとは

「プロンプト」とは、言葉や文を通じて、特定の情報や反応を引き出すための指示や質問のことと意味します。たとえば、コンピュータのプログラムを使って何かを実行する際、指示を与えるテキストやコマンドが「プロンプト」となります。特に、言語モデルにおいては、ユーザーがモデルに与える質問や指示がプロンプトとして働き、それを基にモデルが回答を生成します。同じ内容の質問でも、プロンプトを工夫することで回答精度が上がるることが知られています。さっそく次からは言語モデルのプロンプトとして用いられる考え方をいくつか見てていきましょう！

4.2 Few-shot prompting

この方法は、考え方の例を与えることでモデルに望む出力形式で回答してもらうものです。的確な例示を入力してあげることで回答の精度が上がります。例えば、以下のシンプルな通常のプロンプトだと 前後に不要な「”大規模言語モデル”を英語に翻訳すると、～となります。」の部分も回答として出てきますが、Few-shot prompting を加えるとこれが改善することが分かります。^{*1}

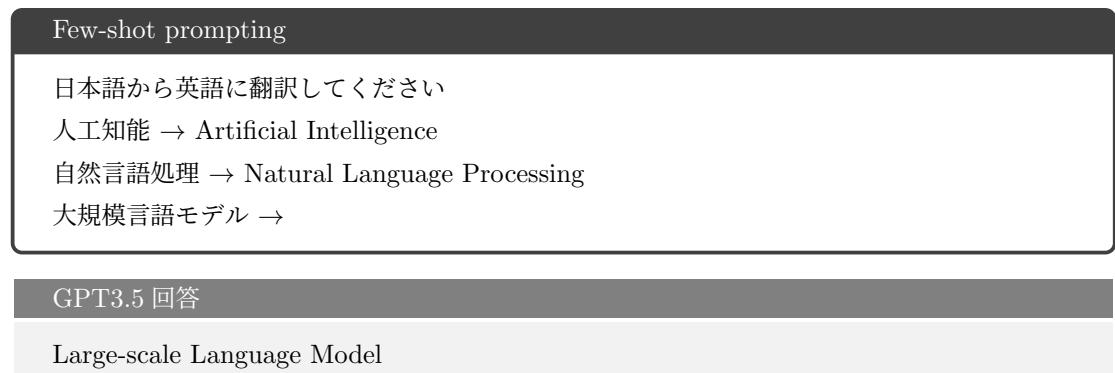
通常のプロンプト (Zero-shot prompting)

日本語から英語に翻訳してください
大規模言語モデル →

GPT3.5 回答

”大規模言語モデル”を英語に翻訳すると、”Large-scale language model”となります。

^{*1} GPT3.5 の回答の一例です。GPT4 はじめ他の言語モデルでは異なる結果となる場合があります。以降に示される例も同様です。



Few-shot Promptingを行うと精度が向上する理由として、In-Context Learning（文脈内学習）が挙げられます。これは与えられた文脈から学習することを意味しています。しかし、ここでの「学習」とはモデルのパラメータを更新することではなく、文脈を理解して回答を生成することを指しているので注意が必要です。図 4.1 によると、与える例の数を増やせば増やすほど回答精度が上がる事が示されており、大規模なモデルであるほど正解率の伸びが大きいことが分かります。

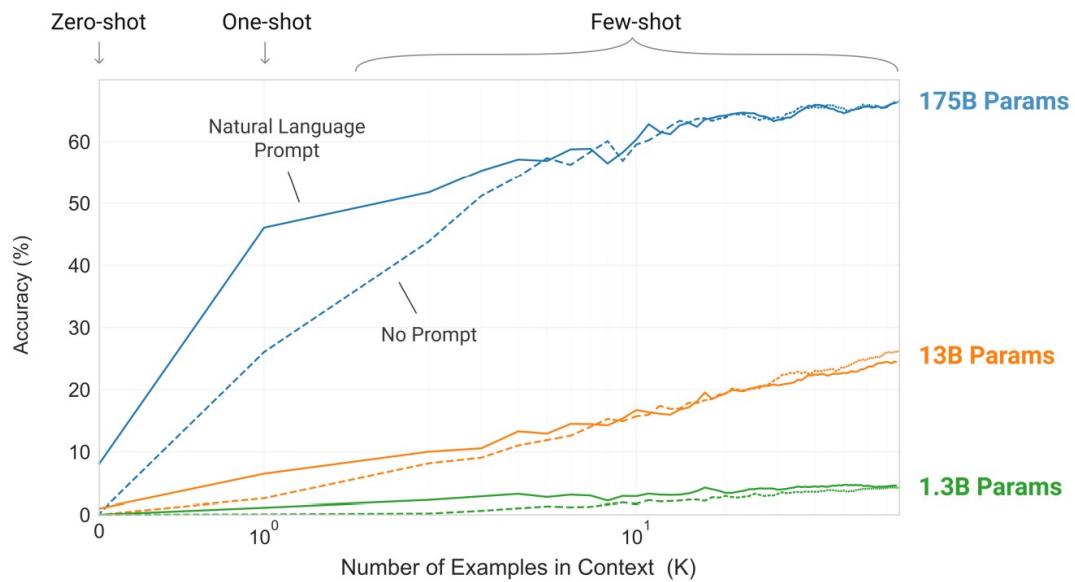


図 4.1: プロンプトに与える例の個数と回答精度の関係。[19] より引用

4.3 Chain of Thoughts

思考過程を明示的に答えさせることで、モデルの性能を向上させるアプローチです。そのなかでも今回はよく使われる2つをご紹介します。

4.3.1 Few-shot Chain of Thoughts

上記 Few-shot prompting で入力する”例”のかわりに、問題の思考過程をステップを踏んで順番に指定することで性能が向上します。論文 [20] の Figure1, Figure3 でもさまざまな例が紹介されています。

通常のプロンプト

問題:

ある長方形の庭は、長辺の幅が短辺の幅よりも 2 メートル長いです。庭の面積は 72 平方メートルです。庭の寸法を求めなさい。

GPT3.5 回答

この問題を解決するために、以下のステップを実行します。

1. 長方形の庭の幅を「 x 」としましょう。
2. 長辺の幅は短辺の幅よりも 2 メートル長いので、長辺の幅は「 $x + 2$ 」となります。
3. 庭の面積は長さと幅の積で求めることができます。したがって、面積は「 $x * (x + 2)$ 」です。
4. 問題文によれば、庭の面積は 72 平方メートルです。したがって、「 $x * (x + 2) = 72$ 」という方程式を得ます。

この方程式を解くために、まず方程式を展開し、次に解を求めます。

$$\begin{aligned}x * (x + 2) &= 72 \\x^2 + 2x &= 72\end{aligned}$$

これを x に関する 2 次方程式として整理します。

$$x^2 + 2x - 72 = 0$$

この 2 次方程式を因数分解するか、クアドラティックフォーミュラを使用して解を求めることができます。因数分解を行うと、次のようになります。

$$(x + 9)(x - 7) = 0$$

したがって、 x の値は 7 または -9 ですが、庭の幅が負の値になることはありませんので、 $x = 7$ メートルです。庭の寸法は次のようになります。

- 長辺の幅： $x + 2 = 7 + 2 = 9$ メートル
- 短辺の幅： $x = 7$ メートル

したがって、庭の寸法は長辺が 9 メートル、短辺が 7 メートルです。

Few-shot Chain of Thoughts

問題:

ある長方形の庭は、長辺の幅が短辺の幅よりも 2 メートル長いです。庭の面積は 72 平方メートルです。庭の寸法を求めなさい。

ガイドステップ:

ステップ 1: 短辺の幅を x メートルとする。

ステップ 2: 幅に基づいて長さを表現する。

ステップ 3: 面積の方程式を設定する。

ステップ 4: 設定した方程式を二次方程式の解の公式を使って解く。

ステップ 5: x が負でない寸法をもたらす解を選ぶ。近似せず分数のまま処理する。

ステップ 6: 選んだ解に基づいて長さと幅を見つける。近似せず分数のまま処理する。

ステップ 7: 長方形の寸法を表示。

GPT3.5 回答

問題を解決するために、以下のステップに従って進めましょう。

ステップ 1: 短辺の幅を x メートルとする。

短辺の幅を x としたとき、長辺の幅は $x + 2$ メートルとなります。

ステップ 2: 幅に基づいて長さを表現する。

長辺の幅が $x + 2$ メートルであるため、長辺の長さは $x + 2$ メートルとなります。

ステップ 3: 面積の方程式を設定する。

庭の面積は 長さ × 幅 なので、面積の方程式は次のように表現できます。

$$\text{面積} = \text{長さ} \times \text{幅}$$

$$72 = (x + 2) \times x$$

ステップ 4: 設定した方程式を二次方程式の解の公式を使って解く。

$$72 = x^2 + 2x$$

この方程式を解くために、二次方程式の解の公式を使用します。

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

ここで、 $a = 1$ 、 $b = 2$ 、 $c = -72$ です。これらの値を代入して計算します。

$$x = \frac{-2 \pm \sqrt{2^2 - 4 \times 1 \times (-72)}}{2 \times 1} = \frac{-2 \pm \sqrt{4 + 288}}{2} = \frac{-2 \pm \sqrt{292}}{2}$$

ステップ 5: x が負でない寸法をもたらす解を選ぶ。近似せず分数のまま処理する。

$x = (-2 + \sqrt{292})/2$ と $x = (-2 - \sqrt{292})/2$ の 2 つの解がありますが、庭の寸法は正の値である必要があるため、 $x = (-2 + \sqrt{292})/2$ を選びます。

ステップ 6: 選んだ解に基づいて長さと幅を見つける。近似せず分数のまま処理する。

短辺の幅 $x = (-2 + \sqrt{292})/2$ メートルを使用して、長辺の幅を計算します。

$$\text{長辺の幅} = x + 2 = \frac{-2 + \sqrt{292}}{2} + 2 = \frac{-2 + \sqrt{292} + 4}{2} = \frac{2 + \sqrt{292}}{2} \text{メートル}$$

ステップ 7: 長方形の寸法を表示。

$$\text{短辺の幅} = \frac{-2 + \sqrt{292}}{2} \text{メートル}$$

$$\text{長辺の幅} = \frac{2 + \sqrt{292}}{2} \text{メートル}$$

したがって、長方形の寸法は短辺の幅が $(-2 + \sqrt{292})/2$ メートルで、長辺の幅が $(2 + \sqrt{292})/2$ メートルです。

このように、手順を明確に指示してあげることで正解の回答を引きだすことができました。

4.3.2 Zero-shot Chain of Thoughts (通称: step by step)

「Let's think step by step」というフレーズを入れるだけで性能が向上するとされ、論文 [21] で提案されました。

通常のプロンプト

質問: あなたは学校のイベントを開催するために合計で\$3700 の予算を与えられています。椅子、テーブル、および音響システムをレンタルする必要があります。音響システムセットは一つ絶対に必要です。テーブルと椅子はセットで、テーブル1つにつき椅子1つ必要です。レンタル料は以下の通りです。以下のアイテムのレンタル料が与えられています。

- 椅子は\$2
- テーブルは\$10
- 音響システムセット（椅子7つ含む）は\$100

出席者全員に食事のための椅子とテーブルを確保しながら、座席のキャパシティを最大化するため、これらのアイテムの間で予算すべて使うとして、どのように割り当てるることができますか？また音響システムもあわせて合計の座席（椅子）の数はいくつですか？

GPT3.5 回答

まず、与えられた予算内でできるだけ多くの座席（椅子）を確保するために、どれだけの椅子とテーブルを購入できるか計算しましょう。

1. 音響システムセットを購入するために\$100 を使います。このセットには7つの椅子が含まれています。
2. 残りの予算は\$3700 - \$100 = \$3600 です。
3. 椅子の個数をできるだけ多く購入するために、\$3600 を椅子の単価である\$2 で割りま

す。\$3600 / \$2 = 1800 個の椅子を購入できます。

4. テーブルと椅子はセットで購入する必要があり、テーブル 1 つにつき椅子 1 つが必要です。したがって、1800 個の椅子に対応するテーブルも必要です。

したがって、合計の座席（椅子）の数は音響システムセットに含まれる 7 つの椅子に、さらに 1800 個の椅子を加えたものです。

合計の座席数 = 7 (音響システムセットに含まれる椅子) + 1800 (購入した椅子) = 1807 席
したがって、予算内で最大の座席数を確保するために、1807 席の椅子を購入し、それに対応するテーブルを購入することができます。また、音響システムセットも購入して、合計の予算内ですべてのアイテムを確保できます。

step by step

質問: あなたは学校のイベントを開催するために合計で\$3700 の予算を与えられています。椅子、テーブル、および音響システムをレンタルする必要があります。音響システムセットは一つ絶対に必要です。テーブルと椅子はセットで、テーブル 1 つにつき椅子 1 つ必要です。
レンタル料は以下の通りです。

- 椅子は\$2
- テーブルは\$10
- 音響システムセット（椅子 7 つ含む）は\$100

出席者全員に食事のための椅子とテーブルを確保しながら、座席のキャパシティを最大化するために、これらのアイテムの間で予算すべて使うとして、どのように割り当てるることができますか？また音響システムもあわせて合計の座席（椅子）の数はいくつですか？

Let's think step by step.

GPT3.5 回答

Step 1: 音響システムセットのレンタル

最初に、音響システムセットをレンタルする必要があります。これは\$100 です。このセットには椅子 7 つが含まれています。残りの予算は\$3700 - \$100 = \$3600 です。

Step 2: テーブルのレンタル

次に、テーブルと椅子を確保します。テーブル 1 つにつき椅子 1 つ必要です。したがって、テーブルと椅子のセットをレンタルすることを考えます。

1 つのテーブルと 1 つの椅子のセットのレンタル料は\$10 + \$2 = \$12 です。

予算を最大限に活用するため、可能な限り多くのテーブルと椅子のセットをレンタルします。
したがって、残りの予算\$3600 を\$12 で割ります。

$$\$3600 \div \$12 = 300$$

これにより、300 のテーブルと 300 の椅子のセットをレンタルできます。

Step 3: 合計座席数の計算

音響システムセットには 7 つの椅子が含まれているため、これらの 7 つの椅子も含めて合計座席数を計算します。

テーブルと椅子のセットが 300 セットあるので、テーブルと椅子のセットから得られる椅子の数は $300 \text{ セット} \times 1 \text{ つのセットあたりの椅子の数 (1 つ)} = 300 \text{ 椅子}$ です。

また、音響システムセットからの椅子の数は 7 つです。

合計座席数は 300 椅子（テーブルと椅子のセットから）+ 7 椅子（音響システムセットから）= 307 席です。

したがって、合計の座席数は 307 席です。

このように、Let's think step by step. を加えることで正解の回答を引きだすことができました。

4.4 Self-translate

このアプローチは、入力を一度英語に翻訳して言語モデルに入力し、出力を得た後に再び日本語に翻訳する方法です。ChatGPT は英語での処理に強く、日本語のまま入力するより高いパフォーマンスを期待できるからです。なぜ英語で入力するといいのかについては、[22] によると次のような理由が考えられそうです。

1. データの豊富さ: 英語はインターネット上で最も豊富に使われる言語の一つであり、その結果、英語のトレーニングデータセットは非常に大きく、多様性があります。これにより、モデルはより正確な英語の表現を学習することができます。
2. リソースの偏り: 機械学習モデルは、訓練に使われるデータに大きく依存しています。多くの場合、研究や産業で使われるデータセットは英語中心で、他の言語に比べてはるかに多くのリソースが存在します。
3. 研究の焦点: 英語は多くの機械学習研究の主要な対象言語です。そのため、英語に関する技術的な改善やアルゴリズムの進化は他の言語よりも速いペースで進行しています。
4. 技術的なインフラ: 英語を処理するための先進的なツールやシステムが存在し、これらは英語の精度を高めるのに寄与しています。
5. 評価基準: 多くのベンチマークや評価基準が英語で設定されており、研究者はこれらの基準に合わせてモデルを最適化するため、英語での性能が優れています。

得意な言語に訳せば急にわかるなんて、まるで本当に人間みたいですよね。ちなみに日本語が得意な言語モデルも開発もあります。日本語コーパスという、自然言語の文章や使い方を大規模に収集し、コンピュータで検索できるよう整理されたデータベースを使い訓練が進められています。

4.5 Take a Deep Breath

「Take a Deep Breath」というフレーズをプロンプトに加えるだけでモデルの性能が向上するとされています [23]。step by step と同様に簡単でわかりやすいですね。step by step でもそうなのですが、なぜ人間のような肉体がない言語モデルでもこのようなプロンプトが有効なのでしょうか。言語モデルが行う推論は、本やウェブから抽出された大量の言語フレーズのデータセットから学習し、その学習結果をもとに行われるものです。[24] によると、このデータセットには、Q&A フォーラムのようなものも含まれており、そこには「Take a Deep Breath」や「step by step」などといったフレーズが多く含まれているそうです。そして、これらのフレーズが含まれる文章データでは往々

にしてより慎重で、問題を丁寧に推論した考え方と策が書いてあります。これらのフレーズが、言語モデルが訓練中に吸収されることで、より良い答えを引き出したり、推論や問題解決のより良い例を生成したりするのに役立っているのかもしれません。

4.6 Echo Prompt

問い合わせを言い換えさせるように指示することで、モデルの性能が向上するとされています。また、先述の step-by-step と組み合わせるとさらに性能が向上することも示されており（図 4.2）、人間と同じように、問い合わせを言い換えることで問い合わせへの理解度が上がるから、あるいは問い合わせの内容をモデルに思い出させることで、より適切な回答が得られるからではないかと考えられています。

EchoPrompt?	Stage-1 Prompt	GSM8K	SVAMP	MultiArith	SingleOp
Zero-shot					
✗	-	16.4	66.8	31.0	91.6
✓	Let's repeat the question. “	20.7(+4.3)	74.7(+7.9)	48.5(+17.5)	91.8(+0.2)
✓	Let's reiterate the question. “	19.7(+3.3)	73.4(+6.6)	51.0(+20.0)	93.0(+1.4)
✓	Let's restate the question. “	19.2(+2.8)	74.6(+7.8)	47.7(+16.7)	89.6(-2.0)
✓	Let's summarize the question. “	20.6(+4.2)	73.2(+6.4)	48.8(+17.8)	93.7(+2.1)
Zero-shot-CoT					
✗	Let's think step by step.	49.3	66.5	76.0	82.9
✓	Let's repeat the question and also think step by step.	44.6(-4.7)	74.7(+8.2)	70.9(-5.1)	92.3(+9.4)
✓	Let's reiterate the question and also think step by step.	51.1(+1.8)	73.9(+7.4)	78.7(+2.7)	92.4(+9.5)
✓	Let's repeat the question and also think step by step. “	42.0(-7.3)	60.4(-6.1)	78.1(+2.1)	88.3(+5.4)
✓	Let's restate the question and also think step by step.	47.0(-2.3)	73.9(+7.4)	79.3(+3.3)	90.2(+7.3)
✓	Let's summarize the question and also think step by step.	49.9(+0.6)	74.2(+7.7)	75.8(-0.2)	90.9(+8.0)

Table 1: **Code-davinci-002: Arithmetic reasoning** Evaluation of EchoPrompt on various prompt templates. All the prompts improve the performance in zero-shot setting. However, we find that only the prompt “Let's reiterate the question and also think step by step.” consistently outperforms baseline Zero-shot-CoT.

図 4.2: 様々な Echo Prompt に対する回答精度。[25] より引用

step by step のみ

Q: ジェイコブには切手のコレクションがあります。もし彼がそのコレクションに 12 枚の切手を追加すれば、その数は妹の切手の数の 2 倍になります。妹が持っている切手は、合計 1000 円なるように買った 50 円の切手と、親にもらった 10 枚と、友達と交換した 4 枚です。もともとジェイコブは何枚の切手を持っていたのでしょうか？ステップバイステップで考えてください。

GPT3.5 回答

問題を解くために、いくつかの情報を整理しましょう。この問題を解決するために、ステップバイステップで進めてみましょう。

1. 妹の切手の数を計算します。
 - 50 円の切手を買ったので、 $50 * 10 = 500$ 円分の切手があります。
 - 親からもらった切手は 10 枚あります。
 - 友達と交換した切手は 4 枚あります。
 - したがって、妹は合計で 10 枚 + 4 枚 = 14 枚の切手を持っています。

2. ジェイコブが 12 枚の切手を追加すると、その数は妹の切手の数の 2 倍になります。つまり、ジェイコブが持っている切手の数は、妹の切手の数の $2 \times 14 + 12 = 28 + 12 = 40$ 枚です。
3. 妹の切手の数が 14 枚なので、ジェイコブが持っている切手の数は $2 \times 14 + 12 = 28 + 12 = 40$ 枚です。

したがって、ジェイコブはもともと 40 枚の切手を持っていました。

step by step + Echo Prompt

Q: ジェイコブには切手のコレクションがあります。もし彼がそのコレクションに 12 枚の切手を追加すれば、その数は妹の切手の数の 2 倍になります。妹が持っている切手は、合計 1000 円なるように買った 50 円の切手と、親にもらった 10 枚と、友達と交換した 4 枚です。もともとジェイコブは何枚の切手を持っていたのでしょうか？まず回答の最初に質問をまとめ、自分の言葉で言い直してください。その後にステップバイステップで考えてください。

GPT3.5 回答

質問の要約: ジェイコブが持っている切手の数を求める問題です。ジェイコブは切手のコレクションを持っており、12 枚の切手を追加すると、その数は妹の切手の数の 2 倍になります。妹は 50 円の切手を合計 1000 円分買い、親から 10 枚もらい、友達から 4 枚交換しました。ジェイコブが最初に何枚の切手を持っていたかを求めます。

ステップバイステップで考えると:

1. 妹が持っている切手の合計金額は、50 円切手を合計 1000 円分買ったことから、 $1000 \text{ 円} / 50 \text{ 円} = 20$ 枚の切手があることがわかります。
2. 妹は親から 10 枚の切手をもらい、友達から 4 枚の切手を交換しました。したがって、妹は合計で $10 \text{ 枚} + 4 \text{ 枚} = 14$ 枚の切手を持っています。
3. ジェイコブが 12 枚の切手を追加すると、その数は妹の切手の数の 2 倍になるという条件があります。つまり、ジェイコブが持っていた切手の数を x とすると、 $x + 12 = 2 * (20 + 14)$ です。
4. 式を解いて x を求めます。まず、右辺を計算します。 $20 + 14 = 34$ です。次に、 $2 * 34 = 68$ です。したがって、式は $x + 12 = 68$ となります。
5. x を求めるために、両辺から 12 を引きります。 $x = 68 - 12$ となります。
6. x を計算すると、 $x = 56$ です。ジェイコブは最初に 56 枚の切手を持っていました。

したがって、ジェイコブが最初に持っていた切手の数は 56 枚です。

このように、質問を言い直すことで要点が整理され正解の回答を引き出せました。

4.7 まとめ

言語モデルのプロンプトについてご説明しました。はじめは慣れないですが、うまく期待した答えがこないときに、同じ質問内容でもプロンプトの工夫により回答がよくなった瞬間がやっていて面白いところですよね！今回の Few-shot prompting、Chain of Thoughts、Self-translate、Take a

Deep Breath、EchoPrompt、以外にも様々な手法が研究されています。これらの手法を適切に利用することで、言語モデルの性能を最大限に引き出すことができます。みなさんもぜひチャレンジしてみてください！

第5章

生成系 AI を巡る法的・倫理的課題

2023年に大きく話題になったChatGPTは、企業が競って開発や利活用をはじめ、レッドオーシャン化が進んでいます。その反面、法的・倫理的観点から多くの懸念点が挙げられており、課題の解決が求められています。開発が急速に進んでいる分野でもあるため、大規模言語モデルひいては生成系AIなどに対する倫理的価値観が定まっておらず、各国、各企業が独自にガイドラインを制定しているという状況です。

この章では現状の生成系AIの法的・倫理的问题と、各組織の動向を記載します。

5.1 開発企業に関する課題

5.1.1 個人情報保護

生成系AIの開発には大量のデータが必要で、Webサイトや書籍、ニュースなどの情報をAIに入力し、学習データとして利用します。そのデータの入手元について著作権法や個人情報保護法との関係性と、その法的根拠が論点になっています。また、こうした問題は個人情報保護法上適法かどうかではなく、そもそも学習のために大量のデータを勝手に利用するということ自体がプライバシーの観点から問題であるという意見も存在します。イタリアでは個人データ保護機関であるGDPRが、イタリア内でChatGPTの利用を制限するという緊急措置を取ったことが話題になりました[26]。

