知的情報処理の統計力学 ―機械学習を始めてみよう―

大関真之 *1
京都大学 大学院情報学研究科システム科学専攻

概要

何か機械学習って流行っているらしい．どういうものかちょっと知りたい．知っているつもりだけど、実はちゃんとあんまり考えたことがない．そう言った聴衆を相手に夏の学校で集中ゼミの話題として「機械学習」を取り上げました．講演とむやみやたらにリンクさせるつもりはありません．少しやってみる気になった時に手がかりとなるように書いた記録です．細かいことは書いていません．これを目にして、機械学習そのものだけではなく、ご自身の分野に役立つものやアイデアが生み出されることを期待しています．

1 機械学習って何？

機械学習という言葉が、今やニュースでも目にするようになりました．機械が学習する、という言葉から人工知能との関係も注目されて、なお一層ブームの熱が高まっているのでしょう．機械学習というのは、

「こういうルールで世の中はできているから、そのルールを学びとってください．」

という指令をコンピュータにおくることです．このルールの抽出を学習と言います．そしてその学びとった結果を利用して、世の中の動きを予測したり、新しい事例が来たときに今までと同じかどうか判断をするわけです．学びとったルールを駆使して、そこで始めて自立するわけです．だけど自立するならもっと早い段階でして欲しいですよね．それこそ**自学自習**して欲しいものです．ここに現代的な機械学習のブレークスルーがあります．

まずは現代的な話の前に、機械学習の学習部分、どうやってそれを可能にするのか？特に物理学との相性に優れた統計的機械学習について、この原稿では触れたいと思います．統計的機械学習では、世の中のデータに対してのある仮説を基本としています．それは、

「データは、ある法則に従って確率的に生じている．」

というものです．データ自体は、無味乾燥な数値の配列にすぎないけど、それをうまく読み取ることで、顔だと判別できるかもしれない．その読み方と法則・ルールが密接に関係しているというわけです．その法則に従いつつも、映り方によっては全くその人の顔かどうかは見えなくなってしまいます．一種の乱雑な変形を受けています．この乱雑で不確実な事情を反映して、データは多様なものが存在すると考えられるから上記の仮説はもっともかな、というわけです．

機械学習の習得に必要な数学は、画像から顔を生み出している絶妙な組み合わせを表現する**線形代数**や、学習の根幹となる最適化問題の計算方法として**微分・積分**の知識．そして初歩的な**確率**で十分です．なんだ、じゃあ自分にもできるんじゃないか？僕は高校生でもできると思っています．

ようこそ、機械学習が拓く新時代へ．そこには誰もが機械学習を実装している社会があると思います．

*1 E-mail: mohzeki@i.kyoto-u.ac.jp

2 データの背後にある関数を掴むための機械学習：回帰

何か実験をやってみた結果、得られたデータの背後に何か法則がありそう．グラフにしてみると何か特徴が見える．関数の存在だ．直線だったら定規を当てれば良いが、曲線だったら一筋縄ではいかない．なぜか実験の授業では先生が、「二次関数でフィッティングしろ」っていうけど、それはおかしい．なんで事前に予想できるような自明な実験のデータ解析をさせるのでしょう？新しいデータがどんな関数形をしているかなど非自明なはずです．これをコンピュータに任せて自動的に行うようにしよう．このような解析を**回帰分析**と言います．

まずたくさんのデータがあり、x 座標 $(x^{(1)}, x^{(2)}, \cdots, x^{(D)})$ と y 座標 $(y^{(1)}, y^{(2)}, \cdots, y^{(D)})$ が得られたとします．単なる直線ではなく曲線なら、二次関数かもしれないし、三次関数、四次関数といろんな関数が組み合わさっている場合もあるでしょう．他にも三角関数や対数関数、指数関数といくつもの関数の組み合わせも考えられます．そう言った関数を $f_1(x)$, $f_2(x)$, $\cdots$, $f_N(x)$ として、

$$y = a_0 + a_1 f_1(x) + a_2 f_2(x) + a_3 f_3(x) + \cdots + a_N f_N(x) = \sum_{k=0}^{N} a_k f_k(x) \tag{1}$$

という関係を妄想することができます．最適な係数 $\mathbf{a} = (a_0, a_1, a_2, \cdots, a_N)$ を見つけるというのが目標です．ここで $f_0(x) = 1$ としています．この様子は、波数 k が異なる三角関数 $f_k(x) = \sin(kx)$ を用いた場合では、奇関数に対するフーリエ級数展開をしていることに相当します．いわば**フーリエ級数展開装置を作っている**というわけです．$f_k(x) = x^k$ とすれば、自動テイラー展開装置となります．

でも測定誤差があったりしたらどうでしょう．理想的な観測値からズレてしまいます．そのような場合、実際に得られる y の値は、自分の指定した式から誤差 z 分はずれたものとなるはずです．

$$y = z + \sum_{k=0}^{N} a_k f_k(x) \tag{2}$$

ずれるというのも正の方向でずれたり、負の方向でずれたり様々ですから、簡単のため平均 0、分散 σ^2 のガウス分布に従うものと想定しましょう．

■条件付き確率によるモデル化　誤差が平均 0、分散 σ^2 のガウス分布に従っているということは、確率変数 z が従う確率分布関数が、

$$g(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{z^2}{2\sigma^2}\right) \tag{3}$$

