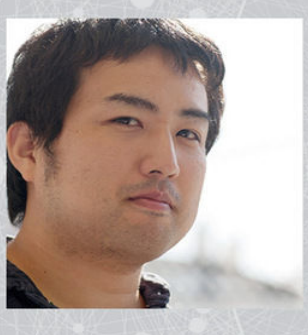
AI技術の最前線

これからのAIを読み解く先端技術73

Preferred Networks 共同創業者
岡野原 大輔 著

NIKKEI Robotics



世界のAI技術の今を手加減なしで執筆

多様体仮説、機械学習の新べき乗則、宝くじ仮説、メタ学習、陰関数微分、世界モデル、環境乱択化、因果と相関、Perceiver、離散化生成モデル、GLOM、AlphaFold、ニューラル空間表現、Transformer、FastWeight、進化戦略、DROIDSLAM、物理を考慮したNNシミュレーション、ほか

日経Roboticsの名物連載を書籍化

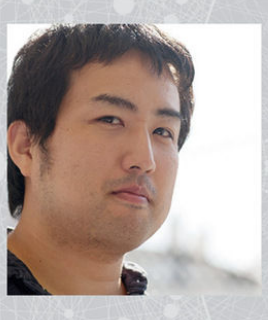
日経BP

AI技術の最前線

これからのAIを読み解く先端技術73

Preferred Networks 共同創業者
岡野原 大輔 著

NIKKEI Robotics



世界のAI技術の今を手加減なしで執筆

多様体仮説、機械学習の新べき乗則、宝くじ仮説、メタ学習、陰関数微分、世界モデル、環境乱択化、因果と相関、Perceiver、離散化生成モデル、GLOM、AlphaFold、ニューラル空間表現、Transformer、FastWeight、進化戦略、DROIDSLAM、物理を考慮したNNシミュレーション、ほか

日経Roboticsの名物連載を書籍化

日経BP

AI技術の最前線

これからのAIを読み解く先端技術73

Preferred Networks 共同創業者
岡野原 大輔 著

NIKKEI Robotics

日経BP

まえがき

この本は私が『日経Robotics』誌に2015年7月から毎月連載している「AI最前線」を一つの冊子にまとめたものです。私が毎日読んでいる論文や、見聞きしたニュースの中から重要な話題を取り上げ書いて来ました。

進歩が早いAI業界においては速報性も大事であり、一番早い例だと重要な論文や研究成果が世の中に公開された場合、1日以内に記事に書き上げ半月以内で誌面に掲載される例もありました。一般の方が読む記事としてはウェブ記事も含めて国内のみならず世界最速であっただろうという自負があります。

近年のAIの進展は著しいものでした。2012年にAlexNetと呼ばれるニューラルネットワークが一般物体認識コンテスト（ImageNet Large Scale Visual Recognition Challenge: ILSVRC）で2位以下に大差をつけて優勝し注目された、いわゆる「AlexNetモーメント」が起きてから急速に世界中の研究者や企業が注目し始め、それからはほぼ毎年、毎月のように世の中をあっといわせるような成果が登場してきました。本書が取り上げる記事はそのような成果の多くをほぼリアルタイムで伝えたものです。

一方、こうした研究や成果は突然登場したものではなく、全て萌芽的な研究や取り組みから逐次的、連続的に発展していったものです。例えば、2022年春頃に登場したDALL·E 2やImagenは、テキストから画像を驚異的な忠実性や表現力で生成でき世の中を驚かせました。この中で使われている拡散確率モデル（Diffusion based probabilistic model）に必要な技術は徐々に成長していきました。2014年の変分自己符号化器（VAE）、2017年のVariational Walkback、2018年のNeural ODEであり、さらに源流を辿れば1995年に登場したヘルムホルツマシンにたどり着きます。これらの先駆的な研究は、結果だけ見れば実用化には程遠いようなものでしたが、研究者の視点からは今後の可能性がみられる興味深いものであり、着実に進歩していきました。先程挙げた手法（VAE, Variational Walkback, Neural ODE）はまだ世の中で注目されていない段階で、全て登場時点で取り上げてきました。

まだ世の中で実用化まで至っていないものの注目している技術も多く取り上げています。例えばNeRF（Neural Radiance Field）などによる新視点シーン生成は今後実世界にAIを導入していく上で非常に大きなインパクトがあると考えられます。また、SLAMなど自己姿勢推定／空間復元の技術はすでにロボットなどで広く利用されていますが、本書で

取り上げた技術はそれらをより高精度に難しい環境下で高速に処理できるような手法です。メタバースやデジタルツインのように実世界と仮想世界の境界が徐々に小さくなる中で、これらの技術はそれらをつなぎ合わせる糊のような役割を果たし重要になると考えられます。

また強化学習による最適制御も注目されながらもまだ実世界での大きなインパクトが生み出されていない分野の一つです。強化学習の現実世界での実用化における重要な問題として大量の試行錯誤が必要なことが挙げられます。本書で紹介したような世界モデル、シミュレーションとの融合が進むことにより、より広い分野で活用が広がると考えられます。

AIの技術発展には3つ特徴があると考えてます。スピード、ボーダーレス、創造性という点です。

1つ目のスピードについては、今のAI技術は論文やコードが瞬時に世界中に共有され、SNSですぐ議論が始まり、YouTubeなどで解説動画が登場します。例えば大きなインパクトがある研究が登場した場合は、またたく間にコミュニティに広がり1カ月後には後続研究が登場し始め、半年や一年後の学会にはそれを利用した手法の論文が登場します。一番最初に手法を提案した論文と、その後続研究、さらにその後続研究が同じ学会の同じセッションで発表されることも稀ではありません。こうしたスピード感も本書で伝えられたらと思います。

2つ目のボーダーレスについては、今のAIは特定分野向けの手法ということがなく一つの分野で成功した手法がすぐに他の分野に広がっていくことです。例えばディープラーニング自身は一番最初は音声認識の分野で成功しました。大量のデータを使って学習すれば様々な工夫をすることもなく学習が成功し、高い性能を発揮できる。この結果を画像認識にも使えないかということで2012年のAlexNetが登場しました。この研究成果は時間差はあれその後、自然言語処理や化合物など他の分野に導入されていきました。同様に自然言語処理（機械翻訳）の分野で最初に登場したTransformerも、その後画像認識など他の分野にまたたく間に広がっていきました。このような手法のボーダーレスは自然界に見られる何らかの共通する法則性があるのではないかと考えさせられます。

3つ目は、AI技術の発展でインパクトのある成果は常識や従来の知識からは外れたところから登場するというものでした。優れた直感と実験能力に基づき驚くような結果が出て、しばらく経ってからそれを説明できるような理論が登場する流れが続いています。こうした結果を見るたびに、技術発展を妨げている最大の要因は創造性の欠如であり、常識に

とらわれない柔軟な発想が必要なのだと思わされます。本書を通じてこうした柔軟な発想も見ていただけたらと思います。

本書や連載記事を執筆するにあたり、編集者の進藤さんにはいつもお世話になりました。毎回技術内容の誤りや不明瞭な点についても鋭く指摘をしていただけました。また連載記事は筆者の勤務先であるPreferred Networksの同僚の方々にもチェックしていただき間違いやコメントをフィードバックしていただきました。

一方、本書中に間違いなどがありましたら筆者の責任です。書いた当時の雰囲気を残すため内容についてはできるだけ連載時の表現を残すようにしています。

本書を通じてAI技術の最前線で何が起きているのかを知っていただき、興味を持っていただければ幸いです。

2022年6月　岡野原 大輔

目次

1

知能とは何か、ディープラーニングとは何か

第1章

原理解明に向けた動き

1-1 なぜディープラーニングがうまく学習できるのか

ディープラーニングが画像認識、音声認識、強化学習などさまざまな分野で大きな成果を上げてはいるが、なぜディープラーニングがこれほどうまくいくのかについては、実はまだよく分かっていない。確かにディープラーニングは従来のモデルに比べてパラメータ数が多く、強力であり、あらゆる関数を近似できる能力を持っているが、それだけでは説明できない。

最適化の問題についてはノーフリーランチ定理というのが知られている。これは機械学習の学習問題についてもあてはまり、その用語で言い直すと「あらゆる問題で性能の良い機械学習モデルは理論上不可能であり、あるモデルが他のモデルより性能が良いのは、解こうとしている特定の問題に対して専門化または特化されている場合のみである」ことを意味する。

ディープラーニングも含めた機械学習手法は、学習データからパラメータそして関数を獲得する。一方で、学習データ以外に最初からもっている知識または仮説を、帰納バイアスと呼ぶ。先程のノーフリーランチ定理と組み合わせていうと、ディープラーニングは、何らかの帰納バイアスを持った上で特定の問題に特化することで他の手法と比べて優れた性能を達成しているといえる。

現実世界の問題に特化

それではディープラーニングはどのような問題に特化しているといえるのだろうか。この問題に対して、論文発表当時20歳であった若い物理学者Henry W. Lin氏が次のような仮説を立てている[1]。世の中にみられる実用上興味のある問題は次のような特徴を持っており、ディープラーニングはそうした問題に特化しているので成功しているのではないかというものである。

(1) 低次の多項式

世の中の一見複雑に見える問題の多くは低い次数の多項式モデルで説明することができる。例えば、重力を支配するニュートン方程式、電磁気学を支配するマクスウェル方程式、流体力学を支配するナビエ・ストークス方程式などの最大次数はたかだか4である。また、画像において意味を変えないような回転や平行移動などの変換は線形変換であり、次数を増やすことはない。

ニューラルネットワークは低次の多項式を近似することが得意である。例えば、4つのニューロンからなるニューラルネットワークで乗算1つをシミュレートすることができ、任意の多項式はその計算に必要な乗算回数の4倍程度の数のニューロンからなるニューラルネットワークで近似することができる。

(2) 局所性

世の中の多くの現象として、近くの物体同士しか影響しないという局所性がみられる。N個の物体が存在するシステムを考えた場合、本当は物体同士の相互作用はNの多項式個の関係で表されるが、実際には近くの定数個の物体同士にしか影響を及ぼさないため、Nに対して線形にしか複雑度は上がらない。

実際、局所性を持つマルコフネットワークはノード数に比例する数のニューロンからなるニューラルネットワークを使って近似することができる。

(3) 対称性

世の中の問題には対称性が多くみられ、これも見かけ上の複雑さを下げることに大きく役に立つ。代表的な対称性として、時間同変性や平行移動同変性、置換同変性などがある。ある変換が同変性を持つとは変換の入力を操作して別の入力にした時、出力もその操作に応じて規則的に変わることを意味する。出力が入力操作に対して変わらない不変性は同変性の特殊例である。移動同変性や時間同変性、置換同変性を明示的にモデルに組み込んだ畳み込みニューラルネットワーク（CNN）やRNN、Transformerは、学習に必要なパラメータ数を大きく減らすことができる。

これらの特徴は一見は複雑にみえるような世の中の問題

が、実は単純で現実的なサイズのニューラルネットワークで十分近似できるということを意味する。

階層性をうまくモデル化

それではニューラルネットワークの階層的な構造はどのような役割を果たしているのだろうか。

世の中で観察されるデータの生成過程にはマルコフ性、つまり直前の状態のみに依存して次のデータが生成されることがみられる。ある画像が観察されるに当たっては、物体の位置や形状、光源との位置関係、カメラとの位置関係が決まれば、最終的に観察される画像が決定される。これらの過程はマルコフ的であり、逐次的に変換され複雑なデータが生成される。

データの生成過程がマルコフ性を持っているとすれば、ニューラルネットワークはそれを逆向きに辿ることで、そのデータ生成の因子を推定することができると考えられている[2]。

また、あるニューラルネットワークを使って入力から出力が生成されたデータを、他のニューラルネットワークを使って近似した場合、データを生成しているニューラルネットワークの各ノードの挙動を真似られるように学習できることがわかっている[3]。世の中のデータ生成がニューラルネットワークと同様であれば、その生成過程自体を再現していることになる。

また、世の中の現象は階層性がみられる。こうした階層性を持った現象は多層のニューラルネットワークでうまく近似することができる。人の顔というものが、目や鼻といった部位の集合から成り立ち、目や鼻が画像上はエッジやコントラストで表現されているといった階層性はニューラルネットワークの各層でうまくモデル化できることがわかっている。

一方で、人や動物が非常に少ない経験から学習できることは、まだ見つかっていない帰納バイアスがあることを示唆している。人や動物はこうした帰納バイアスを進化の過程で脳の構造として獲得してきたが、そこに工学的に参考になる部分はまだ多いにあると考えられる。

1) H. Lin et al.,"Why does deep and cheap learning work so well?,"https://arxiv.org/abs/1608.08225
2) A. Patel et al.,"A Probabilistic Framework for Deep Learning," https://arxiv.org/abs/1612.01936
3) Y. Tian et al.,"Luck Matters: Understanding Training Dynamics of Deep ReLU Networks," https://arxiv.org/abs/1905.13405

（2017年2月号掲載の記事に加筆）

1-2

多様体仮説：現実世界のデータをどうモデル化するか

画像や音声など対象のデータを高次元空間中の1点と考え、データが空間中でどのように分布しているのかを調べることで、データをモデル化することを考える。しかし、一般的に高次元データに対して様々な統計的モデルを推定する場合、「次元の呪い」として知られる問題に遭遇し、推定は困難となる。

例えば、1000次元空間中に分布するデータをモデル化する場合、各次元を正か負かの二値で分けても、2の1000乗個の空間に分けられ、これは宇宙に存在する原子数よりも多く、現実的に観測可能なサンプルから推定はできない。

一方、画像や音声など自然界にみられるデータの多くは、可能な値の組み合わせのうちのほんの一部しか取らない。これらのデータはデータの見かけ上の次元数よりずっと少ないパラメータによって支配される空間に分布していると考えられている。

例えば、画像の各画素値を一様分布からサンプリングし、ランダムに生成した場合、そのほとんど全ての画像が砂嵐のような意味のない画像となる。一方、実際に観察される画像は人の顔画像や風景画像のような構造化されたデータであり、画素間が強い相関を持った画像である。しかも、これらの画像は、顔の傾きを少しずつ変えたり、照明を少しずつ変えたりしていくのに従って滑らかに少しずつ変わっていく。このように現実世界で観察される画像は、画素数よりもずっと少ない数のパラメータ（対象物体の角度、光源の位置、カメラの位置）によって支配されており、またそれらのパラメータを滑らかに変えると画像も滑らかに変わる性質を持つ。このようなデータが少ない数のパラメータによって支配されており、それらのパラメータを変えるとデータも滑らかに変わるという性質は、他の多くのデータでもみられる。これは多様体という概念で説明できる。

各点の周りがn次元的に拡がっているような空間を多様体と呼ぶ。別の言い方をすれば、局所的にはn次元の座標系を使って表すことができ、それがn次元ユークリッド空間と同相（滑らかに変形させていって同じ形にできる）であるような空間である。

現実世界のデータ分布は多くが低次元多様体

例えば、n次元ユークリッド空間自身はどの点でもn次元の拡がりを持った多様体である。また、球面というのは3次元空間中の2次元的な広がりを持つ多様体である。球面全体は2次元と同相ではないが（例えば世界地図では北極、南極で非連続になっている）、各点の周りだけをみれば2次元とみなすことができる。このことは私達が地球上でほとんど2次元の平面上にいるように考えていることからもいえる（多様体の工学的な説明は参考文献[1]を参照）。

現実世界で観測される多くのデータの分布がこのような低次元多様体として捉えられるという仮説を多様体仮説と呼ぶ。宇宙空間で星や銀河が泡の膜のように非常に薄い領域に存在しているのと同様に、多くのデータも高次元空間中に低次元の薄い膜として分布していると考えられている。

GANやVAEは多様体を潜在表現として獲得

データ中に埋め込まれている多様体をどのように推定するか、そして、多様体中の座標と元の世界の座標をどのように対応づけるかについては様々な手法が検討されている。この対応付けを実現する手法として線形であれば主成分分析（PCA）が使えるが、非線形であればIsomap、LLE（locally linear embedding）、Laplacian Eigenmaps、SDE（semidefinite embedding）などが知られている。

近年提唱されたVAE（変分自己符号化器）やGAN（敵対的生成モデル）、正規化フローといった深層生成モデルは、データの低次元多様体をその潜在表現として獲得できることが分かっている。

VAEやGAN、正規化フローではランダムに生成したノイズベクトルzから、生成関数Genを用いて入力xをx=Gen(z)として生成する。Genに多層のニューラルネットワークを使うことで従来手法よりも複雑なデータのモデル化に成功している。ノイズの次元数がデータの次元数より小さく、Genがニューラルネットワークの場合、このxは低次元多様体となることが知られている[2]。

学習が成功した場合、ノイズベクトルを滑らかに動かすと（例えば$z=\alpha z_1+(1-\alpha)z_2$で$\alpha$を0から1へ変える）、生成される画像も滑らかに変わっていくことが分かる。例えば、生成としても低次元多様体への分解としても最も成功しているStyleGANは生成データを構成要素毎に分解することができ（この場合ノイズベクトルではなくスタイルベクトルという形でデータを潜在的に表現している）、ある要素を動かすと対象物体を回転させる、ある要素を動かすと形が滑らかに変わるといように分解できる[2,3]。

最尤推定は低次元多様体の表現に不向き

そもそも低次元多様体であるデータを生成モデル$p(x)$を使って推定する場合、従来の最尤推定などの推定手法が適していないことが分ってきている[4-5]。

2つの確率分布間の距離を測る代表的な手法としてKLダイバージェンスがある。

$$\mathrm{KL}(p\|q)=\int p(x)\log\frac{p(x)}{q(x)}dx$$

このKL$(p\|q)$は2つの分布pとqが一致しない場合はKL$(p\|q)>0$であり、一致する時のみKL$(p\|q)=0$となるような関数である。そのため、目標の確率分布をp、学習対象の確率分布をqとした時、KL$(p\|q)$を最小化するようなq、つまり

$$\begin{aligned}q_{\mathrm{ML}}&=\operatorname*{argmin}_q \mathrm{KL}(p\|q)\\&=\operatorname*{argmin}_q\int_x p(x)\log q(x)dx\\&=\operatorname*{argmin}_q\sum_{x_i\in D}\log q(x_i)\end{aligned}$$

を解くことでpと一致またはそれに近いようなqを探すことができる。ただし、$D=|x_i|$は目標の確率分布pからサンプリングされた学習データである。この推定は観測データの尤度を最大にするような分布を探すことから最尤推定と呼ばれる。

最尤推定は生成モデルの推定の中心となっている手法だが次の問題点がある。1つ目は

$$p(x)=0, q(x)>0$$

つまり、観測されないデータに対してモデルが0ではない確率を割りあてる場合、そのデータの最尤推定における値は$q(x)$の値によらず

$$p(x)\log q(x)=0$$

となってしまう。つまり、モデルが間違ったデータxを生成しても直接ペナルティが課されない。多様体仮説が成り立つデータについては多くのxについて$p(x)=0$であるため、これは問題になる。

さらに、モデルqも低次元多様体であり、多くのxで$q(x)=0$である。$p(x)>0$である領域（これをpのサポートと呼ぶ）と$q(x)>0$である領域が重なっている部分の体積はほぼ0となる[4]。例えば、3次元空間中における厚みが0である紙の体積は0であるがそれと同様に高次元空間中の低次元多様体の体積は0である。KLダイバージェンスはこの場合、不定となる。

これまでは、こうした問題を防ぐために観測データに対してもデータの最後にノイズが含まれるとして、どのxについても$p(x)>0$となるようにし、また学習対象モデルも全てのデータに0ではない確率を割り当てるようにしていた。

最尤推定の問題を回避したGAN

GANはこうした問題を回避し、低次元多様体をモデル化できることが分かっている。

GANの学習は最尤推定ではなくJensen-Shannonダイバージェンスの最小化をしていることが分かっている。Jensen-Shannonダイバージェンスは次のように定義される。

$$\mathrm{JS}(P\|Q)=\frac{1}{2}(\mathrm{KL}(P\|M)+\mathrm{KL}(Q\|M))$$

ただし、$M=(P+Q)/2$である。MはPとQのサポートの和集合をサポートとして持つため、Jensen-ShannonダイバージェンスはPやQが低次元多様体であっても値が存在するようになる。

さらに、GANの改良版であるWasserstein GAN（WGAN）[5]では、分布間の距離として、Wasserstein距離（Earth Moverとも呼ばれる）を使って学習する。

Wasserstein距離は次のように定義される。確率密度関数$p(x)$によって定められる確率分布を、xの位置に$p(x)$だけ砂が存在していると考える。2つの確率分布pとqの砂の総量はどちらも1（確率分布の定義より）であるから、pの各位置にある砂をそれぞれ輸送してqの分布を作ることができる。輸送コストを砂の量に輸送距離をかけたものとする。このとき、pの分布をqの分布に変えるためにそれぞれ輸送した時の輸送コストの総和の最少をWasserstein距離と定義する。

GANの学習と似た式が導出

Wasserstein距離を求めることは困難そうにみえるが、カ

ントロヴィチとルビンスタインの双対定理（Kantorovich-Rubinstein Duality）を使うと、Wasserstein距離の下限がGANの学習とよく似た式として導出される。低次元多様体同士に対してもWasserstein距離はコストとして正しく、学習の際にWasserstein距離を小さくすることで目標分布を精緻にモデル化することができるようになる。

非連続な情報を扱う場合には潜在変数（または多様体における座標）に離散的な変数を使う必要がある。また、人の姿勢のような構造的な多様体を考える場合は階層的な多様体を考える必要がある。

生成モデルの学習を通じて低次元多様体を捉える

近年では自己回帰モデル、正規化フローや拡散モデルといった生成モデルが登場し、高忠実なデータを生成できると共に、データの隠れた特徴や因子を推定できるようになってきた。こうした生成因子は様々なタスクに有効な特徴であり、表現学習として優れている。近年GPT-3をはじめとして様々な生成モデルを学習することにより、事前学習するアプローチが広がっており、それらはデータの隠れた構造を捉えるのに成功していると考えられる。

1）甘利、『情報幾何学の新展開』、サイエンス社.
2）T. Karras et al.,"A Style-Based Generator Architecture for Generative Adversarial Networks," CVPR 2019.
3）A. H. Bermano et al.,"State-of-the-Art in the Architecture, Methods and Applications of StyleGAN," https://arxiv.org/abs/2202.14020
4）M. Arjovsky et al.,"Towards Principled Methods for Training Generative Adversarial Networks," https://arxiv.org/abs/1701.04862
5）M. Arjovsky et al."Wasserstein GAN," https://arxiv.org/abs/1701.078

（2017年6月号掲載の記事に加筆）

なぜディープニューラルネットは汎化するのか

機械学習の目標は、有限の訓練データからルールや知識を獲得し、(同じ分布からサンプリングされる)訓練データには含まれないが、訓練データと同様の性質をもつ未知のデータに対してもうまく推論できるようなモデルを獲得することである。この能力を汎化能力と呼ぶ。

一般に学習は訓練データ$Z=\{z_i\}_{i=1}^{N}$が与えられた時、訓練データに対する損失関数$l(z,f)$の和を最小にするような関数fを求めることで実現する。

$$L_{e(f)}=\frac{1}{N}\sum_i l(z_i,f) \quad \text{(訓練誤差)}$$

この訓練事例に対する損失の和$L_{e(f)}$を訓練誤差(または経験誤差)と呼ぶ。一方、未知のサンプルに対する損失の期待値$L(f)$を汎化誤差と呼ぶ。

$$L(f)=E_z l(z,f) \quad \text{(汎化誤差)}$$

データがどのような分布から生成されているかは未知のため、汎化誤差も未知である。訓練データとは別に用意した正解の分かっている検証用データセットを使って汎化誤差を推定することは行われる。関数fは訓練誤差を最小にするように最適化されているため、一般に訓練誤差は汎化誤差よりも小さくなる。

訓練誤差が小さく汎化誤差が大きい場合が過学習

一般に関数の表現力を大きくすると訓練誤差を小さくできるが、汎化誤差が大きくなる。例えば、関数のパラメータ数を増やす、あるいは内部で扱う関数を線形ではなく非線形にすることによって関数の表現力を上げることができる。このような、モデルの表現力が高く訓練誤差は小さいが、汎化誤差が大きくなっている状態を過学習(オーバーフィット)と呼ぶ。これはモデルの表現力が強すぎるため、訓練データにのみたまたま成り立つような間違った仮説を見つけてしまったり、訓練データを丸暗記して未知のデータの分類に役に立つような普遍的なパターンやルールを獲得できなかったりしたため起こる。逆にモデルの表現力が足りない場合は、訓練誤差をそもそも小さくできない。これを未学習(アンダーフィット)と呼ぶ。

訓練誤差と汎化誤差はトレードオフ

訓練誤差と汎化誤差はトレードオフの関係にあり、学習の際には、訓練誤差と汎化誤差の和が最小になるような適度な表現力を持つ関数を探す必要がある。汎化誤差を減らす場合にはモデルの表現力を落としたり、モデルの表現力が強いとペナルティを与えるような正則化を学習時に適用したりして学習する必要がある。

例えば、パラメータの大きさ(ノルム)に制限やペナルティを与えたり、入力や途中の計算結果にノイズを加えて学習したりなどである。

現在のDNN(ディープニューラルネットワーク)は多くのパラメータを持ち、内部の各ニューロンが非線形関数を表現できるため、どんなに複雑なデータでもフィッティングすることができる非常に表現力の高いモデルである。

DNNの表現力を調べるために実験

Zhang[1]らはDNNの表現力を調べるために次のような実験を行った。訓練データのラベルを全部シャッフルして並び替え、データとラベルの間に意味のある関係がないようにした上で、データからラベルを予測できるようにDNNを学習させた。

このような汎化するようなパターンがない場合でもDNNは、全てのでたらめな訓練事例を正しく分類するように学習できることが分かった。つまり、DNNは汎化に役立つようなパターンを抽出せず、訓練事例を1つひとつ丸暗記し、この訓練事例が来たらこのラベルを返すという関数を盲目的に覚えたことを意味する。

その後の実験では、同じネットワークと学習方法を使ってデータとラベルの間に意味がある場合には、DNNは丸暗記ではなく汎化するようなパターンを学習していることが分かった[2]。DNNは同じネットワークと学習方法を使って可能な場

合は汎化するように学習し、汎化できない場合は丸暗記するような仕組みが備わっていたのである。

なぜ表現力が高いのに汎化誤差が低いのか

DNNは表現力が高いため、容易に過学習するはずである。しかし、DNNは他の機械学習手法と比べても高い汎化能力を持っていることが分かっている。こうした結果は学習時に正則化手法を適用しなくてもみられる現象であり、これまでの機械学習の常識とは一致しない。なぜDNNはこれほど表現力が高いのに、汎化できるのだろうか。

近年、DNNの汎化能力の獲得にはパラメータの最適化で使っているSGD（確率的勾配降下法）が大きな役割を果たしていることが分かってきた。単なる目的関数の最小値を見つける最適化手法が実は正則化の役割を果たし、訓練誤差を下げるだけでなく汎化誤差を下げることにも役立っていたのである。

SGDは今のパラメータを、その勾配の（逆）方向に少しだけ更新することで最適化を行う。さらに、訓練データ全体から勾配を求めず、訓練データからサンプリングした上で勾配の推定量を求めるため、勾配にはノイズが含まれている。

SGDは暗黙的な正則化効果がある

まず分かったこととしては、SGDによって見つかる解はそのパラメータのノルムが最小となるような解であること、である[1-3]。ノルムが小さいというのはモデルの表現力を抑える強い正則化であり、また分類面から各訓練事例までの距離を最大化するマージン最大化とつながりがある。マージンと汎化誤差には直接関係があることが分かっている。

次に、SGDにより見つかる解はベイズ推定に対応することが分かっている[4]。分類をする際に、訓練誤差を小さくするようなパラメータを1つだけ選択して分類するのではなく、訓練誤差を小さくするようなパラメータを無数に選び、それらをその訓練誤差の小ささに応じて重みをつけた上で多数決をして最終的な判断をしているとみなせる。このように複数のモデルで多数決を取ると、たまたま1つのモデルが見つけた誤った仮説を引く可能性は低くなり、過学習を大きく抑えることができる。

SGDはFlat Minimaを見つけることができる

そして最も汎化能力の獲得に関係しているのが次のFlat Manimaである。SGDは最適化の際、サンプリングの選び方によって勾配にノイズが乗るが、これが局所最適解からの脱出に役立っていただけではなく、その周辺も値が小さいようなFlat Minimaを見つけるのに貢献していることが分かった。Flat MinimaはMDL（最小記述長原理）やPAC-Bayesとも関係し、より単純な表現力の低いモデルを見つけることに相当する[5]。

また、SGDでの学習を加速させるようなADAMやRMS Propといった目的関数の曲率を利用した手法は汎化性能を大きく落とすということも分かってきた[6]。こうした手法を使うと特定の訓練データに特化し、過学習してしまうためである。

汎化能力についてはまだ分かっていないことが多くあり理論と実践の間に大きなギャップが存在する。今後、汎化能力の理論解明が必要とされている。

1) C. Zhang et al.,"Understanding deep learning requires rethinking generalization," ICLR 2017.
2) D. Arpit et al.,"A Closer Look at Memorization in Deep Networks," ICML 2017.
3) S. Gunasekar et al.,"Implicit Regularization in Matrix Factorization," NIPS 2017.
4) S. Mandt, "Stochastic Gradient Descent as Approximate Bayesian Inference," JMLR 18, pp.1-35, 2017.
5) S. Hochreiter et al., "Flat Minima," Neural Computation vol.9, no.1, pp.1-42, 1997.
6) A. Wilson et al.,"The Marginal Value of Adaptive Gradient Methods in Machine Learning," NIPS 2017.

（2017年8月号掲載の記事に加筆）

独立成分分析：情報のもつれを解く

教師なし学習の目的は大きく4つある。1つ目は観測データの分布を獲得すること、2つ目は観測データと同じようなデータをサンプリングできるようにすること、3つ目は教師あり学習などのタスクに有効な特徴を獲得すること、そして4つ目はデータの隠れた構造を明らかにすることだ。

例えば、1つ目は確率モデルの学習、2つ目は生成モデルによるサンプリング、3つ目は自己回帰や対比学習を使った事前学習などが対応する。今回はこの4つ目に重要な独立成分分析（ICA：Independent Component Analysis）を紹介する。

データを生成する因子に分解

世の中で観察される多くのデータは複数の因子（または情報源）が複雑に絡まり合って生成されている。例えば、街中で取られた音響データにはすぐ近くの人たちの話し声や遠くの電車の音、木で葉っぱが擦れ合う音などが含まれている。また、車載カメラの映像データには、道路上の複数の車やバイク、道路や背景の木、空の雲などが映りこんでいる。こうしたデータをそれぞれの因子に分解できれば様々な問題を簡単に解くことができる。この問題は「もつれを解く（Disentanglement）」として表現学習の重要なテーマとなっている。

必ずしも因子を抽出できないPCA

例えば、広く使われる主成分分析（PCA：Principal Component Analysis）は、n次元からなる観測データを2乗誤差の意味で最もうまく近似できる$k<n$個の基底を探し、それらを使ってデータを表現する。別の言い方をすれば観測データを分散が最も大きく、互いに直交している方向を大きい順からk個取り出してそれらでデータを表現する。PCAによって得られた基底や各データがどの基底を使って構成されているか（主成分）は、データの傾向を捉えられる。一方でPCAで獲得できる基底は必ずしも因子に対応するわけでなく、因子間で分散の傾向に違いがなければ抽出することはできない。

一方、独立成分分析（ICA：Independent Component Analysis）は因子がお互い独立であるという仮定、また多くの場合、因子の確率分布が非ガウシアンであるという仮定から、データをその生成因子に分解するタスクだ。

例えば、複数の人が話している状況でその音声をマイクで取る場合を考える。マイクがそれぞれの人の声をどの程度の強さで拾えているのか、それぞれの人がどのような音声を発しているのかは分からないが、それぞれの人の声はお互い影響せず独立だと考えることができる。このとき、音響データのみからそれぞれの人の声を分離するタスクがICAだ。

ICAの枠組み

ICAを式で見てみる。m次元からなるn個の観測データは、各データを各行に並べたn行m列からなる行列Xで与えられる。このとき、ICAは

$$X=f(S)$$

を満たすような因子（情報源）$S\in\mathbb{R}^{m\times k}$と、その混合関数$f:\mathbb{R}^k\to\mathbb{R}^m$を求める問題とみなすことができる。ここでは因子は$k$個あるとし、各次元が独立な$k$次元ベクトルであるとしている。特に混合関数が線形である場合、これは

$$X=AS$$

のように混合行列Aを使って表すことができる。これまで、混合関数が線形の場合はデータXのみから混合行列Aと独立成分Sを回転などを除いて同定できることが分かっている。また、大規模なデータであっても高速に求められるアルゴリズムが多く提案されている。特に、各データにおける独立成分が疎（多くの成分値が0）の時、データを白色化した後にk-means法を用いて得られた各クラスターはICAのフィルタ（混合行列の逆行列）になることが示されている[1]。

一方、現実世界のほとんどのデータでは、混合関数は複数回の非線形な変換を経た後に観測データが得られる。しかし、混合関数が非線形の場合、観測データXを生み出すような無数の独立成分と混合関数の組み合わせが存在してしま

い、混合関数と独立成分は不定となる。

時系列データに対する非線形ICA

これに対し、英University College LondonのHyvarinen氏とフィンランドUniversity of Helsinkiの森岡氏らは観測データが時系列データであり一定の条件を満たす場合、非線形独立成分分析が同定可能であり、それが単純なアルゴリズムで実現できることを示した[2-3]。以下ではこれらについて説明する。彼らは独立の情報源が非定常の場合と定常のそれぞれで提案している。

情報源が非定常の場合

始めに非定常の場合を紹介する[2]。非定常といってもデータは短い時間内では定常とみなせ、長い時間では非定常であるようなゆっくりと分布が変わっている場合を考える。そして、各独立成分がガウシアンも含む指数分布族に従って生成されている場合を考える。この時、独立成分は次のステップで同定することができる[2]。

(1) 時系列データを一定時間ごとのセグメントに分割し、それらのセグメントについて最初から$1, 2, 3, \ldots, k$と順に番号をつける。

(2) 各時刻のデータを、それが所属するセグメントのラベルがついた教師ありデータとみなす。

(3) 各時刻のデータが与えられた時、それがどのセグメントから来たのかを分類するkクラス分類問題をニューラルネットワークNN(x)で学習する。ニューラルネットワークの最後の層はソフトマックスを使ってkクラスの確率分布を出すようにする。

(4) NN(x)がセグメントを完全に分類できるようになった時、NN(x)のソフトマックス直前の層$h(x)$が独立成分の線形変換後と一致する。そのため$h(x)$に対して線形の独立成分分析を用いて独立成分、混合行列を同定する。

このアルゴリズムについて直感的な意味を説明する。データの分布はゆっくりと変わっているので、異なるセグメント間では独立成分の生成分布が異なっている。最終層にソフトマックスを使っているので、このモデルによる予測分布$q(y|x)$はソフトマックス直前の層を$h(x)$とした時、

$$q(y|x)=\frac{\exp(<w_y,h(x)>+b_y)}{1+\sum_{y'=2}^{k}\exp(<w_{y'},h(x)>+b_{y'})} \qquad (*)$$

と表される。ただし、一般性を失わずに$y=1$の時、

$$q(1|x)=\frac{1}{1+\sum_{y'=2}^{k}\exp(<w_{y'},h(x)>+b_{y'})} \qquad (**)$$

としている。分母もそれに応じて$1+\sum_{y'=2}^{k}\exp(<w_{y'},h(x)>+b_{y'})$となっている。一方、真の事後確率$p(y|x)$はベイズの定理より

$$p(y|x)=\frac{p(x|y)\,p(y)}{\sum_{y'}p(x|y')p(y')} \qquad (***)$$

となる。完全に分類できるようになったということは(*)と(***)の確率が一致し、

$$\log q(y|x)=\log p(y|x)$$

が成り立つ。両辺から$\log q(1|x)$と$\log p(1|x)$をそれぞれ引き、(*) (**) (***)を組み合わせ整理すると

$$<w_y,h(x)>+b_y=\log p(x|y)-\log p(x|1)$$

となる。

ここで、セグメントは全て同じ大きさなので$p(y)=p(y')$も使っている。(*)の分母は、尤度比を扱うことでうまく消去されている。右辺は独立成分に非線形な混合関数を施した後の確率分布であり、本来は混合関数のヤコビアンの行列式の項ができて厄介なのだが、これも2つの対数尤度の差分(尤度比)から消去でき右辺も独立成分を線形変換した形が出て来る。

この式が全てのyについて成り立つのでそれらを並べて整理すると左辺が$h(x)$の(逆行列が存在するような)線形変換であり、右辺が独立成分sの線形変換の式が出て来る。よって、$h(x)$が独立成分の線形変換後の成分となるので線形のICAを使って独立成分を同定することができる。このアルゴリズムは異なる時間の尤度を比較することからTCL (Time Contrastive Learning)という名がついている。

情報源が定常の場合

次に各情報源が定常の場合を紹介する[3]。この場合、仮定として隣接する時刻間$s(t)$、$s(t-1)$が一様従属である場合

を考える。一様従属とは、$p(s(t), s(t-1))$の対数尤度が$s(t)$と$s(t-1)$について

$$\frac{\partial^2 \log p(s(t), s(t-1))}{\partial s(t) \partial s(t-1)} \neq 0 \text{ for all } (s(t), s(t-1))$$

が成り立つ場合をいう。これは$s(t)$と$s(t-1)$が独立ではないことよりも強い条件だ。このとき、次のアルゴリズムはデータの独立成分を同定する。

(1) 隣りあう時刻のデータ $(u, v) = (x(t), x(t-1))$ を正例（$y=1$）、異なるランダムな2箇所からとったデータ $(u, v) = (x(t), x(t'))$ $(t' \neq t-1)$ を負例（$y=-1$）とする、
(2) $r(u,v) = \sum_i B_i(h_i(u), h_i(v))$ とし、$q(y|u,v) = \frac{1}{1+\exp(-r(u,v))}$ としたロジスティック回帰モデルを学習する。ただし、h_i, B_iはニューラルネットワークを使う。
(3) 訓練事例を完全に分類できるようになった時、$h_i(x)$ を並べて得られる $h(x)$ は成分ごとの単調な変換、添字の並び替えを除き独立成分と一致する。

このアルゴリズムは連続する2つのデータの場合と、ランダムに入れ替えた2つのデータを比較することからPCL（Permutation Contrastive Learning）と名前が付けられている。

これについても直感的には、ロジスティック回帰が十分、分類できるようになった時、

$$r(u,v) = \log p(1|u,v) - \log p(0|u,v)$$

となり、尤度比（対数尤度の差）が得られることから説明できる。

これらの成果は、非線形な独立成分分析が教師なし同定可能であり、学習が容易な分類タスクによって実現できることを示した点で画期的だ。

実用上は課題も

一方、実用的にはいくつか課題が残る。1つはこれらのアルゴリズムは与えられたデータを分類器が完璧に分類できるようになることを仮定している。学習は有限のデータで行われ、ニューラルネットワークの表現力の限界もあることから、実際には達成することは困難だ。

また、実際の観察データにはノイズが多く含まれており、情報源も定常、非定常が混ざり合っている。そのような場合にどこまで正確に独立成分分析が実現できるかは未解決である。

今後理論解明が進み、動画のようなデータにもICAが適用できることが期待されている。

1) A. Vinnikov et al.,"K-means Recovers ICA Filters when Independent Components are Sparse," ICML 2014.
2) A. Hyvarinen et al.,"Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA," NIPS 2016
3) A. Hyvarinen et al.,"Nonlinear ICA of Temporally Dependent Stationary Sources," AISTATS 2017.

（2017年9月号掲載の記事に加筆）

1-5

ディープラーニングの理論解析：ニューラルネットの未解決問題の解明へ大きく前進

ディープラーニングは画像認識や音声認識など多くのアプリケーションで従来手法を大きく上回る性能を達成しているが、なぜうまくいくのかについては実は理論的な説明ができていない。特に最も基本的で重要な

(1) なぜディープラーニングは学習できるのか

(2) なぜディープラーニングは汎化するのか

については完全な解明ができていない。これらの問題とそれらの解決を目指した近年のアプローチについて紹介しよう。

ディープラーニングはなぜ学習できるのか

はじめに、ディープラーニングがどのように学習をしているのかについて簡単におさらいしよう。ディープラーニングは多くの機械学習と同様に、学習データを使ってモデルのパラメータ θ に基づいた目的関数 $L(\theta)$ を設定し、この目的関数をモデルパラメータについて最小化する最適化問題 $\min_\theta L(\theta)$ を解くことで学習する。学習の際には、目的関数のパラメータについての勾配 $v = \frac{\partial L(\theta)}{\partial \theta}$ を計算し、勾配の負の方向に向かってパラメータを $\theta := \theta - \alpha v$ のように更新する（確率的）勾配降下法（SGD）を利用する。ここで $\alpha > 0$ は学習率と呼ばれるハイパーパラメータである。この最適化は、あたかも平面方向を θ、高さを $L(\theta)$ とした波打った曲面上で最も急に下る方向に移動してくようなものである。この最適化の何が問題だろうか。

ディープラーニングの目的関数は非凸関数であり、無数の局所解やプラトー（勾配が 0 に近い平らな領域）、鞍点（馬の鞍のようにある方向には下がっていて、ある方向には上がっている）が存在することが分かっている。そのため、勾配降下法はプラトーや鞍点に遭遇すると勾配が消失し学習できない状態に陥ったり、たとえ局所的に一番小さい領域に到達したとしてもそれが全体の中で最適な解である保証がない。

実際、潜在変数モデル（HMMなど）に対する期待値最大化（EM）法を使ったパラメータの最適化やK-means法によるクラスタリングなどでは、目的関数が非凸で多くの局所解を持つため初期値によっては最適ではない解に収束する。そのため、2012年より以前は、大きなニューラルネットワーク（NN）の学習は不可能なくらい難しいのではないかと思われていた。しかし、2012年にカナダUniversity of TorontoのAlex Graves氏らはいわゆるAlexNetで大きなNNでもなぜか学習に成功することを示し、その後もほとんどのNNの学習では初期値によらず、目的関数を最小化（多くの場合訓練誤差を 0 ）するような解に到達できることが分かっている。

この謎を解くためこれまで多くの理論解析がされてきた。例えば目的関数がどのような形をしているのか（Loss Surface、Loss Landscape）の解析がなされてきた。NNの内部計算で入力から出力までの計算パスが独立であるという近似をした上でスピングラスモデルを適用して解析した例では、局所解が最適解の近くに集まっていることが示されている[1]。しかしこれまでのモデルは、データに仮定を置いたり3層のNNに対しての解析などのみで、実際複雑な学習データで数十層を超えるNNの学習に対する説明ができていなかった。

2018年11月にMicrosoft ResearchのZeyuan Allen-Zhu氏らがこの問題に対し、過剰パラメータを持つNNであれば、SGDを使った学習は最適解に多項式時間で到達できることを示したと主張した[2]（ほぼ同時期に別の2チームも同様の主張をしたが文献[2]が最も一般的で強い主張をしている）。過剰パラメータは学習サンプル数よりパラメータ数の方が多いような場合であり、実際多くのNNの学習では過剰パラメータを使っている。この証明では多層かつ現実的なネットワーク（ReLUを使っており、CNN、ResNetを含む）を扱っており、この問題解決への大きなマイルストーンとなった。

実験的にはNNはパラメータ数が大きければ大きいほど学習しやすいことが分かっている（さらに目的関数の最適化と関係ないが汎化性能も良くなることが分かっている）。通常、制約の数よりパラメータの数の方が大きければ不定問題となり、収束が速くなる場合が多いがNNの場合はそうした問題には遭遇しない。

この証明には高度なテクニックを使っているため、ここでは概要のみを説明する。証明のポイントとなるのは次の2点である。

1つ目はパラメータがランダムに設定された初期値に十分

近ければパラメータについての勾配は小さすぎず、大きすぎないということである。さらに目的関数の値が大きい場合は勾配も大きいということを示せる。これによって途中でプラトーや局所解に遭遇する可能性がほとんどないことが示される。この証明の中では、たとえ学習中に1つのサンプルについて勾配が消失した場合でも他のサンプルが勾配を生み出すことも利用している。

2つ目は、ReLUなど滑らかでない関数を使った場合でも目的関数はある種の滑らかであり、局所的な減少方向である勾配に従って最適化することで目的関数を小さくできるということを示せる。

これらの結果を利用することでNNはどのような初期値であろうが学習データであろうが、過剰パラメータさえ持てば（ほとんど）常に学習が成功できるということが示せる。

ディープラーニングはなぜ汎化するのか

もう1つの問題が、ディープラーニングがなぜ汎化するのかである。機械学習の目標は、学習データに対してうまく振る舞うことではなく、未知データに対してもうまく振る舞えるような汎化能力を獲得することである。

一般的にパラメータ数が多く、強力、つまり多くの関数を表現できるようなモデルはそうでないモデルに対して汎化しにくいことが知られている。例えば、汎化性能の理論でunion boundやラデマッハー複雑度、VC次元などを用いてモデルの複雑度を数値化し、これらを用いて汎化性能を評価することができる。また、「ある事柄を説明する場合に必要以上に多くの仮定を使うべきではない」という“オッカムの剃刀”の原則は、モデルサイズを必要最低限に抑えることを支持していた。

これに対しニューラルネットワークは強力でありながら汎化することは謎であった（これについては本誌2017年5月号でも取り上げている）。実際、学習データのラベルをランダムなラベルに置き換えたデータに対してもディープラーニングモデルは“学習”することができ、高い表現力を持っていることが示されている[3]。

この汎化能力を説明するために、モデルの複雑度だけではなく学習アルゴリズムに依存して汎化性能を説明するPAC-Bayes、一様安定性などを使った手法が提案されている。ここでは学習データとアルゴリズムの関係を考慮した相互情報量に基づいた汎化性能の評価[4]についてみてみよう。

はじめに必要な定義を説明する。確率変数Uは$\log \mathbb{E}[\exp(\lambda(U-\mathbb{E}U))] \leq \lambda^2\sigma^2$を満たす時、$\sigma$-subgaussianであると呼ぶ。これは裾に行くに従って確率密度がガウス分布より速く減衰していくような分布である。

未知の分布μに従って学習データが$S=\{Z_1, Z_2, \ldots, Z_n\}$とサンプリングされたとする。また学習アルゴリズムはSを使って（確率的に）モデルwを$P_{W|S}$に従って選択すると考える。また損失関数を$l(z,w)$とし、SとWの同時確率についてσ-subgaussianであるとする。例えば下限がa、上限がbであるような損失関数は$(b-a)/2$-subgaussianである。あるモデルwの汎化誤差L_μと訓練誤差L_Sをそれぞれ

$$L_\mu(w) = \mathbb{E}_Z[l(w,Z)]$$

$$L_S(w) = \frac{1}{n}\sum_i l(w,Z_i)$$

と定義する。一般にアルゴリズムは訓練誤差に対してモデルを最適化するため、$L_\mu(W) > L_S(W)$であり、この差が小さいと汎化に成功していることになる。この差のSとWの同時確率についての期待値を

$$gen(\mu, P_{W|S}) = \mathbb{E}[L_\mu(W) - L_S(W)]$$

と表す。このとき、SとW間の相互情報量$I(S;W) = \int p(S,W)\log\frac{p(S,W)}{p(S)p(W)}dSdW$を用いて$gen$は次のように抑えられることが分かっている[4]。

$$gen(\mu, P_{W|S}) \leq \sqrt{\frac{2\sigma^2}{n} I(S;W)}$$

この式の意味を考えてみよう。相互情報量はWを知った時にSについてどれだけ情報が得られるかを表している。そのため、この式は学習アルゴリズムは学習データから情報を多く取りすぎない方が、汎化しやすいことを示している。学習データの詳細は忘れ、抽象的な知識のみ抽出すればこのgenを小さくすることができる。

この相互情報量を使った評価は、これまでの汎化理論のツールが扱うことが難しかったデータ分布やアルゴリズム（$P_{W|S}$）を自然に含めることができ、より強力な理論評価を行うことができる。一般にSGDは学習中にノイズを加えていくため$I(W;X)$がどこまでも増加してしまう問題があるが、文献[5]では、仮説空間でのサンプルの近さを利用した解析で

よりタイトな上限を与えている。

どこまで解けたか

なぜNNが学習できるかについては、文献[2]は現実的な条件でそれを証明できたといえる。ただ必要なパラメータ数が深さやサンプル数の多項式になっており、パラメータ数を現実的なサイズに落とした場合の証明や、(過学習しないように)訓練誤差を0にしない場合にどのような解が得られ、それに収束するのかは残された課題である。

なぜ汎化するかについては、まだ大きなギャップがあるといえる。他の多くの機械学習手法と同様、理論による汎化性能の評価はかなり悲観的な評価であり、実際の汎化性能のギャップはまだ大きい。このほかにもフラットな解が選択されることにより、実質的に単純な解が選ばれること、次の章で扱う宝くじ仮説により、パラメータ数が多くても実際にはその中で有効なサブネットワークが選ばれることが示される。一方でなぜ汎化するかは分からないが、どのくらい汎化しているかどうかはデータが$\mathrm{i.i.d.}$であれば交差検定などで正確に求めることができる[6]。今後もしばらくは実用的には汎化性能はコントロールしつつ、汎化する謎を解く試みが続くと思われる。

---

1) A. Choromanska, et al.,"The Loss Surfaces of Multilayer Networks," AISTATS 2015, https://arxiv.org/abs/1412.0233
2) Z. A-Zhu, et al.,"A Convergence Theory for Deep Learning via Over-Parameterization," https://arxiv.org/abs/1811.03962
3) C. Zhang, et al.,"Understanding deep learning requires rethinking generalization," ICLR 2017.
4) A. Xu, et. al.,"Information-theoretic analysis of generalization capability of learning algorithms," NIPS 2017.
5) A. R. Asadi, et al.,"Chaining Mutual Information and Tightening Generalization Bounds," NeuRIPS 2018.
6) K. Kawaguchi, et al.,"Generalization in Deep Learning," https://arxiv.org/abs/1710.05468

(2019年2月号掲載の記事に加筆)

過剰パラメータ表現のニューラルネットワークと宝くじ仮説

ニューラルネットワークは学習データ数と同程度かそれ以上に多くのパラメータを持ちながら、なぜ過学習せず汎化するかはこれまで未解決問題だった。このようなパラメータ数の方が学習データ数（制約数）より多い場合を過剰パラメータ（Over-parameterized）表現と呼ぶ。

一般にモデルはパラメータ数が必要な数より多く、表現力が強すぎる場合、ノイズ由来の訳ったパターンを学習してしまい、訓練データではうまくいくが、テストデータではうまくいかない過学習と呼ばれる現象を起こす。これを防ぐため、学習時にモデルの表現力を抑える正則化を適用することが一般的である。重みのノルムに罰則項を付けたり、使うパラメータ数を最小限にするなどだ（Boostingなど）。しかしニューラルネットワークは学習時にそのような正則化（例えばDropout、Weight Decayやノイズを導出するバッチ正規化）を明示的に適用しなくても過学習しにくいことが分かっている。

それではニューラルネットワーク（NN）の表現力がないのかというとそうではない。実際、学習データのラベルをランダムに変えて作った訓練データに対してニューラルネットワークを学習させても、訓練誤差を0にすることができる[1]。さらにネットワークの幅が大きくパラメータ数が多ければ多いほど学習が容易であり汎化性能が上がることが分かっている。一方で幅を無限大にして得られるNeural Tangent Kernelを使ったNNよりも通常の有限の幅のNNの方が精度が高いことも分かっている[2]。

DNNがなぜ汎化するかを説明する理論

この過剰パラメータ表現のニューラルネットワークがなぜ汎化するかを説明する理論として、米MITのJonathan Frankle氏らは"宝くじ仮説"（Lottery Ticket Hypothesis）[3]を提唱した。この論文はICLR 2019のベストペーパーに選ばれている。これについて説明する。

宝くじ仮説は「ランダムに初期化された密なニューラルネットワークは、うまく学習できるように偶然初期化されたサブネットワークを含んでおり、そのサブネットワークを取り出してそれだけを同じ初期値から学習したら元のニューラルネットワークと同じ精度に同じ学習回数で到達できる」という仮説である。ランダムに初期化された大きなニューラルネットワークは無数の異なる初期値の組み合わせを持つサブネットワーク（くじ）を含んでおり、そのほんの一部がうまく学習できるサブネットワーク（当たりくじ）であるという仮説である。うまく学習するにはネットワークの構造だけでなく、初期値も重要である。そして良い初期値の組み合わせをたまたま引き当てたサブネットワークが当たりくじとなり、それが他のサブネットワークよりも速く学習が進み他のサブネットワークの活性は抑制されていく。そして、ネットワークが大きくなるほど、サブネットワークの数は指数的に大きくなり、良い当たりくじが含まれる可能性が高くなる。そのため、大きなネットワークほどパラメータ数が多くなるにかかわらず汎化するというものである。

この宝くじ仮説を証明するために、彼らは次のような実験を行った。始めにニューラルネットを通常の方法で学習した後に、重みの絶対値が小さいp%（例えば80%）の枝を取り除く。次に、残ったネットワークを、その学習時に使った初期値に戻して再度学習し、精度を評価した。すると、元のネットワークの学習とほぼ同じ学習曲線を描き、近い精度を達成した。それに対し、同様にして枝を除いた後に、その学習に使った初期値とは別の初期値で初期化し再度学習した場合、そのネットワークの精度は大きく低下した。ここから、ネットワーク構造だけでなく、初期値の組み合わせが重要であること、またネットワーク全体の性能はその良い初期値を引いたサブネットワークと同じであることが示された。ニューラルネットワークは無数のサブネットワーク候補を同時に効率的に探索し、良いネットワークを見つけているといえる。また、ニューラルネットワークの学習というのは何かを作り出していくのではなくはじめから存在するうまくいくサブネットワークを削り出しているようなアプローチだとみなすことができる。

この宝くじ仮説は大きなニューラルネットワーク、例えばResNetなどでは成り立たないと最初は報告されていたが、

完全な初期値ではなく少し学習が進んだ値からスタートすると同じような現象が起こることが分かった。不要な枝を除いた際の変化が初期値の段階では大き過ぎるため学習が安定しないが、少し学習が進むと枝を除いても学習が安定するためだと考えられる[4]。

仮説を指示する理論も登場

この仮説は実験結果に基づいたものだけだったが、その仮説を支持する理論が登場している[5]。この理論の提唱者らはMLPからなる教師ネットワークと生徒ネットワークの2つを用意し、教師ネットワークの出力を正解として生徒ネットワークがそれを真似るように学習する問題設定を考えた。教師ネットワークは本来は未知であるデータの生成過程とみなすことができる。教師ネットワークと生徒ネットワークは必ずしも同じノード数、パラメータ数を持つとは限らない。

この学習ダイナミクスを解析すると、生徒ネットワークが教師ネットワークの出力を真似られるようになるにつれ、教師ネットワークの各ノードの振る舞いを生徒ネットワークの各ノードが真似るようになることがわかった。最終出力を一致させるようにするというフィードバックがあるだけで、隠れ層の各ノードが対応するようになるのである。ここでノードの振る舞いとは、どういった入力に対して活性値が0非ゼロになるかという領域のことである。さらに生徒ネットワークの方がノード数、パラメータ数が多い過剰パラメータ設定の場合、教師ネットワークのそれぞれのノードに初期値の時点で最も似ているノードがそのノードの振る舞いを真似るようになり、対応するノードが無い生徒ネットワークのノードの出力枝の重みは0に抑制されていくこともわかった。模倣するのが最もうまくいっているノードが、それよりうまくいっていないノードの活動を抑える、いわゆるWinner-Take-Allのような現象が起きていることもわかった。

実際のニューラルネットワークが学習する対象は必ずしも、他のニューラルネットワークの出力ではなく、自然現象であったり人の活動の結果だったりする。しかし、それらの学習対象のデータ生成過程がニューラルネットワークと同じような構成性や階層性を備えている場合は、その生成過程はニューラルネットワークで近似することができる。そして、その近似を最小限の大きさを持ったニューラルネットワークで模倣することができるため過学習せず汎化すると説明できる。

また、汎化するような解はフラットな解、つまり多くの方向にパラメータを動かしても訓練誤差がほぼ変わらないような解であると考えられている[6]。これも宝くじをひいて他のサブネットワークの出力が抑制されているような状況では多くのパラメータを変えても出力結果が変わらないような状況になっているためだと説明できる。

人の脳の発達過程では最初に多くのシナプスが発生した後にそれらが枝刈りされていくことが分かっている。これも宝くじ仮説と同様、それぞれの個体や環境に合った非常に運の良い初期値を引いたサブネットワークを探すプロセスなのかもしれない。学習時の効率の観点からいうと、最初から有望なサブネットワークを抜き出しそれだけを学習させたくなるが、それを実現できるか、最初から当たりくじを引けるのかは未解決問題である。

---

1) C. Zhang et. al.,"Understanding deep learning requires rethinking generalization," ICLR 2017.
2) S. Arora et al.,"On Exact Computation with an Infinitely Wide Neural Net," https://arxiv.org/pdf/1904.11955.pdf
3) J. Frankle et al.,"The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," ICLR 2019.
4) J. Frankle et. al.,"Stabilizing the Lottery Ticket Hypothesis," https://arxiv.org/pdf/1903.01611.pdf
5) Y. Tian et. al.,"Luck Matters: Understanding Training Dynamics of Deep ReLU Networks," https://arxiv.org/pdf/1905.13405.pdf
6) N. Keskar et al., "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima," ICLR 2017.

(2019年8月号掲載の記事に加筆)

因果と相関：未知の分布まで汎化できるか

機械学習の大きな目標は、訓練データと異なるテストデータでもうまく予測できるような汎化するモデルを獲得することである。訓練データだけうまくいくのでよければ訓練データを丸暗記すればよく、コンピュータは容易に実現できる。しかし、見たことがないテストデータでもうまく予測するためにはデータの背後に隠された法則を見つける必要がある。

一般に機械学習の問題設定では訓練データとテストデータは同じ分布から互いに独立にサンプルされているという、いわゆるiid（独立同分布）を仮定している。このiidの下では訓練データを十分な数集め、訓練データでうまくいくようなモデルさえ作ることができれば、テストデータでうまくいくことも期待できる。これに基づいて訓練データの誤りを最小化する経験誤差最小化（ERM：Empirical Risk Minimization）に基づく学習が、学習のデファクトスタンダードとなっている。モデルが各サンプルでうまく予測できているかを測る損失関数を用意し、訓練データ上で損失関数の平均値が小さくなるようなパラメータを探すというものだ。しかし、世の中の多くの問題ではiidが成り立たない。訓練データを選ぶ際に偏りがあったり、分布が変わるような共変量が存在するなどするためだ。このような場合、ERMを使った学習では誤った相関を学習してしまい訓練データとは異なる環境で性能が落ちる、ひどい場合は全くでたらめな結果となってしまうことが起きてしまう。

例えば、画像の中に写っているのが牛かラクダかを当てる問題を考えてみよう[1]。訓練用のデータでは、大抵の牛は牧草と一緒に写っており、ラクダは砂漠と一緒に写っている。そのため、牧草が写っていれば牛の可能性が高い、砂漠が写っていればラクダの可能性が高いと分類するモデルが得られるだろう。

それではたまたま海辺の砂浜に来ていた牛を撮った画像に対して、これは砂漠（のような砂浜）が写っているからラクダと判断するのが正しいだろうか。これはもちろん誤りである。分布が変わりうる環境ではこのようなことが起きうる。このようにiidではない問題設定においては、訓練データに普遍的にみられる相関を見つけたとしても、それはテストデータの環境でも常に成り立つ相関とは限らない。汎化するモデルは、未来にわたっても安定して成り立つような相関を見つける必要がある。先の例では牛かラクダかを当てるには牛やラクダの形状、見た目から推定するべきであり、背景から推定するのは誤りである。未来にわたって安定ではない相関（先程の例の背景、環境など）をいかに排除できるかが重要となる。そのような安定した相関は一般に分類結果に関係する因果関係に基づく。因果関係に基づいて分類できていれば結果に関係の無い環境部分が変わっても分類結果は変わらないことが期待される。ここまでの話をまとめると、iid時には相関を使ったモデルはうまく動くが、そうでない場合は未来にわたって安定した相関、一般には因果関係を見出さなければうまく動かない。

機械学習は相関を求める手法が多いが、因果関係を求められるような手法も多く提案されている。しかし集められたデータだけから、そのデータの生成過程の仮定なしに因果関係を求めることは一般に不可能であることが知られている（例えば文献[2]を参照）。それに対し生成過程について何かしらの事前知識、仮定があればデータから、どの変数間に因果関係があるかを推定することができる。また強化学習のような、環境に対して直接作用ができるような問題設定でも試行錯誤することで因果関係を求めることができる。例えば治験などでも用いられているような因果推論をする上で最も強力なランダム化比較試験をすることができる。しかし世の中の多くの問題では生成過程もわからなく、何回も自由に試行可能な環境も存在しない。そのため、既に取得したデータだけからでもこのような因果推論ができ汎化するモデルを作ることが求められていた。

こうした中、New York UniversityのMartin Arjovsky氏とFacebook（現Meta）AI ResearchのLeon Bottou氏らはERMに置き換わるような汎化する学習手法として不変誤差最小法（IRM：Invariant Risk Minimization）を提案した[1]。このIRMは安定的な相関を見つけることで、訓練データとは異なる分布に対しても汎化することを目標としている。キーとなるアイディアは様々な環境からの学習データを集めた後に、それらを混ぜずに別々に扱い、全ての環境で最適な分類器を獲得することで汎化するモデルを獲得するというものである。これについて説明しよう。

データ x が与えられたとき、それをNNなどを使って変換して得られたデータ表現を $\Phi(x)$ とし、また、それを入力として分類結果を

返す分類器を w とする。この2つをあわせた、入力から分類結果を返す関数 $w(\Phi(x))$ を合成関数の記号を使って $w \circ \phi$ と表す。また、環境 e におけるこの分類器による期待損失 l を損失関数としたとき、$R^e(w \circ \Phi) := \mathbb{E}_{X^e, Y^e}[l(f(X^e), Y^e)]$ とおく。このとき、全ての環境 $\mathcal{E}$ で損失を最小にできるような分類器が同じである場合、そのデータ表現 Φ は環境にわたって不変な分類器 $w \circ \Phi$ を導出できていると考えられる。

$$w \in \arg\min_{\bar{w}} R^e(\bar{w} \circ \Phi) \quad \text{for all} \quad e \in \mathcal{E}$$

しかし、全ての環境を列挙することは不可能なので、代わりに訓練データとして集めた環境上 $\mathcal{E}_{tr}$ でこれを達成できるものを考えるとする。

$$\min_{\Phi, w} \sum_{e \in \mathcal{E}_{tr}} R^e(w \circ \Phi)$$
$$\text{subject to} \quad w \in \arg\min_{\bar{w}} R^e(\bar{w} \circ \Phi) \quad \text{for all} \quad e \in \mathcal{E}_{tr}$$

これをIRMと呼ぶ。この最適化では全ての環境上で最適となる線形分類器が同じとなるようなデータ表現を獲得することが目標となる。ERMにおいても全ての環境から集めたデータに対する損失を 0 にできるのであれば、各環境の損失も 0 にできている（一般に仮定する損失が非負であるならば）ので同じことができるが、IRMではこの分類器を線形分類器という表現力が小さいモデルに制約していることが重要である。ERMはニューラルネットワークのように過剰パラメータモデルを使う場合、訓練誤差が最少となった場合でも誤った相関を捉えてしまうことが知られている。さらに学習に使うモデルが、訓練誤差やテスト誤差を 0 にできるモデルを含んでいない場合でも誤った相関を見つけてしまう。IRMはこうした場合にERMよりも汎化するモデルが獲得できると主張されている。

このIRMの最適化はそのままでは各環境で最適化が必要であり、内側と外側の二重の最適化が必要なため実用的ではない。そこで次のような変更を考える。まずデータ表現が任意の非線形変換を使うのであれば線形分類器は任意のモデルを仮定して使ってもよいことを利用し、最適な分類器を単にデータ表現の最初の成分をそのまま返すモデルに固定する。次に損失関数として凸関数を利用する場合、モデルの最適性をその勾配が 0 に近いかで評価できることを利用する。

これらを利用して、先程の最適化問題の代わりに次の最適化問題を解くことを考える。

$$\min_{\Phi} \sum_{e \in \mathcal{E}_{tr}} R^e(w \circ \Phi) + \lambda ||\nabla_{w|w=1.0} R^e(w \circ \Phi)||^2$$

ここで w はスカラー値であるダミーの線形分類器であり、データ表現 $\Phi(x)$ の第一成分をそのまま使うという分類器である。この最適化を行うことでオリジナルのIRMと同じ結果が得られる。さらに、第二項のノルムの不変推定量として環境から非復元抽出でサンプルを2つ $(x_1, y_1), (x_2, y_2)$ をとり、それらの w についての勾配の内積を使うことができる。

$$||\nabla_{w|w=1.0} R^e(w \circ \Phi)||^2$$
$$= \mathbb{E}_{(x_1,y_1),(x_2,y_2)}[\langle \nabla_{w|w=1.0} l(w \circ \Phi(x_1), y_1),$$
$$\nabla_{w|w=1.0} l(w \circ \Phi(x_2), y_2) \rangle]$$

このIRMを使って学習して得られた分類器はERMと比較し、新しい環境に汎化しやすいことが簡単な実験では確かめられている[2]。

今の多くの統計的推論、機械学習は相関関係だけを捉え、因果関係は捉えていないため多くの誤った結論を導いていることが昨今指摘されている。一方で、因果関係を捉えるには相関を見つけるよりも多くの条件や仮定が必要となり、まだ汎用的に使える因果推論は登場していない。しかし、因果関係を捉えることは機械学習にとっての究極の目標である汎化を達成するために重要である。この逆に、因果関係をとらえられるような表現をデータから学習で獲得できるのかも課題となっている今後、推論技術が発展すると共にデータも能動的に取得することが進むにつれ様々な因果推論ができ、真に汎化するモデルが獲得できるようになることが期待される。

---

1) M. Arjovsky et al.,"Invariant Risk Minimization," https://arxiv.org/abs/1907.02893
2) J. Pearl, "The Book of Why," Basic Books.

（2019年9月号掲載の記事に加筆）

対称性は学習にどのように活かせられるか

世の中の多くの現象には対称性がみられる。ある対象 M が対称性 S を持つとは、S で指定された操作 $g \in S$ を M に適用しても M が不変であることを言う。例えば、球体は任意の回転操作を適用しても球体のままだし、左右対称な図形は左右反転操作を適用しても図形は変わらない。この対称性は幾何的な対象だけでなく多くの物理現象にみられる。

対称性は有用な帰納バイアスとして働く

この対称性は学習する際の強力な帰納バイアス、つまり学習結果に大きく貢献できる学習データ以外の事前知識として利用することができる。

例えば画像認識タスクにおいて、認識対象の物体が少し上下左右に平行移動したとしてもその画像の分類結果が変わることはない。この場合、分類結果は平行移動操作に対して不変であるといえる。また多くの場合、画像に回転操作や左右の鏡面反射操作を適用しても分類結果が変わることはない。一方で、画像を上下に反転させると意味が変わる場合もある。

このような対称性が最もみられるのは点群データやメッシュデータ、CTスキャンや顕微鏡データ、地理データ、およびその解析結果である。こうしたデータを表現する際、座標や基準となる軸の選び方は恣意的であり、座標を変換したり軸を変えたとしても結果は変わらない。例えば点群データから、それらの各点が物体の何に対応しているのか（頭、足など）は、点群データが回転したとしても関係ない。顕微鏡で与えられた画像中のがんに対応する細胞は、画像が回転したとしてもがんであることには変わらない。

一方、機械学習のモデルやニューラルネットワークは必ずしもこのような入力や問題が持つ対称性を考慮できない。例えば、総結合層からなる多層パーセプトロンで画像を分類する場合、入力画像を平行移動や回転させると、その結果は違うものになってしまう。モデルからすれば、どの部分に対称性があり入力に対するどの変化は考慮して、どの変化は無視するかは自明でないためだ。そのため、データオーグメンテーションと呼ばれる訓練データに対して様々な変換を適用し（画像の場合は平行移動、回転、鏡面反射など）、データや問題が備える対称性を全て別々な現象としてモデルに覚えさせることが一般的だ。この場合、モデルは、本来であれば同じモデルが使える場合も別々のモデルで表現する必要があり、（水増しした）訓練データ数、パラメータ数も大きくなってしまう。

対称性をうまく利用し成功したCNN

このような対称性をモデルに直接組み込むことが多くされてきた。例えば、CNN（畳み込みニューラルネットワーク）は全ての位置で同じパラメータを持った線形変換、つまり畳み込み操作を適用するため、入力が平行移動してもその出力も同様に平行移動するだけである。

CNNは全ての位置の結果を（位置の順序に依存せず）まとめあげる操作、例えばグローバルプーリングなどを適用すれば、入力に対する平行移動操作に対して、結果は不変となる。一方で、プーリング操作を適用しなければこの入力に対する変換は、出力でもそのまま残っている。例えば入力を右に平行移動したら、結果も右に平行移動する。この入力の操作が出力の操作にも残っていることは同変という言葉で説明できる。

不変と同変

入力 x に対してある操作 g を適用して得られた結果 $x' = g(x)$ に対し変換 Φ を適用した結果 $\Phi(x')$ が、元の入力を変換した結果 $\Phi(x)$ と常に一致する場合、つまり $\Phi(x) = \Phi(x') = \Phi(g(x))$ である場合、変換 Φ は操作 g に対し不変（Invariant）であると呼ぶ。

これに対し、入力 x に対してある操作 g を適用して得られた結果 $x' = g(x)$ を変換した結果 $\Phi(x')$ と、元の入力を変換した結果 $\Phi(x)$ の後に、g に対応する操作 $\pi(g)$ を適用した結果とが常に一致する場合、つまり $\pi(g)(\Phi(x)) = \Phi(g(x))$ である場合、変換 Φ は操作 g に対し同変（Equivariant）であると呼ぶ（図1-1）。このと

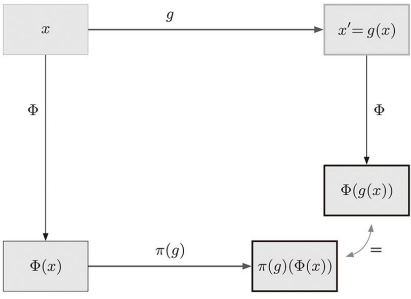


図1-1　変換Φが操作gに対し同変 (Equivariant) である場合
右下の2つの要素について、π(g)(Φ(x))＝Φ(g(x))である場合、変換Φが操作gに対し同変 (Equivariant) であると呼ぶ。

き、g が群の元、x がベクトル空間 V の元、各 g に対して $\pi(g)$ が V 上の線形変換で、操作 g が、$\pi(gh) = \pi(g)\pi(h)$ を満たすとする。このような操作を扱う分野は表現論と呼ばれ、数学や物理などで広く扱われている。

不変というのは同変の特殊例で $\pi(g)$ が恒等変換（入力をそのまま返す変換）である場合である。不変は入力に対する操作情報をつぶしているともいえる。CNNで最後にプーリング操作を使った場合は、入力に対する平行移動情報は消されている。また、(プーリングを使わない) CNNは平行移動操作に対して同変であり、入力を平行移動して変換した結果は、そのままの入力を変換した結果を平行移動した結果と一致する。この場合たまたま $\pi(g) = g$ である。物理分野でも座標のとり方によって物理法則は変わらないという共変性という概念が使われているがこれは同変性の一種である。

同変は、情報の"もつれを解く (Disentanglement)"上で重要な概念である。ある操作に対して同変である変換は、出力に対しその操作が可能 (Steerable) [1]であり、その操作に必要な情報が壊れず失われていないことを意味する。例えば移動、回転に対し同変であるような変換は対象の一種の姿勢情報を内部で保っており、変換後も、その姿勢情報を使うことができる。

同変なネットワークは学習データ効率が良い

同変であるニューラルネットワークは操作間で同じパラメータを共有できるので学習データ効率が良い。例えば回転操作に対し同変であるニューラルネットワークの場合は、学習データで回転しているようなサンプルがあったとしても同じパラメータを使って処理することができる。一般の画像データは向きが揃えられておりメリットは少ないかもしれないが、点群データやCTスキャンデータでは向きを揃えるということが難しく、向きに依存せず同じ学習モデルが使えることは非常に大きな利点となる。

様々な操作に対しDNNを同変にする研究

これまでにニューラルネットワークを様々な操作に対し同変にするにはどうすればよいかが考えられてきた。その中でも代表的な回転操作、鏡面反転操作に対して同変な変換をどのように実現できるかが研究されてきた。

2016年に最初に提案されたGroup Equivariant Convolution Network (G-CNN) は、少数の離散的な回転操作、平行移動、鏡映変換に対し同変であるようなニューラルネットワークである[2]。この場合、畳み込み操作だけでなくプーリング操作や非線形な活性化関数に対する工夫も提案された。一方で、G-CNNは操作数に比例する計算

量、メモリ使用量が必要であり、操作数を増やすことが難しかった。

G-CNNの翌年に発表されたSteerable CNN[1]は、同変である変換の計算量は操作数に比例する必要はなく、その操作群で定義される既約表現の組み合わせのみに制限した形で線形変換（畳み込みの場合はカーネル）を表現すればよいことが示された。これによって、同変である場合はむしろ少ない計算量で処理でき、実際少ないパラメータで表現できることが示された。この平行移動、回転、鏡面反射操作一般に対して、同変とする場合にはどのような既約表現を使えばよいかは既に求められている[2]。

また、球面[3]（ICLR 2018のベストペーパー）やグラフなどユークリッド空間以外の空間の場合に同変とする手法の研究も進んでいる。これらを一般化した等質空間（homogeneous space）上で同変とできるCNN[5]も進んでいる。

これら同変なネットワークは理論的に興味深いだけでなく、実用的にも大きな成果を上げている。代表的なものとして、メッシュを入力とし、どの頂点が参照のどの部位に対応するのかを求めるレジストレーションタスクがある。

このレジストレーションタスクは対称性があるため従来手法は様々な工夫をする必要があったが、同変なCNNを使った場合、特別な工夫をせずほぼ100%に近い精度を達成することができた[6]。また、細胞組織などを顕微鏡で調べた結果に対しても同変なCNNを使った手法[7]が少ないサンプル数から効率的に学習でき、従来手法を上回る最も高い精度を達成したと報告されている。

この同変や既約表現といった概念の理解には高度な数学の知識が必要だが、使うだけであれば、代表的な操作に対して同変な変換を、従来モジュールを置き換えるだけですぐ使うことができる[8]。

空間中の各点がグラフの頂点に対応するような場合、グラフニューラルネットワークを使って同変性を容易に導入できる。例えばEGNN[9]は、各頂点の更新時は、頂点間の相対位置のみを使うことで、座標変換に対し同変であることを達成している。

また、いくつかの対称性については対称性を導入してもNNの表現力を全く失わずに定義することができる[10]。現実世界の問題にはまだ考慮できていない対称性も多く存在し、今後、同変な変換を通じてこれらの対称性をモデルに組み込んでいくことが考えられるだろう。

1) T. Cohen et al.,"Steerable CNNs," ICLR 2017.
2) T. Cohen et al.,"Group Equivariant Convolution Networks," ICML 2016.
3) M. Weiler et al.,"General E(2)-Equivariant Steerable CNNs," NeurIPS 2019.
4) T. Cohen et al.,"Spherical CNNs," ICLR 2018.
5) T. Cohen et al.,"A General Theory of Equivariant CNNs on Homogeneous Spaces," NeurIPS 2019.
6) P. d. Haan et al.,"Gauge Equivariant *Mesh* CNNs: Anisotropic convolutions on geometric graphs," https://arxiv.org/abs/2003.05425
7) S. Graham et al.,"Dense Steerable Filter CNNs for Exploiting Rotational Symmetry in Histology Images," https://arxiv.org/abs/2004.03037
8) https://github.com/QUVA-Lab/e2cnn
9) V. G. Satorras et al.,"E(n) Equivariant Graph Neural Networks," ICML 2021.
10) O. Puny et al.,"Frame Averaging for Invariant and Equivariant Network Design," ICLR 2022.