5.1.2 プライバシーのリスクとセキュリティ

また、入力されたデータの安全管理の確保も生成系AIの開発企業にとって重要な課題です。ユーザーから提供されたデータには、個人情報や機密情報などのプライバシーに関わる情報が含まれることがあり、このようなデータを適切に保護しない場合、プライバシー侵害のリスクが高まります。

開発企業にはユーザーのプライバシーを尊重し、適切にデータを管理する責任があります。

5.1.3 バイアス

他にも、学習データに内在するバイアスや違反行為を助長する発言が、生成系AIの出力段階で再現されてしまう可能性があることが指摘されています。例えば、Web上のデータから、人種差別や男女差別の意図を読み取り差別的な発言をしたり、爆弾の作り方を説明してしまうなどの違反行為を助長する発言をしてしまうことが論点としてあげられています。

5.2 ユーザーに関する課題

生成系 AI を利活用するユーザーにとって多くの論点が存在します。

5.2.1 著作権や知財

生成系 AI と著作権にまつわる問題として、第三者の著作権の侵害可能性が挙げられます。例えば ChatGPT を使って小説を書く場合、その小説が他の著作物と酷似していることは可能性として否定できません。その場合、著作権の侵害が認められるのか、またどのようなときに誰に責任があるのかといったことが論点となります。そのような議論を抜きにしても、第三者の著作物と成果物が酷似している場合には、著作権者との間でトラブルが生じる可能性があり、利用者は注意を払わなければなりません。日本と海外では法規制が異なっており、海外の方が厳しいとされているので、その点も踏まえる必要があります。

他にも、生成系 AI を用いた生成物の著作該当性も議論に上がります。生成系 AI を用いて作成した創作物が他人に勝手に利用された場合、利用者は何を主張できるのかが問題となります。

5.2.2 プライバシーと個人情報保護

一般的な個人情報だけでなく、企業秘密や未公開情報、国家機密などを大規模言語モデルにプロンプトとして入力することのリスクが指摘されています。

また、大規模言語モデルの利用の際に、個人情報保護法に従う必要があります。個人情報保護法は、個人情報の収集、処理、保護に関する法律です。特に、個人情報になるような内容を入力する場合、その利用が利用目的に合致しているかに論点があります。つまり、個人情報保護の観点から、元々定められた目的に沿った利用方法を取っていれば法的懸念は回避されるそうです [27]。また、第三者提供の観点からも、第三者のサービス提供者が個人情報を利用しないことになっている場合は、第三者提供に当たりません。

また、入力したデータが企業の機密情報の場合、自社以外のサーバーに機密情報が保存されるという問題があり、保存先での情報の管理方法やセキュリティを検討し、利用の可否を検討することが必要です。

5.2.3 業法との関係

免許や資格等が必要な特定の業種の業務を自動化するような生成系 AI を提供した場合には業法との抵触も問題になります。

5.2.4 出力されたデータの不適切性

例えば、ChatGPT を利用したサービスを外部向けに提供した場合に、出力された発言にプライバシー権侵害や名誉毀損等になるような発言が含まれていた場合、また差別発言や誤情報を提供した場合の法的・倫理的責任も問題になります。こういった問題発言が出されないような制御はある程度可能ですが、プロンプトの入力次第で回避されることができ、完全に防ぐことはできないかもしれません。

5.3 AI自体に関わる課題

5.3.1 仕事を奪う

生成系AIについては、仕事への影響が特に問題視されています。特にイラストレーターや記者やライターの仕事を奪うというもので、世界中で大きな問題とされています。これまでもAIによって仕事が代替されることに対する懸念はされてきましたが、昨今の生成系AIに関しては特に強い批判が当たられています。

AIの仕事に対する影響については、AIとの協働の重要性が指摘されることも多いです。たとえば、文献の調査や議事録の作成、報告書の記載など、AIをツールとして扱うことで、仕事を効率化し人間にしかできないことに注力するべきであるという意見もあります。

5.3.2 環境上の課題

スタンフォード大学の報告書[28]によれば、GPT-3の学習に必要な二酸化炭素排出量は502トンに上り、アメリカの1人の1年間の排出量の28倍にのぼるそうです。LLMは基本的に巨大なモデルを膨大な量のデータで学習することになるため、電力消費は増大すると予測できます。また推論過程における電力消費も同様に大きくなり、環境上の問題としても挙げられています。

5.4 法規制やガイドラインに関する動向

2023年の3月に、イタリアの個人データ保護機関(GPDP)は、EUが定める規則を満たしていないとして、イタリアの個人情報の使用を制限するという緊急措置を命じました。また、同時にOpenAIに対して改善策の報告を命じました[26]。

その後、OpenAIは、オプトアウト(個人情報を利用されないようにユーザーが選択できる)機能の提供やプライバシーポリシーの拡充、年齢制限などを追加し、4月末にイタリアでの利用が再開しました[29]。その後、欧州データ保護委員会が設置したアドホックタスクフォースの下で継続して事実調査が行われています。

また、イギリスの個人データ保護機関であるICOは、2023年3月15日に生成系AIを開発または利用する企業に対して、データ保護に留意してAIを開発するためのガイダンス「AIとデータ保護に対する方針」を公表しています[30]。

アメリカでも、2022年10月4日に、人工知能の開発の際に考慮すべき5つの原則をまとめた「AI権利章典の青写真」を公開しています[31]。

引用

- **Safe and Effective Systems** : You should be protected from unsafe or ineffective systems.
- **Algorithmic Discrimination Protections** : You should not face discrimination by algorithms and systems should be used and designed in an equitable way.
- **Data Privacy** : You should be protected from abusive data practices via built-in protections and you should have agency over how data about you is used.
- **Notice and Explanation** : You should know when an automated system is being

used and understand how and why it contributes to outcomes that impact you.

- **Human Alternatives, Consideration, and Fallback** : You should be able to opt out, where appropriate, and have access to a person who can quickly consider and remedy problems you encounter.

邦訳

- **安全で効果的なシステム** : 安全でないシステムや効果のないシステムからは保護される必要があります。
- **アルゴリズム差別からの保護** : アルゴリズムによる差別を許すべきではなく、システムは公平なやり方で利用と設計がなされなければなりません。
- **データプライバシー** : 組み込みの保護機能によって不正なデータ行為から保護される必要があり、自分に関するデータがどのように使用されるかを決定する権限を持つ必要があります。
- **注意と説明** : 自動化システムがいつ使われているのかを理解し、それがどのようにして、またはなぜ働いているのかを理解する必要があります。
- **人間による代替手段、検討、フォールバック** : 必要に応じてオプトアウトでき、発生した問題を迅速に検討して解決できる人にアクセスできる必要があります。

中国でも、2023年8月15日に生成系AIを規制する管理規則である「生成AIサービス管理暫定規則」を施行しました。生成AIの提供者や利用者に「社会主義核心価値観」の堅持を要求し、「国家政権転覆を扇動し、社会主義制度を打倒し、国家の安全や利益に危害を加える内容を生成してはならない」と強調しています。ChatGPTの利用が制限されている中国において、こうした措置の法的根拠を整えています。

日本では、政府の個人情報保護委員会が2023年の6月2日にOpenAIを行政指導し、個人情報の取得の方法に関して注意喚起しました。しかし、その後日本の目立った動向はなく、EUや中国、米国などと比べて規制はゆるいとされています。

また5月に行われたG7広島サミットでは、広島AIプロセスと呼ばれるAIの国際ルールについて年内に見解を得ることで合意しました。

第 6 章

最近の LLM の動向

OpenAI が 2022 年の 11 月に ChatGPT を公開して以降、大規模言語モデル (LLM) の開発が非常に活発になり、ChatGPT のほかにも毎週何百もの新しいモデルが発表されています。この百花繚乱の LLM の時代に動向を一つにまとめるのも難しいですが、現時点 (10 月 1 日) での大規模言語モデルの動向について記載します。

6.1 LLM 開発の流れ

LLM は、自然言語処理の分野で使用される、巨大なデータセットで訓練された機械学習モデルの一種です。テキストデータを理解し、生成するために使用され、さまざまな自然言語タスクに適用することができる事が特徴としてあげられます。1990 年代にゆるやかにはじまった第 3 次 AI ブームの中でディープラーニングと自然言語処理の技術が発展し、膨大な学習データによって文章のパターンやルールを獲得できるようになりました。

特に注目されたのは、2022 年の 11 月に公表された ChatGPT です。まさに人間のように会話をすることができ、マスメディアでも大きく取り上げられ、話題になりました。

その後様々な企業が大規模言語モデルの構築に参与するようになり、まさにカオスと言えるでしょう。

6.1.1 OpenAI

2018 年に OpenAI が GPT を発表し、大規模言語モデルの先駆けとなりました [32]。

GPT はテキストを与えられたあと首尾一貫した自然言語を生成できる深層学習ツールで、800 万の Web ページを学習し、機械が人間の言語を理解し、人間のような応答を生成できるようになりました。その後 GPT-2、GPT-3 が開発され、より複雑な言語を理解できるのが特徴とされています。

2023 年には GPT-4 が発表されました。表 1 に GPT-3.5 と GPT-4 の比較を示します。GPT-4 は、GPT-3.5 と比べてさらに大きなデータセットでトレーニングされました。司法試験の上位 10 % に入るほど性能が格段に向上し [33]、画像などのインプットも受け画像の内容を理解することができるようになりました。また、間違いの低減や危険度の低下などの精度も向上しています。なお、パラメータ数に関しては公式で発表されていないので、推定として記載しています。

比較項目	GPT-3.5	GPT-4
パラメータ数	(推定)3550 億	(推定)5000 億～1 兆
機能	文書作成	文書作成、画像入力による文章作成
最大入力トークン数	16,385 token	32,768 token
性能		司法試験上位 10 %の成績 日本語の精度上昇 画像の解釈

表 6.1: GPT-3.5 と GPT-4 の比較 (2023 年 10 月時点)

6.1.2 Google

OpenAI が最初に GPT を公開した数カ月後に、Google が BERT を発表しました。この BERT は自然言語処理のベンチマークで従来のスコアを超え、大きな成功を収めました。その後 Google は様々な大規模言語モデルを発表しています。以下ではそのいくつかを紹介します。

BERT

データセットの規模を増やし、精度を向上させた初期の言語モデルです。パラメータ数は 3.4 億で 2018 年に発表されました [34]。

LaMDA

Transformer をベースとして、対話に特化させたモデルです。2022 年に公開されましたが、パラメータ数は 1370 億となっています [35]。

PaLM

Transformer のパラメータ数を 5400 億まで拡大して高性能を実現したモデルです。2022 年に公開され、さらに高速かつ高性能な新たなモデル PaLM2 が 2023 年 5 月に公開されました [36]。

他にも Google は FLAN という言語モデルも開発しています。

6.1.3 Microsoft

Microsoft は、OpenAI にと手を組み、Azure や Bing に ChatGPT を組み込んでいます。とくにクラウドサービスの Azure で利用できる OpenAI Service や、Microsoft Copilot などがリリースされていて多くの企業が利用を開始しているようです [37]。その他にも Microsoft が開発する製品への生成系 AI の組み込みが期待されています。

6.1.4 Meta

Meta も他のビックテックと同様に大規模言語モデルを公開しています。特に 2023 年に公開された Llama は、最大 130 億という GPT-3 と比べて非常に少ないパラメータで構成されているのにもかかわらず、GPT3 と同等の性能を実現しています [38]。2023 年 8 月に Llama2 が公開され、更に性能が向上しました [39]。Llama の重要な点はオープンソースであることです。Llama を用いて新たな言語モデルを作成している団体も多く、開発競争を促す役割を担っています。

6.1.5 中国の動向

中国ではChatGPTの利用が国内で制限されています。中国政府は米国に頼らない開発環境を作ろうと模索している最中で、制度や開発環境が整いつつあります。アリババやテンセント、百度、各大学などが競って開発競争に参入していて、性能も高いものもあると言われています。

6.1.6 日本の動向

日本でも、大規模言語モデルの開発が進んでいます。NECやサイバーエージェントで日本語LLMが公開されました。どちらも自社サービスの活用を目的にしています。

またNICT(情報通信研究機構)や株式会社rinna、株式会社Preferred Networksも大規模なLLMの開発に取り組んでいます。

第 III 部

ゲーム AI

第7章

引き分けを目指す「忖度オセロ AI」

7.1 忖度オセロ AI を作った理由

将棋、囲碁、オセロなどのボードゲームにおいて、AI はすでに人間の実力を超えています。特にオセロはルールのシンプルさゆえに、1990 年代には AI は人間が勝てないレベルまで達していました [40]。プロのプレイヤーであれば、このような強い AI と対戦をし研究を重ねていくことで、自分の実力を向上させることができるでしょう。しかし初心者の場合、このような強い AI と対戦すると当然ながらあっという間に負けてしまいます。いい練習にはなるでしょうが、「オセロを楽しむ」というゲームの本質からは離れてしまいます。強い AI とは逆に、あえて弱い手を打ち相手を勝たせる「最弱オセロ AI」も存在します [41]。弱い AI であれば、初心者でも勝つことができて快感を味わうことができます。これならオセロを楽しむことができるかもしれません。

しかし、多くの人は圧勝・完敗するゲームよりも、接戦であるほうが面白いと感じるでしょう。つまり、相手が強すぎても弱すぎても面白くないのです。そこで私は、引き分けを目指す**忖度オセロ AI**を開発しました。

7.2 精度

まず、私の忖度オセロ AI がどれだけ引き分けになりやすいかを検証した結果を示します。図 7.1 は、ランダムに手を選ぶプレイヤーを相手に対戦したときの得点差の分布を示しています。1000 回対戦した結果、引き分けになったのは 499 回、2 点差以内に収まったのは 801 回で、24 点差以上つけられた対局はゼロでした。

7.3 忖度オセロ AI のしくみ

ここからは、私の忖度オセロ AI がどのようにして引き分けを目指すのかを説明します。ざっくり言うと、次のようなしくみで動いています。

1. 序盤～中盤：盤面から最終的な得点差を予測するニューラルネットワークを用いて、得点差が小さくなりやすい手を選ぶ。
2. 終盤：終局まで読み切り、得点差が小さくなりやすい手を選ぶ。

以下では、

- 盤面から最終的な得点差を予測するニューラルネットワークの学習方法

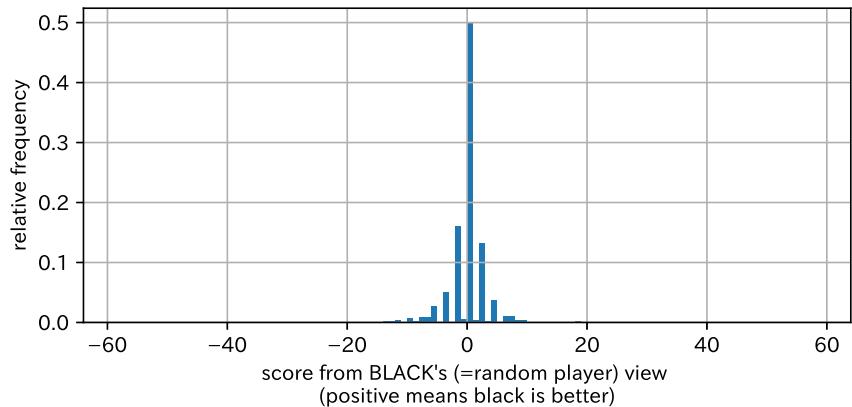


図 7.1: 1000 回ランダムプレイヤーと対戦させた結果

- 引き分けになりやすい手を選ぶための探索アルゴリズムについて、詳細を解説します。

7.4 評価関数の学習

対局を互角に進めるためには、盤面から最終的な得点差を予測する評価関数が必要です。評価関数の作り方にはいろいろな方法が考えられます。例えば、石の配置に関する特徴量を人間が設計し、それらを用いて評価値を計算する方法などが考えられます^{*1}。しかし、今回は特徴量を自動抽出できるニューラルネットワークを用いて評価関数を作成しました。ニューラルネットワークの中でも、画像系のタスクでよく用いられる畳み込みニューラルネットワーク (CNN) を採用しています。オセロの盤面も画像とみなせるため、CNN を用いることで高精度な評価関数を作成することができると考えられます。盤面は (チャンネル, 高さ, 幅)=(9, 8, 8) の 3 次元配列で表現してモデルに入力します。各チャンネルは次のような情報を表現しています。

- 第 1 チャンネル：黒石の位置を 1、それ以外を 0 で埋める
- 第 2 チャンネル：白石の位置を 1、それ以外を 0 で埋める
- 第 3 チャンネル：空白の位置を 1、それ以外を 0 で埋める
- 第 4 チャンネル：合法手の位置を 1、それ以外を 0 で埋める
- 第 5 チャンネル：各マスに打った場合にひっくり返る石の個数で埋める
- 第 6 チャンネル：「隅」「C^{*2}」「X^{*3}」をすべて 1 で埋める
- 第 7 チャンネル：すべて 1 で埋める
- 第 8 チャンネル：すべて 0 で埋める
- 第 9 チャンネル：黒番ならすべて 0 で埋め、白番ならすべて 1 で埋める

第 1・2 チャンネルの情報があれば十分なのではないかと思われるかもしれません、第 3 チャンネ

^{*1} 1997 年に当時の日本人チャンピオン村上健氏を打ち負かした AI 「Logistello」はこの方法を採用しました [42]。推論時間が短く済み、深く探索を行えるなどのメリットがあります。

^{*2} 隅と辺で隣り合ったマス

^{*3} 隅と斜めに隣り合ったマス

ル以降の情報も加えることで学習が進みやすくなると考えられます^{*4}。

ニューラルネットワークのアーキテクチャは以下のとおりです。AlphaGo のモデルを参考にしました[43]。Conv(...) は畳み込み層、Dense(...) は全結合層を表しています。出力は得点差であり、上限と下限が存在するので、最後の層には tanh を使用して -1 以上 1 以下の値を出力するようにしています。得点差 (-64~64) を知りたいときは出力値を 64 倍すればよいです。

- 第 1 層 : Conv(out=100, kernel=(5,5), padding=2) + ReLU
- 第 2~11 層 : Conv(out=100, kernel=(3,3), padding=1) + ReLU + BatchNormalization
- 第 12 層 : Conv(out=100, kernel=(3,3), padding=1)
- 第 13 層 : Conv(out=100, kernel=(1,1), padding=1)
- 第 14 層 : Dense(out=256) + ReLU
- 第 15 層 : Dense(out=1) + tanh

学習データは、世界最強レベルオセロ AI 「Egaroucid」による自己対戦の棋譜データ [46] を使用させていただきました。このデータセットには 200 万対局の棋譜が含まれており、これらを 1 エポック分学習させました。また、データ拡張も行っており、盤面の左右反転や 90 度回転を行うことで 1 つの盤面に対して 8 つの盤面を作成しています。

以上のようにして学習させた評価関数を使用すれば、図 7.2 のように盤面の形勢判断を行うことができます。

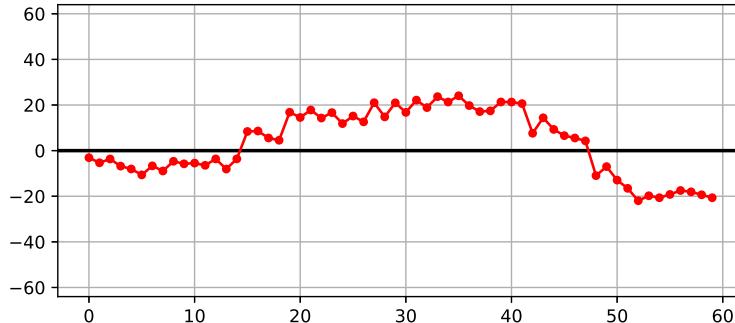


図 7.2: 評価関数の出力値の例

7.5 探索アルゴリズム

7.4 節で説明した評価関数を使えば、評価値 (=予想得点差) の絶対値が最小となる手を選んでいくことで、引き分けになりやすくなることが期待できます。しかし、評価関数の出力値はあくまで予測値であり、後の手の選び方によっては予測値と実際の得点差が大きく異なることもあります。実際、目前の評価関数の値を信頼して手を選んでいっても、引き分け率は 10% 程度にしかなりませんでした(ランダムプレイヤーと対戦)。

^{*4} AlphaGo における入力特微量を参考にしました[43][44]。第 7 チャンネルをすべて 1 で埋めるのは一見無意味に見えますが、これはパディング部分とオセロ盤部分を区別するために導入されていると考えることができます[45]。CNN では、特微量マップが小さくならないように入力特微量マップの周囲を 0 でパディングすることができます。このとき第 7 チャンネルでは、1 で埋められた 8×8 の外部は 0 でパディングされ、オセロ盤の大きさに関する情報を保つことができます。なお、すべて 0 で埋めるチャンネルの意図は不明です…。

そこで効果的なのが「探索(=先読み)」です。付度オセロAIではミニマックス法という探索手法を用いて引き分けになりやすい手を選んでいます。まず、図7.3のように現在の盤面からありうる全ての盤面を先読みし、最終的な得点差絶対値を計算します。場合の数が多くて時間がかかりすぎてしまう場合(序盤～中盤)は途中で探索を打ち切り、前節で解説した評価関数を用いて得点差絶対値を予測します。

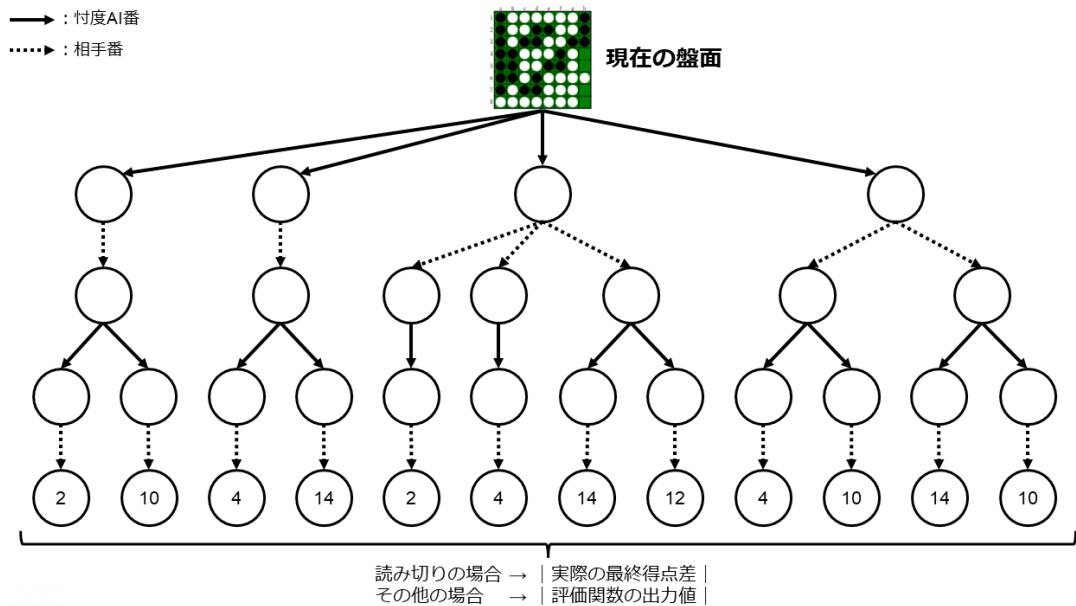


図7.3: 末端ノードに評価値を付与する

ナイーブに考えると、図7.3における一番下の得点差絶対値「2,10,4,...,10」の中から最小となるものを探し、そこに行きつくような手を選べばよいと思われるかもしれません。しかし、相手は自分にとって都合の良い手を打ってくれるとは限りません。

図7.4を見てください。最初のAI番で、得点差絶対値が最小となる「2」に行けるような手を選んだとしましょう。もし次の相手番が一番左の手を選んでくれれば、最終的な得点差絶対値は「2」となりますが、もし一番右の手を選ばれてしまうと、得点差絶対値は「10」となってしまいます。

そこで「得点差が大きくなるような手を相手が打ってきたとしても、なるべく最小の得点差になるような手を選ぶ」ことを考えます。これがミニマックス法の考え方です。図7.5に示すように、AI番では値が小さくなる手を選び、相手番では値が大きくなる手を選ぶと仮定して、各ノード(図中の丸のこと)に評価値を付与していきます。評価値を付けたいノードがAI番ならば、子ノードの評価値のうち最小値を評価値として付与し、逆に相手番ならば子ノードの評価値のうち最大値を付与します。このようにして、最終的に一番上に並んでいる4つのノードに評価値を付与することができます。図の例では、左から「2,4,12,10」と評価値がついているので、評価値最小の一番左の手を選べばよいことがわかります。