ということです．一方で、誤差 z に注目してみると、$z = y - \sum_{k=0}^{N} a_k f_k(x)$ ですから、x と y の関係にガウス分布で決まる誤差という不確実性が含まれていることがわかります．つまり単純な**決定論的関数関係にはない**．そういう時に利用するのが**条件付き確率**です．

$$P(y|x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - \sum_{k=0}^{N} a_k f_k(x))^2}{2\sigma^2}\right) \tag{4}$$

ここで x と y の確率的関係を決めるものを**パラメータ**と呼び、この場合 $\mathbf{a}$ と σ^2 がそれに相当します．

■最尤推定　得られたデータは、実際に起きたこと．その**起こったことが起きる確率こそが最も大きいから生じた**のだ、と考えるのが**最尤推定**と呼ばれる事実に忠実な推定方法です．データとは起こった事実の羅列ですから、ここで起こった事実が起きる確率を計算してみましょう．

$$\prod_{d=1}^{D} P(y = y^{(d)}|x = x^{(d)}) = P(y = y^{(1)}|x = x^{(1)}) \times P(y = y^{(2)}| = x^{(2)}) \times \cdots \times P(y = y^{(D)}|x = x^{(D)}) \tag{5}$$

となります．パラメータが変数としてまだ残っていますから、パラメータについて、何かを言っている量であると言えます．このように起こったイベントに関する確率の積を、**パラメータの尤度**と呼びます．対数をとっ

たものを**対数尤度関数**と呼び、よく利用されます.

$$f(\mathbf{a},\sigma^2)=\log\prod_{d=1}^{D}P(y=y^{(d)}|x=x^{(d)})=-\frac{1}{2\sigma^2}\sum_{d=1}^{D}\left(y^{(d)}-\sum_{k=0}^{N}a_kf_k(x^{(d)})\right)^2-D\log\sqrt{2\pi\sigma^2} \quad (6)$$

意外にも簡単な二次関数に帰着しました. この対数尤度関数の最大化を行うことが**最尤推定**です. σ^2 に関する最大化はすんなり解くことができますね. ちょうど誤差に相当する部分の二乗の経験平均になります. さらに $\mathbf{a}$ に関する対数尤度関数 $f(\mathbf{a},\sigma^2)$ については、マイナスのかかっている部分を小さくすれば対数尤度関数自体は大きくなるので、同じ意味を持つ最小化問題は

$$\min_{\mathbf{a}}\left\{\sum_{d=1}^{D}\left(y^{(d)}-\sum_{k=0}^{N}a_kf_k(x^{(d)})\right)^2\right\} \quad (7)$$

ということがわかります. これはいわゆる**最小二乗法**と呼ばれる回帰分析の基本手法です.

■勾配法 最尤推定を実行するには、$\mathbf{a}$ に関する微分をして、傾きが 0 となるところを探せば良いです. 上記の回帰の場合は手で解けます. もしもそれができないような難しい問題であっても、**勾配法**と呼ばれる自動的に山を登る方法を採用することで実現できます. 対数尤度関数 $f(\mathbf{a},\sigma^2)$ の最大化を目指す時に、適当な値 $\mathbf{a}$ において、傾きが正であれば、パラメータを正の方向に動かします. 逆に傾きが負であれば、そのまま行くと下ってしまうので、パラメータを逆の方向に動かせば良いでしょう. このことからパラメータ $\mathbf{a}$ を以下のように繰り返し更新をすれば良いことがわかります.

$$\mathbf{a}'=\mathbf{a}+\alpha\frac{\partial}{\partial\mathbf{a}}f(\mathbf{a},\sigma^2) \quad (8)$$

ここで $\mathbf{a}'$ が更新後の値です. α は更新幅で適当な値を設定しましょう. 微分のところは grad を計算するといった方が物理の人は通りが良いかもしれません. 更新幅が小さいとゆっくりと進んでいくため効率が悪いものの、正確に傾きが 0 になるところを探してくれます. 一方で更新幅が大きいと、どんどん進んでいきますが、最大値を通り越してしまうことがありますので注意してください. ただし傾きが 0 となるところですから一般には、極大値を探し出していることにご注意ください.

■補助関数法 この勾配法のパラメータ更新の意味を別の観点から考えてみましょう. ある関数 $f(\mathbf{x})$ の最大化を考えていて、ある点 $\mathbf{a}$ での傾きが $\mathbf{g}$ だったとしましょう. (多変数ということは、いろんな方向への傾きがあるのでベクトルです.) こんな時に騙されたと思って次の関数の最大化問題を考えてみます.

$$h(\mathbf{x})=f(\mathbf{a})+\mathbf{g}\cdot(\mathbf{x}-\mathbf{a})-\frac{1}{2\alpha}\|\mathbf{x}-\mathbf{a}\|_2^2 \quad (9)$$

この二次関数 $h(\mathbf{x})$ を**補助関数**と呼びます. ここで $\cdot$ は内積を取るということになります. また $\|\cdot\|_2$ というのはユークリッドノルム、L_2 ノルムのことであり、各成分の二乗和をとりルートをしたものです. $h(\mathbf{a})=f(\mathbf{a})$ ですから、ある点 $\mathbf{a}$ で元の関数と同じ関数値を取り、さらに傾きも同じです. **異なるのは曲がり方**だけです. その曲がり方を決めているのが、とってつけたような α です. じっくり眺めてみると、$\mathbf{x}$ については、単なる二次関数ですから、最大となるところは、

$$\mathbf{x}=\mathbf{a}+\alpha\mathbf{g} \quad (10)$$

であることがわかります. ある点 $\mathbf{a}$ から $\mathbf{g}$ を係数 α 倍した分だけ進め、そこに最大値があるよ、と言っています. 勾配法の更新の式と全く同じことがわかります. とってつけた α は更新幅に対応しているわけです. つまり勾配法は、上記の二次関数の最大化を**逐次的に行っている**ということがわかります.