（2020年6月号掲載の記事に加筆）

1-9

機械学習の新べき乗則：大きなモデルを使うと汎化しサンプル効率も改善する

世の中の現象の多くがべき乗則で説明できる。例えばジップの法則は出現頻度が x 番目に大きい単語の頻度は、1位の単語と比較して $1/x$ の頻度であるというものである。パレートの法則（全体の数字の8割が2割の構成要素で実現されている）、友人の数や地震の大きさの分布などについてもそうだ。式で書けば、ある変数 x とその結果において $f(x) = ax^k + c$ という関係が成り立つというものである。ここで a, k, c は定数である。

2020年10月に米OpenAIの研究者らによって発表された論文[1]は、投入する計算リソース、データサイズ、モデルサイズと深層学習の達成可能な性能（損失）間でべき乗則が成り立つと報告した。そもそも成り立つということが驚きであるとともに、そこから導かれる「（計算リソースやデータサイズが一定なら）大きなモデルを使った方が汎化性能も良く、学習効率も良い」という事実がこれまでの機械学習の常識とは大きく異なるものであったため、研究者の間で大きな話題となった。さらに、予想される最適なモデルが今使われているモデルよりはるかに大きいため、今後の深層学習を使ったシステム開発に大きく影響を与える可能性がある。これが事実だとすれば、深層学習（ディープラーニング）の秘密を解き明かし、今後高性能なシステムを作る道筋を立てた金字塔的な研究成果だと著者は考える。このべき乗則について解説していく。

生成により成り立つべき乗則

彼らは、近年GPT-3などで成功しているTransformerのデコーダのみを使った自己回帰モデルを使って、クロスエントロピー損失を最小化するようにして生成モデルを学習する問題について扱った。元々このモデルは自然言語の事前学習として利用されていたが、今回は画像（各画素を順番に並べた場合と、ベクトル量子化で符号化した場合）、ビデオ、画像と言語間の相互変換、数学の問題（問題文と回答が自然文と記号列で書かれており、問題で条件付けして回答を生成できるかという問題）などについて扱った。

いずれの問題も与えられた系列データは $\mathbf{x} = x_1, x_2, \ldots, x_n$ と表され、各文字をこれまでに生成した文字に条件付けして生成するようにして学習する。

この実験では投入する計算リソース C、データサイズ D、モデルサイズ N を変えた場合にクロスエントロピー損失がどのように変わるのかを調べた。その結果、損失と C、D、N 間に次のようなべき乗則が成り立つことがわかった。

$$L(x) = L_\infty + \left(\frac{x_0}{x}\right)^{\alpha_x}$$

ここで x には C、D、N のいずれかが入る。例えば、3 2 * 3 2 の画像の場合、モデルサイズ N に対して $L(N) = 2.2 + (N/1.9 \times 10^8)^{-0.14}$ という式で損失が予測できる。

このことから、C、D、N を大きくするだけで損失を小さくすることができ、またどれだけ大きくすればどの程度の性能が達成できるのかを学習する前から予測することができる。また、クロスエントロピー損失 L はデータ確率分布 P とモデルが表す確率分布 Q を使って

$$L = H(P) + KL(P||Q)$$

と表される。$H(P)$ は P のエントロピーであり、$KL(P||Q)$ は P から Q へのKLダイバージェンスである。通常のクロスエントロピーの説明では最尤推定としての $L = \sum_{x \sim P} \log Q(x)$ がでてくるがそれに $H(P)$ を加えると上の式が出てくる。このように分解した場合、第一項のエントロピーが L_∞ に対応し、KLダイバージェンスが $\left(\frac{x_0}{x}\right)^{\alpha_x}$ に対応すると考えられる。そして、エントロピーは削減不能な損失であり、KLダイバージェンスは削減可能な損失である。

削減可能損失を減らすことが後続タスクの性能向上に重要

クロスエントロピー損失は生成モデルとしての性能だが、これで学習したモデルを事前学習として使うと他の後続のタスク性能が変わるかについて、OpenAIはImageNetの分類タスクで調べた。すると、削減可能損失が小さくなるほど、後続タスクのテスト損失が小さくなり、汎化性能が改善される

ことが確認された。

削減不能損失のスケールに比べて削減可能損失のスケールは小さいため、学習の後半では損失の減り方が非常に停滞しているようにみえるが、実際は削減可能損失が下がっていく。そして、この部分が後続タスクの汎化性能の向上にとても重要であることがわかった。事前学習したGPT-3が様々な後続タスクに驚くほどうまく適用できているということをサポートする結果となった。

最適な計算リソース、モデルサイズ、データサイズ間の関係

また、計算リソースCが決まっている場合の最適なモデルサイズ$N_{opt}(C)$は$N_{opt} \propto C^{\beta}$の関係が成り立ち、$\beta$は驚くことに問題毎に変わらず0.7付近であることがわかった。

同様に、その固定の計算資源の中での最適なモデルサイズNとデータサイズDの関係にも

$$D \propto N^{0.4}$$

という関係が成り立つことも分かった。ここから、計算資源が増やせる場合はイテレーションを増やして計算時間を増やすより、モデルを大きくする方がよい。またデータを大きくする割合よりも、モデルを大きくする割合を大きくすべきだという、今の常識とは違う結論が導かれる。

これらの数字がどのような意味を持つかについて、表を作った（表1-1）。例えば、2倍の計算リソースが使える場合、最適なモデルサイズは1.6倍、データは1.2倍となる。同様に100倍の計算リソースが使える場合はモデルを25倍、データを3.6倍にするのが最適である。データが増える速度に比べてモデル、計算の方が急速に増えていくことが分かる。これらについては2022年3月に英DeepMind社より報告された論文[20]では修正が必要であり、計算が一定の場合は最適なデータサイズの増加とモデルサイズの増加は同程度であるとされている。この場合、元の論文が報告するよりは最適なモデルサイズの増加率は抑えられる（相対的に同じモデルサイズの時の最適な投入計算リソース、データサイズは大きくなる）。

従来の機械学習ではオッカムの剃刀などであるように、モデルのパラメータ数（複雑度）は必要最低限にする方が汎化性能を高くできるというのが理論的にも説明されていたし、実験的にもそうだった。教科書でも最初の方に書かれている基本的な考えである。それが深層学習の場合はモデルが大きい方が汎化性能がむしろ高いということがわかってきており、それを説明する理論もいくつか登場してきている。今回

表1-1　計算リソースとモデルサイズ・データサイズとの関係

計算リソース	モデルサイズ	データサイズ
1	1	1
2	1.6	1.2
10	5	1.9
100	25	3.6

の結果もそれを支持する。

さらに、今回の結果から予想される最適なモデルサイズは、現在使われているモデルサイズよりも圧倒的に大きい。例えば、8x8という低解像度の画像の生成モデルの最適なモデルサイズは10億パラメータ付近と考えられる。さらに解像度が大きくなった場合（画像の本質的な次元は元の次元よりずっと少ないが）、最適なモデルサイズは指数的に大きくなり、この100倍から1万倍近くになるとみられる。また、言語モデルの学習においては既存の最大サイズのモデルでも学習が収束できず、これより大きなサイズが必要だと予想されており、検討はついていない。

もう1つの意外な事実として、モデルサイズが大きい方がサンプル効率が良い、つまり同じ学習データしかない場合に、より効率的に学習できるということも示された。

他に興味深い実験として画像とそれを説明するキャプションのデータを同時に学習し、画像と言語間の相互情報量についてもモデルサイズに対するべき乗則が成り立つことがわかっている。ここで予想されるテキストから画像を生成するモデルの最適なパラメータ数は約3兆と予想される。また、1枚の絵は千の語に匹敵するという言葉があるが、この実験結果からは32x32の画像は2、3単語分の情報を持っていることがわかった。

今回の知見などに基づき、OpenAIは2021年1月にテキストから画像を生成するDALL·E[8]、画像からそれに紐づくテキストを予測するCLIP[9]を発表した。これらは大きなデータセットと大きなモデルを使って学習し、これまでのシステムと比べてはるかに高い柔軟性と汎化性能を達成したシステムである。

残された課題

今回の発見は様々な問題を投げかけた。まず、今回の発見はあまりに現在の知見とは異なるものであり、そもそも事実なのかどうかを確認する必要がある。これについては、その後他のグループが様々な条件で追試や似たような実験をした際にもべき乗則を支持する結果が得られている。今回は

Transformer＋自己回帰モデル＋クロスエントロピー損失の場合の結果であるが、他のモデル、タスク、損失の場合にも同様に成り立つかの検証が必要である。この傾向についてはTransformer以外のモデル（例えば、MLPなど）でも同様の傾向がみられた。一方自己回帰モデル、クロスエントロピー損失以外でこのような傾向がみられるかについては十分検証されていない。

さらに実験的にこれが事実だとするならば、なぜこのようなべき乗則が成り立つのかについての理論的な理解が必要となる。なぜ異なるドメイン、問題で同じべき乗則が成り立ち、さらに最適モデルサイズについては係数まで一致するのか。なぜモデルを大きくすると汎化性能も改善されるのかといった部分はまだ現在の理解とのギャップが大きい。

実用的な観点でも問題を投げかける。今回の結果は、データ、モデル、計算リソースを大きくしさえすればどこまでも性能を上げられるということを示しており、しかも投資効果が予測可能な形で示されている。これに従い、その後様々な有力な企業や研究グループでのモデルの大規模化がおこなわれた。このような大規模なモデルはFoundation Model[5]と呼ばれ、一度大きなコストを使って作っておいたモデルを様々な後続タスクに使うという流れは広がっている。しかし今回予想された最適サイズのモデルは現在の技術の延長線上で実現するのは困難で、新しい技術やハードウエア開発が必要になると思われる（既にパラメータ数が兆のオーダーのモデルは登場してきているが[6]）。

また、一度大きなモデルを事前学習して作っておき、それを使ってZero-shotやFew-shotで様々なタスクに使うという考え方は自然言語処理以外にも広がるだろう。様々な分野で大きな学習済みモデルが使われるような時代が来る前夜の状態なのかもしれない。

---

1) T. Henighan et al.,"Scaling Laws for Autoregressive Generative Modeling," https://arxiv.org/abs/2010.14701
2) J. Hoffmann et al.,"Training Compute-Optimal Large Language Models," https://arxiv.org/abs/2203.15556
3) https://openai.com/blog/dall-e/
4) https://openai.com/blog/clip/
5) R. Bommasani et al.,"On the Opportunities and Risks of Foundation Models," https://arxiv.org/abs/2108.07258
6) W. Fedus et al.,"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," https://arxiv.org/abs/2101.03961

（2021年3月号掲載の記事に加筆）

頑健なモデルには過剰パラメータ表現が必要

現在のディープラーニングで使われるモデルは過剰パラメータ（Over Parameterized）表現、つまり訓練データ数よりもずっと多くのパラメータ数を持つモデルが使われている。例えば100万枚からなるImageNetを学習データに使い、数億のパラメータからなるモデルを使うことは一般的である。過剰パラメータ表現は効率が悪いだけでなく、従来の機械学習の理論からすれば、過学習しやすいと思われていた。

一般にn個の方程式が与えられた時、n個の未知数を決定すれば良い。同様にn個の訓練データが与えられた時、n個のパラメータを持つモデルを使えば訓練データを完全に予測できるようなモデルを作ることができる。そのため、パラメータ数が訓練データ数よりはるかに多いモデルは冗長なようにみえる。

一方で、最近の実験結果や機械学習の新べき乗則（前節で紹介）が示すように、教師情報が豊富である（自己回帰問題など）などいくつかの条件を満たせば、パラメータ数が多いモデルであればあるほど学習効率が高く、汎化性能が高いことが分かっている。これはパラメータ数が多ければ多いほど、初期値から汎化性能が高いフラットな解に到達しやすいためと考えられている。

機械学習モデルの重要な性能要件として、汎化性能のほかに頑健性がある。これは入力に摂動を加えたとしても予測結果が大きく変わらないというものである。しかし、特に工夫をせず学習したモデルは、入力にほんのわずかな、狙って設計された摂動（敵対的摂動）を加えただけで予測結果が大きく変わってしまう。例えば、パンダの画像に対し、人には気づかれないようなノイズを加えて、パンダ以外の自動車やりんごといった任意の予測結果に変えることができる。

このような敵対的摂動に対する頑健性は、機械学習のモデルを現実世界にデプロイする際に必要不可欠である。例えば、自動運転車やロボットの画像認識に対する敵対的摂動を使った攻撃などが既に多く実証されている。

この頑健性においても過剰パラメータ表現が重要であることがわかってきた。過剰パラメータ表現を使うことは頑健なモデルを作るための必要条件であり、さらに現在のモデルはそこで示唆される必要なモデルサイズに比べてまだかなり小さいことが示唆された[1]。この研究はNeurIPS 2021のOutstanding Paper Award（Best Paper相当）の1つに選ばれた。今回はこの結果について紹介する。

関数のリプシッツ性

はじめに、必要な知識としてリプシッツ性（Lipschitzness）とIsoperimetryについて説明する。リプシッツ性は関数の滑らかさを示す。関数fが定数$L \in R^+$、任意の入力$\mathbf{x}, \mathbf{y}$に対し常に

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L||\mathbf{x} - \mathbf{y}||$$

が成り立つ時、fはL-リプシッツ性を持つとよび、Lをリプシッツ定数と呼ぶ。これは入力がu離れたとしても関数の出力が高々Luしか変わらないことを意味する。

機械学習モデルfが訓練データに対し完全な予測をし、かつ（少なくとも）その訓練データの周辺でも訓練データと同じ結果を出力するためには、関数fはリプシッツ性を持ち、そのリプシッツ定数は小さいことが必要である。

Isoperimetry

また、データ分布μがIsoperimetryを満たすとは、任意のL-リプシッツな有界関数fに対し、関数の出力が関数の期待値とtより大きく異なる確率がガウシアンと同じかそれよりも急速に小さくなることをいう。例えばガウシアンや単位球面上の一様分布、また多様体仮説として知られる低次元のデータが高次元データに埋め込まれている場合はIsoperimetryを満たす。

頑健性とパラメータ数の関係

これを踏まえた上で、2021年6月に米MicrosoftResearchのSébastien Bubeck氏らは次を証明した[1]。なお、数学的に高度な知識が必要となる部分を省略しているので、正確な結果を知りたい場合は文献[1]を確認してほしい。

p個のパラメータからなる関数fを使って$i.i.d.$（独立同分布）であるn個の学習データ$(x_i, y_i)_{i=1}^n$を多少のノイズを許して

フィッティングする。入力は d 次元ベクトルであり、出力は $y_i \in \{+1, -1\}$ とする。また、入力分布はIsoperimetryを満たすとする。この時、関数 f のリプシッツ定数 $\mathrm{Lip}(f)$ は

$$\mathrm{Lip}(f) \geq \sqrt{nd/p}$$

となる。

この結果が示すのはリプシッツ定数を小さくする、特に定数にするためにはパラメータ数 p を $p \sim nd$ とする必要があることである。そして、モデルサイズと、そのモデルの頑健性の間にはトレードオフの関係があるということである。小さいモデルは頑健にすることはできず、大きなモデルの方が頑健にできる。小さなモデルではどのように工夫しても（リプシッツ性という観点で）頑健なモデルを作ることはできない。

簡単な例

ここでは単純な問題設定において頑健なモデルを nd 個のパラメータを使って構成する例を説明する。

入力分布が、単位球面上の一様分布と考える。この場合、異なるサンプル $\mathbf{x}_i, \mathbf{x}_j$ は高い確率で $||\mathbf{x}_i - \mathbf{x}_j|| \geq 1$ を満たす。この時、$g(0) = 1$、$z \geq 1$ の時 $g(z) = 1$ となるような滑らかな関数 g を用意し、この関数を動径基底関数として使った

$$f(\mathbf{x}) = \sum_{i=1}^{n} g(||\mathbf{x} - \mathbf{x}_i||) y_i$$

という関数は訓練データに完全にフィットし、かつリプシッツ定数が $O(1)$ となるような関数である。このパラメータ数は $p = n(d+1)$（n 個の x_i と y_i）であり、先程の下限と一致する。

現在のモデルはまだ小さい

この理論が実際の結果にどのぐらい一致するのかをみてみる[1]。MNISTデータセットは $n = 6 \times 10^4$ 枚の画像からなり、次元数は $28^2 = 784$ である。このデータセットに対して、様々なサイズのニューラルネットワークを使って学習させると、パラメータ数が $10^5 \sim 10^6$ のあたりでモデルの頑健性が大きく変わる。このことから、MNISTの実効次元数は $15 \sim 150$（n/p）程度ではないかと予想される。

また、ImageNetデータセットの場合、1.4×10^7 枚の画像からなり、各画像の次元数は大体 2×10^5（256×256）となる。画像の場合の有効次元数は実際の次元数の1/10、1/100程度と推測されることから、頑健なモデルを実現するにはパラメータ数を $10^{10} \sim 10^{11}$ にすることが必要と予想される。これは現在使われているモデルのパラメータ数 $10^8 \sim 10^9$ 個とは100倍近くの差がある。

現在、ニューラルネットワークは敵対的攻撃に弱いことが報告されており、頑健なモデルにするために様々な研究が進められているが、大きなデータセットにおいて頑健なモデルがまだ存在していないのは、単に現在のモデルサイズが小さすぎるためという可能性がある。

大きなモデルとどのように付き合うか

今回の結果はこれまでの統計や機械学習で考えられていたサンプル数とパラメータ数は同程度必要という常識を破る結果である。実用的な観点からいえばパラメータ数は少ない方がよいが、必要条件となれば、ハードウエアやアルゴリズムで大きなモデルをサポートできるようにするか、もしくは表現学習などを工夫し実効次元（d）を下げ、小さなモデルでも頑健性を持てるようにする必要がある。

1) S. Bubeck et al., "A Universal Law of Robustness via Isoperimetry," NeurIPS 2021.

（2022年3月号掲載の記事に加筆）

第2章

人の学習

2-1 脳内で誤差逆伝播法が起きているか?

ニューラルネットワークは脳（神経系）を模して作られた。神経細胞（ニューロン）は、入力刺激が閾値を超えた場合に発火し、活動電位を発生させ、他の細胞に情報を伝達する。この発火が連鎖的に起こることで情報を伝え、計算を実現する。神経細胞間の情報伝達には、シナプスと呼ばれる化学物質を用いた機構が使われている（別の仕組みもある）。

脳は活動の中で、ニューロン間のつながり（の強さを変えていくことで（例えば、シナプスに放出された化学物質を受け取る受容体の数を変化することで）環境に適応、つまり学習していく。それを模したニューラルネットワークも同様に、ニューロンとシナプスに対応する重みで構成され、学習は重みを調整することで行われる。現在のニューラルネットワークでは誤差逆伝播法（BP：back propagation）がその汎用性と学習効率の高さから広く使われている。しかし、脳内ではどのような学習則が働いているのかについてはまだ解明されておらず、脳内でBPと同様の学習則が働いているかは分かっていない。

BPを利用した学習則について簡単に紹介しよう。学習の多くは、目的関数を最小化する最適化問題を解くことで実現される。パラメータθを最適化対象とする目的関数$L(\theta)$が与えられたとする。例えば、入力xから出力yを予測したい回帰の教師あり学習を実現するには、教師データ$\{(x_i, y_i)\}$と予測関数$f(x;\theta)$、損失関数$l(y, y_i)$が与えられた時、$L(\theta) = \Sigma_i l(f(x_i;\theta), y_i)$を目的関数として使う。目的関数が小さくなる、つまり教師データと予測関数の値が同じになるパラメータを探す問題だ。

この時、目的関数$L(\theta)$を最小化するようなパラメータθ^*は、$L(\theta)$のθについての勾配$v=\frac{\partial L(\theta)}{\partial \theta}$を求め、$\theta := \theta - \alpha v$と更新することで求められる。ただし、$\alpha > 0$は学習率と呼ばれるパラメータである。$v$は教師データの一部を使って推定することもできる。これを確率的最急降下法（SGD）と呼ぶ。現在では、Adamなど2次の勾配情報をオンラインで近似した手法が広く使われている。なお、目的関数が凸でない場合、SGDは必ずしも最適解は与えないが、幅の広い大きなニューラルネットワークの場合、SGDにより見つかった解はほとんど全て同様の性能を持つことが予想されている[1]。

関数fがニューラルネットワークのように複数の関数が組み合わさって構成されている場合、勾配vをどのように求めるかが問題となる。この問題に対し、1986年にDavid Rumelhart氏らが誤差逆伝播法を提案した。これは後ろ向き自動微分とも呼ばれる。出力から入力に向けて逆向きに、目的関数のそれぞれの関数の出力に対する勾配を再帰的に計算していく。誤差を出力から入力へ伝播させていくように見えることから、誤差逆伝播法（BP）と呼ばれる。BPは通常の順計算の約3倍（順計算に加えての場合は2倍、BP全体では3倍）の計算量で求められるため、パラメータ数が多くても効率的に求めることができる。そして、目的関数に対する各パラメータの勾配を直接求めることができるため、学習が非常に効率的であることが知られている。

脳内で誤差逆伝播が難しいと考えられている理由

脳内でBPの実現が難しいと考えられているのには、次の理由がある。1つ目は、BPの中で誤差を出力から入力に向けて逆向きに誤差を伝播させる際、順方向の時に使った重みをWとした時、Wの転置を誤差に掛ける必要があるためだ。ニューロン間のつながりは一方向であるため、Wの転置を求め、それを掛けることはそのままでは実現できない。

2つ目は、BPは順方向の計算をした後に逆方向の計算を必要とする点である。順方向と逆方向の計算が混ざってしまわないよう、全てのニューロン間で「順方向の計算中なのか、逆方向の計算中なのか」の同期を取る必要がある。3つ目は、デジタル計算機では誤差の伝播は正確であり、100層を超えても学習できるが、脳の機構にはアナログな部分が含まれ、雑音もあるため途中で誤差が消失する可能性が高い点だ。

こうした理由から、脳内ではBPではなく別の方法で学習がなされているのではないかと考えられている。例えば、Helmholtz MachineやBoltzmann Machineはいずれも局所的な情報のみを使って重みを更新できることから、脳内の学習機構として有望視されている。一方で、BPは工学的には有効なため、脳の仕組みの上でもBPが実現できると分か

れば、脳の学習機構としても有力候補として挙がってくる。

脳内でのシナプス強化の仕組みであるSTDP

脳内の学習則はまだ分かっていないが、どのような場合にシナプスが強化されるのかは、いくつか現象が知られている。シナプスには向きがあり、シナプス前ニューロンをx、シナプス後ニューロンをyとする。Hebb則は、xとyが同時に発火した場合、シナプスが強化されるという現象である。この現象はより詳細に調べられ、STDP (Spike timing dependent plasticity：スパイクタイミング依存可塑性) と呼ばれる現象があることが分かった。これは、xが発火した直後にyが発火した場合、シナプスが強化され、その逆にxが発火した直前にyが発火した場合、シナプスが弱くなるという現象である。

Yoshua Bengio氏らは、STDPは実は次のような更新則ではないかと予想した。xの発火率を$\rho(x)$、yの発火率の変化を$\dot{y}$、xからyへのシナプスの強さをW_{yx}とした時、

$$\Delta W_{yx} \propto \dot{y}\,\rho(x) \qquad (1)$$

のように表されるというものだ[2]。

このもともとのアイデアはGeoffrey Hinton氏が2007年に提唱したものである。なぜこれがSTDPと一致するかというと、yの発火率が増えている場合、xの直後にyが発火する確率が高くなる。このように考えたのは、(1) の式は次の目的関数の最小化として導出されるからだ。

$$J = \| f(s_{t-1}, \eta_{t-1}) - s_{t+1} \|^2 \qquad (2)$$

ここでs_tは時刻tのニューロン全体の状態であり、η_{t-1}は雑音、fは脳のダイナミクスであり、次の時刻の状態を決める。これを最小化するように重みを決めるということは、脳は1つ前の時刻の状態s_{t-1}と、それに雑音が加えられた情報から、次の時刻の状態を予測できるようになるというものだ。雑音が加わっていることから、雑音除去自己符号化器と同様の効果が期待できる。雑音を加えた情報から、それを除去するように学習することは、確率分布の勾配を推定してその情報の重要な部分を認識し、そこから再生成することを意味する。雑音除去自己符号化器は確率分布自体を推定しているのと同じであることが示されている[3]。

これは生物的にも意味があることと考えられる。これまで得た情報から未来の状態（未来の盛覚器から得られる内部状態）を予測できるようになっていれば、敵を速く察知したり、獲物をより速く捕まえたりできるからである。脳の学習機構が常に将来を予測し、それと実際に観測された結果の誤差を基に学習しているのではないかという予想はこれまでもあった。それが実際に脳で観察されているSTDPとひも付けられた点で、興味深い予想である。

その後、様々な"脳内で実現可能なBP"が提唱されてきた。しかし残念ながら現時点でBPに匹敵するような学習効率や汎用性を持った学習機構は見つかっていない[4]。

脳の学習則はまだ分かっていない部分が多くある。2013年には神経細胞だけでなく樹状突起も発火することが分かり、これが誤差の逆伝播のように働くのではないかとの推察もなされている。また、脳内には電気信号以外にもタンパク質の相互作用や遺伝子発現の制御など様々な形で情報がやりとりされている。これらが脳の仕組みをどう支えているのかはまだ分かっていない[5]。DNAが発見されてから50年、遺伝子の仕組みがますます複雑であることが分かり始め、謎が深まっているのと同様、脳の仕組みについても、解明されるに従って初めてその複雑さが分かってくるのかもしれない。

1) A. Choromanska, et al.,"The Loss Surfaces of Multilayer Networks,"AISTATS, 2015
2) Y. Bengio, et al,"An objective function for STDP," http://arxiv.org/abs/1509.05936
3) Pascal Vincent,"A Connection Between Score Mathing and Denoising Autoencoders,"
4) T. P. Lillicrap et al.,"Backpropagation and the brain," Nature Reviews Neuroscience 2020.
5) http://timdettmers.com/2015/07/27/brain-vs-deep-learning-singularity/

(2015年12月号掲載の記事に加筆)

2-2 脳の学習システム：Learning system in brain

日本と英国の人工知能と神経科学の研究者が集まったGatsby/科研費合同ワークショップ[1]が2017年5月にロンドンで開かれ筆者も講演者として参加してきた。このワークショップは英University College LondonのGatsby計算神経科学研究所（Gatsby研）DirectorであるPeter Dayan氏と沖縄科学技術大学院大学（OIST）教授の銅谷賢治氏らの呼びかけで開催された。

Dayan氏はベイズ法を神経科学に適用し、神経伝達物質が予測誤差や不確実性を表すのに使われていることを明らかにした。このほかにも強化学習でのQ学習やTD（λ）の収束性の証明、今の深層生成モデルにつながるヘルムホルツマシンなど多くの業績がある。英DeepMind社の創設メンバーの多くがGatsby研の出身であり、DeepMind社が強化学習やベイズ法、エピソード記憶など神経科学に着想を得た手法を積極的に手掛けていることからもその影響の大きさが分かる。一方、銅谷氏は後述する脳内での学習システムを明らかにするため実際の動物やシミュレーションモデルを使い脳内での学習システムの解明を進めている。

現時点でも脳がどのように情報を処理し、学習しているのかについては完全に解明されていないが、いくつか有力な考えが提唱されている。ここでは銅谷氏が1999年に提唱した学習モデル[2]を基に説明する。これは脳内では、大脳基底核（Basal Ganglia）による強化学習、小脳（Cerebellum）による教師あり学習、そして大脳新皮質（Cerebral Cortex）による教師なし学習が連動して学習されているというものである。これらを順にみていこう。

強化学習を担う大脳基底核は線条体（Striatum）、淡蒼球（Pallidum）などから構成される。線条体では、現在の状態sにもとづいて状態価値関数$V(s)$や、それと行動aを組合せた行動価値関数$Q(s, a)$を使って状態価値、行動価値が評価される。これらの推定された価値はドーパミンニューロンに渡されTD誤差の計算に使われる。TD誤差は「現時点での価値を使った収益（将来に渡ってもらえる報酬の和）の予測値」と、「今もらった報酬と次の時刻の価値から計算された収益の予測値（TD目標）」とのずれであり、この誤差を少なくすることで将来の収益を予測できる。この予測誤差に応じてドーパミンニューロンからドーパミンが放出され、線条体における価値関数が更新される。一方、価値関数による評価は淡蒼球に渡され、そこで最適な行動を決定するのに使われる。決定された行動は視床を経由して大脳新皮質に伝搬され、実際の運動命令にマッピングされる。

大脳新皮質は教師なし学習を担い、入力をそれを構成する因子、成分に分解し、それらと元の入力とのマッピングを実現する。入力を独立した成分に分解する操作としては独立成分分析（ICA）や主成分分析（PCA）が知られている。例えば、教師なし学習であるボルツマンマシンは入力変数がガウシアン、隠れ変数が2値変数の場合、その最尤推定によって入力の独立成分を見つけられ、入力変数がガウシアン、隠れ変数がガウシアンの場合はその最尤推定は入力の主成分を見つけられることがわかっている[3]。大脳新皮質がボルツマンマシンを実現しているかはわかっていないが、ボルツマンマシンはシナプスの代表的な更新則であるヘブ則に似た更新則を持つ。そのため、大脳新皮質が同じような計算を行い、入力を因子に分解する、また入力と因子間の関係を格納することはできると考えられる。大脳新皮質により、強化学習における状態や教師あり学習における入出力は抽象化され、学習の汎化が実現される。

教師あり学習を担う小脳は入力から出力へのマッピングを学習し、運動制御や、習慣化された行動列の生成を担う。小脳は脳内でニューロン数が最も多く大半を占めているが、これにより無限ともいえる非常に多くの教師あり学習を互いに干渉せずに学習できる。入力から出力を予測する教師あり学習は様々な場面で利用することができる。例えば、強化学習における状態sからの最適な行動決定$a=f(s)$や、ある状態sで行動aを実行したら、次の状態s'がどうなるか$s'=f(s, a)$という環境モデルも教師あり学習で実現される。

これら3つの学習は連動して実現される。例えばピアノ演奏の練習の場合を考えてみよう。練習の最初のうちは一つ一つどの鍵盤を押すかを考えて行動するため時間がかかり、不正確である。ここでは大脳基底核による強化学習が中心的

な役割を果たし、どの行動が最も良さそうかを考え時間をかけて評価をする。この段階では毎回の行動で判断が必要でありBG脳も“疲れ”る。しかし、練習を繰り返していくと、意識せずに、ある状態の次にどう行動するか（どの鍵盤を押すか）を自動的に決定できるようになる。ここでは意志の介入は必要ない。この状態では小脳による教師あり学習が進み、ある状態sの次にどの行動を選ぶのか$a=f(s)$、そしてどの状態になるのか$s'=f(s, a)$が小脳により瞬時に計算されるようになっている。小脳による教師あり学習が機能し始めれば高速かつ正確に弾けるようになり、意識せず弾けるようになる。そのため、ピアノを弾きながら他のことを考えられるようになる。

一般に、ある状態sの時にどのように行動aするかについては、強化学習が主体のモデルフリー、小脳による次の状態予測結果を使って評価したモデルベース、そして、小脳による記憶ベースに分けられる。

$a^*=\mathrm{argmax}\ aQ(s, a)$	モデルフリー（大脳基底核）
$a^*=\mathrm{argmax}\ a[r+V(f(s', a))]$	モデルベース（大脳基底核）
$s'=f(s, a)$	（小脳）
$a^*=g(s)$	記憶ベース（小脳）

このように脳内では様々な異なる学習機構を持つモジュールが連動してタスクを学習していく。

さらに他の学習結果を使って別の学習をするブートストラップも行われる。例えば、入力画像を同じ物体同士にグループ分けするセグメンテーションの学習では、はじめに「一緒に動いているものは一つの物体である」という事前知識を使って、動画からのセグメンテーションをはじめに学習する。次にこの学習結果を使って静止物体のセグメンテーションの正解を作り、静止画像のセグメンテーション方法を学習すると考えられている。

人の脳では大部分の学習のブートストラップの仕組みが遺伝的（生得的）に組み込まれていると考えられている。受精卵から成体への成長の過程では、単純な細胞群が適切なタイミングに分化し相互作用しながら複雑な組織へと変化していく。これと同様に脳内でも様々な学習問題が適切な時期に発動されるように設定されており、成長するに従って今までの学習された結果をモジュールとして使ってより複雑な学習問題を解いていく。

現在の機械学習や深層学習はこのような複雑な学習システムは実現しておらず、かろうじて人が複数の学習機構を組合せて実現している程度である。例えばDeepMind社の囲碁ソフトであるAlphaGo学習の過程では次の手予測の教師あり学習、それを使っての行動価値関数の学習、それを使って盤面の状態価値関数の学習と進んでいく。

神経科学と人工知能は近年、再度接近し交流が始まっている[1]。人や動物の知能の仕組みを参考に学ぶとともに、現在の人工知能が複雑な問題を解けるようになるにつれ、それらの仕組みが実際に脳内で起きているかを調べる動きも出てきている。ワークショップの最後に参加者の一人が次のような興味深いアンケートをとっていた。「将来の人工知能は脳に似たシステムとなるのか、全く違うシステムになるのか。そして脳の仕組みの解明が人工知能の実現に役に立つのか、立たないのか」。私は「脳とは全く違うシステムになるが脳の仕組みの解明が人工知能の実現に役に立つ」と回答した。この結果が正しいのかは10年後に振り返ってみよう。

1) http://www.brain-ai.jp/events/detail/616/
2) K. Doya, “What are the Computations of the cerebellum, the basal ganglia, and the cerebral cortex?,” Neural Networks, vol.12, pp.961-974, 1999.
3) R. Karakida et al.,“Analysing Feature Extraction by Contrastive Divergence Learning in RBMs,” Deep Learning and Representation Learning Workshop, NIPS2014.
4) A. Marblestone et al., “Toward an Integration of Deep Learning and Neuroscience,” https://doi.org/10.1101/058545

（2017年7月号掲載の記事に加筆）

2

学習手法

第3章

学習手法

3-1

学習のエンジン：数理最適化 Adagrad, RMSProp, Adam

今回は学習のエンジンともいえる数理最適化について紹介する。はじめに数理最適化が、どのように学習問題で使われているのかをみてみる。入力xから出力yを予測する回帰を考える。この予測のために、パラメータθで特徴付けられた関数$f(x;\ \theta)$を学習させる。パラメータθは一般には実数ベクトルである。例えば、線形回帰の場合は

$$f(x:\theta)=\langle x,\ \theta\rangle+\theta_0$$

であり、ニューラルネットワークの場合は複数の線形関数と、非線形関数である活性化関数を組み合わせた

$$f(x:\theta)=f_k(\cdots f_2(f_1(x:\theta_1):\theta_2)\cdots:\theta_k)$$
$$\theta=(\theta_1,\ \theta_2,\cdots\theta_k)$$

と表される。学習データ$\{(x_i, y_i)\}$に対して、予測関数の誤りの度合いを表す関数を損失関数と呼ぶ。例えば、回帰の場合は、$l(x, y,\ \theta)=(y-f(x:\theta))^2$のような二乗誤差を使う。損失関数を使って次のように定義される目的関数

$$L(\theta)=\sum_i l(x_i, y_i,\ \theta)$$

を最小化するようなパラメータθ^*を求めれば、学習データをうまく予測できるような関数$f(x:\theta^*)$が得られる。

なお、回帰問題ではなく分類問題の場合はヒンジ損失やロジスティック損失、確率モデルの学習では負の対数尤度を損失関数として利用する。

さらに学習データに過剰適合してしまう問題を防ぐため、パラメータθについての正則化項$R(\theta)$（例えば$|\theta|^2$や$|\theta|$など）を使い、$L(\theta)+R(\theta)$を最小化するようなθを求める最適化問題とするのが一般的である。

基本戦略はボールをより低い位置に動かすこと

最適化問題は隆起のあるコースでのパターゴルフのような問題だと考えることができる。ボールの平面上の位置がパラメータ、ボールの高さが関数の値に対応し、最も低い場所へボールを移動させる問題である。

基本的な戦略は今のボールを重力に任せて、より低い方向に持っていくことである。これは$L(\theta)$の勾配$G(\theta)=\frac{\partial L(\theta)}{\partial\theta}$を求め、$\theta=\theta-\alpha G(\theta)$（$\alpha>0$は学習率）のような更新を繰り返すことに対応し、最急降下法（GD: Gradient Descent）と呼ばれる。

しかし、学習の最適化問題の場合、$L(\theta)$が全データから定義されているため、$G(\theta)$を求めるには全ての学習データを毎回走査する必要があり計算量が大きい。そのため、現在は一部の学習データから勾配の推定量を求め、それを使う確率的最急降下法（SGD: Stochastic Gradient Descent）が使われることがほとんどである。

最適化問題が、穴が1つであり、とにかく下って行けば最適解に到達できる凸最適化問題であれば、最適解を理論保証付きで高速に見つけられることが知られている。例えば、（潜在変数の存在しない）サポートベクトルマシンやロジスティック回帰などを使った場合がそれに当たる。詳細については捕著[1]などを参考にしてほしい。

ニューラルネットでの最適化の難しさ

一方でニューラルネットワークの学習時の最適化問題は次の2点から難しいことが知られている。1つ目は、局所解が多数存在し、しかもその局所解は層の数に対して指数個だけ増えていくという問題である。ゴルフでいえば、そこら中に穴が無数に空いているようなコースを途中の穴に落とさずにボールを低いところまで持って行くという問題である。

2つ目は、ほとんど傾きがない高原のような領域（プラトー）が存在するということである。プラトーでは傾きがほとんどないため、ボールは転がらず止まってしまう。

それでは傾きが小さい場合は更新幅を大きくすれば良いと単純には考えられるが、鞍点と呼ばれる馬の背中のような場所があり、ここではある方向は平らで別な方向は急であるような場所もある。更新幅を大きくしただけでは、方向が傾きが急な方に向いていれば値は振動、発散してしまう。鞍点ではさらに慎重な更新幅の調整が求められる。このようなプラトーや鞍点が指数個存在する。

また、パラメータの数が大きい（数百万から数十億）ため、基本的にパラメータ数に対して線形の計算量を持つアルゴリズムしか現実的には利用可能ではない。

様々な最適化手法が提案

こうした問題を解決するために様々な最適化手法が提案されてきた。ここでは代表的な手法のAdagrad、RMSProp、Adamについて紹介する。

SGDの学習の際に問題となるのは、学習率αの調整である。このαの大きさが適切でない場合、学習が遅く、振動や発散が起こるという問題が発生する。そのため各次元ごとに学習率をコントロールする方が望ましい。

Adagrad[2]は、パラメータの次元ごとのスケールが違う問題に対応できるよう、勾配のノルムの二乗和s_iを求めておき、学習率をそれで割る。

$$s_i := s_i + g_i * g_i$$

$$a_i = \frac{\alpha}{\sqrt{s_i}}$$

ただし、g_iは現在の更新時の勾配のi次元目の値である。これは、T回目の学習率をこれまでの履歴から求められた勾配の期待値と$\sqrt{T}$で割っているようにみることができる。

RMSPropでは勾配の二乗の移動平均m_iを求め、Adagradと同様にそれで学習率を割る。

$$m_i = \alpha m_i + (1-\alpha)\, g_i\, g_i$$

$$a_i = \frac{\alpha}{\sqrt{m_i}}$$

RMSPropはAdagradと比べて最適化問題が途中で変わるような場合に対応することができる。鞍点のような局所的に更新率が変わる場合はRMSPropのような移動平均を使う方が良い。これは、自然勾配法における曲率の対角化近似のオンライン推定とみることもできる。

Adam[3]は、勾配の1次モーメントと2次モーメントの移動平均を求め、それらを使って学習率をコントロールする。

$$m_i = \beta_1 m_i + (1-\beta_1)\, m_i$$

$$v_i = \beta_2\, v_i + (1-\beta_2)\, g_i * g_i$$

$$a_i = \frac{m_i}{\sqrt{v_i}}$$

今の勾配をそのまま使わずにこれまでの移動平均を使うのは、ゴルフでいえば下っている時に勢いがついていて、それをもって降下していくようなイメージである。モーメントを使うことで、収束が速くなることが知られている。なお、Adamの論文にある推定量の修正項は煩雑になるため、ここでは省略している。

ニューラルネットワークの最適化手法はその後も発展を遂げているが、基本的な考え方は変わっておらず、1次モーメント、2次モーメントをオンラインで求め、これらを使って学習率を修正する方法が一般的である。

単に最適解を得るだけでなく汎化性能が高くなるような解を求めることも重要となっている。この場合はフラットな解の重心に近い解を求める必要があり、最適化途中の指数移動平均を使うことで見つけることができる。

1) 海野ほか、『オンライン機械学習』、講談社
2) J. Duchi, et al.,"Adaptive Subgradient Methods for Online Learning and Stochastic Optimization, "JMLR, 12(Jul),pp.2121－2159, 2011.
3) D. P. Kingma et al.,"Adam: A Method for Stochastic Optimization," ICLR 2015.

（2016年3月号掲載の記事に加筆）

3-2

乱択化フーリエ特徴関数：大規模問題でもカーネル法を適用可能に

機械学習の分野ではカーネル法と呼ばれる手法が広く使われている。1990年代後半から流行したSupport Vector Machine（SVM）は、このカーネル法を利用することで非線形な問題を扱えるということで注目された。

カーネル法は、2つの要素$x_1, x_2 \in X$間の"近さ"を定義したカーネル関数$K(x_1, x_2)$を利用する。カーネル関数は2つの値が似ているほど大きな値を取り、離れているほど小さい値を取るような関数である。

例えば、ベクトル間の内積$<x_1, x_2>$はカーネル関数の1つである。よく使われるRBFカーネルはパラメータ$\sigma > 0$を伴って次のように定義される。

$$K(x_1, x_2) = \exp\left(-\frac{\|x_1 - x_2\|^2}{\sigma^2}\right)$$

これは、x_1とx_2が全く同じ値であれば1、違う値（特に$\|x_1 - x_2\| \gg \sigma$）の時、ほぼ0となるような関数である。

自然言語処理などスパースなデータを扱う場合、多項式カーネル

$$K(x_1, x_2) = (<x_1, x_2> + c)^p$$

などが利用される。カーネル関数は何でもよいわけではなく、半正定値関数であることが求められる。次の条件を満たす場合に半正定値関数と呼ぶ。

「任意のn個の要素$(x_1, x_2, \cdots, x_n)$に対し、$K(x_i, x_j) = K(x_j, x_i)$であり、かつ任意の実数$a_i, a_j$について$\sum_{i,j} a_i a_j K(x_i, x_j) \geq 0$が成り立つ」

カーネル関数が半正定値関数の時、カーネル関数は内積の形で

$$K(x_i, x_j) = <\varphi(x_i), \varphi(x_j)>$$

のように表せることが知られている（Mercer's theorem）。つまり、要素をある写像φで別の空間（ヒルベルト空間）に飛ばした上で、そこで内積を取っているようにみることができる。

なお、以降では対象の要素xとして実数ベクトルR^pを考えるが、カーネル関数が定義さえできれば、文字列やグラフといった数値ではない要素も扱うことができる。

非線形問題を線形問題で扱えるようにする

カーネル法の優れた点は非線形の問題を線形の問題の枠組みの中で扱える点である。これを説明するために、少し長くなるが、入力$x \in R^p$から出力$y \in \{-1, 1\}$を推定する二値分類の問題を考えてみる。この二値分類の推定に線形識別器を使う。線形識別器のパラメータの重みベクトルを$w \in R^p$、バイアスを$b \in R$とした時、線形識別器$f(x; w, b)$は

$$y = f(x; w, b) = <w, x> + b$$

のように定義される。線形識別器は空間を、wを法線、切片がbであるような超平面で2つに分割し、与えられた点がその表側か、裏側かで二値分類していると考えられる。学習データ(x_i, y_i)に対し、それらをうまく当てられるかを示す損失関数$l(x, y, f(x; w, b))$を定義する。損失関数は学習データをうまく当てられている場合は0、当てられない場合は大きい値を返すような関数である。

例えば、二値分類でよく使われるヒンジ損失関数は

$$l(x, y, f) = max(0, 1 - yf(x; w, b))$$

と定義される。この損失関数の和を小さくするようなパラメータを求めることで学習を行う。さらに、パラメータの正則化として、重みベクトルの$L2$ノルム$\|w\|^2/2$を加えた次の目的関数を考える。

$$L(w, b) = \frac{\|w\|^2}{2} + \sum_i l(x_i, y_i, f(x; w, b))$$

このとき、損失関数が凸関数であれば、$L2$ノルムは凸関数であるため、それらの和であるLについても凸関数となる。凸関数は入力に対する勾配が0の時に最小値を取る。Lの重みベクトルwについての勾配が0になる条件を調べると、次のようになる。

$$\frac{\partial L(w,b)}{\partial w} = w + \sum_i l'(x_i, y_i, w, b)\, x_i = 0 \qquad \text{(i)}$$

ただし、$l'(x_i,y_i,w,b)$ はlのwに対する勾配である。Lを最小化する最適なパラメータをw^*、b^*とし、$a_i = -l'(x_i, y_i, w^*, b^*)$とおくと、

$$w^* = \sum_i a_i x_i$$

となって、最適な重みベクトルw^*は学習データの重み付き線形和として表せることが分かる。この表現を、最初の線形識別器の式に代入すると、

$$f(x; w^*, b^*) = \langle w^*, x \rangle + b^*$$

$$= \sum_i a_i \langle x_i, x \rangle + b^* \qquad \text{(ii)}$$

と表されて、線形識別器は与えられたデータと、学習データのそれぞれとの内積の重み付き和として定義できることが分かる。

一般に、学習の目的関数がパラメータの$L2$ノルムを含み、かつ損失関数の中でパラメータwと入力xが内積$\langle w,x \rangle$の形で出現する場合に、最適な重みベクトルはこのような訓練データとの内積の重み付き和として表現できる。ロジスティック回帰やAda Boostingなど広い学習器がこれに所属する。

RBFカーネルは複雑な分類曲面を形成

ようやくカーネル法を説明する準備が整った。この最適な線形識別器の表現 (ii) の内積部分 $\langle x_i,x \rangle$ をカーネル関数$K(x_i,x)$に置き換えてみよう。

$$f(x) = \sum_i a_i K(x_i, x) + b^* \qquad \text{(iii)}$$

これは何をしていることになるか。前半で説明したように、カーネル関数は元の入力xを別の空間上の点$\varphi(x)$に変換し、その上で内積を取っているようにみなすことができる。つまり、内積の計算部分をカーネル関数に置き換えるだけで、あたかも$\varphi(x)$の空間で線形識別関数を考えていることになる。元の空間では線形では分類できなかった問題もカーネル関数が定義する空間中では線形に分類可能にでき、その空間上で分類できる。

例えば、RBFカーネルの場合、$\varphi(x)$は無限次元であり、その無限次元上で線形識別器を使っていることに対応する。それを元の空間でみれば、近いものほより近く、遠いものはより遠くにしたような歪んだ空間に対応する。

カーネルが対応する空間での超平面は元の空間では非常に複雑な分類曲面に対応する。また、カーネル関数の計算は高次元に展開して内積を明示的に求めるよりもずっと高速に求めることができる。RBFカーネルは無限次元間の内積であるにもかかわらず、その値はベクトル間の四則演算とexpの計算だけで求められる。これを、カーネルトリックと呼ぶ。

遅かったカーネル法を一変させたRFF

カーネル法は強力であり、SVMをはじめとしたカーネル法は一時期、機械学習の世界を席巻した。一方で、当時のカーネル法は計算量が大きいという致命的な問題を抱えていた。例えば、学習事例数がNの時、カーネル法を使った手法の学習時の計算量は$O(N^3)$ (オンライン学習やスパース化など工夫をしても$O(N^2)$) であり、その利用時の計算量も (iii) の式から分かる通り$O(N)$時間かかる。

機械学習は学習事例が多いほど強力であり、現在は$N=10^6$~10^9が使える時代であるから、カーネル法の計算量は大きな問題となり、「カーネル法は精度が良いが遅く、大規模データには使えない」とこれまで認知されていた。

こうした状況を変えたのが2007年に登場した乱択化フーリエ特徴関数 (RFF: Random Fourier Features) を使ったカーネル法の高速化手法である[1]。RFFはカーネル関数が$K(x_1,x_2) = \varphi(x_1 - x_2)$のように、差で表現される場合に使える方法で、先ほどのRBFカーネルなどがそれに当たる。そのフーリエ変換は、次のように表せることが知られている。

$$K(x_1, x_2) = \int_{R^d} \exp(iw^T(x_1 - x_2))\, dp(w)$$

$$= \int_{R^d} cos(w^T(x_1 - x_2))\, dp(w)$$

ここで、$p(\omega)$はRBFカーネルの場合は、それぞれの成分が正規分布$N(0,s^2)$に従うことが分かっている。この積分値を$p(\omega)$に従ってωをm個サンプリングし、モンテカルロ近似する。

$$\sum_{i=1}^{m} cos(\omega_i (x_1 - x_2))$$

$$= \sum_{i=1}^{m} cos(\omega_i x_1) cos(\omega_i x_2) + sin(\omega_i x_1) sin(\omega_i x_2)$$

$$= <\varphi(x_1), \varphi(x_2)>$$

ただし、

$$\varphi(x_i) = (cos(\omega_1 x_i), cos(\omega_2 x_i), \cdots, cos(\omega_m x_i), sin(\omega_1 x_i), \cdots, sin(\omega_m x_i)) \in \mathbb{R}^{2m}$$

である。ここでは、$cos(\alpha-\beta)=cos(\alpha)cos(\beta)+sin(\alpha)sin(\beta)$を使った。つまり、カーネル関数はランダムに選んだフーリエ特徴の内積で近似できる。

このサンプリング数mは大きくなくてもカーネル値の近似が精度よくできることが分かっている[2]。特徴ベクトルを陽に表せられるため、このRFFを使ったモデルは従来の線形モデルと同様の計算コストですみ、データ数Nに対して$\mathcal{O}(N)$の計算量で学習し、$\mathcal{O}(1)$時間で関数評価をすることが可能である。

ニューラルネットでカーネル関数を学習する研究も

この変換が何をしているかの直感的な説明をしよう。ωは周波数に対応しており、x_1とx_2が近い値であればωが大きい値（高周波数）になったとしても、ωx_i (i=1,2)の値は近い値をとり、結果として対応する特徴値の符号は同期し、それらの積は正の値を取る。一方でx_1とx_2が違う場合、周波数が大きくなると、ωx_iの値は大きく異なる。その特徴値の正負はほぼランダムになり、期待値は0となる。結果として、x_1とx_2が近い値であれば1に近い値となり、そこから外れると急速に0に近づく。

さらに、この乱択化特徴量の計算を高速にするために、それぞれの特徴値を独立に計算するのではなく、構造化された計算を使うことで、元の特徴次元数が多くても高速に計算できる手法も登場している[3]。他のカーネル関数でも同じようなアイディアが適用できるか、またカーネル関数のパラメータについての勾配も計算することでニューラルネットワークと組み合わせてカーネル関数自体を学習するような研究も始まっている。

こうした改善から、カーネル法は今やデータ数に対して線形の計算量で求めることができ、大規模なデータにも適用できるようになった。

1) A. Rahimi et al., "Random features for large-scale kernel machines," NIPS 2007.
2) B. K. Sriperumbudur et al., "Optimal Rates for Random Fourier Transform," NIPS 2015.
3) Q. Le et al., "Fastfood - Approximating Kernel Expansion in Loglinear Time," ICML 2013.

（2016年7月号掲載の記事に加筆）

3-3 正則化：汎化能力をどのように手に入れられるか

機械学習を学ぶ際に最初に考えることは、学習データをそのまま覚えておけばよいのではないかということである。テスト時は単に学習データ中の同じデータ、または似たデータを探し、その結果をそのまま使う手法である。このようなアプローチを丸暗記と呼ぶことにしよう。

丸暗記によって学習データに対してはうまく処理できるようになるが、未知データに対してはうまくいくかの保証はない。特に実世界で学習したい問題の多くは入力が高次元であるため、学習時にありえるデータを全て列挙できず、記憶しておくこともできない。機械学習の主要なテーマは、有限の学習データから学習し、未知のデータをうまく扱えるようなルールや知識を獲得することだ。このような能力を汎化能力と呼ぶ。

一般に学習モデルの表現力が高ければ高いほど、学習データをうまく処理できるようになる。例えば、入力ベクトルから連続値を推定する回帰の問題において、1次式を使うよりは2次式、3次式を使ったほうが、より学習データにフィットする曲線を見つけることができる。モデルというのは粘土のようなものであり、表現力というのは、その“柔らかさ”のようなものだと思ってもらえればよい。粘土の形によって、学習データをうまく扱えるかが決まる。固い粘土（表現力が低いモデル）は力を入れても思った形にならず、柔らかい粘土（表現力が高い）であればぐにゃぐにゃで思い通りに形を変えることができる。それでは、モデルの表現力は高ければ高いほどいいかというと、そうでもない。モデルの表現力が高すぎると、学習データはうまく行っても未知データをうまく分類できなくなる可能性が高くなる。これを過学習や過適合と呼ぶ。

学習が行っていることはモデルの候補の中からモデルを1つ選ぶということであり、表現力が高いということはモデルの候補数が多いということである。例えば、学習データを一番うまく分類できるモデルを100の候補から探す場合と100万の候補から探す場合、後者の方がモデルの種類が多いため、学習データをより正しく分類できるものが見つかる可能性が高い。一方で、見つかったモデルは偶然、学習データを分類できているだけで、そのデータの特性を捉えていない可能性も高い。表現力が高いモデルの中に真のモデルも含まれている可能性もあるが、それを見つけるのは砂浜の中から金の砂粒を見つけるように難しい。

理解を深めるために別の見方をしよう。学習アルゴリズムは学習データに応じてパラメータを推定する。サンプリング時のランダム性によって真のパラメータとは違うパラメータを推定する。表現力が低いモデルは真のモデルを推定することができず、その期待値も真のパラメータからずれている可能性が高いが、推定のバラつきは小さい（high-bias、low-variance）。表現力が高いモデルはその期待値は真のパラメータに近い可能性が高いが（low-bias）、推定のバラつきは大きい（high-variance）。モデルのずれというのは、$bias+\sqrt{variance}$ぐらいで評価できるので、このbiasとvarianceのトレードオフをみて最適なモデルの表現力を選ぶ必要がある。

正則化は汎化性能を改善する手法全般を指すが、ここではその中でもモデルの表現力を落とす方法を考える。表現力を落としても学習データを同じくらいうまく処理できれば未知データをうまく処理できる可能性が高くなる。

近年成功しているニューラルネットワークはパラメータ数が非常に多く、変数の高次の関係も扱えるのでモデルの表現力が非常に高い。どんなに難しい学習問題でも、学習データを全て正しく分類できるようなパラメータは簡単に見つけてしまう。そのため、正則化がとても重要となる。以下にニューラルネットに対する正則化の代表的な手法を紹介しよう。

(1) 表現力にペナルティを与える

モデルの表現力を数値化し、それをコスト関数に加える方法は最もよく使われる手法である。例えば、モデルのパラメータがベクトルvで表されている時、そのノルム$\|v\|_2$（L2ノルム）、$\|v\|_1$（L1ノルム）、$\|v\|_\infty$（L∞ノルム）をコストとして使う。L2ノルムは大きい値に非常に大きなペナルティが、小さい値には非常に小さいペナルティが加えられ、推定結果は多くが小さい値となる。L1ノルムは疎な推定結果、つまり多くの値が0であるような推定結果になりやすくなる。L2ノルム、L1ノルムはそれぞれ、パラメータの事前確率として正規分布、

ラプラス分布を想定している場合の事後確率最大化推定に対応する。パラメータがベクトルではなく行列の場合は、フロベニウスノルム（全成分値の二乗和）や行列の特異値のノルムをコストとする場合が多い。

(2) アーリーストッピング

アーリーストッピングは古典的だが強力な手法である。学習データのほかに評価データを用意し、学習を進めている過程での評価データ上の性能をモニタリングする。そして評価データ上の性能が最も良い時のパラメータを保存しておく。最適化を途中で止めることはL2ノルムによる正則化と同じ効果があることが知られている。そして、アーリーストッピングの場合、L2ノルムの様々なハイパーパラメータの結果を効率的に調べることができる。

(3) アンサンブル/ドロップアウト

複数のモデルを組み合わせて多数決を取るアンサンブルも正則化の重要なテクニックである。なぜ複数のモデルを使うと正則化の効果があるのかを簡単に説明しよう。k個の回帰モデルがあり、i番目のモデルの学習事例に対する誤りはe_i、その分散は$E[e_i^2]=v$、その共分散は$E[e_ie_j]=c$であるとする。この場合、k個のモデルの推定結果の平均を推定として使った場合の誤りは$1/k\Sigma_ie_i$である。一方で、その二乗誤差は$v/k+(k-1)c/k$である。もし各モデルが完全に相関している場合（$c=1$）は期待二乗誤差はvであり一つのモデルを使った場合と同じである。一方で全てのモデルが全く相関していない場合（$c=0$）、二乗誤差の期待値はv/kとなる。つまり、モデルの数を増やせるほど、期待二乗誤差は線形に減らせる。「三人よれば文殊の知恵」ということわざがあるが正確には、「"考え方が違う"三人よれば文殊の知恵」ということだろう。潜在変数モデル、ベイズ推定もアンサンブルの一種としてみることができる。それぞれパラメータの分布に従って重み付けされた無数のモデルでアンサンブルをとっているとみることができる。このアンサンブルを効率的に実現する手法がドロップアウトである。学習時には各ニューロンの出力を確率pでランダムに0とし、テスト時にはニューロンの出力をp倍に増やしたのを使う。学習時は毎回、違うニューラルネットワークを使っているとみなせ、テストはそのアンサンブルとなる。

(4) パラメータ数を減らす

パラメータ数を少なくし表現力を落とす手法も効果的である。例えば、ユニット数がNからMの総結合層のパラメータ数はNMであるが、これらの層の間にユニット数がLの層を挟めば全体のパラメータ数は$NL+LM=(N+M)L$となり、$L<<N,M$のときパラメータ数を減らすことができる。これを通常の機械学習では低ランク近似とよび、層としてボトルネックレイヤーと呼ぶ。また、モデル中のパラメータを共有することもパラメータ数を減らすのと同じ効果がある。例えば、CNNは場所に依存せず同じパラメータを共有することでパラメータ数を劇的に減らし、RNNも時間に依存せず同じパラメータを使うことでパラメータを減らしている。パラメータの表現に使うビット数を減らすことも正則化の効果がある。

(5) 入力にノイズを加えても出力が似た値をとるように制約

入力に何らかのノイズを加えた上で出力の差のL2ノルムを正則化項として使う手法も有効である。入力がガウシアンである場合、このL2ノルムは関数のヤコビアン行列のフロベニウスノルムの近似であることが分かっている。また、VATと呼ばれる手法では出力が確率分布である時、その確率分布間のKLダイバージェンス距離の最小化を行う。これは確率分布から定義されるフィッシャー行列の最大固有値を小さくしている正則化とみなせる。フロベニウスノルムは固有値の二乗和であることから、ガウシアンノイズを加えた場合は全ての固有値を小さくし、VATは最大固有値のみを小さくしているとみなせる。この両者のどちらが良いかは問題の性質に依存する。

このほか、勾配降下法による学習ダイナミクスが生み出す正則化、問題についての事前知識を利用した正則化、モデルに不変性や同変性を導入することによる正則化も有効である。

（2016年8月号掲載の記事に加筆）

3-4 誤差逆伝播法による期待値最大化

機械学習において確率変数を含む最適化問題は様々な問題でみられる。ここでは、その中でも重要な対象関数$f(x)$の期待値が最大となるような確率分布$p(x;\theta)$を求める問題を考える。

$$J(\theta) = E_{p(x;\theta)}[f(x)]$$
$$= \int_x p(x;\theta) f(x) dx$$

この問題の具体例として強化学習の方策勾配法、生成モデルの変分推定を挙げる。

強化学習の方策勾配法

強化学習は、エージェントが環境の中で次々と行動を決定し報酬を受け取る中で、その将来にわたって受け取る報酬の和を最大化するような問題であった。エージェントの状態がsの時、エージェントの行動aを決定する確率分布を方策と呼び、$\pi(a|s;\theta)$で与えられるとし、このエージェントの期待収益を$J(\theta)$とする。また、方策πに従って状態遷移していった時、将来的に状態sに到達する割引付き確率を

$$d^\pi(s) = \sum_{t=0}^{\infty} \gamma^t Pr[s_t = s | s_0, \pi]$$

とする、ただし$0<\gamma\leqq 1$は報酬の割引率である。この時、期待収益$J(\theta)$のパラメータθについての勾配は

$$\nabla_\theta J(\theta) = \sum_s d^\pi(s) \sum_a \nabla_\theta \pi(s,a;\theta) Q(s,a)$$

として求められることが知られている、ただし$Q(s,a)$は行動価値と呼ばれ、状態sの時、行動aをとった時の期待収益である。これを方策勾配定理と呼ぶ。この定理において重要なのは$\nabla_\theta d^\pi$の項がないことである。方策が変わると、将来的に各状態に到達する確率は変わるのだが、それが勾配へ与える影響はない。これにより、本来は行動と状態が独立ではない強化学習の問題において、行動のみの勾配を求めて最適化することができる。この場合、対象関数が$Q(s,a)$であり、確率分布が方策$\pi(a|s;\theta)$である。

生成モデルの変分推定

次の例として、変分自己符号化器（VAE）などで使われる変分法による潜在変数モデルの学習を考えてみよう。はじめに潜在変数zを簡単な分布から生成し、次にデータxを$p(x|z;\theta)$に従って生成するような確率モデル

$$p(x;\theta) = \int_z p(x,z;\theta) dz$$

を考えよう。例えば、

$$z \sim N(0, I)$$
$$x \sim N(\mu(z;\theta), \sigma(z;\theta))$$

ただし$\mu(z;\theta)$と$\sigma(z;\theta)$は、ニューラルネットワークを使う。このパラメータを最尤推定するためには、対数尤度を最大化すればよいが、対数尤度は解析的には求まらないので、事後確率分布$p(z|x)$を近似する別の確率分布$q(z|x;\phi)$を使って、

$$\log p(x;\theta) = \log \int_z p(x,z) dz$$
$$= \log \int q(z|x) \frac{p(x,z)}{q(z|x)} dz$$
$$\geqq \int q(z|x) \log \frac{p(x,z)}{q(z|x)} dz \quad (*)$$
$$= E_{q(z|x)} \log F(x,z)$$

と求まる下限の最大化を求めることで得られる。ただし、

$$F(x,z) = \log p(x,z) - \log q(z|x)$$

であり、(*)はジェンセンの不等式を利用した。この場合、対象関数が$F(x,z)$であり、確率分布が$q(z|x)$である。

勾配最大化

さて、最初の問題に戻り、確率分布が$p(x;\theta)$、対象関数が$f(x)$の時の期待値

$$J(\theta)=E_{p(x;\theta)}[f(x)]$$

をθについて最大化する問題を考えてみよう。これは、勾配$v=\nabla_\theta J(\theta)$を求め、$\theta=\theta+\alpha v$と更新すれば勾配法で求めることができる。ただし、$\alpha>0$は学習率である。しかし、その勾配を次のように直接求めた場合、

$$\nabla_\theta J(\theta)=\int_x f(x)\nabla_\theta p(x;\theta)dx$$

この式は期待値の形ではなく、一般に解析的には解けない。また、サンプリングをしてモンテカルロ推定をするにも$\nabla_\theta p$はもはや確率分布ではないので、サンプリングできない。この問題を解決するために2つの方法が提案されている。

尤度比法、REINFORCE

1つ目は制御分野では尤度比法、強化学習分野ではREINFORCEと呼ばれている方法である。

$$\nabla_\theta \log p(x;\theta)=(1/p(x;\theta))\ \nabla_\theta p(x;\theta)$$

であることに注意すると、

$$\nabla_\theta p(x;\theta)=p(x;\theta)\nabla_\theta \log p(x;\theta)$$

と表わせ、$p(x;\theta)$が外にでてくる。この変形を使うと

$$\nabla_\theta J(\theta)=E_{p(x;\theta)}[f(x)\ \nabla_\theta \log p(x;\theta)]$$

のように勾配をpが確率分布である期待値の形で表すことができる。

尤度比法による推定値は不偏推定量、つまりサンプル数を増やせば増やすほど正確な勾配に近づく値だが、非常に分散が大きい問題が知られている。例えば、$x=\{0,1\}$であり、

$$p(x=1;\theta)=999/1000$$
$$p(x=0;\theta)=1/1000$$

のような確率の時、サンプリング1000回中、平均して1回だけ、$f(0)*1000*\nabla_\theta p(0;\theta)$という非常に大きな値がサンプリングされる。この値がサンプリングされるかどうかでモンテカルロ推定の値は大きく変わり、ばらつきが大きい。この分散を下げるため、次のようなベースラインbを引いた推定量を使う。

$$E_{p(x;\theta)}[(f(x)-b)\nabla_\theta \log p(x;\theta)]$$

ベースラインbがxに依存しないような値の場合、この期待値も$\nabla_\theta J(\theta)$の不偏推定量である。なぜなら、確率分布のパラメータについての勾配の積分は常に0であるためである。

$$\begin{aligned}\int_x \nabla_\theta p(x;\theta)\,dx &= \nabla_\theta \int_x p(x;\theta)\,dx\\ &= \nabla_\theta[1]\\ &= 0\end{aligned}$$

よって

$$\begin{aligned}E_{p(x;\theta)}[b\nabla_\theta \log p(x;\theta)] &= b\int_x \nabla_\theta p(x;\theta)\,dx\\ &= b\cdot 0\\ &= 0\end{aligned}$$

勾配推定量分散が最小になるようなベースラインを厳密に求めるのは大変だが、様々な推定手法が提案されている。例えば、$E_p[(f(x)-b)^2]$が最小になるようなbをオンライン推定で求め、このbを使って勾配を推定する[1]。この拡張として、ベースラインを平均場近似をして求める手法[2]や、複数サンプルを使ったモンテカルロ推定の他のサンプルの平均をベースラインとして使う手法[3]などが提案されている。

強化学習の方策勾配の場合、ベースラインとして状態価値$V(s)$を使うことができる。状態価値は行動に依存しない。この場合、方策勾配法は$A(s,a):=Q(s,a)-V(s)$と定義されるアドバンテージ価値を使って方策を更新する。アドバンテージ価値が正であれば、その行動をとることで平均的な場合より収益を上げられるので、その行動を今よりも取るように更新し、負であればその行動を取らないように更新する。

また、xがベクトルなど構造を持っている場合、その構造に応じて推定に重要な部分のみを解析的に解いたり、サンプリングをするなどして分散を減らす手法も提案されている[4]。

変数変換トリック

もう1つが変数変換トリックである。この変数変換トリックは簡単に使うことができるだけでなく、性能も高いため現在は広く利用されている。変数変換トリックはxが連続量であり、$p(x;\theta)$が微分可能な場合に使える手法である。ϵを$q(\epsilon)$を確率密度関数とする確率変数として、xが

$$x=t(\theta,\epsilon)$$

という決定的な関数tで表されるとする。例えば、確率密度関数が正規分布$N(\mu(\theta),\sigma(\theta))$の場合、

$$t(\theta,\epsilon)=\mu(\theta)+\sigma(\theta)\epsilon$$

のような決定的な関数で表される。この場合、勾配は次のように求められる。

$$\begin{aligned}
&=\nabla_\theta\int p(x;\theta)f(x)\,dx\\
&=\nabla_\theta\int q(\epsilon)f(t(\theta,\epsilon))\,d\epsilon\ (*)\\
&=E_{q(\epsilon)}[\nabla_\theta f(t(\theta,\epsilon))]
\end{aligned}$$

ただし(*)では、$p(x;\theta)\,dx=q(\epsilon)\,d\epsilon$を使った。この場合、尤度比法とは違って$f$の入力に対する勾配

$$\frac{\partial f}{\partial x}$$

の情報を使って、勾配を求められるため、変数変換トリック後の勾配推定は分散を大きく下げることができる。また、実装も変数にノイズを加えるだけで後は誤差逆伝播法で推定できるので簡単である。

変数変換トリックは、変分法による生成モデル（変分自己符号化器）[3]、ニューラルネットワークのパラメータのベイズ推定[4]など広く使われており、強化学習でも環境や報酬を微分可能なモデルで表すモデルベース強化学習などで使われ始めている。

今後の課題

これらの手法を組み合わせることで、確率変数を含む式の最大化問題は解けるようになったが、まだ確率変数が高次元の離散変数の場合や、確率密度関数が微分不可能な場合が問題となっている。また、強化学習の探索と活用のジレンマと同様に、対象関数が複数のピークを持つ場合、勾配情報だけではうまく解けず、どのように探索するかが課題となっている。

---

1) A. Minh, et al.,"Neural Variational Inference and Learning in Belief Networks," ICML 2014.
2) S. Gu, et al.,"MuProp: Unbiased Back Propagation for Stochastic Neural Networks," ICLR 2016.
3) A. Minh, et al.,"Variational inference for Monte Carlo objectics," ICML 2016.
4) M. Tisias, et al.,"Local Expectation Gradient for Black Box Variational Inference," NIPS 2015.
5) D. P. Kingma, et al.,"Auto-Encoding Variational Bayes," ICLR 2014.
6) C. Blundell, et al.,"Weight Uncertainty in Neural Networks," ICML 2015.

（2016年10月号掲載の記事に加筆）

3-5

誤差逆伝播法を使わない学習手法 Feedback Alignment、Synthetic Gradient、Target Prop

現在の深層学習は、誤差逆伝播法を利用して学習している。これは、微分可能な計算要素を組み合わせて入力から出力を求める計算グラフ（順計算）を構築し、その計算グラフ上で誤差（目的関数に対する各状態についての勾配）を順計算とは逆方向に伝播させることで、学習に必要な目的関数に対する各パラメータについての勾配を正確にかつ効率的に推定できる方法である。

誤差逆伝播法は、当初想定されていたよりも非常に多くの問題を解けることが示されており、学習のエンジンとして大きく発展している。

誤差逆伝播法が抱える4つの問題

一方で、誤差逆伝播法はいくつかの問題があることが分かっている。

1つ目の問題は、微分可能でない、または微分がほとんどの場所で0になる計算要素を含む場合、誤差は不定、発散または減衰するため、誤差逆伝播法が使えない。

例えば、活性化関数に入力が負であれば0、それ以外であれば1を返すような閾値関数を利用した場合、微分値は0以外の場所では0、0では不定となり、誤差逆伝播法が使えない。また、注意機構の中でも、注意した部分の情報のみが非0の重みを持つハード注意機構は計算効率や汎化性能の面で有望視されているが、先の閾値関数と同様に殆どの位置で微分が消失し誤差逆伝播法が使えない。

2つ目の問題は、計算グラフが確率層を含む場合である。この場合、誤差逆伝播法は直接使えないため、変数変換トリックや尤度比法を使って勾配の推定値を求める問題に帰着させ学習する。しかし、複数の確率層を含む場合は勾配の推定値の分散が大きくなってしまい、現実的なサンプル数で学習できないという問題がある。

3つ目の問題は、誤差の逆伝播時に順計算時に使った重み行列の転置行列を使う部分である。最近ではハードウエアを使って行列計算を効率的に実現する例がいくつか出ている。例えば、アナログ回路を使って効率的に重み付き和を実現できることが分かっている。しかしこの場合、重み行列の転置を実現することが難しく、誤差を伝播させることができない。

4つ目の問題は、ニューラルネットの層の数が数十層から数百層と深くなるにつれ、各層の計算はその後の層の計算が終わるまで待たなければならない点である。計算機におけるパイプライン処理と同様に、前の誤差逆伝播法のステップが終わる前に次の誤差逆伝播法を走らせた場合、次のステップの計算では更新されていない古い（stale）パラメータを基に勾配を求めることになる。その結果、得られる勾配は（更新されたパラメータを使って求められた勾配と比べて）不正確な勾配となり、収束速度が遅くなったり、良い解が得られなかったりといった問題がある。

こうした問題を解決するために、誤差逆伝播法ではない学習手法が提案されている。以降で、その3つの手法を紹介しよう。

固定のランダム行列を使う

Feedback Alignment[1-2]は誤差を逆伝播する際に重み行列の転置行列を使うのではなく、固定のランダム行列を使う。この場合、最初に求められる更新方向は、勾配のような目標値を最も急に下げる方向ではない。しかし、不思議なことに、この学習を進めていくと、真の勾配と更新方向が一致し始め、あたかもこのランダム行列が転置行列であるかのように表現が変わっていく。つまり学習する方法を学習する（Learning to learn）。これは前述した誤差逆伝播法の問題の3つ目を解決する。専用ハードウエアの学習則や脳内の学習則の候補として注目されている。

さらに、真の誤差を入力に近い層に直接渡してから、それを順計算と同じように伝播させても学習できることが分かっている[2]。この場合、途中で微分不可能な計算要素があったとしても、それより前に誤差を流して学習することができ、1つ目の問題を解決できる。

各層の誤差を推定する専用モデルを用意

Synthetic Gradient[3]は得られるであろう勾配を各層毎に推定するモデルを学習する手法である。各層毎に順方向の

計算とは別に誤差を推定するモデルを用意し、各層の状態と補足情報（例えばラベル）から、その層に到達するであろう誤差を推定する。いうならば各推定モデルは、「この出力だったらこのように間違えそうだ」ということを推定する。各推定モデルは真の勾配との二乗誤差を最小化するように学習する。

各モデルにはニューラルネットワークのような強力なモデルではなく、線形モデルなどが適していることが分かっている。なぜそうなのかはまだよく分かっていない。各推定モデルが真の誤差を近似できる能力はないことから、各推定モデルは現時点での目標関数の近傍のみを簡単なモデルで近似しているのではないかと考えられる。

また、推定モデルの学習時には、真の勾配ではなく1つ上の層の勾配の推定値を目標として学習することもできる。この場合、推定値を基に推定するブートストラップ法であり、強化学習のTD（Temporal Difference）誤差の伝播のように出力層から順に推定が正しくなっていく。

ブートストラップ法であるため学習が不安定という問題がある。しかし、誤差逆伝播法の1つ目から4つ目の問題を全て解決するだけでなく、計算の周期が異なるような学習システム同士を連携させることもできることから、非常に有望な手法である。

誤差ではなく目標を伝播

Target Prop[1,5]は誤差ではなく目標を伝播させていく手法である。隣接する2つの層の状態がそれぞれh_i、h_{i+1}だとする。この2つの層の間の計算をfとすると、

$$h_{i+1}=f(h_i)$$

という関係にある。上の層で今の状態h_{i+1}よりも目的関数の値を下げられる目標値$\hat{h}_{i+1}$が分かっているとする。このとき、下の層の目標は$\hat{h}_i=f(\hat{h}_i)$となるような$\hat{h}_i$となる。そうすれば目的関数の値を下げられるからである。

この目標値は最初、出力層で決定され、次に1つ下の層、次に2つ下の層と順番に伝播されていく。このとき、各層で上の層が目標となるような値を求める必要がある。

Difference Target Prop[5]では、各層毎に自己符号化器を学習し、（非線形関数を含む）順方向の計算$h_{i+1}=f(h_i)$と同時にその逆関数を近似する$h_i=g(h_{i+1})$を学習する。各層ではこの目標値を出力するようにパラメータを更新すると同時に、gを使って目標値を下の層に伝播させていく。

Target Propは、誤差逆伝播法の問題の1つ目から3つ目を解決している。一方で、常に自己符号化器が作れるのか、またそれが不完全な場合にどのような学習が実現されるのかについてはまだ解明されていない。

誤差逆伝播法との組み合わせも可能

今回紹介した手法は、誤差逆伝播法を組み合わせることも可能である。例えば、Synthetic Gradientを大きなブロック毎に行い、各ブロック内では誤差逆伝播法を使うことで、並列化と正確な学習の両方を実現するといったことも考えられる。

---

1) T. Lillicrap et al.,"Random synaptic feedback weights support error backpropagation for deep learning,"Nature Communication 7, 2016.
2) A. Nokland et al.,"Direct Feedback Alignment Provides Learning in Deep Neural Networks," https://arxiv.org/abs/1609.01596
3) M. Jaderberg et al.,"Decoupled Neural Interfaces using Synthetic Gradients," https://arxiv.org/abs/1608.05343
4) Y. Bengio,"How Auto-Encoders Could Provide Credit Assignment in Deep Networks via Target Propagation,"https://arxiv.org/abs/1407.7906
5) D. Lee et al.,"Difference Target Propagation,"https://arxiv.org/abs/1412.7525

（2017年3月号掲載の記事に加筆）

3-6

継続学習：過去の学習結果を忘れずに新しいタスクを学習できるか

現在、深層学習を中心とした機械学習は人間に匹敵するような精度で様々なタスクを解く能力を学習で獲得できることが示されている。一方で、学習の際、人間と比べて膨大な量の学習データが必要なことが大きな問題となっている。

例えば、プロのゲーマーが新しいゲームをプレーする場合、最初は戸惑うが数分も経つとミス無く操作できるようになり、すぐにクリアすることができる。これに対し、現在のDQN（Deep Q-Network）をはじめとした深層強化学習は、ある程度のレベルに達するためだけにも数万～数十万回の経験を必要とする。

新しいタスクに機械学習をどう対応させるか

人の驚異的な学習効率はいかにして達成されているのだろうか。一説には、人間は新しいタスクを学習する際に既存のタスクの学習結果（の一部分）をうまく再利用できているからではないかと考えられている。

これを最も素直に実現する方法がマルチタスク学習である。マルチタスク学習では複数のタスクの学習データを前もって用意し、それらを同時に利用して学習する。タスク間で共通した部分問題に対し共通した計算を利用できるように、ニューラルネットは入力から途中までは共有して、最後にタスクごとに分岐するようなモデルを使う。また、多クラス分類も一種のマルチタスク学習と考えられる。これも途中まで同じネットワークを共有し、最後に分岐してsoftmaxで確率を計算しているためである。

究極的には長い期間にわたって様々なタスクを学習し続ける必要が出てくる。このとき、新しいタスクが登場したり、または新しい学習データが手に入ったりする度に、最初からモデルを学習するのは非現実的であり、現在の学習済みのモデルを改良し続けていくことが必要となる。これを継続学習（continual learning）、または生涯学習（lifelong machine learning）と呼ぶ。

人は継続学習をしており、マルチタスク学習のように複数の学習を同時には行わない。ある期間は自転車の乗り方を練習し、それが終ったら次は歴史の勉強をするといった具合である。自転車の乗り方と歴史の勉強を同時にするということはしない（睡眠中の夢で過去の経験を再生しているという説はある）。新しいタスクを学習しても、過去の学習結果は忘れず問題なく学習することができる。

一方、現在の機械学習器で継続学習を行うと、新しいタスクの学習中に昔のタスクの学習結果を忘れてしまう現象が起こる。こうした現象を専門用語で致命的忘却（catastrophic forgetting）と呼ぶ。また、今の学習が過去の学習結果に干渉し、悪影響をおよぼす現象を致命的干渉（catastrophic interference）と呼ぶ。

タスクごとにパラメータの重要度を算出

コンピュータは過去の経験を学習データセットやリプレイバッファとして蓄積した上で、同時に学習するような問題設定にすれば何とか複数のタスクを学習できる。しかし、人が日々こなしているような数千、数万のタスクを学習するようになってくると致命的忘却の問題が発生してくる。また、今後IoTの普及で様々なデバイスが経験を積んで学習データを集められるようになった時、膨大な量の学習データを全て記憶しておくのが困難になると考えられる。

致命的忘却を防ぎ、継続学習を実現する手法はこれまでいくつか提案されているが、ここでは初期の代表的な研究成果をいくつか紹介しよう。

英DeepMind社により提案された方法[1]では、各タスクを学習し終えた後に、そのタスクにおける各パラメータの重要度を求めておく。そして、新しいタスクを学習する時には、過去のタスクで重要だった重みはできるだけ変わらないように制約した上で新しいタスクを学習する。この学習は、パラメータの固定（consolidation）度合いがタスクに対する重要性で柔軟（elastic）に変わることから、Elastic Weight Consolidation（EWC）と名付けられている。

どの重みが重要であるかどうかは、そのタスクにおけるFisher情報行列の対角項を使って判断する。Fisher情報行列はモデルの対数尤度の各パラメータについての勾配の2次モーメントである。これが大きいということはそのパラメータ

を動かすとモデル（予測分布）が大きく変わる重要なパラメータであることを意味し、小さい場合は重要でないことを意味する。

実験では、家庭用ゲーム機「Atari 2600」の各ゲームを順に学習していった時に、何も制約をかけない通常の学習と、パラメータの重要度を求めず一律に昔のパラメータに近いようにする学習（L2）、そしてEWCを比較した。実験結果では、通常の学習では新しいタスクを学習すると従来タスクの精度が落ちてしまう致命的忘却が起きることが確認された。また、L2の場合は学習時の制約が厳しすぎて新しいタスクの精度を十分上げることができなかった。EWCは新しいタスクを十分な精度がでるように学習しつつ、従来タスクの精度も維持できていることが示された。

局所的な情報から重要度を推定する手法も

一方でEWCでは、各タスクを終えた後にFisher行列の対角項を求めるというステップが必要である。これは現実世界のように、様々なタスクの学習が部分的に次々とやってくるような設定では不向きである。

そこでStanford Universityの研究グループは、各パラメータの重要度をオンラインで推定する手法を提案した[2]。これらの重要度を求める時にはモデル全体の情報は必要とせず、各パラメータの局所的な情報のみが必要なので、脳内でも実現可能な手法と主張している。

この重要度の計算方法の導出は省略するが、各パラメータの重要度は学習時の各ステップでの目的関数に対する偏微分と、更新幅の積の総和としてオンラインで推定することができる。そして、新しいタスクの学習時には、これまでのタスクでの重要度が大きいパラメータについては更新されにくくすることで致命的忘却を回避する。

実際の脳のシナプスは現在のニューラルネットワークで使われているスカラー値のようなパラメータだけではなく、様々な情報を保持し、シナプスの可塑性を動的に変えて継続学習を実現していると考えられている。ニューラルネットワークの世界でも、こうしたアイデアを基に継続学習を実現する様々なアイデアが実現されていくだろう。

1) J. Kirkpatrick et al., "Overcoming catastrophic forgetting in neural networks," PNAS 2017.
2) F. Zenke et. al., "Improved multitask learning through synaptic intelligence," ICML 2017.

