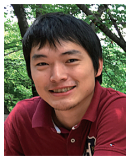
解説

物理屋のための深層学習

瀧　雅人

理化学研究所数理創造プログラム（iTHEMS）
taki@riken.jp



田中章詞

理化学研究所革新知能統合研究センター（AIP）
akinori.tanaka@riken.jp

このところ深層学習（ディープラーニング）という言葉を頻繁に耳にするようになってきました．巷では技術的特異点などのパワーワードに引っ張られ，人工知能が人間の仕事を奪うかもしれない等と報道されるケースも少なくないため，怪しい分野だと思われている方もいるかもしれませんが，そうではありません．機械学習の理論的下地は確率・統計学にあり，むしろ物理学を修めた方なら誰でも腑に落ちるような枠組みに支えられています．

深層学習は，数多ある機械学習の手法群の中の一つの手法です．その一方，親玉である機械学習という分野は，データをもとにそこからパターン・知識を抽出する手法一般の研究開発を指します．物理学者が普段行っているデータ解析のうち，ある程度の割合は機械学習だといっても過言ではないでしょう．

深層学習は，ニューラルネットワーク（ニューラルネット）と呼ばれる特殊な数理モデルを用いる点で，他の機械学習の手法と大きく異なります．ニューラルネットは20世紀の中頃，動物の脳の数理モデルとして提唱されましたが，その後はデータの学習のための機械学習モデルとして広く研究されるようになりました．長い研究の歴史を持つニューラルネットですが，2006年頃になり新しい段階に突入し，やがて加速的な発展期を迎えます．ネットワーク構造を深層化することでニューラルネットが極めて高い学習能力を発揮することが実証され，広範なタスクに対応できるネットワーク構造が次々と開発され始めたのです．

この一連の流れで発見されたアルゴリズム・技術・ノウハウの総体が深層学習だ，といって良いでしょう．深層学習は画像認識にとどまらず，Google翻訳のような自然言語処理，音声の変換・生成，フェイク動画の生成，アルファ碁に代表されるゲームAIなど，急激にその応用範囲を拡大し，産業・科学技術の風景を変えつつあるのは皆さんもご存知の通りです．

では，この間の研究によって全ては理解され尽くされ，応用も試み尽くされたのでしょうか？　実態はそうではなく，むしろその逆です．ニューラルネットが高い性能を実現する理論的なメカニズムは未だほとんど理解されておらず，そのためアドホックなネットワーク構造の設計も当然第一原理に基づくものではありません．そしてゲームAIのような探索的作業への大きな可能性があるにもかかわらず，深層学習の基礎科学への導入は，まだ部分的かつ初歩的な段階にあります．深層学習の高い表現能力や汎化性能の理論的理解や，データサイズに比べてデータ次元が極めて高いような場合に対応できる学習アルゴリズムの発見など，今後物理学者も寄与できる未解決問題も数多くあると考えられます．またこれまでの産業・ソーシャルデジタルデータだけではなく，科学データへの応用を通じて露わになる深層学習の技術的問題点や改良の可能性も数多くあるでしょう．これからは，基礎科学研究によって深層学習の新しい可能性が開けていくものと期待しています．

Keywords

機械学習：
明示的なプログラムをすることなく，機械が学習できる能力を持つようにするアルゴリズムの総称．

ニューラルネットワーク：
脳の神経回路網を模した数理モデル．ノードとエッジからなるグラフ構造，特にある種の層構造を持ち，非常に高いフィッティング能力を持たせることができる．

深層学習：
非常に多くの層を持つニューラルネットワークを用いた機械学習の総称．近年の人工知能ブームの火付け役であり，従来の機械学習手法の常識を覆す能力を持つと期待されている．

敵対的生成ネットワーク：
深層学習の方法を用いた，データを作るタイプのネットワーク．データ生成ネットワークとデータ識別ネットワークを競わせることにより，より本物に近いデータを作り出す．

1. 機械学習と深層学習

深層学習[1,2]は機械学習の一種です．機械学習とは大雑把に言うと，経験を通じて，未知の状況に適応できるようになる能力を機械に与える方法と言えるでしょう．以下ではその中でも基本的な考え方となる，教師あり学習と呼ばれる方法を説明します．

1.1 教師あり学習

教師あり学習とは，二種類の変数xとyの観測値のペアを集めたデータ$\{(x_n, y_n)\}_{n=1}^{N}$から，xとyの間の関係を一般的に学び取る機械学習です．

