Neural Network の基礎

本谷 秀堅

名古屋工業大学

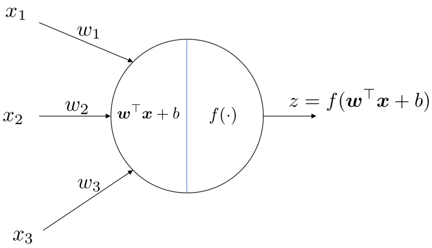
ニューラルネットワークによる関数の表現と学習

ニューラルネットワークの構成単位

ユニットの入出力

- 入力：$\boldsymbol{x} = (x_1, x_2, \ldots,)^\top$
- 出力：$y = f(u)$
  - まず重みとの内積＋定数を計算：
    $u = w_1 x_1 + w_2 x_2 + \cdots + b = \boldsymbol{w}^\top \boldsymbol{x} + b$
  - 次に活性化関数：$f(u) = f(\boldsymbol{w}^\top \boldsymbol{x} + b)$

ニューラルネットワークの構成単位



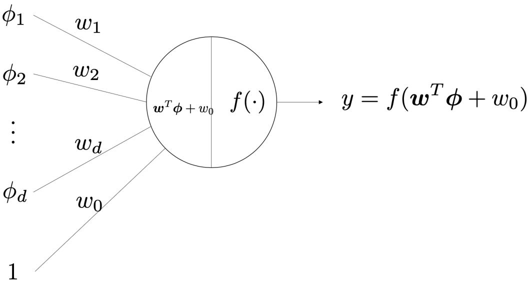
ユニットの入出力

- 入力：$\boldsymbol{x} = (x_1, x_2, \ldots,)^\top$
- 出力：$y = f(u)$
  - まず重みとの内積＋定数を計算：
    $u = w_1 x_1 + w_2 x_2 + \cdots + b = \boldsymbol{w}^\top \boldsymbol{x} + b$
  - 次に活性化関数：$f(u) = f(\boldsymbol{w}^\top \boldsymbol{x} + b)$

これまでとの比較

$\boldsymbol{\phi}(\boldsymbol{x}) = [\phi_1, \phi_2, \ldots, \phi_d]^\top$
活性化関数, $f(\cdot)$

$$y = f(\boldsymbol{w}^\top \boldsymbol{\phi} + w_0)$$



これまでの復習

2 クラス識別（ロジスティック回帰）

入力, $\boldsymbol{x}$, と出力, $z \in \mathbb{R}$ の関係：

$$z(x) = p(y = 1|\boldsymbol{x}) = \sigma(a) = \frac{1}{1 + \exp(-(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + w_0))}$$

学習データ $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)|i = 1, 2, \ldots, N, \;\; y_i \in \{0, 1\}\}$ が与えられている
最尤法による係数 $\boldsymbol{w}$ の求める:

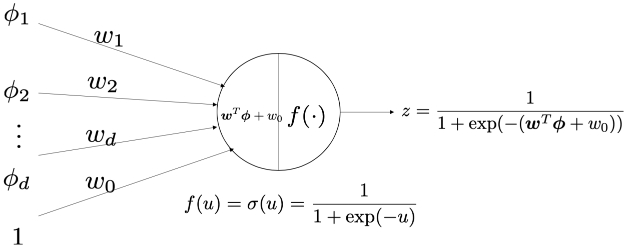
$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left[-\sum_{i=1}^{N} y_i \log p(y_i|\boldsymbol{x}_i) \right]$$

$$= \arg\min_{\boldsymbol{w}} \left[-\sum_{i=1}^{N} \{ y_i \log z(\boldsymbol{x}) + (1 - y_i) \log(1 - z(\boldsymbol{x})) \} \right]$$

ニューラルネットワークによる２クラス識

入力, $\boldsymbol{x}$, と出力, $z \in (0,1)$ の関係

$$z(x) = p(y = 1|\boldsymbol{x}) = \sigma(a) = \frac{1}{1 + \exp(-(\boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{x}) + w_0))}$$