7.6 おわりに

実は、この引き分けを目指すAIは2023年春にKaiRAで開催したAIハッカソンでも開発しました。当時は、強化学習を用いるなどしていましたが、ランダムプレイヤーに対する引き分け率は

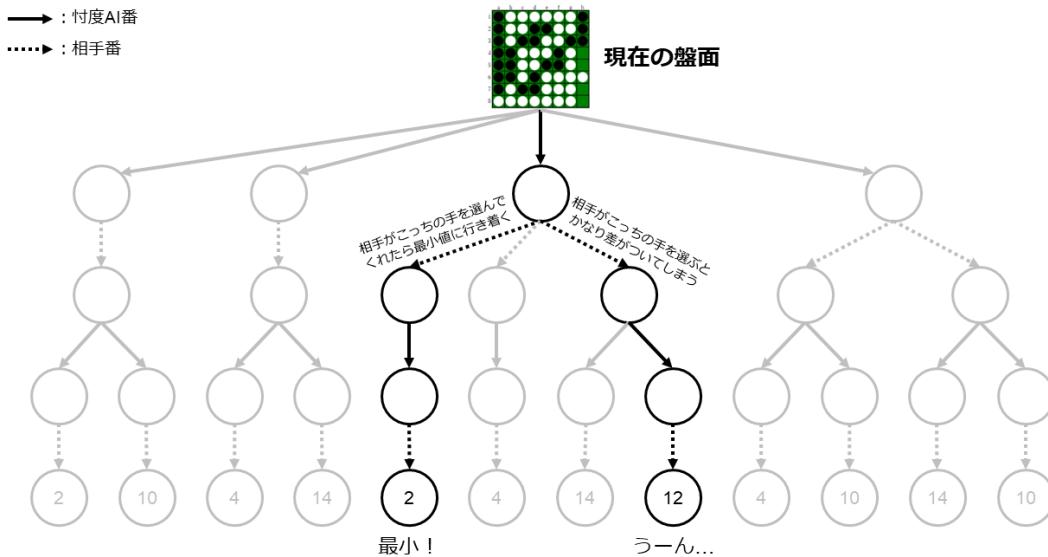


図 7.4: ナイーブな方法の問題点

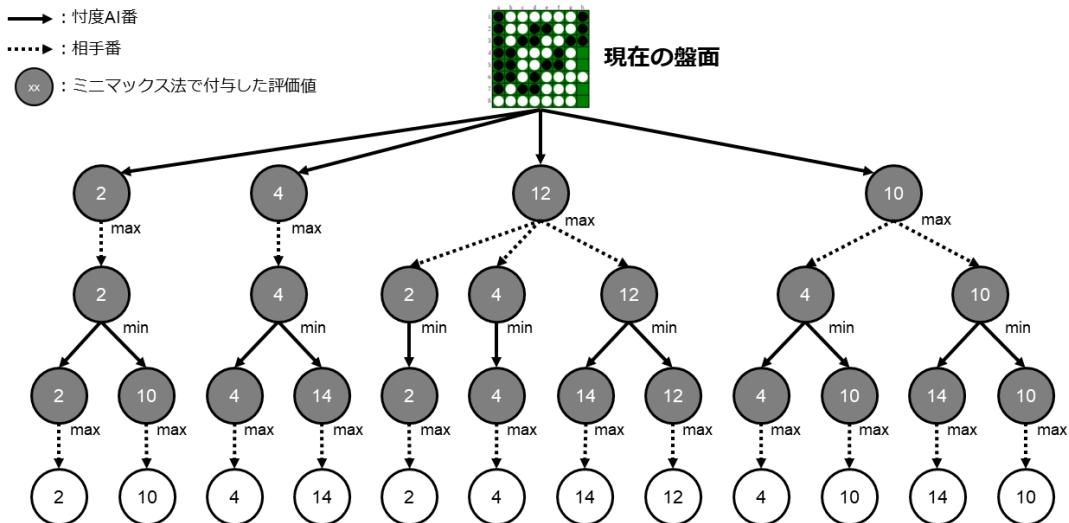


図 7.5: ミニマックス法による評価値付

10 %程度と低く、あまり納得のいく結果ではありませんでした。今回のAIは夏休みの時間を使い、AlphaGoの勉強をしながら様々な改良を加えた結果、引き分け率50%と大きく向上させることができました。

しかし、まだ改善の余地はあります。ミニマックス法の代わりに $\alpha\beta$ 法を使ったり、評価関数を軽量化することで探索を効率化でき、より深い探索が可能になります。また、今回はPythonで実装しましたが、C++などで実装することでさらに高速化が可能です。また機会があれば、このAIをさらに改良していきたいと思います。

第 8 章

将棋 AI の仕組み

8.1 将棋 AI の入力と出力

囲碁は白と黒だけなのに対し、将棋は様々な駒の移動を考慮して、局面と指し手を表現しなければいけません。局面と指し手は、方策ネットワークと価値ネットワークの入力と出力で必要となります。これらのネットワークは画像分類と同じ CNN を土台としていて、8.2 節で詳細を説明します。

8.1.1 局面の表現

まず、局面の表現は、局面の駒を駒ごとにチャンネルに分けて、各チャンネルは、駒の座標を表す 9×9 の二値画像とします。

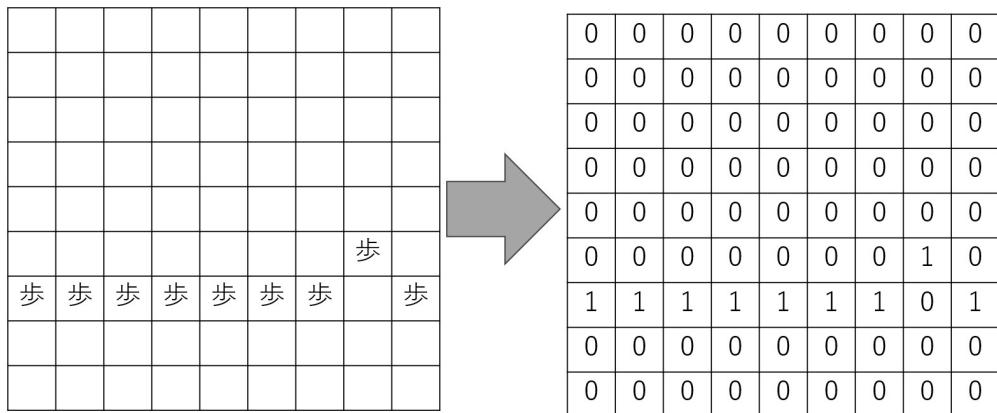


図 8.1: 盤上の駒の表現

盤上の駒は 8 チャンネル、成り駒は別の駒として 6 チャンネルに分けます。持ち駒は、持ち駒の種類ごとに最大枚数分のチャンネルを割り当てます。つまり、持ち駒は王以外の全ての駒 38 チャンネルとします。持ち駒がある場合はすべて 1 の画像とし、持ち駒がない場合はすべて 0 の画像とします。先手後手のチャンネルを割り当てるとき、合計で 104 チャンネルとなります。

8.1.2 指し手の表現

次に、指し手の表現は、指し手をそれぞれラベルとした多クラスとして扱います。駒の移動の種類と移動先の座標を組み合わせると、駒の移動に飛び越しがないので一意的に決まります。移動には

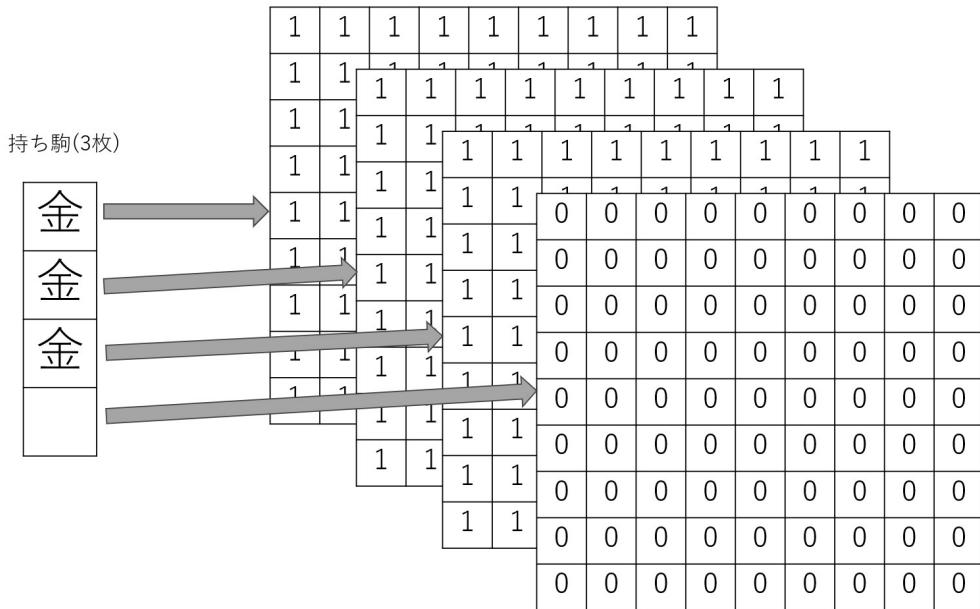


図 8.2: 持ち駒の表現

8方向と桂馬の2方向の10チャンネルがあります。駒が成る場合は、別の移動としてラベルを割り当てます。持ち駒には移動先がないので、持ち駒の7種類にラベルを割り当てます。盤面の座標は 9×9 なので、ラベル数は $27 \times 9 \times 9 = 2187$ となります。

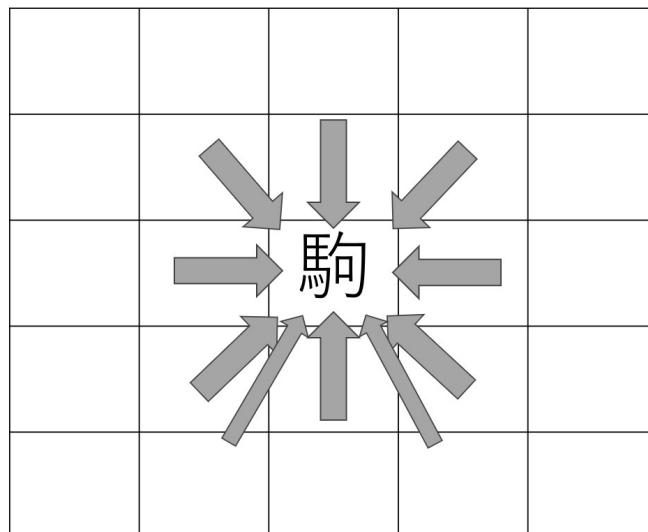


図 8.3: 指し手の表現

8.2 方策ネットワークと価値ネットワーク

将棋AIは、主に方策ネットワークと価値ネットワークの2つを利用しています。方策ネットワークは、対局の中のある局面で次の手を予測するためのネットワークであり、価値ネットワークは、対

局の中のある局面で勝率を予測するためのネットワークです。

8.2.1 方策ネットワーク

方策ネットワークは、局面を 104 チャンネルの 9×9 の画像の入力として、指し手を 2187 クラスの分類問題として予測するニューラルネットワークです。入力層は、フィルターサイズ 3×3 、フィルター数 192、パディング 1、ストライド 1 の畳み込み層とします。活性化関数は ReLU とします。中間層では、入力層と同じ畳み込み層を使用し、残差ネットワークを使用します。ここで特徴的な点は、パディング 1、ストライド 1 で畳み込みを行っても、画像のサイズはそのままなので、局面の位置情報を保持できることです。出力層では、フィルターサイズ 1×1 、フィルター数 27 の畳み込み層を使用し、ソフトマックス関数でラベルごとの確率を出力します。

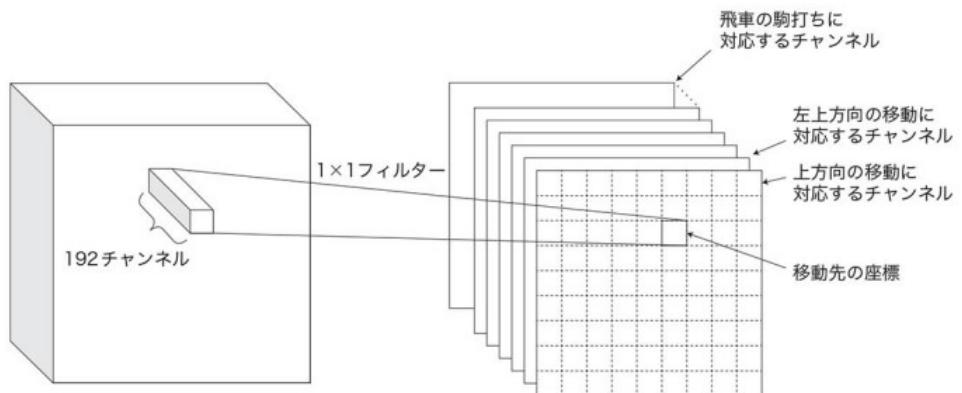


図 8.4: 方策ネットワークの出力層。[47] より引用。

8.2.2 価値ネットワーク

価値ネットワークは、方策ネットワークと同様に、局面を入力として、勝率を 2 値分類問題として予測するニューラルネットワークです。入力層と中間層は、方策ネットワークと同じ構成をしています。出力層では、フィルターサイズ 1×1 、フィルター数 27 の畳み込み層の後に、ユニット数 256 の全結合層を接続し、最後にユニット数 1 の全結合層を接続します。畳み込み層と 1 つ目の全結合層の活性化関数は ReLU とします。最後の全結合層の活性化関数は、シグモイド関数とし、局面の勝率を出力します。

8.2.3 マルチタスク学習

実際の学習では、方策ネットワークと価値ネットワークを 1 つのネットワークとして、同時に学習します。これをマルチタスク学習と言い、タスクに関連がある場合は精度が上がります。さらに、方策ネットワークと価値ネットワークを別々で処理するより、まとめて計算する方が GPU を効率的に利用できます。

どのようにデータから学習するのかというと、プロ棋士の対局のある局面の次の指し手とその局面の勝率を予測し、実際の次の指し手とその対局の勝敗との誤差を小さくするように、確率的勾配降下法で重みを更新します。誤差関数は、方策ネットワークと価値ネットワークにそれぞれクロスエント

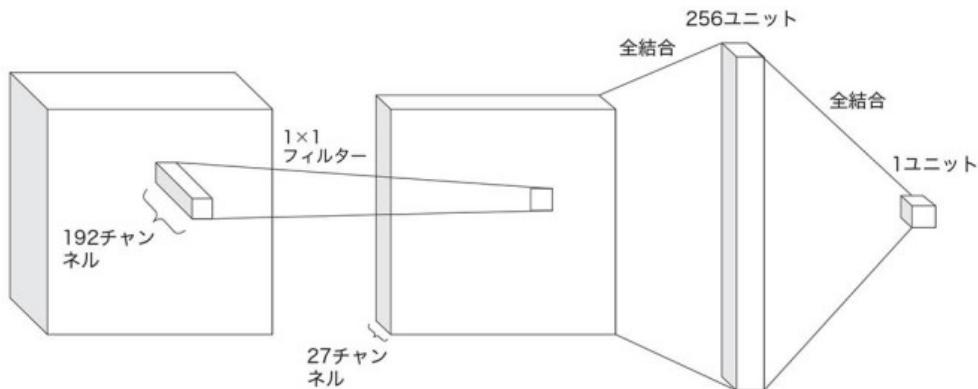


図 8.5: 値値ネットワークの出力層。[47] より引用。

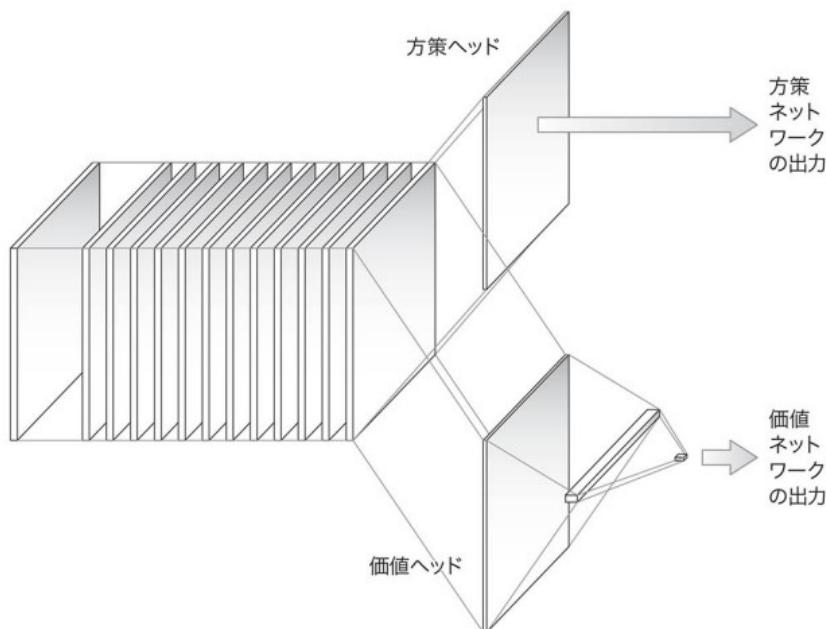


図 8.6: 同時出力するニューラルネットワークの構成。[47] より引用。

コピー関数を使用します。

8.3 モンテカルロ木探索

学習済みの方策ネットワークだけを利用して指し手を予測しても、プロ棋士が指しそうな手を予測して打つだけなので、プロ以上の強さの実現には及びません。そこで、実際にプロ棋士が将棋を打つように、様々な手を進めた局面を想定して次の一手を打つアルゴリズムをモンテカルロ木探索と言います。探索とは、様々な手を進めて良い手を探すことを指します。モンテカルロ木探索では、対局の中の限られた時間で計算しなければならないので、いかに限られた時間で重要な手順を深く探索できるかが肝になります。今回紹介する探索アルゴリズムはアルファゼロが元になっています。

8.3.1 探索ノード

探索では、先の局面を木のように展開していきます。展開した局面をノードと呼び、その中で最上部のノードをルートノード、最下位のノードを葉（リーフ）ノード、自ノードにつながる下位のノードを子ノード、自ノードにつながる上位のノードを親ノードと呼びます。モンテカルロ木探索では、相手はその後の局面の自分の勝率が最小の手を打つとします。

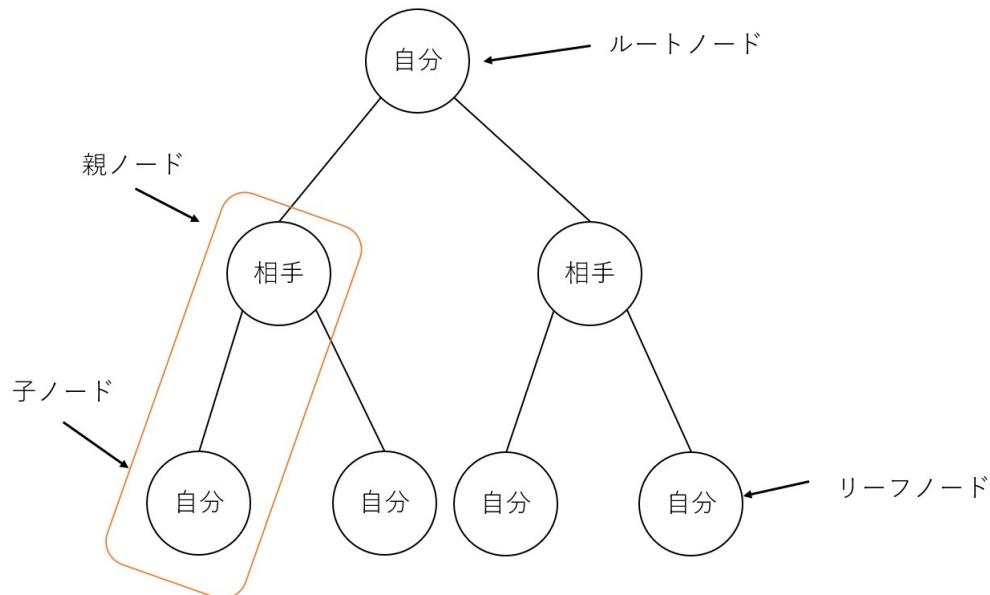


図 8.7: 探索ノード

8.3.2 多腕バンディット問題

探索のシミュレーションに入る前に、探索の例として、多腕バンディットつまりスロットマシンから得られるコインを最大化する問題を考えます。しかし、複数のスロットマシンの期待値はそれぞれ異なり、情報がないものとします。まずは、いろんなスロットを試して、期待値が高そうなスロットを選び、その後、そのスロットを打ち続けます。いろんなスロットを試す回数が多いほど、期待値の高いスロットを選びやすくなりますが、そのスロットを利用する回数が少なくなり、逆に、スロットを試す回数が少ないほど、期待値の低いスロットを選びやすくなります。これを探索と利用のトレードオフと呼びます。

8.3.3 探索のシミュレーション

探索では、現在の局面をルートノードとして、このノードだけの木を作成して、以下のシミュレーションを繰り返し行います。

- Step 1: ルートノードからアーク評価値が最大となる子ノードをたどって木を降りる。
- Step 2: 葉ノードに到達したら、新ノードを一つ展開する。
- Step 3: 各ノードの勝率を更新する。

対局で次に打つ手は、シミュレーションを繰り返し最終的にシミュレーション回数の多い手を選択します。次の節でアーク評価値と勝率の更新を説明します。

8.3.4 アーク評価値と勝率の更新

アーク評価値とは、ノードにおける勝率とバイアスの和で表せます。バイアスは、シミュレーション回数が少ない場合に大きくなり、手の予測確率に比例します。予測確率 $p(s, a)$ は方策ネットワークから計算できます。多腕バンディット問題と同じように、幅広い探索と、勝率の高いノードの探索を両立しています。

$$\begin{aligned}\text{アーク評価値} &= Q(s, a) + u(s, a) \\ Q(s, a) &= \frac{W(s, a)}{N(s, a)} \\ u(s, a) &= C \cdot p(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)}\end{aligned}$$

$W(s, a)$ は子ノードの勝率の合計、 $N(s, a)$ はノードにおけるシミュレーション回数の合計を表します。よって、 $Q(s, a)$ はシミュレーションから得られた勝率の平均です。これらは、step3 で新ノードにおける価値ネットワークの出力 $V(s)$ を用いて、次のように更新します。

$$\begin{aligned}N(s, a) &= N(s, a) + 1 \\ W(s, a) &= W(s, a) + V(s)\end{aligned}$$

8.4 まとめ

今回紹介した将棋 AI は、AlphaGo を基盤にしたニューラルネットワークとモンテカルロ木探索から成ります。ニューラルネットワークの多層構造や、モンテカルロ木探索のノード選択は、人間の思考と似ていたり、実際に元になっていたりします。人間の脳についてもっと理解があるほど、より良い AI が実現できるのかもしれません。

第9章

将棋 AI の思考プロセスを解き明かす： 局面評価の可視化と考察

9.1 はじめに

将棋のプロ棋士、藤井聰太さんが驚異的な実力で八つのタイトルを獲得し、メディアでも大きく取り上げられています。皆さんも「AI を超えた一手」「評価値 99 %からの大逆転」など、彼の活躍について聞いたことがあるでしょう。現在、将棋界では多くのプロ棋士が将棋 AI を活用して研究し、テレビ放送でも AI の評価や予想手が表示されることが一般的になっています。

将棋 AI の歴史は、1990 年から行われている世界コンピュータ将棋選手権を中心に進化してきました。特に、2006 年に登場した「Bonanza」というソフトウェアは、コンピュータチェスの技術を応用して開発され、AI の進化に大きく寄与しました。そして、2012 年にはプロの棋士に対しても勝利する AI が登場しました。さらに、2020 年からはディープラーニングという技術を駆使した AI がますます活躍するようになりました。

ディープラーニングは、2016 年に DeepMind 社が囲碁 AI 「AlphaGo」を開発し、人間のトップ棋士に勝利したことで広く注目されました。この技術は、人間の脳の神経回路の構造を数学的に表現する「ニューラルネットワーク」と呼ばれる手法を多層構造化したものです。情報は入力層から出力層へとつながる中間層で抽象化され、豊かな情報表現を可能にします。ただし、この中間層の思考プロセスは説明が難しいという課題も存在しました。この記事では、ディープラーニング系の将棋 AI がどのように局面を評価するかを可視化する方法を紹介し、可視化された局面について将棋の知識をもとに考察していきます。

