4 誤差逆伝播法

4.1 勾配計算の難しさ

確率的勾配降下法を実行するには、誤差関数$E\left(\mathrm{w}\right)$の勾配$\partial E\left(\mathrm{w}\right)/\partial \mathrm{w}$を計算する必要がある。

$$\mathrm{w}^{(t+1)} = \mathrm{w}^{(t)} - \epsilon \nabla E_i$$
$$\nabla E = \frac{\partial E\left(\mathrm{w}\right)}{\partial \mathrm{w}}$$

- 勾配のベクトルの各成分は、各層の$\partial E/\partial w_{ji}$と$\partial E/\partial b_j$

この誤差関数の微分計算は中間層、特に入力に近い深い層のパラメータほどその計算が面倒になる。

$$\mathbf{u}^{(l+1)} = \mathbf{W}^{(l+1)}\mathbf{z}^{(l)} + \mathbf{b}^{(l+1)}$$
$$\mathbf{z}^{(l+1)} = f\left(\mathbf{u}^{(l+1)}\right)$$

ex)1つのデータx_nに対する二乗誤差を第l層の重み$w_{ji}^{(l)}$で微分

$$E_n = \frac{1}{2}\left[f\left(x_n\right) - y_n\right]^2$$
$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \left[f\left(x_n\right) - y_n\right]\frac{\partial f\left(x_n;\mathrm{w}\right)}{\partial w_{ji}^{(l)}}$$

順伝播型ニューラルネットのドキュメントで活性化関数は以下のように入れ子状態になっていることを示した。

$$f\left(W^{(L)}z^{(L-1)}\left(x_n\right) + b^{(L)}\right)$$
$$= f\left(W^{(L)}f\left(W^{(L-1)}z^{(L-2)}\left(x_n\right) + b^{(L-1)}\right) + b^{(L)}\right)$$
$$= f\left(W^{(L)}f\left(W^{(L-1)}f\left(\cdots f\left(W^{(l)}z^{(l-1)}\left(x_n\right) + b^{(l)}\right)\cdots\right) + b^{(L-1)}\right) + b^{(L)}\right)$$

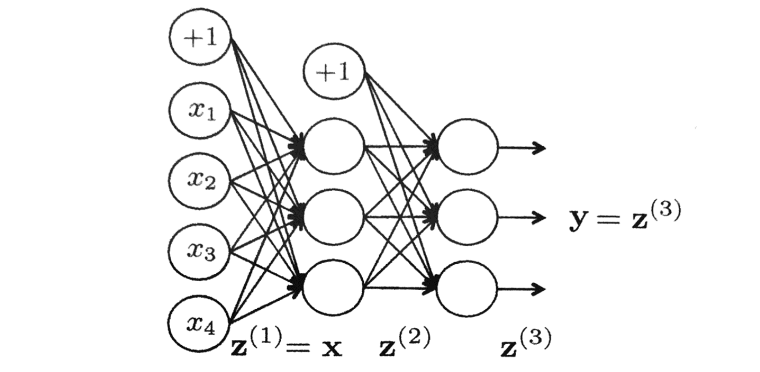
このため、微分の連鎖規則を何度も繰り返す。

→プログラミングが大変面倒

→計算量も大きくなる

この問題を解決する方法として、**誤差逆伝播法**(back propagation)がある。

※この問題は全学習データもしくはミニバッチに対する誤差関数の値に対しても言える。複数のデータに対する誤差関数の値は、各データに対する誤差関数の値の総和であるためである。

> 以降、表記を簡素化するために、下の図のように$+1$をいつも出力する特別な第0番ユニットを各層に導入し、バイアスb_jをそのユニットと各ユニットjとの結合の重み$w_{0j}^{l} = b_j^{(l)}$と考えることとする。
>
> 
>
>
> つまり、l層のユニットへの入力は、$l-1$層の第0ユニットの出力が常に$z_0^{(l-1)} = 1$となることで
>
> $$u_j^l = \sum_{i=1}^{M} w_{ji}^{(l)} z_i^{(l-1)} + b_j = \sum_{i=0}^{M} w_{ji}^{(l)} z_i^{(l-1)}$$
>
> - M：成分数
>
> と簡潔に書ける。

