# Deep Learning from Scratch 1 Note

generated by Gemini AI

2026年2月18日

**概要**

このノートは AI が私の手書きノートを読み取り、LaTeX のコードとして出力したものをほぼそのままコンパイルしたものです。そのため、間違っている箇所や読み取れていない箇所（抜けている箇所）があるかもしれません。その点については悪しからず。

## 1 Softmax-with-Loss レイヤを追う

Softmax 関数への入力を $\boldsymbol{X} : (N, m)$ と仮定する。（バッチサイズが $N$、各データに $m$ 個の値が格納）。

$$\boldsymbol{Y} = \mathrm{softmax}(\boldsymbol{X}) \implies \boldsymbol{Y} : (N, m)$$

教師データ $\boldsymbol{T}$ の形状は $(N, m)$ となる必要があるので、交差エントロピー誤差 $L$ は以下のようになる。

$$L = \mathrm{cross\_entropy\_error}(\boldsymbol{Y}, \boldsymbol{T})$$

### One-hot vector の $\boldsymbol{T}$

$$\boldsymbol{T} = \begin{pmatrix} t_{11} & \cdots & t_{1m} \\ \vdots & \ddots & \vdots \\ t_{N1} & \cdots & t_{Nm} \end{pmatrix}$$

$$(\boldsymbol{T})_{ij} = \begin{cases} 1 & (i\text{番目のデータの正解が}j-1\text{のとき}) \\ 0 & (\text{otherwise}) \end{cases}$$

### 実装上の注意：'cross_entropy_error' 関数の内部

> **Note**
>
> 疑問：なぜ index に変換するのか？これによって $\boldsymbol{t}$ が one-hot から正解ラベルのインデックスになるのはなぜか？
> $\Rightarrow$ そもそも、まず $\boldsymbol{t}$ と $\boldsymbol{y}$ の形状が等しいときは、$\boldsymbol{t}$ が one-hot である可能性がある（$\boldsymbol{y}$ が $(1, m)$ という形状のときは、$\boldsymbol{t}$ がどちらか分からないが）。
> 仮に $\boldsymbol{y} : (N, m)$ とする。そして、$\boldsymbol{t}$ が one-hot のときは、必ず形状が $(N, m)$ で $\boldsymbol{y}$ と同じになる。これによって `if t.size == y.size` というように判別できる。
> `t = t.argmax(axis=1)` とすると、$\boldsymbol{T}(=\boldsymbol{t})$ は $(N, m)$ から $(N, )$ という行ベクトルになる。そしてその各要素には、正解ラベルのインデックス（1 が格納されている要素番号）が入る。

$\boldsymbol{T}$ の形状が $\boldsymbol{Y}$ の形状と等しいとき、$\boldsymbol{T}$ は one-hot vector であるとみなして処理を行う。Return されるのは、

$$\frac{1}{N}\sum_k E_k \quad (\text{ただし } E_k = -\sum_k t_{nk}\log y_{nk})$$

つまり、スカラー（次元は 0）が返る。

### 逆伝播 (Backpropagation)

$$\mathrm{d}\boldsymbol{x} = (\text{self.y} - \text{self.t}) \,/\, \text{batch\_size}$$

> **Note**
>
> なぜバッチサイズで割る？今採用している交差エントロピー誤差は
>
> $$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{m} t_{ik}\log y_{ik}$$
>
> である。そうすると、前に求めた $\frac{\partial L}{\partial y_{ij}}$ 等を、この $L$ の場合で再度求めると、
>
> $$\frac{\partial L}{\partial a_{ij}} = \frac{1}{N}(y_{ij} - t_{ij})$$
>
> になるため。

# 2　学習に関するテクニック (§6)

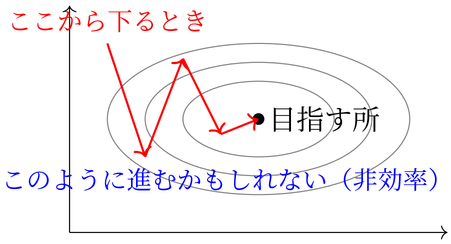
これまでは確率的勾配降下法 (SGD) を用いてきた。

$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{\partial L}{\partial \boldsymbol{W}}$$

However, there are some problems. Those are below.

### SGD のジグザグ問題

ここから下る時、必ずしも目指す所（最小値）へと一直線になるとは限らない。関数の形状が異方性（anisotropic）を持つ場合、勾配の方向が最小値を指さないことがある。




解決策として次の 3 つがある。

## 2.1　Momentum (モーメンタム)

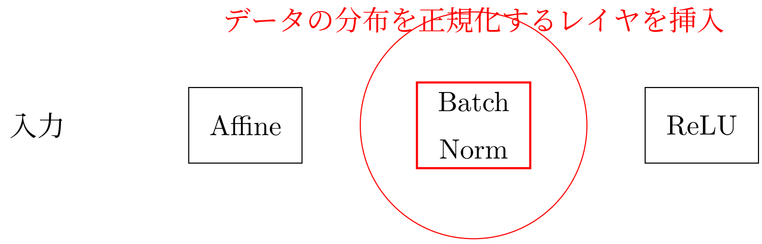
物理法則を借りる手法。速度 (velocity) $\boldsymbol{v}$ という変数を導入。⇒ 今まで動いてきた方向（勢い）を保ち（慣性)、新しい勾配の方向に力を加える。