少し理論的に説明します．まずデータを生み出す確率的な事象が確率分布$P_{\text{data}}(x, y)$として表せるものとし，観測値はその分布から具体的にサンプリングされたものだとみなしてみます：

$$x, y \sim P_{\text{data}}(x, y) \tag{1}$$

教師あり学習の目的は，xとyの間の関数関係を推定することにつきます．ですので，データを生み出す確率分布P_{data}を見つけることができれば一番ですが，通常は与えられたxの値に対してyを予測できれば満足ですのでそこまでは行いません．与えられたxに対して最もそれらしいyの予測値は，xのもとでのyの期待値でしょう：

$$\hat{y}(x) = E_{P_{\text{data}}(y|x)}[y] \tag{2}$$

しかし我々の手元にはもちろん有限のデータしかないので，このような期待値は計算しようがありません．またデータ平均で近似しようにも，それでは見たことのあるxに対してしかyを予測できません．そこで$\hat{y}(x)$として何らかのパラメータ付きモデル

$$\hat{y}_{\text{model}}(x, w) = f(x, w) \tag{3}$$

を仮定できるものとして，パラメータwをデータからフィッティングすると考えます．データへの当てはまりが良いパラメータw^*が見つかれば，$\hat{y}_{\text{model}}(x, w^*)$を使って初見の$x$に対してももっともらしい$y$の値が予測できることになります．これがデータの学習ですので，教師あり学習はデータのフィッティングに他なりません．[*1]

ではどのようなパラメータ付きモデル$f(x, w)$を使えば問題が良く解けるのでしょう？　通常はデータの特性に応じて適切な$f(x, w)$の関数形を我々が設計します．例えば普通のデータ解析でも振動データには三角関数を用いるでしょうし，鼠算式に増えていく数値データには指数関数を仮定するでしょう．そうするとできるだけパラメータが多く，複雑性の高いモデルを用いた方が多様なデータに対応できて良いように思われます．ところが世の中そう簡単にはいきません．オッカムのカミソリよろしく，無用な説明因子を導入することは問題を引き起こします．機械学習の場合，多量のパラメータを用いると，データに含まれる統計的な揺らぎまで学習してしまい，与えたデータに対して過剰に適合してしまい汎用性を失います．この問題は過学習と呼ばれます．

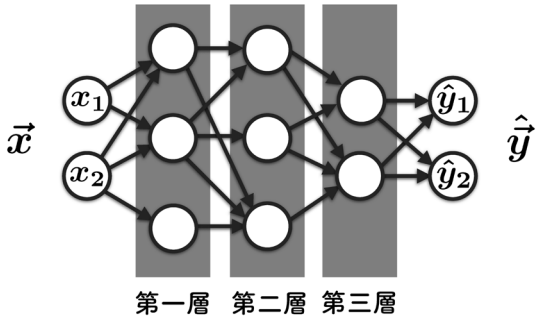


図1　ベクトル値入力データ$\vec{x}$を予測値$\hat{\vec{y}}$へ変換する有向ネットワーク．

そうすると汎用性の高いモデルを望むことは難しく，機械学習はケースバイケースになってしまいそうです．ところがニューラルネットは過学習を起こしにくく，汎用性の高い手法を提供してくれます．

1.2 深層学習

ニューラルネットを使った機械学習とは，一言で言えば有向グラフ（図1）を使ってモデル$f(x, w)$を与える手法群です．グラフを描くことでモデルを設計できるので，良いモデルを探す際に人間の幾何学的直感を活かしやすい手法になっています．そのためか，かなり広いクラスの問題をニューラルネットを使って解くことができます．深層学習はその意味で，機械学習の幾何学化[*2]ということができます．

ネットワークを描く際はまず図のように入力・出力データの変数の数だけユニットを用意し，それら以外に適当な数の隠れユニットというものも用意します．それらを矢印で適当に結んでグラフを一個描くと，自動的にxからyを与えるモデル$\hat{y}=f(x, w)$の関数形が定まります．有向グラフはいろいろなものが考えられますが，ニューラルネットでは，基本構造として層構造を用いることが性能の秘訣であることがわかっています（図1）．その層数が多い場合を「深層」と呼びますので，深層学習は多層のニューラルネットワークを使った機械学習のことに他なりません．