4.3 多層ネットワークでの微分計算

勾配降下法を実行するために各層の重みについて誤差関数を微分する必要がある。本節では、偏導関数を導出する。

- 一つの学習データnの特徴量：$\mathrm{x}_n = (x_{n1}, x_{n2}, \cdots, x_{ni}, \cdots, x_{nI})^T$
- 中間層の出力：$z_j^{(l)} = f\left(u_j^{(l)}\right) = f\left(\sum_i w_{ji}^{(l)} z_i^{(l-1)}\right)$

$$\begin{bmatrix} u_1^{(l)} \\ \vdots \\ u_j^{(l)} \\ \vdots \\ u_J^{(l)} \end{bmatrix} = \begin{bmatrix} w_{11}^{(l)} & \cdots & w_{1i}^{(l)} & \cdots & w_{1I}^{(l)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{j1}^{(l)} & \cdots & w_{ji}^{(l)} & \cdots & w_{jI}^{(l)} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ w_{J1}^{(l)} & \cdots & w_{Ji}^{(l)} & \cdots & w_{JI}^{(l)} \end{bmatrix} \begin{bmatrix} z_1^{(l-1)} \\ \vdots \\ z_i^{(l-1)} \\ \vdots \\ z_I^{(l-1)} \end{bmatrix} + \begin{bmatrix} b_1^{(l)} \\ \vdots \\ b_j^{(l)} \\ \vdots \\ b_J^{(l)} \end{bmatrix}$$

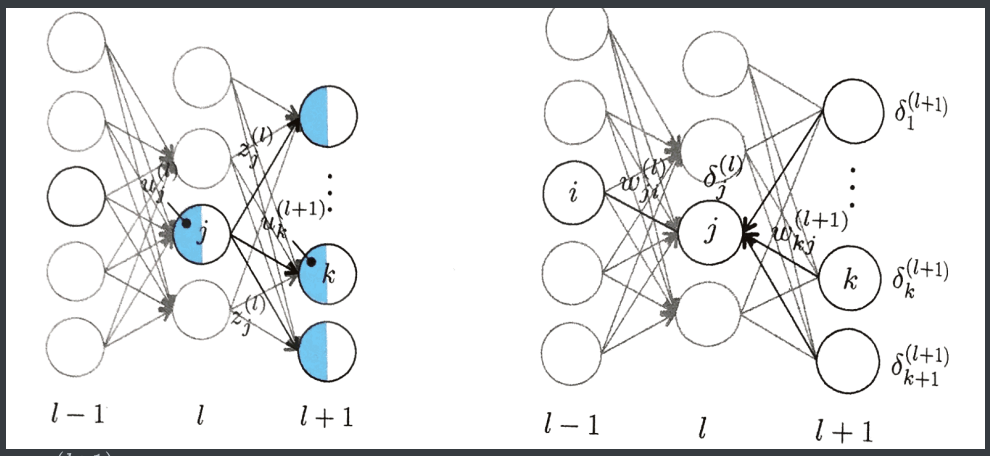
$$\begin{bmatrix} z_1^{(l)} \\ \vdots \\ z_j^{(l)} \\ \vdots \\ z_J^{(l)} \end{bmatrix} = \begin{bmatrix} f\left(u_1^{(l)}\right) \\ \vdots \\ f\left(u_j^{(l)}\right) \\ \vdots \\ f\left(u_J^{(l)}\right) \end{bmatrix}$$

第l層の重み$w_{ji}^{(l)}$について誤差関数を微分したとき簡潔に次式のように表せる。

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}}$$

上式について右辺各項について考える。まず第1項について考える。

第1項は、$u_j^{(l)}$の変動がE_nに与える影響を表しており、このユニットjからの出力$z_j^{(l)}$を通じ、第$l+1$層の各ユニットkの総入力$u_k^{(l+1)}$を変化させることによってのみ生じる。



