計算物理ワークショップ 機械学習編 その１

本編では近年そのさまざまな分野で社会実装が進んでいる機械学習について、その基礎を実習課題も交えながら学んでいく。まずは「学習」とは何かについて、おそらく実験科目でこれまで学んだことがあるであろうデータの最小二乗法フィッティングを例に学ぶ。物理学科であれば学部低学年時でも学ぶ最小二乗法による関数フィッティングは、実はもっとも簡単な機械学習法の１つである。その次に (その２、その３では) 近年の機械学習/人工知能ブームで最も注目されているディープラーニング (深層学習) について、その基礎的原理をいくつかの簡単な実践コードを使いながら習得を目指していく。

最小２乗法と「学習」

最小２乗法とは、ある与えられたデータの組みがあるときに、そのデータを表現する最も確からしい関係式を、仮定された関係式とデータの間の誤差の２乗和を最小にするように決定する方法である。本節では簡単な例として次のような１次関数を「確からしい関係式」の候補として考える：

$$g(\boldsymbol{w}, x) = w_1\, x + w_2. \tag{1}$$

ここで $\boldsymbol{w} = (w_1, w_2)$ は未定の係数である。データの組 $(\boldsymbol{x}, \boldsymbol{y}) = (x_1, x_2, \cdots, x_N, y_1, \cdots, y_N)$ が与えられたときに、上式との誤差の２乗和 $f(\boldsymbol{w})$ は次のようにかける

$$f(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}\{g(\boldsymbol{w}, x_i) - y_i\}^2. \tag{2}$$

いまの目的はこの誤差関数 f を最小にするパラメーターの組 $\boldsymbol{w}$ を探すことであるから, f を w_1, w_2 で偏微分した次の２つの式が共に 0 になるパラメーター空間上の点を探せば良い

$$\frac{\partial f(\boldsymbol{w})}{\partial w_1} = \frac{1}{N}\sum_{i=1}^{N} 2\{g(\boldsymbol{w}, x_i) - y_i\}\frac{\partial g(\boldsymbol{w}, x_i)}{\partial w_1} = \frac{1}{N}\sum_{i=1}^{N} 2\{g(\boldsymbol{w}, x_i) - y_i\}x_1 \tag{3}$$

$$\frac{\partial f(\boldsymbol{w})}{\partial w_2} = \frac{1}{N}\sum_{i=1}^{N} 2\{g(\boldsymbol{w}, x_i) - y_i\}\frac{\partial g(\boldsymbol{w}, x_i)}{\partial w_2} = \frac{1}{N}\sum_{i=1}^{N} 2\{g(\boldsymbol{w}, x_i) - y_i\}. \tag{4}$$

どのようにして最適な $\boldsymbol{w}$ を探せば良いか？ここでは機械学習への応用を考えて勾配降下法という方法を用いる。勾配降下法では次のような $\boldsymbol{w}$ に対する微分方程式を解いていくことを考える。

$$\frac{d\boldsymbol{w}}{dt} = -\alpha \boldsymbol{\nabla}_w f(\boldsymbol{w}) = -\alpha\left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}\right). \tag{5}$$

誤差関数 $f(\boldsymbol{w})$ を空間 (w_1, w_2) における地形ととらえて、その地形の勾配を下ることで底に向かうのである。(5) 式を差分化すると

$$w_{1,2}^{(n+1)} = w_{1,2}^{(n)} - \eta\frac{\partial f}{\partial w_{1,2}} \tag{6}$$

を得る。ここで $\alpha\,\Delta t = \eta$ とおいた。この漸化式を用いて適当に類推したパラメーター $(w_1^{(0)}, w_2^{(0)})$ を出発点として “坂を下っていけば”いつか最適な $\boldsymbol{w}$ が見つかるはずである。

ここで紹介した勾配降下法を実装した python プログラムが次のリンクからダウンロードできる。

`https://colab.research.google.com/drive/1THuUIFZzW4Wi1p1RksQfXt80CovC0JBl?usp=share_link`

このファイルは Jupyter Notebook と呼ばれるアプリ用のファイルであるが、Google アカウントを作成してブラウザ経由で google ドライブにファイルをアップロードしダブルクリックすれば特別な環境設定なしですぐに Python スクリプトを動かすことができる。