9.2 将棋 AI の局面評価を可視化について検討する

ディープラーニング系の AI が情報を処理し、意思決定を行う仕組みを理解することは、その AI を活用したり、改善したりする上で非常に重要です。たとえば、医療診断に使用される AI が患者の病気を判断する場合、その判断プロセスがどのように行われたのかを説明できることが求められます。同様に、自動運転車が交通ルールに従って運転する場合、その判断の根拠を説明できることは安全性の観点から極めて重要です。

言い換えれば、ディープラーニング系の AI が持つ中間層の動作を透明化し、説明可能にすることは、その AI の信頼性を高め、倫理的な側面においても重要な役割を果たします。このテーマについては、本会誌内「説明可能な AI」で詳しく説明しますのでそちらを参照ください。ここでは、画像

処理系AIの思考プロセスを可視化する最近の研究について紹介します。

9.2.1 Grad-CAM (Gradient-weighted Class Activation Mapping)

Grad-CAMは、ニューラルネットワークモデルが画像内のどの部分に焦点を当てているかを可視化するための手法です[48]。ネットワークが処理する画像の中で、出力に大きな影響を与える特徴を見つけ出すことが重要です。これは、特徴が変化すると、ネットワークの出力にも大きな変化が生じるという考えに基づいています。Grad-CAMでは、入力画像に対しての勾配を調べることで、各特徴の重要性を評価します。

具体的な手順は、ネットワークの出力から逆向きに勾配を計算し、各ピクセルや特徴マップが出力クラス分類にどれだけ影響を与えているかを計算します。これにより、重要な特徴がどの部分にあるのかを示すヒートマップが生成されます。このヒートマップは、ネットワークの内部動作を視覚的に理解しやすくし、画像処理タスクにおいてモデルの判断根拠を明らかにするのに役立ちます。

9.2.2 「dlshogi」開発者が提案する将棋AIの可視化手法

「dlshogi」の開発者である山岡忠夫さんは、ブログ記事で非常に興味深い可視化手法を提案しています[49][50]。将棋AIは現在の局面に対して評価値を計算しますが、その評価値を駒を一つずつ取り除いた場合と比較します。この差を見ることで、各駒が評価値にどれだけ寄与しているかがわかります。つまり、どの駒が局面の評価にとって重要なかを把握する手法です。

この手法は、ディープラーニングにおける過学習を抑えるためのテクニックであるDropoutと同様に非常にシンプルでかつ効果的な方法と言えます。局面評価の可視化により、将棋AIの思考プロセスに新たな洞察を得ることができ、将棋AIの戦略や判断の基準を理解するのに役立ちます。

9.3 将棋AIの思考プロセスについて考察する

ディープラーニング系AIの思考プロセスを解き明かしていくためには、入力層と出力層の両方から見ていくことが大切となります。入力層については、山岡忠夫さんの著書「強い将棋ソフトの創りかた Pythonで実装するディープラーニング将棋AI」で紹介されている入力設計について見ていきます[47]。そして、出力層については先に紹介した手法で局面の駒の価値を可視化し、将棋の格言やドメイン知識を活用して、AIの判断プロセスをわかりやすく考察します。

9.3.1 入力層から将棋の特徴について考察する

将棋というゲームの特徴は、駒が14種類あり、それぞれ異なる動きを持つことです。さらに、相手の陣地に入ることで駒の種類を変えたり、相手の駒を取って自分の駒として利用したりできる点も特徴的です。

画像処理系のニューラルネットワークが1つの画像をRGBに分解して3つのチャンネルとして入力するのと同様に、将棋AIでは図9.1のように1つの局面を特徴ごとに複数の入力チャンネルとして分解します。5章では、盤上の駒の14種類、持ち駒の最大数38を先手と後手で合計104の入力チャンネルとして設計されています。また、より強い将棋AIを作ることを目標とした7章では駒の利きや王手かどうかという特微量を加えて表9.1のように設計されています。

入力チャンネルの数が増えると、AIの表現力が向上しますが、学習コストや対局中の計算コスト

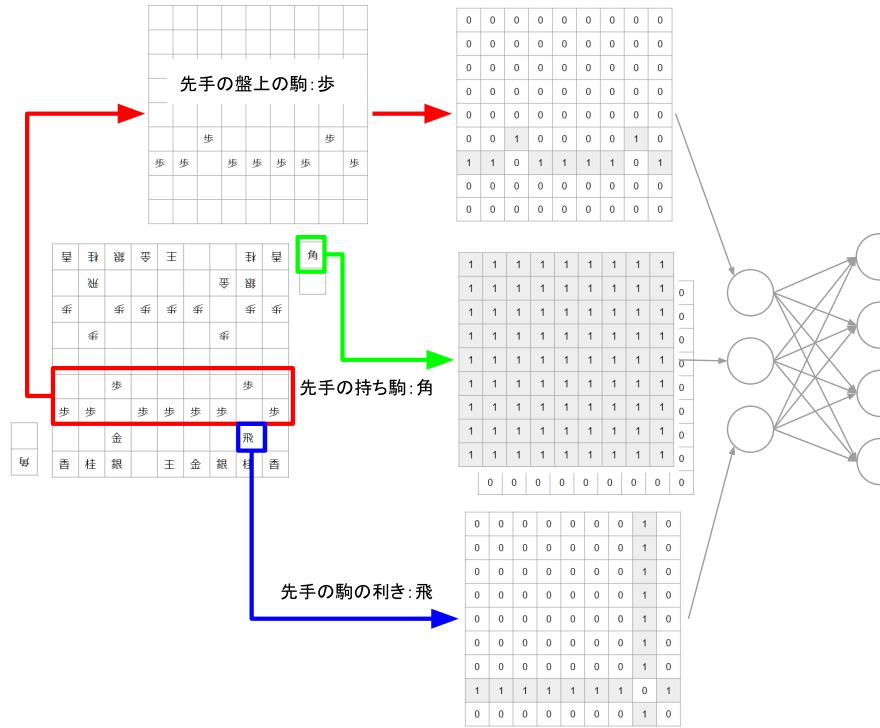


図 9.1: 入力チャンネルイメージ図

表 9.1: 特徴量の入力チャンネル数

特徴量	チャンネル数	備考
先手の盤上の駒	14	歩、香、桂、銀、金、角、飛、王、と、成香、成桂、成銀、馬、竜
先手の持ち駒	28(38)	歩8、香4、桂4、銀4、金4、角2、飛2 (歩を 18 までカウントする場合)
先手の駒の利き	14	
先手のマスごとの駒の利き	3	3つまでカウント
後手の盤上の駒	14	先手参照
後手の持ち駒	28(38)	先手参照
後手の駒の利き	14	先手参照
後手のマスごとの駒の利き	3	先手参照
王手	1	
合計	119	

も増加するというトレードオフが存在します。そのため、入力チャンネルの設計は、限られた計算資源や時間内で効率的な学習を行うために非常に重要です。強い将棋AIを作成するためには、学習させる棋譜データの質や強化学習のための計算資源が重要となっています。

学習に使用するニューラルネットワークは画像識別問題で高い精度を上げているResidual Networkを使用しています。画像識別問題の処理過程を可視化する手法として、畳み込みニューラルネットワークの中間層を画像として表示する方法が存在します。

将棋AIの場合も、例えば角や角の利きに関連する中間層は、角の斜めに動く特徴を捉えた表現になっていると考えられます。つまり、AIは局面を解釈するために、異なる入力チャンネルと中間層を適切に組み合わせ、将棋の戦略的な要素を捉えていると考察できます。

9.3.2 出力された局面の駒の価値を可視化する

それでは9.2.2節で紹介した可視化手法と書籍の中で紹介されている学習済みの将棋AIモデルを用いて、具体的に局面を可視化した例を見てみましょう。図9.2の局面では、先手番は守備の形に「矢倉囲い」、攻撃の形に「棒銀」を目指しています。後手番は守備の形に「雁木囲い」を採用しています。

駒の右下の数値は、どの駒がその局面の評価値に寄与しているかを表示します。数値は先手番が暖色系、後手番が寒色系で色の濃淡でわかりやすく可視化されています。

9.3.3 将棋AIの局面評価について考察1

将棋の有名な格言に「攻めは飛角銀桂、守りは金銀3枚」という言葉があります。序盤はお互いにこの格言のように攻撃の形と守備の形を作ることが重要となります。攻め駒は相手の王に迫っているほど価値が高くなり、守り駒は自分の王を堅く守っているほど価値が高くなります。

それでは、図9.2を使って先手番の駒の価値について具体的に見てみましょう。守備の形は▲6九王の周りに▲7八金、▲5八金、▲7七銀が配置されており、金銀3枚の赤色が濃いことがわかります。

次に攻撃の形を見てみましょう。▲2八飛は縦方向に駒の利きがあるため、2四、2三、2二の地点への攻めを狙っています。▲8八角は斜め方向に駒の利きがあるため、遠く2二の地点への攻めを狙っていますし、7九に移動すれば2四の地点への攻めを狙えます。▲2六銀は守備の銀とは異なり相手の王に向かって駒が進んでいます。▲2九桂は3七に移動後、4五や2五、次に5三や3三を攻める駒となります。

図9.2の可視化された局面は格言と大きな相違はないと言えるのではないでしょうか。

9.3.4 将棋AIの局面評価について考察2

次に図9.3を使って図9.2との駒の価値の違いについて見ていきましょう。2つの局面の違いは王が初期位置から4一ではなく6二に移動したという1点のみです。これは後手から見て王を右に動かすため、「右玉」と呼ばれる戦形で、最近プロ棋士の対局でも採用されることが増えています。

先手番の形は図9.2と全く同じですので攻撃の形は2筋方面を狙っていますが、後手の王が2筋から遠ざかっていますので、相手の王に迫っているほど価値が高くなる攻めの駒の数値がどう変化しているか見てみましょう。

▲2八飛は0.52から0.42と数値が下がっています。同様に▲8八角は0.43から0.38、▲2六銀

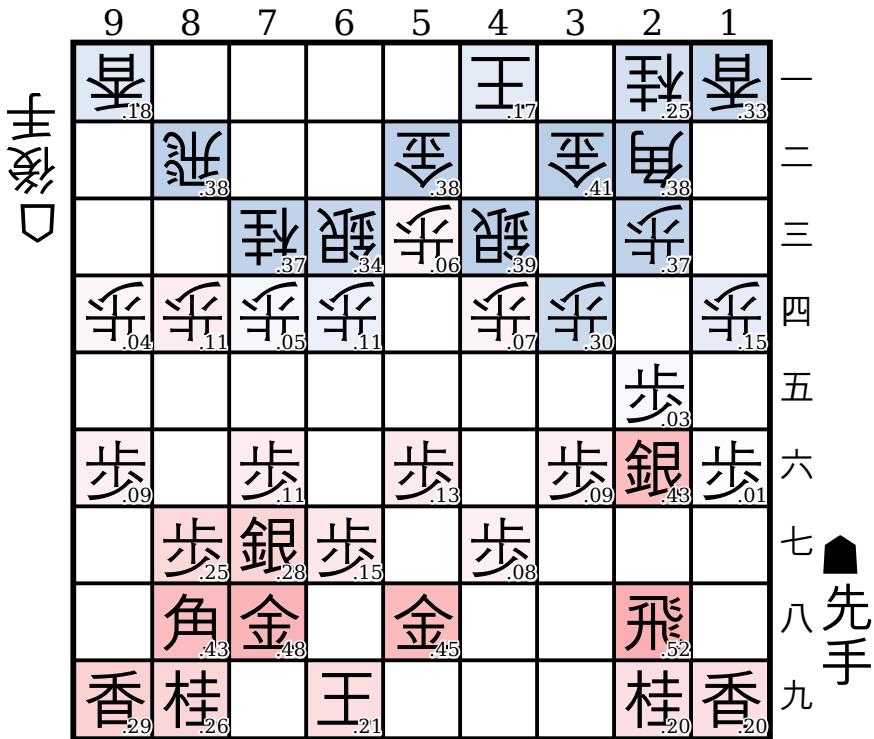


図 9.2: 矢倉 vs 雁木

は 0.43 から 0.33 と数値が下がっています。先手の攻撃の狙いから遠くに王を移動することで、先手の攻撃の駒の価値を大きく減少させていると言えます。「王の早逃げ八手の得」という格言は、終盤に使われることが多いですが、図 9.3 のような序盤でも有効な考え方だと言えるのではないでしょうか。将棋 AI が出てくるまでは、王は 8 八や 2 二など深い位置でしっかり囲うことが普通でしたが、最近の相居飛車の戦形では 5 九のまま戦う居玉や 5 八の中住まいの形が多くなっています。

将棋 AI での研究が進み、プロ棋士の局面の判断も変化してきていると推察します。

9.3.5 将棋 AI の局面評価について考察 3

図 9.4 は序盤で角を交換し、お互いに角を持ち駒にした状態で戦う角換わりという戦形です。非常に激しい変化が多く、将棋 AI で最も研究されている戦形です。

将棋の一番の特徴が相手の駒をとって、自分の駒として使うことができる点にあります。盤上の好きなところに打つことができるため非常に価値が高く、「歩のない将棋は負け将棋」「一步千金」といった歩が持ち駒にあることの価値の高さを表現した格言があります。先手の持ち駒の角は 0.67 と非常に評価が高いことが確認できます。ここでも格言と同じような判断が行われていると考察します。

2023 年 5 月に、将棋 AI の開発者が角換わりという戦形に現れる 1886 局面の変化をすべて AI で解析して先手が +300 の優勢を築くことができると SNS に投稿し話題となりました。将棋 AI 同士の対戦では +300 の差が出ると、次第に差が広がり 9 割近くはそのまま勝敗が決するため、非常に大きな差として捉えられています。しかし、人間同士の対戦においては評価値の捉え方に注意が必要ですので次に事例を紹介します。



図 9.3: 矢倉 vs 右玉

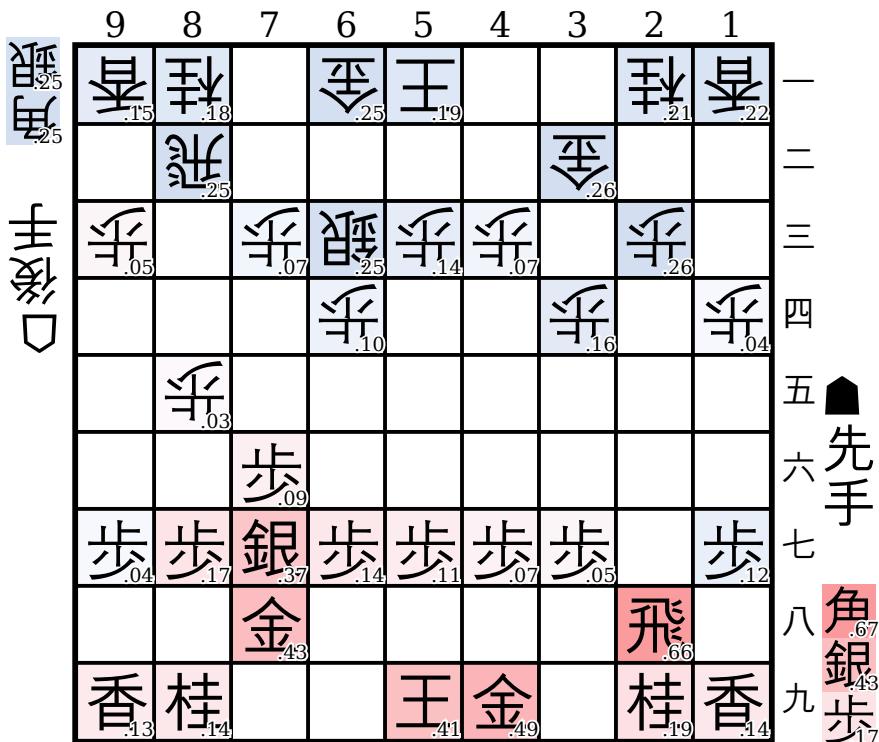


図 9.4: 角換わり

9.3.6 将棋 AI の評価値の注意点

将棋界で話題となったのは、2020年6月8日棋聖戦第1局での渡辺棋聖と藤井聰太挑戦者の対局でした。最終盤で渡辺棋聖が16連続の王手をかけましたが、藤井聰太挑戦者はこれを逃れて勝利し、タイトル奪取に近づきました。王手のかかった局面では候補手が絞りやすい側面もありますが、30手以上の攻防には多くの変化が含まれ、詰むかどうかを人間の目で判断するのは非常に難しい局面で、形勢は互角と言えるでしょう。

将棋 AI の評価値は、お互いに最善手を指し続けた場合の評価に基づいています。しかし、将棋は終盤になるほど駒の取り合いが激しくなり、持ち駒が増えて選択肢も増加するため、逆転の可能性が高まるゲームです。終盤に差し掛かると、人間の感覚と AI の評価値との間に乖離が生じることがあるため、注意が必要です。

9.4 まとめ

将棋界で一般的に活用されるようになってきた将棋 AI について、その思考プロセスの入力から出力まで考察してきました。

入力プロセスでは、将棋 AI が異なる特徴に応じた入力チャンネルと中間層を組み合わせて局面を捉えていることを紹介しました。次に、出力プロセスでは、将棋 AI の局面評価を可視化し、これまでの人間の経験に基づく格言と比較しました。代表的な格言を挙げながら、AI の思考は人間の知識に非常に近いモデルであることを考察しました。

将棋界では AI の発展により、人間が AI を上手く活用することで共に成長し、将棋の奥深さがさらに探求されています。他の分野においても AI の思考プロセスを可視化することで、AI の活用がより積極的になっていくでしょう。

第 IV 部

画像生成

第 10 章

生成モデルの基本

10.1 生成モデルとは

近年 ChatGPT をはじめとした生成モデルが話題になっていますが、生成モデルとは一体どのようなモデルなのでしょうか。生成モデルとは与えられたデータ（以下、訓練データ）をもとに新しいデータを作り出すモデルのことです。注意しなければいけないこととして、生成モデルの目的は訓練データを完全に再現することではないということです。あくまでも新しいデータを作り出すということが重要です。

10.1.1 生成モデルの目的

生成モデルの目的について説明します。訓練データを $D = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}\}$ とし、各 $\mathbf{x}^{(i)}$ が独立にある分布 $p(\mathbf{x})$ からサンプリングされたとします。この時、生成モデルの学習目的は訓練データ D から分布 $p(\mathbf{x})$ を求めることです。しかし $p(\mathbf{x})$ を直接求めるのは困難なので、代わりにパラメータを θ を持つ確率分布 $q_\theta(\mathbf{x})$ を知りたい確率分布 $p(\mathbf{x})$ に近づけることを生成モデルでは行います。つまり、生成モデルの目的はある確率分布 $p(\mathbf{x})$ からサンプリングされたデータ D を元に、 $p(\mathbf{x})$ に近い確率分布 $q_\theta(\mathbf{x})$ を求めることです。

10.1.2 生成モデルによるデータの生成

生成モデルの学習により真に求めたい確率分布 $p(\mathbf{x})$ に近い確率分布 $q_\theta(\mathbf{x})$ が得られたとします。この時、新しいデータ \mathbf{x}^* は次のように生成できます。

$$\mathbf{x}^* \sim q_\theta(\mathbf{x}^*) \quad (10.1.1)$$

つまり、生成モデルで新しいデータを生成するには学習した確率分布 $q_\theta(\mathbf{x})$ からデータをサンプリングすれば良いということが分かります。

10.2 拡散モデルの基礎

ここでは画像を生成する生成モデルの1つである拡散モデルについて説明します。拡散モデルでは画像にノイズを加える操作と、加えたノイズを推定する2つの操作になります。拡散モデルにはいくつかの種類がありますが、ここではDDPMについて説明します。

10.2.1 DDPM

DDPM(Denoising Diffusion Probabilistic Models: DDPM) は拡散過程と逆拡散過程の二つの過程からなる拡散モデルです。拡散過程ではデータにノイズを徐々に加えていき、データを完全なノイズにするという操作を行います。一方、逆拡散過程では拡散過程を逆向きにたどり加えられたノイズを推定するという操作を行います。

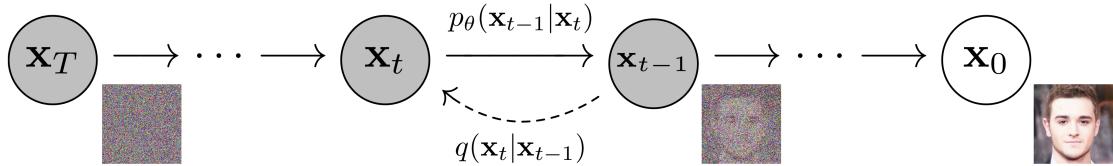


図 10.1: 拡散過程と逆拡散過程。[51] より引用。

これを模式図にすると図 10.1 のようになります。左向きの矢印が逆拡散過程を、右向きの矢印が拡散過程を示しています。

拡散過程

拡散過程ではデータ x_0 に対しノイズを徐々に加えていきデータ x_1, \dots, x_T を得るマルコフ過程が行われています。ここで、マルコフ過程とは次の性質を満たすような過程のことです。

$$q(\mathbf{x}_{t+1}|\mathbf{x}_t) = q(\mathbf{x}_{t+1}|\mathbf{x}_t, \dots, \mathbf{x}_0) \quad (10.2.1)$$

式 (10.2.1) は未来の状態は現在の状態によってのみ決まり、過去の影響を受けないということを表しています。つまり拡散過程のある時刻 t の未来のデータ \mathbf{x}_{t+1} の状態は現在のデータ \mathbf{x}_t によってのみ決まり、それ以前のデータの影響を受けないということです。

拡散過程を数式にすると次のようにになります。以下、 $\mathcal{N}(\mu, \Sigma)$ は平均ベクトルが μ 、共分散行列が Σ の多変量正規分布を表すとします。また、 \mathbf{I} は単位行列を表すとします。

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^T q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad (10.2.2)$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad (10.2.3)$$

ここで $\beta_1, \beta_2, \dots, \beta_T$ は $0 < \beta_1 < \beta_2 < \dots < \beta_T < 1$ であり、各 β_t について $\alpha_t = 1 - \beta_t$ です。したがって、 $\sqrt{\alpha_t}$ は t が大きくなるにつれて 0 に近づき β_t は 1 に近づいていきます。つまり、上の操作を繰り返すとデータ成分は徐々に小さくなり、かつノイズ成分は大きくなっていくので上の操作を繰り返して得られる最終的なデータ $q(\mathbf{x}_T|\mathbf{x}_0)$ は $\mathcal{N}(\mathbf{0}, \mathbf{I})$ とみなすことができます。

さて、式 (10.2.3) を見てみると学習すべきパラメータ θ が含まれていないことが分かります。これは拡散過程では式 (10.2.3) に従ってノイズを加えるだけでよく、ノイズの加え方は学習する必要がないということを表しています。

逆拡散過程とその学習

逆拡散過程では拡散過程により得られた完全なノイズ $\mathcal{N}(\mathbf{0}, \mathbf{I})$ から拡散過程と逆向きのマルコフ過程を行い拡散過程の各ステップで加えられたノイズを推定します。各過程では逆拡散過程で共通のパラメータ θ を持つ平均ベクトルと共に分散行列に従う正規分布を利用します。これを式で書くと次のようになります。

$$p_{\theta}(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^T p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) \quad (10.2.4)$$

$$p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\boldsymbol{\mu}_{\theta}(\mathbf{x}_t, t), \boldsymbol{\Sigma}_{\theta}(\mathbf{x}_t, t)) \quad (10.2.5)$$

$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (10.2.6)$$

一般に逆拡散過程を正規分布で表すことはできませんが、拡散過程における β_t が十分小さい場合には逆拡散分布に正規分布を用いても構わないことが知られています。

次に逆拡散過程の学習方法を説明します。逆拡散過程の学習目標はパラメータ θ を最尤推定により求めることです。最尤推定とは式 (10.2.7) で表される尤度、または式 (10.2.8) で表される対数尤度を最大にするパラメータ θ を求めることです。

$$p_{\theta}(\mathbf{x}_0) = \int p_{\theta}(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \quad (10.2.7)$$

$$\log p_{\theta}(\mathbf{x}_0) = \log \int p_{\theta}(\mathbf{x}_{0:T}) d\mathbf{x}_{1:T} \quad (10.2.8)$$

