Neural Networks の初期化

Shigeki Karita

2018 年 8 月 20 日

目次

1 INTRODUCTION

NN の学習では誤差逆伝搬法に基づく勾配法を使います．単純な多層パーセプトロンを例にすると，勾配法では現在の重み W が損失

$$L = f(Wx + b) \tag{1}$$

を最小化する方向の勾配 $\partial L/\partial W$ を使って少しずつ更新します．ここで $W \in \mathbb{R}^{M\times N}$ はある層の重み行列，$x \in \mathbb{R}^N$ は前層からの出力ベクトル，$f : \mathbb{R}^M \to \mathbb{R}$ は W を含む層から出力を受けとり，後続の層を伝搬して損失を出力する関数とします．なお，多層パーセプトロンにおける層とは以上の W, b によるアフィン変換を基本的に指し，非線形関数などの変換を追加で適用する場合もある処理の単位としています (記号の説明が回りくどいですが，とくに変な仮定はないと思います)．

一般的に勾配法では重みの初期値は最終的な解に影響するので，初期化をどうすれば平均的に学習がうまく行くのか，または巨大な数の重みを学習できるかなど現在でも研究されています．よくある実装では一様分布や正規分布などに基づく乱数を用いて適当に決めます．しかし，一様分布の幅や，正規分布の分散の大きさは非常に重要な設定です．各層の重みで少しづつ出力が大きくなったり小さくなると，誤差逆伝搬における更新量としての勾配

$$\partial L/\partial W = x f'(Wx + b) \tag{2}$$

が前層の出力 x (の深さ乗) 倍されます．その結果，勾配が爆発したり微小になり更新が数値的に不安定になる問題があります．

2 メモ

- 現在の chainer や pytorch の初期化ヘルパーみたいなやつは筋が悪い気がします．Linear や Convolution 層は Grolot の方法で統一的に初期化して，ReLU や Dropout ごとに後処理としてスケーリングするべきと思いました (Gehring の論文も結局そうしている)
- 極論として，ものすごい複雑な関数を通す場合も，事前に標準正規分布から生成したサンプルで順伝搬・逆伝搬してみて，分散=1 になるようにスケールすればいいだけという気もします．batch normalization が効くのはこれを単にミニバッチ単位でやっていて，実際は事前にスケールをサンプリングで求めれば良い気もする?
- pytorch は chainer と違って正規分布ではなく，一様分布を使っているが，一様分布の分散≠ 一様分布の幅なので，このデフォルトの初期化は何を意図しているのかよくわからない (慣習?)
- uniform と normal は結局どちらが優れているのかわからない (入出力サイズにかかわらない単に小さい定数の分散を設定することが大い気もする)．この辺は自己回帰 CNN を扱う Gehring の論文でもよくわからない．
- RNN みたいなデータに応じて任意回数の伝搬が行われ，再帰を含んだ構造はどうすればいいのか, わからない

3 INITIALIZATION FOR NON-RECURRENT NEURAL NETWORKS

3.1 Initialization for Linear Layers

Lecun(彼は初期に畳み込みニューラルネットワークの誤差逆伝搬法を考案しました) が出した解決策としては，出力の平均を 0 付近かつ分散 (平均的な大きさ) が層を重ねても一定 (例えば 1) となるようにバイアス b を 0，重み w を平均 0，分散 $1/N$ (層の入力数の逆数) といった正規乱数で初期化するという方法でした (現在も chainer のデフォルトであり，pytorch も変種を使っています).

LeCun 98, Efficient Backprop `http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf`

例えば上記の初期化を行い，バイアスの期待値を $E[b] = 0$ ，前層の出力が理想的に $E[x] = 0, V[x] = 1$ となっているとし，もし次層の出力もずっと分散 1 なら，平均的には勾配が発散・消失する心配がなさそうです (補足：大抵の NN 学習では前処理として入力データ全体を，このように平均 0，分散 1 となるよう正規化しま

す). このように初期化した場合,

$$y_j = \sum_i w_{i,j} x_i + b \tag{3}$$

$$E[y_j] = \sum_i E[w_{i,j}]E[x_i] = 0 \tag{4}$$

$$\begin{aligned} V[y_j] &= \sum_i V[w_{i,j} x_i] + V[b] \\ &= \sum_i (V[w_{i,j}]V[x_i] + E[w_{i,j}]^2 V[x_i] + V[w_{i,j}]E[x_i]^2) \\ &= \sum_i V[x_i]/N = 1 \end{aligned} \tag{5}$$

という感じで，次層の出力も前層の出力と同じ平均 0，分散 1 になると期待できます．ちなみに多層パーセプトロンではなく，畳み込みニューラルネットワークも同様に各重み毎の入力数 (ただし入力チャネル x フィルタの大きさ) を使って各重みを初期化します．

3.2 Initialization for Nonlinear Functions

先の LeCun の方法は各層出力の分散について考えたものであったため，Grolot はさらに活性化関数や，各層の勾配の分散を考慮しました．通常，NN の各層は，そのまま重ねると $W(Vx + a) + b = WVx + Wa + b$ のように結局は線形写像しか関数近似できないので，層間に非線形関数を入れます $h(W(g(Vx + a) + b)$ ．そうなると当然，非線形関数も含めた各層の出力における分散は異なります．今更ですが，連続変数 x の分散を考えると, 定義は x とその平均との L2 距離の期待値 (確率変数の 2 次の中心化モーメントとも言います) なので

$$V[x] = \int (x - E[x])^2 \, p(x) dx \tag{6}$$

また $x \in \mathbb{R}^N$ 非線形変換した $h(x) \in \mathbb{R}^N$ については

$$V[x] = \int (h(x) - E[h(x)])^2 \, p(x) dx \tag{7}$$

となり， $h : \mathbb{R}^N \to \mathbb{R}^N$ がどういう関数かによって分散を予想することができます．

3.2.1 **TODO** ここの非線形関数の写像の分散について，Grolot の近似を厳密に議論する

ここで Grolot らは当時普及していた非線形関数である sigmoid や tanh といった x=0 で対称かつ傾き 1 の関数をアフィン変換とあわせた際，隣接する層の勾配の分散が等しくなるよう重み W 初期値の理想的な分散を次のように求めました