係数, $\boldsymbol{w}$, 決定のためのコスト関数

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left[-\sum_{i=1}^{N} \{ y_i \log z(\boldsymbol{x}) + (1 - y_i) \log(1 - z(\boldsymbol{x})) \} \right]$$

これまでの復習

K クラス識別

入力, $\boldsymbol{x}$, と出力, $z_k \in (0,1)$ の関係 $(k = 1, 2, \ldots, K)$：

$$z_k(x) = p(y_k = 1|\boldsymbol{x}) = \mathrm{SM}_k(a) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)},$$

$$a_k = \boldsymbol{w}_k^\top \boldsymbol{\phi} + w_{k0}$$

学習データ $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)|i = 1, 2, \ldots, N, \ \ y_{ik} \in \{0, 1\}, \sum_k y_{ik} = 1\}$
最尤法による係数 $\boldsymbol{w}$ の求める:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left[-\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log\{p(y_{ik} = 1|\boldsymbol{x})\} \right]$$

$$= \arg\min_{\boldsymbol{w}} \left[-\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \frac{\exp(a_{ik})}{\sum_{j=1}^{K} \exp(a_{ij})} \right]$$

ニューラルネットワークによる多クラス識別

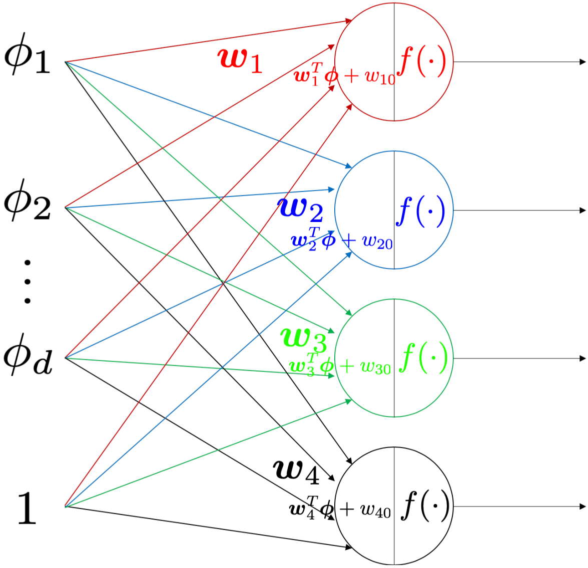
入力, $\boldsymbol{x}$, と出力, $z_k \in (0,1)$ の関係 $(k = 1, 2, \dots, K)$：

$$z_k(x) = p(y_k = 1|\boldsymbol{x}) = \mathrm{SM}_k(a) = \frac{\exp(a_k)}{\sum_{j=1}^{K} \exp(a_j)}, \quad \text{ただし } a_k = \boldsymbol{w}_k^\top \boldsymbol{\phi} + w_{k0}$$

最尤法による係数 $\boldsymbol{w}$ の求める:

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left[-\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \frac{\exp(a_{ik})}{\sum_{j=1}^{K} \exp(a_{ij})} \right]$$

ニューラルネットワークによる多クラス識別



$$z_1 = \frac{\exp(a_1)}{\sum_{j=1}^{K} \exp(a_j)}$$

$$z_2 = \frac{\exp(a_2)}{\sum_{j=1}^{K} \exp(a_j)}$$

$$z_3 = \frac{\exp(a_3)}{\sum_{j=1}^{K} \exp(a_j)}$$

$$z_4 = \frac{\exp(a_4)}{\sum_{j=1}^{K} \exp(a_j)}$$

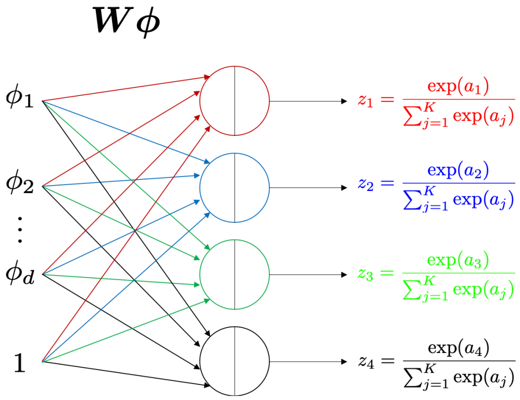
誤差の測り方（復習）

基本的に「負の対数尤度」で測る

問題	出力層の活性化関数	典型的な誤差関数
回帰	恒等写像	二乗誤差
2 クラス識別	ロジスティック関数	交差エントロピー
多クラス識別	ソフトマックス関数	交差エントロピー