（2017年5月号掲載の記事に加筆）

3-7 予測学習：Predictive Learning

深層学習は教師あり学習において大きな成功を収めている。一方、教師データを必要としない教師なし学習はまだ発展途上である。

カナダUniversity of Toronto教授のGeoffrey Hinton氏は、「脳のシナプスは10^{14}個あるが、人は10^9秒しか生きられない。サンプル数よりパラメータ数の方がずっと多いことになる。(これらのシナプスの重みを決定するためには)1秒当たり10^5個の制約が必要となり、多くの教師なし学習をしているとの考えに行き着く。」[1]と述べている。また、米New York University教授のYann LeCun氏は「知能をケーキに例えるならば、教師なし学習はケーキ本体であり、教師あり学習はケーキの飾り、強化学習はケーキ上のサクランボぐらいである。私達はケーキの飾りやサクランボの作り方は分かってきたがケーキ本体の作り方は分かっていない。」[2]と述べている。

学習データがタダでいくらでも手に入る

教師なし学習は様々な手法が提案されているが、この中でも今持っている情報から未来を予測する予測学習が重要になると考えられる。未来の予測タスクであれば常に時間遅れで予測の正解が得られるため、予測と実際の差を直接使って学習することができる。そういう意味では予測学習は学習データがタダでいくらでも手に入る教師あり学習ということもできる。予測学習の重要性は以前より指摘されており、例えば、Jeff Hawkins氏の「On Intelligence(和書名『考える脳 考えるコンピュータ』)」では学習の本質は予測にあると述べられている。

この予測タスクでは、予測できる能力よりも予測タスクを解けるように副作用的に獲得した特徴が重要となる。近年、特徴獲得において自己符号化器(Autoencoder)を使うのはうまくいかず、予測学習をした結果得られた特徴を利用した方が成功していると指摘されている[3]。

自己符号化器では、入力を符号化器で次元数の低い潜在表現に変換し、それを復号器で元の入力に戻せるように学習することで、潜在表現上で本質的な情報を抽出することを試みる。しかし、自己符号化器の問題設定は、潜在表現においてどの情報を捨ててどの情報を残せば良いか一意に決まらない不良設定問題である。例えば、画像において前に写っている人と背景の山のどちらの情報を残すことが重要かは決められない。

予測をするには物体の位置や状態の認識が必要

一方、未来の予測タスクを解けるようにするには物体の位置や状態などの高度な環境認識ができている必要がある。例えば、物体が加速しながら動いているのを見ている状況で未来を予測するためには、物体の位置、速度や加速度が求まっている必要があり、どの領域が1つの物体として一体なのかというセグメンテーションの問題も解く必要がある。また、物体が何であるのかという情報も推定する必要がある。その物体が車であれば直線運動するであろうし、シャボン玉であれば割れるだろうことが予測できる。このように予測タスクを解くことで有用な特徴抽出、つまり高度な認識ができるようになる。

予測タスクを解くことで多くの問題で役に立つ表現を獲得できることは独立成分分析からもいえる。独立成分分析では、混合関数が非線形の場合、独立成分を一意に決定できない。一方で対象データが時系列データであり、各時刻のデータを生成する独立成分が前の時刻の独立成分に依存している場合、与えられたデータ対が時間的に連続する2つのデータか、ランダムなデータかを区別できるような関数は非線形の独立成分分析を達成できることが知られている。予測タスクが解けるようになる、またはその部分問題が解けるようになることで情報を構成する因子を同定することができるようになるのである。

こうした予測学習は脳の中でも起こっていると考えられているが、どのように起こっているかはよく分かっていない。ここではその中で興味深い仮説[4]について紹介する。

大脳新皮質の5-6層で予測処理との仮説

視覚情報が処理されている大脳新皮質は6層構造から成

る。この6層はそれぞれ異なった構造と役割を持ち、4層目が視床などから入力を受取り、その処理は表面側の1〜3層目と奥側の5〜6層目に2つに分かれる。この仮説では表面側の1〜3層目が実際の視覚の処理を担い、5〜6層目が予測処理を担っていると考えている。そして5〜6層で作られた予測結果は視床枕と呼ばれる領域に伝搬され、そこで予測結果が映し出される。

この視床枕ではすぐ後に実際の入力が伝搬され、予測との比較がなされる。この予測と実際の結果はα波に従って100msごとに切り替わるとされる。つまり1秒当たり10回予測していることになる。

視床枕は大脳新皮質のほぼ全ての視覚に携われる部位と双方向で結合しているため、発生した予測誤差をこれらの領域に伝搬可能である。各シナプスについて実際の入力の時のシナプス前後のニューロンの共起頻度と予測の時の共起頻度の差が、予測誤差を小さくするような方向（勾配）となることが分かっている。これにより、予測誤差を小さくするようなシナプスの更新は全てローカルな情報で行われ、過去の記憶を必要としない。そのため、この学習は脳内で実現可能な更新則と考えられる。

また、この予測学習は、始めは位置や移動の学習（Where経路）、次に物体の属性の学習（What経路）、最後にこれらの組み合わせ（Where×What経路）の学習と進む。目は止まっている目標を見続けている際にもマイクロサッカード（固視微動の一種）と呼ばれる微少な不随意運動を常に起こしている。眼球は自分で動かしているため、運動が分かっている中で見え方がどう変わるのかの予測問題を解くことになる。これによりどこに写っているのか、どう変わるのか（Where経路）が学習される。次に、この位置や移動が予測できるようになると追跡できるようになり、物体で何が写っているのか（What経路）が学習できるようになる。最後はこれら2つを組み合わせた予測（Where×What経路）が学習される。

この仮説に従えば、1歳の幼児であれば生まれてから10時間/日×3600秒/時×365日×10Hz=1.3億回の予測学習をしていると考えられる。これは現在のディープラーニングで使われる100万枚から成る訓練画像データ、ImageNetの100倍の大きさになる。さらに、分類タスク（ImageNetでは1000クラスなので10bit）ではなく予測タスク（中心視野では数百万画素）であるので膨大な情報量のフィードバックを得て学習していることになる。

動画など高次元データの予測学習はまだこれから

このように予測学習は重要であるが、まだコンピュータ上では動画といった高次元データに対する予測学習は成功していない。予測学習では膨大な量のフィードバックが得られるとはいえ、入力も出力も高次元であり学習が難しい問題である。Where経路、What経路のような学習と同じように、少しずつ難しい問題を解くカリキュラム学習が必要になると考えられる。また、どれに注目して予測するかといった注目機構も重要になると考えられる。さらに利用している学習データ数が足りないという可能性がある。先ほどの試算のように、現在の教師あり学習の数百倍の規模でようやく成果が出て来るのかもしれない。

1) https://www.reddit.com/r/MachineLearning/comments/2lmo0l/ama_geoffrey_hinton/clyjogf/
2) https://drive.google.com/file/d/0BxKBnD5y2M8NREZod0tVdW5FLTQ/view, Yann LeCun, NIPS 2016, Keynote.
3) https://www.reddit.com/r/MachineLearning/comments/6z51xb/we_are_the_google_brain_team_wed_love_to_answer/dmycc65/
4) R. O'Reilly et al., "Deep Predictive Learning: A Comprehensive Model of Three Visual Streams," https://arxiv.org/abs/1709.04654

（2017年11月号掲載の記事に加筆）

3-8 進化戦略：Evolution Strategy

現在の多くの学習はパラメータθを入力とした目的関数$F(\theta)$の最適化問題（以下、最小化問題とする）を解くことで実現される。最適化対象のパラメータ空間が高次元である場合、パラメータをランダムな方向に摂動（わずかに動かす）させた時に目的関数の値が改善される可能性は低い。これは、目的関数の形は多くの方向が崖のように切り立っている谷のような形をしており、特定の方向に沿った時のみ目的関数を改善できるようになっているためである。

目的関数を改善できる方向を探すためには、勾配情報を使うのが一般的である。目的関数のパラメータについての勾配$v=\nabla F(\theta)/\nabla\theta$は、目的関数の値が最も急激に増加する方向であり（負の勾配方向は最も急激に減少する方向）、現在のパラメータをどの方向に更新すれば目的関数を最も改善できるか正確な情報を与えてくれる。ニューラルネットワークにおける誤差逆伝播法のように、多くの問題では目的関数のパラメータについての勾配を効率的に求めることができ、その勾配情報に従ってパラメータを更新する勾配降下法が広く使われている。

一方、勾配を効率的に求めることができない問題も多く存在する。例えばニューラルネットワークで離散変数を使ったり、階段状の関数（例：$f(x)=[x]$）のように微分が0や不定になってしまうような非線形関数を利用する場合、勾配は0や不定になってしまい、改善する方向を与えてくれない。また、目的関数の一部が不明な場合も効率的に勾配を求められない。例えば強化学習は、環境（現在の状態と行動から、報酬と次の状態を返す関数）はブラックボックスであり、誤差逆伝播法を使って勾配が計算できない。これに加え、学習時のハイパーパラメータ（学習率、モーメント、ユニット数、層数など）や、パラメータの初期値、ネットワークアーキテクチャそのものといった最適化全体を関数とみなした時、これらハイパーパラメータについての勾配も求めたいが、これも誤差逆伝播できず効率的に勾配を計算できない。

こうした問題を解決するため、進化戦略（ES：evolution strategies）や遺伝的アルゴリズムを利用することができる。これらのアルゴリズムは誤差逆伝播法を使った手法より勾配推定の効率は悪いが、目的関数の値さえ得られれば最適化できるという大きな特徴がある。ここでは進化戦略について紹介していく。

進化戦略では、現在のパラメータが確率分布$p_\phi(\theta)$で表されていると考え、この確率分布を更新していくことで学習する。学習ではこの確率分布から複数のパラメータをサンプリングし、それらの目的関数の値を調べ、その情報を元に勾配の推定値を計算する。

ここで、最適化対象パラメータθが平均ϕ、分散$\alpha^2 I$の正規分布$p_\phi(\theta)=N(\phi,\alpha^2 I)$に従って生成されているとする。ここで、$\alpha$は固定のパラメータ、$I$は単位行列である。このとき、この確率分布に従ってパラメータをサンプリングした時の目的関数$F(\theta)$の期待値は、$\mathrm{E}_{\theta\sim p_\phi}F(\theta)$である。

この目的関数のパラメータ平均ϕについての勾配を求めてみよう。ϕは期待値をとっている確率分布のパラメータであり、直接この確率分布を微分して勾配を計算してしまうと、この確率分布からサンプリングをしてモンテカルロ推定ができなくなってしまう。そこで、REINFORCE（第3章 第4節を参照）を使って、勾配計算時にも$p_\phi(\theta)$が残るようにする。

$$\nabla_\phi \mathrm{E}_{\theta\sim p_\phi}F(\theta)=\mathrm{E}_{\theta\sim p_\phi}\{F(\theta)\nabla_\phi \log p_\phi(\theta)\} \qquad (1)$$

ここで、式中の勾配計算の部分に注目すると、

$$\nabla_\phi \log p_\phi(\theta)=\nabla_\phi \log \frac{1}{Z(\sigma)}\exp\left(-\frac{(\theta-\phi)^2}{2\sigma^2}\right)$$

$$=\nabla_\phi -\frac{(\theta-\phi)^2}{2\sigma^2}=\frac{1}{\sigma^2}(\theta-\phi)$$

となる。これを元の式(1)に代入すると、

$$\mathrm{E}_{\theta\sim p_\phi}\left\{F(\theta)\frac{1}{\sigma^2}(\theta-\phi)\right\}=\frac{1}{\sigma}\mathrm{E}_{\theta\sim p_\phi}\{F(\theta)(\theta-\phi)/\sigma\}$$

$$=\frac{1}{\sigma}\mathrm{E}_{\epsilon\sim N(0,I)}\{F(\phi+\epsilon\sigma)\epsilon\}$$

となる。ここで、$\epsilon=\theta-\phi/\sigma$と変数変換をした。つまり、ESの最終的な式では現在のパラメータϕに対し、摂動$\epsilon\sigma$を加えた上で目的関数の値を調べ、その結果に従って重み付けをした上でϵ方向に動くという更新則のようにみなせる。

ESはその後CMA (covariance matrix adaptation) -ESなどさまざまな改善版が提唱されている。CMA-ESは連続行動空間の強化学習で方策学習と比較し、同等の性能が達成できると報告されている[1]。

しかし、ESはミニバッチ学習による並列化と相性が悪いことが報告されている。通常ミニバッチ学習では、複数の学習データに対して同じパラメータのモデルを使って勾配推定することで勾配推定の分散を減らしている。それに対し、ESをミニバッチ学習にそのまま適用すると、各学習データ（グループ）毎に異なるパラメータのモデルを使って勾配推定する必要があり、並列計算できない。そのため、例えばGPUなど並列計算を使って高速化することができない。

FlipOut[2]は、この問題を解決しミニバッチ学習時に各サンプルが摂動によって異なるパラメータを持った場合でも勾配計算を効率的に実現する手法である。

ここからは、参考文献[2]に従ってパラメータを行列Wとし、目的関数は$W^T x$を含むような関数とする。ニューラルネットワークを含む多くのモデルがこのような関数形をとる。また、摂動分布$W \sim p(w)$からのサンプリングが平均$\overline{W}$と、平均に対する摂動ΔWの2つから構成され、$W = \overline{W} + \Delta W$と表されるとする。

ここで摂動分布に対し仮定を2つ置く。1つ目は各成分毎に摂動は独立である、2つ目は摂動は原点について対称であるとする。例えば、摂動分布がガウシアンであったり、DropOut ($\overline{W} = \frac{W}{2}, \Delta W = \pm \frac{W}{2}$) である場合はこの仮定を満たす。次に、$E$を各成分が$\{+1, -1\}$から一様にサンプリングされたような行列とする。摂動分布をp_0とし上記の仮定を満たすとする。このとき、$\Delta \overline{W} \sim p_0$の$\Delta \overline{W}$の分布と、$\Delta W = \Delta \overline{W} \circ E$ (ただし$\circ$は要素毎の積) の分布は一致する。また、ΔWを使って求めた勾配は$\Delta \overline{W}$を使って求めた勾配と一致する。

この事実を利用し、$\Delta W_n = \Delta \overline{W} \circ r_n s_n^T$とする。ただし、$r_n$、$s_n$はそれぞれ各成分が一様に$+1$と$-1$からサンプリングされたようなベクトルであり、$r_n s_n^T$はrankが1の行列である。この上の事実より$\Delta W_n$の分布は$W$と一致する。各$\Delta W_n$はお互い独立ではないが、$r_n s_n^T$によって各成分の符号がランダムにシャッフルされており、お互いの相関が小さくなる。サンプリングをする場合には、サンプルが独立でなくてもサンプル間の相関が小さい場合に推定の分散を減らせることが分かっている。

このとき、n番目のサンプルに対するパラメータとの内積は

$$W^T x_n = (\overline{W} + \Delta W \cdot r_n s_n^T)^T x_n$$

$$= \overline{W}^T x_n + \Delta W^T (x_n \cdot s_n) \cdot r_n$$

この計算は各サンプル毎に独立に行うことができる。各行がr_nからなる行列R、s_nからなる行列Sを用いて上記の計算は、$X\overline{W} + (X \cdot S)\Delta \overline{W} \cdot R$と行列積のみで記述できる。行列積は並列化可能であり、特にGPUなど近年のハードウェアが高速に処理することができる。また、この式はRとSは$\overline{W}$、$\Delta \overline{W}$に対し独立なので、上記の計算に基づき誤差逆伝播することができる。

このFlipOutを進化戦略に適用した結果、並列に学習できGPUによる高速な並列計算を実現し、CPUによる学習より数倍から数十倍の高速化を達成している。FlipOut自身は進化戦略以外にも適用でき、ベイズニューラルネットワークの変分法による最適化、DropConnect正則化などでも高速化、または分散が減ることによる性能向上を達成できている。

コンピュータは本質的に並列計算が得意であり逐次的な更新を必要とする現在の学習手法とは異なる学習手法が今後生まれてくると考えられる。

---

1) T. Salimans et al., "Evolution Strategies as a Scalable Alternative to Reinforcement Learning," https://arxiv.org/abs/1703.03864
2) Y. Wen and et al.,"Flipout: Efficient Pseudo-Independent Weight Perturbations on Mini-Batches," ICLR 2018.

(2018年5月号掲載の記事に加筆)

3-9

メタ学習：学習の仕方を学習する MAMLやNeural Process

今の機械学習の多くは何も学習していない状態から、学習データが与えられ学習していく。この場合、全てを一から学んでいく必要があるため多くの学習データを必要とする。一方で人や動物はそれほど多くの学習データを必要としない。これは過去に経験したり、学習した結果を再利用し、必要な差分だけを学習するためだ。

高データ効率な学習の実現

例えば、自転車の乗り方を学習する際には、立ったり、歩いたり、階段を登ったりするのに必要なスキルが再利用されており、数十回程度の練習で誰でも自転車に乗れるようになる。これは数千から数十万の試行錯誤が必要な現在の強化学習とは大きな違いである。

複数のタスクの学習結果や学習過程を利用し新しいタスクの学習効率を上げるような手法をメタ学習という。“学習の仕方を学習する（Learning to Learn）”ような学習手法といってもよいだろう。

例えば、ImageNetのような大きなデータセットで学習された画像認識モデルを特定のタスク向けの学習データを使ってfinetune（微調整）するテクニックは広く使われているが、これもメタ学習といっていいだろう。既にImageNetの学習済みモデルは基本的な物体の認識はできており、残りの差分だけを学習するためである。今回はメタ学習の中で代表的な2つの手法を紹介する。

MAML：良い初期値を学習する

1つ目の手法はMAML（Model-Agnostic Meta-Learning、マムル）[1]と呼ばれる手法である。メタ学習の問題設定として学習時には複数タスクの学習を行い、テスト時には新しいタスクの学習を少量の学習データで実現することを考える。それぞれのタスク $\mathcal{T}_i$ ごとに学習データ $\{(x_i^{(1)}, y_i^{(1)}), (x_i^{(2)}, y_i^{(2)}), \ldots, \}$ が与えられる。この際、データは必ずしもi.i.dでなくてもよく、強化学習のような前回の結果に依存して次の入力が決まってもよい。

MAMLは各タスクの学習の際に、多くの機械学習と同様に初期パラメータ θ_0 から始め、タスクの目的関数（損失関数）$\mathcal{L}_{\mathcal{T}_i}$ を最小化するようにパラメータをSGD（確率的勾配降下法）で更新していく。学習データが複数ある場合はこのSGDを複数回繰り返す。

$$\theta_i' := \theta_0 - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$$

そして、MAMLはメタ学習として、更新後の目的関数の値の和が小さくなるように初期パラメータを決定することを考える。

$$\min_{\theta_0} = \sum_i \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'}) = \sum_i \mathcal{L}_{\mathcal{T}_i}(f_{\theta_0 - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)})$$

つまり、各タスクで今の初期値からSGDでの更新後にうまくいくような良い初期値を求める。なお説明では1回のSGD後のパラメータとして説明しているが定数回の更新後のパラメータでも良い。

掴む・歩く・避けるなどのスキルを事前にメタ学習

MAMLは学習の仕方を変えるわけではなく、初期パラメータだけを更新し、後は通常の学習をそれぞれのタスクで行うだけである。テスト時にも初期パラメータから同様に定数回の更新を行い学習する。一見すると単純に見えるこのMAMLがメタ学習としてうまくいくことが分かっている。

例えば、各タスクがロボットの様々なタスクであり、パラメータがロボットを制御する方策のパラメータだとしよう。この時、MAMLは各タスクがうまくいくような、掴む、歩く、避けるといった必要な基本スキルがすでに身についたようなものを初期パラメータとして獲得するだろう。そして、各タスク向けに少しだけfinetuneすれば各タスクに特化したような方策を学習することができる。

MAMLの最適化では勾配の勾配を求める操作があることに注意されたい。更新後のパラメータは初期パラメータと勾配の差からなり、これをさらに初期パラメータについて勾配を取る必要がある。現在多くのディープラーニングフレームワークは計算量が数倍増加する程度で勾配の勾配を計算することが可能である。また、その後提案されたReptileは、代

わりに初期値を複数回更新後のパラメータ方向に更新しても近い性能が出せることを示している[2]。

MAMLは階層ベイズモデルの枠組みで捉えられることが分かっている[3]。各タスク固有のパラメータ ϕ_j とし、各パラメータは他のタスク固有のパラメータに影響を受けているとする。これをモデル化するために、各タスク固有のパラメータは共通のパラメータ θ に依存しているとする。全タスクの観測データを $\mathbf{X}$ とし、各タスク $\mathcal{T}_j$ のサンプルを $\mathbf{x}_j^{(1)}, \ldots, \mathbf{x}_j^{(N)}$ とする。このとき、観測データの尤度は

$$p(\mathbf{X}|\theta) = \prod_j \left(\int p(\mathbf{x}_j^{(1)}, \ldots, \mathbf{x}_j^{(N)}|\phi_j) p(\phi_j|\theta) d\theta_j \right)$$

と表される。ここで各タスク固有パラメータは周辺化消去されている。

この尤度を最大化するような θ を求めることは、データから事前分布を求めることから経験ベイズと呼ばれる。この周辺化は計算困難なので、代わりにパラメータ $\hat{\phi}_j$ を1つサンプリングし、それを利用して尤度を評価し、それを最適化する。このサンプリングとして $\hat{\phi}_j = \theta + \alpha \nabla_\theta \log p(\mathbf{x}_j^{(1)}, \ldots, \mathbf{x}_j^{(N)}|\theta)$ を利用した場合、これはMAMLと一致する。つまりMAMLは階層ベイズで各タスク固有のパラメータを点推定して近似した場合の経験ベイズと捉えられる。

Neural Process：関数のメタ学習

もう1つのメタ学習がNeural Process[4]と呼ばれる確率過程を使ったメタ学習である。確率過程は入力が s の時の出力 y の分布 $p(y|x=s)$ を学習する。問題設定としては似ているが異なるたくさんの関数をモデル化するタスクが学習時に与えられ、テスト時には少数の点から新しい関数を予測するタスクを考える。

例えば入力電圧と振動数の関係を様々な機体ごとに関数として学習しておき、新しい機体の少数の観測が与えられた時、その機体固有の関数を学習するような問題である。

確率過程のモデルとしてはガウス過程（GP: Gaussian Process）が有名だが、Neural Processはニューラルネットワークを使って学習する。複雑なモデルを学習できるだけでなく観測点数に対し線形の計算量でモデルを作ることができる。

モデルとして最初に確率的な振る舞いを表す種ベクトル z を正規分布などからサンプリングし、次に z と入力 x からニューラルネットワーク g を使って、正規分布の平均を出力する。

$$p(z, y_{1:n}|x_{1:n}) = p(z) \prod_{i=1}^{n} \mathcal{N}(y|g(z,x), \sigma^2)$$

学習の際には z を周辺化した条件付き対数尤度 $\log p(y_{1:n}|x_{1:n})$ を最大化するが、周辺化は計算困難なので代わりに符号化器 $q(z|x_{1:n}, y_{1:n})$ を用意し、対数尤度の下限（ELBO）を最大化して学習する。

$$\log p(y_{1:n}|x_{1:n})$$
$$\geq E_{q(z|x_{1:n},y_{1:n})} \left[\sum_{i=1}^{n} \log p(y_i|z, x_i) + \log \frac{p(z)}{q(z|x_{1:n}, y_{1:n})} \right]$$

Neural Processの目標は様々な関数をモデル化することなので、学習時には様々な関数を学習する。ここで符号化器の入力が可変個であることが問題となる。しかも、これらの入力は与える順序を変えても結果が変わらないことが望ましい。Neural Processでは単にそれぞれ独立に符号化器に通した後にそれらの平均ベクトルを計算している。これはGenerative Query Network（GQN）などで提案されているのと同様である。ただし、GQNの場合は符号化結果の平均ではなく合計を推論結果として利用している。可変個のサンプルから推論する手法は近年急速に進歩しており、例えばアテンションを使う手法[5]なども登場している。結果として、z はタスクの種類を表すようなベクトルとなり、これを変えることで関数の挙動が変わるようなモデルとなる。テスト時には観測データから z、つまりタスクの種類を推論し、それを使って関数をモデル化する。

このほかにもメタ学習は、パラメータの更新方法自体を学習したり、既存の学習結果を組み合わせて新しい能力を身につけるなど様々な手法が提案されている。今後、機械学習の実用化に向けて非常に重要な要素となるだろう。

1) C. Finn, et al.,"Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks," ICML 2017.
2) A. Nichol, et al., "On First-Order Meta-Learning Algorithms," https://arxiv.org/abs/1803.02999
3) E. Grant, et. al.,"Recasting Gradient-Based Meta-Learning as Hierarchical Bayes," ICLR 2018.
4) M. Garnelo, et al.,"Neural Processes," ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models, https://arxiv.org/abs/1807.01622
5) J. Lee, et al.,"Set Transformer," ICML 2019.

（2019年1月号掲載の記事に加筆）

3-10 陰関数微分：勾配計算で計算グラフをワープする

多くの機械学習は学習問題を最適化問題として定式化し、勾配降下法を使って最適化する。ニューラルネットワーク（NN）は誤差逆伝播法（後ろ向き自動微分）を使うことで複雑な関数でも勾配を効率的に計算できている。

勾配計算のもう1つの重要なアルゴリズムとして陰関数微分がある。深層学習（ディープラーニング）の分野での陰関数微分の利用は限定的であったが、最近は急速に適用範囲が広がっている。今回はこれについて紹介する。

陰関数微分とは

始めに、陰関数微分の基本的な考えを説明する。次のような二変数関数から成る方程式を考える。

$$F(x, y) = 0$$

このとき、ある条件を満たした場合、$y = \phi(x)$ が存在し、

$$F(x, \phi(x)) = 0$$

を満たせることが分かっている（陰関数定理）。つまり $F(x, y) = 0$ という方程式から $y = \phi(x)$ という関数が導出されたことになる。この $y = \phi(x)$ を陰関数と呼ぶ。方程式が単純でも、陰関数は非常に複雑になる場合があり、むしろこのような場合に陰関数は多く使われる。この方程式の両辺を x で微分すると

$$0 = \frac{\mathrm{d}}{\mathrm{d}x} F(x, \phi(x)) = F_x + F_y \frac{\mathrm{d}\phi(x)}{\mathrm{d}x}$$

となる。ここで

$$F_x = \frac{\partial}{\partial x} F(x, \phi(x))$$

$$F_y = \frac{\partial}{\partial y} F(x, \phi(x))$$

はそれぞれ F の x と y についての偏微分である。F_y は正則であることを仮定（これは陰関数定理の条件の1つである）した上でさきほどの式を変形すると

$$\frac{\mathrm{d}\phi(x)}{\mathrm{d}x} = -F_y^{-1} F_x$$

が得られる。これが陰関数の入力についての微分である。このように陰関数がどれだけ複雑な形をしていても、その微分は方程式の偏微分の組み合わせで表現することができる。

Deep Equilibrium Model

それでは実際に陰関数微分を利用したモデルとしてDeep Equilibrium Model[1]をみていく。このモデルでは入力 $\mathbf{x}$ と適当に初期化した隠れ状態 $\mathbf{z}^{[0]}$ に対し、繰り返し同じ関数を適用していくことを考える。

$$\mathbf{z}^{[i+1]} = f_\theta(\mathbf{z}^{[i]}; \mathbf{x})$$

通常のNNとは違って、常に入力が引数として与えられている（入力からスキップ接続がある）、各層のパラメータが共有されているようなモデルである。RNNで入力が時刻によらず固定の場合と考えても良い。このようなモデルとしてUniversal Transformer[2]、Trellis Network[3]などが知られている。

この関数を無限回適用した場合、f_θ が特定の条件を満たすならば、隠れ状態はある値（不動点）に収束する。

$$\lim_{i \to \infty} \mathbf{z}^{[i]} = \lim_{i \to \infty} f_\theta(\mathbf{z}^{[i]}; \mathbf{x}) \equiv f_\theta(\mathbf{z}^*; \mathbf{x}) = \mathbf{z}^*$$

これは f_θ が定義するダイナミクス上で隠れ状態が均衡（Equilibrium）状態に到達したといえる。この入力 $\mathbf{x}$ から、不動点 $\mathbf{z}^*$ を返す一連の操作を1つの関数とみなす。

さきほどの f_θ を繰り返し適用して不動点を求める操作は不動点反復法と呼ばれ、条件を満たせば線形収束する。このほかにも不動点を求める方法であれば、より高速に求められる方法がある。具体的には

$$g_\theta(\mathbf{z}; \mathbf{x}) = f_\theta(\mathbf{z}; \mathbf{x}) - \mathbf{z}$$

という関数を定義し、この関数が 0 をとるような $\mathbf{z}$ を求める問題を考え、（準）ニュートン法を適用する。

$$\mathbf{z}^{[i+1]} = \mathbf{z}^{[i]} - \alpha B g_\theta(\mathbf{z}^{[i]}; \mathbf{x})$$

ただし B は g_θ の $\mathbf{z}^{[i]}$ におけるヤコビアンの逆行列である。こ

の $g_\theta(\mathbf{z};\mathbf{x})=0$ となる解を求めるプロセス全体を $\mathbf{z}^* = \texttt{RootFind}(g_\theta;\mathbf{x})$ というブラックボックスの関数としよう。この関数は上にあるような準ニュートン法などを使うかもしれないし、不動点反復法を収束するまで繰り返したものかもしれない。

このように得られた解 $\mathbf{z}^*$ に損失関数を適用した結果を目的関数とする。

$$l(\mathbf{z}^*) = l(\texttt{RootFind}(g_\theta;\mathbf{x}))$$

学習ではこの目的関数の θ や $\mathbf{x}$ についての勾配が必要となる。ここではそれらをまとめて $\frac{\partial l}{\partial (\cdot)}$ と表し、$(\cdot)$ に θ や $\mathbf{x}$ が入ることとしよう。ここで陰関数微分を使うことで、勾配計算時に `RootFind` を使わずに勾配が求められる。

まず、$\mathbf{z}^* = f_\theta(\mathbf{z}^*;\mathbf{x})$ という条件を利用する。この両辺を $(\cdot)$ について微分すると

$$\frac{\mathrm{d}\mathbf{z}^*}{\mathrm{d}(\cdot)} = \frac{\mathrm{d}f_\theta(\mathbf{z}^*;\mathbf{x})}{\mathrm{d}(\cdot)} = \frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial(\cdot)} + \frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial \mathbf{z}^*}\frac{\mathrm{d}\mathbf{z}^*}{\mathrm{d}(\cdot)}$$

これを式移動すると次が得られる。

$$\left(I - \frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial \mathbf{z}^*}\right)\frac{\mathrm{d}\mathbf{z}^*}{\mathrm{d}(\cdot)} = \frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial(\cdot)}$$

また、$g_\theta(\mathbf{z}^*;\mathbf{x}) = f_\theta(\mathbf{z}^*;\mathbf{x}) - \mathbf{z}^*$ に対し、$\mathbf{z}^*$ について微分をとると

$$J_{g_\theta}|_{\mathbf{z}^*} = -\left(I - \frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial \mathbf{z}^*}\right)$$

が得られる。ここで $J_{g_\theta}|_{\mathbf{z}^*}$ は g_θ の $\mathbf{z}^*$ におけるヤコビアンである。これらを組み合わせると、損失関数の勾配は

$$\frac{\partial l}{\partial(\cdot)} = \frac{\partial l}{\partial \mathbf{z}^*}\frac{\mathrm{d}\mathbf{z}^*}{\mathrm{d}(\cdot)} = -\frac{\partial l}{\partial \mathbf{z}^*}\left(J_{g_\theta}^{-1}|_{\mathbf{z}^*}\right)\frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial(\cdot)}$$

と求められる。これはさきほどの陰関数微分の定理で $J_{g_\theta}^{-1}|_{\mathbf{z}^*}$ が F_y^{-1}、$\frac{\partial f_\theta(\mathbf{z}^*;\mathbf{x})}{\partial(\cdot)}$ が F_x に対応していると思ってもらってよい。

この $\frac{\partial l}{\partial \mathbf{z}^*}J_{g_\theta}^{-1}|_{\mathbf{z}^*}$ を求めるのは困難そうに見えるが、代わりに

$$\left(J_{g_\theta}^{T}|_{\mathbf{z}^*}\right)\mathbf{x}^T + \left(\frac{\partial l}{\partial \mathbf{z}^*}\right)^T = \mathbf{0}$$

という線形方程式の解として求めることができる。第一項のベクトル・ヤコビアン積 (vjp) は自動微分を使うことでヤコビアンを直接求めなくても解くことができる (この点は2019年6月号の本欄のNeural ODEの回でも説明した。また線形方程式も準ニュートン法などを用いて高速に解くことができる。Deep Equilibrium Modelはこの陰関数微分を使うことで従来モデルに比べて1/10程度のメモリ使用量で学習できている (表3-1)。

表3-1　各モデルで言語モデルを学習させた場合のネットワークサイズやメモリ容量 (文献[5]より引用)

分類	DNNモデルの名称	DNNのパラメータ数	embedding以外のモデルサイズ	モデルの性能 (test perplexity、数値が小さいほど良)	メモリ容量
その他	Generic TCN	150M	34M	45.2	—
	Gated Linear ConvNet	230M	—	37.2	—
	AWD-QRNN	159M	51M	33.0	7.1GB
	Relational Memory Core	195M	60M	31.6	—
	Transformer-XL (X-large, adaptive embed., on TPU)	257M	224M	**18.7**	12.0GB
TrellisNetとの比較	70層 TrellisNet (+ auxiliary lossなど)	180M	45M	29.2	24.7GB
	70層 TrellisNet with gradient checkpointing	180M	45M	29.2	5.2GB
	DEQ-TrellisNet (今回の手法)	180M	45M	**29.0**	**3.3GB**
Transformer-XL (medium) との比較	Transformer-XL (medium, 16層)	165M	44M	24.3	8.5GB
	DEQ-Transformer (medium, 今回の手法)	172M	43M	24.7	**2.7GB**
	Transformer-XL (medium, 18層, adaptive embed.)	110M	72M	**23.7**	9.0GB
	DEQ-Transformer (medium, adaptive embed., 今回の手法)	110M	70M	24.0	3.9GB
Transformer-XL (small) との比較	Transformer-XL (small, 4層)	139M	4.9M	35.8	4.8GB
	Transformer-XL (small, weight-tied 16層)	138M	4.5M	34.9	6.8GB
	DEQ-Transformer (small, 今回の手法)	138M	4.5M	**32.4**	**1.1GB**

メタ学習

陰関数微分はメタ学習でも利用されている[4]。メタ学習では似たようなタスクがたくさんある場合に、一部のタスクで学習し、新しいタスクをより少ない学習データで効率よく学習できるようにすることが目的である。ここでは各タスクの学習（内側ループ）と、タスク集合に対するメタ学習用のパラメータの最適化（外側ループ）の2つがある。

これまでは内側ループの最適化プロセス上で誤差逆伝播法を走らせる必要があるため、各タスクの更新回数は数回と限られていた。これに対し、各タスクの最適化プロセスは陰関数で表すことができるため、内側ループでの更新回数によらず、陰関数微分を使うことで外側ループで必要なパラメータの勾配を効率よく求めることができる。

Implicit Deep Learning

今回紹介したように、方程式を満たすような変数間の関係を利用して表現したい関数を表すニューラルネットワークを米University of California BerkeleyのLaurent El Ghaoui氏らはImplicit Deep Learning[5]と呼んでいる。このようなモデルは達成してほしい制約や目標を直接モデルに組み込むことができるメリットがある。さらに、入力から均衡点を返すという関数は、入力の変化に対し滑らかであり安定的であることが示唆されており、遷移関数を工夫することでノルムを変えず、勾配発散/消失が起きないRNNを作れることが分かっている[6]。

一方、SGDによる最適化と同様に、均衡状態に到達する経路や初期値が重要になる可能性もある。例えば、オンライン学習などでは前のパラメータに近いという制約を入れることで新しいデータに対応しつつも以前のデータを記憶することを達成できている。同様に均衡状態に収束する前にあえて止めることで前の記憶を残すことができる。究極的には入力や拘束条件が次々と変化していく環境下で均衡状態に収束する前に次の均衡状態に向かうというダイナミクスを持ったモデルを考える必要があるだろう。

1) S. Bai et al.,"Deep Equilibrium Models," NeurIPS 2019.
2) M. Dehghani et al.,"Universal Transformers," ICLR 2019.
3) S. Bai et al.,"Trellis Networks for Sequence Modeling," ICLR 2019.
4) A. Rajeswaran et al.,"Meta-Learning with Implicit Gradients," NeurIPS 2019.
5) L. E. Ghaoui et al.,"Implicit Deep Learning," https://arxiv.org/abs/1908.06315
6) A. Kag et al.,"RNNs Evolving on an Equilibrium Manifold: A Panacea for Vanishing and Exploding Gradients?,"https://arxiv.org/abs/1908.08574

（2019年11月号掲載の記事に加筆）

教師なし表現学習：異なるビュー間の相互情報量最大化

ディープラーニングはデータの適切な表現方法（または特徴関数）を学習によって獲得し、多くのタスクで人が設計した表現方法を使った場合よりも高い性能を達成できることを示してきた。こうした表現学習は何らかの教師あり学習の副産物として獲得できる。例えばImageNetの画像分類タスクで得られたモデルは画像認識の学習済みモデルとして多く使われている。

しかし、教師あり学習を使って表現学習した場合、基本的にはタスクに関係する表現しか獲得されない。例えば画像分類タスクで学習した場合、分類に不必要な色情報や物体の個数といった情報は表現に含まれない可能性が高い。そのため、その表現を物体検出や画像キャプショニングといった別のタスクに使うことが難しい。このためタスクに依存せず汎用的に使える表現を獲得する方法が求められていた。この最有力候補が教師なし学習による表現学習である。

教師なし学習であれば、より汎用的に使えるような表現を獲得できるタスクを設定でき、かつ正解データを必要としないためデータをいくらでも使える。しかし、これまで教師なし学習による表現学習は教師あり学習による表現学習に大きな差をあけられていた。それが自然言語処理ではBERT（第11章 第4節で解説）の登場によって大きく状況が変わった。BERTで獲得された表現を使って多くのタスクが解けるようになり、非常に大きな教師なしデータを使って、より良い表現を獲得できるようになってきている。

そしてこれまで教師あり学習による表現学習が標準だった画像認識でも、教師なし表現学習が大きく進化してきた。特に2018年から相互情報量最大化を目指した対比符号化（Contrastive Coding）を使った教師なし表現学習のアプローチが成果を上げ始め、2019年に入ると、教師あり表現学習に匹敵する、またはそれを超えるような性能を出すようになってきている。この教師なし表現学習の最新手法であるMoCo[4]は、教師あり表現学習を多くのタスクで初めて上回った。

生成モデルの問題点

はじめに、生成モデルに基づく表現学習の問題点について説明しよう。生成モデルの学習では符号zからデータxがどのように生成されるのか$p(x|z)$を学習し、この符号zをデータの表現として利用する。この生成モデルでは符号はデータ全てを説明する必要がある。しかしデータには、タスクに役に立たない情報も含まれており、これらの説明に多くの表現能力を割り当てる必要がある。例えば、道路の写真が入力として与えられたとしよう。このとき、その道路の両脇に立っている木の葉1つ1つの状態も生成時には全て正確に表現する必要がある。道路の画像を使ったタスクとしては、車や人の検出、道路のセグメンテーション、地図作成の方が、葉が右を向いている個数を数えるよりも可能性が高いだろう。このため、表現としては葉の1つ1つの状態は捨てたほうが良い可能性が高い。このようにデータの適切な表現では、データを正確に表現しなければならない一方で、必要のない情報をいかに取り除けるかが重要となる。

そこでデータを完璧に生成するようにして表現を学習するのではなく、データを符号化し、その符号がデータの必要な情報を十分持っているかで表現を獲得する方法が提唱されている。一種の歪みありデータ圧縮ともいえる。特にInfoMaxは符号zと入力x間の相互情報量を最大化するようにして符号、つまり表現を獲得する。入力xと符号c間の相互情報量は

$$I(x;c) = \sum_{x,c} p(x,c) \log \frac{p(x|c)}{p(x)}$$

として定義される。カナダMicrosoft Research MontréalのR Devon Hjelm氏らはInfoMaxをディープラーニングを使った非線形関数を使えるよう拡張し、相互情報量の推定とその最大化を同時に行うDeepInfoMax[1]を提唱した。そして獲得された表現が有効であることを示している。

この相互情報量を最大化するというアイデアは入力と符号間だけでなく、共通の情報を持つと考えられる異なるデータ間や同じデータの異なるビュー間に対しても適用できる。ビューというのは異なる視点や、異なるモダリティ（カメラと深度センサなど）を表す。

このように異なるデータやビュー間の相互情報量を最大化

した場合には、1つのデータにだけ見られる撹乱要素起因の情報（画像における照明条件、視点位置など）を捨てて、共通して見られる情報（物体や環境の状態）を捉える表現を獲得できる。一般に共通して見られる情報の方が有益な場合が多いため、よりロバストで汎用的な表現を獲得することが期待できる。例えば、英DeepMind社のA. van den Oord氏ら[2]らは異なる情報間の相互情報量最大化による表現学習としてContrastive Predictive Codingを提唱した。同氏はPixelRNNやWaveNetの提案者としても有名である。彼らは予測問題として過去の履歴の情報と、未来の情報間の相互情報量を最大化するようにすることで表現を獲得することを提案した。

2つに限らず複数のビュー間の相互情報量最大化をすることで、より共通して表れる表現を獲得できることも示されている[3]。この場合、より多くのビューを使うほど、各データと符号間の相互情報量はむしろ減り、それにも関わらずその表現を使ったタスクの性能が向上していることがみられた。これは、より重要な情報だけを抽出できるようになっているためと考えられる。

対比損失による相互情報量最大化

それではここから、具体的にどのようにして学習されているのかについてみていこう。与えられたデータセット中で、共通の情報を持っていると考えられるデータのペア (u, v) を用意する。何が共通かは学習設計者が与える。例えば時系列データにおいて、ある時刻のデータとその直後のデータは共通の情報を持っていると考えられる。画像では画像の一部分のパッチと画像全体は共通の情報を持っていると考えられる（象全体が写っている画像の場合、その一部分のパッチ、例えば大きな耳が写っているパッチと象全体の画像は象という共通の情報を持っている）。また、入力に対して意味を変えないような変換、例えば左右反転や明度変換などを適用する前と適用した後のデータは共通の情報を持っていると考えられる。

次に、共通の情報を持っているデータのペア (u, v) をニューラルネットワークによる符号化器 E_q, E_k を使って変換してクエリ $q = E_q(u)$ とキー $k_+ = E_k(v)$ を得る。また同じデータ集合からランダムにデータをサンプリングし、同じ符号化器 E_k を適用して変換して $\{k_1, \ldots, k_K\}$ を得る。k_+ はポジティブキー、$k_1, \ldots, k_K$ はネガティブキーと呼ぶことにする。また、簡略化のために k_+ を k_0 とし、さきほどの得られたキーとあわせて $\{k_0, k_1, \ldots, k_K\}$ というキーの集合を考える。

これらのクエリとキー集合を使ってクエリとポジティブキー間の相互情報量を最大化する。この最大化に対比損失（Contrastive Loss）を使う。対比損失は q が k_+ と近く、それ以外のキー $k_1, \ldots, k_K$ と離れている場合に小さくなるような損失関数である。

$$\mathcal{L}_q = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i/\tau)}$$

ここで τ はハイパーパラメーターであり分布の温度を表す。この損失は k_+ を正解とした（K+1）クラスのSoftmaxを使った分類問題と一緒である。この損失関数の最小化は q と k 間の相互情報量の下限の最大化となることが分かっている。このようにして獲得された符号化器の結果を入力の表現として使うことができる。

MoCo

この対比損失を使った表現学習は、画像のような高次元空間の中でクエリとキーから成る離散的辞書を作っているようにみなせる[1]。この辞書では関係するもの同士は近く、関係しないものは遠く離れるように学習される。この場合重要なのは、クエリは全てのネガティブキーと十分引き離されていることである。実際、学習時に使うネガティブキーの数は増やせば増やすほど表現学習の性能が向上することが分かっている。一般にミニバッチ内の別のサンプルをネガティブキーとして使う場合が多いが、この場合ネガティブキーの数はバッチサイズに制限されてしまう。

そのため、過去に一度計算されたキーを貯めておき、それをネガティブキーとして使う方法も提案されていた。このアプローチをメモリーバンクと呼ぶ。この手法の問題点は符号化器が更新されるたびに全てのキーが変わってしまうので、過去に計算済みのキーが本来のキーと大きくずれており、結果として性能が大きく劣化してしまうことである。

米Facebook社のK. He氏らが提唱したMoCo[3]はこの問題を2つのアイデアで解決した。同氏はResNetやMask R-CNNの提唱者としても知られる。MoCoはネガティブキーをミニバッチからではなくキューに貯めためものを使うようにした。キューは毎回の更新時に現在のミニバッチで使ったポジティブキーをネガティブキーとして追加し、一番古いものを捨てる。これにより今の符号化器に似た符号化器で得られたキーを使うようにした。キューの導入によってバッチサイズよりもずっと大きな数のネガティブキーを使えるようになっ

た（実験ではネガティブキーは65536個を使っている）。2つ目はキーの符号化器をクエリの符号化器のモーメンタムで更新するようにし、学習途中のキーの一貫性を保つようにした。具体的にはキーの符号化器のパラメータをθ_k、クエリの符号化器のパラメータをθ_qとした時、θ_kを以下のように更新する。

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q$$

獲得された表現の評価は、この表現を事前学習済みモデルとして様々なタスクを教師あり学習した時の性能で評価される。MoCoで学習したモデルは、ImageNetの教師あり学習で事前学習して得られたモデルを使った場合と9つのタスクで比較した場合、7つのタスクでMoCoの性能が上回り、はじめて教師なし学習による表現学習が教師あり学習による表現学習を上回った。

今後の教師なし表現学習

教師なし表現学習は画像認識でも少なくとも既存の教師あり表現学習に匹敵する性能を上げるようになってきた。一方で課題がいくつか残っている。1つ目はデータを増やすことによる性能向上がまだ見られない点である。もともと教師なし学習の最大のメリットは教師あり学習よりもずっと大きなデータを使えることにあった。現在のモデルではそれだけ多くのデータを扱えるほどの表現力がないという可能性は十分あるが、この場合大きなモデルを大きなデータセットで学習させるという難しさが出てくる。2つ目は学習の目的関数のより深い理解である。現在の学習は相互情報量最大化を近似して最適化していると捉えられているが、よりタイトに相互情報量を近似できる推定器を使ったとしても性能は向上せず、それよりも符号化器や推定器の構造の方が重要だと分かってきている。教師なし表現学習においても未知の帰納バイアスが存在しているとみられる[5]。教師なし表現学習の技術が確立されたら、汎用な表現を獲得する以外にも、データだけは手に入るような問題やタスクに特化した表現をその場で学習することも増えてくるだろう。

1) R. D. Hjelm, et al., "Learning Deep Representations by Mutual Information Estimation and Maximization," ICLR 2019.
2) A.van den Oord et al.,"Representation Learning with Contrastive Predictive Coding," https://arxiv.org/abs/1807.03748
3) Y. Tian et al., "Contrastive Multiview Coding," https://openreview.net/pdf?id=BkgStySKPB, https://arxiv.org/abs/1906.05849
4) K. He and et al.,"Momentum Contrast for Unsupervised Visual Representation Learning," CVPR 2020.
5) M. Tschannen and et al.,"On Mutual Information Maximization for Representation Learning," ICLR 2020.

（2020年1月号掲載の記事に加筆）

3-12

知識蒸留：巨大なモデルの知識を抽出する

知識蒸留（以下、蒸留）は学習済みモデルの予測結果を学習目標として、他のモデルを学習させる手法である。アルコールの蒸留と同様にモデルから、重要な知識だけを抽出することからこの名前が与えられている。

蒸留はモデル圧縮の手法として2015年にGeoffrey Hinton氏[1]らによって提案（それ以前にも一般の機械学習などで提案されていたがニューラルネットワーク（NN）の分野ではHinton氏らが最初に提案）された。蒸留を使うことで大きなデータセットを大きなモデルで学習し、その結果を小さなモデルに移すことが可能である。

機械学習のべき乗則（第1章 第9節を参照）が示すように、学習データがいくらでも手に入る状況では、モデルが大きいほど汎化性能が高く、学習効率も高いことがわかっている。さらに大きなモデルであるほど少ないデータ数を使った転移学習でも高い性能を達成し、従来の半教師あり学習を大きく凌駕する性能を達成することがわかってきている[2]。そのため、巨大なデータセットで巨大なモデルを事前学習しておき、その後に様々なタスク向けに少ないデータ数で学習するという動きが自然言語処理を中心に広がっている。

一方で大きなモデルは推論時の計算コストが大きいという問題があり、使用可能なハードウエアリソースに制約がある場合は使えない。そこで事前学習時には大きなモデルを利用して学習しておき、個別のタスクで使う場合には、大きなモデルをタスク専用の小さなモデルに蒸留して利用することが考えられ、蒸留が注目されている。

今回は、蒸留とは何か、なぜ汎化性能が高い大きなモデルを小さいモデルに蒸留できるのか、蒸留を成功させる条件とは何か、正則化としての蒸留について紹介していく。

蒸留とは

はじめに蒸留とは何かについて説明しよう[1]。k クラスからなる多クラス分類NNを教師ありデータを使って学習したとする。この学習済みNNの学習結果を異なるNNに移すことを考える。この学習済みNNを教師NNとよび、学習結果を移す先を生徒NNと呼ぶことにする。

教師NNと生徒NNは必ずしも同じネットワークアーキテクチャを持つ必要は無く、多くの場合は教師NNより生徒NNの方が小さいような問題設定を考える。これはモデル圧縮とみなすことができる。

通常の教師あり学習では、教師データの正解を目標に学習する。具体的には教師データの正解に対応するクラスの確率が1であり、それ以外が0であるような分布とのクロスエントロピーを最小化するように学習する。これに対し、蒸留では教師NNの予測分布を目標とし、それとのクロスエントロピーを最小化するように学習する。あるデータに対する教師NNの予測結果を $p_1, p_2, \ldots, p_k$ とし、生徒NNの予測結果を $q_1, q_2, \ldots, q_k$ とした時、この2つの分布間のクロスエントロピーは、以下のように定義される。

$$H(P,Q) = \sum_{i=1}^{k} [-p_i \log q_i]$$

このKLダイバージェンスを最小化するように生徒NNを学習させる。

通常の学習との違いは、教師NNの予測結果が正解ラベル以外にも確率を割り振っていることであり、蒸留は教師NNの間違い方も含め学習しているとみなせる。この正解ラベル以外の予測をHinton氏はDark Knowledge（暗黒知識）とよび、学習に重要だとしている。

蒸留を使うことで、あるモデルの学習結果を他の学習結果に移すことができる。この蒸留を使うケースはいくつかある。1つ目はモデル圧縮で教師NNが大きなモデルやアンサンブルなどでそのまま推論時に使うにはコストが大きい場合に、計算効率の高い小さいNNに移すという場合である。2つ目は正則化として蒸留することで元のモデルよりも汎化性能を改善できることであり、特に同じモデル間で移す自己蒸留である。これらについて紹介していこう。

なぜ汎化性能を保ったまま小さなモデルに蒸留できるのか

なぜ汎化性能が高い大きなモデルを小さなモデルに蒸留できるのか。これを理解するためには、なぜ大きなモデルが

汎化するのかを理解する必要がある。

モデルサイズが大きい方が汎化性能が高くなる理由は大きく2つある。1つ目はモデルサイズが大きい方が汎化性能が高いフラットな解が見つかりやすくなること、2つ目はモデルがより多くのビュー（データの見方、CNNにおけるフィルタ）を持てるようになり、新しいデータに対してもうまく対応できる可能性が高くなることである[3]。

これら2つを元になぜ小さいモデルに汎化性能を保ったまま蒸留できるかをみていこう。1つ目の最適化問題については、学習が終わって汎化性能が高いパラメータが決定されれば、最適化には必要だった冗長なパラメータを消去し、モデルを大幅に小さくすることができる。

2つ目は、もし実際使われるタスクが決まれば、そのタスクに必要の無いビューは捨てることができ小さくすることができる。

一方で、最初からタスクに特化した汎化性能が高い小さなNNを学習させることは難しい。

蒸留の成功条件

蒸留について多くの研究がなされてきたが、2021年6月に米グーグルBrainチームが発表した論文[4]では蒸留を成功させる重要な条件を見つけたと報告した。

1つ目は教師NNと生徒NNが同じデータオーグメンテーションを適用した入力を使うことである。従来は目標となる教師NNの予測分布には計算資源の節約のために特定のデータオーグメンテーション（クロップなど）を適用して計算したものを毎回固定で使い、生徒NNには毎回異なるデータオーグメンテーションを使っていた。この場合、生徒NNからは教師NNの本当の予測とは違う目標を学習していることになる。生徒と同じ入力を教師NNが使うことで、一貫性のある目標を設定することができる。さらに複数の学習データの線形補間を入力に使うmixupを積極的に使うことが重要であった。

2つ目は蒸留時には従来の教師あり学習よりもずっと長い時間学習するということである。データオーグメンテーションが強く、小さいNNへ蒸留するため最適化が難しいためだと考えられる。

これら2つの条件が満たされた場合、大きなデータセットを大きなモデル（BiT-M-R152x2）で学習した結果を小さなモデルのResNet-50に蒸留することで、ImageNetで82.8%の精度を達成できた（元の大きなモデルでは83.0%の精度）。ResNet-50を直接大きなデータセットで学習させていた場合は77.2%であり、5〜6%近くも改善できることが示された。

正則化としての蒸留

蒸留には、それ自身が正則化として働くことがわかっている。蒸留とよく似た正則化としてLabel Smoothingがある。これは正解の確率を少し減らし、その他のクラスの確率を一様に少しずつ上げるという方法である。

Label Smoothingは、各クラスに所属するサンプルをより強固にクラス中心に近づける効果があることがわかっている。これは他のクラスからも一様の力で引っ張られることにより均衡点がクラス中心のみになる効果が発生するためだと考えられている[5]。

一方で蒸留の場合は自己蒸留のたびに使用される基底関数の数が制限され、関数の表現力が制限されることがわかっている[6]。この制約は非線形であり自己蒸留を繰り返すことで、正則化が強くなっていき、一定回数を超えるとUnderfittingを起こし汎化性能は逆に下がってしまう。このような表現力を抑える手法としては、学習を途中で止めるEarly Stoppingが知られているが、Early Stoppingではこのような基底関数を疎にする役割はなく、むしろ使われる基底関数の数自体は多くなることがわかっており、蒸留はEarly Stoppingとは別の仕組みで正則化を実現している。

蒸留の理論的な解明と実用的な発展

本稿では蒸留の仕組みについて解説を試みたが、大きなNNの汎化能力を保ったまま小さなNNになぜ蒸留できるのかについては未解明な部分が多い。

一方で実用的には、今後は巨大な事前学習用データセットを用意して、大きな学習済みモデルを作っておき、それを用途に応じて蒸留して使うような時代が想定される。またほとんどの場合はNN間の蒸留を考えているが、NN内の各モジュール間で蒸留を介して知識のやりとりをすることも考えられる。例えばGLOMは蒸留を使って、重みパラメータを共有することなく位置不変の予測を実現できることを提案している。

今後、蒸留の理論的な解明と実用題での利用が進むと考えられる。

1) G. Hinton et al.,"Distilling the Knowledge in a Neural Network," NeurIPS Deep Learning and Representation Learning Workshop, 2015. https://arxiv.org/abs/1503.02531

2) T. B. Brown et al.,"Language Models are Few-Shot Learners," NeurIPS 2020.
3) Z. Allen-Zhu et al.,"Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning," https://arxiv.org/abs/2012.09816
4) L. Beyer et al.,"Knowledge distillation: A good teacher is patient and consistent," https://arxiv.org/abs/2106.05237
5) R. Müller et al.,"When Does Label Smoothing Help?," NeurIPS 2019.
6) H. Mobahi et al,"Self-Distillation Amplifies Regularization in Hilbert Space," NeurIPS 2020.

(2021年8月号掲載の記事に加筆)

Masked Autoencoder：画像認識でも事前学習革命は起きるのか

深層学習（ディープラーニング）の大きな特徴は、データを問題が解きやすいような表現に変換する方法を学習によって獲得する、いわゆる表現学習ができる点である。データを適切に表現できさえすれば、その後、分類や回帰などの問題は簡単に解けるのに対し、うまく表現されていない場合はその後どれだけ頑張ってもうまく問題を解くことはできない。

また、良い表現方法を事前学習によってあらかじめ獲得しておけば後続タスクの精度を改善できるだけでなく、学習に必要なデータを劇的に減らせる。例えば、画像を入力とした強化学習においても、画像の表現学習を中心とした工夫を組み合わせることにより、必要な経験回数を1/500近くも減らすことができると報告されている[1]。

深層学習が登場した2006年ころは自己符号化器（Autoencoder）などによる教師なし学習によって表現学習を行っていたが、2012年ころに教師あり学習が成功し始めて以降は、教師あり学習による表現学習が盛んとなっていた。しかし教師あり学習による表現学習にはいくつかの問題がある。

1つ目は教師データを作るのにコストがかかり、利用可能なデータが限られる点、2つ目は教師あり学習で必要なタスクに特化した表現が獲得されてしまい、他のタスクに使えないような表現が獲得されてしまう点である。例えば、画像分類の教師あり学習で得られた表現には、画像中の物体の位置情報などは失われてしまっている。

こうした問題があることから、教師なしデータを使って表現学習を行うことが求められていた。これが最初に大きく成功したのが自然言語処理の分野である。2018年のBERTや2020年のGPT-3などの大量のテキストデータを使った事前学習による表現学習は、自然言語処理の多くのアプローチを置き換えてしまった。

自然言語処理で起きた事前学習革命を画像認識の分野でも再現しようと、自己教師あり学習による画像表現学習は多く提案されている。しかし、まだ得られた表現の精度や多様性においては成功していない。

この画像の表現学習に対し、米Facebook社（現Meta Platforms社）AI ResearchのKaiming He氏らがBERTと同様にマスク付き自己符号化器（MAE：Masked Autoencoder）を使って実現する手法を提案した[2]。He氏は画像認識で最も使われているモデルであるResNetやMask R-CNNなどの提案者であり、現在の深層学習を使った画像認識の第一人者であるといってよいだろう。

このMAEはこれまで提案された手法と比べて単純であり、学習を効率的に実現でき、得られた表現性能が従来手法と比べて改善されている。本稿ではこのMAEについて紹介する。

Vision Transformer

MAEは2021年に入って急速に普及したTransformerを使った画像認識システム、ViT（Vision Transformer）をベースにしている。ViTは畳み込み層を使わずTransformerを使って画像を変換する[3]。既にViTを使った画像認識器は数百も提案されており[4]、各画像認識タスクの精度で上位を占めるようになっている。

まず、このViTについて説明しよう。ViTは始めに画像を重なり合わないパッチに分割する。パッチには16x16といったサイズを使い、解像度が224x224の画像は14x14 =196個（224/16=14）のパッチに分割される。次に各パッチに位置符号を加えた上で、それらを線形射影を使って特徴ベクトルに変換する。この特徴ベクトルをトークンと呼ぶ。そして、トークン列をTransformerで繰り返し変換していく。あたかも画像を196個のトークンからなる文に変換し、その文を自然言語処理で使われているモデルと全く同じモデルで変換しているとみなせる。

最後にこれらのトークン列から用途に応じて表現を抽出する。画像全体の1つの表現が必要な場合は、CNNと同様にAveraged Poolingを使ったり、特殊なトークンを1つ用意しておき、そのトークンの表現を使う。

MAEによる表現学習

それではMAEについて説明する。MAEは全体としては

自己符合化器であり、入力の一部分をマスクした上で符号器を使って符号化し、次にそれらの符号を入力とし、復号器を使ってマスクされた入力を予測する。学習が終わった後は復号器は必要ないので捨て、符号器を使って表現を得る。

MAEはViTと同様に画像をパッチに分割した後、ランダムにパッチをマスクし、捨てる（例えば75%のパッチをマスクする）。次に残ったパッチをトークン列に変換し、ViTと同様にTransformerを使って符号を求める。次に符号と位置符号を入力とし、復号器を使ってマスクされたパッチの各画素値を予測し、平均二乗誤差を最小化するよう学習する。

画素値を予測して表現学習する際は将来の多くのタスクには不必要な詳細を捨てたほうがよいという考えもあり、ベクトル量子化など離散化した値を予測するアプローチも提案されているが、今回はパッチ毎に画素値の平均と分散を求め、それらを使って画素値を正規化した上で予測しても同様の品質の表現が獲得できると報告している。

なぜこの方法で画像の良い表現が学習できるのだろうか。マスクされた一部のパッチから残りのパッチの画素を予測するには、残されたパッチから画像全体の情報や意味を推定できるような表現を獲得できなければならない。例えば車が写っている画像の一部がマスクされても復元できるようにするためには、各パッチが車の先頭部分やタイヤ部分を表しているということを符号器は推定できるようにならなければならない。今回はマスクされるパッチの割合は75%と多く、人が見た場合でも画像の意味を推定するのが難しいようなタスクである。

MAEによって獲得された表現を、画像認識や物体検出、セマンティックセグメンテーションなど様々なタスクに適用して評価した結果、従来の自己教師あり表現学習で最高性能を占めていたDINOやMoCoと比べて高い性能を達成できており、さらに教師あり学習により獲得された表現よりも優れていることが分かった。

教師あり学習では大きなモデルを使っても性能は改善されず、むしろ悪化する場合も多いが、今回のような自己教師あり表現学習はモデルが大きいほど性能が改善されることが報告されている。これについては第1章 第9節の「機械学習の新べき乗則、大きなモデルを使うと汎化しサンプル効率も改善する」でも述べた。（マスク付き）自己回帰モデルの場合、データ自体を復元するため教師シグナルの量や種類が非常に多く、大きなモデルであるほど多くの情報を獲得できるためと思われる。また後述するようにMAEは従来手法と比べ効率的に学習できるため、従来では実験できなかった大きなモデルを扱うことができ、そこでも性能が向上していることが実際に確認された。

MAEはBERTと良く似たアプローチであり、これまで画像の表現学習にも多く試されたと思われるが、今回初めて大きく成功した理由は大きく3つ挙げられる。

1つ目はVision Transformerの利用である。CNNと違ってTransformerはマスクされた疎な情報をうまく扱うことができ、またマスクされているという情報も埋め込み符号として、うまく扱うことができる。2つ目はMAEの場合、マスクする割合をBERTが使っていたような15%ではなく75%と大きくしたことである。情報が離散化され凝縮されている単語に比べて画像は空間冗長性が大きい。そのため多くの部分をマスクしないと、画像の意味など推定しなくてもマスクされた画素を周辺パッチからの内挿によって推定できてしまう。簡単に予測できず、なおかつ残されたパッチから画像全体の意味がぎりぎり推定できる割合が75%だったといえる。3つ目に復号器にBERTは簡単なMLPを使っていたが、画像の画素を予測する場合には複雑な問題を解く必要があり、軽いとはいえTransformerを使ったモデルを使っている点である。

MAEは計算効率も優れている

教師なしデータを使った自己教師あり表現学習は計算量が大きい問題があった。教師なしデータは非常に大きく、またモデルは大きければ大きいほど後続タスクの性能が改善される。大量のデータを大きなモデルで学習させるには多くの計算量が必要になってしまう。

MAEは学習の計算効率の観点からも優れている。1つ目は符号器で75%をマスクした後、残りの25%のパッチのみで処理することだ。従来CNNなどでは入力の一部をランダムにマスクしても、疎な計算が苦手なため高速化には寄与しない。これに対しMAEはパッチをランダムにシャッフルした後に前半の25%だけを残し、それに対してViTを適用する。このため、単に入力を短くした場合とみなせ、密な計算で実現できる利点がある。この入力の一部分のみを計算したとしても密に計算できるというのは、この論文の大きな貢献部分であり、他の問題でも広く使われていくと考えられる。

2つ目は符号器と復号器に別のモデルを使う点である。大きなモデルを使う符号器は小さな入力を扱い、小さなモデルがマスクされた入力も含めデータ全体を扱うことで、全体に

必要な計算量を抑えている。

こうした単純だが計算効率としても優れる方法を考えるのは、Kaiming He氏の得意領域である。

大きなデータ、モデルでの検証はこれから

今回のMAEはImageNet-1Kという比較的小さなデータセットに適用されているが、今後は大きなデータセットに適用された場合にどれだけ性能が改善されるのかが注目される。データ数だけでなく、入力解像度を増やしたり、モデルを大きくすることも今回の手法を基にさらに改良すれば可能だと考えられる。こうしたことが実現されれば、自然言語処理で起きたような事前学習革命が画像認識においても起きるのではないかと考えられる。

1) W. Ye et al., "Mastering Atari Games with Limited Data," NeurIPS 2021.
2) K. He et al., "Masked Autoencoders Are Scalable Vision Learners," https://arxiv.org/abs/2111.06377
3) A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," ICLR 2021.
4) S. Khan et al., "Transformers in Vision: A Survey," CM Computing Surveys 2021.

（2022年1月号掲載の記事に加筆）

第4章

強化学習

4-1 強化学習：フィードバックから最適行動を獲得する

強化学習は、人間や動物が現実世界を生き抜いていく時に使っている学習を参考にして作られた技術である。次のような問題を解くために使う。「学習の主体であるエージェントは、時刻ごとに環境から状態sを受け取り、行動aを取る。環境は、エージェントが選んだ行動に依存して報酬rおよび次の時刻の状態s'を決める。これを一定期間繰り返す」。強化学習の目標は、将来にわたっての報酬の合計を最大化するような行動を選ぶ方法を獲得することにある。

例えば、ラジコンカーを学習によって自律的に走らせる場合を考えてみよう。ラジコンカーはカメラやレーダなどセンサから環境の状態を得て、ハンドルを切るか、アクセルを踏むかといった行動を決める。そして、目的地に早く到達した場合に正の報酬を、ぶつかったり止まったりした場合に負の報酬を受け取る。この場合、強化学習を用いることで、ラジコンカーは将来にわたっての報酬を最大化、つまりぶつかったり止まったりせずに、早く目的地に到達するような行動を獲得できるようになる。

教師信号は与えられない

強化学習は機械学習の一種であるが、次の2つの特徴がある。1つは、教師あり学習と異なり、ある状態ではどの行動を取るのが正解というような教師信号は手に入らない点である。このため、どの行動を取るべきか学習するには実際にその行動を取ってみて、環境からのフィードバックを得るしかない。

もう1つの特徴は、エージェントは逐次的に行動を選択し、報酬をもらうという点である。これら行動と報酬の間には時間差があるかもしれない。例えば、ある時刻にブレーキを踏むことはその時点では負の報酬をもらうかもしれないが、その後のコーナーをより速く壁にぶつからずに回ることができれば、将来、より大きな報酬を得ることができるだろう。

後者の特徴は「信用割り当て」と呼ばれる問題と関係がある。報酬を最大化するように行動を修正するには、ある報酬がどの時刻のどの行動と関係したのかを知る必要がある。近年のディープニューラルネットワーク（DNN）は信用割り当て問題を誤差逆伝播法の仕組みを用いて解いていた。強化学習では誤差逆伝播法の代わりに価値（関数）を使ったQ学習などによって信用割り当て問題を解いている。

強化学習は教師あり学習と比べてシステム設計者には優しく、学習システムには厳しい問題設定になっている。システム設計者は正しい行動は何かを考えることや、正解データを作る必要はない。望ましい状態、望ましくない状態は何かを考え、それらに対する報酬を設計するだけでよい。

行動価値関数をニューラルネットでモデル化

強化学習のエージェントは、方策（ポリシー）πと呼ばれる状態sから行動aを決定する関数$\pi: s \to a$を学習を通じて獲得する。この方策は決定的な場合と確率的な場合がある。将来の期待収益は行動価値関数$Q^{\pi}(s, a)$によって表される。これは、状態sにおいて行動aを取り、その後は方策πを取った時に受け取る報酬の合計値の期待値である。環境は、状態遷移と報酬が現在時刻の状態と行動のみに依存するマルコフ過程であるとする。このとき、最適行動価値関数$Q^*(s, a) = max_{\pi} Q^{\pi}(s, a)$について次のベルマン方程式が成り立つ。

$$Q^*(s, a) = r + \gamma \max_{a'} Q^*(s', a')$$

ただし、rは状態sで行動aを取った時にもらえる報酬値、$0 < \gamma < 1$は割引率、s'は状態sの時に行動aをとった時の次の状態である。この右辺をQ目標と呼ぶことにする。

この行動価値関数を何らかの関数でモデル化し、学習により獲得することを考える。例えば、ニューラルネットワークでモデル化し、$Q(s, a; \theta)$で表したとする。ただし、θはニューラルネットワークを特徴付けるパラメータである。このとき、Q学習と呼ばれる方法は行動価値をQ目標に合わせるように学習することで行動価値関数を学習する。

$$L(\theta) = \mathrm{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta)\right)^2\right]$$

2つの工夫で非線形モデルを利用可能に

ニューラルネットワークのような非線形モデルを行動価値関数のモデル化に利用する試みは、値が収束せずに振動し

たり、発散したりするケースが多いことが知られていた。

米グーグルのDeep Mindが提唱した「Deep Q-Learning Network（DQN）」[1]は2つの工夫を加えることで学習に成功した。1つは、リプレイバッファの利用である。エージェントが取った行動を一定期間リプレイバッファに保存し、学習時にはリプレイバッファからランダムにサンプルを取得し、それらに対して最適化を行う。Q学習においてリプレイバッファを使わず、連続する時刻を対象に学習する場合、サンプル間に相関があり、学習が非効率なだけではなく、データの分布が大きく偏っているため、学習が発散する可能性が高くなってしまう。

2つ目の工夫は、学習時に目標値を計算するために使用するパラメータは一定期間ごとに更新し、一定期間内は固定しておく工夫である（上記式内のθ^-）。通常のQ学習のように学習時に目標値も動いた場合は、学習が安定しない。例えば、1回目の更新で目標値がハンドルを左に切るようにパラメータが更新され、次はハンドルを右に切るようにパラメータが更新されるといったように、目標値の振動が起こってしまう。

これら2つの工夫により、強力なニューラルネットワークを行動価値関数のモデル化に利用できるようになった。複雑な状態、例えば画像のように数万次元ある場合でも学習することができるようになった。

DQNの研究では、「Atari 2600」と呼ばれる家庭用ゲーム機に対して強化学習を適用している。ビデオゲームを強化学習の対象とするのは、（学習に使わないとしても）状況が全て把握でき、問題の複雑さや多様性が適当であり、大量のシミュレーションができるためである。

ゲームの画面自体を入力（ゲーム内部の状態ではなく、表示している画素値を入力とする）、ゲームの操作を行動、スコアを報酬とした。49種類のゲームに対し単一の学習システムで既存手法を凌駕した。特に反射神経を要するようなタスクにおいては、かなり上手な人間のゲームプレーヤーのスコアを上回っている。

一方で限界も見えている。「Montezuma's revenge」と呼ばれるゲームのスコアは0、つまり全くプレーできていない。実際にプレーしてみると分かるが、このゲームではプレーヤーはRPGのように数画面分移動し、鍵を取得し、別の場所でその鍵を使ってドアを開けるといった問題を解く必要がある。単純な条件反射のような問題ではなく、問題を分割し、それらの進捗を記憶する必要がある。例えば、ドアを開くには鍵が必要、鍵を見つけるには別の画面に移動しなければいけない、などである。それらに加え「鍵は取得するもの、それはドアで使うもの」といった、試行錯誤では容易には得られない事前知識が必要である。

こうした課題を解くには、新しい種類の報酬が必要になる。例えば、新しい知識を得ること自体に報酬を与えたり、自分の知識を他のエージェントに教えたり、または教えてもらった場合に報酬を与えるといった具合である。こうした好奇心や、他のエージェントに教えることへの報酬設計については、人間や動物の本能などが参考になるかもしれない。

1) V. Mnihk, et. al., "Human-level control through deep reinforcement learning," Nature, 2015

（2015年11月号掲載の記事に加筆）

4-2

世界モデル：world model 想像の中で学習できるか

強化学習は、環境中のエージェントが環境と相互作用していく中で、将来に受け取る報酬の合計を最大化するための最適な行動を選択できるよう学習する問題である。今まだ解けていない人工知能の問題の多くが強化学習の枠組みの中で解けるだろうと考えられている。強化学習は深層学習と組み合わさったことで大きく発展し、AlphaGoによって囲碁の世界トップ棋士に勝つなどの実績を上げている。

しかし、(深層) 強化学習には大きな問題が2つある。1つは学習に非常に多くの学習事例を必要とすること、2つは環境やタスクが変わった場合うまく対応できない、つまり汎化しないことである。

1つ目の学習データを多く必要とする問題は現在の強化学習がモデルフリー、すなわち環境についてのモデルを仮定せず、試行錯誤した経験を使って最適化していることに起因する。モデルフリーの場合、強化学習は状態や行動と報酬との関係を求めるだけでなく、観測から状態をどのように決定するのか、行動が環境にどのような変化をもたらすのかについても推定しなければならない。一方、世の中の多くの問題は環境をモデル化することが困難である。例えば実環境でのロボットの制御問題では、ロボット自体のモデル化に加えて、ロボットと周辺環境との接触部分（摩擦現象）、応力、歪み、熱変位といった部分はモデル化が困難である。

2つ目の汎化しない問題は入力（観測）データをそのまま扱っていることに起因する。現在の強化学習は、環境が変わったり（例：昼が夜になる）、行動の意味が変わった場合（例えば、同じアクセルを選択しても加速度が異なる）に汎化することは難しい。環境や行動が変わったとしても同じモデルが使えるよう汎化するためには、観測や行動をそのまま扱わずに、タスクと関係のない、その問題特有の情報を捨て、抽象化された情報の上で最適化を学習しなければならない。

これらの問題を解決するためには、環境の抽象的なモデルを構築し、そのモデル上で強化学習を行うということが必要になる。最近、3つのグループがこれを実現する手法を発表した。

米Google BrainのDavid Ha氏とスイスIDSIAのJürgen Schmidhuber氏らが提案したWorld Model[1]は、観測xをVAE（変分自己符号化器）を使って低次元の潜在ベクトルzに変換する。この変換は教師なし学習である尤度の変分最適化によって実現される。次に、潜在空間上での前向きモデル$z_{t+1}=f(z_t, a_t)$をRNNを使って学習する。この前向きモデルは元々の入力x_{t+1}を予測するのではなく、潜在ベクトルz_{t+1}を予測し、観測中の不要な詳細を予測しなくても良いようにしている。最後に、潜在空間z_tとRNNの内部状態h_tをつなぎ合わせたベクトルを入力として方策（状態から行動を選択する関数）を線形モデルで作る。RNNは未来を予測できるように学習されるため、その内部状態は未来の情報を含むようになっており、線形の単純なモデルで方策を学習できる。この方策は進化戦略で学習する。

このような環境モデルを学習することは過去にも試みられていたが成功していなかった。このWorld Modelが成功した大きな理由はRNNに混合ガウシアンによってモデル化した確率的遷移を採用したことである。例えば、シューティングゲームにおいて、敵が一定確率で弾を撃ってくる場合を考えてみる。この場合、ある時刻の状態から次の時刻に敵が弾を撃つかどうかは予測できない。このような確率的に起こる現象を1つの未来しかない決定的なモデルで学習してしまうと、ありうる未来の中間を予測するのが最適となってしまう。複数の未来がありうるような環境をモデル化できることで、それに対応できる方策を学習できる。

環境の確率的なモデル化は、学習された方策が特定の環境に特化しないことにも役立つ。もし学習された環境が実際の環境と合っていない場合、その上で学習される方策は実際の環境では起きないような環境モデルの“バグ”をついて、最適化されてしまうといったことが起きる。環境モデルが確率的であり、方策にとって環境の予測が難しくなっていれば方策が環境の不具合を悪用することを防ぐことができる。

2つ目は、英DeepMind社が提案した、外部記憶を利用した強化学習「MERLIN」[2]である。この研究はDifferentiable Neural Computerの後続となっている。MERLINのエージェントは現在の状態を外部記憶に書き出すと共に、外部記憶から必要な情報を読み出して（思い出して）処理することができ

る。このとき問題になるのは、情報をどのような形で外部記憶に書き込むかである。外部記憶の学習では記憶する時と思い出す時に時間差があるので、行動を改善するには情報をどのように記憶すればよかったか、何を記憶すればよかったかを求めることが難しい。MERLINは観測を符号化した上で未来の観測と報酬を予測する問題を最初に解くことで観測をどのような低次元の情報に落とすかを学習する。この学習では、入力だけでなく報酬も予測することで、報酬に関係のある特徴も捉えている。MERLINは予測学習で獲得した符号化器を使って観測を符号化してそれを格納すると共に、その後に生成された状態の割引付き合計$(1-\gamma)\Sigma_{j>i}\gamma^{(-j)}z_i$も一緒に格納する。この割引付き合計によって、ある場面を思い出した時に、その後に何が起きたのかも一緒に思い出すことができる。

この記憶がどのように利用されるのかについて、迷路を解く場合を例に考えてみよう。迷路を解いている間、エージェントは現在の観測に似た過去の記憶を思い出す。さらに、ゴールまでに通過した途中の位置を思い出し、そこを目指して進むといったことができる。実際、エージェントが迷路を解いている途中にどこを思い出しているのかを可視化してみると、ゴール周辺の記憶や、ゴールまでの途中のポイントを思い出しておりエージェントの意図を見ることができて興味深い。予測学習は最適な記憶方法を学習できる有力なタスクである。

3つ目が、University of California BerkeleyよるUniversal Planning Networks（UPN）[3]である。これは前の2つの手法とは違って、模倣学習により最適な表現方法と環境の前向きモデルの学習を実現する。

UPNは観測o_tを潜在状態x_tに変換する符号化器$x_t=E(o_t)$、潜在状態x_tと行動a_tを受け取り次の潜在状態x_{t+1}を返す環境の前向きモデル$x_{t+1}=F(x_t,a_t)$の2つを学習する。

UPNは最適な行動列を決定するプランナーとそのプランナーを使って表現とダイナミクスを学習させる外側の学習システムから構成される。プランナーは初期入力o_1と目標入力o_gを受け取ると、行動列$a_1,a_2,...,a_N$を返すような関数である。

プランナーは符号器を使って、初期入力と目標入力を初期状態x_1と目標状態x_gに変換する。次に行動列$a_1,a_2,...,a_N$を適当な初期値で初期化し、この場合の最終状態を計算する。

$$x_{i+1}=f(x_i,a_i)$$
$$\vdots$$
$$x_{N+1}=f(x_n,a_n)$$

そして、最終状態と目標状態の差を計算し、これを目的関数$L(x_g,x_{N+1})=\|x_g-x_N+1\|$とする。この目的関数$L$を小さくするように勾配降下法を使って各行動$a_i$についての勾配$\frac{\partial L}{\partial a_i}$を計算し、各行動を$a_i=a_i-\eta\frac{\partial L}{\partial a_i}$のように更新する。この更新を数回繰り返し、最終的な行動列$a'_1,a'_2,...,a'_N$を得る。

次に、プランナーが出力した行動列を、エキスパートによる最適な行動列$a^*_1,a^*_2,...,a^*_N$と比較し、この差を最小化するように符号化器と前向きモデルのパラメータを勾配降下法で最適化する。

$$U=\|a'_1,a'_2,..,a'_N\|-|a^*_1,a^*_2,...,a^*_N\|$$

プランナーの計算は微分可能な計算グラフで構成されるため、プランナー内部のEとFも誤差逆伝播法で最適化可能である。

この学習では、最適な表現やモデルはプランナーが勾配降下法によって最適な計画を立てやすいようにという観点で選ばれ、必ずしも元の入力情報を全て保存しない。

強化学習を実用化するためには学習に必要なデータ量を劇的に減らし、汎化させることが必要である。今後もこれらを実現するための表現学習が続けられると考えられる。

---

1) D. Ha et al.,"World Model,"https://worldmodels.github.io/, https://arxiv.org/abs/1803.10122
2) G. Wayneand et al., "Unsupervised Predictive Memory in a Goal-Directed Agent," https://arxiv.org/abs/1803.10760
3) A. Srinivas et al.,"Universal Planning Networks," ICML 2018.