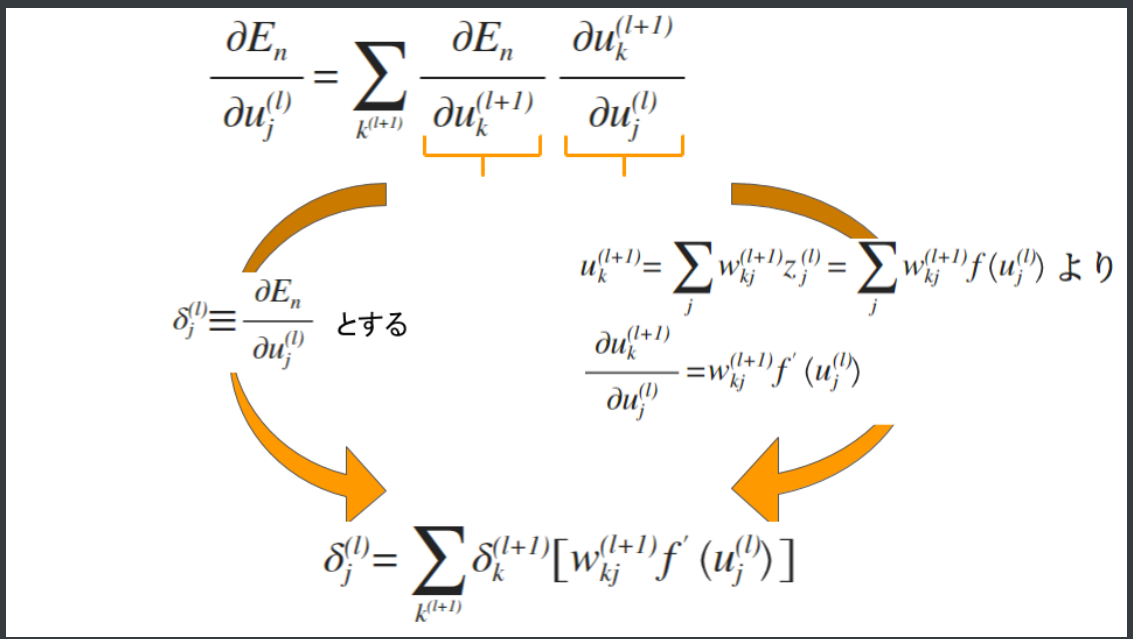
したがって、各$u_k^{(l+1)}$を経由した微分連鎖により、

$$\frac{\partial E_n}{\partial u_j^{(l)}} = \sum_{k^{(l+1)}} \frac{\partial E_n}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}}$$

となる。

- $k^{(l+1)}$：第$l+1$層のユニット数

そして、上式は以下のように



$$\begin{aligned}
\delta_j^{(L-1)} &= \textstyle\sum_{k^{(l+1)}} \delta_k^{(L)} \left[w_{kj}^{(L)} f' \left(u_j^{(L-1)} \right) \right] \\
&\vdots \\
\delta_j^{(l)} &= \textstyle\sum_{k^{(l+1)}} \delta_k^{(l+1)} \left[w_{kj}^{(l+1)} f' \left(u_j^{(l)} \right) \right]
\end{aligned}$$

と表せる。この式を利用して$\delta_k^{(l+2)}, \delta_k^{(l+3)}, \cdots$と出力層$L$まで再帰的に計算することで、第$l$層のデルタが計算できる。

第2項$\partial u_j^{(l)} / \partial w_{ji}^{(l)}$は、$u_j^{(l)} = \sum_i w_{ji}^{(l)} z_i^{(l-1)}$の関係から簡単に

$$\frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} = z_i^{(l-1)}$$

と計算できる。したがって、第l層の重み$w_{ji}^{(l)}$について誤差関数を微分した式は以下になる。

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)}$$

デルタ$\delta_j^{(l)}$の計算は上で述べたように、出力層から入力層へ向かって繰り返し、$\delta_j^{(l)} = \sum_{k^{(l+1)}} \delta_k^{(l+1)} \left[w_{kj}^{(l+1)} f' \left(u_j^{(l)} \right) \right]$を適用することで求められる。逆伝播の最初の値は出力層での$\delta_j^{(L)}$が与えられているが、これは陽に

$$\delta_j^{(L)} = \frac{\partial E_n}{\partial u_j^{(L)}}$$

と計算できる。ただし、具体的な計算は選択した出力層の活性化関数と誤差関数によって変わる。

上記の勾配式は一つの学習データnから求まったものであり、具体的なアルゴリズムは以下の図で示してある。

入力： 訓練サンプル $\mathbf{x}_n$ および目標出力 $\mathbf{d}_n$ のペア 1 つ．
出力： 誤差関数 $E_n(\mathbf{w})$ の各層 l のパラメータについての微分 $\partial E_n / \partial w_{ji}^{(l)}$ $(l = 2, \ldots, L)$．