式 (10.2.7) は逆拡散過程によりデータ \mathbf{x}_0 が生成される確率を表しています。つまり最尤推定では逆拡散過程によりデータ \mathbf{x}_0 が生成される確率を最大にするパラメータ θ を求めるということになります。ここで $p_{\theta}(\mathbf{x}_{0:T})$ は $p_{\theta}(\mathbf{x}_0, \dots, \mathbf{x}_T)$ を、 $d\mathbf{x}_{1:T}$ は \mathbf{x}_1 から \mathbf{x}_T による積分を表します。式 (10.2.7) を見ると元データ \mathbf{x}_0 の尤度の計算には T 個の変数による積分を行う必要があることが分かりますが、この計算を行うことは厳しいです。そこで式 (10.2.7) の代わりに次の式を考えます。

$$\mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[-\log \frac{p_{\theta}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \quad (10.2.9)$$

式 (10.2.9) を変分下限といいます。変分下限は対数尤度より常に小さいという性質があります。つまり変分下限が大きければ対数尤度もそれ以上の値であることが保証されるので、式 (10.2.7) を計算するのが難しい場合には式 (10.2.9) を代わりに最大化することができます。

上でも述べたように式 (10.2.7) の計算は困難なので逆拡散過程の学習では変分下限を最大化する、または負の変分下限を最小化することを行います。ここでは負の変分下限を最小化することを考えます。以下、負の変分下限を $L(\theta)$ と書くことにします。つまり、

$$L(\theta) = -\mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[-\log \frac{p_{\theta}(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \right] \quad (10.2.10)$$

です。式 (10.2.10) を変形すると次のようになります。

$$\mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_{\theta}(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1})} - \log \frac{p_{\theta}(\mathbf{x}_0 | \mathbf{x}_1)}{q(\mathbf{x}_1 | \mathbf{x}_0)} \right] \quad (10.2.11)$$

ここで拡散過程がマルコフ過程であったことを思い出すと式 (10.2.1) より、

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}) = q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) \quad (10.2.12)$$

が成り立ちます。式 (10.2.12) を式 (10.2.11) に代入すると、

$$\mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}{q(\mathbf{x}_1 | \mathbf{x}_0)} \right] \quad (10.2.13)$$

となります。式 (10.2.13) を続けて変形していきます。ベイズの定理を用いると

$$q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0) = \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_t | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_0)} \quad (10.2.14)$$

が成り立ちます。式 (10.2.14) を $q(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_0)$ について解いて式 (10.2.14) に代入すると、

$$\mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[-\log p(\mathbf{x}_T) - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)q(\mathbf{x}_t | \mathbf{x}_0)} - \log \frac{p_\theta(\mathbf{x}_0 | \mathbf{x}_1)}{q(\mathbf{x}_1 | \mathbf{x}_0)} \right] \quad (10.2.15)$$

が成り立ちます。ここで、

$$\begin{aligned} \sum_{t>1} \log \frac{q(\mathbf{x}_{t-1} | \mathbf{x}_0)}{q(\mathbf{x}_t | \mathbf{x}_0)} &= \sum_{t>1} \log q(\mathbf{x}_{t-1} | \mathbf{x}_0) - q(\mathbf{x}_t | \mathbf{x}_0) \\ &= \log q(\mathbf{x}_1 | \mathbf{x}_0) - \log q(\mathbf{x}_T | \mathbf{x}_0) \end{aligned} \quad (10.2.16)$$

が成り立つので、式 (10.2.16) を式 (10.2.15) に代入して整理すると、

$$\mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[-\log \frac{p(\mathbf{x}_T)}{q(\mathbf{x}_T | \mathbf{x}_0)} - \sum_{t>1} \log \frac{p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)}{q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)} - \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \right] \quad (10.2.17)$$

が成り立ちます。ここで、

$$D_{KL}(q || p) = \mathbb{E}_q \left[\log \frac{p(\mathbf{x})}{q(\mathbf{x})} \right] \quad (10.2.18)$$

と置きます。式 (10.2.18) を KL ダイベージェンスといい、2 つの確率分布がどれくらい離れているかを示しています。式 (10.2.18) を用いると、式 (10.2.17) は

$$\begin{aligned} L(\theta) &= \mathbb{E}_{q(\mathbf{x}_{1:T} | \mathbf{x}_0)} \left[D_{KL}(q(\mathbf{x}_T | \mathbf{x}_0) || p(\mathbf{x}_T)) \right. \\ &\quad \left. + \sum_{t>1} D_{KL}(q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0) || p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)) - \log p_\theta(\mathbf{x}_0 | \mathbf{x}_1) \right] \end{aligned} \quad (10.2.19)$$

となります。ここで式 (10.2.19) の第 1 項はパラメータ θ に依存していないため $L(\theta)$ を最小化する際には無視できます。次に式 (10.2.19) の第 3 項は最後の逆拡散過程が次元ごとに独立な正規分布だとみなすことにより計算ができます。最後に式 (10.2.19) の第 2 項の Σ の中の式を求めます。また、式 (10.2.19) の第 2 項に含まれる $q(\mathbf{x}_{t-1} | \mathbf{x}_t, \mathbf{x}_0)$ は次のように書けます。

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0), \tilde{\beta}_t \mathbf{I}) \quad (10.2.20)$$

$$\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{\bar{\beta}_t} \mathbf{x}_0 + \frac{\sqrt{\bar{\alpha}_t}\bar{\beta}_{t-1}}{\bar{\beta}_t} \mathbf{x}_t \quad (10.2.21)$$

$$\tilde{\beta}_t = \frac{\bar{\beta}_{t-1}}{\bar{\beta}_t} \beta_t \quad (10.2.22)$$

$$\bar{\alpha}_t = \prod_{s=1}^t \alpha_s \quad (10.2.23)$$

$$\bar{\beta}_t = 1 - \bar{\alpha}_t \quad (10.2.24)$$

一方 $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ は式 (10.2.5) より平均ベクトルが $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ 、共分散行列 $\Sigma_\theta(\mathbf{x}_t, t)$ の正規分布と分かっています。式 (10.2.20) と式 (10.2.5) を式 (10.2.18) に代入すると、式 (10.2.19) の第 2 項の Σ の中の式は

$$D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \frac{1}{2\sigma_t^2} \|\boldsymbol{\mu}_\theta - \tilde{\boldsymbol{\mu}}_t\|^2 + C \quad (10.2.25)$$

となります。ここで C は θ に依存しない定数です。以上より、式 (10.2.19) の第 2 項は定数部分を無視すると次のように書くことができます。

$$\begin{aligned} & \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\sum_{t>1} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) \right] \\ &= \sum_{t>1} \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} [D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t))] \\ &= \sum_{t>1} \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{1}{2\sigma_t^2} \|\boldsymbol{\mu}_\theta(\theta, t) - \tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right] \end{aligned} \quad (10.2.26)$$

つまり、式 (10.2.19) の第 2 項は時刻 t のサンプル \mathbf{x}_t のひとつ前の時刻のサンプル \mathbf{x}_{t-1} の平均を逆拡散過程の平均 $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$ で推定するということを表しています。ここで、

$$q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t} \mathbf{x}_0, \bar{\beta}_t \mathbf{I}) \quad (10.2.27)$$

を用いると、 \mathbf{x}_t は \mathbf{x}_0 とノイズ ϵ を用いて次のように書くことができます。

$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t} \mathbf{x}_0 + \sqrt{\bar{\beta}_t} \epsilon, \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad (10.2.28)$$

式 (10.2.28) を \mathbf{x}_0 について解くと、

$$\mathbf{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{\bar{\beta}_t} \epsilon) \quad (10.2.29)$$

となります。式 (10.2.28)、式 (10.2.29) を $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$ に代入すると、 $\tilde{\boldsymbol{\mu}}_t(\mathbf{x}_t, \mathbf{x}_0)$ は、

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \tilde{\mu}_t(\mathbf{x}_t, \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{x}_t - \sqrt{\bar{\beta}_t} \epsilon)) \quad (10.2.30)$$

$$= \frac{\sqrt{\bar{\alpha}_{t-1}} \beta_t}{\bar{\beta}_t} \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \beta_t \epsilon) + \frac{\sqrt{\alpha_t} \bar{\beta}_{t-1}}{\bar{\beta}_t} \mathbf{x}_t \quad (10.2.31)$$

$$= \frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \frac{\beta_t}{\sqrt{\bar{\beta}_t}} \epsilon) \quad (10.2.32)$$

となります。ここで式 (10.2.30) から式 (10.2.31) の変形には、 $\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)$ の定義式を用いており、式 (10.2.31) から式 (10.2.32) にかけては $\frac{\sqrt{\bar{\alpha}_{t-1}}}{\sqrt{\bar{\alpha}_t}} = \frac{1}{\sqrt{\alpha_t}}$ と、 $\beta_t + \alpha_t \bar{\beta}_{t-1} = \bar{\beta}_t$ を用いています。このように \mathbf{x}_t の一つ前の時刻におけるデータ \mathbf{x}_{t-1} の平均は拡散過程において加えられたノイズ ϵ を用いた式に書き直すことができます。一方、逆拡散過程においては加えられたノイズは不明です。そこで現在の時刻 t とデータ \mathbf{x}_t から時刻 t で加えられたと考えられるノイズを推定するモデル $\epsilon_\theta(\mathbf{x}_t, t)$ を用意して上の式の ϵ を $\epsilon_\theta(\mathbf{x}_t, t)$ に置き換えると、次の二つの式が得られます。

$$\tilde{\mathbf{x}}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}} (\mathbf{x}_t - \sqrt{\bar{\beta}_t} \epsilon_\theta(\mathbf{x}_t, t)) \quad (10.2.33)$$

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} (\mathbf{x}_t - \frac{\beta_t}{\sqrt{\bar{\beta}_t}} \epsilon_\theta(\mathbf{x}_t, t)) \quad (10.2.34)$$

式 (10.2.33) は推定されたノイズを用いて元データの推測を、式 (10.2.34) は平均を推定することを表しています。これらの結果を式 (10.2.25) に代入し、定数部分を無視すると

$$\begin{aligned} & \sum_{t>1} \mathbb{E}_{q(\mathbf{x}_{1:T}|\mathbf{x}_0)} \left[\frac{1}{2\sigma_t^2} \|\mu_\theta(\theta, t) - \tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0)\|^2 \right] \\ &= \sum_{t>1} \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t \bar{\beta}_t} \|\epsilon - \epsilon_\theta(\mathbf{x}_t, t)\|^2 \right] \\ &= \sum_{t>1} \mathbb{E}_{\mathbf{x}_0, \epsilon} \left[\frac{\beta_t^2}{2\sigma_t^2 \alpha_t \bar{\beta}_t} \|\epsilon - \epsilon_\theta(\sqrt{\alpha_t} \mathbf{x}_0 + \sqrt{\bar{\beta}_t} \epsilon, t)\|^2 \right] \end{aligned} \quad (10.2.35)$$

となります。式 (10.2.35) より、逆拡散過程における学習というのは拡散過程で与えられたノイズをパラメータ θ を用いて推定することを行っていることが分かります。また式 (10.2.35) を見ると時刻 t におけるデータ \mathbf{x}_t は含まれていません。このことからも逆拡散過程で推定したいのは時刻 t のデータそのものではなく、加えられたノイズ ϵ であることが分かります。

10.2.2 拡散モデルの課題

以上が拡散モデルの一つである DDPM についての説明です。10.2.1 節の最後で述べたように拡散モデルでは拡散過程における学習が必要ありません。また、逆拡散過程の学習も式 (10.2.35) を最小化することによるノイズ ϵ の推定でした。そのため拡散モデルでは安定した学習を行うことが可能ですが、しかしその反面欠点もあります。

その一つがデータ生成速度の遅さです。後で説明する GAN では学習した生成器 G に $\mathcal{N}(\mathbf{0}, \mathbf{I})$ からサンプリングされたノイズ z を入力することで新しいデータが生成されます。一方 DDPM では $\mathcal{N}(\mathbf{0}, \mathbf{I})$ からサンプリングされたノイズ z をパラメータ θ を学習した逆拡散過程に入力しノイズを除去することで新しいデータがされます。式 (10.2.5) を見ると逆拡散過程におけるノイズ除去は一度で

はなく T 回に分けて徐々に行われていることが分かります。この T は少なくとも数十程度あるので、GAN では生成器にノイズ \mathbf{z} を入れると一度の計算でデータが生成されるのに対して、拡散モデルでは数十回以上の計算が必要となります。これが拡散モデルにおけるデータ生成速度が遅い原因です。

10.3 GAN の基礎

次に GAN(Generative Adversarial Network: GAN)について説明します。GAN は拡散モデルと同様、生成モデルの一つですがその仕組みは拡散モデルとは異なります。GAN の特徴は識別器と生成器と呼ばれる二つのモデルを互いに学習させるというところにあります。識別器と生成器にはそれぞれ学習目的があります。まず、識別器の学習目的は与えられた画像が生成器が作った画像かどうかを見分けることができるようになります。次に生成器の学習目的は識別器を本物だと思わせるような画像を生成することです。つまり識別器は生成器に騙されないように学習を行い、生成器は識別器を騙そうと学習を行うという”敵対的”な関係が GAN の特徴です。

GAN は発表されて以降、様々な発展形が出てきました。ここでは [52] で説明されている GAN(以下、基本的な GAN) の説明と、それを改善した WGAN について説明します。

10.3.1 基本的な GAN

まず、基本的な GAN の学習について説明します。以下では、生成器を G 、識別器を D で表すことにします。また生成器と識別器のパラメータをそれぞれ θ_G , θ_D とします。

識別器

識別器は入力としてデータを受け取り、それが実際のデータである確率を出力とします。つまり識別器の最終的な目標は生成器が生成した偽物画像については 0 を出力し、本物画像に対しては 1 を出力するようなパラメータ θ_D を求めることです。つまり識別器の目的は、次の式を θ_D に関して最小化することを考えることができます。

$$-\sum_{i=1}^m \log D(\mathbf{x}^{(i)}, \theta_D) + \log (1 - D(G(\mathbf{z}^{(i)}, \theta_G), \theta_D)) \quad (10.3.1)$$

ここで $G(\mathbf{z}^{(i)}, \theta_G)$ は生成器が生成したデータであり、 $\mathbf{z}^{(i)}$ は標準正規分布 $\mathcal{N}(\mathbf{0}, \mathbf{I})$ からサンプリングされた値です。式 (10.3.1) のようにモデルのパラメータを最適にするための指標となる関数を損失関数といいます。

生成器

生成器は標準正規分布 $\mathcal{N}(\mathbf{0}, \mathbf{I})$ からサンプリングされたノイズ \mathbf{z} を入力として受け取り、データ $G(\mathbf{z}, \theta_G)$ を出力とします。生成器の目標は生成したデータ $G(\mathbf{z}, \theta_G)$ を識別器に実際のデータだと思わせることです。つまり識別器に $G(\mathbf{z}, \theta_G)$ を確率 1 で本物だと判断させるということなので、

$$D(G(\mathbf{z}, \theta_G), \theta_D) = 1 \quad (10.3.2)$$

を目指します。したがって生成器の損失関数は、

$$-\sum_{i=1}^m \log(D(G(\mathbf{z}^{(i)}, \theta_G)), \theta_D) \quad (10.3.3)$$

となり、式 (10.3.3) を θ_G について最小化することが生成器の目的です。

GAN の最適解

ここまででは、GAN の生成器と識別器のどちらかが固定されている状況におけるもう片方の損失関数について見てきました。もし生成器と識別器がどちらも固定されていない場合、GAN の損失関数は次のようにになります。

$$L(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D(\mathbf{x}, \theta_D)] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log (1 - D(G(\mathbf{z}, \theta_G), \theta_D))] \quad (10.3.4)$$

$$= \mathbb{E}_{\mathbf{x} \sim p_{data}} [\log D(\mathbf{x}, \theta_D)] + \mathbb{E}_{\mathbf{x} \sim p_g(\mathbf{x})} [\log (1 - (D(\mathbf{x}, \theta_D)))] \quad (10.3.5)$$

ここで $p_{data}(\mathbf{x})$ は実データ \mathbf{x} が従う分布、 $p_z(\mathbf{z})$ は生成器への入力されるノイズ \mathbf{z} が従う分布、そして $p_g(\mathbf{x})$ は生成器から出力されるデータ \mathbf{x} が従う分布を示しています。注意すべき点として、この損失関数を生成器と識別器について最大化すべきなのか最小化すべきかという点についてです。識別器については、先ほど述べたように $D(\mathbf{x}, \theta_D)$ を 1 に、 $D(G(\mathbf{z}, \theta_G), \theta_D)$ を 0 にするのが目的でした。この時、 $L(D, G)$ は最大となるので識別器については損失関数を最大化する必要があるということが分かります。一方で、生成器については $D(G(\mathbf{z}, \theta_G), \theta_D)$ を 1 に近づけるのが目的でした。そのため $\log(1 - D(G(\mathbf{z}, \theta_G), \theta_D))$ は徐々に小さくなるので、生成器については最小化するべきだということが分かります。

では、式 (10.3.4) が G については最小となり、 D について最大となるのはどのような時なのでしょうか。それは $p_{data}(\mathbf{x})$ と $p_g(\mathbf{x})$ が一致するときです。これは次のようにして分かります。

(i) G を固定して D を最大化する。

生成器 G が固定されている、つまり $p_g(\mathbf{x})$ が定まっている場合には、 $L(D, G)$ は識別器 D の出力が次の時、最大となります。

$$D(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \quad (10.3.6)$$

これは正の実数 a 、 b と $x \in [0, 1]$ に対して $f(x) = a \log(x) + b \log(1 - x)$ が

$$x = \frac{a}{a + b} \quad (10.3.7)$$

で最大値を取ることから分かります。

(ii) G について最小化する。

式 (10.3.6) を $L(D, G)$ に代入すると、 $L(D, G)$ は

$$L(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} \left[\log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_g(\mathbf{x})} \left[\log \frac{p_g(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \right] \quad (10.3.8)$$

となります。式 (10.3.8) はすでに D について最大となっているので、 G についてのみの関数と考えることができます。そのため、以下では式 (10.3.8) を $L(G)$ と書くことにします。まず、 $p_{data}(\mathbf{x}) = p_g(\mathbf{x})$ の場合に $L(G)$ は、

$$L(G) = \mathbb{E}_{\mathbf{x} \sim p_{data}} \left[\log \frac{1}{2} \right] + \mathbb{E}_{\mathbf{x} \sim p_g} \left[\log \frac{1}{2} \right] \quad (10.3.9)$$

$$= \log \frac{1}{2} + \log \frac{1}{2} \quad (10.3.10)$$

$$= -\log 4 \quad (10.3.11)$$

となります。これを用いると $L(G)$ は、

$$\begin{aligned}
 L(G) &= -\log 4 + \log 4 + L(G) \\
 &= -\log 4 + \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} \left[\log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \log 2 \\
 &\quad + \mathbb{E}_{\mathbf{x} \sim p_g(\mathbf{x})} \left[\log \frac{p_g(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \log 2 \\
 &= -\log 4 + \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} \left[\log \frac{2p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \right] + \mathbb{E}_{\mathbf{x} \sim p_g(\mathbf{x})} \left[\log \frac{2p_g(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})} \right] \\
 &= -\log 4 + 2D_{JS}(p_{data}(\mathbf{x}) || p_g(\mathbf{x}))
 \end{aligned} \tag{10.3.12}$$

となります。ここで最終行の D_{JS} は JS ダイバージェンスと呼ばれ、式 (10.2.18) で表せる KL ダイバージェンスと同じく 2 つの確率分布がどれくらい離れているかを示す指標の 1 つであり次の式で与えられます。

$$D_{JS}(p(x) || q(x)) = \frac{1}{2} \left(\mathbb{E}_{x \sim p(x)} \left[\log \frac{2p(x)}{p(x) + q(x)} \right] + \mathbb{E}_{x \sim q(x)} \left[\log \frac{2q(x)}{p(x) + q(x)} \right] \right) \tag{10.3.13}$$

JS ダイバージェンスは任意の確率分布 $p(x), q(x)$ について非負であり $p(x) = q(x)$ のときのみ 0 となります。この性質を用いると $L(G)$ の最小値は $-\log 4$ であり、これは式 (10.3.9) より $p_{data}(\mathbf{x}) = p_g(\mathbf{x})$ で成り立つことが分かります。

ここまで、GAN の定式化を行い、 $L(D, G)$ が最小になるのは $p_{data}(\mathbf{x}) = p_g(\mathbf{x})$ の時であることを見てきました。ではこの $p_{data}(\mathbf{x}) = p_g(\mathbf{x})$ は何を意味しているのでしょうか。 $p_{data}(\mathbf{x})$ が実データの分布、 $p_g(\mathbf{x})$ が生成データの分布を表していたので $p_g(\mathbf{x}) = p_{data}(\mathbf{x})$ は生成器 G が識別器 D からの情報を元に実データ分布 $p_{data}(\mathbf{x})$ を完全に学習することができたということになります。このとき、識別器の出力は、常に $\frac{1}{2}$ となります。これは識別器には生成されたデータと本物のデータの見分けがつかなくなったということを意味しています。

10.3.2 WGAN とは

次に上で述べた GAN を改善した WGAN について説明します。WGAN では Wasserstein 距離を用いて GAN の損失関数を定義します。Wasserstein 距離を用いた損失関数を見る前に、上の GAN の問題について説明します。

問題の 1 つとしてあるのが損失関数に JS ダイバージェンスを用いているという点です。最適な識別器 D のもとでの生成器 G の損失関数は式 (10.3.12) より、

$$L(G) = -\log 4 + 2D_{JS}(p_{data}(\mathbf{x}) || p_g(\mathbf{x}))$$

で表せていました。この式に含まれる JS ダイバージェンスは 2 つの確率分布がどのくらい離れているかを表す指標の 1 つでした。しかし JS ダイバージェンスでは確率分布がどれくらい離れているかを正確に表せない場合があります。その例として次のようなものがあります。

$U[0, 1]$ を $[0, 1]$ 上の一様分布として、 $y \sim U[0, 1]$ とします。このとき $\theta \in \mathbb{R}$ として、次の二つの確率分布

$$\begin{cases} P(0, y) \\ Q(\theta, y) \end{cases} \tag{10.3.14}$$

の JS ダイバージェンス $D_{JS}(P||Q)$ を求めてみましょう。直感的には $|\theta|$ が大きくなるにつれて P と Q は離れるので JS ダイバージェンスも大きくなりそうですが、実はそうはありません。まず $\theta = 0$ の時 2 つの確率分布 P と Q は一致し、かつ P と Q は共に一様分布なので JS ダイバージェンスは 0 となります。一方 $\theta \neq 0$ の時 JS ダイバージェンスは、

$$\begin{aligned} D_{JS}(P||Q) &= \frac{1}{2} \left(\mathbb{E}_{x,y \sim P} \left[\log \frac{2P(x,y)}{P(x,y) + Q(x,y)} \right] + \mathbb{E}_{x,y \sim Q} \left[\log \frac{2Q(x,y)}{P(x,y) + Q(x,y)} \right] \right) \\ &= \frac{1}{2} (\mathbb{E}_{x,y \sim P} [\log 2] + \mathbb{E}_{x,y \sim Q} [\log 2]) \\ &= \log 2 \end{aligned} \quad (10.3.15)$$

となります。つまり $\theta \neq 0$ のとき JS ダイバージェンスは常に $\log 2$ となってしまいます。これでは二つの確率分布 P, Q がどれくらい離れているかを推測することができません。先ほども述べた通り、直感的には $|\theta|$ が大きいほど二つの確率分布は離れるので JS ダイバージェンスも大きくなるように思いますが、残念ながらこの例ではそうはありません。このように JS ダイバージェンスは 2 つの確率分布の離れ具合を測る指標としては不適な場合もあります。

上の例のように JS ダイバージェンスを用いた損失関数では学習が上手くいかないことがあります。そのため確率分布間の距離に関する別の指標が必要となります。その 1 つがここで説明する Wasserstein 距離と呼ばれるものです。Wasserstein 距離は 2 つの確率分布 p, q に対して次の式で定義されます。

$$W(p, q) = \inf_{\gamma \sim \Pi(p, q)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|] \quad (10.3.16)$$