（2018年6月号掲載の記事に加筆）

4-3

安全が保証された強化学習：リアプノフ関数で制約満たす方策を導出

強化学習はシミュレーションやゲームなどで人の能力を超える性能を達成しており有望視されているが、現実の問題に適用した場合、その安全性をどのように担保するのかが問題となる。強化学習は環境との相互作用の中で試行錯誤しながら自分の行動がどのような結末をもたらすのかを理解し、自分の行動を改善していく。しかし、試行錯誤のみから危険な状況も理解させるのはあたかも子供に包丁をもたせて、その安全な使い方を自ら学んでもらうようなものであり危険である。そのため、あらかじめ危険な状態や行動を定義しておき、それらを回避した範囲内で行動することが考えられる。しかし、ある時点での行動や状態が最終的に危険な状況につながるのかは一般に未知もしくは計算困難であり、それも推定しなければならない。

本稿では、強化学習の安全性を保証する手法として2018年に英DeepMind社が発表した論文を紹介する。DeepMind社は2016年から グーグルのデータセンターの冷却システムの最適化を進めている。最初はシステムが推奨した結果を基に人が運転をしていたのが、2018年現在はシステムが全自動で運転しており、平均して30%の最適化を達成できたと報告している。今回紹介する論文[1]の研究成果が直接利用されているかは不明だが、他の安全な探索[2]の利用例としてデータセンターの冷却システムが言及されている。データセンターの制御は、失敗すると熱暴走によりコンピュータシステムや施設に甚大な被害をもたらしうるため、安全性の保証が重要な問題である。

はじめに強化学習の問題設定についておさらいしよう。強化学習ではエージェントが環境と相互作用していく中で、自分が将来にわたってもらえる報酬の和が最大となるような行動を選んでいくことが目標となる。状態から行動を選択するルールを方策（policy）とよび、方策はニューラルネットワークなどを使い関数近似する場合が多い。なお、報酬最大化の代わりに最適制御のようにコスト最小化問題を考える場合もあるが、この場合はコストの符号を反転させた関数の最大化を考えれば同じ問題に帰着する。

これを式で表すと、エージェントは時刻 t に環境から状態 s_t を受け取り、それを基に行動 a_t を方策 $a_t = \pi(s_t)$ に従い選択する。環境はこの行動を基に報酬 $r_t = r(s_t, a_t)$ を返す。状態が確率的に遷移する場合は現在の状態と、行動に基づき、$P(s_{t+1}|s_t, a_t)$ に従って次の状態が決まる。強化学習の目標は初期状態 s_0 の時に受け取る報酬の合計の期待値 $R_\pi(s_0) = \mathbb{E}\left[\sum r_t|s_0, \pi\right]$ が最大となるような方策 π を求めることである。

リアプノフ関数の値が少なくなるよう方策を更新

この一般的な強化学習の問題に安全性の保証を与えるため、各状態に依存した制約コスト $d_t = d(s_t)$ を考え、方策の累積制約コストの期待値が一定値 d_0 以下であることを要請する。

$$D_\pi(s_0) := \mathbb{E}\left[\sum d(s_t)|s_0, \pi\right] \leq d_0$$

この累積制約コストはさまざまな問題を扱うことができる。例えば、終了状態に至るまでに危険な状態を1度でも通過する確率を抑えたい場合や、危険な状態に到達する回数を抑えたい場合などをモデル化できる。

この最適化に対し論文[1]ではこの制約に基づいてリアプノフ関数を定義し、それを用いてグローバルな制約問題を、ローカルな制約問題に変換する。方策を更新する際はリアプノフ関数の値が少なくなる場合のみに制約することで、全体の制約を達成することを保証する。このリアプノフ関数について説明する。

原点を含む領域 Ω 上で定義された関数 $V(s)$ が $V(0) = 0$ で、かつ $s \neq 0$ なる任意の $s \in \Omega$ に対して $V(s) > 0$（または $V(s) \geq 0$）を満たす時 $V(s)$ は Ω で正定（または準正定）であるという。また $-V(s)$ が正定（または準正定）である時、$V(s)$ は負定（または準負定）であるという。関数 $V(s)$ が原点を含む領域 Ω で正定であり、かつ状態 $s(t)$ が時刻とともに変化していく時に、$V(s(t))$ の時間微分が準負定であるとき、この関数をリアプノフ関数と呼ぶ。

リアプノフ関数は原点が最も低いようなお椀形をしており、

時間の経過とともにリアプノフ関数の値は減少し続け、原点で安定する。このようなリアプノフ関数 $V(s)$ が存在することが、そのシステムの原点が安定であることの十分条件である。なお、この場合、原点が平衡点であったが、原点以外の点が平衡点である場合も平衡点が原点になるよう変数変換をすることで原点が平衡点である場合に帰着できる。

今回考える強化学習の問題設定では時刻は離散時間であり、状態 s の時、行動 a を選択し、次の状態 s' に決定的に遷移する場合、$V(s) > 0$ かつ $V(s') \leq V(s)$ であることが V がリアプノフ関数である条件である。

ここでは、確率的遷移の場合を考えるため、強化学習問題における一般ベルマンオペレータを定義する。

$$T_{\pi,h}[V](s) = \sum_a \pi(a|s)\Big[h(s,a) + P(s'|s,a)V(s')\Big]$$

ここで、π は方策、h は即時報酬や制約コスト、V は状態価値関数やリアプノフ関数である。状態が確率的に遷移する場合、$T_{\pi_B,d}[L](s) \leq L(s)$ であることが L がリアプノフ関数となるための十分条件である。

はじめに、制約を満たした適当なベースライン方策 π_B が存在すると仮定しよう。このとき、この方策に基づいた次の条件を満たす、リアプノフ関数の集合 $\mathcal{L}_{\pi_0}(x_0, d_0)$ を考えることができる。

全ての非終端状態 s について、$T_{\pi_B,d}[L](s) \leq L(s)$
初期状態 s_0 について $L(s_0) \leq d_0$
終了状態 s_e について $L(s_e) = 0$

なお、$L(s) = D_{\pi_B}(s)$ は上の条件を満たすため、$\mathcal{L}_{\pi_0}(x_0, d_0)$ は必ず空ではない。

このリアプノフ関数 $L \in \mathcal{L}_{\pi_0}(x_0, d_0)$ に対し、$T_{\pi,d}[L](s) \leq L(s)$ を満たすような方策 π を L 導出方策とよび、その集合を F_L と表記する。この集合に含まれる方策は全て制約を達成するような方策である。しかし $F_L(s)$ の中に最適方策が含まれている保証は一般にない。これに対し、次のようにリアプノフ関数として、制約コストに加えて補助制約コスト ϵ を加えたものを考える。

$$L_\epsilon = \mathbb{E}[\sum_{t=0}^{T^*-1} d(s_t) + \epsilon(s_t)|\pi_B, s]$$

もし、最適方策が分かっているならば、それを基に補助制約コストを設定することで、最適方策を導出可能なリアプノフ関数を定義できることが示せる[1]。しかし、現実には前もって最適方策を知ることはできない。そのため、現在の制約を満たした方策にリアプノフ条件を達成できるような中で補助制約コストの合計が最大のものを選択するようにし、最適方策が含まれることを期待する。最大の補助制約コストは次の線形計画法を解くことで得られる。

$$\bar{\epsilon} \in \arg\max_{\epsilon \in \mathbb{S}' \to \mathbb{R}_{\geq 0}} \Big\{ \sum_{s \in \mathbb{S}'} \epsilon(s) : d_0 - \mathcal{D}_{\pi_B}(x_0) \geq \mathbf{1}(s_0)^\top (I - \{P(s'|s,\pi_B)\}_{s,s' \in \mathbb{S}'})^{-1} \epsilon \Big\}.$$

ここで、$\mathbf{1}(x_0)$ は $x = x_0$ の場所だけ 1 であり、その他は 0 であるようなベクトルであり、ϵ は $\epsilon(s)$ を並べたベクトルである。

この制約式の右辺の意味は、ノイマン級数 $(I-A)^{-1} = \sum_{i=0}^{\infty} A^i$ より、

$$(I - \{P(s'|s,\pi_B)\}_{s,s' \in \mathbb{S}'})^{-1} = \sum_{i=0}^{\infty} P(s'|s,\pi_B)^i$$

であり、$\mathbf{1}(s_0)^T (I - \{P(s'|s,\pi_B)\}_{s,s' \in \mathbb{S}'})^{-1}$ は s_0 から開始して各状態に到達する確率の総和を表す。よって、右辺は各状態に到達する確率の総和それぞれに補助制約コストを掛けたものの総和となる。この総和が補助制約コストに設定可能な残りより小さくなることを要請する。

この線形計画法の解として、もし $\epsilon(s)$ が s によらず定数の場合は $\epsilon = d_0 - \mathcal{D}_{\pi_B}(s_0)/\mathbb{E}[T^*|\pi_B, s_0]$ が得られる。ただし、T^* はエピソードが終了する時刻である。このように求められた補助制約コストは、現在の方策が制約としてまだ余裕がある分を各状態に均等にバッファとして与えたものとみなすことができる。

この手法に基づき、はじめに制約を達成した方策から開始し、L_ϵ を更新しながら、F_{L_ϵ} に含まれる方策の中で最適な方策を選択するようにして学習していく。これを交互に繰り返していく。

実験ではいくつかのシミュレーション上において提案手法が学習開始から最後まで制約をほぼ達成しつつ、期待累積報酬が大きいような学習を実現できたことが示されている。

制約コストが状態の変化に対して少しずつしか変わらないという保証ができるのであれば、さらに探索中に効率よく安全性の保証をすることができる[2]。

強化学習の実用化が進むにつれて、安全性の保証はますます重要になってくる。今回の話に加えて、メタ学習、モデルベースとモデルフリーの強化学習の融合、シミュレーションと実世界のギャップの解消が重要になるだろう。

---

1) Y.Chow, et al.,"A Lyapunov-based Approach to Safe Reinforcement Learning," NeurIPS 2018.
2) G. Dalal, et al.,"Safe Exploration in Continuous Action Spaces," https://arxiv.org/abs/1801.08757

(2018年11月号掲載の記事に加筆)

4-4 先読みに基づいたプランニング：学習化シミュレータとモンテカルロ木探索

先読みに基づいたプランニングは強力なエージェントを作る上で重要である。例えばチェス、将棋や囲碁などではトッププレーヤーをはるかに超える強さを実現したAIシステムが先読みによって実現されている。

先読みを行う際、環境で未知の部分があれば予測を必要とする。環境シミュレータといってもよいだろう。しかし、一般に予測は難しい。例えば対戦ゲームでは先読みする上で相手の挙動は未知である。幸い、この相手の挙動の予測に限っては学習途中の自分自身を相手に設定することで少なくとも最善な手を選ぶ相手を作ることができる。さらにチェスや将棋などの対戦ボードゲームでは環境の完璧なシミュレーション（選択した行動に対し環境がどう変わるか分かっている）が使える。そのため、対戦ボードゲームに限っては正確な予測ができ、先読みに基づいたプランニングできる。そのため他の分野に先駆けて非常に強いシステムができたといえる。

しかし世の中の多くの問題ではシミュレータが存在せず難しい予測問題を解く必要があり、正確な先読みができない。この場合、先読みに基づくプランニングを諦め、別のアプローチをとる必要がある。

このように正確な先読みできる問題かどうかによってプランニングは大きく2つの問題に分けられてきた。先読みできる場合は状態が列挙できれば動的計画法で解くことができ、問題が特別な構造を持っている場合は最適化で解くこともできる。これらを満たさない場合でも強力なモンテカルロ木探索（MCTS）を使える。このMCTSについて簡単に説明しよう。

MCTSは、今後取る行動と状態の可能性を木の形で表現する。現在の状態が根に対応し、状態sで行動aを選択した結果、状態s'に遷移した場合は、節点sにaのラベルが付随した枝にぶら下がる形で節点s'が子となる。この木の表現上で有望そうな状態（節点）を順に展開して行き、最適な行動（列）を絞り込んでいく。具体的には毎回根から後述する基準に従って子を順に選択して行き子をもたない節点（葉）までたどり着く。そこでプレイアウトと呼ばれる最終結果が出るまで行う高速なシミュレーションを行って収益（ゲームであれば勝ち負け）を計算し、それをその節点の評価値（強化学習における価値と同じ）とする。最後の状態まで行かなくても報酬が貰える場合はプレイアウトではなく途中までの累積報酬を使う場合もある。子を選択する際は節点の評価値と不確実ボーナスの合計が大きいものを選択する。節点の評価値はその子孫の評価値の平均である。このように各節点の評価値をサンプルからモンテカルロ推定していることからこの名がついている。また不確実性ボーナスはまだ十分探索されていない節点を選択するよう加えられるものであり、節点の訪問回数が多くなるほどボーナスは減っていくように設計しておく。この展開とシミュレーションを時間が許すまで行い、最終的に最初の手の中で訪問回数が最も多いものを選択する。このMCTSは強力であり多くの問題で成功を収めている。

それに対し先読みできない問題ではこのように実際試して評価することができない。そのため過去の経験から現在の状態や行動がどのぐらいの価値（期待される累積報酬）があるのかを推定する。これらは状態価値や行動価値として表される。状態が無数にある場合、これらの価値はニューラルネットワーク（NN）などを使って表し、その関数を特徴付けるパラメーターを、TD目標（現在の時刻の価値が1時刻報酬と次の時刻の価値の和に一致するようにする）などを使って更新していく。この更新によって関数近似誤差がなければこれら価値関数は真の期待累積報酬に収束することが分かっている。

このように先読みできる問題とできない問題は棲み分けされ独立に発展してきた。しかし、近年のモデルベース強化学習はその境界を曖昧にしている。

モデルベース強化学習では環境のシミュレータをNNなどでモデル化し、そのモデルを実際の経験から学習する。このシミュレータは世界モデルとも呼ばれる。世界モデルは一般に現在の状態と行動から将来の観測を復元するように学習する。しかし前述のように予測はとても難しい問題である。そのため世界モデルもかなり単純化された状況以外は成功

していなかった。

そのため予測を完全に解くのではなくプランニングに関係する要素だけ正確に予測できるようにする手法がこの数年登場している。プランニングに背景や見た目が関係なければ、シミュレータはそれらを無視して予測するようにする。例えばレースゲームで大部分の背景はプランニングに影響ないが、次にカーブがあるという看板はプランニングに影響があり、そうした要素だけを予測できるようにする。

この代表がMuZero[1]である。MuZeroはデータからシミュレータを学習し、そのシミュレータを使ってMCTSでプランニングする。このシミュレータは将来の観測を復元することは目標としない。代わりに現在の状態と今後取る行動列を入力として任意ステップ後の報酬、価値、方策（状態から行動を返す関数）が実際の環境を使った場合と一致することを目標とする。方策が一致するということは別の言い方をすれば実際の環境でプランニングした結果とシミュレータ上でプランニングした結果（選択する行動）が一致するようにシミュレータを学習する。シミュレータはプランニングに影響がない部分は予測する必要がないため予測問題は簡単になる。

またMuZeroはAlphaZeroと同様に、方策の学習においてMCTSを実施して得られた最適行動を目標として方策を更新する。先読みした結果は先読みしない方策より常に強いので、良い学習目標となる。このMCTSで得られる目標はQ学習に使われるTD目標に比べて良い学習シグナルであることが分かっており、高速に学習できるだけでなく、最終結果で大きな差がつくことが分かっている。

MuZeroによってシミュレータを持たない問題にもMCTSでプランニングできるようになる。この結果、これまで価値（DQNや最新のRainbowなど）ベースの手法が使われていたAtariゲームなどの問題もMCTSが使えるようになった。この両者を比較した実験結果では従来の価値関数を使った手法と比較しMCTSを使った手法が性能で大きく上回った。これまで多くの改良がなされてきた価値ベースの手法をMCTSの手法が一気に凌駕したことは大きなインパクトがある。MCTSは元々離散的な状態、行動が対象だったが、連続的な状態、行動の場合への拡張も提案されている[2]。

学習化シミュレータとMCTSの組み合わせは有望だが、まだ多くの問題が残されている。MCTSの一番大きな問題は並列化が難しい点である。シミュレーション自体は独立に並列にできるが、どの節点を展開するかを決めるには他の結果を待つ必要があり、それを無視して独立に並列に実行すると皆同じような節点を探索してしまうという問題が起きる。この問題に対し、シミュレーション結果を待たずに先に不確実性ボーナスだけを減らすという単純な工夫が有効であることが分かっている[3]。また並列にシミュレーションといっても独立であり、現在のチップ（GPUやTPUなど）の性能を生かすためのベクトル化も難しい。

また、部分的にしか予測しない世界モデルを学習した場合、誤った因果推論をしてしまい、モデル化していない観測の影響を受けて誤ったプランニングをしてしまうことが分かっている[4]。これらの問題を解決できればより広い問題を強力な先読みを使って汎用的に解けるようになるだろう。

---

1) J. Schrittwieser, et al.,"Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model," https://arxiv.org/abs/1911.08265
2) T. Hubert et al.,"Learning and Planning in Complex Action Spaces," ICML 2021.
3) A. Liu et al., "Watch the Unobserved: A Simple Approach to Parallelizing Monte Carlo Tree Search," ICLR 2020.
4) D. J. Rezende et al.,"Causally Correct Partial Models for Reinforcement Learning," https://arxiv.org/abs/2002.02836

（2020年4月号掲載の記事に加筆）

オフライン強化学習：データ主導型学習に向けて

強化学習は学習ベースの制御を実現し、囲碁のトップ棋士を破ったAlphaGoを代表的な事例として、ロボティクス、自動運転、金融、医療/ヘルスケア（処置の判断）など幅広い分野で成果を収めている。一方で強化学習は学習中に環境との相互作用（試行錯誤）を必要とするため、その実現のハードルが高かった。本稿では、過去に取得した経験データのみを利用し、環境と新たな相互作用を必要とせずに強化学習を実現するオフライン強化学習について紹介する。

オンライン強化学習の問題点

はじめに、強化学習について簡単に説明する。強化学習は、制御対象であるエージェントとその他の情報を司る環境から構成される。時刻tにエージェントは現在の状態s_tを元に行動a_tを決定する。環境は現在の状態と選択された行動から次の状態s_{t+1}と報酬r_{t+1}を決定する。このプロセスを決められた回数や条件を満たすまで繰り返していく。

エージェントの状態から行動を決定するモデルを方策$\pi(a|s)$とよび、また報酬の合計値$\sum_t r_t$を収益と呼ぶ。強化学習の目標は収益を最大化するような方策を獲得することである。例えば、壁にぶつからずに走るロボットカーを学習させる場合には、道沿いに速く進んだら正の報酬、ぶつかったら負の報酬を与えるように設定すれば、ぶつからずに道沿いに速く走るような制御が強化学習で獲得できる。

一般に、強化学習では現在の学習途中の方策でエージェントを環境中で動かし経験データ$\{(s_t, a_t, s_{t+1}, r_{t+1})\}_{t=1}^{N}$を集積し、この経験データを元に最適な方策を推定する。このような強化学習をオンライン強化学習と呼ぶ。現在強化学習というと、このオンライン強化学習を想定する場合が多い。

しかし、世の中には環境で試行錯誤することが困難だったり不可能である問題が多い。例えば、ミッションクリティカルな自動運転やヘルスケア/医療領域などでは、学習途中のエージェントを使って制御をすることは危険であり許されない。ロボットなどを使った場合でも、試行錯誤の過程においてロボット自身や対象物などを壊してしまう恐れがある。

一方でこれらの問題では、人や別のシステムで経験データが取得されている場合がある。例えば、自動運転や医療/ヘルスケアでは過去に人が実行した経験データが存在し、ロボットなどでも安全が保証された別の方法（人による遠隔操作など）で取得した経験データを集めることができる。目的を達成する対話システムを強化学習で実現させたい場合も、強化学習のエージェントによる試行錯誤を行うのは大変だが、過去の人の会話履歴を収集することは容易である。

このような過去に取得した経験データのみを利用し、新たな環境との相互作用をせずに最適な方策を推定する強化学習をオフライン強化学習（バッチ強化学習とも呼ばれる）と呼ぶ[1)]。

オフライン強化学習は何が難しいか

オフライン強化学習は魅力的であるにも関わらず実現は難しい。まず、経験データに含まれていないような状態や状態行動ペアについては、その時の状態遷移や報酬を推定することは不可能である。この問題は経験データ収集時の探索問題であり重要な問題であるが、今回は取り上げない。

以降では経験データで網羅している範囲内で最適化すること考えていく。この問題自体も難しい。経験データ中の状態/状態行動分布と、現在、学習中の方策における状態/状態行動分布に差があるためである。この問題は分布シフト（distribution shift）と呼ばれる。教師あり学習においても訓練データと評価データ間の分布が異なる場合に、差がわずかであっても性能が劇的に悪くなってしまうことが知られている。強化学習においても、この分布の差によって、実際とは異なる収益を推定してしまい、誤った最適行動を導き出してしまう。

強化学習には、現在学習中の方策と異なる方策を使い、それによって収集された経験で学習する方策オフ学習も存在する。方策オフ学習の場合、学習中の方策を使って新た

な経験データを収集する。これにより、例えば収益評価を一時的に誤って推定しても、そこを実際探索し対応する経験データを集められていけば、実際に得られた報酬を元に誤った収益評価は修正される。

それに対し、オフライン強化学習では新たな経験データを集めることができないため、誤った評価は修正されず、さらに強化学習で使われるベルマンバックアップなどで更新を繰り返していく度に誤差は蓄積していってしまう。これを防ぐために、後半で述べる不確実性の評価などが重要となる。

重点サンプリングを利用したオフライン強化学習

それではここからオフライン強化学習のアプローチを紹介していく。まず考えられるのは重点サンプリングを利用した手法である。

一般に、目標分布と異なる分布からサンプリングされたデータを使って学習する場合、重点サンプリングを使って、目標分布で期待値をとった値を推定することができる。

例えば、経験データの確率分布が $q(x)$ 、目標分布が $p(x)$ であり、目的関数の $p(x)$ での期待値を求めたい場合は、

$$\mathbb{E}_{x\sim p(x)}\left[L(x)\right]=\mathbb{E}_{x\sim q(x)}\left[\frac{p(x)}{q(x)}L(x)\right]$$

のように経験データからのサンプル $x\sim q(x)$ を $w(x):=p(x)/q(x)$ で重み付けし直すことで、目標分布の期待値を推定できる。この $w(x)$ を重みと呼ぶ。オフライン強化学習の場合も x を行動状態列 $(x=[s_1,a_1,s_2,a_2\ldots])$、現在の方策による分布を目標分布として、重点サンプリングによって現在の方策による期待収益を推定できる。

重点サンプリングの問題点はデータ分布と目標分布が大きく違う場合、推定の分散が非常に大きくなってしまうことである。例えば経験データでほとんど経験したことがないような状態や状態行動ペア ($q(x)\simeq 0$) の重みは非常に大きくなってしまう。強化学習ではこの x は行動状態列に対応するため $p(x),q(x)$ は状態確率、方策確率の積となる。重みはこれら状態確率、方策確率の比の積となり分散も非常に大きくなってしまう。

この問題を防ぐため様々な手法が提案されている。例えば経験データから複数のデータ x_i をサンプリングし、それらの重みの和 ($\sum_i w(x_i)$) で、各重みを正規化した weighted importance sampling はバイアスは生じるが分散をずっと小さくすることができ広く使われている。

また、経験データの分布と目標分布が近い領域のみで最適化することができれば、分散を小さくすることができる。これは、データ収集時に使った方策から大きく離れない範囲でのみで方策を最適化することで実現できる。例えば学習対象の方策とデータ収集時の方策間のKL-ダイバージェンスをペナルティとして目的関数に追加する手法などが提案されている。

一方で方策が近い範囲内で探索するのは、保守的すぎる場合がある。例えばデータ収集時にランダムに行動を選択するような方策を使ったとしよう。この場合、学習対象の方策も同じように様々な行動を確率的に選択するような方策しか選べない。

このように方策が近い範囲内で探すという制約より、学習対象の方策が到達する状態/状態行動が経験データに含まれるように制約する方が良いと考えられる。このアプローチが次に述べる不確実性を考慮した強化学習である。

不確実性を考慮した強化学習

観測数が少ないために生じる推定結果の不確実性をエピステミック不確実性 (Epistemic uncertainty) と呼ぶ。エピステミック不確実性は観測数を増やしていくことで小さくできる。以降、不確実性といえばこのエピステミック不確実性を指すとする。

この不確実性はブートストラップアンサンプルを利用して推定することができる。例えば複数のニューラルネットワークを用意し、それぞれに別々の初期値と別々の学習データを使って学習する。これらのニューラルネットワークによる推定結果がばらばらであれば不確実性が高く、一致しているのであれば不確実性が低いといえる。

不確実性が高い状態/状態行動は、それに対応する収益 (価値関数) の推定誤差も大きい。強化学習は、収益が最大となるような行動をターゲットとして学習する場合が多く (Q学習など)、不確実性がある場合、常に楽観的な推定をしている行動を選択し、誤った推定結果を使ってしまう。この誤りは更新ごとに別の状態に伝搬していき全体の推定に大きな悪影響をもたらす。

同様にモデルベース強化学習で、環境のダイナミクスを

推定している場合、この予測誤差を価値関数が悪用して不当に高い収益を推定してしまう。モデルの予測誤差を価値関数推定器が悪用する現象はmodel exploitationとして知られる。

これを防ぐため、価値関数を推定する際に、不確実性も同時に推定し、不確実性の分を割り引いて評価することで、悪用を防ぐことができる。オフライン強化学習でない問題設定でも、この不確実性を利用した強化学習は有効であることがわかっている[2]。

今後の展望

オフライン強化学習は強力な学習パラダイムとなりうる。試行錯誤が不可能な問題に強化学習を適用できるだけでなく、最適化対象のシステムのデータを一度取得さえしておけば、その後のシステム最適化を少ないコストで実現できる。これら収集した経験データのみを使って強化学習をすることができれば、データドリブンの強化学習が実現できる。

教師あり学習では一度作った学習データを何回も再利用して手法を改善していくことができたが、強化学習においても同様に、経験データさえ一度コストをかけて収集さえしておけば、後は何回もそれを再利用して手法を改善していくことができる。

また、研究開発も加速されることが期待される。画像や自然言語処理において教師あり学習が急速に進んだ理由として、教師ありデータセットが既に用意されており、新しいアイデアを試す時にすぐに試すことができたことがあげられる。

それに対し強化学習ではシミュレーションや実環境を準備しなければならず、誰でも気軽に始められるものではなかった。それがオフライン強化学習では世界の誰かがデータ収集さえ行っていて公開さえしてくれれば、残りの研究者はシミュレータや実環境のシステムのセットアップなしに新しいアイデアをすぐ試すことができる[3]。

今後、オフライン強化学習の発展とともに、強化学習が適用できる問題が広がることが期待される。

1) S. Levine et al.,"Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," https://arxiv.org/abs/2005.01643
2) K. Lee et al.,"SUNRISE: A Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning," ICML 2021.
3) J. Fu et al., "D4RL: Datasets for Deep Data-Driven Reinforcement Learning," https://sites.google.com/view/d4rl/home

(2020年10月号掲載の記事に加筆)

第5章

高速化・低電力化・インフラ

5-1 ディープニューラルネットの学習をどこまで速くできるのか

深層学習は学習に時間がかかることで有名である。データやモデルにもちろんよるが、学習に数日、数カ月といった単位の期間が必要になることも稀ではない。この学習を高速化することは研究開発の競争力に直結する。新しいアイデアを試してその結果が出るのに1週間以上かかる場合と、それが1時間で終わる場合では、試行錯誤の回数や仕方が大きく変わってくる。また、高速化によって、より大きな学習データセット、大きなモデルを使えるようになる。

GPUの高速化とともにモデルも巨大化

深層学習の学習にどれだけ時間がかかるのか感覚を掴んでもらうために、いくつか例をみてみよう。深層学習のチュートリアルで最初に取り上げられるMNISTデータセットの画像分類の学習から始める。MNISTの学習データは5万枚の28×28の白黒画像の手書き文字からなる。これを3層から成る多層パーセプトロンを使って学習する場合を考える。中間層に100個のニューロンを使った場合、ネットワーク全体で10万弱のパラメータから成る。このとき、1つの画像を推論（Forward）するのに25万回の浮動小数点演算、学習するのにその約3倍の約75万回の浮動小数点演算が必要となる。学習全体では学習データ全体を20回走査し、その場合、CPU1つ（Intel Xeon E5-2667）では3分程度、GPU1つ（NVIDIA Tesla P100 PCIe）では1分程度かかる[注1]。この程度の時間であればぼとんど苦なく試行錯誤できる。

次に、現実的な学習問題の例としてImageNetデータセットの学習を取り上げる。ILSVRC、通称ImageNetデータセットは120万枚の訓練画像から成り、1000クラスの分類問題を解く。

2012年にILSVRCにおいてディープラーニングを使って優勝し金字塔を打ち立てたAlexNetは、当時のGPU（NVIDIA GTX 580、性能は1.6TFlops、Flopsは1秒間に可能な浮動小数点演算回数である）を2台使い、5～6日の学習時間が必要であった。その後、GPUの性能は毎年約2倍のペースで上がり続けるものの（2017年11月時点で最速のNVIDIA Tesla V100は15.7TFlops）、モデルも巨大化している。近年使われるモデルはResNetやDenseNetなど、層の数が数十～数百、チャンネル数（各層のニューロン数は幅×高さ×チャンネル数）は数百から数千という非常に大きなネットワークである。例えば50層から成る中程度の大きさであるResNet-50では、サイズが224×224の画像を推論するのに38億回の浮動小数点演算が必要であり、学習にはその約3倍の100億回の浮動小数点演算が必要となり、1台のGPUでImageNetデータセットを学習するために学習データを約100回走査し、2週間程度必要である。

高速化の主流となっているデータ並列

この学習を高速化するために、現在はデータ並列による高速化が主流となっている。深層学習の学習である確率的勾配降下法（SGD）は次のように行われる。(1) 学習データセットからミニバッチと呼ばれる一定数個のデータをサンプリングし、(2) それらミニバッチから目的関数のパラメータについての勾配を推定し、(3) この勾配の推定値を使ってパラメータを更新する。この勾配の推定は、各データ毎に独立に計算できるため、並列に計算することができる。データ並列による高速化では、複数台のワーカーを用意し、ミニバッチをさらに分割したデータ（マイクロバッチと呼ばれる）を各ワーカーに分配し、ワーカー毎に勾配を独立に推定し、それらを集約することで高速化する。

このデータ並列による高速化には大きな問題が2つある。1つがパラメータのStaleness（新鮮ではないパラメータを使ってしまう）問題である。並列処理においてワーカー間で同期が必要な場合、遅いワーカーに全体の処理が引っ張られてしまう問題があるため、非同期処理が望ましい。そのため以前はDNNの学習でも非同期処理をし、最新ではないパラメータを元に勾配を求め、それを使って更新する手法がとられていた。しかし、古いパラメータを使って勾配を求める悪影響は大きく、特に汎化能力に大きな影響があることが実験的に分かってきた。そのため、常に全ワーカーが最新のパラメータを持ち、それを元に勾配を計算できるようにならなければならない。DNNはパラメータ数が多く全ワーカーで最新のパ

ラメータを保持するには高スループットかつ低レイテンシなネットワークが必要となる。

もう1つの問題として、データ並列による高速化の際にはミニバッチサイズを大きくする必要がある。これは各ワーカーのマイクロバッチが一定以上大きくなければ、各ワーカーが効率的に計算できないためである。現在のGPUやHPCを使った多くの計算処理ではデータ当たりの計算回数、つまり計算密度が高いことが求められる。計算性能の向上に対し、データバンド幅の向上が追いつかないためだ。そのため小さなバッチで計算した場合は実効性能が大幅に落ちてしまう。そのため各マイクロバッチを大きく、結果としてミニバッチサイズも大きくする必要がある。しかし、ミニバッチサイズを大きくすると今度は汎化性能が落ちる現象がみられることが分かっている。つまり訓練誤差は最小化できるが、実際の推論時のテストデータの誤差が大きくなってしまうという問題である。

アニーリングのような効果を持つSGD

この現象の原因は、現時点ではミニバッチサイズを大きくすると勾配のノイズが小さくなり、汎化に悪影響を与えているためだと分かっている[1]。SGDは単に勾配をサンプルから求めることで高速化するだけでなく、勾配に適度なノイズを加えることでアニーリングのように最適化が局所解にはまらないようにするのを助けていたのだ。さらに適切なノイズはベイズ最適化やMDL（minimum description length）などとつながり、汎化性能向上に寄与していることが分かっている。

これに基づいて、ミニバッチサイズが変わっても同じスケールのノイズが加わるように、ミニバッチサイズを大きくするのに合わせてSGDの学習率を大きくする[1]。これによりノイズの大きさが元の学習と同じとなり、汎化性能を保ったまま高速化することができる。

ImageNetの学習がついに15分に

こうしたさまざまな工夫を重ねることでデータ並列による学習の高速化が可能となってきた。この高速化は2017年に急速に進んできた。

2017年1月、Preferred Networks（PFN）はGPUを100台使ってImageNetの学習問題を4時間で同じ精度（以降精度はテストデータでの精度を示す）で学習できることをDeep Learning Summit San Francisco 2017で発表した。このとき、学習率を大きくすれば汎化性能が落ちないという知見はPFNは実験によって独自に得ていた。その後、前述のように、このアイデアが正しいことが理論的にも証明された。

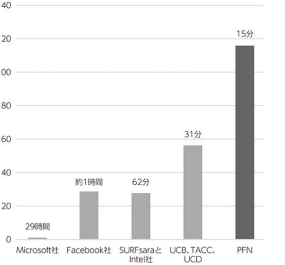


図5-1　ResNet50 ImageNetの学習の高速化の度合い
8台のGPUの場合をベースラインとした際の高速化の度合いを示した。

2017年6月には米Facebook社が256台のGPUを使って1時間で同じ精度で学習できることを示した[2]。また、学習開始時に学習率を徐々に上げ、学習を安定化するwarmup strategyを提案。ミニバッチサイズをn倍にし学習率もn倍にした場合の学習曲線（各epochでの目的関数値や精度の軌跡）が元の学習曲線とほとんど一致することも発見した。

その後、2017年9月に米University of California Berkeleyなどの研究グループがIntel Xeon Phi 7250アクセラレータ付きのCPUを1600個使い30分で学習可能と報告した[3]。彼らは層毎に学習率を正規化するLARSと呼ばれる手法を使って学習の安定化を図ることを提案している。

2017年12月にPFNはGPUを1024台使って15分で同じ精度で学習できると発表した（図5-1）[4]。この時のミニバッチサイズは32768であり、通常使われるミニバッチサイズの100倍近くにもなる。非常に大きな学習率を使うため、より詳細な学習率やモーメンタムの調整に加えて、初期はRMSpropを使い、後半はモーメンタムSGDを使って汎化性能を保つ工夫がなされた。また、PFNの今回の成果はC++で書かれたCaffeではなくChainer（ChainerMN）[5][6]を使っており、Pythonで開発できる効率さを保ったまま学習をスケールできることを示した。

今後も大規模化と高速化は続く

高速化がもたらす今後の可能性について述べてみよう。ま

ず、より大きなデータセットで学習できるようになる。ImageNetより大きなデータセットはすでに数多く登場してきている。代表的なものとして、1000万枚、約5000クラスからなるOpen Images（https://github.com/openimages/dataset）、24万個のビデオ、1000万個のアノテーションからなるYouTube BB（https://research.google.com/youtube-bb/）、1億枚のラベル付き画像からなるYahoo Flickr Creative Commons（http://yfcc100m.appspot.com/）などがある。米グーグルは訓練データを3億枚まで増やして学習した結果、データ数の増加に応じて精度は伸び続けると報告している[5]。そのほか、教師なし学習や半教師あり学習などのアノテーションコストの少ない学習手法が提案されていて、それらを使えばより大量の教師なしデータを活用できる。また、CGを使った学習のように実質無限のデータセットが使える手法も登場している。また、強化学習やGANはアルゴリズムの性質上、収束までに多くのイテレーションが必要であり、学習の高速化が期待されている。

また、今より大きなモデルを使った学習も重要となる。通常の機械学習の理解とは違って、ニューラルネットワークはモデルが大きいほど過学習しないことが予想され、実験的には証明されており、計算コストさえ解決できればより大きなモデルを使うことが望まれている[6]。また、学習時に大きなモデルを使ったとしても、学習が終わった後、性能を落とさず、それをずっと小さく圧縮することができることが分かっており、推論時には大量のリソースを使わずに済む。それなら最初から小さなモデルを使って学習できるのではと考えるが、小さなモデルでは学習は難しくなり、過学習しやすくなってしまう問題が起きる。大きなモデルは表現力を上げるだけでなく学習を容易にしているのだ。脳でも最初に神経回路網を多く作った後に重要ではない部分が刷引かれることが分かっており、この考えに基づいた学習手法も提案されている[7]。今後も高速化、大規模化の流れは続くと考えられる。

1) S. L. Smith et al.,"A Bayesian Perspective on Generalization and Stochastic Gradient Descent," https://arxiv.org/abs/1710.06451
2) P. Goyal et al., "Accurate, Lage Minibatch SGD: Training ImageNet in 1 Hour," https://arxiv.org/abs/1706.02677
3) Y. You et al., "ImageNet Training in Minutes," https://arxiv.org/abs/1709.05011
4) T. Akiba et al.,"Extremely Large Minibatch SGD: Training ResNet-50 on ImageNet in 15 Miniutes," https://arxiv.org/abs/1711.04325
5) C. Shun et al.,"Revisiting Unreasonable Effectiveness of Data in Deep Learning Era," https://arxiv.org/abs/1707.02968
6) M. S. Advani et al.,"High-dimensional dynamics of generalization error in neural networks," https://arxiv.org/abs/1710.03667
7) S. Han et al.,"DSD: Dense-Sparse-Dense Training for Deep Neural Networks," https://arxiv.org/abs/1607.04381

注1）Chainerのexampleにあるtrain_mnist.pyをそのまま使って実測した。

注2）ChainerMNは、Chainerでの学習を複数のノードで分散処理するためのChainerの追加パッケージである。

（2018年1月号掲載の記事に加筆）

モバイル向けのニューラルネットワーク：推論時の電力効率を高める3つの方策

ニューラルネットワークは多くのタスクで高い性能を出しており、特に画像認識や音声認識では既存手法を大きく超える性能を達成している。一方で、ニューラルネットワークは学習、推論（利用）時に、既存手法に比べて多くの計算を必要とすることが知られている。学習時に必要とされる計算はスループット重視であり、計算能力が十分にあるデータセンターやクラウドなどに学習データを集約した上で時間をかけて行えば良い。

一方で、推論時に必要とされる計算は遅延時間（レイテンシ）重視であり、処理はデータが生成される場所の近くですぐ実行したい場合が多い。データセンターやクラウドで推論する場合、ネットワーク通信の通信時間のオーバーヘッドに加えて、ネットワーク通信が途切れる可能性もあり、クリティカルな用途には使えない。

例えば自動運転車やロボット、ドローンの画像認識、今後増えるであろう音声によるインタラクティブな質問応答などでは、数百msから数十msといった低レイテンシの処理を端末近くで実現したい。一方、こうしたエッジのデバイスは使える電力に制約がある。こうした背景から、推論時にはニューラルネットワークは端末で実現し、計算が軽く必要な電力は小さいことが要請される。

1枚の画像で数百億回の浮動小数点の積和演算が必要

今のニューラルネットワークがどの程度の計算コストを必要とするのか具体例でみてみよう。ニューラルネットワークの主要な計算は32ビットや16ビットの浮動小数点演算であり、特に畳み込み層や総結合層の重みと活性値間の行列積の計算コストが支配的である。

一般物体認識、物体検出、セグメンテーションは表5-1のようなパラメータ数と演算数を必要とする（これらの数字は参考文献[2]より）。これらはあくまで目安であるが、このようにパラメータ数は約1億、画像1枚当たり数百億回の浮動小数点の積和演算が必要とされる。これらは、画像サイズが大きくなったり、1秒間に処理が必要なフレーム数が増えたりすると必要な演算数も比例して大きくなる。例えば、物体検出をHD（1280×720）の解像度で30フレーム/秒で動かすには単純計算で秒間14兆回の浮動小数点の積和演算が必要だ。

この必要とされる計算能力は現在のモバイル機器の能力とはかなりのギャップがあるが、これを埋める流れがいくつか出ている。ここではその流れを3つ紹介する。1つ目は省電力のハードウエアの開発、2つ目はネットワークアーキテクチャの改善、3つ目は量子化/2値化ニューラルネットワークの利用である。

省電力なハードウエアが登場

1つ目として、ハードウエアの改善である（2018年3月時点で）。例えば米NVIDIA社はJetson Xavier NXを開発している[※]。15Wまたは20Wで21兆回、10Wで14兆回の演算を提供している。

米グーグルが提供するGoogle Edge TPU[6]を使った場合は1W当たり2兆回のint8演算をサポートする。こうしたデバイスは数千円で提供されている。

アーキテクチャは機械で自動設計

2つ目としてネットワークアーキテクチャの改善である。画像認識において、ImageNetでの一般物体認識などの精度向上は頭打ちとなりつつあるが、計算回数に制限がある中での精度は向上し続けている。その中でも強化学習や遺伝的アルゴリズムによって、ネットワーク構造の設計を行うことにより、省電力でも性能の高いようなネットワーク構造が発見されてきている[5,6]。

例えば、遺伝的アルゴリズムにより見つかったAmoebaNet-C[6]は500万個のパラメータ、5億回の浮動小数点演算によってImageNetのTop-5の精度が92.1%である（何も制限が無い場合の最高精度であるSENetは96.2%）。今後は計算回数やパラメータ数が小さなハードウエアの制限（実効効率や消費電力の最適化）を考慮した上でのネットワークアーキテクチャの自動最適化が進むと考えられる。

計算コストに制限がない部分でも、近いうちに人手で設計したネットワークは機械が設計したネットワークに抜かれる

表5-1　DNNに必要な計算コスト

	一般物体認識	物体検出	セグメンテーション
入力画像の大きさ（画素）	299 × 299	320 × 320	512 × 512
DNNのパラメータ数（個）	1.46	0.36	0.6
必要な積和演算数（個/回）	423	350	800
ネットワーク構成	SENet[1]	SSD300	DeepLabv3 +ResNet-101

と考えられる。

量子化や2値化で電力削減

3つ目はニューラルネットワークの量子化、2値化である。ニューラルネットワークの重みや活性値を量子化によって低ビット整数や1ビットの情報に置き換えることができれば計算コストや消費電力を大幅に減らすことができる。重みを2値化（+1, −1）すれば乗算は加減算に置き換えることができる。さらに、活性値も2値化することができれば乗算はビット間XNOR演算に置き換えることができ、計算コストを大幅に抑えることができる。例えば、32ビットの浮動小数点の乗算はXilinx社のFPGAでは600個前後のLUTとフリップフロップ（FF）が必要だが（数個のDSPスライスを併用すれば100個前後のLUT/FFで実現可能）[7]、1ビットの演算は1個のLUT/FFで実現できる。

計算コストだけでなく、現在の消費電力の大部分はメモリからのデータ転送時に発生している。データを量子化することでデータを小さくすることで、省電力化を達成できる。量子化は微分不可能な演算であり、量子化した上でニューラルネットワークを学習することは困難そうにみえる。しかし、近年では、誤差伝播時には量子化の影響を無視するStraight-Through推定を使っても多くの場合学習できることがわかってきており、精度を落とさないように量子化する研究が進んでいる。

例えば、ドローンの世界最大手である中国DJI社の研究グループが発表した論文[8]では、元の浮動小数点の行列積を複数の2値化した行列積の演算の組み合わせで表現することで、精度低下を抑えつつ重みと活性値の2値化に成功している。ベクトル量子化による高速化も今後重要になるだろう[9]。

モバイルや組み込みでもニューラルネットワークの推論が問題なく使えるようになれば、次は学習もしたくなるだろう。しかし、学習は桁違いに多くの計算リソースが必要であり、量子化を導入したとしても学習時（特に勾配計算時）に浮動小数点演算が必要となってしまう。これらは今後の研究課題である。

1) J. Hu et al., "Squeeze-and-Excitation Networks," https://arxiv.org/abs/1709.01507
2) M. Sandler et al., "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation," https://arxiv.org/abs/1801.04381
3) https://www.nvidia.com/ja-jp/autonomous-machines/embedded-systems/jetson-xavier-nx/
4) https://cloud.google.com/edge-tpu
5) B. Zoph et al., "Learning Transferable Archtectures Scalable Image Recognition," https://arxiv.org/abs/1707.07012
6) E. Real et al., "Regularized Evolution for Image Classifier Architecture Search," https://arxiv.org/abs/1802.01548
7) https://www.xilinx.com/support/documentation/ip_documentation/ru/floating-point.html
8) X. Lin et al., "Towards Accurate Binary Convolutional Neural Network," NIPS 2017.
9) R. Guo et al., "Quantization based Fast Inner Product Search," https://arxiv.org/abs/1509.01469

（2018年4月号掲載の記事に加筆）

5-3 AI研究の苦い教訓

強化学習の創始者の一人であり、この分野を長年にわたってリードしてきたUniversity of Alberta教授のRichard Sutton氏が「The Bitter Lesson」(苦い教訓)というタイトルで記事を投稿した[1]。以下にその内容を簡単にまとめる。

苦い教訓

この70年間のAI研究で得られた最も大きな教訓は、計算能力を活かした汎用の手法が最終的には最も有効であったということである。この背景にあるのはムーアの法則として知られる計算能力の指数的な性能向上である。一般にAIの研究では、エージェントの計算能力は固定だと考える。この場合、AIの性能を向上させるためには人間のドメイン知識をシステムに埋め込むしかない。しかし、一般的な研究プロジェクトの期間より長い期間でみると、計算能力の向上を活かした汎用の手法が最終的に大きな差をつけて有効だと分かる。

例として、チェスでは1997年に米IBM社のDeep Blueが世界王者のGarry Kasparov氏を破った。この実現のために特別なハードウエアを開発し、力技の探索ベースの手法を利用した。多くのAI研究者は人の知識を埋め込んだ手法が勝つことを願ったがそうではなかった。囲碁はそれから20年を必要としたが、英DeepMind社のAlphaGoが世界トップ棋士のイ・セドル氏を破った。ここでは学習にニューラルネットワークに特化したハードウエアであるTPUを利用し、自己対戦学習とモンテカルロ木探索を利用して破った。音声認識と画像認識でも、専門知識を持った研究者が設計した特徴や識別器を使った手法が多く開発されてきたが、それらは現在、大量の学習データと計算リソースを使った深層学習を利用した手法に置き換わっている。

これらの経験をまとめると、AI研究者たちは最初はドメイン知識をシステム上に構築する。それらは短期間では有効だが、いつしか停滞し始め進歩が止まってしまう。そして長期的には全く別のアプローチである計算性能の向上を活かした探索と学習を中心とした手法が大きなブレークスルーを上げ、ドメイン知識を埋め込んだ手法を大きく上回る。これらの成功は人間を中心としたアプローチではないため、AI研究者達には受け入れられず苦味を帯びた成功として受け止められる。

また、知能や心というのは非常に複雑であり、単純な方法では心の内容(場所、物体、複数エージェントなど)を実現できない。これらは非常に複雑な世界の一部分であるためだ。単純な方法を追求する代わりにこれらの複雑な現象や仕組みを見つけ捉えられるようなメタ手法(手法を見つける手法)を探すべきだ。これらの複雑な現象を近似できる手法を見つけるのは私たちではなく、私たちが開発した手法だ。

AI研究において人がAIに取って代わられるか

この記事はAI研究者達の間で議論を巻き起こした。この考えを支持する人も多くいれば、反論する人たちもいる。例えば、米iRobot社や米Rethink Robotics社の創業者としても知られるロボット研究者のRodney Brooks氏は次の点をあげ批判した[2]。(1)現在深層学習が最も成功している画像分類では結局人が設計したCNNが重要な役割を果たしている。(2)これらの学習には人の学習とは違って大量の学習データを必要とする。(3)人の脳は20Wで動くのに対し現在のチップは数百Wを必要とする。(4)ムーアの法則は既に終焉を迎えつつある。

筆者の観点では、(1)と(2)の問題は計算性能の向上と問題の対称性も発見できるような手法の開発によって、いつか解決されるだろうと考えている。

(1)については、既にネットワーク構造や学習手法自体を探索する手法が多く登場してきている。今はまだ人手で設計したアーキテクチャが優勢だが、コンピュータ性能が今後さらに向上し探索手法が改善されれば、より多くのネットワークや最適化法が自動設計されるようになるだろう。多くの問題を解く上で重要な対称性の導入も現在は人手で設計しているが、将来的にはデータや他の知識から自動設計できるようになるだろう。

(2) の学習データを多く必要とする問題は、メタ学習などで複数タスク間の知識を共有化し、1つのタスク当たりに必要な学習データを減らすことができるだろう。現実世界のタスクは多くの共通点がみられる（物理現象、心理モデルなど）。これらを事前にモデル化できていれば、各タスクで新たに知らなければいけないことは少なくなるはずだ。

それに対し、(3)、(4) の問題が今後解決できるかは不透明だ。確かに既にムーアの法則は終焉を迎え、チップの微細化のペースは鈍り、データ転送速度の向上はTSV（through silicon via、Si貫通電極）など3次元実装の技術を用いても計算性能の向上に追いつかなくなりつつある。汎用を捨て、計算モデルに制約を加え、特化することでチップの指数的な性能向上をまだしばらく維持することはできるが、これがどこまで続けられるかは未知数である。2019年はじめの段階では1W当たり1TFLOPSは達成しつつある（例えば筆者の勤務先のPreferred NetworksのアクセラレータであるMN-Coreは半精度で1W当たり1TFLOPSを達成できる）。それに対し、人の脳のリアルタイムシミュレーションには数十PFLOPS必要と推定され、人の脳は20Wで動いているので、1W当たり1PFLOPSの計算が実現できているとされる。全く別のハードウエア、計算機構が必要になるだろうが、エネルギー当たりの計算効率ではまだ1000倍近くの改善は物理的に不可能ではないだろう。

この議論でみえてきた興味深い観点として、AIが人の仕事を奪うという現象は、AI研究それ自体で既にこの20年近く起き続けているということだ。そこでは人が考えた手法やモデルは機械が生み出したものに置き換わりつつある。

しかし人の役割が全く消えたわけではない。人はよりメタな手法を扱うようになっている。例えば、モデルを生み出す機械学習や、さらにその機械学習手法自身を生み出すメタ学習など、より抽象的な問題を扱うようになってきている。さらに学習や検証に使うデータをどのように収集し加工するか、問題をどのように設計するかはより多様化している。

1つ言えることは、AI研究ではAIをうまく使いこなしているということだ。これらによってAIの研究自体はより加速し多くの問題が解かれ、またそれ以上に新しい問題が登場している。そこでは人間中心ではないがAIによって人の可能性を広げている萌芽をみることができる。

1) http://www.incompleteideas.net/IncIdeas/BitterLesson.html
2) https://rodneybrooks.com/a-better-lesson/

（2019年5月号掲載の記事に加筆）

MN-3/MN-Core：世界で最も省電力性能に優れたスーパーコンピュータ

2020年6月、International Supercomputing Conference（ISC 2020）で発表されたスーパーコンピュータの世界ランキングにおいて、著者が所属するPreferred Networks（PFN）のスーパーコンピュータMN-3が電力当たりの演算性能（省電力性能）を競うGreen500において1位を獲得した（図5-2）。また、同時に発表された他の計算性能を競うTOP500など4部門では理化学研究所のスーパーコンピュータ「富岳」が1位を獲得し、全部門を日本発のスーパーコンピュータが独占することとなった[1]。

MN-3は深層学習向けアクセラレーターMN-Coreを内部で使っており、PFNが2018年12月にSEMICON Japan 2018で発表して以来、実装、検証を進めていたものである。

本稿では、なぜ深層学習向けアクセラレーターの開発が進められているか、MN-3はどのような特徴があるのかについて解説していく。

計算性能の向上がAIを進化させる

前節でも紹介したように、計算性能の向上とAIの進化は密接な関係にある。AIの進化において最も重要な貢献を果たしてきたのが計算性能の改善といっても過言ではない。チップの集積度が1年半で倍になるというMoore（ムーア）の法則により、計算性能が毎年指数関数的に向上し、計算当たりのコストが下がるという現象はCPUなど汎用向けチップでは終焉を迎えたが、特定用途チップ、例えばGPUなどではある程度維持されている。例えばGPUの半精度浮動小数点計算当たりのコストは過去4年間で1/10に下がっている[2]。

その一方で、OpenAIのブログ記事「AI and Compute」[2]で挙げられたように、現在最先端のAI研究に必要な計算リソースは3.4カ月倍に倍のペースで増えている。例えば2020年6月に発表されたGShard[3]は6000億パラメータからなるモデルであり、機械翻訳向けのモデル（Mixture of Expertsを使ったTransformer）を2048個のTPU v3（2pod、200PFLOPS相当）で4日で学習させている。これでも学習効率は劇的に効率化されており、その前のGPipeではこれより小さなモデルで学習に6週間必要であった。最も計算コストをかけたモデルの1つとみられるGPT-3[4]は3640PFLOPS×日（NVIDIA社のV100 1000台で数百日相当）で学習したと報告している。

こうしたトレンドは学習手法が教師あり学習手法から自己教師あり学習やシミュレーションを使った学習に代わるにつれ、学習データはいくらでも入手可能になったことが後押ししている。また、学習データが大きい場合、大きいモデルを使った方が汎化性能が高いということが実用的にも理論的にもわかってきている。

こうした学習規模、モデルの巨大化は研究的にはよいかもしれないが商業的にはどうだろうか。一度大きなコストをかけて学習できれば、その後モデルはいくらでもコピーして利用できるため、学習コストは利用分だけ償却できる。さらにはモデルの多くは学習が終われば圧縮や蒸留をしたりして小さくできる場合も多い。それならば最初から小さなモデルを学習すればよいと思うがそれは難しい。大きなモデルを利用した方が汎化能力の高い解を見つけられるような最適化が簡単であるためだ。クラウドサービスで利用する場合は大きなモデルを低価格で使うといったことも可能となる。例えばGPT-3もAPIで利用できるようになっている。Mixture of Expertsのようにモデルの一部分だけを使うconditional computationなどの概念を使えばモデル自体は大きくても推論時のコストを大幅に抑えられる。

とはいえ、学習コストは非常に大きい。計算機自身だけでなく計算に要する電力費用が高額になるためだ。例えば、GPT-3の学習1回にかかったコストは12億円（1ドル100円換算）程度とみられている[5]。多くの研究ではここまでいかないにしても、研究開発や製品開発においてはこの学習を何回も繰り返すことを考えるとコストは大きなものとなる。

そのため、深層学習向け専用アクセラレーターの開発が待望されてきた。従来のCPU（汎用目的計算）に比べて深層学習のワークロードは単純といえる。CPUでは重要だった分岐予測、深いキャッシュメモリ、TLB（仮想記憶）の制御は必要なく、逆に行列、ベクトル演算、畳み込みカーネル計算、線形代数で必要な操作が重要である（一方、GPUは

図5-2　PFNのスーパーコンピュータ「MN-3」の外観
2020年5月から稼働している。(写真：PFN)

徐々に深いキャッシュ、仮想記憶をサポートしつつある)[6]。また、深層学習は計算精度を落とすことに比較的寛容であり、半精度（FP16）やそれより低い精度でも工夫をすれば計算できる。こうした特徴にフォーカスした設計にすることで計算効率の良いチップが開発できる。Google TPU、Graphcore、Cerebras、Intel Habanaなどがある。ほかにもAlibaba社、Huawei社、Esperanto Technologies社、Groq社など世界で100近くの企業がチップを開発しているとみられる（例[7]）。PFNもそのうちの1つで、MN-Coreの開発を進めてきた。

MN-Coreの開発

ここからはPFNが開発している深層学習向け専用アクセラレーターMN-Coreについて解説する。MN-Coreはもともと2006年に、現在PFNの平木敬氏（当時東京大学教授）と神戸大学 教授の牧野淳一郎氏（当時東京大学助教授）が始めた「GRAPE-DRプロジェクト」が源流としてある[8]。PFN代表取締役社長の西川徹氏が平木研究室の学生としてプロジェクトに参加し、MN-Coreの開発チームリーダーである名村健氏が、当時開発を担当していた。GRAPE-DRは2010年にGreen500の前身のLittle Green500で1位をとり、それからちょうど10年後に再度獲得したことになる。

MN-Coreは倍精度浮動小数点演算（以下「小数点演算」は省略）で32.8TFLOPS、単精度で131TFLOPS、半精度では524TFLOPSの性能が出せるよう設計されている。電力当たり性能を重視して設計されており、半精度で1TFLOPS/Wを達成できるように設計されている。4つのダイが1つにパッケージされている。

チップは巨大なSIMD（Single Instruction Multiple Data）プロセッサであり、単一の命令ストリームを受け取り、それを複数のデータに対して並列に実行していく。また、ブロードキャスト/集約などを効率的に実現する階層的なオンチップネットワークを備えている。

近年のチップは計算能力の向上に対し、メモリ帯域向上/レイテンシ短縮が追いつかない"Compute Gap"が問題となっている。計算能力の向上が著しいMN-Coreでは Compute Gap問題が顕著となっている。ニューラルネットワークの各層の演算は、活性値とネットワークのパラメータをメモリから読み込み、行列積、畳み込み演算を行い、要素ごとの演算を行い、結果をメモリに書き込む。MN-Coreはメモリ帯域への負担を抑えるための工夫をハードウエアにもソフトウエアにも行っている。

このMN-Core上でPyTorchで書かれた深層学習モデルをそのまま動かすことができるようなソフトウエアスタックも同時に開発している。その後、ディープラーニングを使った画像認識や材料探索、またネットワークアーキテクチャ探索

などにおいてPyTorchで書かれたコードを動かし、GPUなどと比べて3～6倍の高速化を達成できたと報告している[注]。これにより、一度深層学習モデルのコードを書けば、あとは様々なチップ上（GPU、MN-Core、CPUほか）で利用できるようにすることを目指している。

MN-3とGreen500

このMN-Coreを使ったスーパーコンピュータMN-3は海洋研究開発機構（JAMSTEC）横浜研究所シミュレーター棟に設置され2020年5月より稼働している。現在、48の計算ノードからなり、各ノードは4つのMN-Coreボードを搭載し、MN-Core DirectConnectとRoCEv2で接続されている。

このMN-3を使って、TOP500、Green500に参加した。TOP500、Green500はLINPACKと呼ばれる連立一次方程式を解くベンチマークであり、TOP500は全体の性能で競い合うのに対し、Green500はTOP500に入ったシステムの中から（なので小さすぎるシステムは対象ではない）、計算当たりの電力消費量が最も少ないシステムを評価する。現在のチップの運用コストの大部分は使用電力量や熱で決まる。また、高速化のボトルネックも熱であり、計算当たりの電力消費量を下げることは経済的にも技術的にも重要な課題となっている。

MN-Coreはもともと深層学習アクセラレーター向けに作られており、TOP500で利用されるようなワークロード向けに最適化して作られてはいなかった。それでも、MN-3は21.11GFLOPS/Wを達成、最高性能は1.62e+6 GFLOPS、平均消費電力量は76.8kWであった。Green500においてTOP500のシステム中1位を獲得した。

Green500に向けては新型コロナウイルス感染の拡大に伴い、部品調達や計算機のクラスタ設営が想定通り進まず、またチップやソフトウエア開発においても多くのトラブルが発生する中、プロジェクトメンバーが一丸となり、スケジュール的にかなり難しいとみられていた状況で結果を出すことができた。今回の結果はMN-CoreやMN-3の目指す方向を強く支持してくれるものだと思う。

PFNではMN-Core上で深層学習ができる環境を整え、深層学習の研究開発を加速させていくとともに、様々な応用を見据えて開発を進めていきたいと考えている。

1) 2019 recent trends in GPU price per FLOPS, https://aiimpacts.org/2019-recent-trends-in-gpu-price-per-flops/
2) AI and Compute, "https://openai.com/blog/ai-and-compute/"
3) D. Lepikhin et al., "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding," https://arxiv.org/abs/2006.16668
4) T. B. Brown, "Language Models are Few-Shot Learners," https://arxiv.org/abs/2005.14165
5) https://venturebeat.com/2020/06/01/ai-machine-learning-openai-gpt-3-size-isnt-everything/
6) J. Dean et al., "The Deep Learning Revolution and Its Implications for Computer Architecture and Chip Design," https://arxiv.org/abs/1911.05289
7) https://github.com/basicmi/AI-Chip
8) http://grape-dr.adm.s.u-tokyo.ac.jp/
9) https://tech.preferred.jp/ja/blog/mncore-compiler-1/

注1）この後、2021年6月と2021年11月の2回、Green500において1位となっている。

（2020年9月号掲載の記事に加筆）

3

モデルと
アーキテクチャ

第6章

生成モデル

6-1

Generative Adversarial Networks：ニューラルネットを競合させ生成モデルを鍛える

GAN（Generative Adversarial Networks、敵対的生成ネットワーク）は2014年に登場し[1]、2015年後半に有効な学習手法[2]が確立されてから、急速に注目を受けるようになった。GANは特にこれまで困難だった自然画像の生成に成功し、人の顔、部屋の様子、アニメ画像、漢字、CDジャケットなど様々な種類のデータをうまく生成できる。場合によっては、それが人が描いた絵なのか、機械が生成した絵なのか分からない程度まで作れるようになっている。これまで人の特権領域と思われていた創作活動を機械が実現できる可能性が出てきた。

生成モデル（Generative Model）は、対象データがどのように生成されるのかをモデル化している。例えば、サイコロの目の生成では、1から6の目が全て等確率1/6で生成されると考えられる。生成の過程が複数の操作の組み合わせから成ることも考えられる。例えば、長さ5の文字列$T=t_1t_2t_3t_4t_5$について、それぞれの文字が1文字前だけに依存して生成される場合、

$$P(T)=P(t_1)\,P(t_2|t_1)\,P(t_3|t_2)\,P(t_4|t_3)\,P(t_5|t_4)$$

として1文字ずつ順に生成することで得られる。

機械翻訳などで使われる生成モデル

生成モデルは様々なところで利用されている。与えられたデータの尤度を構成的に計算できるようになっていれば（今回紹介するGANは直接尤度を計算できず、別の手法を組み合わせる必要がある）、データの尤もらしさを評価することができる。これは音声認識や機械翻訳において、生成された文が正しいかどうか言語モデルで利用されている。また、生成モデルはラベル無しデータに適用できることから、条件付き確率モデルと組み合わせることで、半教師あり学習を実現できる。そして何より、新しいデータを生成することができる。

これまで、データが画像や音声など高次元であり、かつ各次元が独立でなく複雑な相関を持っている場合、生成モデルを推定するのは困難だった。例えば、$S(x;\theta)>0$をパラメータθで特徴付けられている正規化されていない確率密度関数として、あるデータの確率密度$p(x;\theta)$を

$$p(x;\theta)=\frac{S(x;\theta)}{N(\theta)}$$

$$N(\theta)=\int S(x;\theta)dx \qquad (1)$$

と表した場合、積分を必要とする正規化項（分配関数とも呼ばれる）$N(\theta)$の推定が問題となる。全てのデータに対する積分が効率良く計算できない場合、$N(\theta)$や$p(x;\theta)$のθについての勾配の推定は非常に困難となる。

生成モデルと判別モデルを競わせて学習

GANは2つのニューラルネットワークを競合させることで、生成モデルを間接的に学習する。1つ目の生成モデルを表すネットワークN_{gen}は、データ集合とそっくりなデータを生成する。はじめに、zを簡単な分布（例えば正規分布$N(0,I)$）から生成し、次に決定的な関数$x=N_{gen}(z;\theta)$で、データxを生成する。zの生成だけが確率的であり、関数$N_{gen}(z;\theta)$は決定的な関数であることに注意してほしい。2つ目の判別モデルを表すネットワーク$N_{dis}(x)$は、与えられたデータがN_{gen}由来なのか、真のデータ由来なのかを判別し、N_{gen}由来と判別したなら0、真のデータ由来と判別したなら1を返すような関数である。

この2つのネットワークは次の目的関数上で競合する。

$$\mathrm{E}_{x_{data}}\left[\log N_{dis}(x)\right]+\mathrm{E}_{z}\left[\log\left(1-N_{dis}(N_{gen}(z))\right)\right] \qquad (2)$$

ただし、x_{data}は真のデータからの分布、zは先ほどの簡単な分布である。生成モデルN_{gen}は、N_{dis}をだますように(2)を最小化するように学習し、N_{dis}はN_{gen}が生成データかを見分けられるように、(2)を最大化するように学習する。例えていうと、N_{gen}は偽金を作る人、N_{dis}は偽金か本物かを見分ける警察官とみることができる。N_{gen}は警察官に見分けられないようにより巧妙な偽金を作るように鍛えられていき、N_{dis}はより些細な違いも見つけてそれを指摘し、偽金かどうかを正しく判定できるように鍛えられていく。

この2つのモデルがうまく競争しあって育っていけば、最終的には$N_{gen}(z)$は真のデータと見分けがつかないデータを作

れるようになる。つまり、N_{gen}は生成モデルとみなすことができるようになる。実際、前述の論文では、うまく学習を行えればN_{gen}は真の生成モデルに漸近するように学習できるということが証明されている。

なぜGANでの画像生成がうまくいくのか

GANを使った生成モデルがなぜここまで成功したかについての理由はいくつか考えられる。

1つ目は、前述のように高次元データの場合、サンプリングによる正規化項の勾配の推定は精度が非常に低いという問題がある。GANは正規化項に関する計算は必要なく、N_{dis}をだますように学習する。別の言い方をすれば、全てのデータを見る必要がないようにN_{dis}がどの部分に注目すればいいのかをN_{gen}に教えてくれる。

2つ目は、特に生成モデル、判別モデルにニューラルネットを使った場合、これが画像のモデル化の良い事前知識になっている可能性があるということである。例えば、判別モデルにCNN（畳み込みニューラルネット）を使った場合、CNNが備える画像の平行移動不変性の特徴を生かして判別することになる。生成においては、画像中の構成要素が平行移動しても同じように生成しやすいようにモデル化することになる。人が画像生成の良さをみる場合、数ピクセルの平行移動はほとんど気にならない。しかし、もし各次元ごとにガウシアンでモデル化し、対数尤度で評価している場合、平行移動は非常に大きな差になってしまう。

3つ目は、一般に生成モデルをニューラルネットワークでモデル化して学習する場合、勾配が計算できるように生成する直前でガウシアンなどを掛け、どのようなデータに対しても、それに対する勾配を計算できるようになっている必要がある。これにより、画像はぼやけたものが生成されやすくなる。一方でGANは、生成時には最初にガウシアンを掛けて、その後は決定的なニューラルネットを使うため、非常にシャープな画像を生成できる。

GANは高次元の確率モデルを学習する手法として非常に有効であり、生成モデル以外に利用する試みも進んでいる。例えば、データの確率分布ではなく、事後確率分布$P(z|x)$のモデル化にGANを使ったり、VAE（変分自己符号化器）と組み合わせたり、GANで獲得されたN_{dis}の方を利用したりなどである。

とはいえ、GANは学習が不安定になりがちである。現状では、GANの学習時は人が学習の状況を注意深くみて、うまく学習を進めていかないと、途中で生成側、判別側、どちらかが勝ち、もう片方がつぶれて、学習が進まなくなってしまう。今後の研究が待たれる。

1) I. Goodfellow et al., "Generative Adversarial Nets," http://arxiv.org/abs/1406.2661
2) A. Radford et al., "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," http://arxiv.org/abs/1511.06434

（2016年5月号掲載の記事に加筆）

6-2

Variational Walkback：再帰確率的ニューラルネットで生成、認識を行う

ニューラルネットワークは多くの問題を解くことができるが、より困難な問題を解くには、複雑な確率分布に基づいたサンプリングや推論ができ、尤度評価ができ柔軟性の高いニューラルネットワークが必要とされている。さらにこうしたニューラルネットワークは教師なしデータを使って学習可能であることが望まれる。

現在、確率分布を扱うニューラルネットワークは2つに大別される。1つ目は有向グラフィカルモデルに基づいた確率モデル（変分自己符号化器（VAE）、敵対的生成モデル（GAN）など）であり、データ分布に従って高速にサンプリングできるという特徴がある。2つ目は無向グラフィカルモデルから導出されるエネルギー関数に基づいた確率モデル（ボルツマンマシン、ホップフィールドネットワークなど）である。この場合、観測変数や潜在変数間の依存関係を直接、設計できるという特徴がある。しかし、サンプリングをするためには、エネルギー関数から導出されるMCMC（マルコフ連鎖モンテカルロ法）を使ってサンプリングする必要があり、時間がかかる。また、このエネルギー関数に対応するニューラルネットワークの重みは対称（$W_{ij}=W_{ji}$）であるという強い制約を課する。脳内の領域間は双結合はみられるが、重みに対称性があるとは考えられていない。そのため、脳がこのようなエネルギー関数に基づいた確率分布を扱っていることは考えられない。

確率的に遷移を計算するVW

これらに対し、最近、確率分布を扱うVariational Walkback（VW）と呼ばれる新しいモデルが提唱された[1]。VWは、RNNのように再帰結合を使って状態を更新していき、各遷移計算は決定的ではなく確率的な計算を利用する。この遷移関数（重み）は対称である制約はなく計算にノイズが含まれており脳内の計算モデルに近いと考えられる。

VWは次の時刻のデータh_{t+1}を現在のデータh_tに条件付けられた確率分布pに従ってサンプリングする。

$$h_{t+1} \sim p\,(h_{t+1} | h_t)$$

このようにして、一定時刻経った後のデータの分布がVWの表す確率分布を表す。

このVWを使って特定の確率分布を学習させる場合、誤差逆伝播法を使うことは困難である。

RNNのような再帰計算が含まれる場合でも、その計算過程を時間方向に展開することで誤差逆伝播法を使うことはできる。これをBPTT（Back Propagation Through Time）と呼ぶ。また計算グラフに確率的分布からのサンプリングが含まれる場合でも、変数変換トリックやREINFORCEなどを使いその勾配の期待値を推定することができる。

しかし、VWのようにサンプリングが繰り返された場合、勾配推定の分散は非常に大きくなってしまい、最終時刻の分布を最適化するために各時刻の遷移関数をどのように修正すればよいかがわからなくなってしまう。

VWでのデータの生成

VWモデルではデータの生成過程を次のように考える。はじめにs_Kを各次元が独立であるガウシアンや一様分布のような簡単な分布$p\,(s_K)$からサンプリングする。

$$s_K \sim p\,(s_K)$$

次のステップのデータは確率的な遷移関数pからサンプリングされる（時刻が逆順である理由は後で説明する）。

$$s_{t-1} \sim p_{T_t}\,(s_{t-1} | s_t)$$

ただし、T_tは時刻tにおける温度である。各時刻で遷移関数pの温度を変えてサンプリングする。温度が高ければ一様分布に近くなり、温度が低ければ決定的な関数に近づく。この温度は、最初は空間全体を遷移できるように大きくしておく。その後、徐々に温度を下げていき、最終的に$t=0$において、s_0がデータ分布に従って生成されるようにする。

この生成過程は次のような同時確率分布を与える。

$$p(s_0^k)=p(s_K)\prod_{t=1}^{K}p_{T_t}(s_{t-1}|s_t)$$

一方で学習データを時刻毎に破壊していく過程を考える。

$$s_t\sim q_{T_t}(s_t|s_{t-1})$$

この過程では徐々に温度を上げていく。そして、最後には一様分布やガウシアンのような単純な分布となるようにする。この破壊過程は次のような同時確率分布を与える。

$$q(s_0^k)=q(s_0)\prod_{t=1}^{K}q_{T_t}(s_t|s_{t-1})$$

このステップ数Kは確率分布から毎回選ばれるような確率変数である。

VWでは、符号化器と復号化器に同じネットワークを使う。つまり、$p=q$である。同時確率分布$p(s_0^k)$、$q(s_0^k)$は初期確率$p(s_K)$、$q(s_0)$の部分だけが異なるようにみなせる。温度を下げていけばノイズ分布をデータ分布に変え、上げていけばデータ分布をノイズ分布に変えるように遷移関数pを学習していく。

学習ではデータ分布$q(s_0)$からスタートし、崩壊過程を使って次の時刻のデータをサンプリングする。そして、元のデータに戻るように学習する。

$$s_t\sim p_T(s|s_{t-1})$$

$$\theta\leftarrow\theta+a\frac{\partial\log p_T(s_{t-1}|s_t)}{\partial\theta}$$

このように各時刻毎に1つ前の時刻の状態に戻るように学習するため、BPTTをする必要がない。

偽のモードを見つけて修正

このアイディアはデノイジング自己符号化器と似ている。デノイジング自己符号化器ではデータに一様なノイズを加えた上で、それが元に戻るように学習する。このとき、学習が進むと、元に戻るような関数はデータの対数尤度の勾配を求めていることに対応することが知られている。一方VWでは一様なノイズではなく、復元と同じ関数を使って遷移させる。

ノイズにも同じ関数を使うことで、モデルが誤って高い確率を割り振っているモード（spurious modes：偽のモード）を見つけ、それを修正することが期待される。遷移関数に従って遷移していくと、誤って高い確率を割り振ってしまった偽のモードに到達し、そこから元のデータ分布の領域に戻るように学習されるからである。

このVWの学習則はエントリーシティクスのように見えるが、変分法による最尤推定として定式化することができる。観測変数をv、潜在変数をhとしたとき、潜在変数を周辺化した対数尤度は次のようになる。

$$\ln p(v)=\ln\sum_h p(v|h)\,p(h)$$

$$=\sum_h p(h|v)\ln\frac{p(v,h)}{q(h|v)}+D_{KL}[q(h|v)\|p(h|v)]$$

ただし、qはpとは異なるかもしれない別の確率分布である。

ここで、$v=s_0$、$h=s_1^k$とした場合、

$$\ln p(v)=\sum_{s_1^k}q(s_1^k|s_0)\ln\frac{p(s_0|s_1^k)\,p(s_1^k)}{q(s_1^k|s_0)}$$

$$+D_{KL}[(q(s_1^k|s_0)\|p(s_1^k|s_0)]$$

この第1項は変分下限Lと呼ばれ、$D_{KL}[(q(s_1^k|s_0)\|p(s_1^k|s_0)]\geq 0$であることから、$\ln p(v)\geq L$のように対数尤度の下限を与える。学習の始めに変分下限を上げるようにpのパラメータについて最大化する。これは、先程のようにqに従ってサンプリングした上でそれが元に戻るように$\log p$を最大化することで達成される。次に$q=p$とすることで、第2項のKLダイバージェンスを小さくする。この2つ目のステップは第1項にも影響があり、必ずしも対数尤度のパラメータqについての勾配には対応していない。この2つのステップを繰り返すことで、対数尤度を最大化する。

統計物理の準静的過程と関連

さらに、このKLダイバージェンスは統計物理の準静的過程と関係があり、生成または崩壊過程で温度をゆっくり変化させていった場合は小さくできる。このKLダイバージェンスは統計物理の準静的過程におけるフリーエネルギーの差と等しくなる。準静的過程において状態を速く変化させない場合はこの差よりも余分な仕事をする必要があり、その余分な仕事は熱として環境に放出される。

同様にこのモデルを使って速く状態を変えたい場合（$q(s_0)$から$p(s_K)$に速く到達する、または、$p(s_K)$から$q(s_0)$に速く到達する）はKLダイバージェンスが大きくなってしまう。統

計物理の準静的過程では多くの研究がなされているため、それらの研究成果を変分法の学習に利用できるかもしれない。

夢の仕組みと関連する可能性

興味深いことに、このVWは、夢の仕組みと関係するかもしれないと指摘されている。脳においてシナプス前ニューロンがシナプス後ニューロンより少し前に発火した場合にシナプスが強化される学習則をSTDPと呼ぶ。これと同じ条件でシナプスが弱まる学習則を逆STDPと呼ぶ。日中、起きている時は、体験した状況が記憶されるようにこのSTDPによって重みが強化される。一方、この強化の結果、経験していないのに誤った記憶が作り出されてしまう可能性がある（上記のspurious modes）。

夢はこのような誤った記憶であるspurious modesを遷移させることで見つけ出し、そこに到達しないように逆STDPによって学習しているとみなすことができる。これはVWの学習と一致する。夢は起きている時の経験を反映しているものの、大抵は破綻しており、おかしな経験になる現象とも一致する。

こうした確率的な遷移関数を使った学習は今後重要になると考えられる。このVariational Wallbackとよく似た拡散モデルが2021年頃から生成モデルにおいて重要となってきており、将来の予測、自然言語処理や強化学習などでこうした技術が必要となると考えられる。

1) A. Goyal et al., "Variational Walkback: Learning a Transition Operator as a Stochastic Recurrent Net," NIPS 2017.