1. $\mathbf{z}^{(1)} = \mathbf{x}_n$ とし，各層 l $(= 2, \ldots, L)$ のユニット入出力 $\mathbf{u}^{(l)}$ および $\mathbf{z}^{(l)}$ を順に計算する（**順伝播**）．

2. 出力層でのデルタ $\delta_j^{(L)}$ を求める（通常は $\delta_j^{(L)} = z_j - d_j$ となる）．

3. 中間層 $l(= L-1, L-2, \ldots, 2)$ での $\delta_j^{(l)}$ を，この順に式 (4.12) に従って計算する（**逆伝播**）．

4. 各層 $l(= 2, \ldots, L)$ のパラメータ $w_{ji}^{(l)}$ に関する微分を式 (4.13) に従って計算する．

そして、ミニバッチなどの複数の学習データに対する誤差の総和 $E = \sum_n E_n$ の勾配もこの手順を学習データ $(\mathrm{x}_n, \mathrm{y}_n)$ ごとに並行に繰り返し、得られる勾配の和を

$$\frac{\partial E}{\partial w_{ji}^{(l)}} = \sum_n \frac{\partial E_n}{\partial w_{ji}^{(l)}}$$

とすることで求められる。

4.4 勾配降下法の完全アルゴリズム

4.4.1 出力層でのデルタ

> 逆伝播計算の起点は出力層でのデルタ $\delta_j^{(L)} = \partial E_n / \partial u_j^{(L)}$ である。その計算は使用する誤差関数および出力層の活性化関数に依存するが、代表的な場合を以下に示す。

回帰

$$E_n = \frac{1}{2} \sum_j \left[y_j - d_j \right]^2$$

出力層の活性化関数：恒等写像 $y_j = z_j^{(L)} = u_j^{(L)}$

$$\delta_j^{(L)} \equiv \frac{\partial E_n}{\partial u_j^{(L)}} = y_j - d_j$$

二値分類

$$E_n = -\left[d \log y + (1-d) \log \left(1 - y \right) \right]$$

出力層の活性化関数：ロジスティック関数 $y = 1 / \left[1 + \exp \left(-u \right) \right]$

$$\delta_j^{(L)} \equiv \frac{\partial E_n}{\partial u_j^{(L)}} = -\frac{d}{y} \frac{dy}{du_j^{(L)}} - \frac{1-d}{1-y} \left(-\frac{dy}{du_j^{(L)}} \right)$$

$dy / du_j^{(L)}$ は

$$
\begin{aligned}
\frac{dy}{du_j^{(L)}} &= \frac{\exp(-u)}{[1+\exp(-u)]^2} \\
&= \frac{1}{1+\exp(-u)} \cdot \frac{\exp(-u)}{1+\exp(-u)} \\
&= y \cdot (1-y)
\end{aligned}
$$

なので、

$$
\begin{aligned}
\frac{\partial E_n}{\partial u_j^{(L)}} &= -\frac{d}{y} y(1-y) + \frac{1-d}{1-y} y(1-y) \\
&= -d(1-y) + (1-d)y \\
&= y - d
\end{aligned}
$$

となる。

多クラス分類

出力層の活性化関数：ソフトマックス関数

$$
E_n = -\sum_k d_k \log y_k = -\sum_k d_k \log \left(\frac{\exp\left(u_k^{(L)}\right)}{\sum_i \exp\left(u_i^{(L)}\right)} \right)
$$

$$
\delta_j^{(L)} \equiv \frac{\partial E_n}{\partial u_j^{(L)}} = -\sum_k d_k \frac{1}{y_k} \frac{\partial y_k}{\partial u_j^{(L)}}
$$