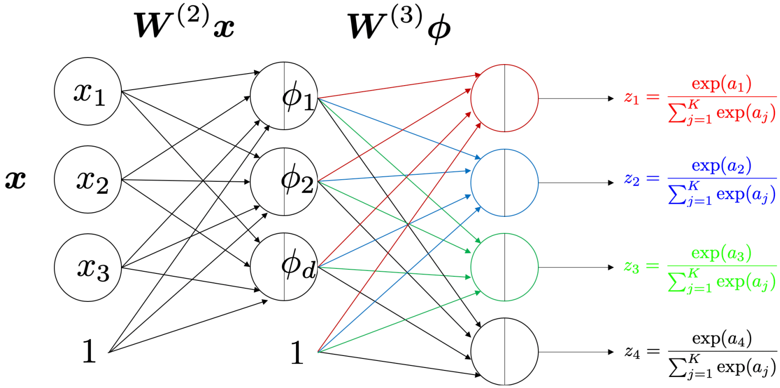
特徴 ϕ の作り方

入力 $\boldsymbol{x}$ を変換するために層を増やす



特徴 ϕ の作り方

入力 $\boldsymbol{x}$ を変換するために層を増やす



特徴 ϕ の作り方

層を増やした後の関数：

$$\boldsymbol{z} = \mathrm{SM}(\boldsymbol{W}^{(3)}\boldsymbol{\phi}(\boldsymbol{x}))$$

クラス k に属する事後確率

$$z_k = p(y_k = 1|\boldsymbol{x}) = \mathrm{SM}_k\left(\boldsymbol{w}_k^{(3)\top}\boldsymbol{\phi}(\boldsymbol{x})\right) = \mathrm{SM}_k\left(\sum_i w_{ki}^{(3)}\phi_i(\boldsymbol{x}) + w_{k0}\right)$$

$$= \mathrm{SM}_k\left(\sum_i w_{ki}^{(3)}\left(\sum_j f(\boldsymbol{w}_{ij}^{(2)\top}\boldsymbol{x} + w_{i0})\right) + w_{k0}\right)$$

特徴 ϕ の作り方

ニューラルネットワークの長所

最尤法による係数 $\boldsymbol{w}$ の推定により適切な特徴（写像）$\boldsymbol{\phi}$ を自動で獲得する

例：K クラス識別

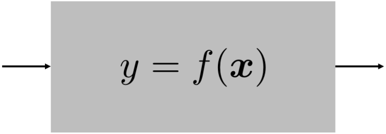
$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}} \left[- \sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log \frac{\exp(a_{ik})}{\sum_{j=1}^{K} \exp(a_{ij})} \right], \quad (a_{ik} = \boldsymbol{w}_k^\top \boldsymbol{\phi}(\boldsymbol{x}_i) + w_{k0})$$

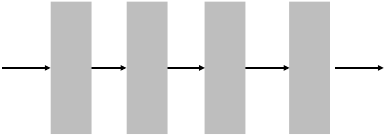
特徴 $\phi(\boldsymbol{x}_i)$ は、前項に沿って書くと、次式のとおり

$$\boldsymbol{\phi}(\boldsymbol{x}_i) = f(\boldsymbol{W}^{(2)} \boldsymbol{x}_i) = \begin{bmatrix} \phi_1(\boldsymbol{x}_i) \\ \phi_2(\boldsymbol{x}_i) \\ \phi_3(\boldsymbol{x}_i) \end{bmatrix} = \begin{bmatrix} f(\boldsymbol{w}_1^{(2)\top} \boldsymbol{x}_i + w_{10}) \\ f(\boldsymbol{w}_2^{(2)\top} \boldsymbol{x}_i + w_{20}) \\ f(\boldsymbol{w}_3^{(2)\top} \boldsymbol{x}_i + w_{30}) \end{bmatrix}$$