α を小さくすれば確実に勾配法で最尤推定を行うことができました. 一方 α を大きくすれば、必ずしも勾配法では最尤推定がうまくいかないということでした. 二次関数の形状で見ると、α が小さい時は尖って膨らむのに対して、α が大きいと大きく膨らみます. 参考にしている $f(\mathbf{x})$ を超えるか、下回るかが α によって変わります. (図 1) うまく α を選べば、$f(\mathbf{x})$ を下から抑えるので、補助関数の最大化を行えば $f(\mathbf{x})$ の最大値を通り過ぎることがありません. どんな対数尤度関数となっていても、二次関数の最大化と同一視できるので便利です. というわけで、基本的には**二次関数の最大化**を考えれば良いことがわかります.

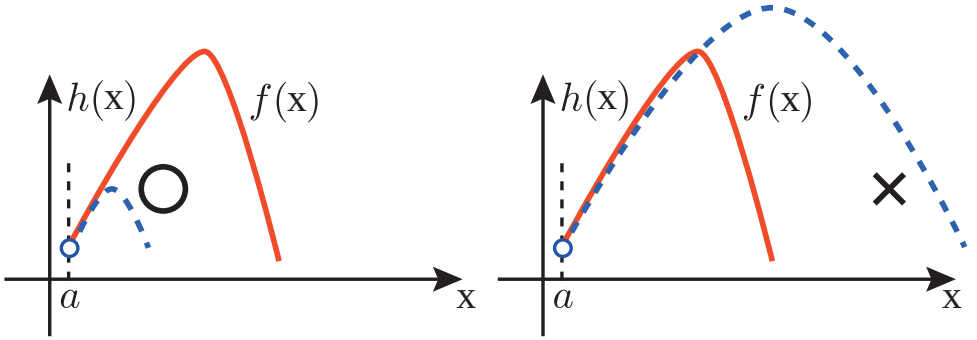

図 1　補助関数の様子．実線が元になる関数．破線が補助関数．左は更新幅が小さい場合．右は更新幅が大きい場合．

■変数選択　最尤推定は、起こった事実こそが全てという精神から来ています．しかし実際には、データが少ないためあんまり信用するのも心許ないです．例えばコインを投げて 3 回続けて表が出ても、コインの表が出る確率が 1 とは考えにくいです．しかし最尤推定ではそういう結果となります．**過適合**または**過学習**という現象です．表が出る確率 1 という推定結果を不自然だと思うのは、こうであるべきだ、という姿を知っているからです．こういうときに利用されるのが**正則化**という手法です．対数尤度関数に、ある項を追加して、一緒に最大化を考えてみましょう．

$$\max_{\mathbf{a},\sigma^2} \left\{ f(\mathbf{a},\sigma^2) - \lambda \|\mathbf{a}\|_1 \right\} \tag{11}$$

この第二項が大きくなるには、係数 $\mathbf{a}$ が $\mathbf{0}$ になる必要が有ります．つまり係数を $\mathbf{0}$ にする効果があります．L_1 ノルムと呼ばれる正則化項です．$\|\cdot\|_1$ はベクトルの各成分の絶対値の和を取るという意味です．同様に、よくある正則化として $\|\cdot\|_2$ の二乗があります．ただしこれは係数を $\mathbf{0}$ にする性能には乏しいという違いが有ります．この L_1 ノルムによる正則化を行うと、係数がほとんど不要、ほとんどがゼロだ（**スパース**）だという事前知識（または願い）を反映させたものが解として得られます．準備のために以下の最大化問題を考えてみましょう．

$$\max_x \left\{ -\frac{1}{2}(x-v)^2 - \lambda|x| \right\} \tag{12}$$

え、実際の最尤推定はそんな簡単な最大化の形をしていないって？先ほどの補助関数法を思い出してください．どんなものも二次関数の最大化に置き換わります．それが第一項．第二項は L_1 ノルムによる正則化から来ている絶対値の項です．え、多変数関数でベクトルのノルムなんか考えていたじゃないかって？各成分で独立に考えてよいじゃないですか．どれも同じ格好なんだし．これを**分離性**と言います．

絶対値関数が第二項にありますので、一見難しそうと思うかもしれませんが場合わけをして、手で解くことができます．やってみましょう．すると結果として、以下のような 3 つの場合が存在することがわかります．

$$x = S_\lambda(v) \equiv \begin{cases} v-\lambda & (\lambda \le v) \\ 0 & (-\lambda \le v < \lambda) \\ v+\lambda & (v < -\lambda) \end{cases} \tag{13}$$

ここで $S_\lambda(v)$ というものを軟判定閾値関数（Soft-thresholding function）と呼びます．そもそも v というのは、二次関数部分のみに注目したときの頂点の場所、つまり最大値の場所です．それを絶対値関数を導入した途端に、$-\lambda \le v < \lambda$ **の範囲にあるなら** 0 **にしてしまう**というわけです．これは頂点の位置が原点に近いなら、いっそのこと 0 にしてしまおうというわけです．

回帰分析の対数尤度関数に L_1 ノルムでの正則化を適用するには、対数尤度関数の代わりに補助関数を使いましょう．補助関数の頂点 $\mathbf{x} = \mathbf{a} + \alpha\partial f(\mathbf{a},\sigma^2)/\partial\mathbf{a}$ に同じ議論を適用すれば良いので、

$$\mathbf{a}' = S_{\alpha\lambda}(\mathbf{x}) = S_{\alpha\lambda}\left(\mathbf{a} + \alpha \frac{\partial}{\partial \mathbf{a}} f(\mathbf{a},\sigma^2) \right) \tag{14}$$