ここで $\Pi(p, q)$ は p と q の同時分布全体の集合を表しています。先ほどの例の場合の Wasserstein 距離は、

$$W(P, Q) = |\theta| \quad (10.3.17)$$

となり、直感通り $|\theta|$ が大きくなると Wasserstein 距離も大きくなります。このように Wasserstein 距離を用いると JS ダイバージェンスでは正確に距離を測れない場合でも正確に測ることができます。ところが Wasserstein 距離には計算の面で問題があります。上の式を見ると Wasserstein 距離は 2 つの確率分布 p と q の全ての同時分布で計算を行っているため、計算に時間がかかるかもしれません。そのため、次の式を考えます。

$$W(p, q) = \frac{1}{K} \sup_{\|f\|_{Lip} \leq K} \{\mathbb{E}_{x \sim p}[f(x)] - \mathbb{E}_{y \sim q}[f(y)]\} \quad (10.3.18)$$

式 (10.3.18) と式 (10.3.16) の最適解は一致することが知られています。ここで、 K は正の実数であり $\|f\|_{Lip} \leq K$ は関数 f が次を満たすことを意味しています。

$$|f(x) - f(y)| \leq K|x - y| \quad (10.3.19)$$

この条件を満たす関数 f を K-Lipschitz 条件を満たしている関数と呼びます。ここで式 (10.3.18) における f, p, q を次のように定めます。

$$\begin{cases} f = D_w(x) \\ p = p_{data}(\mathbf{x}) \\ q = p_g(\mathbf{x}) \end{cases} \quad (10.3.20)$$

ここで w は識別器 D のパラメータとし $f_w(x)$ は K-Lipschitz 条件を満たしているとします。この時、式 (10.3.18) を $\frac{1}{K}$ を無視して書くと、

$$L(p_{data}, p_g) = \sup_{w \in W} (\mathbb{E}_{x \sim p_{data}} [D_w(x)] - \mathbb{E}_{z \sim p_r(z)} [D_w(G(z))]) \quad (10.3.21)$$

となります。そして式 (10.3.21) が Wassertstein 距離を用いた場合の GAN の損失関数となります。

10.3.3 モード崩壊

ここまで基本的な GAN とそれを改善した WGAN について説明してきました。最後に GAN の課題の一つであるモード崩壊について説明します。

モード崩壊とは、生成される画像の多様性が失われてしまう現象のことです。例えば、訓練データに 0 ~ 9 の手書き文字を用いたとしましょう。この時生成モデルは 0 ~ 9 の全てを生成できるようになるのが目標なのですが、一部の数字 (1 と 0 だけなど) のみしか生成しないことがあります。モード崩壊とはこのように訓練データの一部のみしか生成器が生成できない状態のことです。

なぜ、モード崩壊が起こるのかというと生成器がデータの分布 $p_{data}(x)$ を完全に学習できないからです。

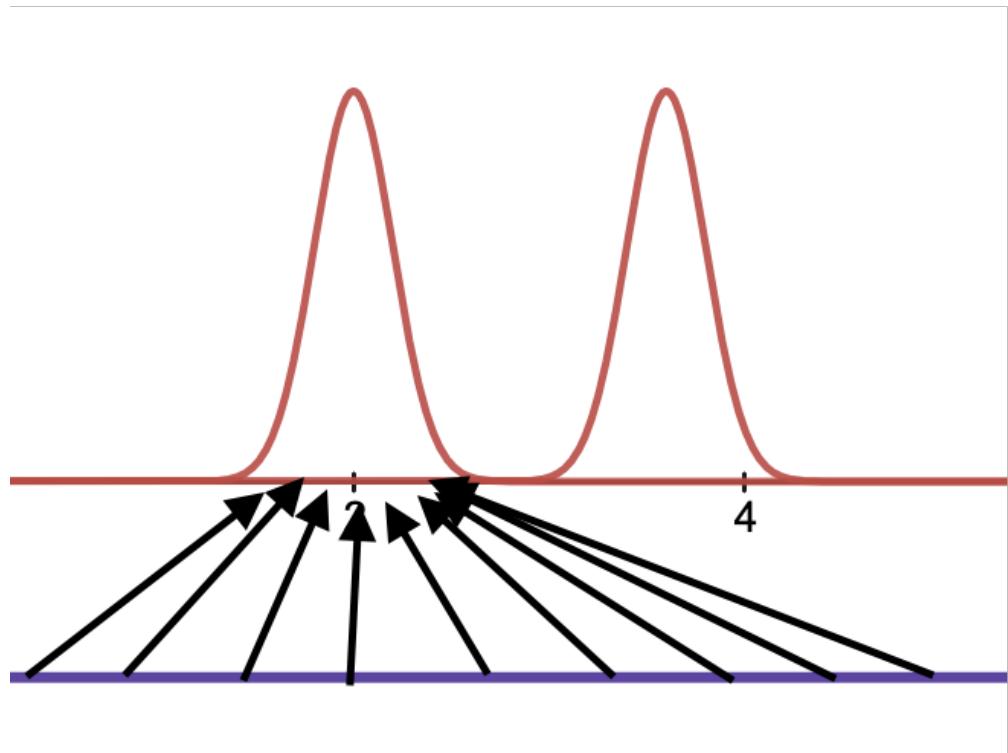


図 10.2: モード崩壊。[53] より引用。

図 10.2 の赤い曲線は $p_{data}(x)$ を、青い直線から出ている矢印は生成器が学習できた $p_{data}(x)$ の領域を表しています。この図を見ると生成器は $p_{data}(x)$ のうち、左半分のみを集中的に学習しており、右半分を全く学習できていません。このように生成器の学習に偏りが起るのがモード崩壊の原因です。

モード崩壊は GAN の重要な課題の一つであり、多くの解決策が考えられてきました。10.3.2 節で紹介した WGAN もモード崩壊を防ぐ方法の一つです。

第 11 章

生成モデルの応用

11.1 Diffusion Model の応用

11.1.1 ガイダンス

ここでは Diffusion Model に対する条件付けの方法について解説します。例えば、Stable Diffusion はテキストを条件とする条件付きモデルになっています。さて、Diffusion Model ではスコア $\nabla \log p(\mathbf{x})$ を予測していたのでした。条件付けのためには、ある条件 \mathbf{y} が与えられた時の条件付きスコア $\nabla \log p(\mathbf{x}|\mathbf{y})$ を予測すればよいことになります。スコアを予測するニューラルネットに条件を追加で入力してやればこのスコアを計算したことになるのですが、実際にはそれに加えてガイダンスと呼ばれる仕組みが広く使われています。そして、ここではガイダンスの代表的な例として Classifier guidance [54] と Classifier-free guidance [55] について紹介します。

まず、Classifier guidance についてです。ベイズの定理 $p(\mathbf{x}|\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})/p(\mathbf{y})$ を用いれば、 $\nabla \log p(\mathbf{x}|\mathbf{y}) = \nabla \log p(\mathbf{y}|\mathbf{x}) + \nabla \log p(\mathbf{x})$ と分解できます ($p(\mathbf{y})$ の \mathbf{x} についての勾配は $\mathbf{0}$ であることに注意が必要です)。第 1 項の $p(\mathbf{y}|\mathbf{x})$ は、入力 \mathbf{x} が条件 \mathbf{y} に合致する確率であり、これは分類モデルや回帰モデルによって計算できます。そしてその勾配も (モデルが微分可能であれば) 計算できます。第 2 項の $\nabla \log p(\mathbf{x})$ は条件を考慮しない Diffusion Model によって計算できます。以上から $\nabla \log p(\mathbf{x}|\mathbf{y})$ が計算できることになります。実用上では、条件付けの強度を操作したい (よりテキストに忠実な画像を生成したい等) 場合があるため、パラメータ w を用いて第 1 項をスケールした式 (11.1.1) が使われます。これが Classifier guidance です。

$$\nabla \log p(\mathbf{x}|\mathbf{y}) \leftarrow w \nabla \log p(\mathbf{y}|\mathbf{x}) + \nabla \log p_\theta(\mathbf{x}) \quad (11.1.1)$$

Classifier guidance のメリットは、条件付けを考慮することなく Diffusion Model の学習を行えるという点です。反対にデメリットは、分類モデル $p(\mathbf{y}|\mathbf{x})$ の構築が大変だという点です。サンプリングの途中で分類モデルに入力される \mathbf{x} はノイズがかかっているため、使用する分類モデルはノイズがかかった画像でも学習されている必要があります。

次に Classifier-free guidance です。Classifier guidance の式 (11.1.1) に再度ベイズの定理を適用し、 $\nabla \log p(\mathbf{y}|\mathbf{x}) = \nabla \log p(\mathbf{x}|\mathbf{y}) - \nabla \log p(\mathbf{x})$ を代入すると式 (11.1.2) が得られます。

$$\nabla \log p(\mathbf{x}|\mathbf{y}) \leftarrow \nabla \log p_\theta(\mathbf{x}|\emptyset) + w (\nabla \log p_\theta(\mathbf{x}|\mathbf{y}) - \nabla \log p_\theta(\mathbf{x}|\emptyset)) \quad (11.1.2)$$

右辺の $\nabla \log p_\theta(\mathbf{x}|\mathbf{y})$ はニューラルネットに条件を追加で入力することで推定されます。 $\nabla \log p_\theta(\mathbf{x}|\emptyset)$ は $\nabla \log p_\theta(\mathbf{x})$ と同じものですが、実際にはゼロベクトルのような特殊なベクトルが「条件無し」という条件として入力されます。これが Classifier-free guidance の仕組みです。

Classifier-free guidance は上記で挙げた Classifier guidance のデメリットを解決することができます。また、(A) 入力に条件が加わること (B) ランダムに「条件無し」条件を入力とすることの 2 点を除けば、通常の Diffusion Model と同じように学習できます。

最後に、簡単なデータでガイダンスによるサンプリングを見てみましょう。scikit-learn の 2 次元データを作成する関数 `make_moons` を使って、図 11.1 のようなデータ点を作成しました。各データは 2 次元平面上の座標が与えられ、データ点は全部で 1000 個あります。また、左上の弧上の点はクラス 0、右下の弧上の点はクラス 1 に分けられています。今回のデータセットの規模では、サンプリング時のスコアをデータセットから直接計算することができます。

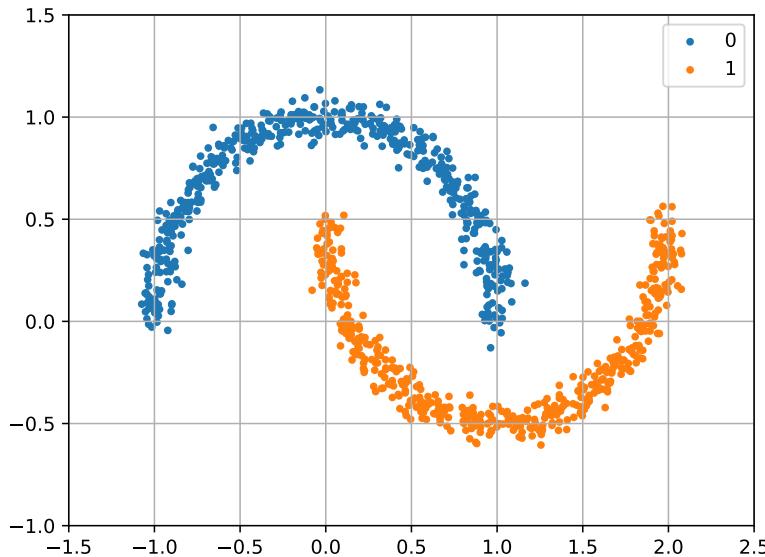


図 11.1: 2 次元データセット。

Classifier-free guidance(以下 CFG と表記)を用いて生成した場合のサンプル(黒星)を図 11.2 に示します。今回の条件はクラス(0 か 1)とします。まず、(a) $w = 0.0$ の場合ですが、式 (11.1.2) から分かる通り、これは CFG を使わずに条件無しで生成していることになります。この場合にはどちらのクラスのデータも万遍なく生成できることがわかります。次に (b) $w = 1.0$ の場合ですが、今度は CFG を使わずに条件ありで生成していることになります。今回はスコアを正確に計算できているため、クラス 1 に属するようなデータだけを万遍なく生成出来ています。最後に (c) $w = 4.0$ 、CFG を用いて条件付けをより強めた場合を見てみます。クラス 1 に属するようなデータは生成出来ていますが、他のクラス(クラス 0)からより離れたデータを生成するようになっています。このように、CFG は与えられた条件によりマッチするように他の条件から離れたデータを生成しようとするため、その結果、生成されるデータの範囲は縮小してしまうというデメリットがあるのです。

なお、このデモプログラムは 11 月祭期間中に KaiRA のホームページ上で公開しています。無料でアクセスできるので、ぜひ試してみてください。

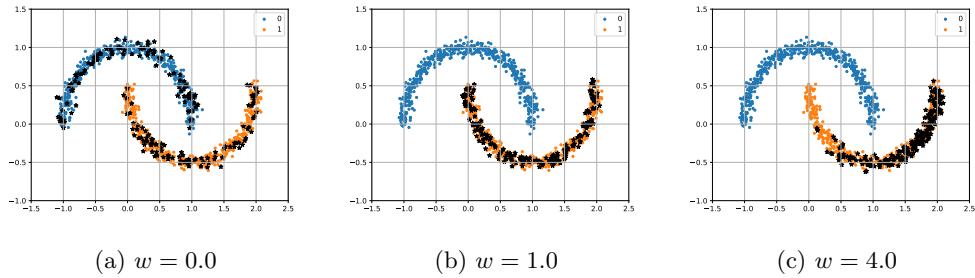


図 11.2: Classifier-free guidance を用いた場合の生成サンプル。

11.1.2 サンプリング手法

これまで DDPM [51] と呼ばれる手法によって反復的に潜在変数を更新することで画像を生成するという説明をしてきました。しかし、潜在変数を更新するためのサンプリング手法には様々な種類があり、手法によって更新回数や生成画像品質が異なるのです。例えば、Stable Diffusion を簡単に実行できる Stable Diffusion WebUI という実行環境では、たくさんの種類のサンプリング手法が選択できることが確認できます(図 11.3)。

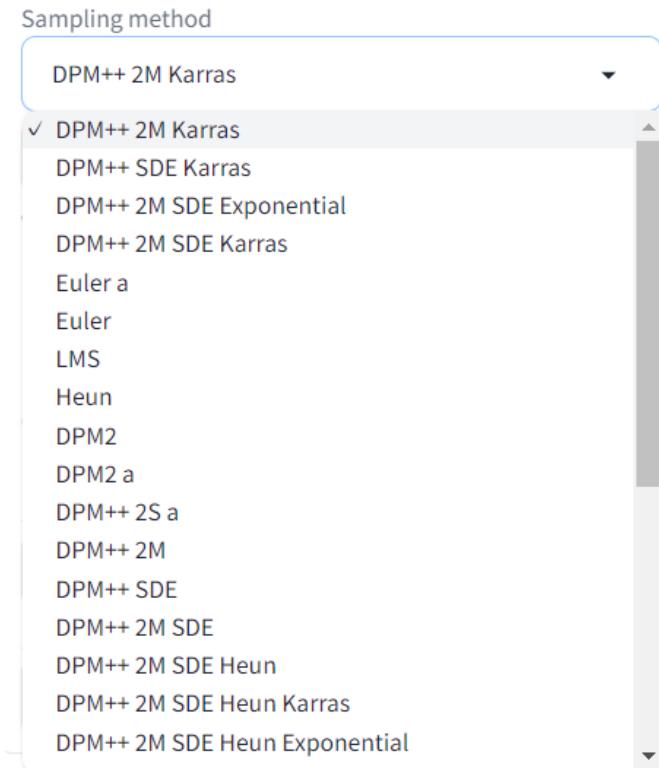


図 11.3: Stable Diffusion WebUI における Sampling method の設定画面。

最も基本となる DDPM では、 $t = T, T-1, T-2, \dots, 0$ と各マルコフ過程をサンプリングして逆拡散過程を計算し、データを生成します。これは十分大きな回数のサンプリングを行う必要があります。

す。それに対して、DDIM [56] は DDPM の確率的な成分（正規分布からのサンプリング）を抜いたサンプリング手法です（Probabilistic と Implicit の頭文字です）。DDIM の更新式を式 (11.1.3) に示します。確率的な成分が無くなると、 $t = t_1$ での更新と $t = t_1 - 1$ での更新がほぼ同じと見なせるため、もはや更新幅を 1 にする必要がありません。つまり、 $t = T, T-10, T-20, \dots, 0$ のように更新してもよいことになります（更新式の $t-1$ の添え字を $t-10$ に置き換えることで計算できます）。DDIM は DDPM に比べて十分の一程度のサンプリング数で済むのがメリットです。

$$\mathbf{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}} \left(\frac{\mathbf{x}_t - \sqrt{\beta_t} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{\bar{\alpha}_t}} \right) + \sqrt{\bar{\beta}_{t-1}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \quad (11.1.3)$$

また、サンプリング幅を 0 の極限に飛ばすことで、サンプリングを微分方程式の形で表すことができます。DDPM のような確率的な成分が挟まる場合、拡散過程は式 (11.1.4)、逆拡散過程は式 (11.1.5) の形で表すことができます。ただし $\mathbf{w}, \bar{\mathbf{w}}$ は標準ウィーナー過程と呼ばれるものです。

$$d\mathbf{x} = f(t)\mathbf{x} dt + g(t)d\mathbf{w} \quad (11.1.4)$$

$$d\mathbf{x} = [f(t)\mathbf{x} - g(t)^2 \nabla \log p_t(\mathbf{x})] dt + g(t)d\bar{\mathbf{w}} \quad (11.1.5)$$

式 (11.1.4) と式 (11.1.5) は確率微分方程式と呼ばれますが、これに従ってサンプリングすることで $p_t(\mathbf{x})$ を得ることができます。しかし、フロー常微分方程式と呼ばれる式 (11.1.6) に従ってサンプリングすることでも $p_t(\mathbf{x})$ を得ることができます。

$$d\mathbf{x} = \left[f(t)\mathbf{x} - \frac{1}{2}g(t)^2 \nabla \log p_t(\mathbf{x}) \right] dt \quad (11.1.6)$$

この常微分方程式を \mathbf{x}_t の点におけるテイラー展開による近似で計算可能な形で解き、近傍の点を \mathbf{x}_{t-1} としてサンプリングするのが DPM [57] という手法です。更新式を式 (11.1.7) に示します。DPM は DDIM よりも少ないサンプリング数で十分な品質のデータを生成することができる事が知られています。なお、テイラー展開を 1 次の項で切った場合の DPM の更新式は DDIM の更新式と同じになるため、DPM は DDIM の一般化と見なすこともできます。

$$\mathbf{x}_{t-1} = \sqrt{\frac{\bar{\alpha}_{t-1}}{\bar{\alpha}_t}} \mathbf{x}_t - \sqrt{\bar{\alpha}_{t-1}} \sum_{n=0}^{k-1} \boldsymbol{\epsilon}_\theta^{(n)}(\mathbf{x}_t, t) \int_{\lambda_t}^{\lambda_{t-1}} e^{-\lambda} \frac{(\lambda - \lambda_t)^n}{n!} d\lambda \quad (11.1.7)$$

その他、一般的な微分方程式の数値解法である Euler 法や Heun 法も使うことができます [58]。このようにサンプリング手法には様々な種類が存在しますが、サンプリング数がより少ない方が計算時間が短いため、特に Stable Diffusion のような大規模なモデルの場合にはどのサンプリング手法を使うかは重要な選択になります。

最後に、ガイダンスの項で扱ったデータを用いて、DDPM、DDIM、DPM の 3 手法について比較してみましょう。データの詳細については 11.1.1 をご覧ください。なお、DPM については上述の通り 1 次の項で切った場合は DDIM と全く同じ手法になってしまいますため、2 次の項で切っています (DPM2)。サンプリング数を 1000, 200, 20 と変えた時の各サンプリング手法での生成結果（黒星が生成データ）を図 11.4 に示します。まず左から 1 列目、サンプリング数が 1000 の場合（サンプリング幅が 1 の場合）は、どの手法も学習データに従ったデータを生成することができます。次に 2 列目、サンプリング数が 200 の場合（サンプリング幅が 5 の場合）は、DDPM は綺麗にデータを生成することができません。一方 DDIM と DPM は綺麗にデータを生成することができます。最後に 3 列目、サンプリング数が 20 の場合（サンプリング幅が 50 の場合）は、DDIM も一部学習デー

タから外れたデータが生成されてしまっています。一方 DPM は未だ綺麗にデータを生成することができており、今回の設定においては DPM が最もサンプリングの効率が良いと言えます。このサンプリング効率については Stable Diffusion のような大規模モデルにおいても同じ傾向が見られ、Stable Diffusion でも DPM を使えば 10 回程度のサンプリング回数で綺麗な画像を生成できます。

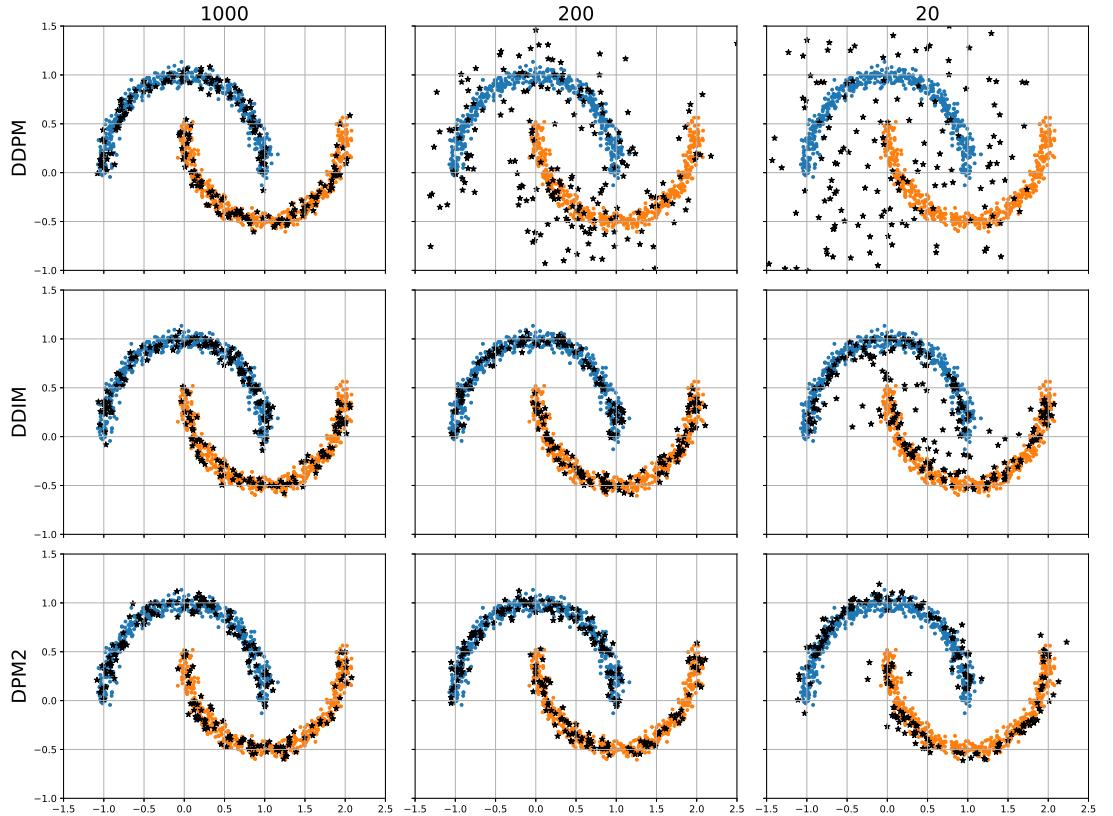


図 11.4: サンプリング数を変えた場合における DDPM、DDIM、DPM による生成サンプル。

また、このデモプログラムも 11 月祭期間中に KaiRA のホームページ上で公開しています。無料でアクセスできるので、ぜひ試してみてください。

11.1.3 Stable Diffusion の活用

Stable Diffusion の大きな特徴として、「柔軟なカスタマイズが可能」という点があります。例えば新たな概念や、テキスト以外の条件付けを簡単に学習させることができます。