（2018年2月号掲載の記事に加筆）

Glow：可逆な生成モデル GANより安定的に学習できる尤度ベースの手法

与えられたデータセットの分布に従ってデータを生成するように学習する生成モデルは、画像や音楽などデータを生成するためだけでなく、観測から現実の世界モデルを構築し、その上でエージェントが学習や計画を立てられるようにしたり、データの意味を捉えた特徴を教師なしで学習できるため、近年非常に注目を集めている。

画像や音声など高次元データの生成モデルとしては大きく尤度ベースの手法と敵対的生成モデル（GAN：Generative Adversarial Network）を使った手法に分けられる。尤度ベースの手法はさらに、WaveNetなどの自己回帰モデル（Auto Regressive Model）、尤度の変分下限を最適化する変分自己符号化器（VAE：Variational AutoEncoder）、フローベースの生成モデルに分けられる。

フローベースの生成モデルは潜在変数を単純な事前分布 $p(z)$（ガウシアンや一様分布）から生成し、それに可逆な変換を繰り返し適用していき目標の分布 $p(x)$ からのサンプルを得る。フローベースの生成モデルは次のような特徴がある。

(1) データの正確な尤度が評価できる。これに対し、VAEは近似である変分下限しか得られない。

(2) データの生成過程全体が可逆な変換のため、データから潜在変数への正確な変換ができる。これを推論または符号化とも呼ぶ。GANには推論の仕組みは備わっておらず、自己回帰モデルには潜在空間が存在しない。

(3) 生成、推論ともに並列計算が可能であり、現在のGPUなど並列プロセッサで高速に生成可能である。これに対し自己回帰モデルはデータの各次元の値を逐次的に生成するため、並列処理ができず生成が遅い。

(4) 潜在変数が他のタスクにも有効である。これに対し自己回帰モデルは潜在空間が存在せず、GANは学習の仕方から全てのデータ点に有効な潜在変数が割り当てられる保証が無い。

(5) 逆変換が可能なため、誤差逆伝播時に途中の活性値も逆向きに順に生成できるため、途中の活性値を保存しておく必要がない。これにより学習時に必要なメモリ量を層数によらず一定にできる。他の生成モデルは逆変換はできない。

これらの特徴を備えるため、フローベースの生成モデルは有望視されてきたが、使える変換に制限があるため他の生成モデルに比べて生成データの品質が低いという問題があった。2018年に米OpenAIのDurk Kingma氏（VAEやAdamの作者）らが、2016年に発表されていたフローベースの生成モデルであるRealNVP[1]を改良したGlow[2]を発表した。Glowはフローベースの生成モデルでありながら最新のGANに匹敵するような品質の画像を生成できることを示した。

以下にRealNVPとGlowの手法について解説する。前述のようにフローベースの生成モデルでは潜在変数をガウシアンなど単純な分布から生成し、$z \sim p(z)$、次に可逆な変換 g を使って $x = g_\theta(z)$ のようにデータを生成する。ここで θ は学習対象のパラメータである。生成時の変換 g は可逆であるため、逆変換 f が存在し、$z = f_\theta(x) = g_\theta^{-1}(x)$ が成り立つ。この f を使って与えられたデータに対応する潜在変数を正確に推論できる。以降 θ は省略する。可逆変換は、一般に複数の可逆変換が連なったものであり、$h_1 = f_1(x), h_2 = f_2(h_1), \ldots, z = f_K(h_{K-1})$ のように表される。このような可逆な変換の連なりを（正規化）フローと呼ぶ。

次に、可逆変換によって密度関数がどう変わるかを説明する。例えば、一次元変数 z の確率密度が $0 \leq z \leq 2$ では $p(z) = 1/2$、それ以外は $p(z) = 0$ であり、変換が $x = g(z) = 3z$ の場合、値が3倍に引き伸ばされているので確率密度はその $1/3$ をかけて $p(x) = p(z) * 1/3$ とする必要がある。これを多次元の変数に拡張した場合、変数変換公式により、確率密度関数が $p(z)$ である変数 z を可逆関数で $x = f(z)$ と変換した時、$\log p(x) = \log p(z) + \log|\det(dz/dx)|$ という式が成り立つ。ここで dz/dx はヤコビアン行列であり、detは行列式である。ヤコビアン行列の行列式は元の空間での体積が変換後に（符号付きで）何倍になったかを示す。ここでヤコビアンは dx/dz でなく、dz/dx であり、$|\det(dz/dx)| = |\det(dx/dz)|^{-1}$ であることに注意する。

このヤコビアン行列の行列式は高次元の場合、非常に複

雑な式になりうるが、ヤコビアン行列が三角行列(対角成分より上または下半分が全て0であるような行列)の場合、その行列式は対角成分の積となり簡単に計算できる。この変数変換公式を使い、データxの対数尤度は次のように求められる。

$$\log p(x) = \log p(z) + \sum_{i=1}^{K} \log |\det(dh_i/dh_{i-1})|$$

フローベースの生成モデルで扱う変換は、可逆変換で逆変換が容易に求まり、また変換のヤコビアン行列の行列式が高速に求まることが望ましい。RealNVPはこれらを満たす変換としてAffine Coupling Layerを提案した。

この変換では、はじめに入力をxをチャンネルを分割するなどして2つの変数x_a, x_bに分割し、次にx_bからニューラルネットワークなど任意の変換NNを使ってスケールsとバイアスtを計算し$(s,t) = NN(x)$、次に$y_a = x_a \odot \exp(s) + t$を計算し、$y_b = x_b$とする。最後に$y_a$と$y_b$を結合して$y$を得る。この$x$から$y$への変換のヤコビアン行列$\frac{dy}{dx}$は次のような上三角行列となり、行列式は$\sum_i \exp(s_i)$と簡単に求まる。

$$\frac{dy}{dx} = \begin{pmatrix} \mathrm{diag}(\exp(\mathrm{s})) & \frac{dy_a}{dx_b} \\ 0 & I \end{pmatrix}$$

また、このAffine Coupling Layerは逆変換が容易に求まる。実際$x_b = y_b$であり、x_bを使ってt, sは順変換と同じようにして求めた上で、$x_a = (y_a - t) \odot \exp(-s)$と求められる。このAffine Coupling Layerでは変数の一部しか変換しないが、次元間の置換などの操作を加え別のAffine Coupling Layerと組み合わせれば全ての次元が変換されるようにできる。

GlowはこのRealNVPを次の2点で改良している。1つ目はActnormと呼ばれる正則化である。高解像度の画像生成を学習する場合、メモリ容量の問題からバッチサイズが1程度しか使えない。そのため、ミニバッチの統計量を必要とするバッチ正規化がそのままでは使えない。Actnormは代わりに最初にミニバッチデータの統計量を使って各チャンネルの出力が平均0、分散1になるようなスケールとバイアスを求め、後はこれらを通常の学習可能なパラメータとして更新する。

2つ目は可逆な1×1畳み込み演算である。RealNVPではチャンネル方向に情報を混ぜるように固定の置換を使っていた。Glowは置換を一般化した1×1畳み込み演算を使い変換の表現力を上げた。1×1畳み込み演算の行列式、逆変換は直接計算する。この場合のコストはチャンネル数がcの時、$\mathcal{O}(c^3)$である。それに対し、高さh、幅wの特徴マップの畳み込み演算の計算量が$\mathcal{O}(hwc^2)$であるのに対し行列式、逆変換のコストは許容できる。

Glowを使って顔画像の256×256など高解像度の画像生成を直接学習することができた。この際、確率密度の温度Tは$p_T(x) \propto p(x)^{T^2}$であるが、温度を1よりも低くしてサンプリングをした。これにより確率が高いところからのみ生成できるようにした。一方で温度が低すぎると生成する画像のバリエーションが少なくなってしまうため、$T = 0.7$ぐらいが画像の質とバリエーションの両方を保つことができると示している。

尤度ベースの手法はGANベースの学習に比べて安定的に学習でき、また性能も尤度で直接評価できることから工学的には優れている手法である。今後フローベースの生成モデルも選択肢の一つとして入ってくるだろう。

1) L. Dinh et al.,"Density Estimation using Real NVP,"ICLR 2017.
2) D. K.Kingma et al.,"Glow: Generative Flow with Invertible 1×1 Convolutions," https://arxiv.org/abs/1807.03039

(2018年9月号掲載の記事に加筆)

自己注意機構：Self-Attention 画像生成や機械翻訳など多くの問題で最高精度

ニューラルネットワークはあらかじめ設計されたネットワーク構造に従ってデータが入力から出力に向かって計算されながら伝搬していく。多くの問題では、事前知識を使って構造を設計することで性能を上げることができる。例えば、畳み込みニューラルネットワーク（CNN）は、画像は近い位置にある情報が関係があるという事前知識を使って、近い位置にあるニューロン間のみをつなぐことでパラメータ数を減らし、特定のモデルが学習されやすいようにして汎化性能を上げている。このような事前知識は帰納バイアスとも呼ばれ、学習が成功するかの重要な要素である。しかし、データの流れ方は学習によって決定し、データに合わせて変わることが望ましい。

自己注意（Self-Attention）機構[1,2]は、データの流れ方自体を学習し決定するような方法である。もともとRNN向けに提案されたが、CNNなど他のニューラルネットワークにも利用されている。自己注意機構は強力であり機械翻訳[3]、質問応答[4]、画像生成[5,6]など、多くの問題で最高精度を達成している。自分自身の途中の計算結果を注意対象とし、そこから読み込むことからこの名がついている。

自己注意機構は、入力系列 $\mathbf{x}=\{x_1,x_2,\ldots,x_n\}$ を出力系列（必ずしも同じ長さである必要はない）$\mathbf{y}=\{y_1,y_2,\ldots,y_n\}$ へと変換する関数である。系列中の各要素はベクトルである。

始めにスケール化内積注意機構と呼ばれる自己注意機構を説明する。各要素 x_i 毎に要素を入力としクエリ q_i、キー k_i、値 v_i の3つのベクトルを計算する。クエリとキーの次元数は同じ d_k であり、値の次元数は d_v とする。このクエリ、キー、値を行ごとに並べて作られた行列を Q,K,V と おく。このとき、スケール化内積注意機構は次のように計算される。

$$\mathrm{Attention}(Q,K,V)=\mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

この右辺は行ごとにsoftmax関数を適用した結果である。

この式の意味を解説する。はじめに各要素毎にクエリ q_i と似ているキー k_j を探す。似ている度合いとして内積 $\langle q_i,k_j\rangle$ を使う。この度合いに従って値を読み込むようにするが、足して1になるようにsoftmax を適用する。しかし、内積をそのまま使うと一部の値だけが非常に大きくなり内積値が最大だった要素以外に勾配が発生しない問題が起きる。これは、平均 0、分散 1、次元数が d_k のランダムなベクトル同士の内積は平均 0、分散 d_k となることからも分かる。そのため、$1/\sqrt{d_k}$ を乗じた後にsoftmax を適用する。そして、似ている度合いに比例して値 v_j を読み込んで足し合わせ、それを出力とする。

この注意機構は、クエリとキーが似ているかどうかで、どの要素の値を読み込むかどうかを制御していることに注意されたい。もし、学習の結果、その要素から値を読み込んだ方がよければ対応するクエリとキーは近づくように更新され、もし読み込まない方が良ければクエリとキーは離れるように更新される。こうして、どの要素を読み込むかどうか、データのルーティングが学習で自動決定される。

この仕組みは、1つの読み込みヘッド（キー）を使って各場所から値を読み込んでいるようにみなすことができる。これを拡張し、複数の読み込みヘッドを使う場合を考えられる。

$$\begin{aligned}&\mathrm{MultiHeadAttention}(Q,K,V)\\&\quad=\mathrm{Concat}(head_1,head_2,\ldots,head_k)W^O\end{aligned}$$

$$head_i=\mathrm{Attention}(QW_i^Q,KW_i^K,VW_i^V)$$

ただし、W_i^Q,W_i^K,W_i^V,W^O は学習対象のパラメータ行列である。ここでは k 個のhead を使って読み込んだ結果を最終的に統合して出力を決定する。パラメータ行列のサイズを $d_k\times d_k/k$ のようにして、Attention を k より小さい次元数で計算するようにしておけば、全体の計算量はオリジナルのAttention と変わらないようにすることができる。実験では $k=8$ などが使われている。

CNNのように局所的な箇所から読み込む場合と違って、自己注意機構は全ての位置から情報を対称的に読み込むようになっており、位置情報が使えない。そのため、入力に位置情報を表すチャンネルを追加する。例えば、$\mathrm{PE}_{pos,2i}=\sin(pos/10000^{2i/d_{model}})$ のようなチャンネルを使う。

一般に、自己注意機構に位置ごとの非線形変換を組み合わせたのが1つの計算ブロックとして扱われる。例えば、Transformer[2]では、自己注意機構の後に2層の総結合層を用いて、間にReluを1回適用する。これはカーネルサイズが1×1の畳み込み層を2つ使うのと同義である。また、スキップ接続や正規化、Dropoutなどの正則化が適用される。

自己注意機構の大きな特徴はCNNやRNNと違って遠い位置にある情報が1回の計算ステップで読み込むことができることである。また、全ての位置の計算が並列にでき、特に最適化がしやすい行列演算で実現できる。

この自己注意機構に対し、さまざまな改良手法が提案されている。Universal Transformer[4]は、同じ自己注意機構を繰り返し適用する。同じ変換を繰り返し適用するというのはRNNと同じであるが、Universal Transformerの場合、系列全体を繰り返し更新するという部分が異なる。Universal Transformerは機械翻訳や質問応答で従来手法を超える性能を出したと報告している。

Weighted Transformer[5]はMultiHeadAttentionの改良を提案している。TransformerはMultiHeadAttentionの後に位置ごとの非線形変換を適用していたが、Weighted TransformerはHead毎に別の非線形変換を適用した後にHeadの情報を統合することを提案している。また統合する際はConcatではなく重みづけ和とし、重みはパラメータとして学習で決定される。これは複数のエキスパートを用意し、どのエキスパートを使うのか自体を学習するMixture of Expertとも関連する手法である。Weighted Transformerは機械翻訳の最高精度を達成している。

自己注意機構の問題点は計算量が大きいことである。特に画像のように注意対象が多くなればなるほど計算量が大きくなる。例えば全ての画素を対称に注意を計算した場合、計算量は画素数の二乗かかることになる。そのため、注意対象を重要な要素だけに絞ることで計算量を減らすことが望ましい。全ての候補に0より大きい注意を与えるのをSoft Attentionと呼び、一部の要素だけ0より大きい注意を与えるのをHard Attentionと呼ぶ。計算量を減らせるのはHard Attentionだが、この場合、注意をあてなかった要素は試されず、勾配が発生しないという問題がある。そのため、強化学習と同様に新しい要素を試してみなければならないという利用と探索のジレンマが発生する。この問題に対し、ベクトルのノルムが大きいものだけを残すことでHard Attentionを達成する手法[7]は単純でありながら、高精度を達成でき、自己注意機構を効率的に実現することができ有望である。

自己注意機構は記憶の仕組みとも大きく関係する。長期記憶を扱うMemory NetworkやDifferentiable Neural Computerなどは、過去の経験に対し自己注意機構と同じ仕組みで、どの経験が関係するかを求めた上で読み込むようになっている。注意対象が過去の経験か、現在の計算の途中結果かということが異なるだけであり、長期記憶と自己注意機構は共通の仕組みを利用することができる。そのため今後は、自分自身の計算結果だけでなく、長期記憶からも同時に読み込むといったことが試されていくだろう。

1) Z. Lin et al., "A Structured Self-attentive Sentence Embedding," ICLR 2017.
2) A. Vaswani et al., "Attention is All You Need," https://arxiv.org/abs/1706.03762
3) K. Ahmed et al., "Weighted Transformer Network for Machine Translation," https://arxiv.org/abs/1711.02132
4) M. Dehghani et al, "Universal Transformers," https://arxiv.org/abs/1807.03819
5) N. Parmar et al., "Image Transformer," ICML 2018.
6) H. Zhang et al., "Self-Attention Generative Adversarial Networks," https://arxiv.org/abs/1805.08318
7) M. Malinowski et al., "Learning Visual Question Answering by Bootstrapping Hard Attention," https://arxiv.org/abs/1808.00300

(2018年10月号掲載の記事に加筆)

連続ダイナミクスを表現可能なニューラルネットワーク

ニューラルネットワークは有限回の変換の組み合わせで構成され、例えば10層のニューラルネットワークは入力を10回変換した結果として表される。これに対し、入力に連続時間の変換を適用し、例えば3.6回分変換といった変換を扱えるNeural ODE[1]が提案された。Neural ODEはNeurIPS 2018のベストペーパーに選ばれている。これについて解説する。

ニューラルネットワークの層のいくつかは変換した値を入力に足しこんでいく形をとる。

$$h_{t+1} = h_t + f(h_t; \theta_t)$$

この f は1層以上から成るニューラルネットワークで表現される。現在の画像認識器の主流となっているResNetやLeaky Unitを使ったRNN、生成モデルの正規化フローなどがこの形をとる。この式は連続な関数を時間 t についてオイラー法で離散化した時の更新式とみなせる。このステップ幅を極限まで小さくしていくと、次のような常微分方程式(ODE：ordinal differential equation)が得られる。

$$\frac{dh(t)}{dt} = f(h(t), t; \theta)$$

この場合、関数 f は入力が $h(t)$, 時刻 t の時の傾分を与えているとみなせる。入力を時刻 0 における値 $h(0)$ とし、上記の常微分方程式に従い、時刻 T における値 $h(T)$ を出力とした関数を考える。このような常微分方程式を使った関数を層として利用したニューラルネットワークをNeural ODE[1]と呼ぶ。

多くの利点があるNeural ODE

このNeural ODEは通常のニューラルネットワークと比べて多くのメリットがある。

(1) メモリ効率が良い。ResNetは各層毎に異なるダイナミクスを表しているとみなせるが、もし隣り合う層が似たダイナミクスを表す場合はそれらの層は1つにまとめることでパラメータ数を減らせる。Neural ODEは自然に似た層を融合した形で表現できる。RNNのようにパラメータを共有した時間変化する変換を適用していると考えても良い。

(2) 速度と精度のトレードオフをとれる。Neural ODEはODEの解を求める際に、通常のRunge-Kutta法などを使うが、その際に評価点数を変えることができる。精度が必要な場合は評価点数を増やし、必要でない場合は評価変数を減らせばよい。

(3) 可逆関数であり逆関数を計算できる。ODEで時間を T から 0 に向かってたどれば出力から入力を計算できる。後述するがパラメータについての勾配を求める際に時間方向を逆にたどるが、その際に計算の途中結果 (活性値) はその場で計算して復元できるため、保存しておく必要がない。

(4) 任意の変換を使っても確率密度を効率的に求められる。正規化フローを使った場合、確率密度の計算には隠れ層のユニット数が n の時 $O(n^3)$ 時間必要であり、そのヤコビアンの行列式が効率的に求まるような計算クラスにするとと $O(n)$ 時間で求まる。Neural ODEは任意の変換を用いて $O(n)$ 時間で求められる。

(5) ODEで表されるような問題などは直接扱え、その場合、外挿などの問題を解ける。

誤差逆伝播法を連続ダイナミクスに拡張

Neural ODEの実現で鍵となるのが、誤差逆伝播法の連続ダイナミクスへの拡張であり以下に説明していく。ODEの解 $z(t_1)$ を入力とした損失関数 L は次のように表される。

$$L(\mathbf{z}(t_1)) = L\left(\mathbf{z}(t_0) + \int_{t_0}^{t_1} f(\mathbf{z}(t), t, \theta) dt\right)$$

この損失関数の入力やパラメータについての勾配 ($\frac{dL}{d\mathbf{z}}$, $\frac{dL}{d\theta}$) を求めることが目標である。まず、各時刻で損失の隠れ状態 $\mathbf{z}(t)$ についての勾配 $\mathbf{a}(t) = \frac{\partial L}{\partial \mathbf{z}(t)}$ を求める。この値 $\mathbf{a}$ は随伴変数と呼ぶ。この随伴変数は次のようなダイナミクスを持つことが示せる。

$$\frac{d\mathbf{a}(t)}{dt} = -\mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t), t, \theta)}{\partial \mathbf{z}}$$

これは、誤差逆伝播法の連続ダイナミクスへの一般化とみなすことができる（誤差逆伝播法では $\mathbf{a}$ が誤差 $\mathbf{e}$ に対応し、誤差に各層のヤコビアン $\frac{\partial f(\mathbf{z}(t),t,\theta)}{\partial \mathbf{z}}$ をかけて逆方向に伝播する）。このODEを $t = T$ から $t = 0$ に向かって逆向きに解く。この際、途中の $\mathbf{z}(t)$ の値も必要となるが、それも同時にODEで求められ、保存しておく必要はない。

この随伴変数が求まれば、損失のパラメータについての勾配は次のように求められる。

$$\frac{dL}{d\theta} = -\int_{t_1}^{t_0} \mathbf{a}(t)^T \frac{\partial f(\mathbf{z}(t),t,\theta)}{\partial \theta}$$

ベクトルとヤコビアンの積 $\mathbf{a}^T \frac{\partial f}{\partial \mathbf{z}}$ は、出力の勾配に $\mathbf{a}$ を設定した上で自動微分を適用すれば、ヤコビアンを明示的に計算せずに前向き計算 f の倍程度の計算量で効率的に求めることができる（vjp：matrix-free vector-Jacobian productと呼ばれる）。

これら随伴変数と損失関数の入力、パラメータについての勾配は同時にODEを解くことで計算できる。このようにNeural ODEでは勾配を伝播する際に途中結果を保存しておく必要がない。

連続正規化フロー

Neural ODEは生成モデルとしても非常に有効である。第6章 第3節でも説明したように入力 $\mathbf{z}_0$ を可逆な関数 f で変換して $\mathbf{z}_1 = f(\mathbf{z}_0)$ が得られた時、その対数尤度 $\log p(\mathbf{z}_1)$ は元の尤度 $\log p(\mathbf{z}_0)$ を使って、次のように表される。

$$\log p(\mathbf{z}_1) = \log p(\mathbf{z}_0) - \log \left| \det \frac{\partial f}{\partial \mathbf{z}_0} \right|$$

このヤコビアンの行列式は、$\mathbf{z}_0$ の空間での体積が、$\mathbf{z}_1$ の空間でどのくらい拡大したかを表しており、拡大した分、密度を小さくしている（ log をとっているので、割り算ではなく引き算となっている）。生成モデルを学習する際はこの計算された対数尤度を最大化するようなパラメータを求めればよい。

この行列式の計算量はノード数が n の時、$O(n^3)$ と大きいため、RealNVPやGlowなどでは効率的に行列式が求まるような変換を扱う手法が提案されていた。一方、連続ダイナミクスを使った場合はこれを $O(n)$ で求めることができる。これについて以下に説明していく。

変数 $\mathbf{z}(t)$ は $p(\mathbf{z}(t))$ を確率密度とする確率変数とする。このとき、確率密度の時間当たりの変化量は次のように表せられる。

$$\frac{\log p(\mathbf{z}(t))}{\partial t} = -\mathrm{tr}\left(\frac{df}{d\mathbf{z}(t)}\right)$$

さきほどの正規化フローの時は行列式 det だったのに対し、連続ダイナミクスを使用した場合はトレース tr であることに注意してほしい。この計算ではヤコビアン行列はそれぞれ独立に計算するため $O(n^2)$ 時間が必要である。

FFJORD（Free-form Jacobian of Reversible Dynamics）[2]は任意の変換を利用しても効率的に確率密度を計算できる手法を提案した。はじめに任意の行列のトレースは、ノイズベクトル $\epsilon \sim p(\epsilon)$ を使って次のように求められる。

$$\mathrm{tr}(A) = \mathrm{E}_{p(\epsilon)}\left[\epsilon^T A \epsilon\right]$$

そしてベクトルとヤコビアン間の積はvjpを使うことでまとめて $O(n)$ 時間で評価できる。$p(\epsilon)$ としては正規分布や、ラデマッハ分布などを利用できる。

このFFJORDは任意の変換を利用でき、尤度を正確に求められ、高速にサンプリングすることができる。これに対し、VAEは変分下限しか得られず厳密な尤度は計算できない。GANは尤度を計算できず、入力 $\mathbf{x}$ から潜在変数 $\mathbf{z}$ への変換は別の符号化器を学習する必要があり、それは容易ではない。正規化フローは尤度が求められ符号化も可逆変換なので容易だが、行列式が高速に求められる限定された変換しか利用できず表現力は限定的である。

このFFJORDを使った生成モデルはGANに比べると学習が容易であり、尤度はVAEなど他の手法を超える性能が得られたと報告されている。

今後はライブラリでもサポートの見込み

このNeural ODEやFFJORDの問題として、複雑なダイナミクスを学習した場合、ODEを解く際に多くの評価点が必要になってしまう問題がある。これに対しては次元を拡張した空間上でODEを解くことで、ダイナミクスを単純化し、評価点数を小さくできることが報告されている[3]。

今はまだ既存のODEソルバーを利用しているが今後は特化したより効率的な手法が登場してくるだろう。また、今後はディープラーニングフレームワークやライブラリなどでこの

連続ダイナミクスをサポートし、他の層と同じように簡単に使えるようになると考えられる。

1) R. T. Q. Chen, et al.,"Neural Ordinary Differential Equations," NeurIPS 2018.
2) W. Grathwohl, et al., "FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models," ICLR 2019.
3) E. Dupont, et al., "Augmented Neural ODEs," https://arxiv.org/abs/1904.01681

(2019年6月号掲載の記事に加筆)

6-6 正規化層：ニューラルネットワークの学習の安定化、高速化、汎化に大きな貢献

ニューラルネットワークの学習で重要な役割を担っているのが正規化層（Normalization Layer）である。正規化層はニューラルネットワークの表現力の維持、学習の安定化、汎化性能の向上に貢献し、現在の深層学習の成功の主因の1つとみなせる。

正規化層は入力分布を元に入力を正規化した結果を返すような層である。多くの正規化層は入力分布の統計量を推定し、それを元に出力分布が次元毎に平均0、分散1となるように正規化した上で、それを次元毎に線形変換した結果を返す。正規化層は総結合層や畳み込み層の直後（活性化関数の直前）、スキップ接続のブロック内の最初、最後などで使われる。今回はなぜ正規化が重要なのかを説明した後に、現在使われている正規化層、新しい正規化層について紹介していく。

なぜ正規化層が重要か

正規化層は様々な役割を持っている。ここでは代表的な役割を3つ紹介する。

1つ目はニューラルネットワークの非線形性の表現力を保つ役割である。現在のニューラルネットワークは活性化関数として$\mathrm{ReLU}\ f(x)=\max(x,0)$のような区分線形関数を使う。$\mathrm{ReLU}$の微分は入力が正の時1、負の時0となる。ニューラルネットワークでは、誤差逆伝播時に誤差には活性化関数の微分が繰り返し掛けられる。ReLUを使うことで勾配が発散したり、消失したりしにくくなるという非常に大きな利点がある。

一方でReLUは$x=0$をまたぐ位置でしか非線形性を生み出さない。そのため、もし活性化関数への入力が常に正であったり負であるように偏っている場合は、活性化関数は線形関数になってしまう。入力分布が$x=0$をまたいで正にも負にも分布することで、非線形を生み出すことができる。

2つ目は勾配降下法による学習を高速にする役割である。例えば、ニューラルネットワークより単純な線形回帰モデル$y=\mathbf{w}^T\mathbf{x}$を使って誤差$l=\frac{1}{2}\|y-y^*\|^2$（y^*は正解）を最小化するような問題を考えてみよう。このとき、yについての勾配を$e_y=\frac{\partial l}{\partial y}=(y-y^*)$とした時、重みについての勾配は$\frac{\partial l}{\partial \mathbf{w}}=e_y\mathbf{x}$と表される。もし入力の値がすべて正であるならば、重みについての勾配の符号はすべて同じであり勾配降下法による最適化は、すべての次元の符号が同じ方向にしか進めない[1]。例えば、パラメータがm次元だとすれば2^mある象限のうち2つの象限の方向にしか進めないことになる。

この議論はそのままニューラルネットワークにもあてはまり、各層の入力が偏っている場合、勾配降下法による収束が遅くなってしまう。また、入力の各次元のスケールが大きく異なる場合も、大きなスケールを持つ次元が更新幅も大きくなり、スケールが小さな次元は無視され使われなくなってしまう。結果としてスケールが小さな次元は無視されてしまう。入力を正規化しておくことで勾配降下法による収束を速めることができる。

3つ目は汎化性能を上げる役割である。元々、正規化層はニューラルネットワーク内の各層の分布が変わっていく共変量シフトを防ぐことで学習を安定化させるという目的で導入された[2]。しかし現在ではこの共変量シフトを抑えることによる安定化の効果は限定的であり[3]、目的関数をなだらかにすることで安定化させていることが分かっている。

正規化層を使うことで活性値や勾配が安定化し、勾配降下法の学習率を大きくしても発散せずに学習できるようになる。大きな学習率を使うことで汎化性能が低いシャープな解にはまることがなく、汎化性能が高いようなフラットな解に到達しやすくなる[3-5]。フラットな解は汎化性能が高いことが分かっている。

代表的な正規化層

正規化層を紹介する前に、ニューラルネットワークの各層の入力である特徴マップの形状を表すテンソルについて簡単に説明しよう。画像データを扱う場合は一般に(N,C,H,W)という形を持ったテンソル$\mathbf{x}$を特徴マップとして扱う。Nはデータ数、Cはチャンネル数、Hは特徴マップの高さ、Wは特徴マップの幅である。このデータは$N*C*H*W$個の要素からなり、$\mathbf{x}[n,c,h,w]$でn番目の

データの位置(h, w)にあるc番目のチャンネルの値を表す。

データを正規化するためには入力分布を最初に求める必要がある。理想は訓練データ全体から分布を推定し、その統計値に基づいて正規化することだが、学習が進み下層のパラメータが変わるにつれて特徴マップの分布も変わってしまい、そのたび訓練データ全体を走査することは計算量的に現実的ではない。

そこで、全ての正規化層は毎回現在のバッチのみから分布を推定する。正規化層はテンソルのとの集団から統計量を推定するかで様々な変種がある。

バッチ正規化

最初に導入されたバッチ正規化（Batch Normalization： BN）[5]は最も広く使われている正規化層である。(N, H, W)の軸に沿って、各チャンネル毎cに平均m_cと分散v_cを求める。例えば平均は、$m_c = \sum_{i,h,w} \mathbf{x}[i, c, h, w]/(NHW)$と求められる。ここでの下付き字$c$はこれがチャンネル毎に異なることを表している。これらの統計値を使ってチャンネルcの要素$\mathbf{x}$を次のように正規化する。

$$z = \frac{x - m_c}{\sqrt{v_c}}$$

さらに、学習可能なパラメータであるγとβで変換する。これにより正規化層の出力は元の入力も必要とあれば使えるだけでなく、そのスケールやバイアスも自動的に制御できるようになる。全体の式は次の通りになる。

$$\mathrm{BN}(h, m_c, v_c; \gamma_c, \beta_c) = \gamma_c \frac{x - m_c}{\sqrt{v_c}} + \beta_c$$

なお、統計量m_cやv_cについても誤差逆伝播することに注意してほしい。

層正規化

層正規化（Layer Normalization：LN）[6]は(C, H, W)の軸に沿ってN個の統計量を求めて、サンプル毎に正規化をする。層正規化はサンプル毎の統計量が大きく変わるような場合に有効であり、RNNやTransformerなどで利用されている。現在Transformerが注目されていることもあり、層正規化も広く使われる様になってきている。

事例正規化

事例正規化（Instance Normalization：IN）[7]は(H, W)の軸に沿ってNC個の統計量を求め、事例-チャンネル毎に正規化する。しかし統計値を推定するサンプル数が少ないため（$H * W$個）、安定化しにくい問題があった。この問題を解決するため、グループ正規化（Group Normalization：GN）[8]ではチャンネルをいくつかのグループGに分割し、(G, H, W)の軸に沿って統計量を計算し、$N * (C/G)$個の統計量を求め、事例-グループ毎に正規化する。事例正規化に比べて使える統計量が増え、学習が安定化するという利点がある。

次元間の相関をなくしさらに学習を加速させる白色化

ここまでに紹介した正規化はいずれも次元毎に統計量を求め正規化を行っており、次元間の相関はみていなかった。一方で次元間の相関を取り除くように変換することで、さらに勾配降下法による収束が速くなることがしられている。このような操作を白色化と呼ぶ。目的の関数の等高線が楕円で斜めになっている場合、それを同心円状にするような操作である。入力の前処理（ZCA：zero-phase component analysisなど）として行われていた。この白色化を行う層も提案されている。

白色化を行うためには入力の共分散行列の固有値分解を行う必要がある。この操作は入力次元数の2乗のオーダーのコストが必要であり、計算量が大きく、毎回行うのは現実的ではないが、例えばすべての次元間ではなく次元をグループに分け、それぞれのグループ内のみで白色化することで計算量を抑えつつ、白色化の効果を得ることができる[9]。また、畳み込み法を使う場合は入力はバッチ内サンプルとなる。このバッチ内サンプルに対し白色化することで、学習が速く収束するように変えることができる[10]。

今後の正規化層

正規化層は学習の安定化、汎化に大きな役割を持ち、優れた正規化層を設計することができれば分野全体に大きなインパクトがある。今後も新しい正規化層が登場してくると考えられる。

1) Y. LeCun et al.,"Efficient backprop," Neural networks:Tricks of the trade, 1998. http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf
2) S. Ioffe et al., "Batch Normalization: Accelerating Deep Network Training by Reducing INternal Covariate Shift," ICML 2015.
3) S. Santurkar et al., "How Does Batch Normalization Help Optimization," NeurIPS 2018.
4) J. Bjorck et al.,"Understanding Batch Normalization," NeurIPS 2018.
5)P. Luo et al.,"Towards Understanding Regularization in Batch Normalization," ICLR 2019.
6) L. Ba et al., "Layer Normalization," https://arxiv.org/abs/1607.06450
7) D. Ulyanov et al., "Instance Normalization: The missing ingredient for fast stylization," https://arxiv.org/abs/1607.08022
8) Y. Wu et al., "Group Normalization," ECCV 2018.
9) L. Huang et al., "Decorrelated Batch Normalization," CVPR 2018.
10) C. Ye et al., "Network Deconvolution," ICLR 2020.

（2020年7月号掲載の記事に加筆）

Energy-Based Model：ノイズを復元して生成モデルを学習する

深層学習を使ってデータを生成する深層生成モデルとして、変分自己符号化器（VAE）、敵対的生成モデル（GAN）、自己回帰モデル、正規化フローなどが知られている。これらのモデルは従来のモデルには難しかった高次元（数万～数百万次元）の自然画像や音声データ生成に成功しており、生成されたサンプルは本物と見分けがつかないほどに高忠実な生成が可能となっている。

これに対し、別の生成モデルとしてエネルギーベースモデル（Energy-Based Model、以下EBM）が近年注目されている。深層学習研究の中心人物の一人であり2018年にチューリング賞を受賞したYann LeCun教授もICLR 2020の招待講演[1]の中で次の重要なモデルはEBMであると話している。

EBMは他の生成モデルにない優れた特性を持つ。確率モデルを扱う際に不可欠な分配関数（全てのデータを列挙し積分をとる操作）をサンプリング時に扱う必要がなく、複数の候補解を列挙でき、また、複数のモデルを組み合わせることもできる。しかし、EBMの学習は難しく、生成モデルの実用的な性能は他の生成モデルに劣っていた。しかし近年の改良の結果、他の生成モデルによる最先端の結果に匹敵する品質の画像生成や音声生成を実現できるようになってきている。

本稿ではEBMおよび、近年の高次元データの学習を成功させた学習手法としてデノイジングスコアマッチングや強いつながりを持つ拡散モデルについて紹介していく。

エネルギーベースモデル（EBM）

EBMは入力を受け取ると、スカラー値であるエネルギーを返すエネルギー関数 $F(x)$ を使ってデータをモデル化する。この $F(x)$ はデータセット中のデータ上では小さな値をとり、そうでない場合は大きな値をとるように学習するのが目標である。EBMはエネルギー関数から導出されるギブス分布を使って確率モデルに変換することができる。

$$p(x) = \frac{\exp(-\beta F(x))}{\int_{x'} \exp(-\beta F(x'))}$$

この β は逆温度パラメータと呼ばれ（$1/\beta$ が温度に対応）、$\beta \to 0$ であれば一様分布、$\beta \to \infty$ であればエネルギーが一番低いところだけ大きな確率を持つような分布となる。この確率モデルを明示的に扱うことはなく、学習やサンプリングは分配関数（分母の正規化項）を扱わずに処理できるように工夫する。そのため、EBMのエネルギー関数は、非正規化密度関数の一種とみなすこともできる。

このモデルからデータをサンプリングするには尤度比 $p(x)/p(x')$ を使ったマルコフ連鎖モンテカルロ法（MCMC）を使ことが一般的であった。尤度比は、正規化項がキャンセルされ、扱う必要がないことに注意してほしい。しかし高次元データで多峰性があるデータではMCMCによるサンプリングは品質が低いことが分かっている。そこで本稿で扱う手法では対数尤度の入力についての勾配であるスコア関数を使ってサンプリングしている。これについては後で述べる。

次に、このエネルギー関数をパラメータ θ を使ったニューラルネットワークなどでモデル化 $F_\theta(x)$ し、訓練データ $D = \{x_i\}_{i=1}^N$ を使って学習することをみていこう。さきほどのギブス分布の対数尤度を最大化する最尤推定でパラメータを推定する場合、負の対数尤度 $\mathcal{L}$ のパラメータ θ についての勾配が求まればよい。この勾配は

$$\frac{\partial \mathcal{L}}{\partial \theta} = \mathbb{E}_{x \sim D}\left[\frac{\partial F_\theta(x)}{\partial \theta}\right] - \mathbb{E}_{x' \sim p_\theta(x)}\left[\frac{\partial F_\theta(x')}{\partial \theta}\right]$$

と求められる（β は省略）。つまり、勾配は、データ分布上の勾配の期待値と、モデル分布上の勾配の期待値の差として表せられる。

対称性があり、きれいな形をしているが、第二項のモデル分布からのサンプリングが難しいため高次元データをこの勾配を計算して学習することは難しく、別の学習手法が提案されていた。本稿ではデノイジングスコアマッチングと呼ばれる手法について扱っていく。

デノイジングスコアマッチング

対数尤度 $\log p(x)$ の入力について勾配をとった関数 $\nabla_x \log p(x)$ をスコア関数と呼ぶ。スコア関数は正規化項が含まれない（正規化項は入力依存項がないので、入力について勾配は0）。データ分布のスコア関数とモデル分布のスコア関数を一致させることできれば分布を学習できる。また、スコア関数が得られれば、ランジュバンサンプリングと呼ばれる方法を使って高次元でも効率的にサンプリングすることができる。これはエネルギー関数の勾配と少しのガウシアンノイズに従ってたどっていきサンプリングする手法である。

観測データ x にガウシアンノイズを加えたデータ $q_\sigma(\tilde{x}|x) = \mathcal{N}(\tilde{x}|x, \sigma^2 I)$ を元のデータにデノイジングする方向をノイズについて期待値をとった値はスコア関数と一致することが分かっている。これを使ってスコア関数を学習する手法をデノイジングスコアマッチング（DSM：Denoising Score Matching）と呼ぶ。具体的には次の目的関数を最小化するような関数 $s_\theta(\tilde{x}, \sigma)$ を求めることで、スコア関数を学習する。

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{x}|x)p(x)p(x)}\left[\left|\left|\sigma s_\theta(\tilde{x}, \sigma) + \frac{\tilde{x} - x}{\sigma}\right|\right|_2^2\right]$$

σ はノイズの大きさを示し、学習中にスケジューリングし調整する。様々な改良を加えることで、この手法を使って学習された生成モデルはGANなどに匹敵する性能を持つと報告されている[2][3]。

拡散モデル

拡散モデルは、データにノイズを徐々に加えていき全ての情報が失われて完全なノイズになるマルコフ過程を逆向きに辿ることで生成モデルを学習する。各拡散ステップが小さい量のガウシアンノイズである場合、その逆向きの生成に対応する遷移も条件付けされたガウシアンからのサンプリングでモデル化できることが知られている。このため、生成モデルはノイズからデータ分布を生成できるような条件付けガウシアンを使って学習できる。

拡散確率モデルの学習は対数尤度の変分下限を最大化することで実現されるが、その目的関数はさきほどのDSMと同じ式になることが分かっており、また高品質の画像を生成できると示された[4]。拡散確率モデルは表現力が高く、自己回帰モデルなどと違って並列にサンプリング操作ができ、推論時に計算コストと生成品質のトレードオフをとることができる特徴を持つ。スペクトログラムからの波形生成でも自己回帰モデルを使った手法に比べて高速に計算可能であり、リアルタイム生成を可能としている。

EBMの今後

EBMの大きな特徴は複数の候補解を列挙できること（特に連続値の場合）、複数のEBMを組み合わせて（例えば $\alpha F(x) + \beta F'(x)$ ）別のEBMを構成できること、複数変数間の双方向的な推論ができること（現在の候補出力結果に応じて入力を補正するなど）などがある。また生成モデルだけでなく、スコア関数は入力の欠損値を復元したり、異常値を発見したりと別の有用性がある。

また最近は複数の生成モデルを組み合わせてモデル化することも多くなってきた。EBMもその長所を生かした形で別の生成モデルと組み合わせて使うことも今後増えると考えられる。

---

1) Yann Lecun and Yoshua Bengio,"Reflections from the Turing Award Winners," ICLR 2020 Invited Talk, https://iclr.cc/virtual_2020/speaker_7.html
2) Y. Song et al., "Generative Modeling by Estimating Gradients of the Data Distribution," NeurIPS 2019.
3) A. Jolicoeur-Martineau et al.,"Adversarial Score Matching and Improved Sampling for Image Generation," https://arxiv.org/abs/2009.05475
4) J. Ho et al., "Denoising Diffusion Probabilistic Models," arXiv:2006.11239 WaveGrad: Estimating Gradients for Waveform Generation," https://arxiv.org/abs/2009.00713

（2020年11月号掲載の記事に加筆）

Transformer：全タスクの標準ネットワークアーキテクチャになるか

深層学習（ディープラーニング）はタスク毎に異なるネットワークアーキテクチャを使ってきた。画像認識であればCNN（畳み込みニューラルネットワーク）、自然言語処理であればRNN（回帰結合型ニューラルネットワーク）、表データや座標など入力が構造を持たないようなタスクに対してはMLP（多層パーセプトロン）、化合物などグラフ構造を持つ場合はグラフNN（ニューラルネットワーク）といったようにだ。

こうしたネットワーク構造は問題が持つ特徴（局所性、制約、入力変換に対する同変性、不変性）を捉えており、問題に対する事前知識をモデルに埋め込む帰納バイアスとして有効である。帰納バイアスは少ない学習データで汎化するのに重要な役割を果たしている。

しかし、最近になって、Transformerと呼ばれるネットワークアーキテクチャが様々なタスクに広く適用することができ、それぞれの分野で最高精度またはそれに近い精度を達成できることが分かってきている。Transformerは第6章 第4節、第11章 第4節でも紹介している。

Transformerは機械翻訳の分野で提案され、従来のRNNベースの手法を上回る性能を達成した。他の自然言語処理でも適用されるようになり、最近ではBERTやGPT-3など大量の言語データを使った事前学習においてもTransformerやそれに類したモデルが有効であると分かってきた。グラフNNもTransformerが使っている注意機構との相性が良く、広く使われていた。そして、最後に残っていた大きな画像認識分野においても、Transformerを使って、従来使われていたCNNのResNetやEfficientNetなどと並ぶ性能が達成できてきたと報告されている。

Transformerとは何か、どのような改善がなされているか、今後Transformerが全タスクを席巻していくのかについてみていこう。

Transformerとは

はじめにTransformerは、どのようなネットワークアーキテクチャなのかについておさらいしよう。入力は m 個の要素の集合 $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \ldots, \mathbf{x}_m$ から成る（太字は列ベクトルとする）。自然言語処理では各単語から求められた特徴ベクトルであり、グラフ問題では各頂点、画像では画像パッチ毎に求められた特徴ベクトルや検出した物体などが対応する。

Transformerは注意機構を使って自分の計算結果から必要な情報を読み取る自己注意機構を利用する。具体的には学習可能な行列 W^Q、W^K、W^V を用意し、要素毎にクエリ $\mathbf{q}_i = \mathbf{x}_i^T W^Q$、キー $\mathbf{k}_i = \mathbf{x}_i^T W^K$、値 $\mathbf{v}_i = \mathbf{x}_i^T W^V$ を計算する。これら行列は全ての位置で共通である。次に、i 番目の要素がどの位置から値を読み取るかを決めるため、クエリとキー間の内積 $\mathbf{q}_i^T \mathbf{k}_j$ を計算する。内積が大きい値であるほど、その位置から情報を読み取ることを表す。そして正規化として内積に読み取り先（先程の添字 j）にわたってSoftmax関数を適用し、確率のように重みは非負かつ合計値が1になるように正規化しておく。また、内積の大きさはランダムなガウシアンで初期化したベクトル間の場合、次元数の平方根に比例するので、内積をキーの次元数 d_k の平方根で割っておく工夫が、実用上重要である。まとめると、i の位置で j の位置から情報を読取る重みは

$$a_{i,j} = \mathrm{softmax}(\mathbf{q}_i^T \mathbf{v}_j / \sqrt{d_k})$$

と計算される。softmaxは添字 j に沿って適用される。そして、重み $a_{i,j}$ に従って各値を足し合わせた $\mathbf{z}_i = \sum_j a_{i,j} \mathbf{v}_j$ を位置 i の計算結果とする。

従来のネットワークでは情報の流れ方は事前に設計した通りにしか流れないが、（自己）注意機構では入力データに応じて流れ方を変えることができる。さらに、遠距離の情報も必要があれば1ステップで読み取ることができる。もしある位置の情報がその後の処理に重要であれば、その要素との内積を大きくするように、クエリとキーが更新される。

上記の操作は全て行列積と要素毎の操作の組み合わせで

$$Z = \mathrm{softmax}(Q^T K / \sqrt{d_k}) V$$

のように計算することができる。ここで Q, K, V はそれぞれクエリ、キー、値のベクトルを列毎に並べて作られた行列で

ある。GPUやディープラーニング向けアクセラレーターが得意とする行列積を使って実現するため、計算効率も高い。このような計算を複数並列に行い、それらの結果を統合するマルチヘッドを使う場合が一般的である。

これらの計算において、各要素がどの位置にあるのかという情報は考慮されていない。位置を考慮するために、位置情報を特徴ベクトルに符号化しておき利用する。

Transformerはこの自己注意機構を使った符号化器（encoder）と復号化器（decoder）から成る。符号化器は複数層からなり、入力を自己注意機構を使って符号ベクトルに変換しておく。復号化器は自己注意機構に加えて、通常の（相互）注意機構を使って符号化器が作った情報を読み込み、処理をしていく。

Transformerの各分野への広がり

前述のようにTransformerは自然言語処理で最初に広く使われるようになった。それが2020年に入ってTransformerを使ったモデルが画像分類や、物体検出などでもCNN（ResNet,EfficientNet）に匹敵する性能を達成するようになってきた。

例えば、物体検出においてはDETR[1]と呼ばれる手法が、物体検出のSOTAの1つであるFaster R-CNNに匹敵する性能を達成し、また物体と背景全てをセグメンテーションするpanopticセグメンテーションも同様にして処理することができることを示した。その後も改良が進められ、速度面でもCNNに匹敵するような速度が達成できるようになっている。

画像分類もVision Transformer[2]と呼ばれる手法が登場した。これは画像をサイズが16x16のパッチに分割し、それぞれのパッチをMLPで特徴ベクトルに変換した後にTransformerの符号化器のみ利用し、その結果をMLPで読み取り分類する。あたかも16x16パッチを単語に置き換え、その単語列から画像が何であるかを分類しているかのようなものだ。訓練データ数が少ない場合はCNNベースの手法が勝っているが、訓練データ数が大きくなり、帰納バイアスの影響が少なくなってくるとTransformerを使った手法が精度面ではほぼ匹敵する性能を達成すると報告されている。さらに計算効率が高いことから、事前学習コストはCNNベース手法の1/4に抑えられることが報告されている。

また、元々可変個の集合を扱うのが得意なため、グラフ情報や点群情報、複数の物体を扱うのにも適しており、Transformerや自己注意機構を使った手法が多くの最高精度を達成するようになっている。

Transformerの効率改善

Transformerの大きな問題は計算量である。入力がN個の要素から成る場合、それぞれの要素と他の全ての要素との内積や重みを計算する必要があるため、計算量は$O(N^2)$かかってしまう。CNNやRNNは局所的な接続に限定することで計算量を$O(N)$に抑えられている。

この計算量を減らすための研究も近年多く報告されている。代表的なBig Bird[3]は注意対象を疎なランダム位置、局所的な周辺情報、全位置が注意対象であるルーティングのような役割を持つ定数個の特殊な要素を組み合わせることで計算量を$O(N)$に抑えながら、Transformerと同じ表現力を持つことができ、実験結果としても性能はほぼ匹敵している。数千や数万といった長い系列の遠距離の関係を扱うことができ、遺伝子配列DNAの遺伝子発現を調整するプロモーター領域をほぼ完璧に予測できるようになっている。

また、Transformerを簡略化する提案もされている。例えばLambdaNetwork[4]は入力全体で計算した特徴ベクトルを全ての要素に適用し、位置依存の処理は限定することで全体の計算効率を改善しつつ、表現力などを高めている。Attention Free Transformer（AFT）はヘッド数を次元数と一緒にし、各ヘッドが1次元しか対応しないようにすることで、計算を要素ごとの演算にできるようにし、キーと値間の計算を先にすることで計算量を線形に抑えることができている。このようにした場合も同じ性能を達成できると報告している[5]。

全てのアーキテクチャがTransformerになるのか

ここまでみたようにTransformerは自然言語処理では既存手法を上回り、他分野でも広く使われ、画像認識などでは学習データが多く存在する場合に匹敵する性能が達成できることが分かっている。それでは今後全ての分野でTransformerが使われるようになるのだろうか。

帰納バイアスの観点からすると、各タスクにより特化したアーキテクチャを利用したほうがTransformerのような汎用的のモデルを使うよりも汎化性能が高く、また最適化もしやすいだろう。しかし、一つのアーキテクチャで多くの分野が解けるようになる利点はいくつかある。

1つ目はTransformerに対し、個別に開発されているアー

キテクチャよりも多くの研究開発投資がなされることで急速に改善され、それぞれの専用モデルの性能を上回る可能性である。計算機においても専用機の性能を汎用機が上回ることが多くあったが何ごとが起きる可能性がある。

2つ目はTransformerに特化したようなハードウエアなどが登場してくることである。個別タスク毎に専用アクセラレーターを作った場合、市場は限られているがTransformerであれば多くのタスクに使えるため市場は大きく、開発投資を回収できる可能性がある。2022年3月に米NVIDIA社が発表したGPU「H100」ではTransformer向けのアクセラレーター「Transformer Engine」を搭載し、高速化している。

現段階では、個別のアーキテクチャが勝つか、Transformerがさらに発展し全分野を席巻するか断定できない状況である。今後の動きに注目が必要である。

---

1) N. Carion et al.,"End-to-End Object Detection with Transformers," ECCV 2020.
2) "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," ICLR 2021.
3) M. Zaheer et al.,"Big Bird: Transformers for Longer Sequences," NeurIPS 2020.
4) "LambdaNetworks: Modeling long-range Interations without Attention," ICLR 2021.
5) "An Attention Free Transformer," https://arxiv.org/abs/2105.14103

(2020年12月号掲載の記事に加筆)

6-9 離散化生成モデル

現実世界の多くが連続値で表される。時空間は連続であり、物体やその性質、サイズ、形状、重さなどを表すのも連続値であり、画像や音声など観測シグナルも連続量で表される。

そのため、現実世界の問題を機械学習や深層学習で扱う場合、連続量の入力を受け取り連続値のベクトルを内部状態として持ち処理するのが自然である。最終的な出力結果だけ、分類結果など離散的な情報を扱う場合もあるが、基本的に内部はすべて連続量で処理される。こうした連続値を使った表現は誤差逆伝播法を使った勾配法による最適化とも相性がとても良く、スケーラブルな学習を実現する。

これに対し、情報を連続値で扱わずに離散値で扱う表現が提案されている。例えば画像の潜在表現に離散値を使うVQ-VAE[1]がある。オリジナルのVAE（変分自己符号化器）は潜在変数を使った生成モデルであり、連続値を持った潜在ベクトルから復号化器を経由し画像を生成する。これに対しVQ-VAEは、離散値を持つ潜在変数を生成し、次に、各離散値に対応した埋め込みベクトルを入力とした復号器で画像を生成する。埋め込みベクトルの数は離散値の種類数しか無いので、画像を離散的な情報で表しているといえる。

こうした離散表現は元の連続量を近似しているため情報を大きく失っているように思えるが、組み合わせることで驚くほど元の情報を正確に保存することができる。なぜ離散値で表現しても元の情報を復元できるかというと、多様体仮説にあるように、画像データ全体は画像より少ない次元数を持つ多様体から生成されていると考えられ、各離散値に対応する埋め込みベクトルがこれらの多様体上の領域を代表点として十分カバーし近似できているためと考えられる。

一方で、離散値列の代表は言語データであり、各単語が離散値であり文は離散値列である。こうした言語データはGPT-3を代表とするようにTransformerを使った自己回帰モデルによる生成モデルを学習することでデータ生成や補間、条件付け生成だけでなく様々なタスクに有効な特徴を獲得することができると示されてきた。

さきほど連続量を持ったデータも情報をほとんど失わずに離散値列に変換できることを示した。以降では言語データと同じようにみなせるということでこれらの離散値をトークン、そして離散値列をトークン列と呼ぶことにする。そしてこれらのトークン列の生成モデルを言語データの生成と全く同じようにTransformerを使った自己回帰モデルを使って学習できる。生成されたトークン列は復号化器でほぼ損失の無い形で元の形に戻せる。この連続量をトークン列にして生成モデルを学習し、元の連続量に戻すというアプローチは画像[2]、音声/音楽[3]、動画[4]生成などに適用され、元のデータの生成モデルを実現できることが示された。しかも高次元データの場合、連続量を直接扱う生成モデルと比べ、トークン化した場合、生成品質やその多様性、条件付け生成などで大きく成功している。なぜ成功しているのかについて以降で解説する。

離散表現には多くの利点がある

離散表現を使った生成モデルには多くの利点がある。ここでは代表的な5つの利点を挙げよう。

1つ目は生成モデルを学習しやすい。連続表現を使った生成モデルの学習は一般に難しい。その理由は観測していないデータを列挙することが難しいためである[5]。最大推定やエネルギーモデルでは観測データの尤度を上げ（エネルギーを下げ）、観測していないデータの尤度を下げる（エネルギーを上げる）必要がある。高次元データでは観測していないデータを列挙することが不可能であり、高次元の呪いでも説明がつくように、ほんの一部の領域が誤って尤度が高くなっていたとしてもそれを見つけることは難しい。

これに対し、離散表現+自己回帰モデルを使った場合は離散値の種類数は多くても数百から数千程度なので全列挙することができ、全ての非観測事例を列挙して、それらの尤度を明示的に下げることができる。さらに不確実性も離散表現の方が自然に扱え、可能性のある離散値に確率を割り当てることは可能であるが、高次元の連続表現でこうした確率を正確に割り当てることは難しい。

2つ目に生成モデルの正則化として有効である。生成モデルを学習する際、誤って尤度が高いデータが発生してしまわないよう、尤度が高い（エネルギーが低い）領域の容量に制限をかけておくことが有効であり、疎な表現を使ったり離散値を使うことで制限できる。

3つ目にモダリティの異なる様々なデータを統合できる。例えばDALL-E[2]では、キャプション付き画像データを使ってテキストで条件付けして画像を生成するモデルを考えている。テキストと画像を統一的な生成モデルで扱うことは難しいが、画像をトークン列に変換しておき、テキストと画像のトークン列をつなげた1つのトークン列に対して生成モデルを学習する。これによってキャプションに対応する画像生成が実現される。

4つ目は効率的な保存が実現できる。トークンは疎なベクトルとみなすことができ、連続ベクトルを保存するのと比べて非常にコンパクトに保存しておける。例えば文献[2]は256x256のフルカラー画像を、32*32でそれぞれが8192(2^{13})種類のトークンに離散化しており、無圧縮で192Kバイト(=256*256*3byte)の画像を1.7Kバイト(=32*32*13/8byte)で保存できる。

5つ目は自己回帰モデルなど自身の予測を条件付けに用いて繰り返し予測を行う場合のDriftingを防ぐことができる。高次元で予測を繰り返していく時、各予測に誤差が含まれるため、長い時間予測を行う場合、予測がずれてくる現状が起きる。これに対し離散化されている場合は高次元由来の誤差が含まれにくいため、長期間予測しても誤差が大きくならず、大きく間違えることが少ない。

離散表現は学習が難しい

一方で離散化は学習で様々な問題がある。ニューラルネットワーク(NN)の学習は誤差逆伝播法を使い、その際には用いる計算はすべて微分可能で勾配が消失しないことが求められる。一方で離散化処理は微分が計算できない/もしくは消失する。例えば離散化処理の例として連続なスカラー値 x を受け取り、その符号が正なら $+1$、負なら -1 を返す $\mathrm{sign}(x)$ という関数を考えてみよう。この関数の微分は $x \neq 0$ では 0、$x = 0$ で $+\infty$ となりすべての位置で微分が消失するか発散する。

この問題に対して大きく2つ方法がある。Straight Through Estimator(STE)は前向き処理はそのまま離散化処理を行い、誤差逆伝播時は離散化処理を省略し、そのまま誤差を伝播させる。この場合、正しい勾配を使わず学習するが多くの問題で学習できることがわかっており、前述のVQ-VAEもSTEを利用している。また、シグモイド関数を使った学習で射影勾配法を適用した場合、誤差逆伝播時はシグモイド関数がキャンセル消去されSTEと同じ形をとり、離散化処理がシグモイド関数を急峻にしたものと考えるとSTEは正しい射影勾配法を実現しているとみなせる[6]。

もう1つの方法がGumbel Softmaxである。Gumbel SoftmaxはSoftmaxで定義する確率分布からのサンプリングを近似し、変数変換トリックを使った微分を可能としながら、温度パラメータを下げていく時に離散化処理に一致するような関数である。学習初期には温度を高くし、学習終盤に温度を0に近づけていくことで学習中は勾配を発生させながら離散化処理を実現する。

離散表現と言語

離散表現は詳細を捨てた抽象的な表現とみなすこともでき、究極的には言語データのような抽象的な思考や推論の実現につながると考えられる。まだ離散表現上での操作や計算は発展途上だがグラフNNや注意機構などを組み合わせていき、離散表現上で様々な処理ができるようになることが期待される。

1) A. Oord et al.,"Neural Discrete Representation Learning," https://arxiv.org/abs/1711.00937
2) A. Ramesh et al.,"Zero-Shot Text-to-Image Generation (DALL-E)," https://arxiv.org/abs/2102.12092
3) P. Dhariwal et al.,"Jukebox: A Generative Model for Music," https://arxiv.org/abs/2005.00341
4) W. Yan et al.,"VideoGPT: Video Generation using VQ-VAE and Transformers," https://arxiv.org/abs/2104.10157
5) Y. LeCun,"The Energy-Based View of Self-Supervised Learning," GTC 2021, https://drive.google.com/file/d/1tHlYoh_2ZRGOwvPGtSEact520ABEgV1R/
6) "Optimizing with constraints:reparametrization and geometry,"https://vene.ro/blog/mirror-descent.html

(2021年7月号掲載の記事に加筆)

6-10

Perceiver：多様な入出力に対応可能なニューラルネットワーク

生物は視覚、聴覚、触覚など様々なモダリティを持つ高次元データを同時に感知することができる。一方でニューラルネットワークでこうしたデータを扱う場合は、それぞれのモダリティ毎に専用の感知モデルを設計し利用する必要がある。例えば画像であればデータは2次元のグリッド上の信号だとし、その上で畳み込み層で変換する。また複数のモダリティを処理する必要がある場合はそれらを統合するネットワークを用意し、問題の出力形式に合わせた専用のネットワークを設計する必要がある。こうした事前知識の利用は帰納バイアスとして有効である一方、複数のモダリティをどのように扱うか、それらをどのように統合するのかは自明ではない。

Perceiver[1]はあらゆるモダリティのデータを扱えるように設計されたニューラルネットワークである。入力がどのような構造を持つのかは仮定せず、必要な情報が付与されたバイト列として扱う。そして非対称の交差注意機構を利用し、入力全体を固定サイズの潜在変数列に変換して並列に読み込んだ上で、潜在変数列上で自己注意機構を使った変換を繰り返し適用した上で出力結果を求める。

さらにPerceiver IO[2]はあらゆる種類の出力を扱うように拡張された。例えば出力として、分類のようなスカラー値だけではなく、入力と同じような構造を持ったようなベクトルやテンソル、集合を扱えるようにした。これによりあらゆるタスクを1つのニューラルネットワークアーキテクチャで扱うことができる。さらに単に扱うことができるだけでなく、それぞれの問題に専用で設計されたモデルに匹敵または超える性能を達成できると報告された。例えば、自然言語理解、画像分類、オプティカルフロー（隣接フレーム間での対応画素の移動予測）、音声、動画の自己符号化器、複数エージェントの強化学習などのタスクにおいてである。本稿ではこれらPerceiver、Perceiver IOについて紹介していく。

Perceiverの構造

Perceiverは2つの要素から構成される。1つ目の要素は交差注意機構で、入力のバイト列と潜在変数列から、潜在変数列へと変換する。2つ目は潜在変数列から潜在変数列へ変換する自己注意機構である。

はじめに、従来の自己注意機構について説明し、それを直接入力列に適用する場合の問題点を説明しよう。自己注意機構またはTransformerは入力列からそれと同じ長さを持つクエリQ、キーK、値VをMLPで計算し、クエリとキー間の内積の大きさ（注意）に応じて値を読み込むような手法である。この場合、全要素間の注意を計算する必要があるため、入力列のサイズがMの時、$O(M^2)$の計算量が必要である。入力列は一般に大きく、例えば画素数が224x224=50176のImageNet画像や48kHzでサンプリングした1秒間の音声はサイズが約5万の入力列である。こうした入力列に対し、自己注意機構を直接適用することは計算量が大きすぎるためできない。

この計算量が大きいという問題をPerceiverは交差注意機構を使って解決する。交差注意機構はキーKと値Vは入力列から求め、クエリQは潜在変数列から求める。潜在変数列のサイズをNとした時、この交差注意機構の計算量は$O(MN)$となる。潜在変数列のサイズNが入力変数列のサイズMよりずっと小さいため（例えばN=512など）、この計算量は対処可能である。次にPerceiverはこの潜在変数列上でTransformerを適用する。潜在変数列のサイズは小さいためTransformerを直接適用できる。適用した場合の計算量は$O(N^2)$であり、潜在変数列上でL層からなる自己注意機構を適用した場合の計算量は$O(LN^2)$である。

これら交差注意機構と自己注意機構をあわせるとPerceiver全体の計算量は$O(MN+LN^2)$である。重要なのは入力サイズMと層数Lが分かれていることである。入力サイズに依存せず問題の複雑さに応じて層数を自由に変えることができる。

また、Perceiverは交差注意機構を使った入力からの読み込みを、最初だけではなく、途中で複数回行う。これによりこれまでの認識結果に応じて入力の必要な情報を、元の解像度のまま読み込むことができる。

そして、Perceiverは交差注意機構や自己注意機構のパラメータを層間で共有する。これによりパラメータ数は大きく

減らすことができ、過学習を防ぐことができる。このようにPerceiverはパラメータが共有された処理を潜在変数列に対し繰り返し適用しているようにみなせるのでRNNの一種とみなすことができる。

位置符号化で入力の位置情報を与える

注意機構は入力集合中の要素の置換操作に対し同変（特殊な場合は不変）である。この性質は点群など置換操作に対し同変であってほしいデータには望ましいが、位置も重要である画像や音声などには適切ではない。そこで位置情報も必要な問題については、入力にはBERTやNeRFなどでも使われるフーリエ位置特徴を付与する。具体的には d 次元目の位置が x_d である場合、$[\sin(f_k \pi x_d), \cos(f_k \pi x_d)]$ という値を付与する、ただし f_k は周波数バンク中の k 番目の周波数である。この位置符号化によって、Perceiverは位置情報も利用した上で処理を行っていく。

Perceiver IO

Perceiverは任意の入力を扱えるようにしていたが、それを任意の出力を扱えるように拡張したのがPerceiver IO[2]である。Perceiver IOは出力も注意機構を使って求めるようにする。具体的には、出力の各要素毎にクエリを設定し、このクエリを使って潜在変数ベクトル列から必要な情報を読み込んで出力を決定する。

クエリは出力の各要素を特徴づけるのに十分な情報が得られるように設定しておく。例えば位置符号化された位置情報などをクエリとして利用し、オプティカルフローの場合は位置情報に加え、さらに対応する元の入力画像情報を付与する。Perceiver IOはこれらのクエリを使って適切なデータが読み取られるように学習していく。この出力側が必要な情報を処理した結果からクエリを使って読み取る方式は、他にも米Tesla社の画像システムのVector Space[3]、Transformerを使った物体検出モデル[4]などで使われている。以降、Perceiver、Perceiver IOは共にPerceiverとまとめて言及するようにする。

Perceiverは問題特化の手法に匹敵する性能を達成

実験では最初に挙げたような様々なタスクに対しPerceiverを使って評価した。いずれもそれぞれのモダリティに特化した手法の最高性能に近い性能を達成し、特にこれまで進歩が著しかった画像分類において同様の性能が達成できたことは特筆に値する。また画像の画素を特定の順序に従い置換させた場合の問題設定においてもPerceiverは全く性能が落ちなかった。これに対し、画像の位置を前提とするCNNなどは位置情報を与えたとしても性能が大きく落ちていた。Perceiverは位置情報をどのように扱うのかも学習から獲得できるということになる。

一方、Perceiverは幾何情報に関する同変性や不変性などは扱えていない。今後、事前知識として重要な幾何情報に関する不変性、同変性を加えていく改良が考えられる。例えばAlphaFoldにおける不変点注意機構などである。

今後の展望

Perceiverは様々なモダリティを1つのネットワークで統一的に扱えるようにした。特に音声や動画など異なるモダリティを持つデータを統合し、また別の出力に自由に変形できるという点でニューラルネットワークやTransformerの汎用性をさらに拡張したものとみなせる。

これにより、特定の問題を解く場合に専用のネットワークアーキテクチャを設計することなく、まずはPerceiverを試してみることができる。また、様々なタスクが1つのネットワークアーキテクチャに統一されていくことにより専用ハードウエア設計がより簡単になっていくと考えられる。

1) A. Jaegle et al., "Perceiver: General Perception with Iterative Attention," ICML 2021.
2) A. Jaegle et al., "Perceiver IO: A General Architecture for Structured Inputs & Outputs," https://arxiv.org/abs/2107.14795
3) Tesla AI Day, Vector Space https://www.youtube.com/watch?v=j0z4FweCy4M&t=3450s
4) N. Carion et al., "End-to-end Object Detection with Transformers," ECCV 2020.

（2021年11月号掲載の記事に加筆）

第 7 章

記憶の仕組み

7-1 “Fast Weight”：アテンションで短期記憶を実現する

短期記憶の仕組みは高度なタスクの実現に必要不可欠である。例えば、複数の作業を組み合わせたタスクを実現する場合、自分がこれまでどのような作業をして何が残っているのかを知らなくてはいけない。

このような短期記憶はリカレントニューラルネットワーク（RNN：Recurrent Neural Networks）でも、ある程度実現されている。RNNは内部状態を保持し、各時刻毎に入力を受取って内部状態を更新し出力を決定する。例えば、時刻tに入力$x(t)$を受け取り、内部状態$h(t)$を$h(t+1)$へと更新し、出力$y(t)$を決定するRNNは次のような関数fとして与えられる。

$$(y(t), h(t+1)) = f(x(t), h(t); \theta) \quad (for\ t = 1 \cdots n)$$
$$= f(Wh(t) + Cx(t))$$

この例では、入力と内部状態を線形変換して与えた上で、RELUなどの非線形変換を適用している。内部状態をより長期間変えずに保持したい場合はLSTMやGRUなどより複雑な関数を利用する。RNNの場合、内部状態が短期記憶の役割を果たし、過去の情報を符号化した上で記憶している。RNNは自然言語処理や部分観測マルコフ決定過程における記憶を備えた強化学習などで成功を収めている。一方でRNNの記憶容量は内部状態hのユニット数によって上限があるという問題がある。例えば、与えられたキーと値のペアを次々と記憶していき、最後にキーだけが与えられそのキーに対応した値を推定する問題を考えよう。この場合、内部状態で記憶可能な量より多くのペアを記憶しようと思った場合、過去の記憶した情報を忘れてしまう、または記憶が汚染されるという問題が発生する。

この短期記憶の問題に対しては、外部記憶を用意し、内部状態を外部記憶にコピーし、必要になれば外部記憶から内部状態へ読み込むことで解決される。このような方法が実際の脳で実現されているかは分からないが、コンピュータ的には実現は容易であり、Neural Turing Machines[1]やDifferentiable Neural Computers[2]などで実現されている。

重みに短期記憶を一時的に蓄える

今回はそれとは別に記憶容量を大きく向上させる手法としてニューラルネットワークの重みを一時的に変える“Fast Weight”と呼ばれる手法[3]を紹介する。人や動物の脳では、ニューロン間のあるシナプスが使われた場合、そのシナプスが一時的に増強される現象が知られている。これと同様に、RNNの重みを一時的に変えることで短期記憶、特に連想記憶を実現する。内部状態のユニット数がNの時、そのユニット間をつなぐシナプスに相当する重みの個数は$O(N^2)$であり、内部状態のユニット数Nに比べて非常に大きい。そのため、重みには非常に多くの情報を記憶することができる。

ニューラルネットワークの重みは通常、目的関数を最小化するように学習単位毎（ミニバッチ毎）に確率的勾配法を使って更新する。これらの通常の重みはゆっくり更新されることから“Slow Weight”と呼ぶことにする。一方で、1つの系列を処理する間に一時的に重みを変えていくものを、速く変わっていく重みということでFast Weightと呼ぶことにする。

このFast WeightはN行N列の行列で実現される。$A(t)$は前の時刻の重み$A(t-1)$を減衰させ、今の内部状態の外積を加えることで、

$$A(t) = \lambda A(t-1) + \eta h(t) h(t)^T$$

として得られる。今の内部状態の外積を加えたということは、今の内部状態と似た状態の次に今の内部状態が出現しやすいということをモデル化している。例えば、ネコを見たとし、次の時刻にネコの半分が障害物で隠れて見えなくなった場合を考える。この場合、ネコの半分しか見えないにもかかわらず“Fast Weight”が過去の全体が見えている時の内部状態を“思い出し”、次の時刻にはネコ全体に対応する内部状態を作ることができる。

現在の内部状態から次の内部状態は次のように計算する。はじめにSlow Weightを使って次の状態の初期値を作る。

$$h_0(t+1) = f(Wh(t) + Cx(t))$$

次に、Fast Weightを適当に決めたS回適用し、内部状態を次のように更新する。

$$h_{s+1}(t+1) = f([Wh(t) + Cx(t)] + A(t)h_s(t+1)) \quad (for\ s = 1 \cdots S)$$

この際、最初にSlow Weightで求めた部分は境界条件の役割を果たし、Fast Weightによる影響を加えて定常状態を求める。そして、この内部ループの最後の状態が、次の時刻の内部状態となる。

$$h(t+1) = h_s(t+1)$$

このFast Weightは、AがN^2個のパラメータが存在し、明示的に計算した場合、計算量が大きいという問題がある。また、ミニバッチ学習をした場合、各サンプル毎に別々のAを保持し、計算しなければいけない問題が発生する。そのため、直接Aは扱わずに、$A(t) = \eta \sum_{\tau=1}^{t} \lambda^{t-\tau} h(\tau) h(\tau)^T$であることに注意すると、$A(t)h_s(t+1)$は次のようにして求められる。

$$A(t)h_s(t+1) = \eta \sum_{\tau=1}^{t-1} \lambda^{t-\tau} h(\tau) h(\tau)^T h_s(t+1)$$

つまり、Fast Weightは現在の内部状態と過去の内部状態との内積$h(\tau)^T h_s(t+1)$を求め、その値でアテンションをかけて、過去の内部状態を思い出すのに相当する。このようにすればFast Weightの行列を明示的に使わず、アテンションの仕組みのみで短期記憶を実現できる。

このFast Weightの効果を調べるために、Fast Weight論文の著者らは最初の実験として、アルファベットのキーと数字の値のペアの列からなる文字列を入力として与えられた後に、キーのみが与えられ、それに対応する値を報告するタスクの学習を行った。この場合、RNNでは内部状態のユニット数が十分大きくなければキーと値のペアを覚えることができない。RNNの内部状態数が少ないと、Fast Weightを使わない場合は十分記憶することができず、例えば内部状態数が20の時、Fast Weightを使った場合のエラー率は1.81%であったのに対し、RNNの一種であるLSTMのみの場合のエラー率は60.81%であった。

次のタスクとして、画像認識を行った。現在主流である畳み込みニューラルネットワーク（CNN）は画像の全ての位置について同じ計算量をかけて処理をするため、タスクに関係のない大部分の処理が無駄になっている。一方で人や動物は画像を見る場合、はじめに全体を見た後に、その情報に応じて目や鼻など関係しそうな部分に注目することで認識を行う。これと同様に、RNNが画像の部分を順に注目して見ていき、認識を行うことで、CNNと同じ精度が出たと報告された。

このほかにも、複数の画像を総合的に見ないと判断できないようなタスクや、記憶が必要な強化学習のタスクにもFast Weightを使った手法の有用性が報告されている。

Fast Weightは非常に単純なアテンションの仕組みで短期記憶を実現できるということで興味深い研究である。一方で実用的には、過去の内部状態を記憶しておかなければならず、また内部ループでの計算量も大きいという問題点がある。この改良として、例えば内積が大きい部分だけを確率的に思い出したり、またはミニバッチ学習の場合も含めてSlow Weightを効率的に計算する手法が今後重要になると考えられる。

1) A. Graves et al.,"Neural Turing Machines,"https://arxiv.org/abs/1410.5401
2) A. Graves et al.,"Hybrid computing using a neural network with dynamic external memory," Nature, vol.538, pp.471-476.
3) J. Ba et al.,"Using Fast Weights to Attend to the Recent Past," NIPS 2016.

（2016年12月号掲載の記事に加筆）

7-2 Differentiable Neural Computers：外部記憶を備えたニューラルネットワーク

現在のデジタルコンピュータは計算処理を担当するプロセッサと1次記憶や長期記憶を担当する外部メモリから構成される。

計算の際に外部記憶に格納されている値を参照する仕組みは変数や関数の仕組みの実現につながり、問題の抽象化を達成する。例えば$f(x,y)=2x+y$という関数は変数xとyを変えることで再利用でき、別の関数の結果を変数として使うことで関数同士を組み合わせることもできる。さらには関数自身を変数とみなすこともでき、高階関数のような高度な計算能力を備えた計算機を実現できる。

また、拡張可能な外部記憶は、新しいタスクを効率よく覚えられるという長所がある。記憶容量が固定の場合、新しいタスクを覚えるたびに既存の記憶が汚染され、既存のタスクの性能が低下する恐れがある。新しいタスクを覚えるごとに記憶領域を増やすことで、既存の知識を汚染せずに学習することができる。

Neural Turing Machineの後継版

このような外部記憶を備えたニューラルネットワークとして、Differentiable Neural Computers（DNC）[1]を紹介する。これはNeural Turing Machine（NTM）の後継であり、NTMより複雑なタスクを解けるようになった。

DNCはNTMと同様にプロセッサに相当する微分可能なコントローラと、外部メモリに相当する外部記憶から構成される。DNCは外部記憶にデータを読み書きするが、この位置を決めるのがソフトアテンションと呼ばれる機能であり、通常の記憶装置と同様に書き込みヘッド、読み込みヘッドと呼ぶことにする。DNCのヘッドは通常の離散的なアドレッシングとは違って、位置に対する重み分布を使ってどこに書き込むか、どこから読み込むのかを決める。外部記憶はN個の行から構成される$N\times W$の行列Mで表される。別の言い方をすれば外部記憶は1次元のアドレス$[1,2,\ldots,N]$を持ち、各アドレスには長さWのベクトルが格納されている。読み込み時のアテンションは長さNの実数ベクトルw^rで表され、これにより読み込み結果のベクトルrは

$$r=\sum_{i=1}^{N} M[i,\cdot]w^r[i]$$

として表される。ただし、$M[i,\cdot]$はMのi行目のベクトルを意味する。同様に、ベクトルvを外部記憶に書き込む場合は書き込みヘッドである長さNのベクトルw^wを使い、さらに長さMの消去ベクトルeを使い、次のように外部記憶を更新する。

$$M[i,j]=M[i,j](1-w^w[i]\,e[j])+w^w[i]\,v[j]$$

コントローラは各時刻に、入力xと前の時刻の読み込み結果rを受取り、これらの入力情報とRNNの内部状態に基づいて出力yと書き込む内容v、そして読み込みヘッドw^r、書き込みヘッドw^wを出力する。この書き込みヘッドに基づいて、外部記憶にvが記録される。

アテンションの仕組みを活用

DNCの核心はどのように読み込み／書き込みヘッドw^r、w^wを決めるのかである。DNCは3つの微分可能なアテンションを組み合わせてヘッドを決定する。

1つ目は、コンテンツベースのアテンションである。連想記憶と同様に、コントローラにより決定されたキーベクトルkとのコサイン距離を重みとしてアテンションを決め、似ているほど1に近く、似ていない場合は0になるような重み分布を与える。このコンテンツベースのアテンションは補完の役割を果たす。例えば、猫の半分しか写っていない画像に対応するキーから、過去の猫の全体を捉えた記憶を思い出すことができる。

2つ目は、連続するステップで書き込んだ順を覚えておき、それを利用してアテンションを決める。N行N列の時間遷移行列Lは、$L[i,j]$が位置jに書き込んだ後に位置iに書き込んだ場合に1に近く、そうでない場合は0となるような行列とする。ある時刻の重み分布をwとした時、行列Lを掛けたLwは次の時刻に書き込んだ位置に相当する重み分布となり、$L^T w$は1つ前の時刻に書き込んだ位置に相当する重み分布

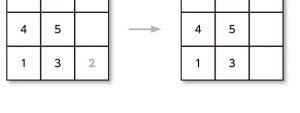


図7-1　ブロックパズル「Mini-SHRDLU」タスクの例
「各列の一番上にある数字を、別の列の一番上に移動できる」というルールである。

となる。

このLを使えば、過去に書き込んだ順番に応じて読み込むことが可能となる。以前のNTMでは連続した位置、例えばiの次は$i+1$といったように書き込むようにしていた。しかし、コンテンツベースのアテンションの仕組みがあると、書き込む位置は位置上としては非連続となり、NTMは位置情報ベースだけでは途中で記憶をたどれなくなるという問題があった。DNCはこの時間遷移行列Lを使うことで外部記憶上に仮想的な遷移グラフを作り、多様な記憶の系列を保存し、読み込むことが可能となった。

3つ目は、使っていない空いている領域にアテンションを掛ける仕組みである。これはメモリ割り当てにおけるフリーリストと同様の役割を果たす。各位置にその位置がどの程度重要な新しい情報を含んでいるか、過去に使われたかといった情報を保存しておく。そして書き込む際は最も使われていない領域にアテンションを掛け、そこへ書き込むようにする。

この3つのアテンションの仕組みを組み合わせて最終的な読み込みヘッダ、書き込みヘッダを決める。DNC全体は誤差逆伝播法を使って学習可能であり、教師あり学習や強化学習と組み合わせて使うことができる。

質問応答タスクで過去最高のスコア

DNCの性能評価として、最初に米Facebook社のAI Researchが公開しているbAbIデータセットでDNCを評価した。これは20種類の質問応答タスクからなる。例えば「羊は狼が怖いです。ドリーは羊です。ネズミは猫が怖いです。ドリーは何が怖いですか」(答：狼) といった質問応答である。この場合は、短期記憶、推論能力に加えて、関係の無い情報に惑わされない（ネズミは関係ない）能力が求められる。

DNCは20種類の質問と10000の質問応答タスクにおいて3.8%のエラー率となり、従来手法の最高精度である7.5%を大きく超えた。また、これは同様に短期記憶を実現するLSTM (25.2%) やNTM (20.1%) のエラー率を大きく上回っている。

bAbIは自然言語データではあるが、そこに書かれている事実は（羊－怖い↦狼）のようにエンティティをノード、関係をエッジ上のラベルとしたラベル付き有向グラフで表すことができ、質問応答はそのグラフ上での操作に対応付けることができる。DNCは時間遷移行列を使ってグラフ構造を記憶でき、コンテンツベースのアテンションでノードや枝の類似性に従ってたどることができると考えられる。

このグラフを扱う能力をさらに検証するため、2つ目の実験としてグラフデータに対するタスクでDNCを評価した。具体的にはランダム有向グラフや地下鉄の路線図、家系図といったグラフデータを与えた上で、巡回問題（例：大手町駅から三田線、次に有楽町線に乗って豊洲に行くにはどの駅をたどるのか）、最短経路問題（例：大手町から渋谷駅まで最短でどのような経路をたどればよいか）、推論問題（例：徳川慶喜の母方の大叔父は誰か）といったタスクで評価した。

グラフ情報は、bAbIと同じように各エッジを順番に提示することで読み込ませている。学習の際は、カリキュラム学習を利用しており、簡単で小さなグラフでタスクを学習させてから、順に大きくて複雑なグラフを与えて学習を行った。DNCは多くのグラフの問題を解くことができ、従来の学習手法では全く解けないような場合も解くことができた。

最後のタスクとして、Mini-SHRDLUというブロックパズルゲームを解かせた（図7-1）。このタスクでは、例えば、3×3のマス目に1から6の数字が積み重ねられており、各列の一番上にある数字を別の列の一番上に移動することができる。その上で6は2の下、4は1の右といった拘束条件が与えられ、初期状態から、与えられた拘束条件を全て満たすような状態への遷移方法を求めることが目標となる。DNCはこのMini-SHRDLUもカリキュラム学習を通じて学習することができ、多くの問題を解くことができた。興味深いことに、DNCはこれから実行する行動を最初に外部記憶へ書き込んでいることがわかった。つまりDNCは計画を立てた上で、実行してい

ることがわかった。

人では思い付かないアルゴリズム考案の可能性も

ニューラルネットワークが連続的で確率的な問題だけではなく、このような離散的で手続き的なタスクを学習できると示したことは非常に重要である。人はこれまでアルゴリズムを発明し、プログラムを書くことができたが、全てが得意なわけではない。例えば、性能が出て正しく動作する並列アルゴリズムを書くことは困難である。DNCは並列で読み書きをし、効率よく解くアルゴリズムを発明し、今後の進化次第では人では思いもつかないようなアルゴリズムを発明できる可能性もある。

また、外部記憶が拡張可能であり（アテンションの仕組みは外部記憶のサイズに依存していない）、次々と新しいタスクを学び続けることができる方向が示されたことも重要である。DNCの成果は既に実用段階にある画像認識や音声認識と比べてまだ萌芽的ではあるが、今後大きな発展が期待される。

---

1) A. Graves et al.,"Hybrid computing using a neural network with dynamic external memory," Nature, 538, pp.471-476, 2016.