> Eq. (12), Glorot & Bengio, AISTATS 2010 - `http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf`

$$V[W] = \frac{2}{N + M} \tag{8}$$

ものすごくザックリ導出過程をまとめると，実は sigmoid や tanh のような $h'(0) = 1$ となる場合は近似的に，勾配 $\partial L / \partial W = x f'(Wx + b)$ の分散は $1/M$ となる場合に全ての層で一定の分散で伝搬できます．さらに Lecun の結果と連立させるには $2/(M + N)$ の分散となれば良いということです．

次に He らは 2013 年ごろから普及してきた非対称な非線形関数の ReLU 関数に対して，同様に分散を考えました (補足：経験的に sigmoid や tanh より ReLU の方が計算も収束も速かった).

> He et al., ICCV 2015 - https://www.cv-foundation.org/openaccess/content_iccv_2015/papers/He_Delving_Deep_into_ICCV_2015_paper.pdf

ものすごくザックリ説明すると ReLU 関数 $y = \max(0, Wx + b)$ に対して，今までの議論と同様に W と x が期待値 0 かつ対称に分布しているなら， $V[y] = E[\max(0, W^2x^2 + b^2)] = V[W]V[x]/2$ となるので， $V[y] = V[x]$ とするには $V[w_{i,j}] = 2/N$ というわけで，sigmoid や tanh の二倍の分散が必要になります．あと勾配に関する議論は Grolot と同じはずです.

ちなみに He の文献における Figure 3. は衝撃的です．初期化に用いる正規分布の分散を真面目に考えないと，とても深いネットワークでは全く学習できなくなることが示唆されています.

3.3 Initialization for Layers Suceeded by Non-Feedforward Operations

最近の NN には dropout，residual connection，attention，gating といった沢山の複雑な構造をもっており，Gehring らはそれらも考慮した初期化および非線形関数の正規化を考案しました．ただし大雑把に言うとマスキングや足し算，掛け算をやっているだけなので，先述のように各構造の分散を計算すると，きちんと学習できるようになったという報告です (本編は CNN を用いた Sequence-to-sequence の話で，初期化の話はすごく長い appendix に書いてあります).

> Gehring et al., ICML2017 - https://arxiv.org/abs/1705.03122

4 INITIALIZATION AND NORMALIZATION FOR RECURRENT NEURAL NETWORKS

たとえば ReLU や sigmoid, tanh の RNN (全ての変数の期待値は 0, とくに x_t の分散を 1 とする．独立でない変数同士 x_t, h_t, h_{t-1} の積の期待値は分解できないことに注意)

$$h_t = f(Wx_t + Uh_{t-1} + b) \tag{9}$$

$$E[h_t] = \alpha\,(E[W]E[x_t] + E[U]E[h_{t-1}]) = 0 \tag{10}$$

$$\begin{aligned}
V[h_t] &= E[h_t^2] = \alpha E[(Wx_t + Uh_{t-1})^2 + (Wx_t + Uh_{t-1})b + b^2] \\
&= \alpha E[(Wx_t + Uh_{t-1})^2] \\
&= \alpha E[W^2x_t^2 + Wx_tUh_{t-1} + U^2h_{t-1}^2] \\
&= \alpha\,(V[W]V[x_t] + E[W]E[U]E[x_th_{t-1}] + V[U]V[h_{t-1}]) \\
&= \alpha\,(V[W]V[x_t] + V[U]V[h_{t-1}]) \\
&= \alpha\,(V[W]V[x_t] + \alpha V[U]\,(V[W]V[x_{t-1}] + V[U]V[h_{t-2}])) \\
&= \sum_{k=1}^{t} (\alpha V[U])^{t-k} V[W]V[x_k]
\end{aligned} \tag{11}$$

よって $V[h_t]$ を全ての t で一定 (=1) とする一例としては

$$V[W] = 1/tN \tag{12}$$
$$V[U] = 1/\alpha M \tag{13}$$

ここで，スケール α は非線形関数 f が ReLU のとき 0.5，また sigmoid/tanh のときは 1.0 である．
考えられる実装としては以下の通り

1. W, U を LeCun Normal で初期化する．b は 0
2. 各時刻の入力を $1/t$ でスケールする $h_t = f(Wx_t/t + Vh_t + b)$'

補足：時刻が進むと Wx_t が消失するのでよくないかも... 別の一定にする方法を考える

- chrono init との比較・併用
  - `http://www.yann-ollivier.org/rech/publs/chrono.pdf`
  - `https://github.com/JosvanderWesthuizen/janet`
- IRNN との比較・併用
  - `https://arxiv.org/pdf/1504.00941.pdf`
- LSTM の場合の導出 (ConvS2S のゲート部分を見る)
  - Appendix A `https://arxiv.org/abs/1705.03122`

実験

- めっちゃ長い人工データ (Copy/Add tasks)
- めっちゃ深い RNN
- 勾配の発散・消失の図
- 音声認識/翻訳
- 言語モデル

Emacs 25.2.2 (Org mode 9.1.8)