とすれば、勝手に数少ない重要な変数だけが残ってくれるというわけです．このように数少ない重要な変数を選ぶことを**変数選択**と呼びます．この方法を知っていれば、よく分からないデータが出てきた時に、適当な

数少ない関数を選択して適切にフィッティングをしてくれることがわかります．どんな関数形か、人間（先生？）が想像つかない時に、とりあえず知る限りの関数を適当に並べるとしましょう．この技術を利用すれば、機械が自立して勝手に選んでくれるというわけです．**人間が苦労から解放されて、機械が自立して学習する瞬間**です．このように重要な部分を見抜くための仕掛けを用意して、データから自動的に重要な物事を抽出する所作を**スパースモデリング**と呼びます．

この原稿も、最初とにかく詰め込んで、10 ページまでという正則化をすることで本質部分だけ残しました．

3 データの特徴量を掴む機械学習：識別

機械学習ができることの代表的なものに回帰の他に、**識別**があります．犬と猫を分ける、といった異なるものを区別して、どちらのグループに所属するのかということを判定することを識別といいます．それでは画像データに基づいて、犬か猫かなど動物を識別することを考えましょう．画像の画素値をベクトル $\mathbf{x} = (x_1, x_2, \cdots, x_N)$ とします．なんとなく人間がみるときには、目のような形があり、耳があり、その画像をみて総合的に判断しますから、画素値の組み合わせから犬か猫かを識別できるような適当な重みつきの和、つまり内積 $\mathbf{p}\cdot\mathbf{x}$ を考えたらよいのではないでしょうか．$\mathbf{p}$ を結合の重みといいます．これはつまり犬っぽいかどうか猫っぽいかどうかということです．そして犬か猫かの境目（**分離超平面**）を調べることで識別をしようというわけです．これが Rosenblatt が考えたパーセプトロンのアイデアです．ご存知かもしれませんがニューラルネットワークの基礎であり、最近隆盛を極めるいわゆる**深層学習（Deep learning）**の原型です．

分離超平面を得るためには様々な手法がありますが、ここでは統計的機械学習の方法に準じて考えてみましょう．データは何かの法則に従って確率的に生じている、と考えるのが統計的機械学習の基本でした．そうなると犬なり猫なりを指定したらその画像データが生じる確率を考えてみようということになります．もちろんそんなものを僕らは知らないから、**ある程度の確率モデルを設計して、詳細はデータから学び取る**わけです．

犬の画像であれば、犬の特徴的なパターンを出力できればよいわけですから、$\mathbf{w}_1 = (w_{11}, w_{12}, \cdots, w_{1N})$ を犬のパターンとして、その値に従って画像データの画素値が選ばれるとしましょう．パターンという言葉は非常に抽象的です．たとえばこのパターンというものが、犬の画像の平均値を定めているとしましょう．そして画像はその平均値からガウス分布に従って生じているとします．どの動物かを指定する数字 z という入力を使って、条件付き確率を書いてみましょう．

$$P(\mathbf{x}|z) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left\{-\frac{1}{2\sigma_z^2}\left(\mathbf{x}-\mathbf{w}_z\right)^2\right\} \tag{15}$$

とすれば良いことがわかります．分散もそれぞれの動物ごとに異なるとしましょう．簡単のため同じ動物内の分散を共通にしましたが、共分散行列を導入してもっとリッチな構造を考えても方向性は同じです．

先ほどの回帰の例と同様に、データに基づいた最尤推定をやってみましょう．画像 $\mathbf{x}^{(d)}$ をいくつか得て、それが犬か猫かにわとりかなど動物の指定を $z^{(d)}$ も知っているとしましょう．その時に実際に起こったイベントが起きる確率の積を計算するとパラメータの尤度が計算できます．さらにその対数をとった、対数尤度関数を計算すると以下のものが得られます．

$$f(W, \boldsymbol{\sigma}) = \log \prod_{d=1}^{D} P(\mathbf{x} = \mathbf{x}^{(d)}|z = z^{(d)}) = -\sum_{d=1}^{D}\left\{\frac{1}{2\sigma_{z^{(d)}}^2}\left(\mathbf{x}^{(d)} - \mathbf{w}_{z^{(d)}}\right)^2 + \frac{1}{2}\log\left(2\pi\sigma_{z^{(d)}}^2\right)\right\} \tag{16}$$

実際に起きたことが一番確率が高いはずだという信念のもと、最尤推定を実行すれば、識別をするためのパラメータ $W = (\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_M)$ と $\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \cdots, \sigma_M)$ を得ることができます．ここで M は識別したいパターン数、動物の種類の数としています．微分が 0 となるところがその解で、手で計算することができます．仮にその計算が難しい条件付き確率を設定したとしても、そのときは勾配法に任せましょう．

さていざ識別をするときは、どうしたらよいでしょうか．適切なパラメータを機械が学習して、その学習の成果を利用しようというわけです．学習では、動物を指定してどんな画像が出るのかを示す条件付き確率を考えました．その詳細に当たるパラメータを推定したわけです．学習の成果を試す場面では、画像を入力して、

何の動物かを指定して欲しいので逆のことをしなければなりません．つまり $P(\mathbf{x}|z)$ から $P(z|\mathbf{x})$ へとひっくり返したいというわけです．