1.3 ユニットの演算規則

各ユニットの信号処理のルールが決まればネットワーク全体の振る舞いがわかりますので，各ユニット周辺の振る舞いを説明します．まず矢印の上には矢印の向きに沿って，数値が信号として伝播するものと考えます．さらに各矢印には固有のパラメータが与えられています．

ある一個のユニットが，たくさんの矢印を通じて数値$z_{i=1,2,\cdots}$を受け取っているものとします（図2）．するとこのユニットは次の数値zを外向きの矢印に沿って，他のユ

[*1] 実際にはどのようにフィッティングさせるのか，という部分も非常に重要になりますが本稿では省略します．

[*2] 一般相対論が重力理論の幾何学化と呼ばれる，という程度のニュアンスです．

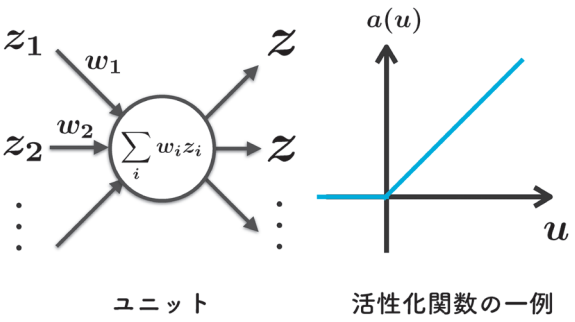


図2　一つのユニットにおける信号処理と，活性化関数の一例（ReLU）．

ニットへ向かい出力します：

$$z = a\left(\sum_i w_i z_i\right) \tag{4}$$

ここで，i番目の矢印に沿ってやってきた数値z_iにはパラメータw_iが掛かっていることに注意します．さらにそれらの総和$\sum_i w_i z_i$は，活性化関数と呼ばれる関数$a(\ )$で変換されてから出力されます．ではなぜ単に総和を出力せずに，一度$a(\ )$を作用させるのでしょう？

一つの答えは，$a(\ )$を用いないと各ユニットは単に線形な変換（アフィン変換）を行うだけであり，するとネットワーク全体としても線形変換を行うだけになってしまいます．これでは線形回帰と変わりません．そこでモデルの非線形性のタネとして，$a(\ )$という関数を挿入するのです．

またこの関数は，通常は閾値を持つものが用いられます（図2）．実はニューラルネットは，脳のような動物の神経回路を模倣した数理モデルになっています．そして実際の神経細胞の応答には，入力に対する閾値があると言われています．したがって生物学の知見から，そのような活性化関数を用いることはもっともらしいと考えることができます．

2.　学習の実際：SGD

モデル$f(x, w)$を一旦決めてしまえば，あとはその中から良いパラメータw^*を探し出せば良いわけですが，データ$\{(x_n, y_n)\}_{n=1}^N$が与えられたとして，どのようにw^*を探せば良いでしょうか．深層学習の文脈では，xを入力としたときのネットワークの出力値$f(x, w)$と教師信号yの差を測る関数$l(f, y)$を導入します．これは結局wの関数とみなせるため，この関数をなるべく小さくするようなw^*を求めれば良いことになります．そこで

$$l(w) = \frac{1}{N}\sum_{i=1}^{N} l(f(x_i, w), y_i) \tag{5}$$

として$l(w)$の最小点を求める問題を考えます．とは言っても深層学習ではこれを真面目に解こうとはしません．なぜなら深層ニューラルネットワークではそもそもwの次元が高すぎるため，いかに数値計算といえども厳密な最小値を求めるのは難しいからです．加えて，そもそも本当に小さくしたいのは1データ辺りの誤差$l(f(x_i, w), y_i)$の$P_{\text{data}}(x, y)$（式(1)）について期待値なので，その近似と思える今の「経験」誤差(5)を真面目に小さくしても恩恵が少なそうだというわけです．

更に，誤差逆伝搬法と呼ばれるニューラルネットワークの性質をうまく利用した$l(w)$の各wに対する微分係数の高速計算アルゴリズムが知られています．このことと上述の理由などから，深層学習では重力ポテンシャル中を転がっていくボールのように$l(w)$の微分係数が減っていく方向にwをアップデートしていく手法を使います．それが勾配降下法です：

$$w^{t+1} = w^t - \epsilon \nabla_w l(w)\ . \tag{6}$$

ここでw^tは現在のパラメータを表し，w^{t+1}は更新後のパラメータを表します．また，ϵは学習率と呼ばれるパラメータで，この値が大きいと一回あたりのwの更新が大きくなる代わりに，探したい最小点を飛び越えてしまう可能性があったり，動きすぎたがゆえにポテンシャル$l(w)$の壁にぶつかってしまい振動が引き起こされてしまったりします．従って小さな学習率のほうが良いような気がしますが，そうすると今度は見かけの極小点にはまって抜けられなくなったり，学習が進みづらくなったりしてしまうので，良い塩梅のϵを問題ごとに選ぶ必要があります．[*3]