新たな概念の学習の例として、「ペットの名前（テキスト）を入れるとペットの画像を生成できるようにしたい」という例が考えられます（図 11.5）。これを実現するための手法には、Textual Inversion [59] や DreamBooth [60] が挙げられます。Textual Inversion は、新たな単語の埋め込みベクトルを用意し、その単語が目的の画像を生成するよう埋め込みベクトルを最適化します。DreamBooth はそれに加えてモデル自体も fine-tuning する手法です。Stable Diffusion 自体を再学習させようとすると数週間レベルの時間が必要になりますが、これらの手法は数十分～数時間で学習可能であり、一般的のユーザーでも簡単に実行できることが魅力です。

別の例としては「ゴッホの画風で画像を生成できるようにしたい」という例が考えられます。これを実現するための手法としては、LoRA [61] があります。LoRA は学習済みの重み $W \in \mathbb{R}^{d \times d}$



図 11.5: 新たな概念の学習。左の犬の画像を学習させることで、右のようにその犬を含んだ画像を生成できる。[60] より引用。

に対して残差的に新たな重みを追加して学習させますが、この重みの追加方法にコツがあります。 $A \in \mathbb{R}^{d \times r}$ と $B \in \mathbb{R}^{r \times d}$ を用いて、その積 AB を残差的に接続するのです。元のモデルの学習済み重み W はフルランクではない（つまり本質的には d 次元以下でよい）場合が多いため、 $r < d$ と設定してもよいことになります（行列 AB のランクは最大でも r になるため）。こうすることで学習済みパラメータより小さい $d \times r$ 行列 2 個を追加すればよいことになり、学習パラメータを削減することができます。また、 B を 0 で初期化することで、学習初期はゼロ行列を加算していることになり、安定した学習が可能になります。このような理由から、先ほどの例と同様に、一般のユーザーでも容易に学習させることができます。

テキスト以外の条件付けの例としては、「ポーズを指定してキャラクターを生成したい」という例が考えられます（図 11.6）。これを実現するための手法には、ControlNet [62] が挙げられます。ControlNet は残差的に新たな学習パラメータを接続することでテキスト以外の入力を処理できるようになっています。また、LoRA 同様後半のパラメータを 0 にすることで安定した高速学習が可能になります。

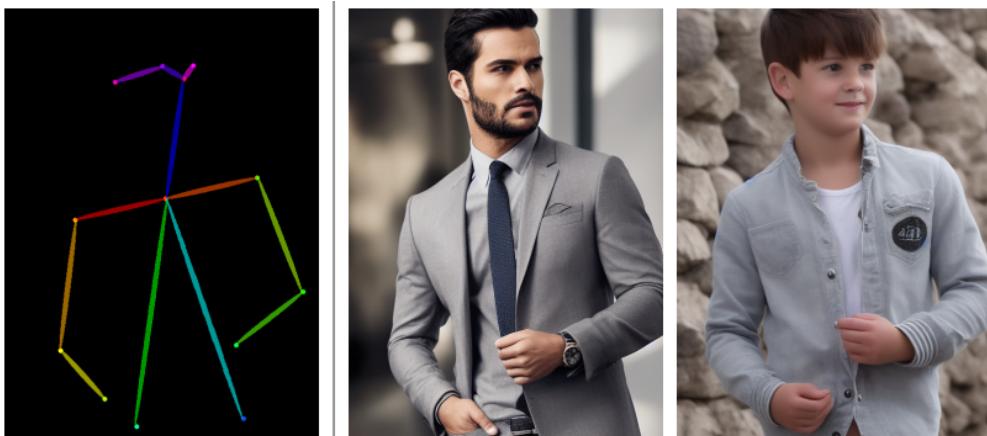


図 11.6: ポーズを指定した生成の例。左のポーズ画像を入力することで、右のようにそのポーズに沿った人間を生成できる。[62] より引用。

ここで紹介したように、Stable Diffusion では一般的なユーザーでも比較的簡単に追加学習することができます。一方、勝手に個人情報を含むデータを使って悪用することもできるという社会的リスクも存在します。画像生成に限らず、今後の生成 AI はユーザーフレンドリーになっていく反面、よ

り一層我々のユーザーのモラルが求められることになります。

11.2 GAN の応用

ここからは実用的に使われる様々な GAN の手法について紹介します。2015 年に GAN の理論を CNN を用いて実装した DCGAN [63] が発表された後、DCGAN の改良のための研究がいくつも発表されました。その中でも、2017 年に発表された PGGAN(ProgressiveGAN) [64] は 1024×1024 (HD 画質と同程度) での画像生成を可能にしました。PGGAN の最大の特徴は Progressive Training という学習手法です。これは、最初は 4×4 の解像度で学習させ、ある程度学習が進めば今度は 8×8 で学習させ…、というように解像度を順々に大きくしていく手法で、高解像度でも安定した学習を可能にしました。2018 年には PGGAN にさらに改良を重ねた StyleGAN [65] が発表されました。StyleGAN はその名の通り”スタイル”を操作することができ、輪郭や顔パーツ、色味のような属性を潜在変数によって指定することができます。また、2019 年には StyleGAN に対して改良を施した StyleGAN2 [66] が発表されました。StyleGAN2 では、StyleGAN の生成画像の不自然さが Progressive Training によって生じているとし、Progressive Training を廃止する手法を提案しています。その結果、学習の効率化と生成画像の品質向上を可能にしました(図 11.7)。



図 11.7: GAN の生成画像。画像はそれぞれ [63, 64, 65, 66] より引用。

その後、2021 年に StyleGAN3 [67] が発表されました。StyleGAN3 は GAN に全体的に見られるエイリアシングという現象を解決した手法です。エイリアシングとは、ある周波数成分を再現しようとした時に本来とは違う周波数成分として再現されてしまうことを言います。例えば、ヘリコプターのプロペラが回る様子をビデオで記録した際にプロペラが止まっているように見えることがあります、これは実際の周波数(実際のプロペラの速度に対応)成分が偽の周波数(ビデオを通すとプロペラが止まって見える=速度 0 に対応)成分として再現されていることになります。画像生成においては生成画像の信号成分と解像度の関係でエイリアシングが起こる可能性があり、実際多くの画像生成モデルではエイリアシングが起きていました。一方、画像生成をするうえで位置不变性、つまり『『された画像を生成した結果』と『生成した画像をずらした結果』が等しい』という性質が成り立たないことが知られていました。StyleGAN3 ではこの位置不变性を崩している原因がエイリアシングになると仮定し、エイリアシングを防ぐことで結果的に位置不变性の獲得に成功しました。この位置不变性は静止画の生成にはあまり役立ちませんが、動画を生成する時に「物体の位置が変化しても見た目は変わらない」という現実世界の性質をより自然に反映できるという大きな利点があります。

最後に、今年発表された GigaGAN [68] について紹介します。StyleGAN を含めて、GAN は基

本的には Diffusion Model ほど多様な画像を生成することは難しいとされていました。しかし、この GigaGAN は FID という評価指標において Stable Diffusion と同レベルの多様性と品質を達成しています。図 11.8 にいくつか生成例を示します。GigaGAN の生成画像の多様性と品質は Stable Diffusion と同レベルですが、GigaGAN の方が 10 倍以上高速に画像を生成できます。生成画像の解像度がより大きくなると GAN の速度優位性が更に際立つため、今後の発展に期待します。



A hot air balloon in shape of a heart. Grand Canyon



low poly bunny with cute eyes



A cube made of denim on a wooden table

図 11.8: GigaGAN の生成画像。画像の下に書いてあるキャプションが入力。[68] より引用。

第 V 部

説明可能な AI(XAI)

第 12 章

LIME

「AIはどこを見て判断しているのだろう」と気になる方は多いでしょう。近年では精度の高いAIが多く開発されていますが、AIの判断理由が分からぬこと(=ブラックボックス化)が問題視されています。XAI(eXplainable AI; 説明可能なAI)の分野では、そんなブラックボックス化したAIを説明する手法が研究されています。ここでは、その中でも代表的な手法である「LIME」「SHAP」「Counterfactual Explanation(反実仮想説明)」についてご紹介します。

LIME(Local Interpretable Model-agnostic Explanations)は2016年に発表された論文[69]で提案された手法です。この論文ではXAIの有用性が多数示され、LIMEを皮切りにXAIの研究が盛んになりました。

12.1 LIMEによる説明の例

まずは、LIMEを使うとどのようにAIの説明ができるのかを具体的に見てみましょう。

■例1: テキスト分類モデル ある文章がキリスト教のものか無神論のものかを分類するAIが、文章のどこを見て判断しているのかをLIMEで調査した結果が図12.1です。左右は、異なる2つのAIの判断根拠を示しています。図中の棒グラフは、どの単語をどれくらい重要視したのかを示しています。例えば左側の棒グラフを見ると「GOD」が一番上にあるので、左側のAIは「GOD」に最も注目して判断していることが分かります。人間の考え方としても「GOD」という単語が含まれているかどうかは判断根拠として重要だと考えられ、左側のAIの判断は妥当といえます。

では、右側のAIはどうでしょうか。棒グラフの上位にあるのは「Posting」や「Host」です。これらの単語は文章のヘッダーに含まれる単語であり、キリスト教か無神論かを判断する上で重要な単語とは言えません。したがって、右側のAIの判断は妥当ではないといえます。

このようにLIMEを使用してAIの判断根拠を調べることができ、AIの信頼性について検証することができます。図の例では左右どちらのAIも「キリスト教の文章」と判断し正解しており、表面上はどちらも同じように見えてしまいます。AIの判断根拠を知ることがいかに重要かを示す例と言えるでしょう。

■例2: 画像分類モデル ある画像がオオカミかハスキー犬かを分類するAIが、画像のどこを見て判断しているのかをLIMEで調べてみた結果が図12.2です。左側の画像は元画像で、右側の画像はAIが着目している部分のみを残しそれ以外はグレーで塗りつぶしたものです。この結果を見ると、動物の本体ではなく、背景の雪の部分に着目して判断てしまっていることが分かります。その原因は学習データにあります。学習データを調査してみると、オオカミの画像の背景には必ず雪が写ってお

り、ハスキー犬の画像の背景には雪が写っていないことが判明しました。つまり、AI は「雪が写っている画像はオオカミ、雪が写っていない画像はハスキー犬」と学習してしまったのです。このように LIME を使用して AI の判断根拠を調査することで、学習方法の問題点に気づくこともできるのです。

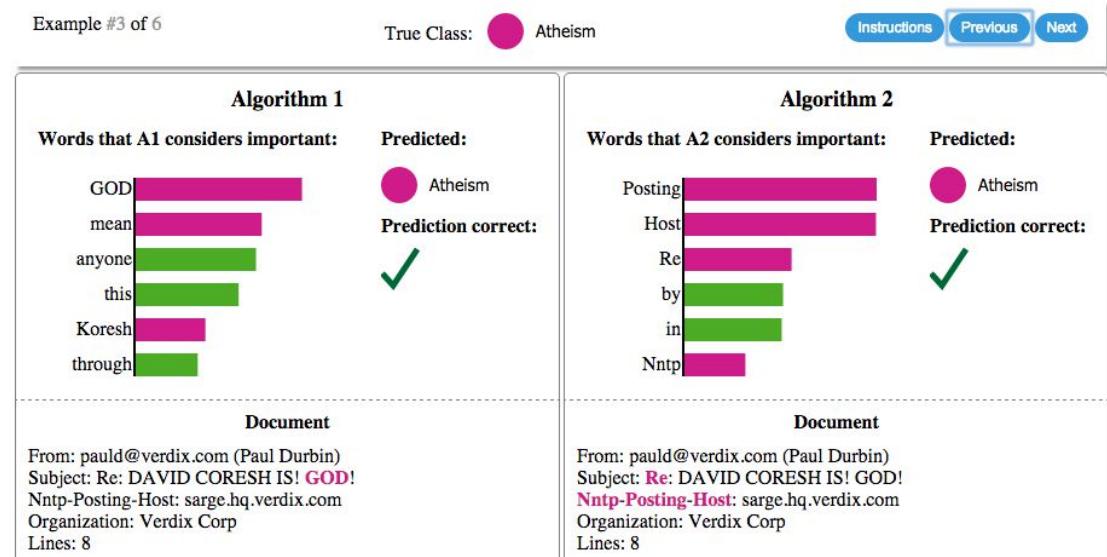


図 12.1: テキスト分類モデルに LIME を適用した例 [69]



図 12.2: 画像分類モデルに LIME を適用した例 [69]

12.2 LIME の仕組みを 3 段階のレベルで説明

LIME がどのような仕組みで AI を説明するのか、易しい順に 3 段階のレベルでこれから説明していきます。

【レベル 1】 LIME とは「AI を理解するための AI」である

ひと言で説明すると、LIME は「AI を理解するための AI を作って解釈を行う手法」です。「犬か猫を判別する AI」を例にとって LIME のアイデアを説明します。AI に画像を入力すると、AI は画

像の特徴を抽出して犬か猫かを判別します。しかし、AI がどのような特徴を抽出して判断しているのかはよく分かりません。そこで LIME は AI の判断理由を説明するために、この「犬か猫を判別する AI」(=以降「元モデル」と呼びます) に似た AI(=以降「説明器」と呼びます)を作ります。

どういうことか、もう少し詳しく説明します。説明器は元モデルと同じように、画像を入力すると犬か猫かを判別します。しかし説明器は元モデルとは異なり、どこを見て判断したのかが分かるモデル、つまり中身が見える (=グラスボックスな) モデルを使って学習を行います。説明器の学習では、説明器が元モデルと同じような振る舞いをするように学習します。つまり、元モデルの”子ども”のような説明器を作ります。このようにして説明器を作れば、中身が見やすい”子ども”のモデルの中身を見ることで、間接的に元モデルの中身を知ることができます。

【レベル 2】元の AI と局所的に似た AI を作る

レベル 1 では、元モデルに似た説明器を作り、説明器の中身を見ることで元モデルの中身を知ることができるなどを説明しました。しかしこのような疑問を持つ方がいるかもしれません。

「元モデルと同じような説明器を作れるのなら、最初から説明器で学習を行えばよいのでは？」

しかし残念ながら、説明器で精度の高いモデルを作ることは難しいと言われています。これには図 12.3 に示す「説明可能性と精度のトレードオフ」が関係しています。つまり、

- 説明がしやすい AI は精度が低い
- 精度が高い^{*1} AI は説明がしづらい

ということです。「説明ができる AI」の例としては、線形回帰モデルや決定木モデルなどがあります。これらのモデルは比較的シンプルであり、入力が予測にどう寄与しているのかが分かりやすいですが、シンプルさゆえに精度は劣ります。一方「精度が高い AI」の例としては、ニューラルネットワークやランダムフォレストなどがあります。これらのモデルは複雑であり、より多様なパターンを学習できるため精度が高いですが、複雑さゆえに入力が予測にどう寄与しているのかが分かりにくくなります。このような理由から、元モデルの精度を保ったまま説明性の高いモデルを作ることは難しいのです。

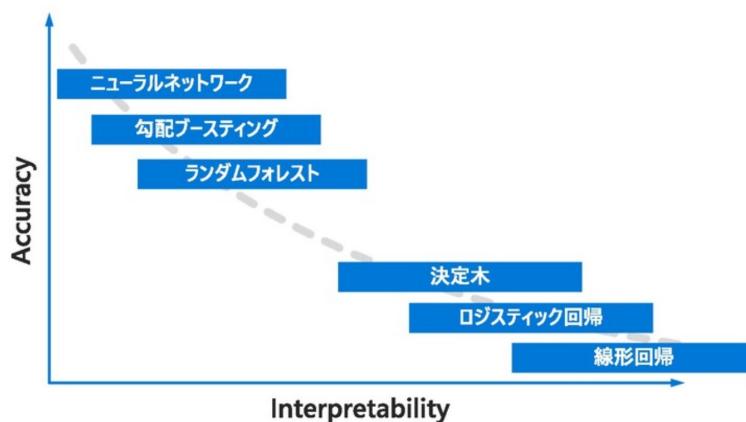


図 12.3: AI の説明可能性と精度のトレードオフ [70]

しかし、データ全体に対して元モデルの精度を保つのは難しいとしても、データの一部に対して元

^{*1} 「表現力が高い」と言った方が正確かもしれません。

モデルの精度を保つことは可能です。図 12.4 は、元モデルの決定境界と説明器の決定境界を示しています。全体で見ると元モデルは複雑な形をしていますが、あるデータ点(大きな“+”マーク)の周辺に着目すると、元モデルの決定境界は直線に近い形をしています。このように、あるデータ点の周辺のみに着目すると元モデルの振る舞いが線形に近くなることはよくあります。そこで、元モデルを局所的に線形回帰モデルで近似してしまいます。先ほど述べたように線形回帰モデルは説明がしやすいモデルの 1 つなので^{*2}、この近似モデルを説明器として用いることで、元モデルの説明を行います。以上が LIME のざっくりとしたアイデアです。

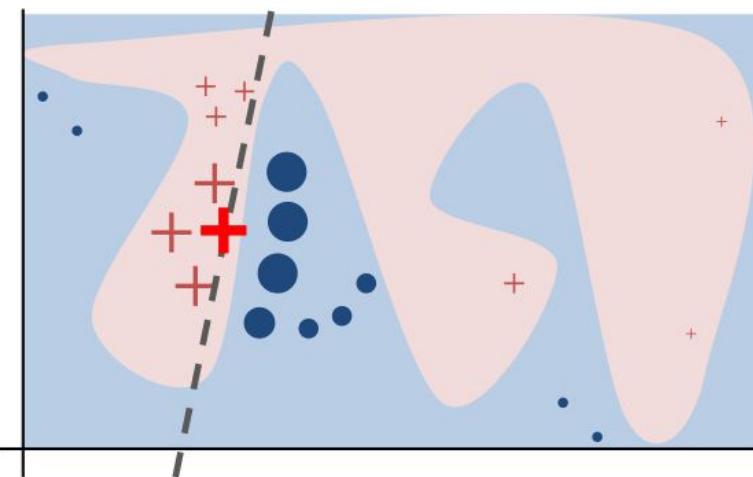


図 12.4: 元モデルの決定境界(塗りつぶし)と説明器の決定境界(点線)[69]

線形回帰モデルはなぜ説明がしやすいモデルなのか?

線形回帰モデルでは次のような式を使って予測を行います。

$$g(x_1, \dots, x_n) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_n x_n \quad (12.2.1)$$

x_1, \dots, x_n は入力で、パラメータ $\beta_0, \beta_1, \dots, \beta_n$ は学習データを用いて最適化されます。例えば x_1 を年齢(歳)、 x_2 を身長(cm)、 x_3 を体重(kg)として、年収(万円)をこのモデルで予測することを考えます。データを学習させた結果が下のようになります(数字は適当です)。

$$g(x_1, x_2, x_3) = 200 + 6x_1 + 0.8x_2 - 0.7x_3 \quad (12.2.2)$$

この式からいくつか読み取ることができます。例えば x_1 の係数は 6 なので、年齢が 1 歳上がると年収は 6 万円上がる事が分かります。また x_2 の係数は 0.8 なので、身長が 1cm 上がると年収は 0.8 万円上がる事が分かります^a。このように、線形回帰モデルでは係数を見ることで各入力が予測にどのように寄与しているのかが分かるので、説明性の高いモデルと言えます。

^a 因果関係があるような書き方をしていますが、線形回帰でこのような結果が得られたからといって因果関係があると断定することはできません。モデルはあくまで入力と出力の相関関係を学習しているだけです。

^{*2} 線形回帰モデルの代わりに、決定木なども用いることができます。

【レベル 3】説明器を作るための目的関数

では、どのようにして元モデルと似た説明器を作るのでしょうか？まずは説明器が満たしてほしい条件を考えてみましょう。論文 [69] では、次の 2 つの条件を満たす説明器を作ることを目指しています。

- 説明対象のデータ点 x の近傍で、説明器 g と元モデル f が近似的に一致すること
- 説明器 g がシンプルであること

これらを満たす説明器を作るには、例えば式 (12.2.3) のような目的関数を考え、これを最小化するような説明器 g を見つけければよいです。

$$\mathcal{L}(f, g, \pi_x) + \Omega(g) \quad (12.2.3)$$

目的関数 (12.2.3) の第 1 項 $\mathcal{L}(f, g, \pi_x)$ は、データ点 x の近傍で元モデル f と説明器 g が似ているほど小さくなる関数です。 π_x はデータ x との類似度を示す関数で、「データ点 x の近傍」の「近傍」がどの程度の近傍なのかを制御します。説明器 g が線形回帰モデルの場合、 $\mathcal{L}(f, g, \pi_x)$ は次のような形が考えられます。

$$\mathcal{L}(f, g, \pi_x) = \sum_{z, z' \in Z} \pi_x(z)(f(z) - g(z'))^2 \quad (12.2.4)$$

$$\pi_x(z) \equiv \exp(-d(x, z)^2 / \sigma^2) \quad (d : \text{距離関数}) \quad (12.2.5)$$

式 (12.2.4) は、データ点 x 近傍のデータ点 z と z' について^{*3} 元モデル f と説明器 g の出力の差の二乗を足し合わせたものです。ただし x からの距離に応じて、式 (12.2.5) のような類似度関数 π_x で重み付けされています。一方、目的関数 (12.2.3) の第 2 項 $\Omega(g)$ は、説明器 g がシンプルであるほど小さくなる関数です。論文 [69] では説明器 g は線形回帰モデルと仮定し、 $\Omega(g)$ は「説明器 g の非ゼロな回帰係数の個数が K 個以下なら 0、 K 個を超えたら ∞ 」として定義されています^{*4}。

このようにして目的関数 (12.2.3) を定義し最小化問題を解くことで、データ x に対する説明器 g を得ることができます。

^{*3} z, z' はどちらも実質的には同じものを表していますが、プライム「 $'$ 」がついた変数は解釈性の高い表現に変換したものだということを表しています。例えば自然言語では、

z : 単語ベクトル (人間には理解不能な表現) z' : One-hot ベクトル (人間が理解しやすい表現)

を表しています。入力変数に解釈性のある表現を用いることで、LIME で得られる説明器の解釈性が高まります。

^{*4} これを式で書くと次のようになります:

$$\Omega(g) = \infty \mathbb{1}[\|w_g\|_0 > K]$$

ただし、 $\mathbb{1}$ は定義関数 (指示関数、特性関数)、 w_g は g の回帰係数を並べたベクトル、 $\|\cdot\|_0$ は L0 ノルム (ベクトルの非ゼロ要素の個数) です。

第 13 章

SHAP

次に説明可能 AI の一つである SHAP について、結果の見方やその仕組みについて説明します。まずは、SHAP の結果の見方について説明します。次の図 13.1 は SHAP の Waterfall と呼ばれるメソッドを用いた結果です。

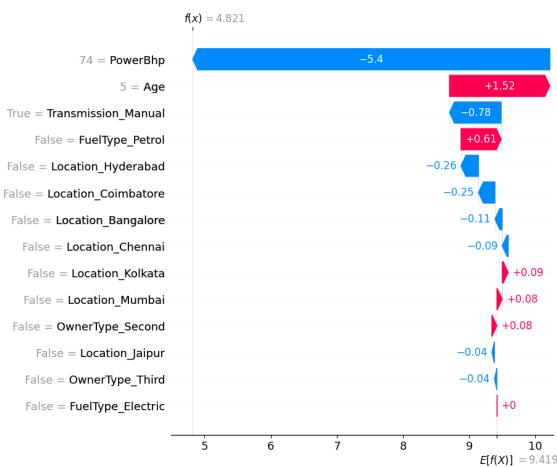


図 13.1: SHAP による影響度の可視化 [71]

図 13.1 はある自動車の値段を線形回帰により予測した際の特徴量の影響を SHAP を用いて表しています。青の矢印が示されている特徴量は自動車の値段を下げる方向に影響を与えていていることを、赤の矢印が示されている特徴量は自動車の値段を上げる方向に影響を与えていていることを表しています。このように SHAP を用いることにより特徴量の影響を可視化することができますが、特徴量の影響とはどのように調べられるのでしょうか。

13.1 シャープレイ値

SHAP が特徴量の予測への影響を測る際に用いるのがシャープレイ値という値です。ここではシャープレイ値がどのようなアイデアから作られたかということと、その数式的な説明をします。

13.1.1 シャープレイ値のアイデア

シャープレイ値のアイデアを理解するために次の問題を考えてみましょう。[72]