（2017年1月号掲載の記事に加筆）

4

アプリケーション

第8章

画像

8-1 画像認識で大きな成果上げるCNN：分類のエラー率は1年ごとに半分近くに減少

深層学習（ディープラーニング）が大きく成功した一分野が画像認識である。この数年で画像分類や物体検出、セグメンテーション（画素ごとにクラス分類するタスク）の精度が飛躍的に向上した。従来の画像認識技術は、SIFTやHOGなどに代表されるような専門家が設計した特徴を使い、認識を行っていた。これに対し深層学習は、特徴を学習し、特徴計算から最終的なタスク（分類、検出など）まで一貫して同時に学習できる。

画像をデータとみた場合、2次元上（x, y）の各画素に複数チャンネルc（白黒の場合1、カラーの場合は3）を持った（x, y, c）の3次元データ（白黒の場合）と考えることができる。ニューラルネットの各層はこのような3次元データから3次元データ（入力層以外はチャンネル数は任意、例えば32や512など）への変換を繰り返していき、タスクに応じた値に変換を行う。

現在、深層学習による画像認識では、CNN（畳み込みニューラルネット）が広く使われている。CNNは全てのニューロン間をつなげた総結合層と比べて2つの特徴がある。

1つ目は、CNNでは層間の接続が局所的であることだ。3x3、7x7といったカーネルと呼ばれる領域の全てのチャンネルと、出力の1つのチャンネルがつながっている。複数チャンネルを出力する場合は、出力チャンネル数分、このような結合がある。このつながり方は、画像では近い位置の情報が影響し合うという事前知識を使って考案されたものだ。

もう1つの特徴は、同じ重みを画像内の違う場所で共有することだ。これは、写っているものが画像内で移動しても、その意味は大きく変わらないという、画像の平行移動不変性に関する事前知識を使っている。これにより、学習対象の重みのパラメータを減らすことができ、学習効率が上がるだけでなく、計算効率も上がる（必要メモリバンド幅を減らせる）。この畳み込み層を通じて、CNNに入力された画像は次第により小さく、チャンネル数が大きい画像に変換されていく。例えば、画像分類の場合、最後の層では大きさ1x1でチャンネル数は数百から数千の特徴に変換され、これを使って分類を行う。

コンテストにより毎年認識率が向上

初期の深層学習による画像認識の進化の歴史を、「ILSVRC（ImageNet Large Scale Visual Recognition Challenge）」という画像分類コンテストの優勝チームから紹介しよう。

このコンテストでは画像分類や、物体検出などのタスクがある。例えば、画像分類タスクでは、与えられた画像が1000クラスのどれに相当するかという分類問題を扱う。また進化の様子が分かるように優勝チーム以下、上位5位のエラー率を付記する（表8-1）。ただし、タスクは主催者により毎年微妙に変えられており、精度も手法以外の工夫（ハイパーパラメータの調整や学習手法など）で変わるため、あくまで参考程度にみてほしい。

2012年のImageNetコンテストはカナダUniversity of TorontoのAlex Krizhevsky氏、Ilya Sutskever氏、Geoffrey Hinton氏らのSupervisionが優勝した（15.3%）。彼らは単に優勝しただけでなく、2位（26.2%）以下に大差をつけたため、世の中に深層学習の登場を鮮烈に示した。

この3人はその後、深層学習の中心人物として活躍している。この時、Supervisionで使われた畳み込み層とプーリング層を交互に繰り返すニューラルネットは、開発者の名前からAlexNetと呼ばれており、現在も多くのCNNで使われる祖先のような存在となっている。

ちなみにその後の研究では、プーリング層を使うと情報が落ちてしまう、または情報が偏ってしまうという問題が分かってきた。このため最近では全て畳み込み層で処理し、

表8-1 ILSVRCの上位5位の成績

2012年	
チーム名	エラー率
SuperVision	15.3%
ISI	26.2%
OXFORD_VGG	27.0%
XRCE/INRIA	27.1%
University of Amsterdam	29.6%

プーリング層は計算量削減の目的のみで使うことが多くなっている。

2013年以降、上位勢は全て深層学習

2013年は上位が全て深層学習ベースの手法になり、米New York UniversityのMatthew Zeiler氏によるClarifaiが優勝した（11.7%）。Zeiler氏はニューラルネットの学習を調べるために逆畳み込み（deconvolution）を使い、ニューラルネットのモデルを最適化した。その後、逆畳み込みは画像生成やセグメンテーションでも広く使われるようになった。

2014年のコンテストでは米グーグルによるGoogLeNet（6.7%）が優勝した。GoogLeNetは、Inceptionと呼ばれる異なるカーネルサイズを持つ畳み込み層を組み合わせて、次の層のチャンネルを計算するという特徴がある。他のCNNに比べて計算量を小さくできる。また、2位の英Oxford UniversityのAndrew Zisserman氏らによるVGG（7.3%）も注目されている。VGGは3x3と1x1の畳み込み層を繰り返し、11～17層といった深いCNNを利用する。VGGは構造が非常に単純で、メモリ使用量も少ないため広く使われている。

この後、グーグル、マイクロソフト、中国Baidu社の3社で最高精度の更新競争がしばらく続く。こうした中で、深層学習が人間による分類性能の目安の5%を切り、「機械の精度が人を超えた」とニュースになった。ただし、この5%は個人が試しに測った精度であり、人間でも一定程度の訓練をすれば3%程度は達成されるだろうといわれている。

2015年のコンテストで優勝したのは、Microsoft Research Asia（MSRA）で、152層からなるCNNを利用した（3.6%）。これだけ深いニューラルネットは今まで学習することができていなかったが、今回、同社はResNetと呼ばれる層を利用することで、深い層でも学習を容易にした。ResNetは1000層を超える場合でも学習できることが示されている。2位のグーグルもGoogLeNetを改良してきており、恐らく統計的にはほぼ差の無い精度を達成している（3.6%）。

キャプション生成など他のタスクでも成功

2012年の深層学習の登場以来、毎年エラー率が半分近くに下がるということからも、深層学習による画像認識の進化がいかに急速か分かるだろう。この他の物体検出、セグメンテーション、動画分類、画像からのキャプション生成といったタスクでも、深層学習による画像認識は広く成功している。

深層学習による画像認識は、産業界でも広く利用され始めている。自動車のADAS（先進運転支援システム）や自動運転における車両や人の認識、ロボットにおける物体認識などである。これらの用途の場合、リアルタイム性、携帯機器でも動くような省メモリ化、省計算量化、省電力化（データの移動を少なくする）といった工夫が多く研究されている。

2013年

チーム名	エラー率
Clarifai	11.7%
NUS	13.0%
ZF	13.5%
Andrew Howard	13.6%
OverFeat-NYU	14.2%

2014年

チーム名	エラー率
GoogLeNet	6.7%
VGG	7.3%
MSRA Visual Computing	8.1%
Andrew Howard	8.1%
DeeperVision	9.5%

2015年

チーム名	エラー率
MSRA	3.6%
ReCeption	3.6%
Trimps-Soushen	4.6%
Qualcomm Research	4.9%
VUNO	5.0%

（2016年2月号掲載の記事に加筆）

8-2 GLOM：パース木による画像認識の実現を目指して

人間は画像認識を行う際、対象をパース木として認識していると考えられる[1]。パース木の根（計算科学で扱う木のように根を一番上、葉を一番下と逆さまで表された木構造である）は対象の全体の概念を表し、各節点の子は節点が表す概念の部分を表す。例えば、猫を認識したパース木は猫全体を表す根があり、その子に、頭、胴、足を表す節点があり、さらに頭を表す節点の子に口や眉毛を表す節点、足を表す節点の子に爪に対応する子が対応する。各節点はその部位で局所座標系を持ち、親や子間の枝には座標系を変換する行列が付随している。親の座標系を移動、回転した場合は、対応する子孫の座標系も自然に変換される（CGではこうしたモデル化は一般的である）。猫を概念上回転させれば、頭や胴、それに付随する口や爪も違和感なく回転するだろうが、その実現には上のような座標変換が必須である。また、認識の際に部分がわかれば親の認識に役立ち（耳や爪を見れば猫か犬だろうと予測でき）、逆に親が分かれば子の認識に役立つ（車だとわかっていれば目のように見える部分はフロントライトだというように）。

このように認識結果としてパース木を得ることで、人間のように汎化する画像認識システムを作れると考えられてきた。特に深層学習の産みの親の一人であるカナダUniversity of Toronto名誉教授のHinton氏は1980年頃から心理学的な実験結果から人間はこのように認識していると考え、CNNだけではその実現が難しいとして、様々な方法を提案している[1]。

そんな中、Hinton氏は2021年2月に画像認識結果をパース木で表現できるシステムGLOMの構想を発表した[2), 注1)]。GLOMはTransformer（自己注意機構）、NeRFなどで注目されるニューラル場、自己教師あり学習で成功している対比学習、学習済みモデルを他のモデルに転移させる蒸留、そしてカプセルのアイデアを組み合わせた、野心的なシステムである。また画像だけでなく言語も対象としており、現在の自然言語処理でTransformerを使った自己教師ありモデルがなぜ成功しているのかも説明することを試みている。このGLOMについて今回説明していく。

カプセルの問題点

そもそも認識結果をパース木で表現することは自然に見えるが、人間の脳内でそれを実現しようとすると困難である（同じように解析結果をパース木で扱う自然言語処理も同様の問題がある）。脳はハードウエアとしては瞬時の再構成は不可能であるため、パース木に対応する木構造を神経回路網で毎回構築しているとは考えられない。さらに認識対象が次々と変わる場合、すぐに対応する全く異なる形を持ったパース木を導出でき、また複数の認識対象を扱う際は複数のパース木を扱える必要がある。

このパース木による認識を脳内で実現可能な方法として、Hinton氏らは2017年にカプセルを提案した[3]。カプセルはパース木の各階層に様々なタイプのカプセルをあらかじめ用意しておく。例えば、1番下の階層は眉毛、爪、指、2つ目の階層には頭、手、胴といった具合だ。そして認識対象毎に、それに対応するカプセルだけを発火させパース木を抽出する。また、各カプセルは姿勢や属性などを表す状態を保持し、カプセル間が持つ状態間の制約（例えば胴と腕は相対的な位置として、このように接続しているはず）を満たすかを調べる。ルーティングアルゴリズムによって制約を満たすパース木の抽出と状態決定が同時に行われる。

カプセルは画期的だったが重要な問題点が2つある[3]。1つ目は認識対象毎に別のカプセル群を使う必要があることだ。ほとんどのカプセルは使われず非効率であり、複数対象を同時に認識することが困難である。2つ目に各カプセルは別々のタイプを扱い、共通点がある概念を効率的に扱えない。例えば、車のヘッドライトと人間の目は概念として共通点は多くあるが、別のカプセルとして扱う必要がある。また、何より汎化性能で現在のCNNを改良したモデルや自己注意機構を使ったモデルに劣っている。GLOMではこれらの解決を目指した。

GLOM万能カプセルとカラム集合による認識

GLOMはピクセルやパッチなどの局所領域に対応するカ

ラムの集合からなる。各カラムは複数階層（5程度）からなり、その局所領域のパース木を表す。例えば、あるカラムが猫の耳の一部分に対応していると考えよう。この場合、一番下の階層は毛、次は耳、その次は顔、そして最後の階層は猫という概念に対応する埋め込みベクトルを持つと考える。カラム毎にパース木のその領域に対応する根から葉までのパス上の概念を保持しているといってもよい。このようにしてカラム集合全体でパース木を表すことができ、また認識対象が複数物体で構成される場合も対応することができ、カプセルの1つ目の問題は解決される。

世の中にはこれより多くの階層がある場合も多く存在するが（例えば原子から宇宙スケールまでには数十から百の階層があってもよい）、その場合はスケールの一部分だけを抜き出しズームアップした上で扱っていると考えている。

また、全カラムは同じパラメータを共有しており、全く同じ挙動をとる。入力の違いによってそれらが違う概念に収束する。これは入力座標を与えて異なる関数挙動を示すニューラル場のアイデアと同様である。カラムの各階層は従来のカプセルのようにタイプ別のカプセル集合を用意するのではなく、万能カプセルを1つ用意しておき、位置情報や周辺情報によってその万能カプセルが特定のタイプに分化すると考えた。論文中では類推として、万能カプセルが周辺情報で特定のタイプに分化していくのは、細胞の分化において、全ての細胞が同じ設計図（DNA）を持ちながら周辺の刺激によって別の種類の細胞に分化していく現象と似ていると説明している。これにより様々な概念を1つのカプセルで共通化して効率的に持つことができ、カプセルの2つ目の問題点を解決できる。

カプセル同様、GLOMもボトムアップとトップダウンの両方の情報を使って各概念を決めていきたいため、静止画像からパース木をフィードフォワードで一発で出力するのでなく、各状態を親、子、周辺状態と逐次的に合意をとりながら決めていく。GLOMは時刻毎に入力を受け取り、状態を更新していく。静止データは各時刻毎に同じ入力が与えられ状態を更新していくモデル、動画などはそのまま時刻毎に対応する入力が与えられる問題だと考えよう。位置 p にあるカラムの時刻 t における l 層目の状態ベクトルを $h^{(t,p,l)}$ と表すことにする。この状態 $h^{(t,p,l)}$ は次の4つの和として決定される。

1. 下層の状態 $h^{(t-1,p,l-1)}$ からの予測
2. 上層の状態 $h^{(t-1,p,l+1)}$ からの予測
3. 同じ層の状態 $h^{(t-1,p,l)}$
4. 注意機構で集約された同じ層の周辺状態 $h^{(t-1,q,l)}, q \in N(p)$

ただし $N(p)$ は p の位置の近傍のカラム集合を表す。ここで情報集約の際、1と2が非線形変換も使う複雑な予測であるのに対し、3と4は状態そのものという単純なモデルをあえて利用する。特に4は自己注意機構としてクエリ、キー、値はいずれも状態そのものを考える。この場合、周辺の自分と似ているベクトル集合のみからの平均値を使うようにみなせる。

例えば物体境界近くの物体内部側に位置するカラムの状態における4は物体外の状態は注意機構でフィルタリングされ、物体内部側にあるカラムのベクトルのみの情報を集めることができる。

学習はBERTや自己教師あり学習の対比学習で使われているような摂乱された入力全体から一部の入力を予測するタスクで行われるか、対比学習で行うことを考えている。これであれば外部から教師信号を与えずともデータの多くのデータ構造を学習することができる。対比学習は実験的に優れているだけでなく理論的にもデータ生成過程の逆写像を学習し、因子を推定することができるという興味深い考察もされている[4]。

このGLOMはアイデアの提案論文であり、元々はチーム内で実験を行ってもらうためにアイデアをまとめた資料が大きくなったものだと説明している。

脳内でニューラルネットの手法を実現できるか

この論文ではさらに脳内でどのように最近のアルゴリズムを実現するかについて面白いアイデアを紹介している。

1つ目は重み共有についてである。CNNやRNN、GNNのようにニューラルネットワークにおいて重みを共有するというのは非常に有効な帰納バイアスであり、学習の効率化、強力な正則化として重要だとわかっている。それに対し、脳内では重みというはシナプス結合強度やニューロン内での状態変化で表されるならば、ある場所の重み群がそっくりそのまま異なる場所で共有される仕組みの実現は難しい。そのため重み共有はあくまで工学的な工夫であると考えられてきた。

これに対し、異なるカラム間で同じ入力に対し出力結果が一致するように学習する「蒸留」を使うことで重み共有と同

じような正則化効果が得られるのではないかと提案している。蒸留の場合、内部の重み表現は異なるが関数としては同じになるように制約がかかり、重み共有の正則化と同様に効果が得られると考えられる。蒸留であれば脳内でも実現は簡単である。さらに近年の理論解析では自分で一から学習するより、他が学習した結果を蒸留して教えてもらったほうが予測に使う特徴のカバレッジが増え、汎化しやすいということもわかっている[5]。単に重み共有するよりも汎化としては効率的な可能性がある。

2つ目は対比学習が脳内でどのように実現されているかについてである。対比学習は正しいペア（正例、例えば動画の隣接するフレーム、同じ画像に異なるオーグメンテーションをした2つの結果）が、正しくないペア（負例、動画の隣接していないフレーム、ランダムに選んだ画像をオーグメンテーションした結果）より似ているような埋め込みを学習することで実現される。人間が起きている時は正例は時系列の隣接フレームなどから手に入るが、負例を扱うには全く異なる記憶を呼び起こさなければならず、実現は難しい。一方で、寝ている間には昼間記憶している状態が早送りされたりシャッフルして思い出される現象が起こることがわかっている。これらが負例となり、それを使って対比学習しているのではないかというアイデアである。睡眠が学習に重要なことはほぼ確実だが、従来の睡眠の作用に関する研究では、関係の無い接続を整理するという考え方が主流だった。生成モデルのヘルムホルツマシン（VAEなどの元になっている）が、寝ている間にサンプルを生成し、それを使って認識器の勾配を推定しているという説もあったが、今回はそれが対比学習に使われているのではないかというアイデアである。

画像認識は毎年驚くほどの性能向上をあげてきたが、一方で人間では簡単に解けるタスクの多くが解けていない（例えば姿勢推定）、もしくは学習に膨大な量のデータを必要とする。今回のGLOMのようなシステムによって人が実現しているような画像認識能力を獲得できるのではないかと期待される。

---

1) G. Hinton,"Some demonstrations of the effects of structural descriptions in mental imagery," Cognitive Science, vol.3, issue 3, pp.231-250, 1979.
2) G. Hinton,"How to represent part-whole hierarchies in a neural network," https://arxiv.org/abs/2102.12627
3) S. Sabour,"Dynamic routing between capsules," NeurIPS 2017.
4) R. Zimmermann,"Contrastive LearningInverts the Data Generating Process," https://arxiv.org/abs/2102.08850
5) Z. Allen-Zhu,"Towards Understanding Ensemble, Knowledge Distillation and Self-Distillation in Deep Learning," https://arxiv.org/abs/2012.09816

注1) GLOMとは、「塊にする」という意味を持つ単語「agglomerate」に由来しているという。

（2021年5月号掲載の記事に加筆）

第9章

音声

9-1

WaveNet：自然な音声や音楽を生成可能なニューラルネットワーク

ニューラルネットワークは画像認識や音声認識といった認識のタスクだけでなく、生成の分野でも成功している。例えば、GANをはじめとした手法では、現実の写真と見違えるような自然画像や人が書いたような絵を生成可能となっている。

この認識と生成のタスクは表裏の関係にある。認識は与えられたデータからそのデータの因子またはそれを構成する要素を推定するタスクであり、逆に生成は因子からデータを生成するようなタスクである。物理学者のRichard Feynman氏はこれを「作ってみることができなければ理解したとはいえない」と端的に言い表している。

2016年9月に英DeepMind社が自然な音声を生成可能なニューラルネットワークであるWaveNetを発表した[1]。WaveNetによって合成された音声は現在の最高レベルの音声合成と比べて、主観ブラインドテストで50%もの性能向上がみられた。実際に生成された音声や音楽については同社のWebサイト[注1]で聞くことができるので聞いてみて欲しい。

従来とは全く異なるアプローチを採用

WaveNetは従来の音声合成とは全く異なる手法を用いて音声を合成している。従来手法では、例えば1人の話者から採取した大規模な音声データをフラグメント（断片）に分け、それらを組み合わせる手法が使われていた。この場合、話者を変えたり、強調や感情を入れたりすることが難しかった。一方で、音声モデルをパラメータで表現する手法の場合は自由度は上げられるが、不自然な音声が生成される問題があった。

WaveNetはこれらの手法とは全く異なるアプローチを取っている。16KHzの音声データであれば1秒間に16000点の連続値があるデータとみなし、各値を順に次々と生成する問題とみなした。この場合、音声は低周波から高周波領域まで相関があるので、数千ステップも離れたデータとの相関を保てるようなモデル化が必要となる。

WaveNetがこれを実現できたのには3つの技術的背景がある。

自己回帰モデル

1つ目は自己回帰モデルである。自己回帰モデルとは自分が過去に出力した値に依存して次の値を出力するようなモデルである。

例えば、時刻tの値をx_tとした時、x_tが出力される確率を$p(x_t|x_{t-1}, x_{t-2}, \ldots)$のように定義する。これらの条件付き確率の積は同時確率のため、自己回帰モデルはデータxに関する次のような生成モデルとみなすことができる。

$$p(x_1, x_2, \ldots x_N) = \prod_{t=1}^{N} p(x_t|x_{t-1}, \ldots)$$

自己回帰モデルは、複雑な確率モデルを単純な確率分布の積に分解できるために複雑な確率分布を学習しやすい。

現在のニューラルネットワークの生成モデルの主流は潜在変数モデルを使ったモデルであり、$\int_z p(x|z)p(z)dz$のように表される。全出力が潜在変数に条件付けられ、まとめて生成されるようなモデルである。潜在変数の場合、データ全体の因子を表すことは得意であるが、データの詳細な相関を表すことは難しい。

2015年末頃から画像生成に自己回帰モデルを使ったモデルが登場し[2,3]、それらが従来の潜在変数モデルを使った手法よりも尤度が高く（つまり実際に観測されたデータに対し、より高い確率を割り当てられる）、生成された画像の品質も高いため注目されていた。画像の場合、画素同士の依存関係に方向はないが、適当に生成の順序を決め、各画素を左上などから順番に既に生成した画素に条件付けをして生成をしていく。

自己回帰モデルの場合、データを1つずつ順に生成をするためステップ数が大きくなる問題があったが、学習についてはCNN（畳み込みニューラルネットワーク）のようにまとめて生成をさせるようにすることで、学習の効率を大きく上げることができた。

この場合、CNNとしては過去の情報に対応するニューロンから未来の情報に対応するニューロンにしかつながらない

ようにすることで達成できる。

CNNでのつながりをスキップ

2つ目はDilated Convolutionである。CNNは層間で近傍同士がつながっているような構造をとっている。ある層のニューロンが下の層のどの範囲のニューロンとつながるかのサイズをカーネルサイズとよび、その下の層も含めて対応する入力の範囲を受容野と呼ぶ。

例えばカーネルサイズが5×5の場合、あるニューロンは下の層の5×5の領域とつながっている。次の層のカーネルサイズも5×5の場合、5×5のそれぞれのニューロンが5×5でつながるので2つ下の層の9×9の領域が対応する。一般にカーネルサイズがk、層数がlの時、その受容野のサイズは$l(k-1)+1\times l(k-1)+1$となる。

CNNでは層を増やしても、その受容野サイズは線形にしか増えないので、入力全体を受容野にしたい場合は層の数を入力サイズに比例して増やさなければならない。この問題はCNNにおいて上の層で一定間隔ごとにニューロンをスキップして詰めることで解決できる。

例えば上の層で1個おきにスキップするようにすれば、受容野は2倍ずつ広がる。しかし、この場合は対応するfeature mapのサイズも1/2となる。最終的にfeature mapを1×1にしたい分類タスクのような場合はスキップで十分だが、セグメンテーションや今回の自己回帰モデルのような入力と出力のサイズが同じ場合、feature mapのサイズを入力と同じようにしたい。

Dilated Convolutionは上の層になるほどk倍スキップしたつながり方をすることでこれを解決する[6]。例えばk=2で4層の場合、1番上の層は2^3=8個おきにつながり、2番目の層は2^2=4個おきにつながり、3番目の層は2^1=2個、4番目の層は2^1=0個おきにつながる。この場合、1番上の層のニューロンの受容野のサイズは16×2=32となる。

WaveNetは9層のDilated Convolutionを使い、512ステップ、受容野は1024であり、240msに相当する。このDilated Convolutionを複数層重ねて使う。実際には数万ステップ離れた値の情報も使って予測することが可能となる。

混合ガウシアンは使わず離散分布で生成

3つ目は連続値を混合ガウシアンでモデル化するのではなくカテゴリ値としてみなし、離散分布を使って生成する手法である。些細な違いのようだが、生成されるデータの品質が大きく変わってくる。このアイデアは、WaveNetと同じくDeepMind社が考案した画像向け生成モデルであるPixelCNNで最初に使われ、今回それを応用した。離散分布の場合、任意の確率分布を表現することができ、ガウシアンと違ってデータ分布に対する仮定が必要ない。一方、離散分布の場合、パラメータ数が多く推定が難しい問題があるが、ニューラルネットワークによってパラメータ共有することでこの問題も解決する。

今回の技術は音声認識にも波及

WaveNetは、音声を1つずつ生成するという非常にシンプルなアプローチで自然に聞こえる音声や音楽を生成できたという点で非常に意義が大きい。

今後の課題として一番大きいのは生成が遅い点である。WaveNetは自己回帰モデルを使っているため、生成時はデータを1つずつ順に生成しなければならない（学習時はこの問題がないことに注意）。新たに生成する場合、各層の計算の大部分は既に計算済みであるため高速化できるもの[注1]、リアルタイム合成をするためには、いくつかのブロックをまとめて生成するなどの工夫が必要そうだ。

もう1つの課題としては、認識への応用である。音声認識分野では従来、メル周波数ケプストラム係数のような工学的に設計された特徴が使われていたが、ここ最近は生の音声を直接モデル化し、このような特徴も学習で獲得するような流れに変わってきた。WaveNetを使ったモデル化では、生の音声データからの音声認識精度は最高精度を達成している。既存手法と比べると認識精度の面でまだギャップはあるものの、今後、従来の認識手法を性能面でも上回る可能性がある。

1) A. Oord et al.,"WaveNet: A Generative Model for Raw Audio,"http://arxiv.org/pdf/1609.03499v2.pdf
2) A. Oord et al.,"Pixel Recurrent Neural Networks,"https://arxiv.org/pdf/1601.06759v3.pdf
3) A. Oord et al.,"Conditional Image Generation with PixelCNN Decoders,"https://arxiv.org/pdf/1606.05328v2.pdf
4) F. Yu et al.,"Multi-Scale Context Aggregation by Dilated Convolutions,"https://arxiv.org/pdf/1511.07122v3.pdf
5) https://github.com/tomlepaine/fast-wavenet

注1) https://deepmind.com/blog/wavenet-generative-model-raw-audio/

（2016年11月号掲載の記事に加筆）

第10章

空間生成/認識

10-1

Generative Query Network：画像から3次元構造を理解し生成する

人は2次元の画像のみから物体の3次元情報を推定でき、その形状や位置関係、構成関係を認識することができる。さらに、物体をさまざまな視点から見れば、ほぼ完璧な3次元情報を得ることができる。例えば、部屋の中に入ってその中を歩き回って見ているうちに部屋の中のどの位置にどんな物体が置かれているのかという3次元地図ができてくる。このような3次元地図さえできれば、違う視点からどのように見えるのかを推定することもできる。

このような3次元構造の認識は多くのタスクの実現に重要でありこれまで多くの研究が存在する。複数視点の画像から3次元構造を推定するタスクはStructure From Motionと呼ばれる。この中で特に自己位置推定も同時にしながらオンラインで位置推定と3次元構造の推定を行うタスクはSLAMと呼ばれる。

また、3次元構造が得られている時、それがどのように見えるのかを解く問題はコンピュータグラフィックス（CG）のレンダリングの問題と同じである。レンダリングの場合は光源から放たれた光が各物体でどのように反射し、強度を変えて視点に入ってくるのかをシミュレーションすることで画像を生成する。

これらStructure From MotionとレンダリングはⅠ、どちらも非常に複雑なプロセスで構成され、綿密なモデリングと幾何計算によって作られてきた。さらに、認識と生成方法が問題になるだけでなく、3次元構造をコンピュータ上でどのように表現するのかが問題となる。

例えば、点群で物体を表現することを考えてみる。この場合、どんなに複雑な物体であっても表現することは可能だが、物体をひとかたまりとして操作するといったことが困難となる。また、点数が多くなるにつれて計算量も増加する。この他にもメッシュやディスク（法線方向と直径の集合で表す）、球面調和関数による物体の表現はそれぞれに利点と欠点が存在する。また、不確実な情報の表現も難しい。例えば、車の正面だけが見えていて、裏側が見えていない場合、その裏側の不確実性をどのように表現すればよいだろうか（セダンなのか、ワゴンなのか、はたまたリムジンなのかもしれない）。

3次元環境を連続値のベクトルで表現

英DeepMind社により発表されたGenerative Query Network（GQN）[1]は、これらStructure From Motionとレンダリングをニューラルネットワークで学習によって獲得する。複数枚の画像を見せるだけでその3次元構造を推定し、新しい視点からの画像を推定することができる。内部では3次元構造は特徴ベクトルで表現されており、特徴ベクトル上で物体の追加や変更といった演算をすることができる。また、不確実な3次元構造を扱えるように、復元は確率的な生成モデルを利用している。

GQNは学習時に入力画像x_i^kと視点情報v_i^kのペアからなる

$$D=\{(x_i^k, v_i^k)\}_{i=1...N, k=1...K}$$

を学習データとして利用する。ただし、Nはシーンの数、Kはそれぞれのシーンの中での記録された画像の数であり、視点情報v_iはカメラの3次元の絶対位置座標wとヨー角y,ピッチ角pからなる5次元ベクトル(w, y, p)から構成される。テスト時には新しいシーンにおけるM個の画像とその視点情報$\{(x^m, v^m)\}_{m=1...M}$が与えられ、クエリ視点v^qからの画像x^qを予測する。ここでMは$M\geq 0$を満たす任意の正数である。

GQNのネットワーク詳細については後半に述べるとし、まずはその実験結果を述べていく。実験はシミュレーション上の3D環境で行われた。まず、GQNは複数物体がある場合でも正しくシーンを予測することができた。また、物体の存在や位置に不確実性がある場合はその可能性をサンプリングすることができた。

さらにword2vecなどで得られる単語ベクトルの連続表現と同じように、3次元環境のベクトル上で加算減算を行い、物体の属性を変えることができることがわかった。このベクトル上では3次元環境の情報が分解されて表現されており、操作できるようになっている。例えば、青色の球体があるシーンに対応するベクトルから、赤色の球面があるシーンに対応するベクトルを引き、赤色の三角形があるシーンに対応するベクトルを足すと、青色の三角形があるシーンに対応するベクトルが作れる。これは属性のうち赤色、球体は打ち消し合

い、青色、三角形が残るためだ。

このように3次元構造はベクトルでは分解された形で表現されている。入力画像の代わりに、シーンに対応するベクトルを入力とした強化学習はタスクを非常に効率よく学習できることが示されている。

自己回帰型の生成モデルを利用

それではGQNの手法についてみていこう。GQNは潜在変数モデルを利用した画像の生成モデルを考える。このとき、条件付きとして他の視点からの画像とクエリ視点、$y=((x',v'),v'')$ を使う。

$$g(x|y)=\int g(x|z,y)\pi(z|y)dz$$

この生成モデルは z についての積分を含むが、ELBO（evidence lower bound）と呼ばれる変分下限は次のように求められ、この最大化はVAEなどでも使われる変数変換トリックを使って効率よく実現できる。

$$\log g(x|y)\geq\int q(z|x,y)\log\frac{g(x|z,y)\pi(z|y)}{q(z|x,y)}dz$$

$$=\mathrm{E}_{q(z|x,y)}\log g(x|z,y)-\mathrm{KL}(q(z|x,y)\|\pi(z|y))$$

ここで、$g(x|z,y)$ は生成器であり、$q(z|x,y)$ は真の事後確率 $p(z|x,y)$ を近似する推論器である。ここまでは条件 p を除いて変分自己符号化器（VAE）と同じである。

次に条件をどのようにモデル化するかについて述べる。入力の M 個の画像と視点 (x^k,v^k) は次のようにして1つのベクトルに変換される。

$$r=f(x^k,v^k)$$

この f は表現ネットワークと呼ばれ、

$$\tilde{v}^k=(v^k,\cos(y^k),\sin(y^k),\cos(b^k),\sin(b^k))$$

$$r=\sum_{k=1}^{M}\psi(x^k,\tilde{v}^k)$$

のように、各観察を独立にベクトルに変換し、それらの和を計算することで得られる。

この表現ネットワークは異なる視点間の関係を無視しており単純にみえるが、観察の順序に結果が依存しないという特徴があり、集合やグラフなどを入力とするネットワークでよく使われている。また関数 ψ は入力画像、視点を3次元構造を表すベクトルへ変換するニューラルネットワークである。

このようにして得られた推定された3次元構造を表す r とクエリ視点 v'' で条件付けし、生成器 $g(x|z,v'',r)$、事前分布 $\pi(z|v'',r)$、推論器 $q(z|x'',v'',r)$ をモデル化する。

これらはそれぞれ、出力の各次元を順に生成し、生成した値に条件付けして次の次元の値を決めるという自己回帰モデルを使ってモデル化される。こうした自己回帰モデルは複雑なモデルを扱えることがわかっており、PixelCNNやWaveNetなどで使われている。本手法では、RNNを使ってモデル化しており、convolutional LSTMを使って内部状態の更新と各ステップの出力を決定していく。

学習時にはELBOを最大化するように学習し、クエリ視点からの画像を生成する時には、他の視点からの画像で条件付けられた事前分布から画像を生成する。

本手法は、3次元構造の認識と生成がニューラルネットワークにより学習で獲得でき、3次元構造が演算可能なベクトルで表現できることを示した点で画期的である。

GQNをベースに多くの発展が見込める

今後はこの手法をベースにさまざまな発展が考えられる。今回の手法では、自己位置と姿勢は入力として与えられているが、自己位置と姿勢も学習から推定できるようになれば動画のみから3次元構造を復元できるようになる。それまでの速度、角速度の履歴から自己位置と姿勢を予測するように学習したRNNは脳内のグリット状細胞と同様の空間状のパターンに基づいて認識することがわかっている[2]。さらに、より高度な生成として、異なる時刻や異なる条件（例えば、このドアを押したらどうなるか）の場合もできるようになるだろう。

今回はシミュレーション上の単純な画像や3次元構造の場合に適用されたが、実世界の場合に同様にできるかは今後の課題だろう。一方で、タスクを達成するためには必ずしも画像の詳細な生成は必要なく、例えばインスタンス・セグメンテーションレベルで3次元構造や画像の推定ができれば多くの問題が解けるだろう。

非常に大きな3次元構造をどのように表現するのかも課題になるだろう。例えば都市レベルの3次元構造を扱うためには巨大なメモリの一部分にアテンションをあてて書き込んだり、読み込んだりするようになるだろう。

1) S. Eslami et al. "Neural scene representation and rendering," Science, vo.360, pp.1204-1210, 2018.
2) A. Banino et al. "Vector-based navigation using grid-like representations in artificial agents," Nature, vo.557, no.7705, pp.429-433, 2018.

（2018年8月号掲載の記事に加筆）

10-2 自己教師あり学習による深度と自己移動の推定

障害物や壁など対象物までの距離を測れるLIDARなどの深度（depth、距離）センサはロボットや自動運転車などで広く使われている。一般に深度センサは変調したレーザーなどを環境に照射し、その反射波の位相差や到達時間を測定することで深度を推定している。こうしたレーザーなどを自ら照射する方式はアクティブセンサと呼ばれる。生物ではコウモリなどが音波でこのような仕組みを利用している。

一方で人や動物は視覚で物体までの距離を推定できる。これらは自らは光を照射せず、受信した光のみを使うため、パッシブセンサと呼ばれる。視覚による深度推定では、両眼視差を利用したり、自分が動いた時に近くのものより遠くのものがゆっくり動くといった現象を利用して推定している。

対象物を片目で見たり、写真に写った物体までの距離を推定できることから分かるように、静止画像だけからも距離をある程度は推定できることが分かっている。本来であれば静止画像から深度は一意に求まらない不定問題であるが、物体や環境についての事前知識を組み合わせることで深度を推定している（これを利用して本当は凹んでいるのに盛り上がっているように見える騙し造形もある）。

深層学習で静止画像の距離を推定

近年の深層学習（ディープラーニング）を中心とした画像認識技術の進歩により、深度センサを使わなくても静止画像から直接深度を推定でき、さらに自己位置、自己移動を推定できるようになってきた。カメラは安く広く普及しておりコストを下げられるだけでなく、アクティブセンサが使えないような環境（たとえば鏡や透明な物体がある場合など）でも深度が推定でき、望ましい。

この実現のためには従来、カメラと深度センサを両方備えた撮影機器を用いて通常画像と深度（距離画像）のペアからなる教師ありデータを作成し、教師あり学習を行う手法が主流だった。しかし最近は教師ありデータを使わず、動画データだけを使って深度推定を学習する手法が登場し、教師あり学習に匹敵する性能を上げはじめている。この学習は教師なし学習といえるが、近いフレーム間で成立する制約を利用して教師を作っており、自己教師あり学習と呼ぶことができる。

さらに深度だけでなく副産物として自己位置推定も同時に解くため、動画から3次元環境とカメラ位置を同時に推定するSfM（Structure from Motion）を解くことができる。YouTubeなどの記録動画は膨大に存在するため、自己教師あり学習による深度推定は、ほぼ無限の学習データを利用することができる。今回はこの自己教師あり学習による深度、自己位置推定について解説する。

運動視差を利用

この学習ではある視点からみた景色が別の視点でどのように見え方が変わるのか（運動視差）ということを利用して深度の教師データを作成する[1]。

まず、この背景にある画像処理の基礎について簡単に説明しよう。空間中のカメラがある方向を向いて画像を撮影したとする。このとき、空間中に焦点距離だけ離れた位置に仮想的なスクリーンを設置し、カメラの中心と光源を結ぶ光線とスクリーンとの交点を並べたものが、得られる画像となる。

環境中のカメラの位置、姿勢を表す行列をカメラの外部パラメータ行列とよび、カメラの焦点距離や、カメラのスクリーンの原点からのオフセットから成る行列を内部パラメータ行列と呼ぶ。環境中の座標（世界座標と呼ばれる）に外部パラメータ行列をかけるとカメラ座標系での位置へ変換でき、さらにそれに内部パラメータをかけると画像中の位置へ変換できる。

ある位置、姿勢で撮影した画像 I_t 中の位置 p_t が、別の位置、姿勢で撮影した画像 I_s 中のどの位置 p_s に対応するかは次の式で求められる。

$$p_s \sim KT_{t\to s}DK^{-1}p_t$$

ここで、座標は同次座標を使って表し、K は内部カメラ行列であり D は深度、$T_{t\to s}$ は視点間の相対姿勢行列（つまり自己位置移動）である。この式は視点が移動した時、遠く

にある点は少しし か動かず、近くにある点は大きく動くということをより一般化した式である。

一般に対応する位置 p_s は整数ではなく小数となる。一方で元の画像 I_s は整数の位置上にある画素で表現されているため、厳密には小数で表される位置に対応する画素は存在しない。そこで、小数に応じた双線形補間を使い、対応する画素をその近傍の画素の重み付き平均として求める。この双線形補間はソフト注意機構と同じように、複数の対応する画素候補がある中でどれが実際に対応するのかを誤差逆伝播法で学習するのを可能としている。

カメラの見え方で色などが変わらないと仮定するならば、この p_s からサンプリングされた画素値と、p_t の画素値は一致するはずである。この画素値が一致するかを損失関数とする。

$$\sum_{p} |I_t(p) - \hat{I}_s(p)|$$

ここで、$\hat{I}_s(p)$ は対応する画素から計算された推定した画素値である。この画素値の計算には上記の対応する位置を求める式を利用する。この損失関数は T および D も変数とした関数である。

モデルは I_t を入力として深度を出力するネットワーク（Depth CNN）と、I_s と I_t を入力として相対姿勢を推定するネットワーク（Pose CNN）を用意し、それらの出力を使って先程の損失関数を定義する。この損失を最小化するように誤差逆伝播法を使ってDepth CNN、Pose CNNを同時に学習する。

この自己教師あり学習ではいくつかの仮定をおいている。1つ目は動いている物体は存在しないということだ。実際の映像では人や車、揺れる葉など動く物体が存在し、それらの姿勢変化も考慮しなければ対応する位置は求まらない。そのため学習の際には移動しそうな物体をあらかじめマスクして削除しておき、静止している環境、物体だけを対象に損失を計算することで精度を上げられると報告している[2-3]。

もう1つは、内部パラメータ行列 K が既知であり固定であるということだ。1つのカメラだけを使い、焦点距離を変えない場合はこれが成り立つが、様々な動画を集めてきた場合や、同じカメラであっても焦点距離が変わる場合はこの仮定は成り立たない。そのため、内部パラメータ行列をデータごとに推定する手法が提案されている[3]。一般に並進移動だけでは内部パラメータ行列は一意に決まらないが、回転を含む場合は内部パラメータ行列は一意に決まる。

最後の仮定は物体の対応する箇所は視点が変わっても画素値が変わらないことだ。実際は真の光源との関係によって視点が変わると画素値が大きく変わる場合がありうる。元の画素値を直接使わずにニューラルネットワークの途中の特徴量などを比較するといったことが必要となるだろう。

今後は他のセンサ情報とも融合

今後は動く物体がある場合もモデル化できたり、不確実性を考慮した深度推定もできるようになるだろう。また、必ずしも単眼カメラだけで処理する必要はなく、ステレオカメラや複数カメラ、速度情報[2]、IMU、GPSなどと組み合わせてより精度を上げることも考えられる。

---

1) T. Zhou et. al.,"Unsupervised Learning of Depth and Ego-Motion from Video," CVPR 2017.
2) V. Guizilini et al., "PackNet-SfM: 3D Packing for Self-Supervised Monocular Depth Estimation," https://arxiv.org/abs/1905.02693
3) A. Gordon et. al.,"Depth from Videos in the Wild: Unsupervised Monocular Depth Learning from Unknown Cameras," https://arxiv.org/abs/1904.04998

（2019年7月号掲載の記事に加筆）

10-3 3次元形状をどのように表現するか

物体の形状や環境（構造物や背景）の3次元情報をどのように表現するかは難しい問題である。基本的な問題に思えるが実際どのように表現するかを考えてみると難しい。形状表現は少ない容量（パラメータ数）で表すことができ、衝突判定や法線計算など様々な処理が高速に行えるか、似た部品や形状を共通に扱えるかが問題となる。

現在広く使われている表現方法として、点群、メッシュ、ボクセル、陰関数表現などがある。これらについて簡単に紹介しよう。

点群は物体が存在する領域をその物体の表面や内部に属する点の集合で表した手法だ。この点群は任意の形状を表すことができるが複雑な形状を表そうと思うと点数を増やす必要があり、保存容量が大きくなる。また衝突判定や法線推定などの処理も、その計算量はその点数に比例し大きくなってしまう問題がある。高速化のためのデータ構造、例えばKD木などを使って計算量を削減することもできるが最悪計算量を抑えることは困難であるし、実際にも計算量は大きい。

メッシュは物体の表面を三角形やn角系の集合で表現する。単純な形状であれば少ない数の三角形で近似できるため、点群に比べてずっと少ないパラメータ数で形状を表現できる。また、様々な処理もメッシュ数に比例する計算量で実現でき、計算効率も高い。一方で複雑な図形を表現する場合はメッシュ数は急激に大きくなる。また与えられた形状に対して、それに対応するメッシュ表現に変換することは容易ではない。多くの場合、システムが出した候補解を人手で修正する必要がある。

ボクセルは立方体の集合で形状を表現する。異なるサイズの立方体を組み合わせることも多い。ボクセル表現では他の表現と比べて表面だけでなく物体内部など立体情報の情報を表せる利点がある。一方、ボクセルは容量は大きくなりがちであり、また複雑な形状を表そうと思った場合、多数の小さな立方体を利用する必要があり、容量や計算量が大きくなりやすい。

陰関数は関数を利用して物体を表現する。具体的には座標 $u = (x, y, z)$ を与えてそれが物体の表面であれば0、内部であれば負、外部であれば正を返すような関数 $f(u; \theta)$ を用いる（様々な変種が存在する）。この表現では物体の表面といった情報は明示的に表現されないため、物体表面上の点の座標を求めるにはやや複雑な計算が必要となる。一方で複雑な形状であってもうまく関数を選ぶことで少ない数のパラメータで表現できる。レンダリング時に任意の解像度を選択できることも特長である。

これらの形状表現において、ニューラルネットワークを使った手法が成功を収めている。ここではメッシュを生成するPolyGen[1]、陰関数を使った物体の表現[2]、またそれに輝度情報も加えて任意の視点からの写実的なシーンの復元を可能としたNeRF[3]、そして基本的な図形の組み合わせで複雑な環境を復元できるBSP-Net[4]を紹介しよう。

メッシュを生成するニューラルネットワーク

はじめに、メッシュを生成するニューラルネットワークであるPolyGen[1]を紹介する。PolyGenは画像などを入力とし条件付けした上で、メッシュの生成モデルを定義し、メッシュをサンプリングすることができる。離散的な情報であるメッシュをはじめて高精度に生成できるようになったモデルである。

メッシュの生成モデルを考える場合、頂点数や面数が可変であるため、固定次元上で定義される確率モデル（例えばVAEやFlowベースモデル）が使えない。PolyGenは、これまで生成した頂点に条件付けして次の頂点を逐次的に生成する自己回帰モデルを利用する。各頂点を生成した後に、そこで生成を止めるかも確率的に決めることで可変数の頂点集合に対する生成分布を定義できる。

人手で設計したメッシュデータなどに基づいて、各頂点の生成の尤度を最大化するモデルを学習する。もともとのメッシュデータに生成順はないが、この研究では頂点はz軸順に従って頂点を順に生成する。

次にどの位置に頂点を生成するかは空間を量子化し、多項分布上の確率分布をモデル化し、そこからサンプリングすることで生成する。連続空間上で生成する場合は、ガウシアン分布が使われることが多いが、形状には対称性があり、次

の頂点の生成分布は多峰性になりやすい（例えば右手を生成するか、左手を生成するか）。そのため連続分布上の生成分布ではなく、多峰性も含め分布の表現力の高い離散分布上の確率分布を利用する。

また過去に生成した頂点にどのように条件付けするのかが問題となる。ここでは自然言語処理で成功を収めている Transformer（自己注意機構）を利用し、過去に生成したどの頂点に注目した上で生成するかを決められるようにしている。Transformerは可変個の要素に条件づけできることに注意されたい。

このPolyGenは机や椅子などの形状に対応するメッシュデータを直接生成できる。既存のメッシュ生成手法に比べて、複雑な形状に対応できるだけでなく、既存手法でありがちだった冗長なメッシュ分割がみられることも少ない。また、2つの形状をなめらかに補間してモーフィングのように変形しながらそれぞれの場合のメッシュも生成できる。

陰関数による形状表現

次に陰関数を使った形状表現を紹介する。陰関数による表現では座標 $u=(x,y,z)$ を入力とし、その座標が物体の外部か表面か内部かを返す関数 $f(u)$ を利用して物体を表現する。より情報量の多い物体表面までの符号付き距離を返すSDF（signed distance function）も多く利用される。形状表面は $\{u|f(u)=0\}$ となる座標集合で表せる。

複雑な形状を表現しようと思うと、関数も高い表現力を持つ必要がある。そのため、高い表現力を持つニューラルネットワークを使った陰関数表現が期待されてきた。そして、この数年で、ニューラルネットワークを陰関数として使った形状表現はこれまでの関数では不可能だった複雑な形状も表現できることが分かってきた。

この形状を表現する陰関数を学習する際には、法線（陰関数の入力に対する勾配）のノルムの大きさを1とする正則化が有効であることが分かっている[8]。この正則化によって人体など非常に複雑な形状も正確に表現できるようになる。

一方でニューラルネットワークは、可能な限り単純な関数を学習するという性質がある。この性質は一般の学習問題においては汎化として役立つが、複雑な関数を表現させようと思った場合、高周波成分、つまり物体形状が急激に変わるような場所を表現しにくい問題がある。

これに対し、Transformerなどで用いられている位置符号化を使うことで高周波成分も正確に表現できるようになる[9]。位置符号化は位置情報を位置周波数成分毎の表現で表す手法であり、異なる周波数の三角関数を使って座標を変換する手法である。

形状だけでなく、視点方向に依存した輝度も返すようにすることで写実的な画像を生成することもできる[9]。生成された画像はこれまでのニューラルネットワークを使った画像復元に比べて、現実と見極めがつかないぐらい写実的であり、DCGANやGGNに匹敵する大きなブレークスルーだと考えられる。ここでは省略するが、この手法ではその問題を解く過程で、深度推定や形状推定など多くの問題が教師なしで解けることを示している。

単純な形状の組み合わせで表すBSP-Net

これまで紹介してきた手法は、形状が組み合わせ可能だという性質を利用していない。実際、世の中にみられる多くの形状は単純な形状や既存の形状の組み合わせである。このような単純な形状の組み合わせでなることをうまく表現できると、その上で学習した手法が汎化しやすくなる。

BSP-Net[10]は単純な形状の組合せで形状表現することを学習で獲得する。単純な形状自体は陰関数表現を使う。

BSP-Netは3層から成る。1層目は空間を分割する平面を生成する。平面であれば直線（$ax+by+c=0$）、空間であれば平面（$ax+by+cz+d=0$）を表現するパラメータ (a,b,c) または (a,b,c,d) を生成する。この関数の値が負であるような座標の集合が物体内部である。2層目は生成された平面のANDをとった凸領域を学習する。複数の領域のAND領域はmaxで表現される。陰関数表現では負の値を持つ領域は物体内部であり、2つの関数が同時に負である領域は、この2つの関数それぞれが表す領域のAND領域を意味する。複数の領域のANDをとった場合の結果は凸形状を表すことができる。3層目はこの凸形状のORを扱う。これは先程の逆でmin操作で扱える。最終的にはmax, minを使うが、学習時にはこれらの操作では勾配が消失してしまうので、それぞれ勾配が消失しない表現を使って学習する。このようにして学習されたネットワークは陰関数表現で形状を表現するが、1層目、2層目は単純な形状に対応するような帰納バイアスを与えている。

BSP-Netでは、陰関数表現や他手法に比べて少ないパラメータ数で表現できるだけでなく、共通した部分的な形状を持つ場合も同じ表現で表すことができる。

形状表現の可能性

これら形状は画像や他の情報を基に条件付き生成することもできる。例えばNeRF[3]は複数の視点付き画像から環境の3次元情報を精緻に復元することができる。車載カメラやGPS情報付き画像などから都市や地域全体といった大規模の形状をニューラルネットワークで表現することも将来的には可能だろう。今後、ロボットの軌道生成時の衝突回避や、レイトレーシングCG、モデリングなど様々な分野で利用が進むと考えられる。

---

1) C. Nash et al.,"PolyGen: An Autoregressive Generative Model of 3D Meshes," https://arxiv.org/abs/2002.10880
2) A. Gropp et al.,"Implicit Geometric Regularization for Learning Shapes," https://arxiv.org/abs/2002.10099
3) B. Mildenhall et al.,"NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," https://arxiv.org/abs/2003.08934
4) Z. Chen et al.,"BSP-Net: Generating Compact Meshes via Binary Space Partitioning," https://arxiv.org/abs/1911.06971

（2020年5月号掲載の記事に加筆）

画像からの3次元シーン理解に向けた局所特徴量に基づく画像マッチング

対象物を違う視点から撮影した画像間では3次元空間中の同じ位置に対応する箇所について対応をとることができる。計算量や精度の観点から疎な特徴点を抽出し、それらの対応を求めることが一般的である。このようなタスクを局所特徴量に基づく画像マッチング（以降省略して画像マッチング）と呼ぶ。

この画像マッチングは様々なタスクで重要である。ステレオカメラで深度推定する場合は左の画像と右の画像の対応点（ステレオ対応点）を求め、三角測量と同じ原理で深度推定ができる。複数の異なる視点位置で撮影した画像群から3次元復元を行うStructure From Motion（SfM）は、画像マッチングを求め、それを基にカメラパラメータと疎な3次元点群を推定する。Simultaneous Localization and Mapping（SLAM）でもSfMと同様に対応点検出を行い、それを利用してリアルタイムの自己位置姿勢やそれに付随した3次元復元を実現する。動画の隣接するフレーム間で、ある時刻のフレームの各画素が次の時刻のフレームのどの画素に対応しているのかを求めるタスクはオプティカルフローと呼ばれ、対象物の動き検出や、自己位置推定に使われるが、これも広い意味での画像マッチングといえるだろう。

このように画像マッチングは画像から3次元環境を認識する上で重要な役割を果たす。既に画像マッチングを利用した製品は多く存在し、ステレオカメラによる深度推定や、外部センサなしでのVR機器の自己位置／姿勢推定（Oculus Questなど）、画像ベースのSLAMなどが実現されている。一方でチャレンジングな領域（屋外、撮影カメラ間のベースラインが大きい場合、精密な3次元復元が必要な場合）ではまだ多くの課題が残っている。画像マッチングにおいても深層学習を使った手法が大きく発展し、精度、速度面で従来手法を凌駕している。本稿では最新手法と今後の課題について解説する。

従来手法

はじめに、局所特徴量に基づく画像マッチングのワークフローについて簡単におさらいする。

1）画像から疎な特徴点を検出する。例えばコーナー領域は良い特徴点とされ、物体内部やエッジはどの点を選ぶかで自由度があり対応がとりにくいため良くない特徴点とされる。
2）特徴点毎に特徴記述子を求める。多くの場合ベクトルである。周囲情報から計算され、異なる視点からの画像中でも同じ特徴子となるよう、記述子はスケール、回転、アフィン、射影変換に対し不変であることが望ましい。
3）マッチング対象画像それぞれで求められた特徴点集合同士で、特徴記述子が似ているかどうかを求め、似ている場合は対応しているとみなす。

局所特徴量の検出および記述子の代表手法がSIFT（Scale-Invariant Feature Transform）である。1999年に発表されて以来20年近く使われており現在も多く使われている。SIFTの代表特許が2020年に切れ、これからさらに利用が広がる可能性がある。SIFTについては多くの解説文献があるためここでは詳しく説明しないが、その大きな特徴はスケールや回転不変で特徴を抽出できる点である。これらの不変性は現在の手法でも達成できていない場合が多い。またSURFやORBなど速度を改善したり、射影変換不変性などを導入し精度を改善した手法も提案されている。

このように従来手法は人が特徴点の検出方法や記述子の計算方法を設計してきた。しかし、画像分類や物体検出において人間が設計した手法をデータドリブンで学習したCNNモデルが凌駕したように、画像マッチングにおいてもデータドリブンで求めた手法が人が設計した手法を凌駕するのではないかと考えられ、多くの研究がなされてきた。画像分類や検出に比べてだいぶ時間がかかったものの、この数年は従来の人間が設計した手法を上回る手法が登場している。

SuperPoint／SuperGlue

ここでは学習ベースの画像マッチングの代表的な手法としてSuperPoint[1]、SuperGlue[2]について紹介する。これらはAR機器を開発するスタートアップのMagic Leap社の研究

者が発表した（研究の中心メンバーであるTomasz Malisiewicz氏は2020年に米Amazon Robotics社に移籍、Andrew Rabinovich氏は米Headroom社というスタートアップを創業し、CTOになっている）。

SuperPointとSuperGlueを組み合わせた手法はCVPR 2020やECCV 2020のVisual LocalizationやImage Matching Challengeなど7部門で優勝している。

SuperPointは局所特徴の検出および、その記述子をCNNを使って出力する。CNNは画像を入力とし、最初に1/64にダウンスケールした特徴マップを出力した後、検出用と記述子用のヘッドに分岐する。検出用のヘッドは各位置毎に65チャンネル出力し、64個の各位置で検出対象があるか、もしくは検出対象が1つもないかを出力する。記述子のヘッドも各位置毎に記述子を出力した後、バイキュービック補間を使って元の解像度にアップサンプリングする。

学習は人工的に作った教師ありデータを使って教師あり学習する。はじめに三角形や立方体、チェッカーボードなどの単純な図形からなる人工データを用意し、物体のコーナーや交差点を正解とした特徴点検出器を学習させる。この学習されたモデルをMagicPointと呼ぶ。次にMagicPointをMS-COCOなど実世界のデータに適用し特徴点を作り、さらに画像に射影変換（物体がフラットな平面だと仮定し視点変化をしていることになる）をかけてカメラ姿勢変換を模倣し、その結果得られる特徴点の対応を学習データとして利用する。人工的なデータであり射影変換のみで学習しているものの、想定以上に汎化することができる。

SuperPointはSIFTなど従来手法の精度を凌駕するとともに、全てCNNで処理できるため処理速度も速く、GPU1台で480×640解像度の画像に対して70fpsで動くと報告している。

次に検出された特徴点とそれらの記述子を基に、対応関係を求めるSuperGlueについて説明する。従来は記述子で近傍探索を行い最も似ている特徴点間で対応関係があるとし、さらに最も似ている特徴点と二番目に似ている特徴点との類似度の比が一定以上の場合のみ採用するなどのヒューリスティックスを利用しフィルタリングを行っていた。

SuperGlueは与えられた特徴点集合から対応関係を、グラフニューラルネットワークにより直接出力する。M 個の特徴点が検出された画像Aと N 個の特徴点が検出された画像B間の特徴マッチングを行うことを考える。目標は $M \times N$ の対応行列 $\boldsymbol{P}$ を出力することであり、i 番目の特徴点と j 番目の特徴点が対応する場合に $P_{i,j} = 1$、そうでない場合 0 となるような行列である。各特徴点は 1 箇所と対応するか、もしくは対応点が無い場合かを満たすため、$\boldsymbol{P}\mathbf{1}_M \leq \mathbf{1}, \boldsymbol{P}^T\mathbf{1}_N \leq \mathbf{1}$（$\boldsymbol{P}$ の各行、各列に 1 が高々1個しかない）を満たす。

各特徴点は画像中の位置と記述子で表される。はじめに位置を位置符号化を使ってベクトルとし、記述子と組み合わせる。次に自己注意機構を使って同じ画像内の特徴点集合から情報を集約（自己注意）するのと、相手の画像の特徴点集合から情報を集約（相互注意）し、内部状態を更新する。自己注意では他の特徴点からどのような物体を表しているのか、画像全体は何を表しているのかといった情報を集約することができる。相互注意では対応する特徴点候補の情報を集約できる。そして、特徴点毎にベクトル出力する。次に全特徴点間で内積を計算し、特徴点間の類似度を求め類似度行列を作成する。最後に類似度行列の各要素を負の輸送コストだとみなし、確率分布間の最小輸送コストを求められる最適輸送を使って最終的な対応を求める。与えられた輸送コストから最適輸送を求める操作は微分可能なSinkhornアルゴリズムを使って求められる。これによって全体の操作が微分可能であり、end-to-endで学習することができる。

SuperGlueを使うことによって、従来のSIFTを使った場合でも性能向上がみられるが、SuperPointなど学習ベースの記述子を使った場合に大きな性能向上がみられる。これはSIFT記述子を使った後の対応を見つける処理（類似度比フィルタ等）は多く研究が進んでいるが、学習ベースで獲得した記述子に対する対応点検出には改善の余地があり、学習ベースで最適化しやすかったとみられる。

その他の特徴点検出・記述子計算

SuperPointも、どの特徴点が良いか（コーナーを使うなど）は人手で設計している。画像マッチングがうまくいくような特徴点検出をend-to-endで学習する手法も登場している。検出ステップでどの特徴点を抽出するかは微分可能な操作にしにくいが、例えばDISC[8]は各操作を確率的な行動とし、強化学習を使って学習する。DISCはSIFTやSuperPointと比べてはるかに多くの有効な特徴点を検出することができ、後段のSfM処理などの性能を改善できている。また、HyNet[9]は記述子学習時に使うTriplet損失や正例間と負例間で異なる特性をもつことから、L2正則化付き内積やL2距離を組み合わせた損失を利用し、また適切な正規化を組み

合わせることで大きく性能を改善している。

これらの手法を利用した深度推定や自己位置推定、3次元復元は性能が向上し続けており、今後多くの用途が広がると考えられる。

1) D. DeTone et al., "SuperPoint: Self-supervised interest point detection and description," In CVPR Workshop on Deep Learning for Visual SLAM, 2018.
2) Paul-Edouard Sarlin et al., "SuperGlue: Learning Feature Matching with Graph Neural Networks," CVPR 2020.
3) M. J. Tyszkiewicz et al., "DISK: Learning local features with policy gradient," NeurIPS 2020.
4) Y. Tian et al., "HyNet: Learning Local Descriptor with Hybrid Similarity Measure and Triplet Loss," NeurIPS 2020.

（2021年1月号掲載の記事に加筆）

10-5

人や動物の空間理解の仕組みをAIに活かせるか

人や動物が空間をどのように理解し処理しているかについてある程度わかってきている。例えば、ネズミを使った実験では、脳内には特定の位置にいるときだけ反応する場所細胞、特定のグリッド上に存在する時に反応するグリッド細胞が存在し、これらを組み合わせて空間中のどこにいるのかを表現したり、ナビゲーションできることがわかっている。これらを発見した研究者らには2014年、ノーベル生理学・医学賞が与えられている。さらに、頭がどの方向を向いているのかを表す頭方位細胞、特定の距離と方向に壁などの境界が存在する時に反応する境界細胞もある。余談だがグリッド細胞は異なる周波数の三角関数を使った位置符号化手法（NeRFなど）、境界細胞は符号化付き距離関数とよく似ている。さらには、速度情報から現在の位置を予測するようにRNNを学習させた場合、RNNの各ユニットはグリッド細胞と同じような役割を果たすようになることもわかっている[1]。

一方で大きな謎として残っているのは人や動物は視覚や加速度などを自己中心表現（egocentric representation：カメラ座標系といってよい）で得るのに対し、これらの位置情報やナビゲーションは他者中心表現（allocentric representation：世界座標系といってもよい）で実現されており、自己中心表現を他者中心表現にどのように変換できるかが解明されていない。

これは3次元シーン理解とも関連する。観測から自分がどのような位置、姿勢であるかを推定したりナビゲーションを解く問題である。人は初めて入った建物や施設でも頑健に自己位置を推定し、入り口から出口までスムーズにナビゲーションすることができる。異なる視点からの情報も容易に統合できることから、自己中心の観測の相対的な情報だけでなく絶対的な空間表現を得ることができている。また、狭い場所でものを運ぶ際にも周辺にぶつからずに移動計画を立てることができる。現在のロボットプランニングでも同じようなことはできつつあるが、人のような頑健性や汎用性はまだ実現できていない。そのため、人や動物の空間理解や処理方法はどのように実現されているかという科学的な興味だけでなく実用的に役立つシステムを作る上でも活かせる部分があるのではないかと考えられる。

視覚情報による想起を速度情報のみから予測

この自己中心表現から他者中心表現への変換が、他者中心表現の教師データを使わずにNN（ニューラルネットワーク）で自己移動情報（加速度）から視覚で得られた情報を予測することで創発されることがわかってきた。人や動物でも実現可能な仕組みであり、かつ現在の脳の研究でわかっている機構と類似しているため、人や動物の機構と似ている可能性が高い。これについて詳しく説明する。

英DeepMind社のBenigno Uria氏らは、現在わかっている脳の記憶の仕組みを参考に次のようなNNアーキテクチャを考えた[2]。このモデルでは速度情報から、視覚情報によってどの記憶が想起されるのかを予測する。視覚情報自身を予測するものではないことに注意してほしい。なお、以降の説明では簡略化のため論文中にある時刻を示す添字 t は省略する。

まず、視覚情報（画像）をCNNで圧縮し符号 $\mathbf{y}$ を得る。この圧縮は自己符号化器をあらかじめ学習しておきその符号化器を使って行う。次に R 個のRNN $\{F_r\}_{r\in 1\dots R}$ を用意する。これらのRNNはそれぞれ異なる種類の入力情報（角加速度のみ、角加速度と速度、入力なし）を元に時刻毎に内部状態 $\mathbf{x}_r$ を更新する。そして、次のような S 個のスロットから構成される記憶領域を用意する。

$$\mathcal{M} = \{(\mathbf{m}_s^{(\mathbf{y})}, \mathbf{m}_{1,s}^{(\mathbf{x})} \dots \mathbf{m}_{R,s}^{(\mathbf{x})}\}_{s\in 1\dots S}$$

各スロットは視覚情報を符号化した $\mathbf{y}$ に関する記憶（上付き添字が $(\mathbf{y})$）とRNNからの内部状態に関する記憶（上付き添字が $(\mathbf{x})$）の2つから構成される。

まず、視覚情報を圧縮して得た符号 $\mathbf{y}$ からどのスロットを思い出すか、発火させるかの確率を次のように定義する。

$$P_{react}(s|\mathbf{y}, \mathcal{M}) \propto \exp(\beta \mathbf{y}^T \mathbf{m}_s^{(\mathbf{y})})$$

$\beta > 0$ は発火するスロットをどの程度疎にするのかを調整するパラメータである。また、速度情報からどのスロットが発火

するのかの確率を次のように定義する。

$$P_{pred}(s|\mathbf{x}_1,\ldots,\mathbf{x}_R,\mathcal{M}) \propto \prod_{r=1}^{R} \exp(\pi_r \mathbf{x}_r^T \mathbf{m}_{r,s}^{(\mathbf{x})})$$

$\pi_r > 0$ は各RNNが相対的にどの程度重要かを表すパラメータである。R 個のRNNによるエキスパートを使ったPoE（Product of Expert）モデルともいえ、全エキスパートのスコアが高い時のみ確率が高くなるような分布である。

そして、P_{pred} からの P_{react} のクロスエントロピー誤差が小さくなるように全てのパラメータを学習する。

$$L = \sum_{s=1}^{S} P_{react}(s|\mathbf{y},\mathcal{M}) \log P_{pred}(s|\mathbf{x}_1,\ldots,\mathbf{x}_R,\mathcal{M})$$

このようにして、速度情報だけしか使っていない P_{pred} が視覚情報によってどのスロットを想起するのかを予測できるように学習する。

この学習の際、RNNの記憶 $\mathbf{m}_{r,s}^{(\mathbf{x})}$ は更新されるが、$\mathbf{m}^{(\mathbf{y})}$ だけは学習目標であるため最適化対象から除いておく。代わりに一定の確率でランダムにスロット s を選択し、現在の状態をスロット s に次のようにアサインする。

$$(\mathbf{m}_s^{(\mathbf{y})}, \mathbf{m}_{1,s}^{(\mathbf{x})} \ldots \mathbf{m}_{R,s}^{(\mathbf{x})}) := (\mathbf{y}_t, \mathbf{x}_{1,t}, \ldots, \mathbf{x}_{R,t})$$

また、RNNは視覚情報を使わないように設計されているが、視覚情報由来の情報を使って位置や方向を利用できるようにし、また累積誤差を解消できるようにしなければいけない。そこで、一定の確率 $P = 0.1$ で内部状態の修正符号を次のように計算する。

$$\tilde{\mathbf{x}}_{r,t} = \sum_{s=1}^{S} w_{s,t} \mathbf{m}_{r,s}^{(\mathbf{x})} \quad w_{s,t} \propto \exp(\gamma \mathbf{y}_t^T \mathbf{m}_s^{(\mathbf{y})})$$

つまり、現在の視覚情報により想起されたスロットを想起できるような内部状態を計算する。そしてこの修正符号と現在の内部状態から、次の内部状態をRNNで計算する。ここでも意図的に、視覚情報が内部状態にどのスロットを想起するかという情報のみに限定して渡すようにしている。

NNによる自己中心表現から他者中心表現への変換の創発

このように設計したNNを異なる種類の入力を受け取るRNNグループを使って実験した結果、RNNは他者中心表現を計算するのに必要な情報を自然と獲得することがわかった。具体的には角速度情報のみを受け取って学習させたRNNは、位置に依存せず方向のみに依存して発火するようになり、頭方位細胞と良く似た機能を持つようになった。角速度情報と速度情報を受け取ったRNNは自己中心的境界細胞（egoBVC）と良く似た役割を持つことがわかった。例えばある方向の50cm先に壁がある時に発火するといったような細胞である。egoBVCは自己中心情報を他者中心情報へ変換する中心的役割を果たすとみられている。一切速度情報を受け取らず、視覚情報由来の修正のみで学習したRNNは他者中心BVCと良く似た役割を果たすことがわかった。BVCは場所細胞の活性に大きな役割を果たしていることが予測されている。これらRNNはそれぞれお互いが苦手としている部分を補いあって視覚がどの記憶を想起するかを予測できるよう学習し、その結果自己中心表現から他者中心表現に必要な情報を計算できるようになることがわかった。

このほかにも、作られたRNNは環境操作に対しても頑健であったり、新しい環境に対しても大きな学習率による更新可能な記憶領域のおかげですぐ適用できるなど優れた性能を持つことも確認された。

現在人が設計した空間表現やナビゲーション手法に加えて、これらの人や動物の空間理解から得られた知見をうまく組み合わせてより（人を超えるような）頑健で正確な空間理解やその上での処理が実現できるのではないかと期待している。

1) C. Cueva et al.,"Emergence of grid-like representaions by training recurrent neural networks to perform spatial localization," ICLR 2018.
2) B. Uria et al., "The Spatial Memory Pipeline: a model of egocentric to allocentric understanding in mammalian brains," https://doi.org/10.1101/2020.11.11.378141

（2021年2月号掲載の記事に加筆）

10-6

Rotation Averaging：高速かつ最適な姿勢推定を実現する

ビジョンやロボティクスタスクにおいて姿勢推定は重要なタスクである。複数の異なる視点での観測から、各観測時のカメラの位置姿勢（以下姿勢）を推定することで現在の状態を推定したり3次元復元などの幾何認識を実現できる。

姿勢推定の最適化問題は最適化が難しく、かつ計算量が非常に大きいことが知られている。初期値をかなりうまく取らなければ局所解に収束し、推定が失敗する。また、推定が成功しているかも保障できないことが大きな問題であった。問題の性質から最適解を求めることは不可能と考えられていた。

その姿勢推定においてRotation Averagingと呼ばれる技術が注目されている。これは複数の観測間の相対回転姿勢の推定結果が与えられた時、全観測の絶対回転姿勢を推定する問題である[1]。姿勢は回転成分と平行移動成分から成るが、そのうち回転成分だけを先に推定するというものだ。回転成分が決定されれば、平行移動成分は比較的容易に推定できる（損失を避ベば凸最適化問題として解ける）。また、2つの観測間の相対回転姿勢は、共通する特徴点から基本行列を推定し、その固有値分解を行うことで推定できる。ただRotation Averagingは自体も難しい問題と考えられていた。

2018年にオーストラリアQueensland University of TechnologyのAnders Eriksson氏らはRotation Averagingは相対回転姿勢の推定結果が一定以下のノイズしか含まれない場合は（初期値に依存せず）絶対姿勢の最適解を求めることができ、また最適性保障ができることを示した[2][3]。姿勢推定において大きな前進であり、実際提案手法は従来手法に比べはるかに正確な姿勢を与えることができた。しかし実用的には入力数が増えるにつれ計算量が急激に大きくなり、数百点程度までしか現実的な時間で求めることができず広く使われることがなかった。

それが2021年になりオーストラリアUniversity of AdelaideのÁlvaro Parra ら（前者のEriksson氏らも含む）が、これを改良し、最適解を得られかつ効率的に推定できる手法を提案した[6]。計算量は入力数に対し線形であり、入力視点数が千を増えても高速に計算でき、従来手法と比べ数十倍近く高速に解を求めることができる。現在でも入力画像数が千や万のオーダーとなると、姿勢推定に数十分や数時間かかる商用ソフトウエアも多い中、それらが数秒から数十秒で求められることになり大きなインパクトがある。これらについて説明していく。

Rotation Averaging

はじめにRotation Averagingを説明する。この問題は無向グラフ上で定義される。グラフは n 個の頂点からなり、枝 (i, j) は頂点 i と頂点 j 間の相対回転姿勢 $\hat{R}_{i,j}$ を表す。目標は全ての枝 E について $R_i\hat{R}_{ij} = R_j$ ができるだけ成り立つような絶対姿勢 $R_1, R_2, \ldots, R_n$ を推定することである。この式は R_i をさらに R_{ij} だけ回転させたら R_j になるはずという幾何制約からきている。相対回転姿勢の推定誤差を考慮し、距離関数 d を使って、次のような最適化問題を解くことで絶対姿勢を推定するタスクをRotation Averagingと呼ぶ。

$$\arg \min_{R_1,\ldots,R_n \in SO(3)} \sum_{(i,j)\in E} d(R_i R_{ij}, R_j) \quad (1)$$

ここで $SO(3)$ は回転群を表し、サイズが 3×3 の回転行列、つまり直交行列かつ行列式が1であるような行列の集合で表される（一意性を保障するため鏡像反転に対応する行列式が -1 であるような行列は除く）。

オリジナルの問題は複数の回転姿勢の推定 $\{R_1, \ldots, R_n\}$ が与えられた時、距離関数 d 上での平均を求めるものであったためRotation Averagingという名がつけられていた。

$$\arg \min_{R \in SO(3)} \sum_{i=1}^{n} d(R_i, R)$$

そして、今回扱う問題は複数の回転（絶対回転と相対回転の積）の平均を同時に求めるような問題とみなすことができる。このRotation Averagingは簡単そうに見えるが回転行列 $SO(3)$ が凸集合でないことから全体は非凸最適化問題であり最適化が難しい。

Rotation Averagingの強双対性

Eriksson氏らは[2-3]、このRotation Averagingの双対問題はある条件下では強双対性を持ち、双対問題（実際は双対問題の双対問題）を解くことで最適解を得られることを示した。

まず、距離として双対問題が扱いやすくなるコーダル距離 $d(A,B) := ||A-B||_F^2$ を利用する。コーダル距離は R、S がそれぞれ回転行列である場合、$d(R,S) = 4(1-\cos\alpha)$、ただし α は R と S の相対角度であり、相対角度が小さいほど0に近づくような距離関数である。

次に式(1)を次のように整理する。コーダル距離は行列積のTrace（対角和）で表すことができる。

$$||R_i\tilde{R}_{ij} - R_j||_F^2 = \mathrm{Tr}(R_i\tilde{R}_{ij}R_j^T)$$

また、枝 ij が存在する時 $a_{ij}=1$、そうでない時は $a_{ij}=0$ と定め、ij ブロック目を $a_{ij}\tilde{R}_{ij}$ としたサイズが $3n\times 3n$ の行列 $\tilde{R}$ を用意する。また n 個の回転行列を並べたサイズ $3\times n$ の行列 $R=[R_1R_2\ldots R_n]$ を用意する。

最後に、回転行列が行列式が1を持つという制約を除く（集合は $O(3)^n$）。この場合、直交行列が行列式が1を持つ集合と -1 を持つ集合と2つの集合に分割され、元の問題の極小解は新しい問題の極小解になっている保障がある。解を得た後に行列式の値が -1 であれば反転させれば良い。これらによって、式(1)は次のように整理される。これを主問題とする。

$$\min -\mathrm{tr}(R\tilde{R}R^T)$$
$$s.t.\quad R\in O(3)^n$$

次に、この問題のラグランジュ未定乗数法を考える。

$$L(R,\Lambda) = -\mathrm{tr}(R\tilde{R}R^T) - \mathrm{tr}(\Lambda(I-R^TR))$$
$$= \mathrm{tr}(R(\Lambda-\tilde{R})R^T) - \mathrm{tr}(\Lambda) \quad (2)$$

ここで、Λ は制約 $R\in O(3)^n$ に対応するラグランジュ乗数であり 3×3 のブロック行列が対角に並んでいるサイズが $3n\times 3n$ のブロック対角行列である。この極値条件を調べるため、$L(R,\Lambda)$ を R について微分をとり0となる条件を求めると

$$(\Lambda^*-\tilde{R})R^{*T} = 0 \quad (3)$$

が得られる。この式を変形し、$\tilde{R}$ の定義も踏まえると、最適値 Λ^* は次のような形をとる。

$$\Lambda_i^* = \sum_{j\in\mathcal{M}} a_{ij}\tilde{R}_{ij}R_j^{*T}R_i^*$$

このラグランジュ未定乗数法を元に次の双対問題を考える。

$$\max_{\Lambda}\min_{R} L(R,\Lambda)$$

$\min_R L(R,\Lambda)$ は(2)から、$\Lambda-\tilde{R}\geq 0$ の時、$-\mathrm{tr}(\Lambda)$、そうで無い場合は $-\infty$ をとる（$A\geq 0$ は A が半正定値行列であると定義）。ここで、$\Lambda^*(I-R^TR)=0$ であることと、(3)から

$$-\mathrm{tr}(\Lambda^*) = -\mathrm{tr}(\Lambda^*R^{*T}R^*) = -\mathrm{tr}(R^*\tilde{R}R^{*T})$$

が成り立つ。つまり、$\Lambda^*-R\geq 0$ が成り立つ時、双対問題と主問題の解にギャップが無い強双対性が成り立ち、双対問題の最適解が主問題の最適解を与えることが分かる。

この $\Lambda^*-R\geq 0$ が成り立つ条件も解析的に求めることができ、$\tilde{R}$ が真の値と大きくずれていない場合に達成される[2-3]。例えばグラフが完全グラフであれば、それぞれが42.9度以下の誤差であれば成り立つことが示される。

この条件は観測ノイズとしては十分許容できるようにみえるが、実際は相対回転姿勢は別の推定手法（対応点検出など）を使って推定するため大きく間違えてしまう場合も多い。この保障については最後に述べる。

双対問題をどのように解くのか

この双対問題は主問題とは違って任意の初期値から最適解が得られる保障がある半正定値凸最適化問題であり、内点法などを使って解けるが入力点数に対しスケールしない。そこで、再度、この双対問題を最適化問題としてみた場合の

双対問題を考え、その上で最適化問題解くことを考える。

$$\min_{Y} -\mathrm{tr}(\tilde{R}Y) \quad (4)$$

$$s.t. Y_{ii} = I_3, Y \geq 0$$

ここで行列 Y は $3n \times 3n$ の行列であり、最適解は主問題の最適解と一致することが示せる。この問題は小さいブロック毎に半正定値計画問題を解析的に解くブロック座標降下法を使って解くことができる。内点法より速いものの、計算量は入力数の2乗に比例してしまう問題があった。

回転座標降下法[1]はこの最適化変数を $Y = QQ^T$ と分解された形で表現した上で (4) を解く。Y は最適値のみ $Y^* = Q^*Q^{*T}$ という行列分解が存在でき、他は分解できる保障がないが、最適化途中では常にこのような分解が可能であることを示し、Q 上で最適化を行う。さらに、最適化の途中で Q の行列式が -1 であれば 1 に変換することで現在の回転候補として使うことも可能であり、局所的な情報を使った最適化などを行うことも可能である。このように更新することで、計算量を入力に対して線形に減らし、数十倍という劇的な高速化を達成することができる。

検証可能な幾何認識

近年このRotation Averaging以外にも、幾何認識タスクで認識が成功しているかどうかを検証できる手法が多く登場してきている（例えば文献[3]）。これによってSLAMやSfMなどの復元に使え、ミッションクリティカルなタスクで認識が失敗していることを検知できるようになってくる。今後、高速、高精度かつ高信頼な認識技術が発展してくると考えられる。

1) R. Hartley,"Rotation Averaging," International Journal of Computer Vision, vol.103, no.3, pp.267-305, 2013.
2) A. Eriksson et al.,"Rotation Averaging and Strong Duality," CVPR 2018.
3) A. Eriksson et al.,"Rotation Averaging with the Chordal Distance: Global Minimizers and Strong Duality," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.43, Issue 1, pp.256-268.
4) A. Parra et al.,"Rotation Coordinate Descent for Fast Globally Optimal Rotation Averaging," CVPR 2021.
5) H. Yang et al.,"One ring to rule them all: Certifiably robust geometric perception with outliers," NeurIPS 2020.