■ベイズの定理　条件付き確率をひっくり返すためには、ベイズの定理を使います．2 つ以上の異なるイベント（例えば $\mathbf{x}$ と z）があるとき、それが発生する確率を記述するものを同時確率 $P(\mathbf{x}, z)$ と言います．それに対して、順番に何か立て続けに順番に起こるとして分解して書いたものが条件付き確率 $P(\mathbf{x}|z)$、$P(z|\mathbf{x})$ です．それらの間には

$$P(\mathbf{x}, z) = P(\mathbf{x}|z)P(z) = P(z|\mathbf{x})P(\mathbf{x}) \tag{17}$$

という定義的な関係が成立します．これを利用すると、

$$P(z|\mathbf{x}) = \frac{P(\mathbf{x}|z)P(z)}{P(\mathbf{x})} \tag{18}$$

と条件付き確率の入力と出力の関係をひっくり返すことができます．これを**ベイズの定理**といいます．ひっくり返した条件付き確率を、**事後確率**と呼びます．この事後確率を利用するときには、条件にあたる部分は $\mathbf{x}$、今の場合ではもらった画像ですから起こった事実．その事実が起きた後の確率的事象について言及しているため、事後確率というわけです．右辺にでてくる $P(z)$ は、これから出てくる z の候補について、事前にどんな動物がでてくるかの可能性について言及しているので、**事前確率**と呼ばれます．たとえば犬と猫はよく見るけど、コンドルはあまり見ない、コンドルに相当する z のところの確率は低いだろうといった事前知識を入れることで推定の特性を変えることができます．この事後確率の最大化により、尤もらしい z を推定することを**事後確率最大化法**と呼びます．今の場合は、ある画像 $\mathbf{x}$ をもらったときに、どの動物に近いかを識別するための z に関する最大化ですから、対数を取ったのちに z に関係する項のみに注目すれば、

$$\max_z \{\log P(\mathbf{x}|z) + \log P(z)\} \tag{19}$$

を考えればよいということになります．学習したパラメータと入力された画像 $\mathbf{x}$ という事実にもとづく、対数尤度関数が第一項に再び登場しました．さらに事前確率として事前情報の影響を考慮する正則化項が登場しました．先ほどの変数選択ではパラメータに関する正則化でしたので、**学習時の工夫**でした．ここでは推定したいものについての正則化ですから、学習の成果を発揮する**実践の場面での工夫**であるという違いがあります．

簡単のため犬か猫かふたつにひとつという識別をするときは、$z = 1$ のときと $z = 2$ のときの大小比較をすればよいので、それぞれの事後確率が等しくなるところが犬か猫かの境目の分離超平面 $\mathbf{p} \cdot \mathbf{x} + q = 0$ となります．これを **Fisher の線形判別分析**といいます．入力された画像データ $\mathbf{x}$ に重み付きの和を計算することで犬っぽいか猫っぽいかを計算する識別器ができた、というわけです．

■特徴量と深層学習　識別器をつくるアルゴリズムは他にもあり多彩を極めますが、さてこれで本当に良い識別器は作ることができるでしょうか？実はできません．残念なことに失敗に終わりました．考えてみると当たり前です．画素一つ一つに意味があるでしょうか？ありません．犬の画像は回転したら画素単位では異なる画像ですが、同じ犬です．画素ひとつひとつを見て、オムライスとオムレツの区別がつくでしょうか？もちろん研究者たちはそれに気付き、画像を直接入力するのではなく、何か構造を先に捉えて、それの有無や何かの基準に照らした数量など前処理をして入力したほうがよいのではないかということを提案しました．**特徴量**と呼ばれるものです．問題はどんな特徴量を入れると良いかがわからない．人間が工夫を凝らしてなんとかしていた時代があります．やっぱり機械は人間におんぶにだっこ．それなら入力を適当に組み合わせて、その組み合わせをさらに組み合わせることを繰り返すことで、**特徴量すらも学習**をしようじゃないか．まさに自学自習です．これが深層学習の基本アイデアです（図 2）．入力 $\mathbf{x}$ から途中の変換を通じて、何か特徴量を抽出したのち、その特徴量の組み合わせで識別をするというわけです．しかし多層の場合には、入力から出力までが遠いという問題があり、勾配法を適用しても勾配が小さくなる**勾配消失問題**が発生して、効果的に学習を行うことが困難でした．そのため人々は多層を利用した識別の実現を諦め、異なる方向性による識別精度の向上を目指したというわけです．それが単層でも性能の良いものを作ろうという方向性の**サポートベクターマシン**や**カーネル法**と呼ばれる方法です．

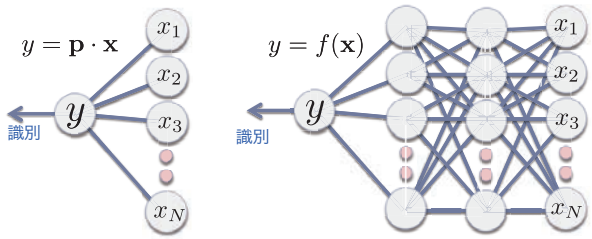

図 2　精度の良い識別器に向けた多層化．左が単層の識別器．右が多層の識別器．途中は特徴量抽出器となることが期待される．

それでは、統計的機械学習の話に戻りましょう．回帰も識別も機械学習の基本なわけですが、どちらも同じように条件付き確率を設計して、得られたデータに基づくパラメータの対数尤度関数を計算して、最大化するという同じ手続きでなされるものだとわかりました．こうやって共通する概念めいたものが見えてくると物理学者は、途端に統一理論を作りたくなります．その統一理論が、ボルツマン機械学習に相当します．