目的は適切な識別関数の実現

複数の素朴な関数の荷重和と合成で複雑な関数を表現



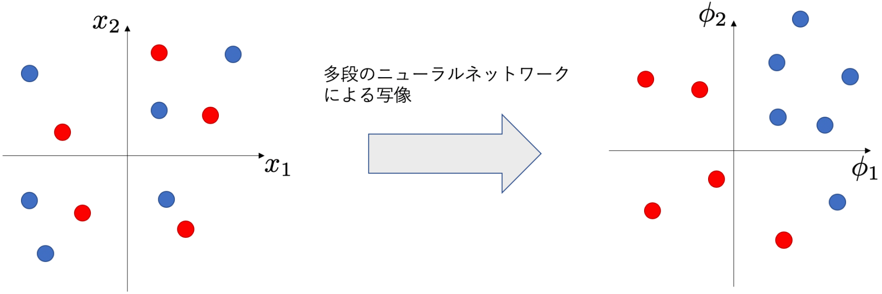


$$y = f(\boldsymbol{x})$$

$$y = f(\boldsymbol{x}) = f_L \circ f_{L-1} \circ \ldots f_1 \circ \boldsymbol{x}$$

複雑な関数による適切な写像

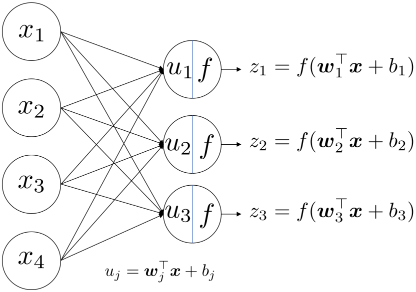
パターン認識であれば線形分離性を高める特徴を得たい



参考書


深層学習
改訂第2版
Deep Learning
岡谷貴之
Machine Learning Professional Series
MLP

行列表記

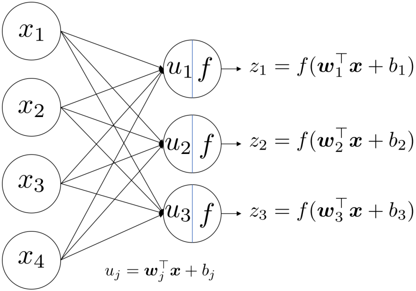


- 入力: $\boldsymbol{x} = (x_1, x_2, x_3, x_4)^\top$
- 出力: $\boldsymbol{z} = (z_1, z_2, z_3)^\top$
- $z_j = f(u_j)$, $u_j = \boldsymbol{w}_j^\top \boldsymbol{x} + b_j$

ただし、

$$\boldsymbol{w}_j = (w_{j1}, w_{j2}, \ldots, w_{jI})^\top$$

行列表記

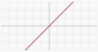
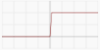
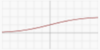
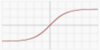
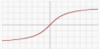
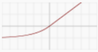
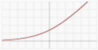


活性化関数も込みで、下記のように表す

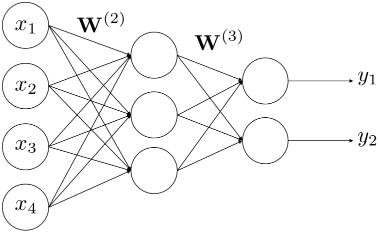
$$\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{u})$$