$k = j$のときの$\partial y_k / \partial u_j^{(L)}$

$$
\begin{aligned}
\frac{\partial y_k}{\partial u_j^{(L)}} &= \frac{\exp\left(u_j^{(L)}\right) \sum_i \exp\left(u_i^{(L)}\right) - \exp\left(u_j^{(L)}\right) \exp\left(u_j^{(L)}\right)}{\left[\sum_i \exp\left(u_i^{(L)}\right)\right]^2} \\
&= y_j - y_j^2
\end{aligned}
$$

$k \neq j$のときの$\partial y_k / \partial u_j^{(L)}$

$$
\begin{aligned}
\frac{\partial y_k}{\partial u_j^{(L)}} &= \frac{-exp\left(u_k^{(L)}\right) exp\left(u_j^{(L)}\right)}{\left[\sum_i exp\left(u_i^{(L)}\right)\right]^2} \\
&= -y_k y_j
\end{aligned}
$$

上式より

$$
\begin{aligned}
\frac{\partial E_n}{\partial u_j^{(L)}} &= -d_j \frac{1}{y_j}\frac{\partial y_j}{\partial u_j^{(L)}} - \sum_{k\neq j} d_k \frac{1}{y_k}\frac{\partial y_k}{\partial u_j^{(L)}} \\
&= -d_j \frac{1}{y_j}\frac{\partial y_j}{\partial u_j^{(L)}} - \sum_{k\neq j} d_k \frac{1}{y_k}\frac{\partial y_k}{\partial u_j^{(L)}} \\
&= -d_j \frac{1}{y_j}\left(y_j - y_j^2\right) - \sum_{k\neq j} d_k \frac{1}{y_k}\left(-y_k y_j\right) \\
&= y_j - d_j
\end{aligned}
$$

となる。ただし、$\sum_k d_k = 1$となる。

> 上記のように回帰、二値分類およびクラス分類のいずれかにおいても、出力層Lのユニットjのデルタ$\delta_j^{(L)}$は、ネットワークの出力y_jとその目標出力d_jの差になる。

4.4.2 順伝播と逆伝播の行列計算

> 順伝播、逆伝搬およびパラメータ更新の各計算は、各層ごとに行列計算として表記できます。そこで以下では、ミニバッチを使った確率的勾配降下法の全計算を行列を使って表記します。

- 重みw_{ji}を(j,i)成分に持つ$(J\times I)$の行列：W
- $(J\times 1)$のバイアスベクトル：b
- ミニバッチ：$D_t \equiv \mathrm{X} = [\mathrm{x}_1, \cdots, \mathrm{x}_N]$
- 特徴ベクトル：$\mathrm{x}_n = (x_1, \cdots, x_p)^T$
- データx_nを入力したときのl層の各ユニットについて、その総入力を成分に並べたベクトル：$\mathrm{u}_n^{(l)}$
- ベクトル$\mathrm{u}_n^{(l)}$に活性化関数を作用させた各ユニットの出力を成分に持つベクトル：$\mathrm{z}_n^{(l)}$

$$
\begin{aligned}
\mathrm{U}^{(l)} &= \left[\mathrm{u}_1^{(l)} \cdots \mathrm{u}_N^{(l)}\right] \\
\mathrm{Z}^{(l)} &= \left[\mathrm{z}_1^{(l)} \cdots \mathrm{z}_N^{(l)}\right] \\
\mathrm{Z}^{(1)} &\equiv \mathrm{X}
\end{aligned}
$$