4　ボルツマン機械学習

ボルツマン機械学習の本来の目的は、多次元ベクトルで表現される高次元データが与えられたときに、そのデータを創りだしている本質部分である、**生成モデル**を明らかにすることです．生成モデルは、データそのものを発生させる確率分布のことを指します．先ほどの例であれば、犬と猫などの画像をとにかく生成する確率モデル $P(\mathbf{x}, z) = P(\mathbf{x}|z)P(z)$ を明らかにするということです．この生成モデルが明らかとなれば、同様の性質を有する高次元データを自在に用意することが可能であり、また生成モデルそのものの特徴から、データの背後にある本質に迫ることができるというわけです．

ボルツマン機械学習では、統計力学で基本となるカノニカル分布に従ってデータ $\mathbf{x}$ が出力されると考えます．

$$P_{\mathbf{u}}(\mathbf{x}) = \frac{1}{Z_{\mathbf{u}}} \exp\left(-E_{\mathbf{u}}(\mathbf{x})\right). \tag{20}$$

ここで $Z_{\mathbf{u}}$ は規格化定数としての分配関数で、$\mathbf{u}$ はデータの構造を表すエネルギー関数 $E_{\mathbf{u}}(\mathbf{x})$ を形作るパラメータです．上記の仮定のもと、与えられた大量のデータの経験分布

$$P_D(\mathbf{x}) = \frac{1}{D} \sum_{d=1}^{D} \delta(\mathbf{x} - \mathbf{x}^{(d)}) \tag{21}$$

に最も近いカノニカル分布を探してくることがボルツマン機械学習の目標です．

■KL 最小化と最尤推定、そしてエントロピー最大化　さてそうなると 2 つの異なる確率分布を持ってきたときに、それらが近いか遠いかを調べるための計量が必要です．最も一般的に用いられるのがカルバック・ライブラー（KL）情報量です．

$$D_{\mathrm{KL}}(P|Q) = \int d\mathbf{x} P(\mathbf{x}) \log\left(\frac{P(\mathbf{x})}{Q(\mathbf{x})}\right) \tag{22}$$

$Q(\mathbf{x})$ を未知のパラメータを持つ $P_{\mathbf{u}}(\mathbf{x})$ に、$P(\mathbf{x})$ をデータの経験分布 $P_D(\mathbf{x})$ としましょう．このとき KL 情報量の最小化を考えます．パラメータ $\mathbf{u}$ が関与する部分に注目すると、以下の最大化問題と等価であることが分かります．

$$\mathbf{u}^* = \arg\max_{\mathbf{u}} L(\mathbf{u}). \tag{23}$$

ここで $L(\mathbf{u})$ は、

$$L(\mathbf{u}) = \frac{1}{D} \sum_{d=1}^{D} \log P_{\mathbf{u}}(\mathbf{x} = \mathbf{x}^{(d)}). \tag{24}$$

ここで対数尤度関数が再び登場しました．さらに少し計算してみると明らかとなるが、対数尤度関数はエネルギー関数（の経験平均）の符号を変えたものと、分配関数の対数（自由エネルギー）からなることが確認でき

ます．つまり統計力学の言葉でいえば、自由エネルギーとエネルギー（温度は 1 として）の差であるエントロピーが最大となるようなエネルギー関数の形を求めることが機械学習の核である最尤推定であることが分かります．

それでは最尤推定を実行するために、お得意の勾配法を試してみましょう．ここで対数尤度関数の微分が必要となるので、パラメータ $\mathbf{u}$ について対数尤度関数の微分を取ってみます．

$$\frac{\partial L(\mathbf{u})}{\partial \mathbf{u}} = -\frac{1}{D}\sum_{d=1}^{D}\frac{\partial E_{\mathbf{u}}(\mathbf{x}=\mathbf{x}^{(d)})}{\partial \mathbf{u}} + \left\langle \frac{\partial E_{\mathbf{u}}(\mathbf{x})}{\partial \mathbf{u}} \right\rangle_{\mathbf{u}}. \tag{25}$$

第 1 項は、エネルギー関数の形を知っていれば評価は容易です．一方第 2 項は、熱平均の計算 $\langle\cdots\rangle_{\mathbf{u}} = \sum_{\mathbf{x}}\cdots\times P(\mathbf{x}|\mathbf{u})$ が必要となります．これまでの例は、分配関数が計算できて熱平均も計算できる簡単な例だったというわけです．しかし一般には、適当な初期条件のもと、勾配法の計算を行おうとすると、何度も何度も異なるパラメータでの熱平均の計算が必要となります．これが**ボルツマン機械学習の運命**を決めました．正確な実行のために必要な計算量が非常に膨大となり、現実的な学習法ではないと、かつてその黎明期では判断されたのです．しかしながら 2000 年代に入ってから、データ数の大規模化と新しい計算手法の提案を契機に、現実的な計算手法であると受け入れられて今日に至ります．肝となるのは熱平均の計算部分です．そこに注力して統計力学の方法論を整備する方向性と、いやいや得られたデータが重要なのだからそれを利用するのが良いという賢い方向性があります．前者で有名なものは**平均場近似（Mean Field Approximation）**と**マルコフ連鎖モンテカルロ法（Markov Chain Monte Carlo method）**です．平均場近似のよいところはその計算の速さにあります．一方マルコフ連鎖モンテカルロ法では計算の速さは期待できませんが、近似がないので統計誤差を除いて厳密なメリットがあります．ここでは物理の人には馴染みのない、後者の方向性について紹介しておきましょう．