ただし $\boldsymbol{u} = \mathbf{W}\boldsymbol{x} + \boldsymbol{b}$

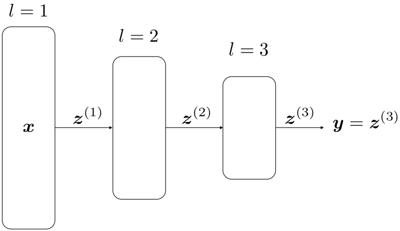
活性化関数

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x \neq 0 \\ ? & \text{for} \quad x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \dfrac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \dfrac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \dfrac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for} \quad x < 0 \\ x & \text{for} \quad x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \dfrac{1}{1 + e^{-x}}$

層を増やす



x_1
x_2
x_3
x_4
$\mathbf{W}^{(2)}$
$\mathbf{W}^{(3)}$
y_1
y_2



$l = 1$
$l = 2$
$l = 3$
$\boldsymbol{x}$
$\boldsymbol{z}^{(1)}$
$\boldsymbol{z}^{(2)}$
$\boldsymbol{z}^{(3)}$
$\boldsymbol{y} = \boldsymbol{z}^{(3)}$

層を増やす

$$\boldsymbol{z}^{(1)} = \boldsymbol{x}$$

$$\begin{aligned}\boldsymbol{u}^{(2)} &= \mathbf{W}^{(2)}\boldsymbol{z}^{(1)} + \boldsymbol{b}^{(2)} \\ \boldsymbol{z}^{(2)} &= \boldsymbol{f}(\boldsymbol{u}^{(2)})\end{aligned}$$

前ページの図では、$\mathbf{W}^{(2)}$ は 3×4 の行列。

$$\begin{aligned}\boldsymbol{u}^{(3)} &= \mathbf{W}^{(3)}\boldsymbol{z}^{(2)} + \boldsymbol{b}^{(3)} \\ \boldsymbol{z}^{(3)} &= \boldsymbol{f}(\boldsymbol{u}^{(3)})\end{aligned}$$

前ページの図では、$\mathbf{W}^{(3)}$ は 2×3 の行列。
多層ニューラルネットワークの $(l+1)$ 番目の層は、直前の l 層の出力 $\boldsymbol{z}^{(l)}$ を受け取り次式を計算する

$$\begin{aligned}\boldsymbol{u}^{(l+1)} &= \mathbf{W}^{(l+1)}\boldsymbol{z}^{(l)} + \boldsymbol{b}^{(l+1)} \\ \boldsymbol{z}^{(l+1)} &= \boldsymbol{f}(\boldsymbol{u}^{(l+1)})\end{aligned}$$

略記法

以下、バイアス項 b も重みに組み込んで記述を見やすくする

- $\boldsymbol{x} \leftarrow (\boldsymbol{x}^\top, 1)$
- $\boldsymbol{w} \leftarrow (\boldsymbol{w}^\top, b)$
- $\boldsymbol{w}^\top \boldsymbol{x} \leftarrow \boldsymbol{w}^\top \boldsymbol{x} + b$

目的

学習データの集合, $\mathcal{X} = \{(\boldsymbol{x}_1, \boldsymbol{d}_1), (\boldsymbol{x}_2, \boldsymbol{d}_2), \ldots, (\boldsymbol{x}_N, \boldsymbol{d}_N)\}$ を用いて、ニューラルネットワークが望ましい関数を表すように、各層の重み係数 $\mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \ldots$ を求めること

最後の層 L からの出力 $\boldsymbol{y} = \boldsymbol{z}^{(L)}$ は入力 $\boldsymbol{x}$ と重み係数 $\mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \ldots, \mathbf{W}^{(L)}$ の関数：

$$\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{y}(\boldsymbol{x}; \mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \ldots, \mathbf{W}^{(L)})$$

(すべての重み係数を $\boldsymbol{w}$ で略記: $\boldsymbol{w} = \{\mathbf{W}^{(2)}, \mathbf{W}^{(3)}, \ldots, \mathbf{W}^{(L)}\}$)

学習の方針

方針

データ $\boldsymbol{x}_n$ を入力したときの出力, $\boldsymbol{y}(\boldsymbol{x}_n; \boldsymbol{w})$, と $\boldsymbol{d}_n$ の誤差を $L_n(\boldsymbol{w})$ で表す。全データに対する誤差の総和を小さくするように重み係数 $\boldsymbol{w}$ を求める