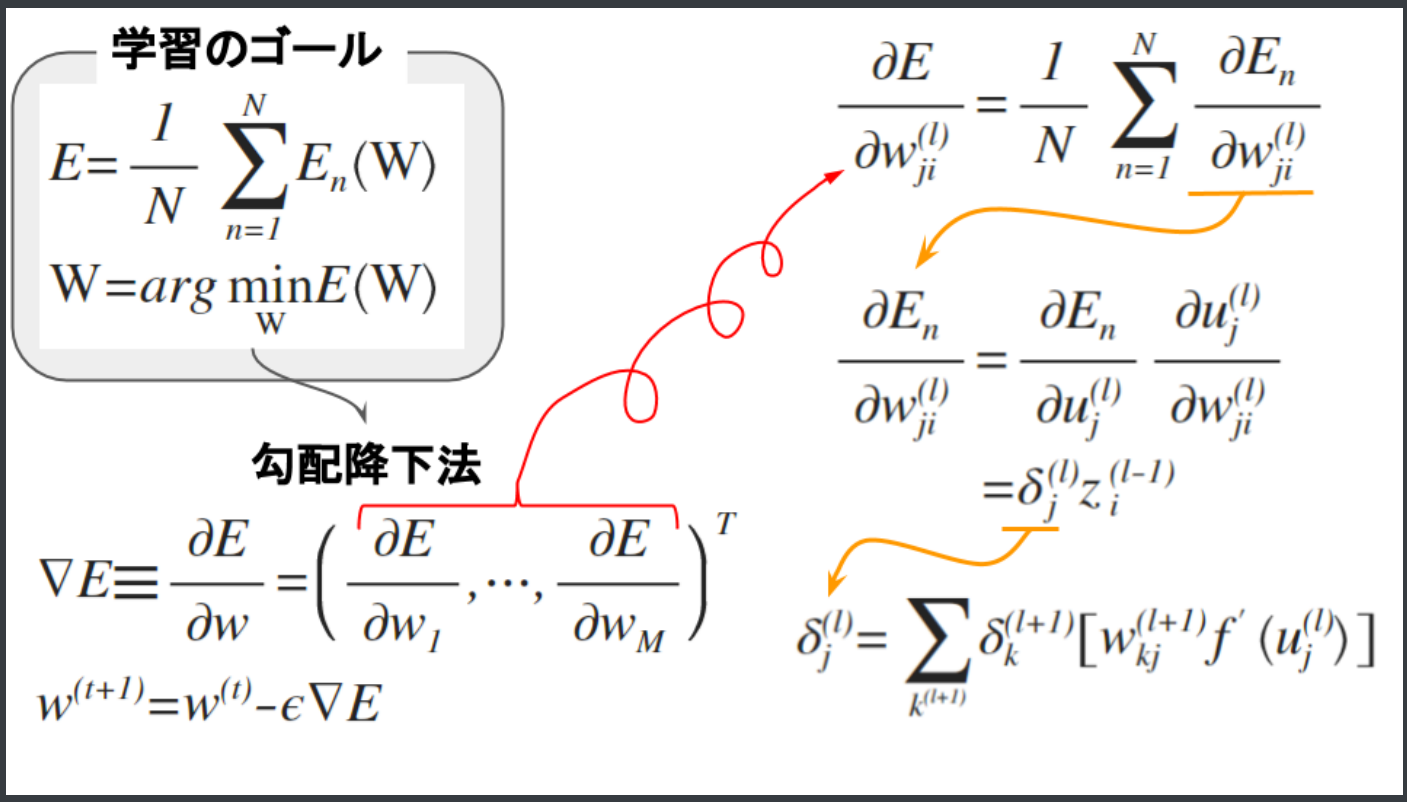
上記の表記を用いて、$l = 2, \ldots, L$について次の2つの式の計算を行える。

$$\underset{(J\times N)}{U^{(l)}}=\underset{(J\times I)}{W^{(l)}}\underset{(I\times N)}{Z^{(l-1)}}+\underset{(J\times 1)}{b^{(l)}}\underset{(1\times N)}{1_N^T}$$

$$\underset{(J\times N)}{Z^{(l)}}=f^{(l)}(\underset{(J\times N)}{U^{(l)}})$$

- 1_N：1をN個並べたベクトル
- $f^{(l)}(\cdot)$：行列の各成分に並行に活性化関数を適用し、値を同サイズの行列で返すもの

上記の出力を行うために、選んだ誤差関数に対して最小値を与える重みを探索する方法として、勾配降下法を用いる。各層$l=2,\ldots,L$の重みの更新アルゴリズムを行列表記で示す。以下の行列表記は、実際にプログラムを実装する際に有用である。



ミニバッチD_tごとに各層$l=2,\ldots,L$について並行に行列計算する。

勾配

$$\underset{(J\times I)}{\partial \mathrm{W}^{(l)}} = \frac{1}{N} \underset{(J\times N)}{\Delta^{(l)}} \underset{(N\times I)}{Z^{(l-1)T}}$$

$$\underset{(J\times 1)}{\partial \mathrm{b}^{(l)}} = \frac{1}{N} \underset{(J\times N)}{\Delta^{(l)}} \underset{(N\times 1)}{1_N}$$