（2021年6月号掲載の記事に加筆）

DROID-SLAM：逐次的な修正で環境に対応する

SLAMは移動体がオンラインでセンシングを行いながら自己位置推定および環境地図の作成を行うタスクであり、ロボットやドローン、自動車の自動運転などで重要なタスクである。センサとしてLIDARや車輪のエンコーダ（オドメトリ）を使う場合が多いが、カメラを利用したVisual SLAMも増えつつある。

このSLAMタスクにおいて2021年8月に発表されたDROID-SLAM[1]は従来のVisual SLAMの精度、成功確率（通常のSLAMは自己位置を見失うなど失敗する場合も多い）を大きく改善した。具体的にはECCV 2020 SLAM Competitionの優勝チームと比べて自己位置推定精度で62％誤差を小さくしたほか、EuRoC、TUM-RGBD、ETH3D-SLAMなど、他のデータセットでも従来手法の誤差を60～80％近く改善している。また成功確率もETH3D-SLAMで従来手法が19/32回成功だったのに対し、DROID-SLAMは30/32回成功した。

DROID-SLAMの主著者であるPrinceton UniversityのZachary Teed氏はこれまでVision SLAMの一つであるDeepV2D[2]、オプティカルフロー（動画の隣接フレーム間での対応画素間の移動量を推定するタスク）の最高精度を達成したRAFT[3]、3次元変換群に対する微分[4]などの研究を発表しており、今回のDROID-SLAMはこれらの研究の集大成といえる。RAFTは画像認識のトップ学会の1つであるECCV 2020でベストペーパーを受賞している。DROID-SLAMがどのように問題を解いているのか、なぜ大きく性能を改善できたのかについて解説する。

従来のSLAM手法との比較

SLAMは従来、確率的モデルで定式化され、カメラ姿勢推定と環境復元を交互に最適化することで実現されていた。近年ではバンドル調整（Bundle Adjustment：BA）と呼ばれるエネルギー最小化のような最適化問題を定義し、カメラ姿勢と3次元環境の地図推定を同時に推定するアプローチが主流となっていた。この定式化は複数の異なるセンサを組み合わせられる利点がある。

これに対し、DROID-SLAMは教師あり学習を使い、画像群を入力としカメラ姿勢と各画像の深度として表される環境地図を出力するよう学習する。SLAM問題を直接学習することは学習自体が難しかったり、汎化しない問題があった。DROID-SLAMはこの問題に対し、バンドル調整を内部で利用することで問題が持つ物理制約をモデルに取り込み、またRAFTで使っていた微分可能な再帰的最適化（Differentiable Recurrent Optimization-Inspired Design：DROID）を利用することで、環境に対する適応能力を大幅に改善した。最適化方法を学習しているともいえる。

DROID-SLAMの手法詳細

それではDROID-SLAMの具体的な手法について説明しよう。

入力として画像集合 $\{I_t\}_{t=0}^{N}$ が与えられたとする。各画像 I_t はカメラ姿勢 $\mathbf{G}_t \in SE(3)$（回転と並進）と画素ごとの深度の逆数 $\mathbf{d}_t \in R_+^{H \times W}$ を状態として持つ。これらの状態がモデルの推定対象である。またフレームグラフ $(\mathcal{V}, \mathcal{E})$ を考え、各頂点は画像であり、枝 $(i, j) \in \mathcal{E}$ は画像ペア I_i, I_j 間に共通して見えている箇所がある場合を表すとする。状態が更新されるたびに、共通して見ている箇所も変わるので、フレームグラフもそれに応じて更新される。

始めに、現在のカメラ姿勢 $\mathbf{G}$ と深度 $\mathbf{d}$ を使って、ある画像 I_i の画素が別の画像 I_j のどの画素に対応するのかを求める。画像 I_i 上の画素グリッドの座標を並べたベクトルを $\mathbf{p}_i \in R^{H \times W \times 2}$ とする（例えば $(0,0), (0,1) ..., (H-1, W-1)$ のように）。この時、I_j 上での対応する画素の座標は次のように求められる。

$$\mathbf{p}_{ij} = \Pi_c(\mathbf{G}_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}_i))$$

ここで Π_c はカメラモデルに基づき3次元中の点を画像中の位置へ変換する操作であり、Π_c^{-1} は逆に画像中の位置から、フレーム座標の3次元中の位置に変換する操作である。また $\mathbf{G}_{ij} = \mathbf{G}_j \circ \mathbf{G}_i^{-1}$ は画像 I_i のフレーム座標を世界座標に戻した

上で画像I_jのフレーム座標に変換する操作である。

次に画像由来の情報でこれらの対応を補正することを考える。画像I_i, I_j間の対応関係を求める場合、まずそれぞれの画像をCNN（gで表す）を使って1/8の解像度にした特徴マップ$g(I_i), g(I_j)$に変換し、次に画像ペアの全画素間の対応を表す4D特徴ボリュームC^{ij}を次のように求める。

$$C^{ij}_{u_1v_1u_2v_2} = \langle g(I_i)_{u_1v_1}, g(I_j)_{u_2v_2} \rangle$$

この4D特徴ボリュームの各値は特徴マップの内積で得られる。次にこの4D特徴ボリュームの片方の画像の解像度だけをダウンサンプリングで減らした特徴ピラミッドを作る。この4D特徴ボリュームは画素間にどのような対応があるのか、補正すべきかといった情報を表している。

次に$\mathbf{p}_{ij}$を使って4D特徴ボリュームの対応する特徴を読み込み、それを使ってConvGRU（convolutional gated recurrent unit）の内部状態$\mathbf{h}_{ij}$を更新する。そして、更新された状態から対応の補正量$\mathbf{r}_{ij} \in R^{H \times W \times 2}$と確信度$\mathbf{w}_{ij} \in R^{H \times W \times 2}$を出力し、補正した対応を$\mathbf{p}^*_{ij}$とする。

$$\mathbf{p}^*_{ij} = \mathbf{r}_{ij} + \mathbf{p}_{ij}$$

現在の対応関係を画像由来の対応を使って修正する操作とみなせる。そして補正された対応および確信度を使ってバンドル調整を行う。バンドル調整は次の目的関数の最小化問題を解くことで達成される。

$$\mathbf{E}(\mathbf{G}', \mathbf{d}') = \sum_{(i,j)\in\mathcal{E}} ||\mathbf{p}^*_{ij} - \Pi_c(\mathbf{G}'_{ij} \circ \Pi_c^{-1}(\mathbf{p}_i, \mathbf{d}'_i))||^2_{\Sigma_{ij}}$$

ただし$||\cdot||_\Sigma$はマハラノビス距離であり、$\Sigma_{ij} = \mathrm{diag}\mathbf{w}_{ij}$である。先程の位置を推定している場合と似ているが、今回は入力が対応$\mathbf{p}^*_{ij}$と確信度Σ_{ij}であり、最適化対象の変数がカメラ姿勢$\mathbf{G}'$と深度$\mathbf{d}'$である。

この最適化は線形化した上で、ガウス・ニュートン法による解を解析的に求められる。これは、$\mathbf{p}^*_{ij}$を入力とし、最適化された$\mathbf{G}'$、$\mathbf{d}'$が出力される微分可能な層とみなすことができる。そして、更新されたカメラ姿勢と深度を再度入力として、上の操作を同じパラメータを共有したネットワークに再度通し、更新することを繰り返す。このように、DROID-SLAMは画素間の対応関係、姿勢、深度を更新していく操作を繰り返していく。これら全体が1つのネットワークとなっている。

このようにして作られたネットワークを使ったSLAMシステムはフロントエンドとバックエンドから構成される。フロントエンドは画像を受け取り、特徴を抽出し、キーフレームを選び、局所的にカメラ姿勢と地図を更新する。バックエンドは定期的に全画像でカメラ姿勢と地図を更新する。4D特徴ボリュームは大きいのでキャッシュせず利用時に毎回計算する。

CGを教師データに使って学習した

学習にはTartanAir[5]データセットを利用した。TartanAirはUnreal Engineを使って作られたCGデータセットであり、屋内外の30セットの環境から成る1037シーケンスから構成される。各シーケンスは500から4000画像で構成され、データ全体では100万超の画像から成る。これは従来利用されていたデータセットと比べて数十倍大きい。CGなので画素間の対応関係や姿勢の正解が得られることに注意されたい。

学習は姿勢損失と対応損失を使った教師あり学習で行った。姿勢損失は推定した姿勢と真の姿勢との誤差とし、対応損失は真の深度と姿勢によって求められた対応と、推定された対応間の誤差とした。これらの誤差は再帰的に更新をしている全てのステップで与えられるとした。

DROID-SLAMはこのように全てのステップでその途中の推定結果を元にそれを改善するようにして学習し、このステップを繰り返していけば真の解に収束するように学習する。このように学習されたDROID-SLAMは、最初に述べたように多くのデータセットやタスクで従来手法を大きく凌駕する精度と頑健性を達成できた。また処理速度も従来のVisual SLAMより速く、2台のGPUを使えばリアルタイムで処理することもできる。

なぜ大きな改善を達成したのか

DROID-SLAMが大きな改善を達成した理由としては、TartanAirデータセットという非常に大きなデータセットを利用して教師あり学習をしたことにあり、特に画素間対応という大量の教師信号を使ってモデルの精度や頑健性を大幅に改善できたとみられる。

一方で従来はCGを使った学習は大きく成功していなかった。シミュレーションと現実世界間のギャップが大きく、CGで学習したモデルを現実世界の問題に使えなかった。しかしDROID-SLAMはCGのみで学習し、実世界データでのfine-

tuningをしていないにもかかわらず多様な現実環境に汎化できており、このギャップを克服している。

ギャップを克服できた理由として、再帰的に修正が入る仕組みが推論時に環境に適応する役割を果たしているのではないかと考えられる。例えば現実世界のデータを入力とした場合、最初のイテレーションでは大きく推定がずれたとしても、その時の誤差を基に後続のイテレーションで修正できる。

また、ConvGRUの状態が入力データの特徴を理解し挙動を変えているとみなせる。このような状態を持ったネットワークが環境に適応する例は、シミュレーションで学習したロボットがルービックキューブを解く例でも報告がされていた[6]。人間もツルツルとした氷の上を歩く場合は1歩目、2歩目は滑って思い通りに歩けないが、すぐに最初の誤差を基に自分の挙動を調整し、それ以降は歩くことができる。

このような入力や誤差を基にした状態を使って自分の挙動を逐次的かつ再帰的に修正が入る仕組みは、他のタスクにおいても高い汎化性能を達成する上で有効だと思われる。

---

1) Z. Teed et al., "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras," https://arxiv.org/abs/2108.10869
2) Z. Teed et al., "DeepV2D: Video to Depth with Differentiable Structure from Motion," ICLR 2020.
3) Z. Teed et al., "RAFT: Recurrent All-Pairs Field Transforms for Optical Flow," ECCV2020.
4) Z. Teed et al., "Tangent Space Backpropagation for 3D Transformation Groups," CVPR 2021.
5) W. Wang et al., "TartanAir: A Dataset to Push the Limits of Visual SLAM," IROS 2020.
6) OpenAI, "Solving Rubik's Cube with a Robot Hand," https://arxiv.org/abs/1910.07113

（2021年12月号掲載の記事に加筆）

10-8

Neural Descriptor Fields：少数教師からの学習を可能とする物体や3次元環境の同変表現

ロボットにタスクを指示する際、人がロボットを動かして、実際にそのタスクをデモンストレーションし、それを繰り返すというアプローチは直感的かつ強力である。しかし、このアプローチは汎化という面で強い制限がある。

例えば、グリッパー付きのロボットアームを使ってマグカップを収納ラックに立てかけるようなタスクを考え、デモンストレーションをしたと考える。

この場合、デモンストレーションした時と少しでも異なる環境、例えば様々な形状のマグカップ、収納ラック、様々な初期状態に対応することは容易ではない。マグカップが少し背が低く、取っ手が大きくなっている場合はどうか。マグカップが斜めに配置されている場合はどうかといったようにである。

このような問題に対し、Neural Descriptor Fields（NDF、ニューラル記述子場）を利用することで、数例のデモンストレーションからでも環境が異なる場合に対応できることが示された[1]。NDFは例えば10回の試行のみで、様々な形状のマグカップ、初期状態に対し、90%の成功率でタスクを成功させることができた。従来の強化学習や模倣学習と比べて劇的に少ない試行で学習することができる。今回はこのNDF、およびその実現に貢献しているVector Neuron[2]について解説する。

NDF環境中の位置や姿勢の記述子場

NDFは対象タスク中における各位置や各姿勢を記述子（ベクトル）に変換して表すネットワークであり、位置や姿勢の場を表現する。

この記述子は、形状が多少異なっている場合や、座標軸が変わったとしても対応するような位置/姿勢に対し同じような記述子を与えるように設計されている。これにより、異なる環境間での位置や姿勢の対応関係を求められるとともに、参照位置/姿勢に対して、現状の位置/姿勢がどの程度異なっているのかを表すエネルギー場を自然に定義することができ、このエネルギー場上での最適化を行うことで参照位置/姿勢までの軌道を求めることができる。

Neural Point Descriptor

始めに、位置（点）に対する記述子を与えるNeural Point Descriptor Fieldsを紹介する。Neural Point Descriptor Fieldsは関数で表現される。この関数 $f(\mathbf{x}|\mathbf{P})$ は環境から得られた点群 $\mathbf{P}$ で条件付けされ、3次元座標入力 $\mathbf{x} \in \mathbb{R}^3$ を与えると、その記述子を表すベクトル $\mathbf{d} \in \mathbb{R}^k$ を返す。この記述子 $\mathbf{d}$ は点群 $\mathbf{P}$ が表す環境における位置 $\mathbf{x}$ の空間的な意味を符号化しており、例えばマグカップがある位置/姿勢における点群の場合、各点について、それが取っ手に近い、縁である、内部での右側であるといったような情報を符号化する。

NDFは記述子を与える関数を、物体形状を表すOccupancy Networkと呼ばれる関数を利用して表現する。Occupancy Networksは点群と入力座標が与えられた時、その入力座標が点群として観測されている物体の内部であるか外部であるかの2値を分類するようなネットワークである。この分類境界面は物体表面を表す。このOccupancy Networkを学習するためには点群データと物体の内外判定情報のみ必要であり、対応関係の教師ラベルなどが必要ないことに注意してほしい。

このOccupancy Networkは点群を潜在表現に変換する符号化器と、潜在表現と入力座標がデータと与えられ、2値分類を行う分類器で構成される。この分類器の各層は物体表面を段階的に細かく分類するのに対応し、その活性値は物体の特徴を表している。そこで、Occupancy Networkの分類器の各層の活性値を結合して得られたベクトルを記述子として利用する。この記述子は先程述べた似た意味の位置に対し似た記述子を与えるという特徴を達成している。

また、この記述子は取っている座標軸が変わったとしても同じであってほしい。これを式で表すと任意のSE（3）変換 $(\mathbf{R}, \mathbf{t}) \in \mathbf{SE}(3)$（$\mathbf{R}$ は回転行列、$\mathbf{t}$ は並進移動ベクトル）に対し、次が成り立つことが必要である。

$$f(\mathbf{x}|\mathbf{P}) \equiv f(\mathbf{Rx} + \mathbf{t}|\mathbf{RP} + \mathbf{t})$$

これを達成するため f はSE（3）変換に対し同変であるよう

にする。並進移動については点群と入力座標を点群の中心を引いた形にすれば達成でき、回転同変については後述するVector Neuronを使って達成できる。

Neural Pose Descriptor

次に方向（姿勢）についての記述子であるNeural Pose Descriptorを説明する。操作タスクにおいては位置だけでなく方向（姿勢）も重要となる。

例えばグリッパーを使ってマグの取っ手を掴む際は、グリッパーのマグに対する角度が重要となる。この姿勢は並進と方向からなる6次元の自由度を持ち、これを指定できる必要がある。Neural Pose Descriptor Fieldsは固定のクエリ点群を用意し、各クエリ点の記述子を結合したベクトルを、姿勢を表す記述として利用する。

エネルギー最小化による軌道生成

Neural Point Descriptorでは現在の入力点群 $\mathbf{P}$ と位置 $\mathbf{x}$、参照点群 $\hat{\mathbf{P}}$ と参照位置 $\hat{\mathbf{x}}$ が与えられた時、現在の点群における位置がどれだけ参照点群における参照位置と離れているかは、次のエネルギー場で与えられる。

$$E(\mathbf{x}|\hat{\mathbf{P}}, \mathbf{P}, \hat{\mathbf{x}}) = ||f(\hat{\mathbf{x}}|\hat{\mathbf{P}}) - f(\mathbf{x}|\mathbf{P})||$$

これは入力と参照が対応している場合、それらの記述子が似ている場合はエネルギーは小さく、そうでなければエネルギーは高くなることを利用している。

このエネルギー場上で、対応する点 $\hat{\mathbf{x}}$ は最小化問題を解くことで求められる。

$$\bar{\mathbf{x}} = \arg\min_{\mathbf{x}} E(\mathbf{x}|\hat{\mathbf{P}}, \mathbf{P}, \hat{\mathbf{x}})$$

よって現在の点から対応点まではエネルギー上で最適化し、最小化するような点を探すことで達成できる。これと同様に姿勢についても姿勢 $\mathbf{T}$ についてのエネルギー場を考え、その最小化問題を解くことで姿勢を求めることができる。

このエネルギー関数を使い、デモンストレーション中の接触位置や姿勢を参照点/姿勢とし、エネルギー最小化問題を解くことで、デモンストレーションを模倣することができる。冒頭にあげたように、このアプローチによって非常に少ない試行回数で環境が異なる場合でも高い成功確率を達成することができている。

Vector Neurons SE(3)同変なNN

このNDFは入力に対するSE(3)（並進移動と回転）変換について同変となっていることが重要である。学習中にはなかったような姿勢にも対応でき、高い汎化性能を達成する。この実現に貢献しているのがVector Neuronである。

Vector NeuronはSE(3)のうち回転同変を達成するようなニューラルネットワークの設計方法である。これまでもSE(3)同変を達成するニューラルネットワークは多く提案されていたが、いずれもネットワーク設計の制約が大きかったり、専門知識を必要とした。これに対しVector Neuronは汎用的なネットワークであり、専門知識を必要とせず単純である。これについて以下で説明する。

通常のニューラルネットワークは内部状態の各ニューロンはスカラー値 $z \in \mathbb{R}$ を持ち、チャンネル数が C の層の活性値は $[z_1, z_2, \ldots, z_C] \in \mathbb{R}^C$ という C 次元ベクトルで表される。これに対しVector Neuronは各ニューロンを3次元のベクトル $\mathbf{v}$ で表す。そしてチャンネル数が C の時の活性値を $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_C] \in \mathbb{R}^{C\times 3}$ という行列で表し、これらが N 個集まった集合（点群など）からなる入力の活性値は $\mathcal{V} = \{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_N\} \in \mathbb{R}^{N\times C\times 3}$ のようなテンソルで表す。関数 f は入力が $\mathcal{V}$、任意の回転操作 $\mathbf{R} \in \mathbf{SO}(3)$ に対し、$f(\mathcal{V}R;\theta) = f(\mathcal{V};\theta)R$ を満たせば回転同変である。これを満たせるように結合層、活性化関数を設計していく。

始めに結合層であるが、重み行列 $\mathbf{W} \in \mathbb{R}^{C'\times C}$ を使って $\mathbf{V}' = f_{lin}(\mathbf{V};\mathbf{W}) = \mathbf{W}\mathbf{V} \in \mathbb{R}^{C'\times 3}$ と定義する。このとき、任意の回転行列 R に対し

$$f_{lin}(\mathbf{V}R;\mathbf{W}) = \mathbf{W}\mathbf{V}R = f_{lin}(\mathbf{V};\mathbf{W})R = \mathbf{V}'R$$

を満たすため、この結合層は回転同変である。

次に活性化関数であるが、通常のReLUは固定の境界面を使って非線形変換を行うので回転同変ではない。回転同変にするためには、入力を回転させた時、境界面も同様に回転する必要がある。そこで、固定の境界面でなく、入力から回転同変な変換を使って境界面を決定し、その境界面を使って非線形変換を行う。プーリング層もこれと同じアプローチで実現できる。

最後に、同変から不変にする方法だが、回転同変な特徴量 $\mathbf{V} \in \mathbb{R}^{C\times 3}$、と他の回転同変な特徴量 $\mathbf{T} \in \mathbb{R}^{C'\times 3}$ の転置 $\mathbf{T}^T$ 間の積は回転不変になることを利用する。

$$(\mathbf{V}R)(\mathbf{T}R)^T = \mathbf{V}RR^T\mathbf{T}^T = \mathbf{V}\mathbf{T}^T$$

今の特徴量との転置との積 $\mathbf{V}\mathbf{V}^T$ でも不変となるが、次元数が $C \times C$ と大きくなるため、代わりにそのチャンネル数を少なくした特徴量の転置との積を計算して不変量を得る。

NDFは3次元でのタスク教示を大きく変えうる

Neural Descriptor Fieldsは3次元空間で操作を伴うようなタスクにおいて、対応関係を教師なしで求められ、形状が違っても汎化できること、また、そこでSE (3) 同変な記述子を用いることで、初期状態や姿勢の違いについても汎化できることを示した。これにより劇的に少ないデモンストレーション数でタスクを指示できることを示した。

また今回、入力として点群を利用しているが、近年進展している画像からの3次元復元を用いれば、今後、通常のカメラや単眼カメラを用いても同様の手法を実現できる可能性がある。

---

1) A. Simeonov et al.,"Neural Descriptor Fields:SE (3) -Equivariant Object Representations for Manipulation," https://arxiv.org/abs/2112.05124

2) C. Deng et al.,"Vector Neurons: A General Framework for SO (3) -Equivariant Networks," https://arxiv.org/abs/2104.12229

(2022年2月号掲載の記事に加筆)

第11章

言語

11-1 seq2seq：文から文を生成するニューラルネットワーク

自然言語処理分野において、この数年で大きく進展したのが文の認識と生成である。この中心的な役割を果たしたのがseq2seq（Sequence To Sequence）と呼ばれる系列モデル[1]だ。

文はサイズが可変である一方、機械学習が使うモデルのサイズは固定である。このため、文を機械学習のモデルで扱うには、何らかの形で文を機械が扱える固定長の表現に変換する必要がある。

従来手法では文を表現するには、Bag of Wordsと呼ばれる、文に含まれる単語の集合による表現（文書中の単語の位置は無視し、文書中にある単語が出現していたら、その単語に対応する次元を1、それ以外を0にしたような表現）か、または文の意味を解析し、述語項解析をした上でその主語、述語、目的語などの3つの組を求め、これを使うといった方法が採られていた。さらにこの問題は、機械翻訳のように入力も文、出力も文であるような問題の場合、より複雑化する。入出力ともに可変長のサイズの問題を、機械学習でどのように扱うかは長年の課題であり、従来はアドホックな方法が使われていた。

可変長の文を扱いやすくしたseq2seq

この問題を解決したのがseq2seqである。seq2seqは文から文を生成するモデルである。はじめに、入力文を単語ごとに順に読み込み、固定長の内部表現に変換し、またその固定長の内部表現から可変長の文を単語ごとに順に生成する。seq2seqは入力系列を固定長に収めるという、かなり強い制約のもとで学習を行う。

文$X=x_1, x_2 \cdots x_n$から、文$Y=y_1, y_2 \cdots y_m$を推定する問題を考えてみよう。seq2seqはRNNを利用する。はじめに符号化器f_{e}は単語を1つずつ読み込み、内部表現を順に更新していく。つまり、i番目の内部状態をs_i、入力単語をx_iとした時、$i+1$番目の内部状態は、

$$s_{i+1}=f_{\mathrm{e}}(s_i, x_i)$$

と求められる。そして、文の終端を表すEOSという単語を受け取ったら、次に復号化器f_{d}は単語を1つずつ生成する。この際、直前で何の単語を生成したかの情報は加えるようにする。

$$(y_i, s_{i+1})=f_{\mathrm{d}}(s_i, y_{i-1})$$

ここで、y_iは直前に出力した単語であり、y_0＝EOSである。復号化がEOSを生成したら、文の生成は終了したとみなせる。符号化器f_{e}、f_{d}はともに異なるモデルを使ってもよい。また、この場合は1層のモデルだが、内部状態の隠れ層を複数階層、用意することもできる。

このseq2seqは、EOSを受け取った瞬間の内部状態s_nに、入力Xの情報が全て詰め込まれており、入力Xの情報を固定長の内部表現に符号化したと考えられる。そして、その情報から出力Yを順に生成できるようにf_{e}、f_{d}を学習する。

復号化時に各単語を生成する際に、y_iはsoftmaxなどを使い、単語集合に対する確率分布を生成するようにし、各出力の単語を予測できるように学習した場合、

$$\Sigma_i \log p\left(y_i | s_i, y_{i-1}\right)$$
$$=\Sigma_i \log p\left(y_i | X, y_1, \cdots y_{i-1}\right)$$

である。このため、seq2seqはYのXでの条件付き確率$p(Y|X)$を学習していることに相当する。可変長の入出力を扱っているが、モデルf_{e}、f_{d}は固定長のサイズである。

このRNNでは長期の依存関係が重要になるので、LSTM（Long Short Term Memory）などゲート付きのユニットを利用する。こうすることで、情報を長期的に記憶し、学習の際、遠くの層までエラー（目的関数の勾配）を減衰せずに伝播させることができる。

seq2seqは機械翻訳、構文解析、画像からのキャプション生成、対話モデルなど広い問題に利用され、大きな成功を収めた。自然言語処理では文の構造は木で表す考えが主流だった。しかし、seq2seqのように内部状態に入力を畳み込んでいく方法でも、ゲートがあれば長い距離の単語間の関係が扱える。seq2seqは構造が単純ながら様々なタスクで木構造を使った手法と同程度かそれ以上の精度を達成できるた

め、最近では「木構造は本当に必要なのか」という議論もされている。

自己符号化器としてのseq2seq

入力文と出力文が同じ場合は、seq2seqは系列に対する自己符号化器（AutoEncoder）として考えることができる。自己符号化器が事前学習に有効であったのと同様に、seq2seqも系列データに対する事前学習として使うことができる[2]。

RNNは従来、学習が難しいことが知られていた。ハイパーパラメータの組み合わせ次第で学習が全く進まなかったり、発散したりしてしまうことがあり、うまく学習ができるのは非常に狭い領域だけである。そのため、RNNの学習を安定化させるような研究が進められていた。

seq2seqを利用した事前学習は、このRNNの学習を安定化させるのに役立つ。正しく文を要約できるような符号化と表現方法を獲得できているため、これを使ってより詳細なタスクに調整することができるからだ。この場合、入力と出力は同じ文を利用し、文を読み込み、その文を生成できるように学習する。

従来の自己符号化器と同様に、内部表現は表現力（次元数）が抑えられているため、seq2seqでも文をそのまま記憶することはできず、文を要約して圧縮した形で保存しなければならない。そして、符号化器は文から良い内部表現を作れるように、復号化器は内部表現から文を作れるように学習をする。また、単語単位のドロップアウトも有効であることが報告されている。文の中から多少単語が抜け落ちたとしても、正しく必要な情報が抽出できるように学習される。もともと自然言語はかなり冗長にできているため、このようなことをしても学習できる。

このように学習したRNNを初期値として利用し、その後、文書分類や意味解析など特定のタスクを学習するようにすることで、安定して学習でき、精度も従来手法と比べて高くなることが報告されている。

1) I. Sutskever, et al.,"Sequence to Sequence Learning with Neural Networks," NIPS 2014.
2) A. Dai, et al.,"Semi-supervised Sequence Learning", http://arxiv.org/abs/1511.01432

（2016年1月号掲載の記事に加筆）

11-2 言語の創発：機械はどのようにコミュニケーションできるのか

現在、機械同士のコミュニケーションは人が設計したプロトコルに従って実現されている。そして、タスク、コンテキストに応じて情報をどのように表現し、それを送り手、受け手がどのように処理するかは人が仕様を決め、プログラムなどで実現されている。

一方、人同士は主に自然言語を用いて多様かつ柔軟なコミュニケーションを実現できる。自然言語の表現力は高く、ありとあらゆる現象、情報、知識を表現することができ、それを相手に伝えることができる。

例えば、昨日の野球の試合結果を自然言語では「3-0でチームAが勝利した。負けたチームBは、先発投手は安定していたが中継ぎが崩壊した。チームBにとって唯一のチャンスであった8回裏はサインプレーのミスでダブルプレーに終わり、新加入Cの連携不足が露呈した。」と表すことができる。この文章を読めば、実際に起こった野球の場面を想像することができる。この情報を他の手段（例えばJSONやXML、RDFなど）で表現することは困難である。自然言語が人の情報の表現手法、コミュニケーション方法の"デファクトスタンダード"になっている。

機械ならではのコミュニケーション

しかし、自然言語はあくまで人という動物が進化の過程で獲得した一手段である。人は自然言語を使って考え、情報を表現し伝えているので、あたかも情報の表現方法が自然言語しか存在しないように考えてしまうが、自然言語以外にも情報の表現方法やコミュニケーション方法があってもおかしくないだろう。特に機械、そして有線・無線ネットワークは人間とは異なる特性をもつので、現在の自然言語とは異なる機械に適した情報の表現方法、コミュニケーション方法があってもよい。

それでは自然言語によるコミュニケーションに匹敵、またはそれを超えるようなコミュニケーション方法を機械はどのように作ることができるだろうか。

機械が新しい言語を生み出すということは古くから試みられている[1]が、ここでは機械同士が協調してタスクを解く過程で文字、単語や文に相当するような新しい単語を生成する例をみてみよう[2]。

情報の送り手は自分が持っている画像に関する情報をメッセージで送り、受け手はメッセージを受けとって、たくさんある画像の中からどれを指しているのか推定するゲームを解くことを考えてみる。送り手側は受け手が対象画像を持っていることは知っているが、その他にどのような画像集合をもっているのかを知ることはできない。そして、このメッセージは自然言語のような離散記号の可変長系列から成るとする。

送り手、受け手ともにRNNを使い、ユニットにはLSTMを利用する。送り手は入力画像を基に可変長の離散記号からなるメッセージを1つずつ生成し、メッセージの終わりを示す記号を生成して終了する。受け手はメッセージを1つずつ終わりまで受信した上で、どの画像を指しているのかを推定する。受け手の最終状態から画像マッチング用のベクトルを計算し、このベクトルとの内積が最も大きい画像を推定結果とする。学習の目的関数には、ヒンジ損失関数を使い、正解の画像のスコアが不正解の画像のスコアより大きくなるように学習する。

キリンやクマなど意味を持つ離散記号を獲得

学習は、この送り手側の操作（符号化）と受け手側の操作（復号化）をつなげて全体の過程を1つの計算グラフとみなして行う。誤差逆伝播法で符号化、復号化のモデルを学習する。しかし、この計算グラフには離散記号を決定する部分で微分不可能な離散化ステップを含むために、直接、誤差逆伝播法を使って学習することができない。

こうした離散分布を含む場合、従来は尤度比法を使って勾配を推定していたが、その場合、分散が大きすぎて現実的に学習することができなかった。この研究では、Gumbel-Softmax Trickという手法を使う。これは、順計算時は離散化した上で計算し、逆計算時にはSoftmax分布を決定的な関数とGumbel分布からのノイズの組み合わせで構成して学習する手法である。順計算と逆計算で異なる計算グラフを使うため勾配推定にはバイアスが含まれるが、分散が低く離

散変数の種類数が大きくても学習することができる。彼らの実験では離散記号の種類は10000種だった。

彼らの実験では、受け手が127枚の候補画像から1枚を推定するタスクにおいて95%の精度で当てられるように学習することができた。

そして、獲得されたメッセージの離散記号は意味を持っていた。ある記号に割り当てられたのは動物であり、その後に続く記号によってその動物がキリンなのかクマなのかといった分類をしていることが分かった。つまり、自然言語における文字や単語のようなものが学習によって創発されていたことになる。

実験では、生成された言語がどの程度、自然言語に近いのかも確かめられ、ある程度、概念に共通部分があることが分かった。このような学習の過程により、究極的には自然言語に匹敵するような文法構造や品詞体系を獲得できるのかはこれからの課題である。

機械は人間とは違って並列にコミュニケーションをとることができ、さらにベクトルのような連続表現も直接伝えることができる。必ずしも離散的な系列で情報を表現する必要はない。さらに、人間は数人としか同時にコミュニケーションがとれないが、機械は数万台ともコミュニケーションをとることができる。こうした機械の特性を活かした新しい表現方法、コミュニケーションを作ることができるかが注目されている。

1) S. Kirby,"Natural Language from Artificial Life," Artificial Life, vol.8, no.2, pp.185-215, 2002.
2) S. Havrylov et al.,"Emergence of Language with Multi-agent Game : Learning to Communicate with Sequences of Symbols," ICLR 2017 workshop submission.

（2017年4月号掲載の記事に加筆）

11-3 自由な言葉でロボットに指示をする
Unconstrained Spoken Language Instruction for robots

筆者の勤務先のPreferred Networks（PFN）の論文[1]がICRA 2018のHuman Robot Interactionのベストペーパーに選ばれた。この論文ではロボットに対し自由な言葉でピッキングタスクの指示を出し、ロボットがそれに応じて作業するという研究を扱っている（図11-1）。今回は、この技術背景や使われた技術の詳細について紹介していく。

従来、ロボットに対してタスクの指示をする場合、プログラムを書くか、ダイレクトティーチングをするか、人間や別のシステムの動作を参考に模倣学習するか、といった方法が取られてきた。これに対し、人が他人に指示を出す時のように、ロボットに対しても話し掛けることでタスクを指示する方法は、従来の手法と比較しても有望である。言語は訓練なしに誰でも使うことができるコミニケーションツールである。無限ともいえる表現力があり、抽象的で時空間を超えた表現（例：明日までに1階の部屋の床にちらかっているものを全部片付けて）を扱うことができるからである。

しかし、これまで言語を使ったタスクの指示はまだ実用化されていない。自由に話された言語を理解することや、それらを現実世界の作業にマッピングすることが技術的に困難であったからである。それがディープラーニングの登場以降、音声認識、言語理解、画像認識の急速な発展、そしてそれらの結果を自然に統合できるようになり、言語を使ってロボットに指示することが可能となる素地が整ってきていた。

物体を別の箱に移すタスクを想定

それでは今回の研究の紹介に移ろう。今回の研究[1]はロボットに対し自由な言葉で指示を出し、ロボットがその指示に従ってピッキングタスクをこなすものである。初めて現実的な難易度、スケールの問題を必要な機能を全て統合して解くことができた。

具体的な課題として、4つの箱の中にばらまかれた日用品を、別の箱に移動するタスクを扱った。このタスクの指示を言葉で出す。扱う日用品の種類は100種類弱である。ここから20種類程度の日用品をランダムに選択し、ボックスにばらまく。これらの商品はお互い重なっている場合もあり、姿勢も自由である（図11-2）。また、作ったモデルが汎化するかを調べるため、テスト時には学習時にはなかった22種類の未知の日用品も加えて評価している。さらに、日用品のいくつかは名前が思いつかないようなものも用意しているほか（例えば日本のコニシの接着剤「ボンド」は海外出身者には馴染みがなく、黄色く赤い蓋がある容器と説明された）、同じ製品で種類が違うものなども用意している。こうした場合、指示者は形状や色、穴が開いているなどの容器、似ているものなどを表現して指示をする必要がある。

例えば、指示者は次のように指示を出す。「茶色いふわふわしたものを、その下の箱に移動して」（論文では英語による指示の例として紹介されている）「右下にあるティッシュボックスを左上の箱に移動して」「たくさん穴のあいたものを右上の箱に移動して」などである。また、指示だけからは複数の候補までにしか対象の物体を絞れず、曖昧さが残る場合は、ロボットが次のように聞き直す。

作業者「ティッシュボックスを左上に移動して」
ロボット「どれですか」（複数のティッシュボックスの候補を示した上で）
作業者「ぬいぐるみの近くにある方」

といった具合である。これらの指示は話し言葉であることから、書き言葉に比べてかなりくだけた表現になっており、文法的にも正しくない場合も多い。

このシステムは大きく5つのサブシステムから構成される。

1）音声指示を書き起こす音声認識システム
2）画像に対し物体認識を行い、候補物体のbounding boxを出力する画像認識システム
3）書き起こされた指示と、画像認識結果を受け取り、どの物体に対し指示しているのかを推定する物体選択システム
4）書き起こされた指示から、どの箱への移動を指示しているかを推定する箱選択システム
5）指示が曖昧であると判断した場合は、詳細を聞き直す確

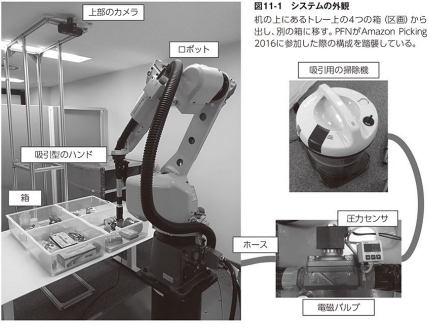


図11-1　システムの外観
机の上にあるトレー上の4つの箱（区画）から物体を取り出し、別の箱に移す。PFNがAmazon Picking Challenge 2016に参加した際の構成を踏襲している。

認システム

これらの詳細を順番に説明する。

1）音声認識システムには今回、クラウドの音声認識サービスを利用した。こうした音声認識システムは汎用的にデザインされており、特定のタスクでは性能が劣化する傾向がみられた。例えば、今回のタスクでは箱にはない物体を音声認識結果として出すことが多くみられたほか、「真下」を「明日」と聞き間違えることがみられた。音声認識システムをタスクや話者、他のセンサ情報を使って特化させ、他のモジュールと繋げてend-to-endで学習することで性能を大きく向上できるだろうと考えられる。これらは今後の課題である。

2）画像認識システムには速度と性能のバランスからSSD（Single Shot Detector）を使った。SSDはCNN（畳み込みニューラルネットワーク）を使い、特徴マップの各位置から候補とその確信度を生成する。また、物体検出時には物体の種類は考慮せず、物体であるか背景であるかだけを判断し、「物体らしさ」に従って検出している。これにより、さまざまな物体検出の学習データセットを利用することができる上、未知の物体も検出するよう汎化することが期待できる。

3）物体選択システムは、画像情報に由来する部分と音声情報に由来する部分から成る。画像情報に由来する部分は始めにSSDの検出結果のbounding boxに従って、各物体が写っている画像領域を切り出す。次に切り出された画像領域に対しCNNとMLP（多層パーセプトロン）を使って各物体の特徴ベクトルを計算する。音声情報に由来する部分は、音声認識システムによって書き起こされた指示からLSTMとMLPを使って、特徴ベクトルを計算する。指示はくだけた表現になっており、文法的にも間違っている場合が多いことか

図11-2　物体の並べ方の例
4つの箱に実際に物体を並べた様子。物体同士が重なりあっているところもある。

ら、人手で作ったルール、辞書や文法処理は一切使わず、ニューラルネットワークのみで処理をしている。そして各物体の特徴ベクトルと指示の特徴ベクトル間のコサイン類似度を計算し、類似度が高いサンプルを対象物体として決定する。

4) 箱選択システムは音声の指示を基にどの箱への移動を指示しているのかを推定する。この推定モジュールは、指示を特徴ベクトルに変換する3) のネットワークと同じ構造を利用している。

5) 確認システムは認識結果に自信が無い場合に聞き直す。自信が無いかは、一位の指示対象物体候補のスコアと二位以下の指示対象物体候補のスコアとの間の差が小さいかどうかで判断し、小さい場合、聞き直すようにする。また、学習時には正解の対象物体のスコアがそれ以外の物体のスコアに大きなマージンをもって差がつくように学習する。今後は認識結果の不確実性を考慮して聞き直すといったことも考えられるだろう。これらのシステムは全てつながっており、学習時はサブシステム毎に学習し、それらのパラメータを最適化した。

システム全体の精度は73.1%

性能はシミュレーション（各サブシステムごとの性能評価）および、実際のロボットを使って評価した。ロボットにはファナックの「M-10iA」を使い、ハンドには吸引型を利用した。このロボット構成はPFNが「Amazon Picking Challenge 2016」に参加した際のものを踏襲しているが、ピッキング性能は精度、スピードともに大幅に向上している。

学習・検証データはクラウドソーシングサービスのAmazon Mechanical Turkを使って構築した。指示の学習データの構築は、実際のタスクとは逆に、「この物体をここに移すとしたらどういう指示を出すか」という質問をアノテーターに対してすることでデータを作った。この学習データはサイト[1]で公開している。

システムの性能は各サブシステムごとの評価と実際にタスクがこなせたかのend-to-endの評価、その両方を行った。実際のロボットを使った実験で、正しい目標の箱を推定する精度は89.7%、正しい目標物体を選択する精度は75.3%、物体をピックして別の箱に置くピッキングシステムの精度は97.3%であった。また、これらのサブシステムを全て繋げたend-to-endのシステムの精度は73.1%であった。1つの物体移動の指示を出す際、平均して0.45回の聞き直しをしていた。曖昧な場合に聞き直しをすることで正しい目標物体を選択する精度はシミュレーター上では88.0%から92.7%に向上しており、聞き直すことが重要であることが再確認できた。

本研究は自由な話し言葉を使ってロボットに物理的なタスクを指示する最初の研究である。今後、精度向上、安定性向上を目指すとともに、よりチャレンジングなタスクに取り組んでいきたい。

1) J. Hatori et al., "Interactively Picking Real-World Objects with Unconstrained Spoken Language Instructions," ICRA2018.
2) https://projects.preferred.jp/interactive-robot/

（2018年7月号掲載の記事に加筆）

BERT：言語理解の事前学習

自然言語処理においてBERTと呼ばれる事前学習手法は大きなブレークスルーを起こした。これまで画像認識の分野ではImageNetの画像分類タスクで学習して得られたモデルを他のタスクの初期パラメータとして使う事前学習がよく使われてきた。事前学習によって、様々な画像認識を解くのに必要な特徴抽出器が既に得られており、新しいタスクを学習する場合にはそのタスクに固有の部分だけ学習すれば済むため、学習データが少ない場合には特に有効なアプローチである。

自然言語処理でも事前学習が有効なのではないかと以前から考えられていた。例えばWord2VecやGloveなどの単語表現の事前学習では、次の単語を予測するタスクを解くことで、各単語の連続なベクトル表現を得ることを可能とした。これらの連続表現は単語の意味を表し、そのベクトル上で様々な演算が可能であり、その表現を使うことでその後のタスクの性能を向上できる場合もあった。この後、質問応答や文意理解などで求められるような文や段落レベルでの表現を事前学習するparagraph vectorやskip thoughtなども登場していたが、その利用は限定的であった。2018年に登場したELMoは双方向で2つの言語モデル（次の単語予測モデル）を学習し、その表現を事前学習として使うことで様々なタスクを解けることを示した。

SoTAを大きく更新したBERT

そのような中、事前学習手法として2018年10月に米グーグルAIからBERT（Bidirectional Encoder Representation from Transformers）が提案された[1]。BERTは多くの言語理解タスクのSoTA（最高精度）を大きく更新した。さらにその後、BERTを改善した手法が次々登場し性能改善されている。例えばSuperGLUE[2]と呼ばれる、複数の言語理解タスクから成るベンチマークでは、BERTをベースにしたモデルは人の性能に近づくか並ぶまでになっている[3]（人のスコアが89.8なのに対し、BERTを元にしたRoBERTaが84.6、単語の連続表現を利用したCBoWベースの手法は44.5）。BERTが2018年のNAACLのベストペーパーを受賞しただけでなく、発表からわずか1年弱で引用数が1000を超え（4年後の2022年4月時点で3万7000を超えている）、その利用が自然言語処理以外にも広がっていることもそのインパクトの大きさを示している。

BERTのアイディア自体は単純である。まず、複数の連続する文を境界記号を挟んだ上でつなげた系列を入力とする（オリジナルのBERTでは他の文のサンプリング方法も提案されていたが、文献3）では連続する文で十分であることが示されている）。次にランダムに一部（例えば15%）の単語をマスクし（適当なマスク記号で置き換え）、マスクされた単語を周囲の文脈から予測するタスクを解く。このタスクを解けるように学習されたモデルを事前学習済みモデルとし、様々なタスクの学習時の初期モデルとして利用する。各タスク上ではfine-tuningを行う。なお、事前学習時には入力の一部がマスクされているのに対し、その後に使う時のタスクでは入力はマスクされていないので入力分布の不一致が発生してしまう。そこで事前学習時の予測タスクでは予測対象の単語は全てマスク記号に置き換えず、一部はそのまま単語を残し、一部は別の単語にランダムに置き換えることで、この不一致を抑えるようにしている。

なぜBERTはうまくいったのか

このBERTが成功した点は次の二点である。

1つ目はBERTは予測の際に前後の文脈を使うという点である。似たようなタスクとしてELMoでも使われた言語モデルがある。それまでの文から次の単語を予測するようなタスクであり、言語モデルを事前学習として使う例は多く存在している。言語モデルでは、予測は前の文脈から次の単語を予測するという一方向であり、前から順番に単語を生成する生成モデルを学習しているとみなすことができる。それに対しBERTでは、予測する際には前後の文脈をみることができ、生成モデルを諦めたモデルを使っている。この双方向による予測は言語理解では重要であり、例えば単語を予測する場合にその前方の文脈をみて、次に後方の文脈をみて、最後にそれらを総合して予測するといったより複雑なプロセスを使

うことができる。このような文の前後を状況に応じて使うことは言語理解において重要である。また、前後の文脈として同一文内にある単語だけでなく、隣接する周囲の文も使って予測する点が重要である。入力は連続する文をつなげて構成しており、予測に必要であれば周囲の文の情報を使えるようになっている。これにより文を超えた言語理解が可能となる。

2つ目はTransformer（自己注意機構、self-attention）を使っている点である。Transformerは注意機構を使い、データの流れ方をそれまでの処理結果に応じて動的に変えられる仕組みを持っている。これにより、現在までの処理に応じて例えば前方の文脈から情報を集めたり、または後方の位置での処理に情報を渡したりすることができる。さらにTransformerは離れた位置にある情報も1ステップで集約することができる。それに対し、CNNやRNNの場合は離れた位置にある情報を集約するにはその距離に比例または近いステップ数が必要となる。言語理解では、離れた位置にある情報が必要になることも多く（例えば、ある単語を前にも言及している場合でその単語の周辺情報を集めたい場合）、そうした場合でもTransformerは必要に応じて1ステップで集めることができる。またTransformerは疎な注意機構を使っているとみなせるのでMixture of Expertモデル[6]と同様にモデルの表現力を上げやすく、表現力が必要な言語処理においては有効である。

一部分をマスクする、前後の文脈を使うというアイディア自体は単純であり思いつくため、おそらく多くの人がこれまで考えて試してみて、良い結果が出ていなかったのではないかと思われる。BERTが成功したのはTransformerの登場が大きいが、もう1つは、大量の学習データで大きなモデルを使い莫大な計算コストをかけて学習させる実験が容易に可能になったためだと考えられる。

BERTを改良する研究は既に多く発表されている。その中でも米University of WashingtonのPaul G. Allen School of Computer Science & Engineering、米Facebook AIから成るチームはBERTを改良したRoBERTaを提案[5]しており、現在最も性能が良い。RoBERTaはBERTに加え、1）各iteration毎に異なるランダムなマスクを使う、2）学習時のバッチサイズを大きくする、3）訓練データの走査回数を10倍近く増やすことで性能を大きく改善している。基本的に現在のBERTはUnderfit（未学習）しており、より大きなモデルを使い、より長い学習時間をかければ性能はまだ上がると報告している。

RoBERTaは言語理解タスク上では人に匹敵する成績を上げており、言語理解に向けて大きく前進したといえる。一方でこの結果をもって現時点で言語に関連するタスクが何でも解けるようになったとは考えないほうがよいだろう。この言語理解タスクカバーしているのはほんの一部である。実用化においては例えば、現実世界と言語間のマッピングをとる記号接地問題、心理モデル、会話などの教示が難しいタスクをどのように教えるかなど多くの重要な問題が残されている。今後、自然言語処理研究が加速していくのは間違いないが、何が解けているかについては慎重に評価が必要だ。

計算負荷は非常に重い

BERTを使った事前学習、およびそれを使った推論には大きな計算リソースが必要とされる。そのため、グーグル、Facebook社、米Microsoft社など大きな計算リソースを利用できる組織でしか新しいモデルを作れないともされ、実際にベンチマークの上位はこれらの企業から成る。例えばクラウドでの学習の1回の実験に、最初のBERTは約73万円、関連した手法であるXLNetは648万円必要であると報告されている[4]。オリジナルのBERTの学習では16個のCloud TPU v3 Podを使い3日の学習時間が必要であったが、2）で提案されているステップ回数を使う場合、学習に48日必要な計算となる。こうしたことから並列化による学習の高速化も必要とされ、TPUv3 16枚で3日必要だった学習時間を、TPUv3を1024枚使い、データ並列を進めることで76分まで少なくできたことが報告されている[5]。

一方で大きなモデル、データを使い、走査回数を増やすことで性能はまだ上がることが予想されている。2017年にグーグルのGeoffrey Hinton氏らは自然言語処理において、パラメータ数が1兆から成るモデルを1兆個の単語から成るデータ上で学習させることが目標と述べているが[6]、2021年に登場したSwitch Transformer[7]は1.6兆パラメータを利用し、またデータ数も大きくなっている。しかし、まだデータ数やモデルサイズの増加に対する汎化性能や後続タスクの性能向上の伸びはみられ、さらに大きなモデルが必要とみられる。

また、学習時だけでなく推論時にも大きな計算量が必要となる。既に米NVIDIA社はBERTで作ったモデルを効率的に推論可能なライブラリを提供しはじめている[8]。

言語理解の技術が急速に向上

言語や画像タスクはデータが無数に手に入るような問題設定であり、教師なし学習、自己教師あり学習による学習が最終的に有効になるだろうと考えられてきた。BERTは自然言語処理においてそのようなことが実際に可能であることを示した。言語理解タスクはこれまでとは違うペースで急速に解かれ始めている。画像認識（相互情報量最大化による自己教師あり学習）や機械翻訳（Back Translation）などにおいても同様の現象がみられつつある。

人が教師なし学習、自己教師あり学習を今のAIよりもずっとうまくできている理由として、AIにとって学習可能な決定的な方法がみつかっていないだけではなく、モデルの大きさやモデルの更新回数に人とAIでは圧倒的な差があるからという可能性がある。乳児は周辺環境をぼんやりと見ていたり、人が話をしているのをずっと聞いていて、数年たったら突然周りのものを認識できるようになったり、話せたりするようになる。その間に予測や補完などの自己教師あり学習を今のAIよりもずっと大きなモデルで大量に実行しているのかもしれない。今後はなくともBERTなどの手法が今よりもずっと大きなモデル、データを使うことでより難しい問題を解けるかが確かめられていくだろう。

---

1) J. Devlin et al., "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," https://arxiv.org/abs/1810.04805
2) https://super.gluebenchmark.com/
3) Y. Liu et al., "RoBERTa: A Robustly Optimized BERT Pretraining Approach," https://arxiv.org/abs/1907.11692
4) https://syncedreview.com/2019/06/27/the-staggering-cost-of-training-sota-ai-models/
5) Y. You, et al.,"Large Batch Optimization for Deep Learning: Training BERT in 76 minutes," https://arxiv.org/abs/1904.00962
6) N. Shazeer et al., "Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer," https://arxiv.org/abs/1701.06538
7) W. Fedus et al.,"Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity," https://arxiv.org/abs/2101.03961
8) https://devblogs.nvidia.com/nlu-with-tensorrt-bert

（2019年10月号掲載の記事に加筆）

第12章

制御

12-1 確率的制御：不正確な制御が学習を助ける

現在の多くの制御が最適制御によって実現されている。これは、制御の挙動そのものの悪さまたは良さを表すコスト関数を設定し、このコスト関数を最小化するような操作系列を求めることで制御を行うものである。

この場合、制御ははじめに最適制御によって軌道などの計画を立て、次にその計画に沿って実行される。例えば、ものをつかむような軌道を生成したい場合は開始点と終了点、速度や加速度などの初期条件の下でコスト関数を最小化するような経路を前もって求め、実行中にはその経路をトラックするようにして制御を実現する。実行中に、外力やノイズなどによって計画から外れそうになると、計画に戻るようにフィードバックが掛かる。これにより、何回繰り返しても全く同じ正確な作業が実現される。

人や動物は計画自体を常に修正

一方、人や動物の動作（例えば走る動作や、ものをつかむ動作など）は全く同じ動作を繰り返すことはなく、各動作間にはずれがあることが知られている[1]。最適制御のように経路の計画を立てた後、それを変えずに実行するのとは違って、人や動物の制御では実行しながら状況に応じて計画や制御出力を修正し続け、目的を達成するように制御されている。

これには様々な理由がある。1つは環境中の予測不可能な外乱や内部システム（筋肉や腱など）のエラーが大きいため、遠くの未来がどうなるのかを正確に予測するのは困難だからである。そのため綿密な計画は立てずに、最終的な目標を見据えつつも、実行しながら近い未来だけを計画を立てていくのが合理的だからである。

2つ目はエネルギー最小化の点からである。状態を計画に沿うように修正するのにもエネルギーが必要であるし、修正によって新たに生じたノイズを修正するのにもエネルギーが必要である。そのため、状態修正の際にはタスクを達成するのに必要な部分だけ修正し、タスクとは関係のないずれは修正しない。

3つ目は行動の多様性を生むことで様々な情報を集めることができ学習における探索の面から有利である点である。

試行錯誤で情報を集める強化学習

人や動物の場合、最終的な制御は学習によって獲得される。この学習は強化学習の枠組みで捉えることができる。最適制御はコスト関数を最小化するような制御を求める問題であり、強化学習は将来にわたっての期待報酬の合計（期待収益）を最大化するような行動を選択する方策を獲得する問題である。

ここで、報酬を負のコスト関数とすれば、最適制御と強化学習が解いている問題は同じとみなすことができる。強化学習においても、期待収益を最大化するような最適な方策は決定的、つまり与えられた状態に対して最適な行動が一意に定まる。一方、期待収益を最大化するだけでなく探索も実現したい場合、様々な試行錯誤を行って情報を集めることが必要になり、確率的な行動選択が必要となる。

多様な動作によって様々な行動やその結果を経験しておくことによりノイズに強くなるだけでなく、その後の別のタスクにも役立つようになる。例えば、“もの”のつかみ方も一番良いつかみ方を1つだけ獲得するのではなく、少し性能は劣るが異なるつかみ方をたくさん学習しておくことで、次に発展したタスク（つかんだものを穴に入れる）や別のタスクを学習する際に、違うつかみ方を使って効率よく学習することができる。

見たことがない状態を効率的に探索

従来、このような確率的な方策の多くはヒューリスティックスによって設定されていた。例えば、ϵ-貪欲方策では、一定の確率でランダムな行動を選択することによって探索を実現する。

近年はこの探索の効果を定式化し、確率的方策を導出する手法が登場している。一例として、今回はUniversity of California BerkeleyのHaarnoja氏らによるエネルギーモデルを使った強化学習[2]を紹介する。 強化学習は、各時刻tにおいてエージェントは環境から状態s_tを受けとり、行動a_tを方策$\pi(a_t|s_t)$に基づいて選択し、環境から報酬$r(s_t, a_t)$を

受ける。環境はその行動に基づいて更新する。

従来の強化学習では、将来の期待収益$\Sigma_t \mathrm{E}[r(s_t, a_t)]$を最大化するような方策を求めていたが、これでは探索は考慮されない。そこで、まだ見たことがないような状態を効率的に探索できるように、報酬に加えて方策のエントロピー$H(\pi(\cdot|s_t))$も最大化する問題を考える。

$$\pi^* = \max_{\pi} \sum_t \mathrm{E}[r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))]$$

ただし、期待値は方策に従う状態行動分布(s_t, a_t)に従ってとり、$\alpha > 0$は探索をどれだけ重視するかを決めるハイパーパラメータである。この方策は現在の方策のエントロピーを最大化するだけでなく、将来にわたってのエントロピーを最大化していることに注意してほしい。これにより、まだ観察したことがないようなエントロピーが高い状態を好んで探索するようになる。

この定式化における最適な方策はソフト行動価値関数$Q^*(s, a)$と、ソフト状態価値関数$V^*(s_t)$を使って、

$$\pi^*(a_t|s_t) = \exp\left(\frac{1}{\alpha}(Q^*(s, a) - V^*(s_t))\right)$$

$$Q^*(s_t, a_t) = r_t + \mathrm{E}\left[\sum_{t'} \gamma^{t'}\left(r_{t+1} + \alpha H(\pi^*(\cdot|s_{t+1}))\right)\right]$$

$$V^*(s, t) = \alpha \log \int \exp\left(\frac{1}{\alpha} Q^*(s_t, a')\right) da'$$

のような確率的な方策になることが分かっている。

各行動は、$\exp(Q^*(s_t, a_t))$に比例する確率で選択され、各行動価値関数が分子、状態価値関数は分母にあるような関数となる。この場合、$\alpha = 0$とした場合が従来の最適方策と同じとなり、その時の最大の行動価値関数の値を持つ行動が決定的に選ばれる。

このように探索も考慮したエントロピー最大化を含んだ場合でも方策や価値関数の間には美しい関係が成り立つが、問題となるのは、状態や行動が高次元の連続空間の場合にどのように行動価値関数$Q(s_t, a_t)$をモデル化するかである。行動価値関数は十分な表現を持ちつつ、状態s_tが与えられた時、各行動a_tを高速にサンプリングできる必要がある。

実際に効率的な探索を実現

論文[2]では、$Q(s_t, a_t)$にはニューラルネットワークを使って回帰モデルを作り、$\pi^*(a_t|s_t)$から行動a_tのサンプリングは高次元のエネルギー関数を基にした生成モデルの1つであるAmortized SVGDを使うことで解決している。このAmortized SVGDはエネルギー関数に従った確率分布に従ったサンプルを、決定的な関数を使ってノイズから生成（GANと同様のアイデア）することができ、MCMC（マルコフ連鎖モンテカルロ法）などとは違って非常に高速に大量にサンプリングすることができる。

Haarnoja氏らの実験では、確率的方策を使うことで実際に効率的に状態空間を探索することができ、例えば望ましい状態が複数ある場合も1つだけ調べるのではなく複数調べることができるようになっている。また、この方法によって学習されたモデルを基により難しい問題を解くことができることも示されている。

このような“不正確な”制御が学習に役に立つことは興味深い知見である。

1) E. Todorov et al.,"Optimal feedback control as a theory of motor coordination," Nature Neuroscience vol.5, no.11,pp.1226-1235, 2002.
2) T. Haarnoja et al.,"Reinforcement Learning with Deep Energy-Based Policies," ICML 2017.

（2017年10月号掲載の記事に加筆）

12-2

オンライン学習と最適制御
未知ノイズにも頑健な制御手法

線形ダイナミクスを持ったシステムに対する制御問題は古典的な問題であり広い分野でみられる。この制御問題に対し、近年オンライン学習を適用することでノイズやコストに対する仮定を大幅に緩和し、かつコスト関数が動的に変わる場合も対応できる方法が提案されている。これについて紹介しよう。

本稿では離散時間の線形システムの最適制御問題を扱う。時刻 t におけるシステムの状態を x_t、行動を u_t、遷移ノイズ（または外乱）を w_t とした時、次の時刻の状態が次のように決まるとする。

$$x_{t+1} = Ax_t + Bu_t + w_t$$

今回の問題設定では遷移ダイナミクスを表す行列 A、行動の状態への影響を表す行列 B は既知であるとする。各時刻でコスト $c(x_t, u_t)$ が定義され、時刻 $t=1$ から $t=T$ までのコスト合計 $\sum_{t=1}^{T} c_t(x_t, u_t)$ を最小化するような行動列 $\{u_t\}_{t=1}^{T}$ を求めることが目標である。これまでの情報から行動を決定する関数を方策と呼ぶことにする。例えば、現在の状態に線形に依存して行動を決める線形方策はよく使われる。

$$u_t = -Kx_t$$

従来、このような制御問題ではノイズ w_t がガウシアンであると仮定される場合が多い。しかし現実の問題でこの仮定が成り立たない場合も多い。これに対しノイズが最悪の場合を考慮し、その場合の最適制御を目指す H_∞ 制御が提案されている。これはノイズが最悪だと想定した場合の各時刻の最適な線形制御を求める問題であり、次のような問題を解いて制御を得る。

$$\min_{K_1} \max_{w_{1:T}} \min_{K_2} \cdots \min_{K_t} \max_{w_T} \sum_t c_t(x_t, u_t)$$

H_∞ 制御は頑健な制御を達成できるが、最悪のノイズの場合を考えた悲観的な制御であり、より現実的なノイズの場合に達成可能な最適性が損なわれる可能性がある。また、コスト $c(x_t, u_t)$ が x_t, u_t の二次関数で表されると仮定する場合も多い。この場合の最適制御は線形二次レギュレータ（LQR：Linear Quadratic Regulator）として知られ、Riccati方程式を解くことで最適制御を得ることができる。

この最適制御問題に対してオンライン学習を適用することで、任意のノイズを対象とし、またコスト関数も二次関数より広い凸関数を対象にした場合でも最適性を達成することができる。はじめにオンライン学習を説明しよう。

オンライン学習

オンライン学習とは次々と観測されるデータ x_t に対し、行動 u_t を決定し、その結果に応じてフィードバック $c_t(x_t, u_t)$ を受け取り、パラメータを即時更新して適応していくような手法である。現在のニューラルネットワークの学習では同じ訓練データを何回も走査し、パラメータをゆっくり更新していくのに対し、オンライン学習では基本的にデータは一度しか観測しない。

オンライン学習で特徴的なのは、そのアルゴリズムの性能をRegret（後悔）と呼ばれる指標を使って厳密に評価できる点である。Regretは方策による行動決定が最適な方策による行動決定に対してどれだけ悪いのかを評価する。"自分が最適な方策を使っていたらこれだけ達成できたのに"という後悔を測ることになる。

それでは今回の問題設定におけるRegretを定義する[1]。はじめに任意の方策 $\mathcal{A}$ の性能を

$$J_T(\mathcal{A}) = \sum_{t=1}^{T} c_t(x_t, u_t)$$

と定義する。次に比較対象として、先程あげた線形制御を使い、（一般に未知である）最適な線形方策に対して方策 $\mathcal{A}$ がどれだけ悪かったかとしてRegretを定義する。$J_T(K)$ は線形方策を使った場合の性能とする。

$$\texttt{Regret} = J_T(\mathcal{A}) - \min_K J_T(K)$$

Regretは時間 T に対し、どのようなオーダーを持つのか

が重要である。もし、オーダーが $\mathcal{O}(T)$ より小さければ、時刻当たりのRegretは時刻 T が大きくなるに従い 0 に収束する。つまり $\mathcal{A}$ と最適な線形制御 K^*の時間当たりのコストはほぼ一致することを意味する。

Regretが $\mathcal{O}(\sqrt{T})$ であれば、一回当たりの誤差は $1/\sqrt{T}$ に比例して減っていき、$\mathcal{O}(\log T)$ であれば、$1/T$ ($\log T$ はほとんど定数とみなせるため) に比例して減っていく。これは非常に速い収束であり、この条件下ではこれより速い収束は達成できないことが知られている。

オンライン学習を使った制御はコスト関数が凸関数であればRegretは $\mathcal{O}(\sqrt{T})$、さらに強凸関数であれば $\mathcal{O}(\log T)$ を達成できる[2]。なお、凸関数は1入力関数であれば二次微分が非負、強凸は二次微分が正であるような関数である。

オンライン学習による最適制御

それではオンライン学習を使ってどのように制御するのかについて説明しよう。オンライン学習は手法自体は単純であるが、その背後にある理論や証明は紙面の都合上紹介できない。ここでは代わりに基本的な考え方のみ説明する。最初にいくつか概念を紹介する。これらは全てオンライン学習による制御に必要な概念である。

はじめにノイズが無い場合の線形制御の安定性について考えよう。この場合、次の時刻の状態は $\bar{A} = A - BK$ と定義した行列で支配され、非零の状態がどのように時間発展するかを表す。

$$\begin{aligned} x_{t+1} &= Ax_t + Bu_t \\ &= Ax_t - BKx_t \\ &= (A - BK)x_t \\ &= \bar{A}^t x_1 \end{aligned}$$

線形フィードバックゲイン制御 K により整形された行列 $\bar{A}$ が $\bar{A} = QLQ^{-1}$ と分解され、L は対角行列であり $||L_i|| \leq 1-\gamma$, $||K|| \leq \kappa$, $||Q||, ||Q^{-1}|| \leq \kappa$ を満たす時、この K は強 (κ, γ) 安定であると呼ぶ。

次に外乱がある場合を考えよう。方策の中でもこれまでの外乱 $(w_{t-1}, w_{t-2}, \ldots)$ に線形に依存して次の制御が決まる方策を外乱行動方策と呼ぶことにする。

$$u_t = -Kx_t + \sum_{i=1}^{H} M_t^{(i)} w_{t-i}$$

ただし、K は固定とする。なお、ノイズ w_{t-1} は直接観測できないが、次の状態 x_t が求まれば $w_{t-1} = x_t - Ax_{t-1} - Bu_{t-1}$ と計算できることに注意してほしい。

この外乱行動方策は $M_t = \left(M_t^{(1)}, M_t^{(2)}, \ldots, M_t^{(H)}\right)$ というパラメータで特徴づけられる。外乱行動方策は表現できる関数の自由度よりパラメータ数の方が多い過剰パラメータモデルである。一見すると外乱に依存した形で方策を定義することは自然ではないように感じられるが、このような定式化によってコスト関数が最適化対象パラメータを線形変換した結果に対して凸であり、コスト関数の凸や強凸といった性質を保てることが分かっている。

また、時刻 $t-H$ の状態を $x_{t-H} = 0$ と仮定し、それから各時刻において $M_{t-H}, \ldots, M_t$ のパラメータを持った方策で制御 u_t を選択していった結果、時刻 t に到達した状態を理想状態 y_{t+1} 、またその時に時刻 t で取る行動を理想行動 v_t と定義する。この理想状態、理想行動から決まるコストを理想コストと呼ぶ。

$$f_t(M_{t-H}, \ldots, M_t) = c_t(y_t(M_{t-H}, \ldots, M_{t-1}), v_t(M_{t-H}, \ldots, M_t))$$

この理想コストは $M_{t-H}, \ldots, M_t$ に対して線形であることに注意されたい。この理想コストは、実際に観測するコスト $c_t(x_t, u_t)$ とは異なるが、強 (κ, γ) 安定である方策間ではこれらのコスト差が κ, γ を使った関数で抑えられ、理想コストに対して最適化をしても実際のコストを抑えられることが分かっている。

また、$f_t(M) = f_t(M, \ldots, M)$ とおく。つまり過去の時刻全てで M というパラメータで決まった方策に従って行動を選択した場合の理想コストを表す。これは記憶を備えたオンライン学習で使われる手法である。

それではオンライン学習を使った制御を説明しよう。$||B|| \leq \kappa_B$ と仮定する。はじめに強 (κ, γ) 安定である線形制御 K を1つ求める。これは線形ダイナミクスが分かっていれば、求められることが知られている。この K は以降固定とする。次に重み $M_0 = \{M_0^{[1]}, M_0^{[2]}, \ldots\}$ を適当に初期化する。このアルゴリズムは各時刻 t において外乱行動方策

$$u_t = -Kx_t + \sum_{i=1}^{H} M_t^{[i]} w_{t-i}$$

で行動を選択し、次時刻の方策のパラメータを以下の射影

付勾配降下法で更新する。

$$M_{t+1} = \Pi_{\mathcal{M}}(M_t - \eta \nabla f_t(M_t))$$

ただし $\Pi_{\mathcal{M}}$ は凸領域 $\mathcal{M}$ への射影であり、

$$\mathcal{M} = \{M = \{M^{[1]}, \ldots, M^{[H]}\} : ||M^{[i]}|| \leq \kappa^3 \kappa_B (1-\gamma)^i\}$$

である。

このアルゴリズムによるRegretはコスト関数が凸関数であれば $\mathcal{O}(\sqrt{T})$ 、強凸関数であれば $\mathcal{O}(\log T)$ を達成することが示せる。

その他の問題

この問題では線形ダイナミクスのパラメータは既知であるとしたが、未知の場合も勾配降下法を使ってこれらのパラメータを推定できることが分かっている[3]。これらの手法を組み合わせることで、未知のダイナミクスを持ったシステム上でオンライン学習で制御することが考えられる。

また、今回の問題では状態変数が直接観測できる場合を扱った。実際には状態変数には直接観測できるものとできないものがあるのが一般的である。この場合は観測から状態変数をオンライン推定する必要があり、カルマンフィルタやパーティクルフィルタなどが知られている。近年ではカルマンフィルタとDNNを組み合わせて推定する手法も登場している[4]。これらの手法では状態を分布として持つことで不確実性に対応している。オンライン学習を使った手法でも不確実性を考慮した最適制御が今後考えられるだろう。

オンライン学習は統計的学習理論とは違って、性能保証が確率的ではなく、常に成り立つ形で与えられるという特徴がある。Regretは未知の最適方策に対する相対的な性能評価であり、絶対的な性能保証ではないが、何らかの形で最適方策（最適パラメータ）の存在やその性能が分かる場合には、強い理論的な性能保証が与えられることになる。

またオンライン適応というのも大きな魅力である。今回の制御問題においても、コストが動的に次々変わっていく場合でも、どの時間を切り出しても性能保証が達成できる、つまりそれぞれの最適な方策にすぐに切り替えられることが言える。

1) N. Agarwal et al., "Online Control with Adversarial Disturbances," ICML 2019.

2) N. Agarwal et al., "Logarithmic Regret for Online Control," NeurIPS 2019.
3) M. Hardt et al., "Gradient Descent Learns Linear Dynamical Systems," JMLR 2018.
4) P. Becker et al., "Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces," ICML 2019.

（2019年12月号掲載の記事に加筆）

第13章

シミュレーション

13-1 AIによるシミュレーションの進化

シミュレーションまたはコンピュータモデリングはシミュレーション自体の技術の発展と計算性能や性能コスト比の指数的な改善によって広い分野で使われるようになっている。シミュレーションは対象問題の原理・原則、公式が分かっているなら、データに基づかなくても正確にシミュレーションできる。化学分野の言葉を借りれば"ab initio"(第一原理のみに従って)としてデータや経験を使わずとも複雑な現象を非常に正確にシミュレーションできる。一方でシミュレーションが達成できる精度と必要な計算量は改善の余地が大きい。精度と計算量はトレードオフの関係にあり、高精度のシミュレーションを実現する場合には計算量が問題となる。扱う要素数(物体数、電子数など)が増えるにつれ、計算量は急激に増加する。また、デジタル計算機では連続量は扱えないため、時間や空間を量子化する必要がある。この場合も精度の高いシミュレーションのためには高い解像度が必要となるが、計算量は増加する。

こうした問題を解決する手法として、AIを使ってシミュレーションを高速化したり高精度化することができるようになっている。今回は剛体物理シミュレーション、量子化学シミュレーション、コンピュータレンダリング(光学系のシミュレーション)においてAIがシミュレーションでどのように使われているのかについて紹介していく。

剛体物理シミュレーション

始めに質量を持った複数の剛体が相互作用しながら将来どのような軌道を描くのかをシミュレーションする問題を考えよう。ここではニュートン力学やそれを一般化したハミルトン力学で記述できる古典力学を扱う。

外力が加わらないような環境においては、n 個の物体の座標 $\mathbf{q} = (q_1, q_2, \ldots, q_n)$ と運動量 $\mathbf{p} = (p_1, p_2, \ldots, p_n)$ (時間を引数にとるがここでは省略している) のダイナミクスはハミルトニアン $\mathcal{H}(q, p, t)$ と呼ばれる関数の勾配に従い記述できることが分かっている。

$$\frac{\partial \mathbf{q}}{\partial t} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}}, \ \frac{\partial \mathbf{p}}{\partial t} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}$$

このダイナミクスは可逆でありエネルギー保存の法則が成り立つ。また、この式を用いることで現在の状態(位置と運動量)が与えられれば将来や過去の状態は一意に決定される。例えば一定時刻後の位置を求めるには、このダイナミクスに従い、変化量の積分を時間方向に求めれば良い。しかし実際には誤差が生じる。

このようなシミュレーションで生じる誤差は大別してモデル化誤差(例:系の全ての物体が考慮されていない、推定したハミルトニアンが間違っている)、数値計算誤差などがある。例えば、重力の相互作用が起こる場合の多体問題であったり、複数の物体が複雑な制約条件の下に動くような場合、この誤差を抑えるためには時間ステップ幅を小さくする必要があり、計算量は急激に増大する。

この一定時間後の状態変化、つまり数値積分操作をニューラルネットワーク(NN)を用いて高速に解くことを考える。これは現在の状態から将来の状態を予測するニューラルネットを教師あり学習で訓練すれば実現できそうである。しかし、現在の状態から将来の状態をニューラルネットワークで直接学習した場合、短期的には正確だが、長期的には誤差が蓄積し、真の値や軌道から離れていってしまう。このような長期予測がずれていく現象は普遍的にみられる問題であり、予測に何らかの修正機構や不変性を導入しなければならない。

そこで、Hamiltonian Neural Networks(HNN)[1]は次の状態を直接予測するのではなく、ハミルトン力学に従って予測する。HNNは状態 $(\mathbf{q}, \mathbf{p})$、からハミルトニアンのスカラー値を出力するネットワーク $\mathcal{H}_\theta$ を学習し、それらの勾配を使って次の状態を予測する。学習の際は観測データから得られた状態変化量が勾配と一致することを目的関数として学習する。

$$L(\theta) = \| \frac{\partial H_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \|_2 + \| \frac{\partial H_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \|_2$$

このようにして学習されたHNNは、ハミルトニアンを介さず直接将来状態をNNで予測した場合に比べて、長期予測の精度が大幅に改善されたと報告されている。このHNNでは状態は既知として扱っていたが、Hamiltonian Generative Network[2]は状態は未知である場合でも観測から状態を推定するニューラルネットワークを使うことで正確な長期予測

が可能だと報告されている。

Deep Lagrangian Networks[3]は古典力学の別形式であるラグランジュ力学に従って、状態の時間発展をモデル化する。この場合、未知のイナーシャおよび外力をニューラルネットワークでモデル化し、データからこれらを求めた上で、シミュレーションを実行する。実際のロボットを使った実験では、数分のうちにシステム同定することができ、正確に制御できるようになったと報告している。その後、こうした帰納バイアスではエネルギー保存則ではなく、二次微分方程式で表される部分だとわかってきている[4]。

このように従来のシミュレーションモデルのうちモデル化が難しい部分をニューラルネットワークで置き換えるというアプローチは他の分野でも成功している。特に多数の要素が影響しており、真面目に計算した場合に大きな計算量を必要とする問題はニューラルネットワークを使うことで精度を落とさず劇的な高速化を達成できる場合がある。流体シミュレーションにおいても計算量が大きい部分をCNNで置き換え劇的な高速化を達成できていると報告されている[5]。

量子化学シミュレーション

計算化学分野において、材料の性質を推定する一手法として、電子と陽子の状態から定まるハミルトニアンを求め、それに基づいたシュレディンガー方程式を解くことで分子のポテンシャル関数、電子状態、様々な特性（反応性、伝導性など）を推定することができる。一般に電子数が増えるにつれ計算量は急激に増加するため、現実的な時間で計算するためには何らかの工夫や近似が必要となる。例えば、現在の主要な手法の1つである密度汎関数理論（DFT）では、本来電子の数だけ波動関数を求めなければならなかった問題を解く代わりに、同じ結果が得られる電子数に依存しない電子密度を推定する問題に帰着し、状態を推定する。しかしこのような工夫をしたとしても必要な計算量は依然として大きく、シミュレーションには時間がかかる。

代わりに分子構造を入力として与えそれからニューラルネットワークを用いてポテンシャル関数を直接推定する手法が登場している。例えばANI-1[6]は5万8000分子とそれらのとりうる1720万種類の配座（各原子の位置）とそのときのエネルギーを学習データに使い、分子のとりうる配座とその時のエネルギーをニューラルネットワークで学習し、推定する。

このようにして学習されたモデルはニューラルネットワークを使って与えられた構造に対する配座とエネルギーを高速に推定できるだけでなく、配座がどのように変化するかを高速にシミュレーションできるようになる。実装や精度によるが、従来シミュレーションよりも1万倍近い高速化が達成可能であり、より多くの材料を探索することが可能となる。

必要な情報をより多く与える方法も登場している。例えばSchNet[7]は3次元構造をグリッドを使わず連続フィルタを使ってNNに入力することより自然に3次元構造を考慮することでポテンシャル関数をより正確に推定することができるようになっている。

レンダリング

コンピュータグラフィックスで使われるレンダリングは光学系のシミュレーションであり、光源から発生した無数の光線が様々な物質で反射しながらカメラにどのように入射するのかという、いわゆる光輸送を求める問題である。近年の高品質なレンダリングで使われる手法ではレイトレーシングやフォトンマッピングと呼ばれる、より光学系のシミュレーションを正確に行った手法が使われるようになっている。

このような光輸送は計算量が大きな問題となる。例えば雲のシミュレーションを考えてみよう[8]。この場合、雲に入射した光線は雲の中で無数の反射を繰り返して最終的に雲の外側に光線が出る。この反射は数回や数十回で打ち切ると現実的にみられるような雲の見た目は生成できず、数百回や数千回で雲らしいパターンが生成される。そのため、1つのリアリスティックな雲を生成するために数百のシミュレーションが必要である。この無数の反射は積分計算とみなすことができ、その近似にニューラルネットワークの利用が有効である。Deep Scatteringはこの無数回反射し、各位置にどれだけ光エネルギーが溜められているのかを雲の状態から直接推定する。この学習には長い時間シミュレーションした高精度な結果を使う。

光輸送のもう1つの大きな問題は、光源から放たれる光線のうち視点に入ってくるのは全ての光線の中のほんのわずかであり、ほとんどの光輸送シミュレーションが無駄になってしまう点である（レイトレーシングではこの問題に対処するため、視点から光源に向かって逆方向に追跡するが、光源が小さい場合、同じ問題が起こる）。このようないわゆる"Zero Contribution"の光輸送を抑えるには、乱反射の際、貢献しそうな光線を求め、それを優先的にシミュレーションすることが必要となる（不偏推定とするためには重点サンプリングで補正する）。あたかも光源から逐次的に行動（反射方向）を選

択し、視点というゴールに向かって経路を選ぶ問題とみなせる。実際、光輸送の計算式は強化学習における価値関数の更新式と共通点があり、強化学習で使われているテクニックを使って光輸送計算の高速化をすることができる[8]。このような技術は現在の映画作品やゲームなどの高品質なレンダリングで使われている。

なぜニューラルネットワークで高速化できるのか

シミュレーションの求解にニューラルネットワークを使うことでなぜ高速化できるのか。

1つ目はシミュレーションが扱う問題セットは実際起き得る現象のほんの一部であり、そのような限られた領域に対する関数近似はニューラルネットワークが最も得意とする分野であるためだ。このような領域の一部では解は入力変化に対し一種の滑らかさをもっており、少数の訓練サンプルで汎化することが期待できる。空間、時間、処理方向に積分操作がある場合でもその結果が一種の滑らかさを持っているのであれば、積分を直接扱わず関数近似で高い精度で近似することができる。シミュレーションでも、計算量が大きい一部分だけ抜き出しそこだけニューラルネットワークで高速に近似すれば、シミュレーションが持つより強力な汎化性能も期待できる。ニューラルネットワークはいまだに外挿問題に弱いが、常に成り立つ不変性や制約（例えばハミルトニアン力学）を組み込めば、より多くの問題を外挿できるように汎化性能を発揮できる。

2つ目はニューラルネットワークの計算は計算の粒度が大きく（大きな行列、テンソルの積や畳み込み操作を繰り返す）、計算バス毎の計算コストの偏りもほとんどない。そのため、現在のGPUやTPUなどの並列計算機の実行効率を上げやすい。また、今後もニューラルネットワーク計算に最適化されたアクセラレーターが登場することが期待できる。

今後のニューラルネットワークのシミュレーターの問題は、より複雑な問題を扱えるようモデリングをどのように自動化するか（例えば物理シミュレーションの環境設定）、事前知識や法則をいかにニューラルネットワークに埋め込んでいけるかが重要となるだろう。

1) S. Greydanus et al.,"Hamiltonian Neural Networks," NeurIPS 2019.
2) P. Toth and et al.,"Hamiltonian Generative Networks," ICLR2020.
3) M. Lutter et al.,"Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning," ICLR 2019.
4) N. Gruver et al.,"Deconstructing the Inductive Biases of Hamiltonian Neural Networks," ICLR 2022.
5) B. Kim et al.,"Deep Fluids: A Generative Network for Parameterized Fluid Simulations," Eurographics 2019.
6) J. S. Smith et al.,"ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost," Chemical Science, 2017, vol.8., pp.3192-3203.
7) K. T. Schütt et al.,"SchNet: A continuous-filter convolutional neural network for modeling quantum interactions," NeurIPS 2017.
8) S. Kallweit et al.,"Deep Scattering: Rendering Atmospheric Clouds with Radiance-Predicting Neural Networks," SIGGRAPH Asia 2017.
9) K. Dahm et al.,"Learning Light Transport the Reinforced Way," https://arxiv.org/pdf/1701.07403.pdf

（2020年3月号掲載の記事に加筆）

シミュレーションに基づく推論：観測からパラメータを帰納的に推定する

多くの科学領域でシミュレーションが使われるようになっている。例えば、素粒子物理、分子動力学、流体力学、疫学、経済学、気象モデル、生態学、機械工学などである。これらの分野で使われるシミュレーションは、対象問題を忠実に再現できるようになっているが、推論、つまり観測からその現象を記述するパラメータや、観測できない潜在パラメータを推定する逆問題に適用することが難しい場合がある。しかし、近年登場したニューラルネットワークによる代理関数、確率的プログラミングフレームワークと自動微分フレームワークによって状況は大きく変わり、これまで推論が難しかった場合でもできるようになってきている[1]。本稿ではそもそも推論とはどのようなタスクか、なぜ難しかったか、どのようにこれらが解決できるかについて解説する。

本稿ではシミュレータは以下のような単純化された計算モデルとする。はじめに、シミュレータは、その挙動を特徴づける入力ベクトルθを受け取り、次に直接観測できない内部状態$z_i \sim p_i(z_i|\theta, z_{<i})$を順に生成し、最後に観測量をこれらに条件付けて生成する$x \sim p(x|\theta, z)$。例えば、物理系のシミュレーションではθ（の一部）はハミルトニアンの係数であり、zは物体の内部状態や環境との相互作用の状態、観測はある時刻の物体の位置などとなる。シミュレーションの各ステップが確率的な場合、シミュレータは生成までのステップを記述した確率的プログラムといえる。

推論は、与えられた観測xからパラメータθや途中の内部状態z_iを推定することである。確率モデルでいえば、$p(\theta|x), p(z_i|x)$となり、いわゆる事後確率分布である。例えば気象モデルのシミュレータをもっていて、現在の観測（地表の各点での気温や湿度）を元に気象モデルのシミュレータの未知のパラメータ（各グリッドの湿度や密度、風速、境界条件など）を求める場合はこの推論が必要になる。物理や化学などで実験や観測データから未知のパラメータを推定することはよく行われる。データ同化（この場合、確率モデルを使わない場合も多い）やパラメータ同定と呼ばれるタスクの一般化である。

有名なベイズの公式を使えば、これらの推論は

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta')p(\theta')d\theta'}$$

として求められる（$p(z_i|x)$も同様）。このように尤度$p(x|\theta)$が求められ、また、パラメータについて積分をとることができれば推論はできる。しかし、ほとんどのシミュレータではそもそも尤度計算が難しい。

シミュレータの尤度計算が難しい理由は途中の潜在変数を周辺化しなければいけないためである。

$$p(x|\theta) = \int p(x, z|\theta)dz$$

尤度が求められないシミュレーションを使って、観測から観測できないパラメータを推論することをlikelihood-free inferenceと呼ぶ。このような問題の場合に最も広く使われている手法は近似ベイズ計算（ABC：approximate Bayesian computation）である。最も単純な形では、ABCは最初に初期値θ'を事前分布からサンプリングし、次にシミュレーションを実行し、擬似的な観測x'（以下サンプル）を得る。そしてこのサンプルが観測xと設定した距離関数Dにおいて近ければ（$D(x, x') < \epsilon$）このθ'を事後確率分布からのサンプルとみなす。この許容条件ϵが0に近づけば近づくほど、このABCによるサンプリングは真の事後確率分布からのサンプルに近づく。しかし、ランダムに試した結果がたまたま観測に一致するというのは非常に確率が小さく、観測が高次元であれば、必要なサンプル数は実現不可能なぐらいに大きくなってしまう。

近年この問題がシミュレーションベースの推論として大きく改善されてきた。サンプル効率、推論の品質、償却化（推論時で同じモデルを共有し、推論1回当たりのコストが小さくなる）が達成できる。

ニューラルネットワークによる代理関数

ニューラルネットを使った強力な生成モデルを使って、シミュレータの生成分布$p(x|\theta)$や事後確率分布$p(\theta|x)$を直接モデル化することができる。特に正規化フローは、効率的にサンプリングでき、かつ尤度を正確に評価できる。元の関

数と同じような挙動をとりながら扱いやすい特徴をもった関数は代理関数（Surrogate function）と呼ばれる。この代理関数は、潜在モデルにとって興味の無い潜在変数の周辺化を暗黙的に実現する。

さらに、微分可能な代理関数を使った場合、尤度だけでなく潜在変数やパラメータについての勾配を効率的に求めることができる。これらの勾配情報は指数型分布が対象の場合は十分統計量であり、また勾配情報を使って、最も情報が得られるような位置をシミュレータでサンプリングする、いわゆる能動学習を実現することができる。

異なるパラメータθ, θ'を使った場合の尤度比$p(x,z|\theta)/p(x,z|\theta')$のSurrogate関数はこれらのパラメータを使って生成したサンプルを分類するような学習となる。これらのサンプルを最適に分類するモデルは尤度比と一致するためである。これは敵対的生成モデルの識別器と一緒である。

代理モデルを利用することで推論を償却化することができる。つまり新しい観測データが得られた時でも、また再度モデルを学習する必要はなく既に他の観測データを使って学習してあるモデルを再利用して推論することができ、効率的に推論することができる。

代理関数の適用先として尤度、事後確率分布、尤度比の3つがある。推論が目的であれば事後確率分布を学習すれば良いように思えるが、（事後確率分布の中に含まれる）事前確率分布が固定になってしまう問題がある。後で事前確率を変えたい場合は尤度をモデル化しておき、後で事前確率分布を変えられるようにしておくほうがよい。また尤度比の学習は分類と同じようにでき、分布自体を学習する生成問題に比べて安定的に学習できるメリットがある。尤度比だけが必要ならば尤度比を直接モデル化するのが良い。

確率的/自動微分プログラミングフレームワークの利用

ここまではシミュレータ自体は変更せず、そのサンプルを利用して効率的に推論を行う手法であった。もし、シミュレータ自身も変えることができればより効率的にすることができる。1つは確率的プログラミングフレームワークの利用である。確率的プログラミングでは、従来のシミュレーションのコードの一部をフレームワークを使って書き直すと、特定の変数を周辺化したり、事後確率分布に従ってサンプリングする操作を自動的に実行してくれる。確率的プログラミングを利用することでシミュレータ毎に煩雑な周辺化や推論、効率的なサンプリングのコードを書く必要がなくなる。例えば、ABCにおいても、確率的プログラミングフレームワークを使うことで内部でMCMCなどを使って観測データと一致しそうなサンプルをサンプリングするようにでき、サンプリング効率を劇的に改善できる。

もう1つが自動微分を備えたフレームワークを利用することで、パラメータや途中変数など任意の変数についての勾配を効率的に求めることができる。微分可能でない操作がある場合（離散化、微分不可能な関数など）も、それらを緩和して微分可能にする手法を使える。これらができれば勾配を計算する効率的な推論を実現できる。

シミュレータ適用範囲は拡大する

ニューラルネットを使って高次元データを直接扱えるようになり、また正規化フローなど直接尤度を評価でき、かつサンプリングもできるモデルを代理関数として使うことで、計算量的に困難だった周辺化を行った結果を直接推定できるようになった。これらのモデルは任意の変数についての勾配情報を効率的にモデル化できる。そして何よりディープラーニングフレームワークや、確率的プログラミングフレームワークなど、プログラミング環境が整い、モデルを記述することが容易になった。こうしたフレームワークを使うことでGPUや大規模並列化環境などを利用することも容易となった。多くの科学領域で既にこれらの利用が進んでいる。

シミュレーションのパラメータを、観測データから帰納的に推定することが容易となり、シミュレータの適用が広がると考えられる。

1) K. Crammer et al.,"The frontier of simulation-based inference," PNAS 2020.