2.1　確率的勾配降下法（SGD）

データを十分沢山集めると，冗長性が生じます．例えば猫の画像データを沢山集めれば，中には似た猫が写っているデータが多数存在するでしょう．単に猫であることを認識させたいのであれば，猫のポーズ等の詳細はどうでも良いので，これらの猫画像はほとんど同じ情報を持っていることになります．このような場合，元のデータから式(5)を計算し，式(6)でパラメータ更新することは余り効率的ではないように思われます．

そこでデータをn個のデータから成る部分データに小分けにし，部分データについて式(5)を計算，式(6)でパラメータ更新を繰り返す方法が考えられます．部分データのことをミニバッチ，そのサイズnのことをバッチサイズと呼びます．ミニバッチを全て使い終わると，元のデータをシャッフルした後，またデータを分割し学習を繰り返します．全データ単位で何回学習した段階にあるかの単位をエポックと呼びます．従って1エポックは次に相当します：

$$\underbrace{\{\underbrace{(x_1, y_1), \ldots, (x_n, y_n)}_{\text{ミニバッチ1}}, \underbrace{(x_{n+1}, y_{n+1}), \ldots (x_{2n}, y_{2n})}_{\text{ミニバッチ2}}, \ldots\}}_{N\text{個のデータ}} \tag{7}$$

このような手法を確率的勾配降下法（Stochastic Gradient Descent method, SGD）と呼びます．SGDでは$l(w)$の定義が毎回の更新で変化するため，極小点にはまりにくい方法であることが期待されます．深層学習における教師あり学習のプロセスをまとめておくと，

[*3] このϵの値を学習ごとに自動調節してくれるアルゴリズムも存在しますが，その場合でも手で決めなければならない別のパラメータがあります．

1. データ $\{(x_n, y_n)\}_{n=1}^{N}$ を用意する．
2. グラフを描いて $f(x, w)$ を設計する．
3. $\{(x_n, y_n)\}_{n=1}^{N}$ と $f(x, w)$ を，SGD に適応し w^* を探索する．

ということになります．

3. なぜ深層か？：深層学習の折り紙

この疑問には完全な答えは未だに与えられていませんが，いくつか魅力的な仮説[4]がありますので一つだけ紹介しましょう．

解かねばならない問題が，図3左側のような二種類の点の分類問題だとします．すると問題は，二種類の点の間に複雑な境界線を引く問題だということになります．込み入った問題の場合，この境界は複雑で，その推定は簡単ではないでしょう．

ところがもしこの空間を折りたたんでいくことができ，最終的に図3の右側のように折りたたむことができたとします．すると分類は，折りたたんだ空間の上に単純な境界線を引く問題になります．つまり入力の空間を折りたたんでデータの分布具合を簡単化させられたとすると，分類問題は単純化するのです．

一方，実際に $a(z) = \max(0, z)$ という活性化関数（ReLU）を用いると，二層ニューラルネットだけを使って

$$z_1 = |x_1|, \quad z_2 = |x_2| \tag{8}$$

というニューラルネット $(x_{1,2}) \to (z_{1,2})$ を作ることができます．$z_{1,2}$ の値の観点から見ると，入力空間の第一象限から第四象限までは全て同一視されます．つまり元の空間が四分の一に折りたたまれているのです．ニューラルネットに層を幾重にも重ねるとは，このような折りたたみを何回も繰り返すことを意味します．すると複雑な分類曲線が，ニューラルネットの上流側では単純な曲線として表現し直すことができ，ニューラルネットを通して見ることで問題が簡単化する可能性があるのです．

もっと複雑で現実的なニューラルネットにおいても，このような現象が起こっている可能性が提唱されています．[4] 単層ごとでは単純な折り返しをしているだけでも，層の数だけ繰り返していくことで，最終的には難しい分類問題が簡単に解けるというわけです．多層性により，複雑な分類パターンを捉える表現能力を獲得しているとも言いなおすことができます．これが多層にしなければならない理由の一つです．正確な議論は文献4などを参照ください．