■疑似最尤法（Pseudo Likelihood Estimation**）**　平均場近似と似た精神を持っている簡単な仕組みでありながら、データ数 D が非常に大きい場合に漸近的に最尤推定と一致する解を得る手法です [1]．エネルギー関数が変数毎の和に分解できる（$E(\mathbf{x}) = \sum_{i=1}^{N} E_{\mathbf{u}}(x_i|\mathbf{x}_{/i})$．ここで $\mathbf{x}_{/i}$ は自由度 x_i の周辺の自由度）場合に、得られたデータを利用した近似を分配関数についておこなうことで得られます．

$$Z \approx \prod_{i=1}^{N}\sum_{x_i}\exp\left\{-E_{\mathbf{u}}(x_i|\mathbf{x}_{/i}=\mathbf{x}_{/i}^{(d)})\right\}. \tag{26}$$

周辺の自由度が期待値で置き換えるのが平均場近似ですが、疑似最尤法は $x_i^{(d)}$、つまり「得られたデータ」を周辺の自由度に置き換えるというところが大きく異なります．そのためデータが多いと、近似的な期待値で置き換える平均場近似よりも正確な期待値を用いるという理由で、擬似最尤法の方が有効となります．

■最小確率流法（Minimum Probability Flow**）**　尤度関数の代わりに新しいコスト関数を導入して、パラメータ推定を行う比較的最近の手法を紹介します．最小確率流法（Minimum Probability Flow）は疑似最尤法よりも収束性に優れた手法として知られています [2]．原理は単純です．連続時間のマスター方程式を考えます．初期条件は与えられたデータによる経験分布とします．そこから適当なパラメータ $\mathbf{u}$ により決まるボルツマン分布への緩和を考えてみます．このとき仮のパラメータが、与えられたデータに適合するものであれば、初期分布は定常分布に非常に近いものとなるはずです．そのためマスター方程式による分布の変動は非常に小さいものとなるでしょう．この考察に基づいて初期分布から微小時間経過したときの KL 情報量をパラメータについて最小化する方法が最小確率流法です．簡単な割に奥深く、そして性能が良いということで非常に興味深い手法です．

5　更なる複雑度への挑戦：深層学習

ボルツマン機械学習のエネルギー関数 $E_{\mathbf{u}}(\mathbf{x})$ を複雑にすればするほど生成モデルの表現の幅が大きく向上します．犬や猫の画像を表現しうるかもしれません．しかしそうなると、熱期待値の計算は困難となり、また

擬似最尤法や最小確率流法なども効果的ではなくなってしまいます．やはり難しくすればする程、大変なものとなってしまうわけです．この方向性は単純すぎました．やはり画素値そのものに意味はありません．その画素値同士のもつ構造など組み合わせたものが意味を持ちます．それを含ませるために、データを表す部分だけではなく、データとは直接関係のない**隠れた変数**部分同士が相互作用をすることで、間接的にデータの複雑さを示すことができるのではないか？と考えることにします．そうしてできたのが深層学習の基礎となる隠れ変数有り（多層の）ボルツマン機械学習です．

■制限ありボルツマン機械　データなどを代入する**可視変数**とは別に、入力されたデータを幾つか組み合わせたものを置くための人工的な変数として**隠れ変数**の導入を行います．さらに隠れ変数を層状に組み上げて複雑度をあげておきましょう．つまり $\mathbf{x}$ というデータを表すためには、可視変数部分だけでなく隠れ変数による冗長化を必要とするというわけです（図 3）．ここでは簡単のため一層に限った話をします．ここで難しくないギリギリを攻めるために、**可視変数同士、隠れ変数同士には結合を考えず**、可視変数と隠れ変数だけに結合がある**制限有りボルツマン機械**というものを考えます．

$$P_{\mathbf{u}}(\mathbf{v},\mathbf{h}) = \frac{1}{Z_{\mathbf{u}}} \exp\left(\sum_{i=1}^{N_v} b_i v_i + \sum_{j=1}^{N_h} c_j h_j + \sum_{i,j} W_{ij} v_i h_j \right). \tag{27}$$

ここで、$Z_{\mathbf{u}}$ は規格化定数で、パラメータは $\mathbf{u} = (W, \mathbf{x}, \mathbf{c})$ です．便宜的に N_v この可視変数、N_h 個の隠れ変数を用意しました．これらは可変です．この隠れ変数について和をとった周辺確率 $P_{\mathbf{u}}(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v},\mathbf{h})$ が生成モデルの役目を果たします．それぞれの変数の取りうる値域は様々なものがあります．ここでは簡単のた

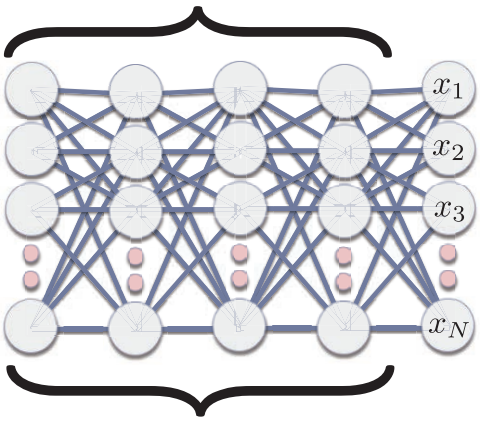

図 3　制限有りボルツマン機械学習を利用した深層学習．カッコで囲まれた左側の層が隠れ変数による層．可視変数はデータが入力される部分．

め、両者ともに ± 1 に限定します．さてここで隠れ変数について和を取り、可視変数のみの周辺分布を見てみましょう．

$$P_{\mathbf{u}}(\mathbf{v}) = \frac{1}{Z_{\mathbf{u}}} \exp\left\{ \sum_{i=1}^{N_v} b_i v_i + \sum_{j=1}^{N_h} \log 2\cosh\left(c_j + \sum_{i=1}^{N_v} W_{ij} v_i \right) \right\}. \tag{28}$$