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \sum_n L_n(\boldsymbol{w})$$

誤差の測り方は問題に応じて変える

係数の求め方

コスト関数の勾配

$$\nabla L(\boldsymbol{w}) = \frac{\partial L}{\partial \boldsymbol{w}} = \left[\frac{\partial L}{\partial w_1}, \ldots, \frac{\partial L}{\partial w_M}\right]^\top$$

勾配降下法によるコストの更新

1. $\boldsymbol{w} = \boldsymbol{w}^0$ で初期化する
2. $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \epsilon \nabla L$ 　による更新を繰り返す。

鞍点でよく停まる

コスト関数の鞍点

確率的勾配降下法

学習データの一部のみで勾配を求めて係数 $\boldsymbol{w}$ を更新する

Stochastic Gradient Descent: SGD

1. N 個の学習データの内の 1 つ, $\boldsymbol{x}_{n^*}$, をランダムに選択
2. $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \epsilon \nabla L_{n^*}$ により係数を更新。

鞍点を回避しやすい

注：学習率

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \epsilon \nabla L_{n^*}$$

係数 ϵ を学習率と呼ぶ。

- 小さすぎると収束に時間がかかる
- 大きすぎると局所解を通り過ぎる

実験により適切な値を定める必要がある

最適化法について

局所解と大域解

Neural Network の学習で得られる解は（ほぼ全て）局所解

- 局所解：近傍でコスト最小
  - 複数の局所解が存在
  - 初期値に依存して得られる解は変化
- 大域解：定義域全体の中でコスト最小

勾配降下法

- 確率的勾配降下法 (SGD): 訓練データをランダムに一つずつ利用
- ミニバッチ勾配降下法：訓練データの一部を複数個利用
- 勾配降下法：訓練データの全てを毎回利用

ミニバッチとエポック

- ミニバッチ：訓練データを複数の「ミニバッチ」に分割
  - 訓練データをランダムに並べ替えて M 分割 $(\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_M)$
- エポック：全てのデータを 1 回ずつ更新に使うこと
  - 全データを M 個のバッチに分割して，全てのバッチ $\mathcal{B}_1, \mathcal{B}_2, \ldots, \mathcal{B}_M$ を 1 回ずつ更新に使うと 1 エポック

モメンタム

各種勾配降下法：更新時のパラメータの値における勾配のみ利用

- 収束に時間がかかることが多い

モメンタムを利用する更新法：過去の履歴を参照

- momentum ＝ 「慣性」
    - 過去の更新傾向を今回の更新に反映
    - 振動成分を無視して「傾向」を求める

モメンタム SGD

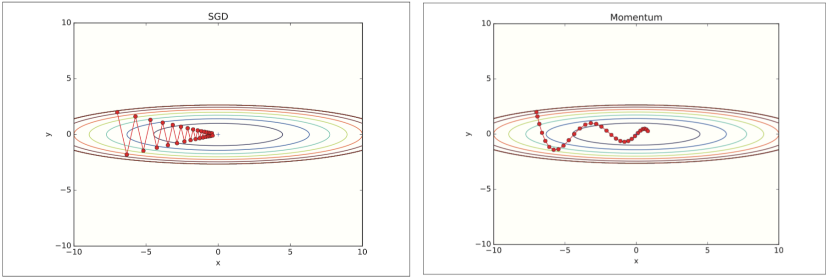
モメンタム項を加えた SGD の更新式

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \epsilon \nabla L_{n^*} + \mu \boldsymbol{v}^{t}$$