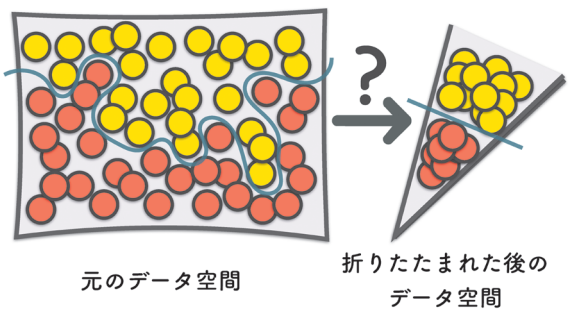


図3　元のデータ空間での散布図と，折りたたまれた空間から見た散布図．

4. 敵対的生成ネットワークの数理

深層学習における興味深い発展の一つに，敵対的生成ネットワークを用いた画像生成が挙げられます．[5] 画像データ x が

$$x \sim P_{\text{data}}(x) \tag{9}$$

で生成されているとしましょう．敵対的生成ネットワークではこのデータ生成分布を真似することで，現実にありそうなデータを生成することを目的とします．まず真似をするためにデータの種 z が何らかのサンプリングしやすい確率分布 $Q(z)$ から与えられるとします：$z \sim Q(z)$．この種を用いて生成ネットワーク G が画像 $G(z)$ を生成します．

敵対的生成ネットワークではもう一つのネットワーク D を用意し，G と D を戦わせることで G が本物らしいデータを生成できるようにします．G を「偽金作り」とし D を「警察官」とするとわかりやすいかもしれません．「警察官」は偽物と本物を区別するようになり，「偽金作り」はその警察官の目を欺けるような偽物を作るようになるというわけです．具体的には D や G を交互に調整することで，

$$V(G, D) = E_{P_{\text{data}}}[\log D(x)] + E_Q[\log\{1 - D(G(z))\}] \tag{10}$$

の値を D の最適化では大きく，G の最適化では小さくなるように学習します．実際には D と G を交互に学習させるので，式(10)の均衡点，つまり D にとってはこれ以上 V を大きくすることができず，G にとってはこれ以上 V を小さくすることができないような D, G の組み合わせに近づけるのが学習のゴールとなります．

これでなぜ現実にありそうなデータが作れるのか，元論文[5]では以下のように説明しています．まず D について式(10)の最大値を取ると，それが確率分布 P_{data} と G による Q の push forward の間の距離を与えることが示せます．それを最小化するには距離の性質から両者が一致するしかない，というわけです．しかしながら，理論と実際の間にはやや隔たりがあることがわかっています．例えば実際の学習にて D は強くなりすぎてしまうと，G はヘソを曲げてしまい学習できなくなってしまうことが知られています．近年ではこの問題を解決するのに最適輸送問題の考え方が応用できること[9]が指摘されるなど，学習を安定化させるためのアイデアが，幾何学や物理学から持ち込まれてきています．

5. 汎化ギャップの謎

深層学習には多くの理論的な謎があり，理論研究者の挑戦を待っているのですが，ここではその一つだけを取り上げましょう．

これまでに比較的安価な計算資源であるGPUを利用して深層学習の計算を並列・分散化させる研究が進んできました．深層学習の学習はかなりの計算時間を要するものの，SGDの各ステップにおいては各データ点ごとに計算は独立しています．したがってミニバッチをさらに細かく分割し，たくさんの計算機に計算を割り振ることで計算時間を大幅に短縮できます．

つまり大きなサイズのミニバッチを用いた方が，計算資源を効果的に利用し尽くすことができるのです．ところが計算が速くなる一方で，看過できない問題が発生することが数値的に発見されました．それはミニバッチを大きくするにつれ性能が下がる問題です．これは汎化ギャップと呼ばれ，ミニバッチのサイズを大きくしていくと，学習後の性能（汎化性能）が低下するという問題です．

これは深層学習のスケーラビリティを脅かす深刻な問題です．実用的な観点からも，問題の原因を特定してこのギャップを解消することが求められます．

この問題の理論的な理解の仕方として，SGD $w^{t+1} \leftarrow w^t$ をランダムポテンシャル中のブラウン運動としてモデル化するアイデアが提唱されています．[3] ランダムポテンシャルにおける高次元ランダムウォークの研究結果から，拡散距離は勾配の更新時間に対して対数で遅くなることがわかります：

$$\|w^t - w^0\| \sim \log t \tag{11}$$

物理側ではこれは異常拡散と呼ばれる現象で，ランダム性の高いポテンシャルのために拡散速度が極端に遅くなっているのです．しかも深層学習の数値的な実験から，この効果はバッチサイズの増大とともに深刻になることが予測されています．