更新式

$$\mathrm{W}^{(l)(t+1)} \leftarrow \mathrm{W}^{(l)(t)} - \epsilon \partial \mathrm{W}^{(l)}$$

$$\mathrm{b}^{(l)(t+1)} \leftarrow \mathrm{b}^{(l)(t)} - \epsilon \partial \mathrm{b}^{(l)}$$

勾配

$$\begin{bmatrix} \frac{\partial E}{\partial w_{11}^{(l)}} & \cdots & \frac{\partial E}{\partial w_{1i}^{(l)}} & \cdots & \frac{\partial E}{\partial w_{1I}^{(l)}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{j1}^{(l)}} & \cdots & \frac{\partial E}{\partial w_{ji}^{(l)}} & \cdots & \frac{\partial E}{\partial w_{jI}^{(l)}} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{J1}^{(l)}} & \cdots & \frac{\partial E}{\partial w_{Ji}^{(l)}} & \cdots & \frac{\partial E}{\partial w_{JI}^{(l)}} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} \delta_1^{(l)}{}_1 & \cdots & \delta_1^{(l)}{}_n & \cdots & \delta_1^{(l)}{}_N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \delta_j^{(l)}{}_1 & \cdots & \delta_j^{(l)}{}_n & \cdots & \delta_j^{(l)}{}_N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \delta_J^{(l)}{}_1 & \cdots & \delta_J^{(l)}{}_n & \cdots & \delta_J^{(l)}{}_N \end{bmatrix} \begin{bmatrix} z_1^{(l-1)}{}_1 & \cdots & z_i^{(l-1)}{}_1 & \cdots & z_I^{(l-1)}{}_1 \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_1^{(l-1)}{}_n & \cdots & z_i^{(l-1)}{}_n & \cdots & z_I^{(l-1)}{}_n \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ z_1^{(l-1)}{}_N & \cdots & z_i^{(l-1)}{}_N & \cdots & z_I^{(l-1)}{}_N \end{bmatrix}$$

$$\begin{bmatrix} \partial E/\partial b_1^{(l)} \\ \vdots \\ \partial E/\partial b_j^{(l)} \\ \vdots \\ \partial E/\partial b_J^{(l)} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} \delta_1^{(l)}{}_1 & \cdots & \delta_1^{(l)}{}_n & \cdots & \delta_1^{(l)}{}_N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \delta_j^{(l)}{}_1 & \cdots & \delta_j^{(l)}{}_n & \cdots & \delta_j^{(l)}{}_N \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \delta_J^{(l)}{}_1 & \cdots & \delta_J^{(l)}{}_n & \cdots & \delta_J^{(l)}{}_N \end{bmatrix} \begin{bmatrix} 1_1 \\ \vdots \\ 1_n \\ \vdots \\ 1_N \end{bmatrix}$$

デルタ$\delta_j^{(l)} \equiv \frac{\partial E_n}{\partial u_j^{(l)}}$の行列計算

$$\Delta^{(l)} = f^{(l)\prime}\left(\mathrm{U}^{(l)}\right) \odot \left(\mathrm{W}^{(l+1)T} \Delta^{(l+1)}\right)$$

- $\Delta^{(l)}$: $(J \times N)$の行列
- $\mathrm{W}^{(l)}$: $(J \times K)$の行列
- $\mathrm{W}^{(l+1)}$: $(K \times N)$の行列
- J : 第l層のユニット数
- K : 第$l+1$層のユニット数

- N : ミニバッチの学習データ数
- $f^{(l)\prime}\left(\mathrm{U}^{(l)}\right)$: Uの各成分にl層の活性化関数の微分$f'(\cdot)$を並行に適用して得られる同サイズの行列

4.4.5 勾配の差分近似計算

> もしネットワークの構造や活性化関数を自分で実装した場合、誤差関数の勾配が正しく計算されているかを検証する必要がある。そのために、勾配を数値微分で計算し、両者を比較することが有効である。

誤差関数Eの勾配の**差分近似**(difference approximation)は

$$\frac{\partial E}{\partial w_{ji}^{(l)}} = \frac{E\left(\ldots, w_{ji}^{(l)} + \epsilon, \ldots\right) - E\left(\ldots, w_{ji}^{(l)}, \ldots\right)}{\epsilon}$$