（2020年8月号掲載の記事に加筆）

13-3 深層学習を使った物理シミュレーションの高速化

様々な問題の物理シミュレーションは、シミュレーション技術の改良や計算機の性能向上により実応用が進んできたものの、まだ精度や速度面で困難な場合が多くある。

物理シミュレーションが本質的に難しい理由の一つとして、物理世界の情報（空間や時間、内部状態）が連続量であり、それらの上での微積分などが必要なのに対し、計算機上で扱えるのは離散化された情報だけであり微積分もほとんどの場合は解析的に解けないため数値解析で近似的に解く必要があることが挙げられる。

例えば空間をグリッドに分割して近似した場合、グリッドの解像度を上げることで精度を改善できるが、計算量は解像度の2乗や3乗のオーダーで増えていく。さらに厄介なのが、多くの問題で微視的な現象が巨視的な現象に大きな影響を与える場合があり、シミュレーションで扱う最小スケールと最大スケール間が数百万倍から数億倍も異なるマルチスケールな問題を扱う必要があることである。

例えば翼の流体シミュレーションにおいては、2mの翼の周辺におこる乱流をシミュレーションする際にはスケールが10^{-6}m（熱に変換する直前の最小のスケールでコルモゴロフスケールと呼ばれる）の渦を扱う必要がある。これらを無視したり近似してしまうと全体の系が収束しなかったり違う結果が得られてしまう。また、分子の挙動を分子動力学計算で計算する場合は1ステップが原子の固有振動を扱えるような10^{-15}（フェムト）秒といった時間を扱いつつ、10^{-6}（マイクロ）から10^{-3}（ミリ）秒といった単位での変化をシミュレーションしたい場合がある。

深層学習を使ってシミュレーションを高速化する

こうしたギャップを減らすため、機械学習、特に深層学習（ディープラーニング）を使ってシミュレーションを高速化すると、同じ精度の結果を得るのに数百倍から場合によって数十億倍[1]と劇的に高速化できることがわかってきた。基本的に計算量と精度はトレードオフなため、今までと同じ計算量をかければ精度向上もできる。深層学習を使ったシミュレーションの学習は、計算コストをかけさえすれば学習データを得られる。遅いが高精度な結果が得られるシミュレーション（例えば解像度を高くしたモデル）で大量の学習データを生成し、低精度なシミュレーション結果から高精度なシミュレーション結果を推定する深層学習モデルを学習して構築する。問題設定としては低解像度の画像から高解像度の画像を推定する超解像度の学習にとても近い。

また、深層学習を使うことでGPUやTPUなどのアクセラレーターに最適化されたライブラリを利用でき、実行効率の高いシミュレーションを作ることができる。既存シミュレーターの並列化やアクセラレーターを使った高速化は進んでいるものの、並列化できない部分がボトルネックになってしまうというアムダールの法則にあるように、性能改善は簡単ではない。深層学習を使うことでCNNや行列演算など最適化されたモジュールを使うことができ、高い実行効率のシミュレーターを作ることができる。

深層学習の利用方法には大きく2つのアプローチがある。1つ目のアプローチではシミュレーション全体を深層学習のモデルに置き換える。問題の事前知識はネットアーキテクチャ設計に多少組み込むが、全体はブラックボックスなモデルで扱う。この場合、ニューラルネットワークは定数時間で処理できるということもあり、劇的な高速化が達成されるが、モデルに物理世界の事前知識や制約を入れ込むことは簡単でないため、訓練データ外に汎化することは難しい。一方、このシミュレーターを使おうとする問題設定を十分カバレッジできる訓練データを作れる場合には有効なアプローチである。

このアプローチは単に深層学習がシミュレーション結果をまるごと暗記し、実行時にはそれを参照しているだけとみなすこともできる。一方で高次元入力の結果をそのまま記録し実行時に参照することは容量的にも速度面でも困難である。深層学習はシミュレーションの大量の入出力の関係をそのニューラルネットワーク内のパラメータで効率的に記憶しておくことができ、驚異的な内挿能力によって近い値が訓練事例に含まれていれば高い性能で予測できる。似た解は少ないオーバーヘッドで保存することができる。もし大量のシミュレーション結果で学習しておけば代表的な解を記憶できる。

2つ目のアプローチは、シミュレーター自体は既存の物理知識に基づいて作られたモデルを使う一方で、離散化や数値解析のところで生じる誤差の補正項や係数を学習によって決定するものだ。近似の仕方、修正の仕方を人間が設計するのではなくデータから学習しているともみなせる。このアプローチでは物理知識や制約を組み込んで使うことができるため学習データ外にも汎化し、様々な保存則（エネルギー保存則など）も守られるため、長時間シミュレーションでも結果が発散しないという大きな強みがある。今回はこの2つ目のアプローチについて紹介していく。

流体シミュレーションの高速化

流体シミュレーションは非線形偏微分方程式であるナビエ・ストークス方程式で記述される。流体シミュレーションは気象予報、車、飛行機、エンジン、タービン設計など産業上重要なタスクである一方、シミュレーションが難しい代表的な問題であり、そもそもナビエ・ストークス方程式で解が存在するかを示すことさえ未解決な問題だ。最新の流体シミュレーターではメッシュサイズが1億近くに達しているもののまだ精度に課題がある。

米グーグルの研究チームは既存の流体シミュレーションを深層学習を使って補正することで、より小さな解像度を使っても同じ精度を達成することができ、結果として同じシミュレーション精度で40倍から80倍近く高速化できることを示した[2]。具体的には、既存の有限体積法で離散化した際の連続係数と速度場の補正量を推定する。

解像度が高い時のシミュレーション結果を学習目標とし、解像度が低いシミュレーションがその結果を出力できるよう、補間と修正量を学習する。この学習は既存の有限体積法の解を求めるソルバをJAX（GPUやTPU対応の数値計算ライブラリ）で書き直して微分可能とし、誤差が小さくなるような補間や補正を出力するよう学習する。

既存シミュレーターの補間と補正で実現されていたため、機械学習で懸念されるデータ分布外への汎化能力もあり、異なるサイズやレイノルズ数（慣性力と粘性力の比で流れの特徴を特徴づける）への汎化が大きいことがわかっている。

これと同様のアプローチとしてドイツTechnical University of MunichやフランスTelecom Paris、米Columbia Universityのグループ[3]では流体シミュレーションの離散化誤差をCNNを使って直接修正するアプローチで取り組んだ。従来手法では修正した結果が将来のシミュレーション結果にどのように影響を与えるかというのは制御することができなかったが、微分可能な流体シミュレーターを利用することで、修正した結果による将来の影響を見た上で、その勾配を計算できるようにした。このような微分可能シミュレーターを使うことで、修正精度、汎化性能を大きく改善することができたと報告している。

既存シミュレーターを必要最低限補うようにして学習

フランスConservatoire National des Arts et Métiers（国立工芸院）、フランスSorbonne Universitéらのグループは、部分的にはモデル化できているが、残りが未知であるような問題に対するシミュレーションを行う際、ダイナミクスがわかっていて推定するF_pとブラックボックスなモデルで推定するF_aに分け、ブラックボックスなF_aが必要最低限な分だけ補うよう、そのノルムが小さくなるという正則化を加えて学習する手法を提案した[4]。F_aのノルムは、評価位置の関数値の二乗和を利用している。

解析動力学などで、既知ダイナミクスのシミュレーションを学習した場合はfittingに限界があるが、この手法はよりfittingする上に汎化するように学習することができた。

深層学習を使ったシミュレーターはどこまで汎化するか

非学習ベースのシミュレーションを使った場合はかなり広い範囲の入力で正しい結果を出すことが保証できる。一方で学習ベースのシミュレーターでは常に学習データ外への汎化が問題となる。

しかし、もしシミュレーション対象が部分問題に分解でき、全体の解がそれらの組み合わせで表現できる場合には従来考えられている場合よりも遥かに汎化できる可能性があることが示されている。特にMLPでなく、GNNのようなモデルを使い、ネットアーキテクチャや非線形関数などが実際の問題で扱うモデルと一致している時は外挿できることが示されている[5]。今後、各問題をよく理解した上で深層学習をうまく使えば、様々な問題でのシミュレーションの高速化、精度改善に利用できると考えられる。

---

1) M. F. Kasim et al.,"Building high accuracy emulators for scientific simulations with deep neural architecture search," https://arxiv.org/abs/2001.08055
2) D. Kochkov et al.,"MachineLearning Accelerated Computational Fluid Dynamics," https://arxiv.org/abs/2102.01010
3) K. Um and et al., "Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers,"

NeuRIPS 2020.
4) V. Le Guen et al.,"Augmenting Physical Models with Deep Networks for Complex Dynamics Forecasting," ICLR 2021.
5) K. Xu and et al., "How Neural Networks Extrapolate: From Feedforward to Graph Neural Networks," ICLR 2021.

（2021年4月号掲載の記事に加筆）

13-4

AIを使った汎用原子レベルシミュレーター Matlantis

2021年7月6日、Preferred NetworksとENEOSの共同出資で設立されたPreferred Computational Chemistry（略称PFCC、筆者が代表を務めている）は汎用原子レベルシミュレーターMatlantis[1]をクラウドサービスとして提供開始したことを発表した。原子スケールのシミュレーションを従来のシミュレーションと比べて精度を保ったまま10万倍から数千万倍高速化したほか、55種類の元素をサポートし未知の分子や結晶など未知の材料に対してもシミュレーションできる汎用性を兼ね備えている。この元になっているのが深層学習（ディープラーニング）を使ったNNP（ニューラルネットワークポテンシャル）である。

本稿では、現代における材料探索の重要性を説明した後、NNPやMatlantisの技術、現在の課題と将来の展望について解説する。

材料探索は持続可能な社会実現の鍵となっている

これまで新しい材料や素材は新しい生活や産業を切り開いてきた。代表例として窒素固定による農業生産革命、鉄鋼や炭素繊維を使った自動車や飛行機、半導体による計算機の実現が挙げられる。ここでは新しい触媒の発見が持続可能な社会を実現する上で重要になっていることを説明する[2]。

近年、地球温暖化対策として、自然エネルギー由来の再生可能エネルギーの普及が重要とされている。しかし、太陽光や風など自然エネルギーを使った再生可能エネルギーは、天候に依存し生成量にばらつきがあるため、再生可能エネルギーを基幹エネルギーとして利用するには莫大な量のエネルギーを貯蔵する仕組みを作る必要がある。

例えば米国で曇りや風の弱い日が1週間続いた場合、現在の電力の70%と電力以外のエネルギーを賄うには250TWhの貯蔵が必要と試算される。これだけの貯蔵をLiイオン2次電池で実現するには128兆米ドルかかり、揚水発電で実現するにはエリー湖並みの大きさのダム（琵琶湖17個）分の水を汲み上げる必要がありどちらも非現実的である。

そこで現実解として有望視されているのは既存のガスインフラを使える水素貯蔵である。天然ガス向けに作られているインフラを使うことができ、既存インフラを使った場合でも206TWh程度は保存できる。貯蔵した水素から電力を取り出す際は、燃料電池を使う。水素貯蔵の課題はエネルギー変換効率（再生可能エネルギーの余剰発電分を貯め、それを再び交流の電力として取り出すまで）が35%と低いことであり、これは電池による貯蔵時の効率の半分から1/3程度である。この変換効率を向上できるよう、燃料電池向けにより優れた触媒の発見が求められる。さらに現状の燃料電池は触媒に希少な貴金属（IrやPt）を利用しており、高コストであることも問題となっている。

このように有望な触媒の発見が再生可能エネルギーを基幹エネルギーとして利用する上で重要であることから、2020年に米Facebook社と米Carnegie Mellon UniversityはOpen Catalyst[3]と呼ばれるプロジェクトを開始し、材料探索用のデータセットを公開するほか、探索に必要な技術のコンテストを開催している。

材料シミュレーターは材料探索と原理解明に重要

こうした材料・素材開発を進める上で、以前よりシミュレーターの利用が重要だと考えられてきた。シミュレーターには有望な材料を探索し絞り込む役割と、実験結果が得られた時に、そこでどのような現象が起きているのかを明らかにする目的がある。

例えば合金触媒の性能を調べるために60種類の金属から5つ選ぶ場合を考えると、組み合わせ数は800万通りになり、それらの比率や組み合わせ方まで含めるとさらに膨大な数となる。実験でこれだけ膨大な材料を試行することは不可能であるため、シミュレーターを使って有望な材料を絞り込み、その後に実験をして探索することが期待されている。

また、実験結果が得られても具体的に何が起きているのかまでは分からないため、改善するためにシミュレーターを使って材料上で何が起きているのかを調べることは重要である。例えば前述した農業生産革命をもたらしたアンモニアの触媒合成であるハーバー・ボッシュ法は1906年に発見され世界中で使われている。その一方で反応としては

$N_2 + 3H_2 \rightarrow 2NH_3$ と単純であるが、未だに何が起きているのかの全容解明はできていない。Karoliina Honkala氏らが2005年に、この反応が金属触媒上で少なくとも12段階の反応を経由して進み、各反応が金属原子の多孔質の配置に依存することを後述するDFT（density functional theory）によるシミュレーションで明らかにしたことは画期的であった。現在世界中の研究者がシミュレーターと実験を組み合わせて材料開発を進めている。

量子力学で記述される原子レベルシミュレーション

次に原子レベルシミュレーションに使われている技術について概説する。原子や分子の挙動においては特に電子が重要な役割を果たす。この電子の挙動はマクロな世界で成り立つ古典力学では記述できず、量子力学で記述する必要がある。量子力学は私達が通常観測する古典力学とは異なる性質がみられる。例えば系の状態が決定論的でなく確率分布によって表現され、エネルギーは離散的な値しかとらず、粒子と波の両方の性質を持ち、電子などフェルミ粒子は2つ以上の粒子が同一の量子状態を占めないというパウリの排他原理に従うなどだ。古典的粒子がニュートンの運動方程式に従うように、量子的粒子が従うのは量子力学を支配する次のシュレディンガー方程式（時間非依存の場合）である。

$$\hat{H}\Psi(\mathbf{r}) = E\Psi(\mathbf{r})$$

ここで $\mathbf{r}$ は各粒子の位置を表し、N 個の粒子からなる場合 $3N$ 次元からなる。$\Psi(\mathbf{r})$ は波動関数と呼ばれ粒子系の状態を表し、$|\Psi(\mathbf{r})|^2$ は粒子系がその状態をとる確率を表す。$\hat{H}$ はハミルトニアン演算子と呼ばれ、系の全エネルギーを表し、E がエネルギーである。固有値問題の形をとり、ハミルトニアンの固有関数が $\Psi(\mathbf{r})$ 、固有値がその時のエネルギーに対応する。

この方程式を解くことで波動関数やエネルギー、力を求められるが、解くことは難しい。本質的に難しい理由は、この問題が多体問題、つまり互いに相互作用する3体以上を含む系の状態を求める問題であるためである。そのため、解析解は一番小さい水素原子から成る単純な系しか得られない。さらに問題として興味のある系のサイズは数百から数万といった原子を含む分子であり、電子数はこの数十倍のオーダーとなり、近似解ですら現実的な時間で解くことは難しい。

この問題に対し、1964、1965年にWalter Kohn氏らが電子が基底状態にある場合は、$3N$ 次元の波動関数を直接求める代わりに、ずっと次元数の少ない3次元の電子密度を介して求められることを示し、汎関数（関数を引数にとる関数）の変分法による最適化問題を解くことでこれが得られることを示した。これを密度汎関数理論法（DFT：density functional theory）と呼ぶ。DFTはその後の手法改良やJohn Pople氏らが開発したGaussianなど優れたソフトウエアが登場したことで広く利用されるようになり、現在は多くのシミュレーションで中心的に使われている。Kohn氏とPople氏はこの功績により1998年にノーベル化学賞を受賞している。

しかし、DFTは計算量を劇的に減らしたとはいえ、計算量は原子数の3乗程度のオーダーで増加し、最近のスーパーコンピュータを使ったとしても数千の原子を扱うには数カ月計算が必要だった。

NNP：ニューラルネットワークポテンシャル

こうした計算量の問題を回避するため、シュレディンガー方程式を解かずにポテンシャルを求めそれに従って分子動力学法でシミュレーションする方法が研究されてきた。しかし、精度と汎用性の面で問題があった。こうした問題を解決するためにニューラルネットワークポテンシャル（NNP）が提唱された。NNPは各原子の位置を入力すると、エネルギーを回帰するニューラルネットワークである。NNPは1990年台後半に提唱された後、近年の深層学習の進展に伴い再度注目されていた。NNPはエネルギーを推定できるだけでなく、エネルギーの原子の位置についての勾配を誤差逆伝播法で求めることで、各原子にかかる力を求めることができ（ポテンシャルエネルギーの移動当たりの変化量が力に相当する）、それを使ってシミュレーションすることができる。力を求めた後は通常の古典力学に従った分子動力学法シミュレーションを行えばよい。このほか、様々な物性値を推定できることがわかっている。

NNPは時間のかかる電子状態の緩和計算について行うことなく、ニューラルネットワークで入力から出力をフィードフォワード処理で求められるので高速であり、かつ原子数の増加に対しても計算量の増加は低く（線形から2乗のオーダーの間）、高速に処理することができる。

NNPは既存のシミュレーターを使って学習データを生成し、それを使って教師あり学習する。高精度なNNPを実現する上で重要な点が2つある。1つ目は学習させるデータセッ

トである。従来のDFTを使って様々な条件下でのシミュレーションを行って結果を作り、エネルギーや力を求めておき、これらを使ってNNPを学習させる。どのようなデータを用意するかで性能が大きく変わってくる。

2つ目はネットワークアーキテクチャである。原子の位置からエネルギーを求める際に、原子位置に対する回転、平行移動、鏡像変換に対して不変であることが要請される。こうした不変性をネットワークに埋め込んでおくことで高い汎化性能を達成できる。さらに3つ以上の原子間の相対位置や角度が重要であり、こうした情報を扱えるようになっている必要がある。

10万日分のGPU時間で学習したPFP

今回、PFCCが公開した原子レベルシミュレーターMatlantisはPFP[4]と呼ばれるNNPを利用している。このPFPがどのような学習データとネットワークを使っているかについて説明しよう。

学習データは55種類の原子からなるデータセットを作成し、1台のGPUで延べ10万日分、すなわち約273年分のシミュレーション時間をかけて作られている（2022年6月時点で1000年分のシミュレーション時間まで増えている）。これはAI研究でも大規模な学習として知られるGPT-3が13万日分の計算をかけているのに匹敵する計算量である。

量が大きいだけでなく、特徴的な点として従来のNNPのデータセットは既知の構造や安定構造を使ってデータ生成しているのに対し、PFPのデータセットでは高温、高圧下で非安定構造など非常に広い条件でデータを作っている。ニューラルネットワークは内挿能力が優れており、現実的ではないような条件のデータも加えることで従来と比べてずっと高い汎化性能を達成することを目指した。

次にネットワークアーキテクチャの工夫について説明する。PFNではアーキテクチャを各コンポーネントから独自開発している。先程の不変性や複数原子情報は全て兼ね備えた上で、内部情報として高階のテンソル量を保持し畳み込み層を利用することで原子間の高次の情報を扱えるようになっている。例えばπ結合のような向きを持った角度情報などを表現することができる。これにより、同じ学習データセットを使って既存のネットワークと比較した場合でも高い性能を達成している。先程のOpen Catalystコンペティションで現時点のリーダーボードで1位の手法（絶対エネルギー誤差 0.261）[5]より2桁少ない学習データで、より高い性能（同 0.226 validation dataset）を達成している。

このようにデータセットとアーキテクチャを改良することで高い汎用性と精度を達成している。例えば、既に水素と二酸化炭素から炭化水素を作る上で重要な触媒の探索、ナノサイズの微小細孔を持つ金属有機構造体材料における水の吸着と拡散、潤滑油の粘度計算、全固体Liイオン2次電池のLiイオン拡散係数を求めるタスクなどでMatlantisのPFPを使った事例がある。

また、信州大学 教授の吉田通久氏らのグループ[6]は自動車の燃料電池向け合金触媒を開発版のMatlantisを使ってシミュレーションし、従来は2000~3000コアを使って数カ月で1つの結果が得られたのを0.1秒で実行できたと報告している。

こうしたことでこれまで不可能だったような数千や数万種類の材料を網羅的に探索したり、より複雑な系や長い時間のシミュレーションを行うことで、材料探索が加速されることが期待される。

可能性と今後の課題

最後に、現在のDFTやその特徴を引き継ぐNNP、PFPにどのような課題があるのか、発展性があるのかについて述べる。

DFTは理想的な汎関数を使えば厳密に正しい解を与えるが、そのような汎関数はまだ見つかっておらず、それを近似した汎関数を利用する（この違いの部分は交換相関汎関数と呼ばれる）。どの交換相関汎関数を使うかで特徴が異なり、苦手とする系では得られた解と実験結果にギャップが起きる。

例えば、よく交換相関汎関数として使われるLDAやGGAは強相関電子系と呼ばれる局在電子を有する物質群（酸化鉄など）を扱うことを苦手としている。また、DFTは電子が基底状態にある場合に対する定式化であり、発光現象など励起状態を扱うことはそのままではできない。また分散力（ファンデルワールス力）など非定常な分極が引き起こす相互作用を扱うことも苦手である。

DFTの創業への応用も期待されているが課題が残されている。創薬では、ターゲットとなるタンパク質により強く結合する化合物を探索する必要があり、結合自由エネルギーの予測が重要となり、DFTを使ってより高精度な予測ができることが期待されている。一方でタンパク質や化合物の原子数は非常に大きく、また自由エネルギーを求めるには長時間かつ

大量のシミュレーションを行う必要がある。

さらに、シュレディンガー方程式の波動関数を直接ニューラルネットワークを使って求める試みも進んでいる。例えばFermiNet[7]は波動関数を直接ニューラルネットワークでモデル化（試行関数Ansantzと呼ばれる）する。この際、2つの電子を交換すると波動関数の符号が逆転するというパウリの排他原理を満たすようにニューラルネットワークを設計し、変分モンテカルロ法で最適化することで波動関数を求める。実験で得られる限界と同じ化学的精度（1kcal/molの誤差）を得ることができたが、小さい分子（30電子からなる場合、ビシクロブタン）の波動関数を求めるだけでGPUで1000時間かかる問題がある[8]。

本稿ではAIを使って原子レベルシミュレーションがどのように進化してきたかを紹介した。今後もさらなる計算機の性能向上やAI技術の進化によって新たな材料や素材の発見が加速されると期待される。

---

1) https://matlantis.com/
2) C. Lawrence Zitnick et al.,"An Introduction to Electrocatalyst Design using Machine Learning for Renewable Energy Storage," https://arxiv.org/abs/2010.09435
3) https://opencatalystproject.org/
4) S. Takamoto et al.,"Towards Universal Neural Network Potential for Material Discovery Applicable to Arbitrary Combination of 45 Elements," Nature Communications 2022.
5) https://opencatalystproject.org/leaderboard_s2ef.html
6) G. V. Huerta et al., "Calculations of Real-System Nanoparticles Using Universal Neural Network Potential PFP," https://arxiv.org/abs/2107.00963
7) D. Pfau et al., "Ab initio solution of the many-electron Schrödinger equation with deep neural networks," Phys. Rev. Research.2020
8) D. A. Rufa et al.,"Towards chemical accuracy for alchemical free energy calculations with hybrid physics-based machine learning/molecular mechanics potentials", https://www.biorxiv.org/content/10.1101/2020.07.29.227959v1

（2021年9月号掲載の記事に加筆）

第14章

ゲーム

14-1

AlphaGo：CNNと強化学習を組み合わせたコンピュータ囲碁

英DeepMind社の創業者であるDemis Hassabis氏が2015年末、「囲碁でもすごい結果を近々公表する」と話していた通り、AlphaGoと呼ばれるコンピュータ囲碁プログラムがヨーロッパチャンピオンを破ったと話題になった。2016年3月には、世界的なトップのプロ棋士との対戦を控えている。

今後、コンピュータ囲碁が強くなっていくことを考えると、今回の対戦が人間のトップとの最後の対決になる可能性もある。コンピュータ囲碁研究者が集うメーリングリストでAlphaGoの成果を紹介したスレッドのタイトルが「Game Over」であったことが印象的であった。

プロ棋士との差が大きかったコンピュータ囲碁

囲碁は、探索空間が広く（候補手の平均数は250、深さは150）、盤面評価をするための特徴設計、評価関数設計が難しいことともあり、チェスや将棋と比べてコンピュータ囲碁は弱かった。

その中でもモンテカルロ探索を使った手法が近年注目されていた。これはある盤面からシミュレーションで大量に対戦させ（この対戦をロールアウトと呼ぶ）、その勝敗で盤面評価する手法である。しかし、プロ棋士との差は大きく、追い付くにはまだ数年かかると思われていた中、AlphaGoが一気に追い付いた。

AlphaGoはニューラルネットワークを使い、強化学習を使って学習することで盤面評価を正確に行う[1]。入力を19×19（盤面全体）の画像とみなし、CNN（畳み込みニューラルネット）を利用し、盤面を表現した上で、次の手予測や盤面評価を行う。

AlphaGoは次の4種類のネットワークを学習し、利用する。

- 強い人による次の手を予測する、正確だが遅いネットワーク p_σ
- 強い人による次の手を予測する、不正確だが高速なネットワーク p_π
- 盤面が与えられた時、最も勝つ可能性の高い手を予測するネットワーク p_σ
- 盤面が与えられた時、どちらが勝つかを予測するネットワーク v_θ

教師あり学習や強化学習を活用

これらのネットワークをどのように学習するかを順に説明する。

はじめに、強い棋士の指し手をまねるように次の手を予測するニューラルネットワークp_σを作る。この学習には、オンライン囲碁サイトKGSの囲碁の対戦記録のうち、6段から9段にランクされている強い人の対戦記録16万局から3000万手を利用した。教師あり学習を利用し、精度は55.4％であり、これまでの予測の最高精度の44.4％を10％近く改善している。この予測精度の差は最終的な強さに大きく影響する。

また、ロールアウト用に、線形識別器を使った精度は悪いものの高速な予測器p_πをTygemサーバの800万局面を利用して作る。p_πの予測精度は24.2％だが、p_πが3msかかるのに対し、p_πは2μsで予測できる。

次に、どの手を打てば最終的に勝つかを予測するp_σを強化学習で学習する。初期値にはp_σを利用し、少なくとも強い人の手を予測できる程度に強い状態から始める。どの手を打つかの行動選択を確率分布とした時に、その確率分布を勝率を上げる方向にパラメータを更新する。

先ほどの教師あり学習は、強い人なら次にどの手を打つのかを予測していたのに対し、この強化学習では最終的に勝つためにはどの手を打つべきかを予測しており、より直接的に勝てる手を学習しているのに注意されたい。

p_σの時点で、オープンソースの囲碁ソフトで最強であるPachiに85％で勝利する。教師あり学習の結果を使った場合の勝率は11％であることから、単なる次の手予測ではなく、最終的に勝てる手を指せるように強化学習をすることが重要であることが分かる。

最後に、p_σを使い、与えられた盤面でどちらが勝つのかを盤面評価する予測器v_θを作る。この盤面評価の学習のために、強化学習で得られた強いp_σを利用し3000万局の対戦をさせ、この結果を利用して盤面評価を学習する。

50個のGPUで1カ月近く分散学習

これで4種類のネットワークができた。対戦時にはこれらのネットワークを次のように利用する。

現在の盤面から従来手法と同様に探索木を展開し、候補手を探す。この各手の展開は最も強い強化学習で得られたp_ρではなく、人の手をまねたp_σを利用する。これは、人間の手の方が多様性があり、結果として探索する範囲が広がり、強くなるためである。

次に、探索木の末端（葉）において盤面評価関数v_θと、そこから高速な予測器p_πによるロールアウトの対戦結果を組み合わせて、盤面評価を行う。これを時間が許す限り行い、最も有力な手を選択する。

盤面評価関数v_θ単体でもモンテカルロ探索を使った既存の囲碁プログラムより強いが、この2つを組み合わせることでより強い碁ができる。学習はいずれも50個のGPUを用いた分散学習を用いており、次の手予測が3週間、強化学習による次の手予測が1日、盤面評価に1週間かかっている。

AlphaGoの貢献は次の通りである

(1) 複数のネットワークを組み合わせた新たな探索アルゴリズムを設計した。
(2) ある程度強くなったコンピュータ同士を対戦させ、強化学習をし、正確な盤面評価を実現した。
(3) 50個のGPUで1カ月近くの分散学習を行えば、これらの学習が可能であることを示した。

また、細部ではDeepMind社が培っている深層学習や強化学習の成果が随所に使われており、彼らの研究の集大成といえる。次の問題を探しつつ、トップ棋士との対戦を楽しみにしたい。

1) D. Silver, et al.,"Mastering the game of Go with deep neural networks and tree search," Nature, vol.529, 28 January 2016.

（2016年4月号掲載の記事に加筆）

AlphaGo Zero：ゼロから学習で人を超える

英DeepMind社がAlphaGoの改良版であるAlphaGo Zeroの論文を『Nature』誌で発表した[1]。AlphaGoは登場以来急速に強くなり続けている。2015年に登場したAlphaGo Fan（同社はAlphaGoの各バージョンに名前を付けている）はヨーロッパの囲碁チャンピオンであるFan Hui氏を破り、それから半年後に改良されたAlphaGo Leeは世界トップ棋士の1人であるLee Sedol氏を4-1で破った。その後もAlphaGoは強くなり続け、2017年初めごろにネット囲碁対局で登場したAlphaGo Masterは世界中のトップ棋士達に60連勝した。今回のAlphaGo ZeroはそのAlphaGo Masterに100回中89回勝てるほど強い。

AlphaGo Zeroは強いだけでなくこれまでのAlphaGoと大きく違う点がある。それは人の棋譜を真似てから学習するのではなく0から学習した点である。しかも学習を開始してから数日で多くの囲碁の定石を発見した。未知の定石も多く発見し、従来のAlphaGoよりはるかに強くなることができた。

この実現の中心となったのが、自分より少し強い指し方を目標に学習する新しい強化学習手法である。AlphaGo Zeroは1つのネットワーク$f_\theta(s)$を使う。従来のAlphaGoが次の手予測、盤面評価用、探索用と別々のネットワークを使っていたのに対し、AlphaGo Zeroではこれらのモデルを1つに共有することでこれらのタスクに共通して有用な特徴を抽出し学習を効率化できる。このモデルf_θは盤面sが与えられた時、各手を打つ事前確率ベクトル$\boldsymbol{p}$と、その盤面の評価値vのペア$(\boldsymbol{p}, v)=f_\theta(s, a)$を出力する。

次にこのネットワークを使って、モンテカルロ木探索（MCTS）を行う。探索木の各枝(s, a)には、事前確率$P(s, a)$、訪問回数$N(s, a)$、状態行動評価関数$Q(s, a)$を格納しておく。探索は現在の盤面に対応する根から始まり、各節点において次のように計算されるUCB（upper confidence bound）

$$Q(s, a)+U(s, a)$$

が大きい手aを選択し、次の状態s'へ移動する。ただし、Uはその手を探索した回数を$N(s, a)$とした時、$U(s, a) \propto P(s, a)/(1+N(s, a))$で与えられる。また、$Q(s, a)$は現時点のシミュレーションで集められた評価であり、$Q(s, a)=\sum_{s' \mid s, a \to s'} V(s')/N(s, a)$で与えられる。ただし、$s, a \to s'$は盤面$s$で手$a$を選択した後、最終的に盤面$s'$に到達したことを示す。探索が葉に対応する盤面$s_L$まで到達したら、その盤面$s_L$を一度だけ評価し、$(\boldsymbol{p}, s)=f_\theta(s_L)$を得て、各枝に$P(s, a)$を格納する。各枝のUCBは最初$(N(s, a)=0)$は事前確率$P(s, a)$と一致し、探索回数が増えてくると実績値の平均$Q(s, a)$が支配的になってくる。

最終的な手は最も多く探索された手、つまり最も有望な手の分布$\boldsymbol{\pi}(s, a) \propto N(s, a)^{1/\tau}$に従って選択する。この$\tau$は温度パラメーターであり、温度が高いほど一様な手を選ぶように、低いほど探索回数が最大の手のみを選ぶようになる。

この探索をした上で得られた分布$\boldsymbol{\pi}$は元々の$\boldsymbol{p}$よりも強い手を選べるような分布である。なぜなら、$\boldsymbol{p}$は今の盤面だけから評価しているのに対し、$\boldsymbol{\pi}$は実際に探索の過程で各手を試し、より先の状態で評価した結果をまとめた結果だからだ。

今の方策分布$\boldsymbol{p}$を、より強い分布$\boldsymbol{\pi}$に合わせるようにすることでより強い手を打てるようになる。これは$\boldsymbol{\pi}$と$\boldsymbol{p}$のクロスエントロピー損失関数$\boldsymbol{\pi}^T \log \boldsymbol{p}$の最小化で実現できる。

一方、盤面評価の目標はその手を打つことで勝てるかどうかとなる。盤面がsの時に、その時の手番の人が勝った場合$z=1$、負けた場合$z=-1$という確率変数zを用意しこれをvが予測できるように最小二乗法で最適化する。

これら2つを合わせた最終的な目的関数は次の通りである。

$$(\boldsymbol{p}, v)=f_\theta(s)$$
$$l=(z-v)^2-\boldsymbol{\pi}^T \log \boldsymbol{p}+c\|\theta\|^2$$

ただし、最終項$\|\theta\|^2$は正則化項であり、$c>0$は正則化項の強さを決めるハイパーパラメータである。

従来の強化学習ではベルマン方程式を基にしたTD（時間差）誤差を基に目標にしていたり、直接収益を最大化するように方策を最適化する方策勾配法が使われたりしていた。AlphaGo Zeroの新しい学習手法はMCTSによって得られ

たより良い手の分布πと実際の収益zを目標に学習することで安定にかつ効率的に改善することができる。

従来もself-playからの強化学習手法が提案されていたが、学習が安定しない、昔学習した結果を忘れてしまうという問題が存在した。今回の提案手法はこれらの問題はみられなかったと報告されているが、理由の解明は今後の課題である。

最初の実験ではAlphaGo Zeroには小さなモデルを利用し、既存のAlphaGoとの比較を行った。各盤面で手を選ぶためのMCTSは1600回のシミュレーションからなり、それは0.4秒間で実行された。各ミニバッチは2048個のサンプルであり合計で70万回の更新を行った。ニューラルネットワークは20層のResidual Networkを利用した。

AlphaGo Zeroは学習を開始してから36時間後にはAlphaGo Leeの強さに到達し、72時間後には、AlphaGo Leeに対し100勝0敗するほどに強くなった。AlphaGo Leeの学習が数カ月必要だったのと比べると学習が非常に速くなっていることが分かる。また、人の対戦棋譜を予測するように教師あり学習されたモデルと比べ、AlphaGo Zeroは人の打つ手の予測精度は低いことが分かった。これはAlphaGo Zeroが人の指し方とは異なる指し方を学んだ上で強くなっていることを示している。

AlphaGo Zeroは囲碁の最低限のルールのみ（どこに石を置くのがルール上許されるのか）を教えた状態から学習を開始し囲碁の基本的なルールや既知の定石を次々と見つけ、まだ見つかっていない新しい定石を見つけることができた。

学習手法の改良のみならず、盤面評価と指し手の予測モデルを共有することも改善に大きな効果をもたらした（Eloレートで600点）。また、画像認識などで大きく成功しているネットワークであるResidual Networkを使うことも改善に大きく貢献（レートで600点）していることが分かった。

最終的な性能を測るために40層からなる大きなネットワークを作り40日間学習させた。この学習では2900万局対戦を行い、2048個のサンプルで310万回更新を行った。このネットワークはAlphaGo Masterに89勝11敗で大きく勝っていることが分かった。AlphaGoの開発チームはAlphaGoをこれ以上強くする研究開発は終了すると発表している[2]。

人が数千年かけて培ってきた囲碁の技術を数日で再現し上回ったことは大きな意義がある。また、学習アルゴリズム自体を変更することで、学習効率や得られるモデルも劇的に改善できることが示された。最終的に得られたモデルも4つのTPU（Tensor Processor Unit）を搭載した1台のマシンで実行することができ、計算能力の増加以外にも学習手法の改善でまだ大きな伸びしろがあることを示した。

しかし、このAlphaGo Zeroの成果がそのまま他の問題に適用できるわけではない。今回のAlphaGo Zeroが成功した条件がいくつかある。1つ目は囲碁が完全情報ゲームであり情報が全てプレーヤーに提示されている点、2つ目はゼロサム対戦ゲームであり対戦相手も自分と同じモデルを使いシミュレーションできる点、3つ目は環境のシミュレーションができる点である。

多くの問題ではこれらの条件は満たされない。例えばロボットのシミュレーションの場合、必要な情報は全て入手できず環境の一部分しか観測できない。相互作用する環境（例えば人など）も完全に再現できず、シミュレーションが困難である。多くの強化学習分野の研究はこれらの問題への解決に取り組みつつある。

1) D. Silver, et al.,"Mastering the game of Go without human knowledge," Nature, vo.550, pp.345-359.
2) https://www.reddit.com/r/MachineLearning/comments/76xjb5/ama_we_are_david_silver_and_julian_schrittwieser/doiii1s/

（2017年12月号掲載の記事に加筆）

14-3

AlphaStar:多様性のある学習環境で高度なスキルを獲得

2016年、英DeepMind社が開発したAlphaGoが囲碁トップ棋士であるイ・セドル（Lee Sedol氏）氏に4-1で勝利し話題となった。囲碁は途中の盤面種類数が非常に多く、盤面の評価には感覚や大局的な視点が必要でありコンピュータには不得意とされてきた。AlphaGoはこうした見方を打ち破り、純粋にデータとゲームのルールだけからこのような能力を獲得できることを示した。

一方で、コンピュータが人間のトッププレーヤーに及ばないゲームはまだ多く存在している。その代表的なゲームの1つがリアルタイム戦略（Real-Time Strategy：RTS）ゲームであるStarCraft Ⅱ（スタークラフト2）だ。スタークラフト2は他のリアルタイム戦略ゲームと同様に、複数のユニットにリアルタイムに指示を出しゲームを進めていく。労働者ユニットは環境上に散らばった資材を収集し、新しい建物やテクノロジーの開発を進めていく。ゲームに勝利するためにはそれらの建物やテクノロジーを使って、作成した戦闘ユニットを操り相手の陣地を破壊することが目標となる。

こうしたリアルタイム戦略ゲームでは、大局的な戦略に基づいて経済や戦闘を管理するマクロ操作と各ユニットを細かく操作して局地的な戦いに勝利するミクロ操作のバランスが重要となる。前者は戦略シミュレーションゲーム（例えば信長の野望など）に近く、後者はアクションゲームに近い要素が含まれる。

最強のプロゲームプレーヤーに勝利

2019年1月、DeepMind社はこのスタークラフト2において、同社が開発したAlphaStarが最強のゲームプレーヤーの1人であるMaNa（Grzegorz Komincz）氏に勝利したと発表した[1]。このスタークラフト2をコンピュータで解かせる場合の問題は次の3つである。

1つ目にゲームの情報が全て得られない不完全情報ゲームであることである。味方ユニットの周辺しか視野が得られず、相手が何をしているか、行ってみなければわからない。例えば、時には相手陣地に入って情報を得る斥候が必要となる。観測できていない部分については様々な可能性を想定して行動しなければならない。

2つ目に長期計画が必要なことである。1つのゲームは1時間近くに及ぶ場合もあり、状況に応じて戦略を柔軟に変えていく必要がある。ユニット間にはじゃんけんのような強弱関係があり、駆け引きをしながらユニットを生産していく必要がある。状況に応じて、計画を常に修正し続けることが必要となる。さらに、ある時点で行った行動が最終的にどのような効果をもたらすかはかなり時間がたたないと分からず、信用割当問題を難しくする。

3つ目に行動空間が非常に大きいという点である。ゲーム中のユニット数や建物数は最大数百にものぼり、それぞれに命令できる操作種類数も多い。1分当たりの操作数はAlphaStarの場合、280にも上る（ちなみに、MaNa氏の1分当たりの操作回数は390回であった）。

このような難しい点があるにもかかわらずAlphaStarはMaNa氏に5-0で勝利した。MaNa氏は「想像していなかった人間らしいプレースタイルに感動した。自分がどれだけ相手にミスさせるようにプレーし、その反応を利用していたのかに気付かされた。（今回の対戦は）このゲームの新しい方向を示してくれた」と感想を述べている。

それでは、AlphaStarがどのように作られたのかについてみてみよう。AlphaStarはニューラルネットワークによって行動を生成する。このニューラルネットワークは入力として各ゲームユニットとその状態からなるリストを受け取り、行動の列を出力する（ただし、MaNa氏との対戦後に、人のプレーと同様に画面そのものを入力とし、画面範囲内のユニットだけに指示できるバージョンを作成している）。

ニューラルネットワークはTransformerを胴体とし、LSTM、ポインタネットワークが付随した自己回帰モデルによる方策ヘッド、複数エージェント間で共有するベースラインを出力する[2]。ポインタネットワークは注意機構を使って特定の入力をそのまま読み込む機構である。ベースラインは方策勾配で、採用した行動が平均的な動作に比べて良かったかどうかを評価するために必要であり、学習の成功率、速度に大きく寄与する。

最初は人のプレイからの教師あり学習、次はリーグ戦で強化学習

AlphaStarはAlphaGoと同様に最初は人のプレーヤーからそれを真似るように教師あり学習を行った。このように学習されたモデルの強さは人のプレーヤーの"ゴールドレベル"(6段階のレベルの上から4番目)程度であった。

次にこのようにある程度強くなったエージェント同士を対戦させて強化学習をさせていく。ここでは単に1つの強いエージェントを作るのではなく多様性のあるエージェント群を作るようにした。

具体的には仮想リーグ戦を用意し、そこでそれぞれのエージェントに異なる目標や内発的動機を与えて、多様性のあるエージェント群を育てていった。例えば、あるエージェントにはライバルを設定し、それに勝つことを目標とさせたり、あるエージェントは他のエージェント群全体に対する勝率に加えて特定のユニットを多く生産することを目標とした。これにより、エージェントが様々なスキルを身に着けつつ、それらが身に着けるスキルの弱点を他のエージェントが見つけ、それに対応するように成長していくことができている。

この多様性のあるエージェント群は学習の基本問題である破滅的忘却を防ぐことにも役立っている。学習で難しいのが、新しいスキルを身に着けることによって昔身に着けた能力を忘れてしまうという破滅的忘却が起きることである。エージェント群の存在によって、多様なスキルをプールしておくことができる。例えば、エージェントがある弱点を克服したのを忘れたとしても、他のエージェントが再度その弱点を突いてくれることで再度そのスキルを身に着けることができる。

このような多様性のある環境は適度なランダムネスを与えることで学習が局所最適解に陥ることも防いでくれる。ハイパーパラメータ探索やネットワーク探索でも使われている技術をエージェントの学習に適用したと考えられる。学習の過程では、トッププレーヤーが実際使っている様々な戦術が発見された。また、ある戦術が発見され、すぐにそれへの対抗策を他のエージェントが発見することがみられた。

この仮想リーグはグーグルが開発した機械学習アクセラレータチップ「TPU v3」を用いて14日間の学習を行い、各エージェント当たり16個のTPUが使われた。ゲームは並列に実行され、この期間で各エージェントは200年間に相当するプレーを経験する。どの程度の数のエージェント(種類)を学習したかについてDeepMind社からの発表には明確な記述はないが、グラフからは600程度のエージェントが学習されたとみられる(この場合、利用したTPUは5000～10000個程度となる)。最終的に用いるエージェントはナッシュ平均化[2]と呼ばれる操作を適用し、それらのエージェントをそれらの相対的な強さによって重み付けした試行環境上で最も強いエージェントを1つ学習させる。

AlphaStarがこれほど強くなった理由としては最新のモデルや学習手法の利用もあるが、最も大きな貢献はリーグの存在であると考えられる。多数の異なる特徴を備えたエージェント同士を戦わせることで、弱点を見出しそれを潰すとともに、順当に成長してきたエージェントが想像もできないような全く新しいスキルを獲得できる可能性を増やすことができる。今後はモデルや学習手法だけでなく、学習環境の進展が重要になってくると考えられる。

1) "AlphaStar: Mastering the Real-Time Strategy Game StarCraft II", https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/
2) J. N. Foerster, et. al.,"Counterfactual Multi-Agent Policy Gradients," AAAI 2018.
3) D. Balduzzi, et al."Re-evaluating Evaluation," NuerIPS 2018.

(2019年4月号掲載の記事に加筆)

第15章

バイオ・生命科学

15-1

AlphaFold：50年間の生命科学のグランドチャレンジを解く

遺伝子配列からタンパク質の立体構造を決定する問題、いわゆるProtein Folding問題は生命科学におけるグランドチャレンジとして50年近く、多くの研究者が取り組んできた。タンパク質の立体構造がわかれば生体内の様々な機構解明につながる上、疾病の原因解明や創薬につながると期待されている。

タンパク質の立体構造は実験を通じて決定できる場合もあるが多くの労力とコストが必要である。決定するには対象のタンパク質を大量に発現させ、精製した上でX線結晶構造解析、低温電子顕微鏡（クライオEM）などで測定し構造を決定する。1つの構造を決定するのに数日から数カ月を要し、時には数年かかっても決定できない。また、多くのタンパク質は実験的に構造決定が難しい場合が多い。例えば生命現象としても創薬のターゲットとしても重要な膜タンパク質は膜の中の構造を保ったまま取り出すことが難しく、疎水性があり結晶化も難しい。ヒトのタンパク質の17%が構造決定されており、残りは未決定であった。

この問題に対し、米Alphabet社の子会社である英DeepMind社はAlphaFoldと呼ばれるシステムを構築し、2020年11月にCASP 14と呼ばれるProtein Foldingの精度を競い合うコンテストで2位以下を圧倒する成績を達成して優勝した。ほぼ実験結果に近い構造を予測することができた。

CASPコンテストを創設した米University of Maryland教授のJohn Moult氏は「ある意味で、この（Protein Folding）問題は解けたと言える」、「AIが初めて重要な科学的問題を解いた事例だ」と述べた。

DeepMind社は2021年7月には技術詳細の論文を公開[1]するとと同時に、推論部分のソースコード[2]を公開し、また学習済モデルもあわせて公開した。世界中の多くの研究者がAlphaFoldを動かして自分の持っているデータを決定し、その中には数年間構造決定できなかったのがAlphaFoldで決定できたと述べている研究者もいた。

また、DeepMind社と欧州分子生物学研究所（EMBL）はAlphaFoldで構造予測した2万のヒトプロテオームすべてを含む35万を超えるタンパク質のデータベースを公開した[3-4]。これまでヒトのタンパク質の17%が実験的に構造決定されてきたが今回のAlphaFoldを使って98.5%を予測できた。そのうち36%については非常に高い確信度で決定できたとしている。今後数カ月で1億を超える配列のほとんどを構造予測し公開すると述べている。

本稿ではこの問題の意義とAlphaFoldの技術詳細、今後の展望について述べる。

タンパク質フォールディング

タンパク質は20種類のアミノ酸がペプチド結合で鎖状につながってできた高分子化合物であり生体の構造や機能の中心的な役割を果たしている。DNAに保存された遺伝情報はmRNAを介してタンパク質として発現される。タンパク質の構造は、アミノ酸の種類に関係しない中心のつながりを主鎖とよび、枝のように主鎖から分岐し横に広がるアミノ酸に固有の部分を側鎖と呼ぶ。また1つのアミノ酸のユニットを残基と呼ぶ。タンパク質は疎水性相互作用、分子内水素結合、ファンデルワールス力によって導かれる自発的過程で非常に複雑な構造を一瞬（数μsから数ms）で構成する。構造決定はこの主鎖と側鎖がとる構造を決定することである。

前述のように実験を通じてタンパク質の構造を決定するには多くの時間、コスト、労力を必要とするため、アミノ酸配列を入力とし、タンパク質の構造を予測するようなシステムを作ることが生命科学の1つの目標であった。この目標達成を促進させるため、1994年より世界中の研究コミュニティが予測手法を競い合うCASPが2年毎に開催されていた。このコンテストでは世の中でまだ構造が公開されていないタンパク質を対象に各チームが構造を予測し、実験的に決定された構造と予測結果を比較し評価する。

こうした中DeepMind社は2018年にCASP 13に出場しAlphaFold v1で優勝したが2位以下とは接戦であった。そして2020年11月にAlphaFold v1とは異なるアーキテクチャを持ったAlphaFold2でCASP 14に出場し、2位を圧倒するスコアで優勝した。主鎖の精度は0.96Å $RMSD_{95}$（95%の

残基のカバレッジでの平均二乗誤差）であり、2位の2.8Å $RMSD_{95}$を大きく超えていた。ちなみに炭素-炭素間共有結合距離が1.4Åである。また、銅鋲も含めた全体の精度は1.5Åであり、2位の3.5Åを大きく超えていた。従来の予測方法を超える高精度な予測が可能である。以降の説明でAlphaFoldはこのAlphaFold2を指す。

AlphaFoldの技術詳細

AlphaFoldは新しいネットワークアーキテクチャと学習方法を用いてこの結果を達成した。以前のAlphaGoと同様に、AlphaFoldも全く新しい手法を採用して達成したというよりは多くの手法をうまく組み合わせ工学的にそれを使えるようにまとめあげた部分の貢献が大きい。とはいえ、通常の論文でいえば10報分近く新しいアイデアや工夫が盛り込まれている。

これまでタンパク質の構造解析には熱力学や動力学に基づいて決定する方法と、既に構造が決定されている進化系統樹的に近い遺伝子配列を探し、それを参考にして推定する進化論に基づく方法が使われていた。前者はタンパク質の分子量が非常に大きいこと、また環境要因で安定状態を求めることが計算量的に難しい問題があった。ここ数年は多くの構造が決定されてProtein Data Bankに蓄積されてきたこともあり進化論に基づくアプローチが有力となっていた。しかし、データベース中に似た配列が存在しない場合は進化論的なアプローチは有効ではなく、またその精度も十分ではなかった。AlphaFoldはこれらのアプローチを組み込みながら、機械学習で構造を直接推定する。

AlphaFoldは大きく2つのステージから成る。最初のステージは入力配列の特徴を決定し、続くステージは求められた特徴から構造を決定する。

最初のステージでは、従来の進化論に基づくアプローチと同様にMSA（Multiple Sequence Alignment）と呼ばれる入力配列と似た多数の配列をデータベースから探索し、対応する残基が並ぶように整列させたデータを作る。

次に、N_{seq}をMSAで集めた配列数、N_{res}を配列長とした時、MSAを表す$N_{seq} \times N_{res}$のサイズを持った行列と、残基間のペアを表す$N_{res} \times N_{res}$というサイズをもった行列を保持し、これらを更新していく。これらの行列ではチャンネル方向は省略しているが実際は$N_{seq} \times N_{res} \times C$というテンソルを保持する。前者をMSA表現、後者をペア表現と呼ぶことにする。MSA表現はMSAの結果で初期化し、ペア表現はMSAの配列毎の外積を計算してそれらを足し合わせたものとして得る。

次に、Evoformerと呼ぶネットワークを使ってMSA表現とpair表現を更新していく。MSA表現は軸毎の注意機構を使って更新する。ペア表現は、残差間のグラフだと考えi, j, kの3つの残基を考えた時にj, k間の更新をする際にij, ik間の関係を考慮した上で更新するTriangle multiplicative updateを適用する。この手法自体は注意機構の代わりに使われていた手法だがそれを利用した。そしてペア表現の情報は再度MSA表現に残基間の注意機構時のバイアスとして利用される。このEvoformerを48ブロック重ね、MSA表現とペア表現を得る。

2つ目のステージではMSA表現での元の入力に対応する行、ペア情報、そして主鎖の現在の推定結果を入力とし構造を出力する。主鎖の構造はグローバルフレームに対する各残基のローカルフレーム$T_i := (R_i, \mathbf{t}_i)$として表現される。各残基のローカル座標$\mathbf{x}_{local}$は$\mathbf{x}_{global} = R_i\mathbf{x}_{local} + \mathbf{t}_i$としてグローバル座標$\mathbf{x}_{global}$と変換される。ネットワークはローカルフレームと、主鎖と側鎖のねじれ角（Torsion Angle）を逐次的に出力し、構造を修正していく。

はじめは不変点注意機構（IPA：Invariant Point Attention）を使って内部状態を更新する。このIPAはグローバルフレームの回転や平行移動に対して結果が不変になるように設計した注意機構である。これらで特徴を更新した後に各位置でフレームを更新すると共にねじれ角を更新する。そして、主鎖の構造が決定された後に側鎖の構造を、独立に更新する。

全てのフレームが原点で同じ位置、向きにいる状態で初期化し、次にこれらの操作を繰り返し適用していくことで逐次的に構造を決定していく。学習時には途中の構造でも正解との差で誤差を計算し、途中の構造にも有効な学習シグナルが流れるようにする。

そして、最終的には分子動力学に基づくfine-tuneを行って物理的におかしな構造になっている部分を修正する。

このようにして入力から構造が決定されるが、この出力を重みを共有した同じネットワークに再度渡し、繰り返し構造を改善していく。これをネットワークリサイクリングと呼ぶ。リサイクリングで同じネットワークに何回も入れた上で構造を修正する。実際どのように構造が決定されていくのかをみてみると徐々に構造が決定されていくのをみることができる。

AlphaFoldはラベルデータとラベル無しデータの両方を

使って学習する。ラベル有りデータにはPDBデータを利用した。ラベル無しデータは、教師ありで学習して作ったモデルを使って35万の多様な配列の構造を決定し、確信度の高い予測結果を正解に加える。自己蒸留（noisy-studentと呼ばれる）として、自分で予測した結果を目標にして予測する際には、強いオーグメンテーションを入力に加えておくことで、難しい問題設定でも予測できるように学習する。このnoisy studentは他の多くのタスクでも成功している。

また、BERTと同様にランダムに配列の一部を消したり、あるいは別のアミノ酸に変更し、マスクされたこれらの入力を予測できるようにして、教師なし学習を行っている。

従来はドメイン毎に構造予測していたが、AlphaFoldは配列全体を処理しドメイン間で相互作用する場合に対応することができる。

AlphaFold以後はどうなるか

AlphaFoldによってタンパク質の構造予測が格段に容易になったのは確かであり生命科学の進展が大きく加速されるとみられる。生命科学の研究の仕方や創薬の仕方が変わることは確かだろう。例えばAlphaFoldによって構造がはじめてわかったタンパク質で薬が結合しやすいスポットを探索し、それにあった化合物を探すことや、構造変化を起こしやすい遺伝子変異などを特定し、それらを修復するといったことが進められるだろう。

だが、これだけで新しい薬がすぐできたり生命現象が一気に解明されると考えるべきではない。生体内ではタンパク質は単独では存在せず非常に複雑な相互作用によって生体機構が実現されているためだ。また、AlphaFoldが苦手な問題設定も多く存在する。例えば今回のCASPやAlphaFoldは単一のタンパク質の構造決定が目標だったが、生体内ではタンパク質は複数のタンパク質や他の物質と結合した状態で存在し異なる構造をとることが一般的だ。場合によっては複合体も予測できる場合も報告されているがギャップがある。また、AlphaFoldは構造決定の学習シグナルの大部分は教師あり学習から来ており、未知のタンパク質の予測精度は十分ではない。精度も創薬などではオングストローム（Å）よりも細かいサブオングストロームのオーダーが求められ、ペプチド以外の低・中分子化合物の構造予測やそれらの結合予測ができるようになるのが課題である。

これより先に進むにはさらなるネットワーク、学習手法の発展に加えて、量子化学や熱力学などの理論から構造を決定するシミュレーション技術、実験解析技術、生体内での情報を読み取る技術の進展が必要になると考えられる。

1) J. Jumper et al., "Highly accurate protein structure prediction with AlphaFold," Nature 2021.
2) https://github.com/deepmind/alphafold
3) K. Tunyasuvunakool et al., "Highly accurate protein structure prediction for the human proteome," Nature 2021.
4) https://alphafold.ebi.ac.uk/

（2021年10月号掲載の記事に加筆）

第16章

ロボット

16-1

全自動の片付けロボットシステムをいかに開発したか 高精度な物体認識により初めて片付けが可能に

ロボットを使って家事をこなしてほしいというのは昔から存在するテーマである。例えば、2008年に米Stanford Universityが実施した実験では[1]、ロボットのPR1を人が遠隔操作することで、冷蔵庫を開けてビールを取り、蓋を開けて人に渡したり、床に散らかったおもちゃを片付けたり、食洗機に皿を入れたりといったさまざまなタスクがこなせることを示した。この実験からは、家事をこなすためのハードウエア環境は整っており、あとは人が遠隔操作をすることで解いていた部分、環境を認識しそれに応じてロボットを適切に制御することさえできれば解決できるとみられていた。

Preferred NetworksはCEATEC JAPAN 2018で「全自動お片付けロボットシステム」を展示した[2]。この展示ではトヨタ自動車が開発し、研究開発用に提供している生活支援ロボットHSR（Human Support Robot）を使い、ロボットが部屋を自動的に片付けるデモを行った（図16-1）。一般家庭の部屋の床に片付け対象アイテムがランダムに散らかっており、ロボットがそれらを指定された場所に片付けるというものである。家具は全て本物であり、片付け対象アイテムは日常生活で見られるものから約100種類を選んでいる。ロボットはそれらのアイテムを認識し、把持し、落とさないよう運び、物体の種類ごとに指定された場所へ、指定された状態で片付けていく。

また、ロボットに対し、音声やジェスチャーを使って、どこに片付けるのかを指示することができる。指示内容を把握するために、音声認識や人の姿勢認識を実現している。ロボットは全ての物体がどこにあるのかを把握しているため、人が探しているものがどこにあるのかをロボットに問うと、「靴下は玄関の積み木の近くにあります」といったように答えてくれる。このように現実世界の環境がデジタル化され、コンピュータが扱えるようになることで、今後様々なサービスが生まれてくると考えられる。

このロボットで1つのものを片付けるのには、大体30秒から1分かかり、散らかった部屋は1時間程度で片付けることができる。4日間のCEATEC開催期間中、2台のロボットが片付けたアイテムの総数は約5000個に上り、その間、定期的な電池交換や数回起きた物体の巻き込み時を除いて、ほぼ人が介入せず安定して動き続けることができた。

この開発にはデータ収集を含めて半年程度の期間を要し、期間中で平均して20人程度が携わった（図16-2）。以降では、この展示に使われた技術について紹介していく。

デモシステムは始めに4台の天井カメラを使い、部屋の中のどのあたりにどの物体があるのかを認識する。天井カメラがなくてもロボットは片付けることはできるが、その場合、ロボットは最初に部屋を見回し、どのアイテムがどこにあるのかを把握する必要がある。今回はデモのスピードアップのため、天井カメラを利用した。

次に、ロボットがどの物体を取りにいくのかをプランナが決める。今回の展示では最も近くにあるものを取りにいくという単純な戦略を採用したが、重要度の高いものから片付けるといったような様々な要求に応じたプランを作ることは容易だろう。その後、HSRに備え付けられたカメラを使って物体を認識し、その物体の種類や位置、つかむ位置を推定する。この認識結果に従ってロボットは物体を把持し、片付ける位置まで運んでいき、適切な置き方を実行する。

世界2位の成績を収めた深層学習モデルを利用

今回、特に難しかったのは物体認識と把持計画である。扱う物体の種類が約100種類と非常に多く、置かれ方のバリエーションも様々である。対象物体もタオルや靴下など不定形のものも多い。アイテムを検出することと自体が難しい上に、その物体の形状や姿勢を推定することも難しい。検出位置も高い精度が要求され、つかむ位置が数cmずれただけで、物体をつかむことができなくなる。つかむ位置も限定されており、おもちゃのジョウロや、けん玉などは正解の狭い位置はわずかしかない。

この物体認識にディープラーニングによる画像認識器を利用した。画像認識はImageNetで人の認識精度を超えた後も驚異的に性能が向上し続けている。今回のデモシステムに利用した画像認識は、Preferred NetworksがGoogle AI Open Images Challenge 2018に出場し世界2位の成績を収

めた際に用いたPFDet[39]を拡張したモデルを利用しており、100台を超えるGPUを利用し学習している。訓練データは、実際にリビングのような部屋を作って様々な日用品を配置し、写真撮影を行って作成した。

このように学習されたモデルは高い認識精度と環境変化へのロバスト性を達成し、今回のタスク実現の中心的な役割を果たした。印象的な例として、展示期間中に照明が消えるトラブルが発生し、撮影環境が大きく変化した際も、デモは止まらずロボットは正確に片付けを続けることができていた。画像認識の推論はロボットとは別に展示ブースの制御室に置かれた外部のGPU付きサーバーで実行し、1秒間に2〜3枚の画像を推論することができた。今後は、ロボットに搭載されたチップ上で推論するようになるだろう。

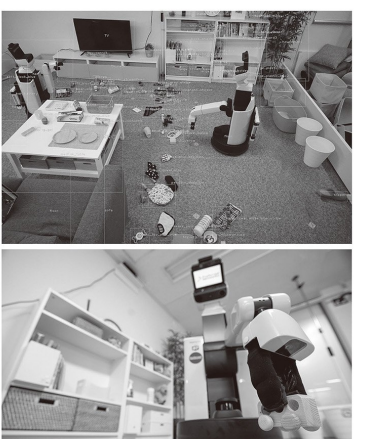


図16-1　CEATEC会場でのデモの様子
写真上は部屋の中にある物体の認識の結果、写真下はおもちゃを実際にロボットが把持している様子。

音声やジェスチャーなどで指示可能

ロボットには音声で指示を出すことができ、この実現のための音声認識は展示会場のノイズに強くなるような工夫を施している。また、今回の問題設定にあわせて対象物体や指示に特化するよう言語モデルの最適化も行っている。

ユーザーは音声で、どの部分を片付けてほしいか、物体をどこに運んでほしいか、物体はどこにあるのかを指示することができる。これらの指示はロボットの動作途中でも行うことができ、また何回も繰り返すことができる。ロボットへの指示が失敗しても、すぐにもう一度繰り返し指示し、認識できるようになっている。このようにすぐ繰り返せることで認識精度が100%を達成しなくても、利用者がストレスなく指示できるようになっている。こうしたロボットを音声で指示する技術については論文[41]も参考にされたい。

ジェスチャーによる指示もできる。場所の指定など言語だけでは指示しにくい場合や（机の特定の足の付近など）や、名前の分からない物体に対する指示はジェスチャーを使うことが適しているだろう。今回のデモではジェスチャーの認識には天井付近に備え付けられた距離画像センサ（Kinect）を利用しているが、今後はロボットに備え付けられたカメラで認識するようになるだろう。

ロボットを利用する場合、ロボットが何を考えているのか分からない、指示が的確に伝わっているかが分からないという問題がある。今回はAR技術を使ってロボットが何を考えているのかを見せるデモも行った。これによりロボットがどの物体をどのように認識しているか、ロボットが今何を考えているかを直感的に理解することができる。

こうした目に見える機能の他にも、今回のシステム開発には多数の技術が投入されている。例えば展示会場は電波干渉が強いため、ロボットが安定的に外部サーバと通信できるよう無線ネットワークには多くの準備を行った。また、多くのサブシステムが関わる中でも開発スピードを落とさないようテスト自動化などCI（継続的インテグレーション）の実現に多くの開発リソースを割いている。

図16-2　ロボットシステムの開発メンバー
約20人が開発に携わった（写真左）。CEATEC期間中には経済産業大臣（当時）の世耕弘成氏も視察に訪れ、筆者が説明を行った（写真右）。

課題と今後

今回のロボットによる自動片付けを一般家庭で実用化するための技術的な課題は残っている。把持に関してはより多くの種類に対応する必要があり、特に大型の衣類やタオル、センサが苦手な透明なボトル、微妙な力加減が必要なガラス製品などを安定して把持するためには新しい技術開発が必要になるだろう。未知物体でも安定して把持することが求められる。今回はHSRに標準搭載されている平行グリッパーと吸引ハンドを利用したが、対応物体を増やすためには新しい機構やセンサも必要になると考えられる。

また、部屋の環境も様々な状況に対応する必要がある。段差やカーペットがある場合への対応や、環境や光源が変わっても安定して対応できる物体認識が必要だ。さらに、部屋や家の中の正確な地図情報が前もって得られなくてもこうしたシステムが立ち上げられることが望ましい。

言語やジェスチャーによる指示も、部屋の中の多様性をカバーできるようにより表現力を増やすことが求められる。

片付けに限らず他の家事もロボットで解いていくだろう。片付けと同じ技術を必要とするタスクは解けるだろうが（特定のモノを特定の場所に運ぶ、モノを集めるなど）、道具を使う場合や、モノを加工する（例えばダンボールの箱を開封する）といったタスクは難易度が大きく上がる。しかし、冒頭に挙げた実験でもあるように、他のタスクも人が遠隔操作さえすれば解けることは示されているので、これを自動化することは不可能ではないと考えている。

1) http://personalrobotics.stanford.edu/
2) https://projects.preferred.jp/tidying-up-robot/
3) T. Akiba, et al."PFDet: 2nd Place Solution to Open Images Challenge 2018 Object Detection Track," https://arxiv.org/abs/1809.00778
4) J. Hatori, et al."Interactively Picking Real-World Objects with Unconstrained Spoken Language Instructions," ICRA 2018.

（2018年12月号掲載の記事に加筆）

環境乱択化：Domain Randomization

機械学習に必要なデータを現実世界で収集することはコストや時間がかかるだけでなく、ロボットのような物理的な動きを伴う機械の場合、危険であったり、まれな事象でそもそも集められない場合も多い。そのため、現実世界をシミュレーションした環境上でデータ収集することが期待されてきた。

しかし、シミュレーション上で収集されたデータを学習データや検証データとして使う場合、シミュレーションと現実世界とのギャップ、いわゆるsim2realギャップが問題となる。シミュレーション上で学習したモデルはシミュレーション環境に過学習してしまい、現実世界の問題では大きく性能が劣化してしまう。特にニューラルネットワークのような強力なモデルを使った場合、モデルはシミュレータ上でしか起きない現象を不正に利用して問題を解こうとしてしまう。この問題を避けるためはシミュレーションを実世界の環境に近づけることが必要だが、シミュレータ開発は多くの場合困難であり、ロボットなどのハードウェアのシミュレーションだけでなく物理現象としての摩擦や衝突、カメラのシミュレーションなどは現実とのギャップが特に大きい。

この問題を克服するために環境乱択化（Domain Randomization）と呼ばれる手法が提案された。これは環境の様々なパラメータをランダムに変えたバージョンをたくさん用意し、それら様々な環境の全てでうまくいくようなモデルを学習する。パラメータとしては例えば摩擦係数や衝突係数、物体のテクスチャ、光源モデルなどがある。様々な環境上で学習されたモデルは環境に多少の変化があっても対応できる、つまり環境に対して汎化しているため、環境の変種の1つである現実世界の環境に対しても、うまく動作することが期待できる。

例えば、米OpenAIは環境乱択化を利用し5指ハンドの制御をシミュレーション上で学習し、それを現実の5指ハンド上に適用して高度な制御が実現できることを示した[1]。具体的には最も複雑な5指ハンドの1つであるShadow Dexterous Handを使い、手の中にあるサイコロなどの物体を狙った向きに回転させるタスクを学習した。5指ハンドは高価であるだけでなく、故障しやすく、個体差や摩耗などによる経時変化も大きい。そのため実際の5指ハンド上で大量の試行データを収集することは不可能とみられていた。このタスクをシミュレータ環境上で構築し、環境乱択化を適用し強化学習を使って学習させた。現在の強化学習は多くの試行回数を必要とするが、このタスクでもシミュレータ上で、1つの環境向けに5指ハンドの制御を学習するには実世界換算で約3年分の試行が必要であった。また、環境乱択化の上で様々な環境に汎化した制御を獲得するには実世界換算で約100年分の試行が必要であった。これは8個のGPU、6144個のCPUコアを使ったシミュレータ上では3年分は1.5時間、100年分は50時間で実現できる。

環境乱択化に加えて、本手法では強化学習のエージェントがRNNを使っていることが特徴的である。これによりエピソード前半の経験を元にエピソード後半の方策を変えられるようになっている。エピソード前半で環境がどのようなパラメータを持っているのかをRNNが推定し（例えば、物体が滑りやすいなど）、それに応じてその後の行動を調整し、エージェントがエピソード中に環境に自動的に適応することを期待してのものだ。

このように、環境乱択化はシミュレーションで学習したモデルを現実世界の問題に適用することを可能としたが、いくつか問題がある。その1つに本来は現実世界の問題さえ解ければ良いのに、環境乱択化を用いて学習する場合は、それよりはるかに難しい多くの環境でうまくいくようなモデルを学習しなければいけないということが挙げられる。そのため、モデルは必要以上に強力である必要があり、それを学習させるために、より多くの学習データが必要となる。実際、先程の5指ハンドの制御の場合は1つの環境に対する学習の30倍近くの試行が環境乱択化を適用した問題では必要となっていた。

この問題を解決するために、米X社、米グーグルBrainらのチームは、環境乱択化を使って直接方策を学習するのではなく、観測を共通フォーマットのような正準形式（canonical version）に変換する方法を提案した[2]。シミュレータと現実世界の観測を両方、共通の形式である正準形式に変換し、その上で制御することができれば、シミュレータで学習した

モデルをそのまま現実世界に適用できる。

問題は観測から正準形式への変換をどのように学習するかである。シミュレータ上の観測から正準形式への変換は、学習データをシミュレータ上でいくらでも作ることができるため容易である。それに対し実世界の観測を正準形式へと変換するための学習データは人手によるアノテーションが必要となり、膨大な時間とコストが必要となる。これを解決するため彼らは環境乱択化を使い、様々な環境の観測を正準形式に変換することをまず学習した。この変換は環境の変種に対して汎化するため、環境の変種の1つである実世界の観測に対しても正準形式に変換することが可能となる。

彼らは、ロボットによる把持タスクについて、ロボットの肩越しに設置されたカメラからの画像を入力とし、クローズドループでアームとハンドを制御する問題を扱った。この問題では入力カメラの画像のシミュレーション結果を実際に得られる画像と一致させることが困難であった。

彼らは、正準形式として、背景、物体を置くトレー、ロボットの腕を単色としたような表現を採用した。また、把持対象物体はテクスチャ情報のみを残した上で単色に変換したような表現を採用し物体の情報（姿勢や部位）が得られるようにした。また、腕の状態も画像から得られるように腕の各リンクはそれぞれ違う色にした。光源は固定された位置に置かれたものにした。この正準形式はsemantic segmentationと似ているが、物体のテクスチャや、正準化された光源が生み出す影などにより、より詳細な情報が表現できる形式になっている。

環境乱択化を適用した上で各環境下の画像から正準形式への変換を学習する際には、学習を助けるために、正準形式に加えて、セグメンテーションマスク、深度情報も同時に予測するタスクを解いた。学習ではシャープな画像を変換して生成できるようにGAN（敵対的生成モデル）を利用している。

把持学習の試行データとして、方策オフ型と方策オン型の2種類のデータが必要となる。全ての試行データを実際のロボットを使って集めた場合では方策オフ用の学習データ作成に実世界で58万回の把持の試行が必要であった。この試行には7台のロボットで数週間を要している。これに方策オン型の5000回のデータを加えると87%の精度、28000回のデータを加えると96%の精度を達成できる[3]。

それに対し、正準形式に変換する提案手法ではシミュレータ上の学習データのみを使って70%の精度で把持できるようになる。さらに実世界の方策オン用の5000回のデータを加えると精度は91%となり、28000回のデータを加えると精度は94%となる。もともと実世界でトータル60万回弱の試行で87%達成していたのが、提案手法は5000回の試行のみで91%の精度を達成でき、実世界の試行回数を1/100に減らすことができている。

環境乱択化は人手で作ったシミュレーションだけでなく、観測から学習したモデル上に対しても適用することができる。例えば観測から環境モデルを1つではなく複数学習し、それら全てに対してうまくいくようにメタ学習する手法が提案されている[6]。計算性能が今後も向上し続けることが期待される中でこのような仮想環境上での学習はより重要になってくると考えられる。

1) M. Andrychwicz et al.,"Learning Dexterous In-Hand Manipulation," https://arxiv.org/abs/1808.00177
2) S. James et al.,"Sim-to-Real via Sim-to-Sim: Data-efficient Robotic Grasping via Randomized-to-Canonical Adaptation Networks," CVPR 2019.
3) D. Kalashnikov et al.,"QT-Opt: Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation," CoRL 2018.
4) I. Clavera et al.,"Model-Based Reinforcement Learning via Meta-Policy Optimization,"CoRL 2018.

（2019年3月号掲載の記事に加筆）

初出一覧

本書での掲載　　初出の雑誌掲載の時期

第1章

第2章

第3章

第4章

第5章

第6章

第14章

第15章

第16章

著者略歴

岡野原 大輔　Okanohara Daisuke

2010年、東京大学大学院情報理工学系研究科コンピュータ科学専攻博士課程修了(情報理工学博士)。在学中の2006年、友人らとPreferred Infrastructureを共同で創業。また2014年にPreferred Networksを創業。現在はPreferred Networksの代表取締役CERおよびPreferred Computational Chemistryの代表取締役社長を務める。

AI 技術の最前線
これからの AI を読み解く先端技術 73

電子書籍版データ作成日　2022 年 7 月 26 日　第 1 版

著者　岡野原 大輔
発行者　戸川 尚樹
発行　株式会社日経 BP
装丁　土井 直明（市川事務所）
制作　マップス
編集　進藤 智則（日経 Robotics）

●この電子書籍は、印刷物として刊行された「AI 技術の最前線　これからの AI を読み解く先端技術 73」（2022 年 8 月 8 日第 1 版第 1 刷発行）に基づき制作しました。文章中の固有名詞などは掲載当時のものです。また、掲載時の誌面とは一部異なる場合があります。

《電子書籍版について》

●おことわり

ご覧になる端末機器や、著作権の制約上、写真や図表、一部の項目をやむなく割愛させていただいている場合があります。また、端末機器の機種により、表示に差が認められることがあります。あらかじめご了承ください。