右辺第三項がモメンタム項（慣性項）

$$\boldsymbol{v}^{t} = \boldsymbol{w}^{t} - \boldsymbol{w}^{t-1}$$

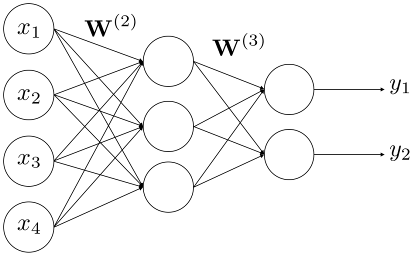
SGD vs Momentum SGD



https://www.dragonarrow.work/articles/195

誤差逆伝搬

誤差逆伝播法（準備）



2 層ネットワークを題材に

入力は $\boldsymbol{x} = (x_1, x_2, x_3, x_4)^\top$。 $z_j^{(1)} = x_j$ とする。
次の層の出力は次のとおり

$$z_j^{(2)} = f(u_j^{(2)}) = f\left(\sum_i w_{ji}^{(2)} z_i^{(1)}\right)$$

誤差逆伝播法（準備）

出力層は次のとおり。回帰を想定して活性化関数は恒等写像

$$y_j(\boldsymbol{x}) = z_j^{(3)} = u_j^{(3)} = \sum_i w_{ji}^{(3)} z_i^{(2)}$$

誤差関数は二乗誤差とする

$$L_n = \frac{1}{2}\|\boldsymbol{y}(\boldsymbol{x}_n) - \boldsymbol{d}_n\|^2 = \frac{1}{2}\sum_j (y_j(\boldsymbol{x}_n) - d_{nj})^2$$

誤差逆伝播法

まず出力層の重みで微分

$$\frac{\partial L_n}{\partial w_{ji}^{(3)}} = (\boldsymbol{y}(\boldsymbol{x}_n) - \boldsymbol{d}_n)^\top \frac{\partial \boldsymbol{y}}{\partial w_{ji}^{(3)}}$$

$\partial \boldsymbol{y}/\partial w_{ji}^{(3)}$ は第 j 成分のみ $z_i^{(2)}$。

$$\frac{\partial \boldsymbol{y}}{\partial w_{ji}^{(3)}} = [0, \ldots, 0, z_i^{(2)}, 0, \ldots, 0]$$

よって

$$\frac{\partial L_n}{\partial w_{ji}^{(3)}} = (y_j(\boldsymbol{x}_n) - d_j) z_i^{(2)}$$

誤差逆伝播法

中間層の重みによる微分 $\partial L_n / \partial w_{ji}^{(2)}$ の計算。重み $w_{ji}^{(2)}$ は $u_j^{(2)} = \sum_i w_{ji}^{(2)} z_i(1)$ の中でのみ現れる。

$$\frac{\partial L_n}{\partial w_{ji}^{(2)}} = \frac{\partial L_n}{\partial u_j^{(2)}} \frac{\partial u_j^{(2)}}{\partial w_{ji}^{(2)}}$$

右辺第二項は $u_j^{(2)} = \sum_i w_{ji}^{(2)} z_i^{(1)}$ より次の通り

$$\frac{\partial u_j^{(2)}}{\partial w_{ji}^{(2)}} = z_i^{(1)}$$

右辺第一項 $\partial L_n / \partial u_j^{(2)}$ は、$u_j^{(2)}$ の変化による L_n の変化を表現。$u_j^{(2)}$ の変化は活性化関数を介して出力層の $u_k^{(3)}$ を変化させる。L_n は $u_k^{(3)}$ の変化に伴い変化する。

$$\frac{\partial L_n}{\partial u_j^{(2)}} = \sum_k \frac{\partial L_n}{\partial u_k^{(3)}} \frac{\partial u_k^{(3)}}{\partial u_j^{(2)}}$$