すると汎化ギャップとは，実は単にSGDがまだ収束に達していないだけだ，と予想することができます．すると計算時間をかけてSGDを継続させることでギャップが解消することになります．実際，訓練性能や検証性能に変化が見られなくなった後もSGDをしばらく継続することで，汎化性能が上昇することが観察されています．ただし計算時間を伸ばすことはコストの観点からは歓迎されないため，現在でも様々な汎化ギャップの解消法が研究されています．

6. いろいろなライブラリ

深層学習の実装は幾らかの段階に分けることができます．データの前処理，ネットワークの設計，SGD，学習結果の評価などです．一から自分で書きたくなるかもしれませんが，既に多くのライブラリが（主にPython向きですが）無料で配布されておりインストールも容易なので，これを使わない手はないでしょう．制作者の考え方によって上記の操作をいかに実装すべきかというところでライブラリのテイストにも幾らかの差が出てきますが，ここでは3つの有名なライブラリを紹介します（表1）．

表1　有名なライブラリ．

ライブラリ名	制作者	思想	文献
TensorFlow	Google	define and run	8
PyTorch	Facebook	define by run	7
Keras	François Chollet	—	6

まずGoogleのTensorFlowですが，これはdefine and run形式と呼ばれ，まず計算すべきプロセスを全てグラフで表現し（define），その後で（and），実際にデータの値などをグラフに入力し，様々な処理を実行します（run）．たとえば足し算を実行する場合，a と b を用意し，出力は $a+b$ とする，と書いた後に「計算（run）用の環境」にモードを切り替えて，$a=1$，$b=2$ を入れ，答え3を得る，というような手順を取ります．

一方でFacebookのPyTorch等はdefine by run形式と呼ばれ，define and run形式でやっていた処理をすべて同じレベルで取り扱います．実際に書く情報は変わりませんが，こちらでは「計算用の環境」というのはなく，定義と実行を同じ環境で行います．（ちなみにTensofFlowでもdefine by run形式が使えるようになったようです．）どちらがわかりやすいかは人によると思います．

この二つはいずれもライブラリ内の「お作法」をある程度学習する必要がありますが，Kerasは学習コストが二つに比べて圧倒的に低く，初めに使ってみるのに非常におすすめのライブラリです．しかも，もっと込み入った処理を書くこともでき，柔軟性も二つのライブラリに匹敵します．そのためKerasだけ使い続けるというので全く問題はありません．しかし，もし最新の論文の手法などをいち早く取り込みたい場合，論文の著者が必ずしもKerasを使っているとは限りません．そういう意味でTensorFlowとPyTorch又は他のライブラリも余裕があれば使えるようになっていると良いと思われます．

参考文献

1）Y. LeCun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015).
2）I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016).
3）E. Hoffer, I. Hubara, and D. Soudry, Advances in neural information processing systems (2017) p. 1731.
4）G. F. Montufar et al., Advances in neural information processing systems (2014) p. 2924.
5）I. J. Goodfellow et al., Advances in neural information processing systems (2014) p. 2672.
6）F. Chollet et al., https://keras.io/ja/.
7）Facebook, https://pytorch.org.
8）Google, https://www.tensorflow.org/?hl=ja.
9）M. Arjovsky et al., arXiv:1701.07875 (2017).
10）瀧　雅人，『これならわかる深層学習入門（機械学習スタートアップシリーズ）』（講談社，2017）．

著者紹介

瀧　雅人氏：　元の専門は超弦理論．この数年の研究対象は深層学習．医療・科学データへの応用から理論的研究まで雑食性で行っている．趣味は古墳鑑賞，化石の採取とクリーニング．

田中章詞氏：　元の専門は超対称ゲージ理論．最近は機械学習や深層学習を物理学の問題に応用する研究を行っている．

（2018年9月19日原稿受付）

Deep Learning for Physicists

Masato Taki and Akinori Tanaka

abstract:　Basic concept and few advanced topics of deep learning are reviewed. We explain supervised machine learning, neural nets, optimization with SGD, theoretical reason to make network deep, adversarial nets, an open question on generalization, and deep learning libraries.

Vol. 25−2（7月15日発行）目次