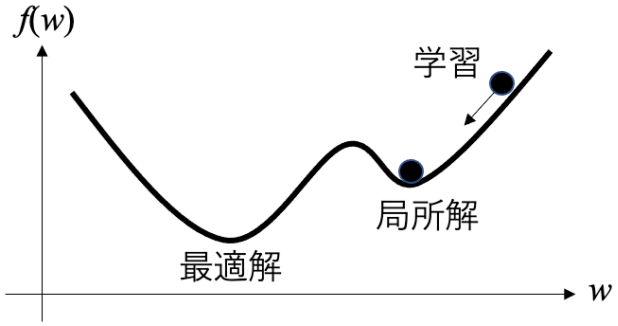
コードが動かせたら次のことを確認/考察してみよう。

- 何回くらいのステップで収束するか？
- サンプルプログラムでは η を w_1 用と w_2 用で異なるものを使っているが、それはなぜか？

サンプルプログラムで体験したように、目的とする状態へと試行 (今の場合は勾配を下ること) を重ねながら最適解を探すことを「学習」と呼ぶ。今の場合は勾配降下法により学習を重ねた結果、２乗誤差関数を最小にする関数 $g(\boldsymbol{w}, x)$ が学習できたことになる。

より効果的な学習法

1. 確率的勾配降下法 (Stochastic Gradient Descent; SGD): 上の例で見たような停留点が１つだけの簡単な誤差関数の場合には単純な勾配降下法でも時間をかければ必ず学習は成功する。しなしながら、問題によっては停留点がいくつもあり、その中で誤差関数の最小値を与える最適な解が１つしかない場合は、初期条件によっては最適解に到達する前に局所的な極小解にトラップされて学習がそこで終わってしまう場合がある (下図参照)。こういった場合の対処法としては問題に合わせて初期条件を変更することが考えられるが、問題ごとに最適な初期条件を探るのでは時間がかかりすぎる。そこで考えられるのは、学習に確率的なノイズを加えて局所解から脱出できるようにする方法である。SGD 法では (2) 式で表される誤差関数を計算する際に、すべてのデータ i に関する平均を取るのではなく、ランダムに選んだ一部のデータに対してのみ平均を取ることで誤差関数に確率的なノイズを加えて勾配効果の挙動をわざと乱すことを考える。誤差関数の評価に一部のデータのみを使用することによる計算量削減の効果も期待される。



2. Momentum 法:SGD 法では局所的な勾配だけで移動先を決めるが、Moment 法では過去の履歴も引きずりつつ局所勾配の情報で軌跡を修正し、かつ平衡点近傍で振動しないように減速させる。高度な方法のように聞こえるかもしれないが、物理とのアナロジーで考えれば容易に理解することができ、次のような摩擦入りの運動方程式を考えることに相当する

$$\frac{d\boldsymbol{w}}{dt} = \boldsymbol{v} \tag{7}$$

$$\frac{d\boldsymbol{v}}{dt} = -\alpha\boldsymbol{v} - \eta\boldsymbol{\nabla}_w f(\boldsymbol{w}) = -\alpha\boldsymbol{v} - \eta\left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}\right). \tag{8}$$

これを差分化すると

$$w_{1,2}^{(n+1)} = w_{1,2}^{(n)} + v_{1,2}^{(n)} \tag{9}$$

$$v_{1,2}^{(n+1)} = (1-\alpha)\, v_{1,2}^{(n)} - \eta\frac{\partial f}{\partial w_{1,2}} \tag{10}$$

を得る。ここで $\Delta t = 1$ とした (安定に解ければなんでも良い)。

Momentum 法を SGD 法の上に実装してみよう。できたら次のことを確認/考察してみよう。

- SGD 法に比べて収束に要するステップ数はどう変化するか？
- SGD 法に比べて軌跡はどう変化するか？

3. Adaptive gradient(Ada) 法: 学習を進める際に 1 ステップでの「学習の大きさ」を決めるパラメーター η(サンプルプログラム中では `lr_w1`, `lr_w2`) はこれまで一定にしてきた。より効率的な学習を目指すならば、勾配が大きくて学習が不十分なうちは大きな η、学習が進んで勾配が小さくなってきたら小さな η を用いることが思いつくだろう。それを実装する方法として次のような進化する学習パラメーターが考えられる。

$$\eta \rightarrow \frac{\eta}{\sqrt{h/h_0}} \tag{11}$$

$$\frac{dh}{dt} = \left(\boldsymbol{\nabla}_w f(\boldsymbol{w})\right)^2 \tag{12}$$

$$h_0 = \left(\boldsymbol{\nabla}_w f(\boldsymbol{w})\right)^2 |_{t=0} \tag{13}$$

課題：これまでと同様に $\Delta t = 1$ として (12) 式を差分化し、SGD+Moment 法の上に実装せよ。実装できたら SGD+Moment 法や、SGD 法と収束性を比較せよ。SGD+Moment 法では学習パラメータ η を小さくとらないと停留点で大きな振動をしてしまうが、それに Ada 法を加えると、初期に η を大きくとっても良く収束することが確かめられるはずである。Ada 法と Moment 法とを組み合わせた方法は **Adam 法**と呼ばれて、機械学習の分野でもっともよく使用される学習方法になっている。