**Note**

AI からの質問 (Memo) 活性化関数を ReLU から Sigmoid にすると、層が深くなっていくと逆伝播な勾配はどうなっていくか？→ Vanished (勾配消失)。パラパラと第 6 章を読んでいくのが良さそう。

# 3 Batch Normalization

Batch Norm レイヤは、$\hat{x}_i$ $(i = 1, \ldots, m)$ に対し、次のような変換をする。




## 順伝播 (Forward)

学習を行う際、ミニバッチを単位として、ミニバッチ毎に正規化を行う。i.e. データの分布について、平均が 0、分散が 1 になるようにする。

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{1}$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_B)^2 \tag{2}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \quad (\epsilon \text{は 0 除算防止の微小値}) \tag{3}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \tag{4}$$

$\gamma, \beta$ はパラメータ。学習により調整されていく（初期値：$\gamma = 1, \beta = 0$)。これだけだと、単なる標準化のみになってしまうため、固有のスケール（$\gamma$）とシフト（$\beta$）を与える。

## 逆伝播 (Backward)

微分の連鎖律を用いて導出する。

$$\frac{\partial L}{\partial \hat{x}_i} = \frac{\partial L}{\partial y_i} \cdot \frac{\partial y_i}{\partial \hat{x}_i} = \frac{\partial L}{\partial y_i} \cdot \gamma$$

ここからが複雑である。$x_1 \sim x_m$ のすべてが各 $\hat{x}_i$ に影響を与えているため、和をとる必要がある箇所に注意。

**Definition**

分散に関する微分

$$\frac{\partial\sigma_B^2}{\partial x_i} = \frac{2}{m}(x_i - \mu_B)$$

※ ノート内の取り消し線部修正：平均 $\mu_B$ も $x_i$ の関数であるため、厳密にはもう少し複雑だが、ここでは簡略化して記述されている可能性あり。

具体的な逆伝播のステップ（ノートの計算過程）：

$$\frac{\partial\hat{x}_i}{\partial\sigma_B^2} = (x_i - \mu_B)\cdot\left(-\frac{1}{2}(\sigma_B^2+\epsilon)^{-3/2}\right) \tag{5}$$

$$= -\frac{x_i - \mu_B}{2(\sigma_B^2+\epsilon)\sqrt{\sigma_B^2+\epsilon}} \tag{6}$$

$$\frac{\partial L}{\partial\sigma_B^2} = \sum_{i=1}^{m}\frac{\partial L}{\partial\hat{x}_i}\frac{\partial\hat{x}_i}{\partial\sigma_B^2} \tag{7}$$

$$= -\sum_{i=1}^{m}\left(\frac{\partial L}{\partial\hat{x}_i}\cdot\frac{x_i-\mu_B}{2(\sigma_B^2+\epsilon)^{1.5}}\right) \tag{8}$$

また、平均 $\mu_B$ に関する微分は、$\hat{x}_i$ からのルートと、$\sigma_B^2$ からのルートの 2 つがあることに注意。

$$\frac{\partial L}{\partial\mu_B} = \sum_{i=1}^{m}\frac{\partial L}{\partial\hat{x}_i}\frac{\partial\hat{x}_i}{\partial\mu_B} + \frac{\partial L}{\partial\sigma_B^2}\frac{\partial\sigma_B^2}{\partial\mu_B}$$

最終的に $\frac{\partial L}{\partial x_i}$ を求めるには、上記の各項を統合する。

**Note**

計算グラフによる理解計算グラフを書くことで、何が何にどのように影響を与えているかが見えてくる。順方向：分散 $\Rightarrow$ 集計。逆方向：分配（コピー）。

# 4　畳み込みニューラルネットワーク (CNN) (§7)

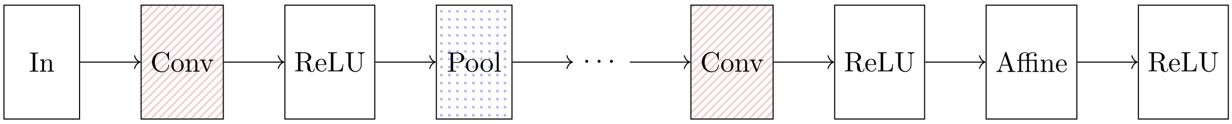
## 7.1 全体の構造

これまでに見たネットワーク（全結合層）：

$$\texttt{Affine} \to \texttt{ReLU} \to \texttt{Affine} \to \texttt{ReLU} \to \texttt{...}$$

**CNN (Convolutional Neural Network):** 新出レイヤとして「Convolution Layer (畳み込み層)」と「Pooling Layer (プーリング層)」が登場する。

どのようにレイヤを組み合わせるか？




出力に近い層では、これまでの `Affine` $\to$ `ReLU` の形に戻るのが一般的。