可視変数について実効的に相互作用を持つようになります．ちょうど**実空間繰り込み群**の計算と同じです．つまり本来可視変数同士は結合がなかったものの、**隠れ変数を通して結合している**効果が現れています．同様にして、隠れ変数についての周辺分布を計算すると

$$P_{\mathbf{u}}(\mathbf{h}) = \frac{1}{Z_{\mathbf{u}}} \exp\left\{ \sum_{j=1}^{N_h} c_j h_j + \sum_{i=1}^{N_v} \log 2\cosh\left(b_i + \sum_{j=1}^{N_h} W_{ij} h_j \right) \right\}. \tag{29}$$

となります．これを用いて、隠れ変数を知った上での条件付き確率 $P_{\mathbf{u}}(\mathbf{v}|\mathbf{h}) = P_{\mathbf{u}}(\mathbf{v},\mathbf{h})/P_{\mathbf{u}}(\mathbf{h})$ を計算すると、

$$P_{\mathbf{u}}(\mathbf{v}|\mathbf{h}) = \prod_{i=1}^{N_v} \exp\left\{ b_i v_i + \sum_{j=1}^{N_h} W_{ij} v_i h_j - \log 2\cosh\left(b_i + \sum_{i=1}^{N_h} W_{ij} h_j \right) \right\}. \tag{30}$$

という格好になります．特徴的なのが、**可視変数について独立**となることです．可視変数を知った上での隠れ変数の条件付き確率についても同様に条件付き独立性を有した格好となります．このような条件付き確率における独立性を、**条件付き独立**と呼びます．この性質によりそれぞれの変数を交互にサンプリングすることが容易にできます．統計力学に登場する多くの模型で分配関数の計算や熱期待値を計算するためのアンサンブルを用意するのに苦心するのは、変数同士に相関があるためです．制限有りボルツマン機械学習では、可視変数と隠れ変数の間の交互のサンプリングにおいては、その問題がありません．計算が容易で、かつ表現力が豊かなギリギリの生成モデルというわけです．しかしこの**ギリギリの表現力も多層に組み上げれば無限の表現力を得る**ことができます．それが制限ボルツマンマシンの最大の特徴であり、これを利用して層状に同様の手続きを踏めば多層化が可能であり、深層学習への発展を促しました．

制限有りボルツマン機械学習を利用した深層学習では、実際に起こった事実に関する確率の積を、可視変数の条件付き確率から計算して、その対数尤度関数の最大化を目指すことでパラメータの学習を行います．

$$L(\mathbf{u}) = \frac{1}{D}\sum_{d=1}^{D} \log P_{\mathbf{u}}(\mathbf{v} = \mathbf{x}^{(d)}) \tag{31}$$

このとき勾配法を適用すると、熱平均値が必要となりますが、制限ありボルツマン機械の特性である条件付き独立性を利用することでサンプリングが比較的容易だから心配ご無用というわけです．この性質を利用したものが**コントラスティヴ・ダイヴァージェンス法（Contrastive divergence)（CD 法）**です [3]．CD 法では、データの経験分布を可視変数の初期条件として、可視変数と隠れ変数の “条件付き”サンプリングを交互に数ステップのみ（1 度きりでも良い性能！）行うことにより、熱期待値の計算を実行します．

6 そして今

CD 法の他にも、オートエンコーダと呼ばれる方法論が登場しました．オートエンコーダでは、**各層毎で入力データをそのまま出力するように学習する**ことで、とりあえず中間層を学習するというものです．本当の出力となる**いわば教師を無視した事前学習**を行うことで重みの最適化をとりあえず行います．その後出力と入力に矛盾がないように通常の最適化を行うことで、深層学習が実行できることがわかりました．じゃあ何が間違っていたのだろうか？その再検討もようやく終わりを迎え、勾配消失問題は、モデルを少し変えたり、非常に多くのデータと変数を利用すると回避できることがわかりました．勾配が小さくなるといっても鞍点だったなのでうまく更新の方向を選べばよかったことがわかってきました．多数のデータを扱うにあたり、データに関する和を計算するのもサボる**確率勾配法** [4] などの方法論も見直されました．そういう過去の方法論やアイデアが統合され練りあがっているのが、機械学習の分野の本当に面白いところなんじゃないかと思います．

統計機械学習の基本と最近の話題である深層学習のさわりまでを早足で紹介しました．これをきっかけに機械学習の理論的な部分に興味をもち、実践的な部分でご自身のデータ解析に生かしてもらえたら幸いです．

参考文献

[1] J. Besag: Journal of the Royal Statistical Society. Series D (The Statistician), **24** (1975) 179.
[2] J. Sohl-Dickstein, and P. B. Battaglino and M. R. DeWeese: Phys. Rev. Lett., **107**, (2011) 220601.
[3] M. Welling, and G. Hinton: Artificial Neural Networks, **2414**, (2002) 351.
[4] H.Robbins, and S.Monro: Annals of Mathematical Statistics, **22**, (1951) 400.