3 人のアルバイト A さん、B さん、C さんがいます。この三人が単独でそれぞれバイトをした場合

と協力してバイトをした場合に次の表に書かれた額だけ稼げるとします。

バイトに参加する人	稼げる金額(万円)
A	6
B	4
C	2
A, B	20
A, C	15
B, C	10
A, B, C	24

表 13.1: バイトに参加する人の組み合わせと稼げる額

今 A, B, C が三人でバイトを行い 24 万円を稼いだとします。この時 24 万円はどのように分配したらいいでしょうか。気持ち的には貢献度の高い人に多く分配したいところです。そこである人がバイトに参加した場合にどれだけ多く稼げたかということを考えます。例えば既に A さんと C さんがバイトをしていて、B さんもバイトに参加したとすると

$$24 - 15 = 9$$

となり、A さんと C さんだけでバイトを行っているよりも 9 万円多く稼げることになるので B さんは 9 万円分の貢献をしたということになります。このようにその人が参加したかどうかでどれくらい稼げる額に変化があったかを調べることでその人の貢献度が分かります。ただし注意する点として貢献度は参加する順番に依存するということです。例えば A さんと C さんがバイトをしている時に、B さんも参加したとすると B さんの貢献度は 9 万円でした。一方で A さんだけがバイトをしている際に、B さんもバイトに参加したとすると B さんの貢献度は

$$20 - 6 = 14$$

となり、B さんは 14 万円分の貢献をしたことになります。そのため B さんが参加したタイミングごとに貢献度を調べて、その平均をとるというのが良さそうです。

さて上の例の A さん、B さん、C さんを機械学習の特徴量、稼いだ金額を機械学習の予測値として見ると先ほどの問題は特徴量の貢献度を測っているということになります。これがシャープレイ値のアイデアです。つまりシャープレイ値はある特徴量をモデルに含めた場合に含めなかった場合と比べてどれくらい予測精度が上がったかを見ることによって特徴量の貢献度を調べているというわけです。

13.1.2 シャープレイ値の具体的な計算方法

ここでは先ほどのアイデアを元にシャープレイ値の具体的な計算方法について説明します。 f を予測モデル、 F を予測モデルに用いた特徴量全体の集合とします。このとき特徴量 $i \in F$ のシャープレイ値は次の式で計算されます。

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} [f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S)] \quad (13.1.1)$$

ここで、 x_S は S に含まれる特徴量をモデル f の入力に用いたという意味です。また、 $S \subseteq F \setminus \{i\}$ は F から i を除いた集合を表しています。式 (13.1.1) では 13.1.1 節で述べた通り特徴量 i の貢献度を計算し、それを特徴量 i が予測に参加した時の状態全てについて足し合わせてその平均を取ることを意味しています。

13.1.3 シャープレイ値の性質

ここまでシャープレイ値のアイデアと計算方法について説明しました。ここでは式 (13.1.1) が持つ性質について Additive feature Attribution Methods という説明可能モデルの観点から説明します。

以下、 f を元の予測モデルとし \mathbf{x} を f への入力とします。また g を説明可能モデルとします。この時 g への入力を \mathbf{x} ではなく、ある関数 h_x について

$$\mathbf{x} = h_x(\mathbf{x}') \quad (13.1.2)$$

を満たす \mathbf{x}' とします。式 (13.1.2) により説明可能モデル g への入力 \mathbf{x}' は \mathbf{x} に比べて単純化されます。 h_x の例としてある特徴量 \mathbf{x} が用いられていれば 1、そうでなければ 0 とするものがあります。つまり、

$$h_x^{-1}(\mathbf{x}_i) = \begin{cases} 1 & (\mathbf{x}_i \text{ がモデル } f \text{ に使われている}) \\ 0 & (\mathbf{x}_i \text{ をモデル } f \text{ に使われていない}) \end{cases} \quad (13.1.3)$$

です。以下では、 h_x が式 (13.1.3) の場合について説明します。

この時、 $\mathbf{z}' \sim \mathbf{x}'$ ならば $g(\mathbf{z}') \sim f(h_x(\mathbf{z}'))$ を満たすような g を作ることを Local Methods といい、特に g の形として次式のような形を用いる方法を Additive Feature Methods といいます。

$$g(\mathbf{z}') = \phi_0 + \sum_{i=1}^M \phi_i z'_i \quad (13.1.4)$$

ここで各 z'_i について $z'_i \in \{0, 1\}$ であり、 M は単純化された特徴量の数を表します。式 (13.1.4) は元のモデルの出力を各特徴量の貢献度 ϕ_i の和で f の出力を近似しようとしていることを意味します。

式 (13.1.4) の ϕ_i については様々な指定方法がありますが、ここでは ϕ_i が次の 3 条件を満たすとします。

性質 1. (Local Accuracy) $h_x(\mathbf{x}') = \mathbf{x}$ の時、

$$f(\mathbf{x}) = g(\mathbf{x}') \quad (13.1.5)$$

が成り立つ。

性質 (1) は元のモデル f と説明可能モデル g の値が少なくとも $\mathbf{x} = h_x(\mathbf{x}')$ の時には等しいということを主張しています。

性質 2. (Missingness)

$$\mathbf{x}'_i = 0 \Rightarrow \phi_i = 0 \quad (13.1.6)$$

性質2は特徴量 x'_i の値が0ならばその特徴量はモデルの予測に貢献しない、つまり $\phi_i = 0$ であるべきということを主張しています。

性質3. (Consistency) $f(h_x(\mathbf{z}'))$ を $f(\mathbf{z}')$ で表し、 $\mathbf{z}'_i = 0$ を $\mathbf{z}' \setminus i$ で表すこととする。この時任意の入力 \mathbf{z}' について任意の二つのモデル f , f' が

$$f'_x(\mathbf{z}') - f'_x(\mathbf{z}' \setminus i) \geq f_x(\mathbf{z}') - f_x(\mathbf{z}' \setminus i) \quad (13.1.7)$$

をみたすとき

$$\phi_i(f', \mathbf{x}) \geq \phi_i(f, \mathbf{x}) \quad (13.1.8)$$

が成り立つ。

性質3はある特徴量 i のモデル f' における貢献度がモデル f における貢献度以上の場合には、モデル f' における ϕ_i の値はモデル f における ϕ_i の値以上であるべきだということを主張しています。この時、性質1~3を満たす ϕ_i は一意に決まり、次のように表すことが出来ることが知られています。

$$\phi_i(f, \mathbf{x}) = \sum_{\mathbf{z}' \subseteq \mathbf{x}'} \frac{|\mathbf{z}'|!(|M| - |\mathbf{z}'| - 1)!}{|M|!} [f_x(\mathbf{z}') - f_x(\mathbf{z}' \setminus i)] \quad (13.1.9)$$

ここで $|\mathbf{z}'|$ は成分が1の数、 $\mathbf{z}' \subseteq \mathbf{x}'$ は \mathbf{x}' の0でない要素全体からなる部分集合を表しています。式(13.1.9)を見ると式(13.1.1)で定義したシャープレイ値に似た形をしていることが分かります。実際に式(13.1.9)の値はシャープレイ値に一致することが知られています。以上よりシャープレイ値によって計算される特徴量は性質1~3を満たすことが分かります。

第 14 章

タラレバを機械で実現する方法

今まででは、説明可能な AI の実現手段として LIME や SHAP を用いて説明してきました。この章では反実仮想説明 (Counterfactual Explanations, CE) について説明します。

14.1 反実仮想説明とは

反実仮想という言葉を聞いたことがあるでしょうか。もしかしたら高校の古文授業などで出てきたことがあるかもしれません。簡単にいうと「もし～だったら～となるだろう」という実際には起こり得ない状況を想像して予測することです。

説明可能 AI でも反実仮想「説明」という形で、同じような概念が出てきます。今まで説明してきた LIME や SHAP は局所説明法と呼ばれ、「予想の根拠となる特徴量を明示する」ものでした。それに対して、今回扱う反実仮想説明では、「所望の結果を得るために特徴量をどう変化させればよいかを自動で生成してくれる」ものです。

例えばこのような状況を考えてみましょう。ある大学の奨学金制度では、その審査に年齢や成績、両親の経済状況などから総合的に判断する機械学習モデルを構築して運用されているとします。

この奨学金の審査に合格したいあなたは、奨学金機構に申請をしました。一ヶ月後、待ちに待った結果通知では審査に落ちてしまったことしか書かれておらず、なぜ不合格だったのかが明記されていません。納得できないあなたは、運営に問い合わせたのですが、運営側にも機械学習モデルがこのような結果を出したのかがわからないようです。

このように機械学習の分野では、実装した人でも自分で作ったモデルがなぜこのような判断をしたのかが想像がつかないことはよくあります。この際に役立つのが、前節で説明した LIME や SHAP です。

LIME や SHAP を用いると、例えば両親の経済状況や成績などその判断で用いられた重要なファクターを出力し、変更しなければならない特徴量を教えてくれます。しかし LIME や SHAP では、それをどれだけ変更すればよいのかは教えてくれません。両親の経済状況など、自分では変えることができない要因も出力するかもしれません。そこで役に立つのが反実仮想説明です。

反実仮想説明では「成績の平均を 3.5 まで上げれば審査に合格する」などといった具体的なシナリオを提示することができます。そのシナリオも適当なものではなく、今の状況からなるべく達成しやすくかつ、いろいろなシナリオを提示することができます。つまり、局所説明法では「なぜ」という間に答えてくれるのに対して、反実仮想説明では「どうすれば」という具体的な行動案を提示してくれるというところにポイントがあります。

以降では、反実仮想説明において有名なライブラリである DiCE を取り上げて、反実仮想説明の仕

組みについて紹介していきます。

14.2 反実仮想説明の仕組み

ここでは、反実仮想説明の代表例である、DiCE(Diverse Counterfactual Explanations)[73] いうライブラリとその論文で紹介されているアルゴリズムを紹介します。

審査結果に合格できるようなシナリオを得るために、成績や年齢など変数の値をいろいろ変化させれば見つかるかもしれません。しかし、無造作に変数を調整しても思うようにはいきません。審査に合格する変数の組み合わせは膨大にありますし、年齢などのようにすぐに変化させることができない変数も含まれるからです。

14.2.1 理想的な反実仮想の条件

まずここでは、理想的な反実仮想の条件を考えてみましょう。

1. 反実仮想は、望んだ結果を出力する
2. 反実仮想は、現在の状況と似ている
3. 変数の変化が現実的である
4. 反実仮想が多様である

1は当然ですね。

2は、例えば成績をすべて最高値にすれば奨学金には確実に合格するでしょうが、実際には大変すぎてやりたくないかもしれませんよね。なるべくまだ頑張れそうだなというシナリオを設定するための条件が2です。

3は、両親の経済状況や年齢などの変更が難しい変数はシナリオからは除外したいという条件です。

4は、1~3の条件が満たされたシナリオを複数出力するとき、なるべく多様なシナリオを出力させたいという条件です。

14.2.2 反実仮想のモデリング

上記の制約を満たすように変数を変化させれば、理想的な反実仮想を見つけ出せそうです。

一つのシナリオだけを出力すれば良い場合、基本的なアイデアは以下の式で表現されます[74]。

$$\mathbf{c} = \underset{\mathbf{c}}{\operatorname{argmin}} \text{loss}(f(\mathbf{c}), y) + |\mathbf{c} - \mathbf{x}| \quad (14.2.1)$$

使われている文字：

- f : 元の機械学習モデル
- y : 望ましい結果
- \mathbf{x} : 入力する特徴量
- \mathbf{c} : シナリオ

この式では、第一項で反実仮想が望ましい結果 y を出力するという条件(条件1)を、第二項で現在の状況 \mathbf{x} と反実仮想 \mathbf{c} の距離を計算することで、現在の状況と似ているという条件(条件2)を表現しています。

この式をもとにして、DiCE では条件 3 と条件 4 を組み込み、以下の式で最適化しています

$$C(\mathbf{x}) = \operatorname{argmin}_{c_1, \dots, c_k} \frac{1}{k} \sum_{i=1}^k \text{loss}(f(c_i), y) + \frac{\lambda_1}{k} \sum_{i=1}^k \text{dist}(c_i, \mathbf{x}) - \lambda_2 \text{dpp_diversity}(c_1, \dots, c_k) \quad (14.2.2)$$

使われている文字：

- $C(\mathbf{x})$: 反実仮想の集合
- k : 反実仮想の数
- f : 元の機械学習モデル
- y : 望ましい結果
- \mathbf{x} : 入力する特徴量
- $\text{dist}(\cdot)$: 現在の状況からの距離 (l_1 距離などが想定される)
- $\text{dpp_diversity}(\cdot)$: 行列点過程の多様性 (後述)
- λ_1, λ_2 : ハイパーパラメータ

この式では、第一項で、反実仮想が望ましい結果 y を出力するという条件 (条件 1) を、第二項で現在の状況からの距離つまり現在の状況と似ているという条件 (条件 2) を、第三項でシナリオ全体の多様性 (条件 4) を表現しています。

また λ_1 や λ_2 を調整することで、多様性や現在の状況からの距離の重み付けを調整することができます。

式中で出てきた dpp_diversity は、条件 4 を反映させるために導入されています。複数の反実仮想 (c) を考える場合、多様なパターンがあったほうが望ましく、似たようなベクトルは何個もいりません。そのため、複数の反実仮想ベクトル (c_i) 間の距離を定義し、その値ができるだけ遠くなるように最適化をかけます。よって $\text{dpp_diversity}(\cdot)$ は行列式を用いて以下のように表すことができます。

$$\text{dpp_diversity} = \det(\mathbf{K}) \quad (14.2.3)$$

$$K_{i,j} = \frac{1}{1 + \text{dist}(c_i, c_j)} \quad (14.2.4)$$

さらに、変更できない特徴量が存在することも考慮しなければなりません (条件 3)。例えば、性別や年齢が若くなったりするなどです。DiCE では、そのようなケースに対応するために、出力から除外する特徴量を選択できるようになっています。

DiCE では以上のようなアルゴリズムをもとに実装されています。以下の URL から DiCE にアクセスできるので興味のある人は実際に使ってみると良いでしょう。

<https://github.com/interpretml/DiCE>

参考文献

- [1] 山田亜虎, 酒井邦嘉. ブローカ野における文法処理. *Brain and nerve*, Vol. 69, No. 4, pp. 479–487, 2017.
- [2] ユヴァル・ノア・ハラリ (著), 柴田裕之 (訳). 「サピエンス全史 (上) 文明の構造と人類の幸福」. 河出書房新社, 2016.
- [3] ニック・ボストロム (著), 倉骨彰 (訳). 「スーパーインテリジェンス: 超絶AIと人類の命運」. 日本経済新聞出版, 2017.
- [4] Robin Hanson. Shall we vote on values, but bet on beliefs? *Journal of Political Philosophy*, Vol. 21, No. 2, pp. 151–178, 2013.
- [5] 岡田章. ゲーム理論の歴史と現在. 経済学史研究, Vol. 49, No. 1, pp. 137–154, 2007.
- [6] John Von Neumann and Oskar Morgenstern. Theory of games and economic behavior, 2nd rev. 1947.
- [7] Claude E Shannon. Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, Vol. 41, No. 314, pp. 256–275, 1950.
- [8] Enders A Robinson. Cybernetics, or control and communication in the animal and the machine, 1963.
- [9] 池原正戈夫, 彌永昌吉, 室賀三郎, 戸田巖訳. ウィーナー サイバネティックス. 岩波書店, 2011.
- [10] Michael Wooldridge 著, 神林靖訳. AI技術史 考える機械への道とディープラーニング. インプレス, 2022.
- [11] John McCarthy, Marvin L Minsky, Nathaniel Rochester, and Claude E Shannon. A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, Vol. 27, No. 4, pp. 12–12, 2006.
- [12] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, Vol. 9, No. 1, pp. 36–45, 1966.
- [13] Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. 1971.
- [14] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
- [15] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.
- [16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N

- Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, Vol. 30, , 2017.
- [17] OpenAI. Openai introducing chatgpt. <https://openai.com/blog/chatgpt>. (accessed 2023-09-17).
- [18] Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, Vol. 33, pp. 3008–3021, 2020.
- [19] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, Vol. 33, pp. 1877–1901, 2020.
- [20] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, Vol. 35, pp. 24824–24837, 2022.
- [21] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, Vol. 35, pp. 22199–22213, 2022.
- [22] Viet Dac Lai, Nghia Trung Ngo, Amir Pouran Ben Veyseh, Hieu Man, Franck Dernoncourt, Trung Bui, and Thien Huu Nguyen. Chatgpt beyond english: Towards a comprehensive evaluation of large language models in multilingual learning. *arXiv preprint arXiv:2304.05613*, 2023.
- [23] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. *arXiv preprint arXiv:2309.03409*, 2023.
- [24] Benj Edwards. Telling ai model to “take a deep breath” causes math scores to soar in study. <https://arstechnica.com/information-technology/2023/09/telling-ai-model-to-take-a-deep-breath-causes-math-scores-to-soar-in-study/>, 2023. (accessed 2023-11-05).
- [25] Rajasekhar Reddy Mekala, Yasaman Razeghi, and Sameer Singh. Echoprompt: Instructing the model to rephrase queries for improved in-context learning. *arXiv preprint arXiv:2309.10687*, 2023.
- [26] garante per la protezione dei dati personali (Italy). Artificial intelligence: stop to chatgpt by the italian sa personal data is collected unlawfully, no age verification system is in place for children. 2023.
- [27] 田中浩之, 河瀬季, 古川直裕, 大井 哲也力太, 佐藤健太郎, 柴崎拓, 橋詰卓司, 仮屋崎崇, 唐津真美, 清水音輝, 松尾剛行. ChatGPT の法律. 中央経済社, 2023.
- [28] Nestor Maslej, Loredana Fattorini, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, Helen Ngo, Juan Carlos Niebles, Vanessa Parli, et al. Artificial intelligence index report 2023. *arXiv preprint arXiv:2310.03715*, 2023.
- [29] Supantha Mukherjee and Giselda Vagnoni. Italy restores chatgpt after openai responds to regulator. 2023.
- [30] Information Commissioner’s Office. Guidance on ai and data protection. 2023.

- [31] the white house. Blueprint for an ai bill of rights: A vision for protecting our civil rights in the algorithmic age. 2022.
- [32] OpenAI. Chatgpt. <https://openai.com/chatgpt>. (accessed 2023-10-25).
- [33] OpenAI. Gpt-4 technical report, 2023.
- [34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [35] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.
- [36] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- [37] ZDNET Japan. マイクロソフト、生成 ai の国内導入は 560 社以上と報告—パートナー新施策も予告. <https://japan.zdnet.com/article/35210604/>. (accessed 2023-10-25).
- [38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [39] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [40] Michael Buro. The othello match of the year — takeshi murakami (0) vs. logistello (6). <https://skatgame.net/mburo/event.html>, 1997. (accessed 2023-09-29).
- [41] 株式会社 AVILEN. 最弱オセロ. <https://corp.avilen.co.jp/projects/othello/>. (accessed 2023-09-29).
- [42] Michael Buro. Experiments with multi-probcut and a new high-quality evaluation function for othello. *Games in AI Research*, Vol. 34, No. 4, pp. 77–96, 1997.
- [43] 大槻知史. 最強囲碁 AI アルファ碁 解体新書 増補改訂版. 翔泳社, 2018.
- [44] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, Vol. 529, No. 7587, pp. 484–489, 2016.
- [45] Stack Exchange Inc. Why is a constant plane of ones added into the input features of alphago? <https://ai.stackexchange.com/questions/11014/why-is-a-constant-plane-of-ones-added-into-the-input-features-of-alphago>. (accessed 2023-10-04).
- [46] 山名琢翔. Egaroucid 自己対戦の棋譜. <https://www.egaroucid.nyanyan.dev/ja/technology/transcript/>. (accessed 2023-09-29).
- [47] 山岡忠夫. 強い将棋ソフトの創りかた: Python で実装するディープラーニング将棋 AI. マイナビ出版, 2021.
- [48] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi

- Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pp. 618–626, 2017.
- [49] 山岡忠夫. どの駒が評価値に寄与しているかを可視化する. <https://tadaoyamaoka.hatenablog.com/entry/2022/12/16/213308>, 2022. (accessed 2023-09-18).
- [50] 山岡忠夫. 続：どの駒が評価値に寄与しているかを可視化する. <https://tadaoyamaoka.hatenablog.com/entry/2023/01/17/234944>, 2023. (accessed 2023-09-18).
- [51] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, Vol. 33, pp. 6840–6851, 2020.
- [52] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, Vol. 27, , 2014.
- [53] Diplav, Gaurav Singh, and Srinath Naik Ajmeera. Mode collapse in generative adversarial networks: A survey. https://web.cs.ucla.edu/~srinath/static/projects/CS260_Course_Project_Report_.pdf.
- [54] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, Vol. 34, pp. 8780–8794, 2021.
- [55] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.
- [56] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.
- [57] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, Vol. 35, pp. 5775–5787, 2022.
- [58] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in Neural Information Processing Systems*, Vol. 35, pp. 26565–26577, 2022.
- [59] Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *arXiv preprint arXiv:2208.01618*, 2022.
- [60] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22500–22510, 2023.
- [61] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [62] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023.
- [63] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

- [64] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.
- [65] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.
- [66] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8110–8119, 2020.
- [67] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, Vol. 34, pp. 852–863, 2021.
- [68] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10124–10134, 2023.
- [69] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1135–1144, 2016.
- [70] 女部田啓太. Azure machine learning ハンズオンシリーズ モデル解釈編. <https://speakerdeck.com/konabuta/azure-machine-learning-hanzzon-moderujiye-shi?slide=16>. Speaker Deck (accessed 2023-09-19).
- [71] Pradeeptha Mishra. 実践 XAI 機械学習の予測を説明するための Python コーディング. インプレス, 2023.
- [72] 森下光之助. Shap(shapley additive explanations) で機械学習モデルを解釈する. <https://dropout009.hatenablog.com/entry/2019/11/20/091450#f-79fa9b71>. (accessed 2023-09-26).
- [73] Ramaravind K Mothilal, Amit Sharma, and Chenhao Tan. Explaining machine learning classifiers through diverse counterfactual explanations. In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pp. 607–617, 2020.
- [74] Sandra Wachter, Brent Mittelstadt, and Chris Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, Vol. 31, p. 841, 2017.
- [75] Julen Etxaniz, Gorka Azkune, Aitor Soroa, Oier Lopez de Lacalle, and Mikel Artetxe. Do multilingual language models think better in english? *arXiv preprint arXiv:2308.01223*, 2023.
- [76] Davis E King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, Vol. 10, pp. 1755–1758, 2009.
- [77] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, Vol. 35, pp. 27730–27744, 2022.
- [78] 斎藤康毅. ゼロから作る Deep Learning 4 —強化学習編. O'Reilly Japan, 2022.
- [79] 斎藤康毅. ゼロから作る Deep Learning 2 —自然言語処理編 O'Reilly Japan. O'Reilly Japan,

2022.

- [80] Fédération Française d'Othello. La base wthor. <https://www.ffothello.org/informatique/la-base-wthor/>. (accessed 2023-09-29).
- [81] Lilian Weng. From gan to wgan. *arXiv preprint arXiv:1904.08994*, 2019.
- [82] 岡野原 大輔. 拡散モデル - データ生成技術の数理 -. 岩波書店, 2023.
- [83] 平井 有三. はじめてのパターン認識 ディープラーニング編. 森北出版, 2022.
- [84] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, Vol. 30, , 2017.

本会誌について

執筆者

第Ⅰ部：AIの歴史と未来 有福遼太郎

第Ⅱ部：ChatGPT 大澤衡正（2章）、山下素数（3章）、池田輝（4章）、有福遼太郎（5・6章）

第Ⅲ部：ゲームAI 松田拓巳（7章）、宮前明生（8章）、戸田庸介（9章）

第Ⅳ部：画像生成 前田康介（10章）、三宅大貴（11章）

第Ⅴ部：XAI 松田拓巳（12章）、前田康介（13章）、有福遼太郎（14章）

京都大学人工知能研究会 KaiRA

代 表 松田拓巳

活動日時 毎週木曜日 18:30～

活動場所 京都大学附属図書館 3階 共同研究室5（変更の可能性あり）

入会資格 大学、学部、回生問わず、AIに興味がある方

会 費 無料

活動内容 機械学習に関する本の輪読会、学んだ知識を活かした実装開発、kaggleへの参加など

会誌 vol.7

発行日	2023年11月吉日 初版発行
発行	京都大学人工知能研究会 KaiRA
公式サイト	https://kyoto-kaira.github.io/
メールアドレス	kyoto.kaira@gmail.com
X(旧Twitter)	https://twitter.com/kyoto_kaira