誤差逆伝播法

$L_n = 1/2 \sum_k (y_k(\boldsymbol{x}_n) - d_k)^2 = 1/2 \sum_k (u_k^{(3)} - d_k)^2$

$$\frac{\partial L_n}{\partial u_k^{(3)}} = u_k^{(3)} - d_k$$

$u_k^{(3)} = \sum_i w_{ki}^{(3)} f(u_i^{(2)})$ より

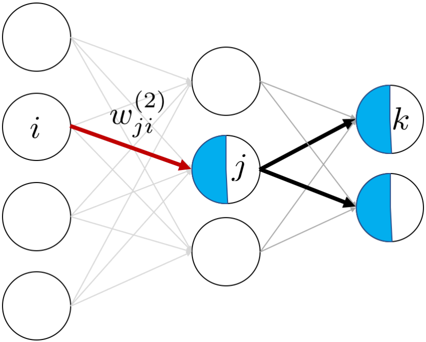
$$\frac{\partial u_k^{(3)}}{\partial u_j^{(2)}} = w_{kj}^{(3)} f'(u_j^{(2)})$$

以上より

$$\frac{\partial L_n}{\partial w_{ji}^{(2)}} = \left(f'(u_j^{(2)}) \sum_k w_{kj}^{(3)} (u_k^{(3)} - d_k) \right) z_i^{(1)}$$

誤差逆伝播法

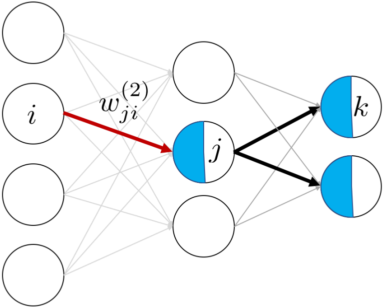
$$\frac{\partial L_n}{\partial w_{ji}^{(2)}} = \left(f'(u_j^{(2)}) \sum_k w_{kj}^{(3)} (u_k^{(3)} - d_k) \right) z_i^{(1)}$$



誤差逆伝播法

より一般的に

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



右辺第一項は $u_j^{(l)}$ の変化により L_n に生じる変化。$u_j^{(l)}$ の変化は、次の $(l+1)$ 層の $u_k^{(l+1)}$ を介して出力を変化させて、コスト関数に影響を与える。

誤差逆伝播法

$$\frac{\partial L_n}{\partial u_j^{(l)}} = \sum_k \frac{\partial L_n}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(j+1)}}{\partial u_j^{(l)}}$$

ここで、第 l 層の第 j 番目のユニットについて、次式を定義する

$$\delta_j^{(l)} = \frac{\partial L_n}{\partial u_j^{(l)}}$$

$u_k^{(l+1)} = \sum_i w_{ki}^{(l+1)} f(u_i^{(l)})$ より

$$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} \left(w_{kj}^{(l+1)} f'(u_j^{(l)}) \right)$$

$\delta_j^{(l)}$ は、ひとつ上の層の $\delta_k^{(l+1)}$ から計算できる。

誤差逆伝播法

出力層から入力層に向かって $\delta_j^{(l)}$ の計算を「伝播」させる。
$\partial u_j^{(l)}/\partial w_{ji}^{(l)} = z_l^{(l-1)}$ より結局次式を得る

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

各計算にネットワーク全体を見る必要はない。局所的に計算可能。　出力層での微分は採用したコスト関数に依存して変わる。

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

誤差逆伝播法

誤差逆伝播による勾配の計算方法

- 入力：学習データ $(\boldsymbol{x}_n, \boldsymbol{d}_n)$
- 出力：コスト関数 $L_n(\boldsymbol{w})$ の、各重み係数による微分, $\partial L_n / \partial w_{ji}^{(l)}$

1. $\boldsymbol{z}^{(1)} = \boldsymbol{x}_n$。各層への入力 $\boldsymbol{u}^{(l)}$ と出力 $\boldsymbol{z}^{(l)}$ を順に計算する
2. 出力層での $\delta_j^{(L)}$ を計算する。普通は $\delta_j^{(L)} = z_j - d_j$
3. 中間層での $\delta_j^{(l)}$ を次式に従って逆向きに計算する $(l = L, L-1, \dots)$

$$\delta_j^{(l)} = \sum_k \delta^{(l+1)} \left(w_{kj}^{(l+1)} f'(u_j^{(l)}) \right)$$

4. 各層 (l) のパラメータによる微分を次式で計算する

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