のように計算される。$\epsilon \to 0$の極限をとると、上の式は偏微分$\partial E\left(\mathrm{w}\right)/\partial w_{ji}^{(l)}$の定義に一致する。

ϵの値は、近似の精度がよくなるように十分に小さな値を選ぶ必要がある。値としては以下のように選ぶのが一般的である。

$$\epsilon \equiv \sqrt{\epsilon_c}\,|w_{ji}|$$

ただし、右辺を評価した結果$\epsilon = 0$になってしまう場合には、$\epsilon \equiv \sqrt{\epsilon_c}$と設定し直します。

4.5 勾配消失問題

本節では、誤差逆伝播法から深いニューラルネットの学習がなぜ難しいのかを示す。

誤差関数を重み$w_{ji}^{(l)}$で微分すると以下の式になる。

$$
\begin{aligned}
\frac{\partial E}{\partial w_{ji}^{(l)}} &= \frac{1}{N}\sum_{n=1}^{N}\frac{\partial E_n}{\partial w_{ji}^{(l)}} \\
&= \frac{1}{N}\sum_{n=1}^{N}\frac{\partial E_n}{\partial u_j^{(l)}}\frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} \\
&= \frac{1}{N}\sum_{n=1}^{N}\left\{\sum_{k^{(l+1)}}\delta_k^{(l+1)}\left[w_{kj}^{(l+1)}f'\left(u_j^{(l)}\right)\right]\right\}z_i^{(l-1)} \\
&= \frac{1}{N}\sum_{n=1}^{N}\left\{\sum_{k^{(l+1)}}\sum_{k^{(l+1)}}\left\{\cdots\left\{\sum_{k^{(L)}}\delta_k^{(L)}\left[w_{kj}^{(L)}f'\left(u_j^{(L-1)}\right)\right]\right\}\cdots\left[w_{kj}^{(l+2)}f'\left(u_j^{(l+1)}\right)\right]\right\}\left[w_{kj}^{(l+1)}f'\left(u_j^{(l)}\right)\right]\right\}z_i^{(l-1)}
\end{aligned}
$$

このとき、活性化関数が恒等関数であると仮定すると簡潔に以下の式になる。

$$
\begin{aligned}
\frac{\partial E}{\partial w_{ji}^{(l)}} &= \frac{1}{N}\sum_{n=1}^{N}\frac{\partial E_n}{\partial w_{ji}^{(l)}} \\
&= \frac{1}{N}\sum_{n=1}^{N}\left\{\sum_{k^{(l+1)}}\sum_{k^{(l+1)}}\left\{\cdots\left\{\sum_{k^{(L)}}\delta_k^{(L)}w_{kj}^{(L)}\right\}\cdots w_{kj}^{(l+2)}\right\}w_{kj}^{(l+1)}\right\}z_i^{(l-1)}
\end{aligned}
$$

重みは学習中にどのような値をとるかわからない。そのため、以下のような問題が生じる。

- パラメータが比較的大きく$w_{kj}^{(L)} = w_{kj}^{(L-1)} = \cdots = w_{kj}^{(l)} = 100$の場合は、上記の微分の値は非常に大きな値になり、損失に対して非常に感度が高い状態となる
  - →この状態を**勾配爆発**(exploding gradients)と呼ぶ。
  - オーバーフローが起こりやすいなど計算機での計算が不安定な状態になる。
- パラメータが比較的小さく$w_{kj}^{(L)} = w_{kj}^{(L-1)} = \cdots = w_{kj}^{(l)} = 0.01$の場合は、上記の微分の値は非常に小さい値になり、損失に対して非常に感度が低い状態となる
  - →この状態を**勾配消失**(vanishing gradients)と呼ぶ。
  - 損失の値が非常に小さい値としてしか伝わらないため先の勾配法を使って更新する場合には$w_{ji}^{(l)}$のパラメータがなかなか更新されない、つまり学習が進まない問題が発生する

このように中間層をもっと増やすことでより困難が増える。深層学習が広く使われるようになった背景には、これらの課題を克服するようなネットワーク構造が発明されたことも一因で、それらの手法の有効性は誤差逆伝播の仕組みで説明できる。