ニューラルネットワーク

中田和秀

東京工業大学 工学院 経営工学系

機械学習入門

http://www.iee.e.titech.ac.jp/~nakatalab/text/lecture.html

概要

ここではニューラルネットワークについて説明をする。この手法は、近年飛躍的に研究が進み社会実装もさかんに行われている深層学習のベースとなったものである。よって、本スライドの内容は深層学習の理解にも役立つ。

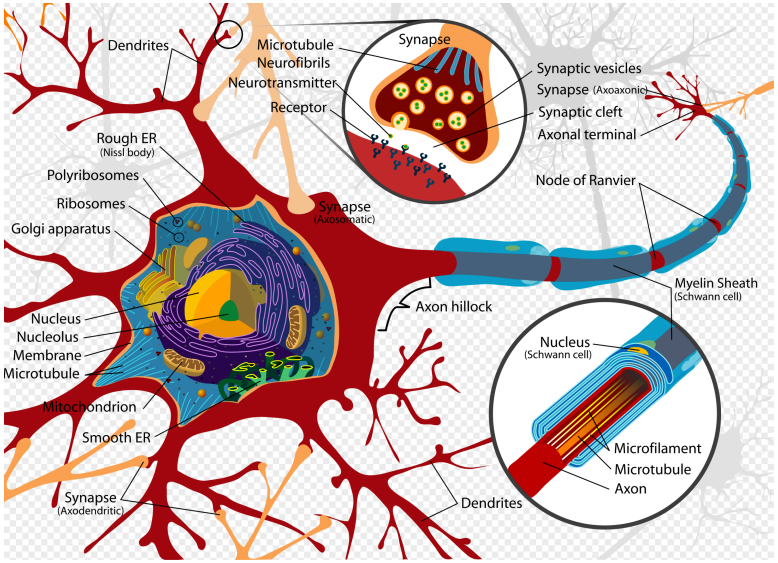
目次：

記号の使い方：

- $A := B$ は、B で A を定義する、B を A に代入することを意味する
- $[n]$ は n までのインデックスの集合を表し $[n] := \{1, 2, \cdots, n\}$

ニューラルネットワーク・深層学習

ニューロンにおける一定以上の電気信号を受けとるとシナプスに信号を流す仕組みの模倣



https://ja.wikipedia.org/wiki/神経細胞

- 回帰、判別、特徴抽出など様々な用途に利用でき応用範囲が大きい
- 中間層が多いニューラルネットワークを深層学習と呼ぶ。
  上手く学習させるには、ネットワークや学習に様々な工夫が必要が、大量のデータとそれを処理する計算資源があれば汎化性能は高い

回帰と分類

データ：$\{(\boldsymbol{x}_d, \boldsymbol{y}_d)\}_{d\in[D]}$　　$\boldsymbol{x}_d \in \mathbb{R}^n$

２つのタスク

- （ベクトル）回帰

  $\boldsymbol{y}_d \in \mathbb{R}^m$　　$\cdots$　　出力が複数ある回帰

  回帰関数　　$f : \mathbb{R}^n \to \mathbb{R}^m$

- 多クラス判別

  $\boldsymbol{y}_d \in \mathbb{R}^m$　　$\cdots$　　どのクラスに所属しているかを表す one-hot ベクトル

  判別関数　　$f : \mathbb{R}^n \to \mathbb{R}^m$

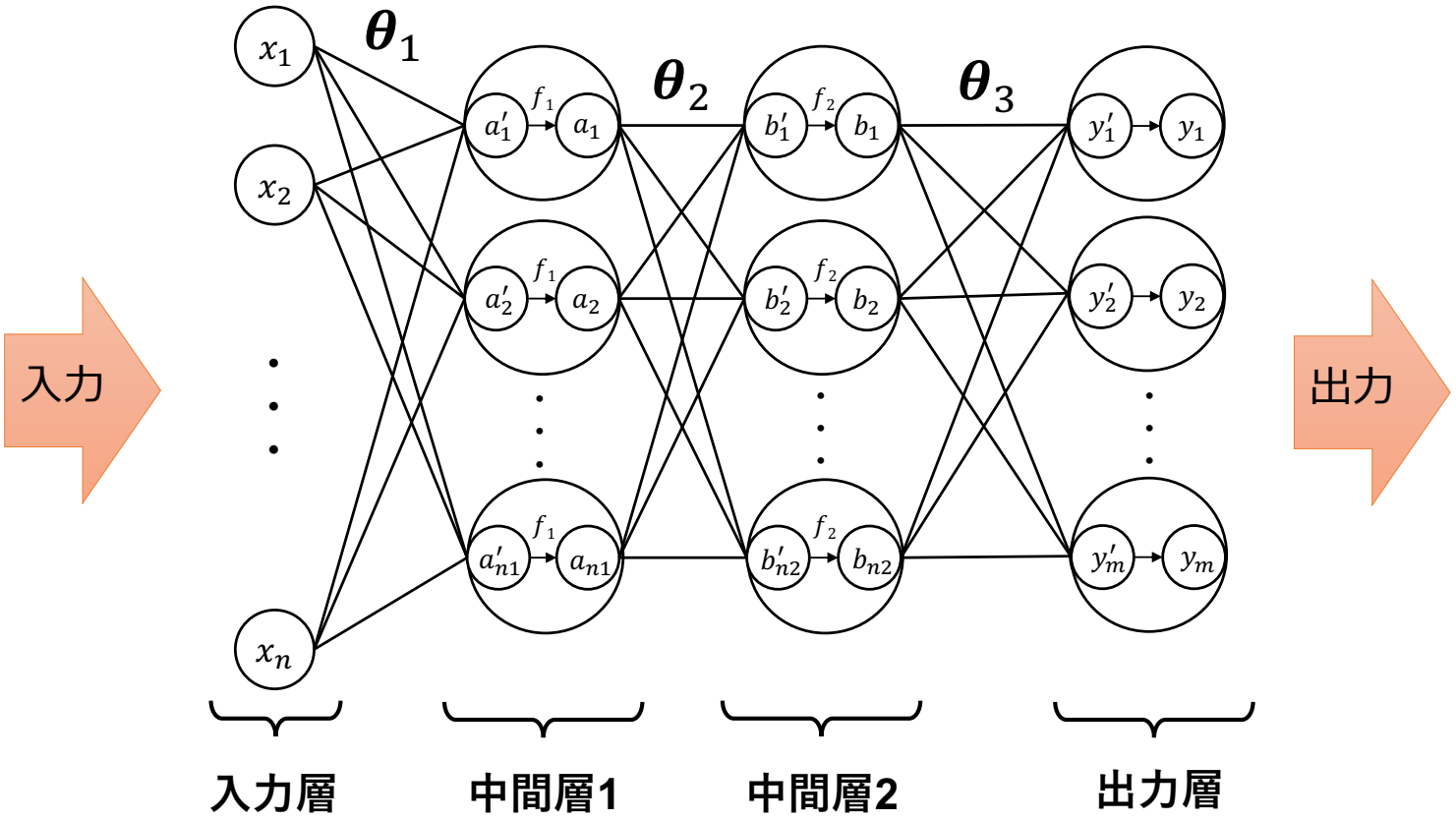
  各クラスの帰属確率を求める確率的判別モデル

※ 出力層の計算と誤差関数以外、あまり変わらない。

以下では、４層のネットワークを例に取り説明を行う

回帰のネットワーク

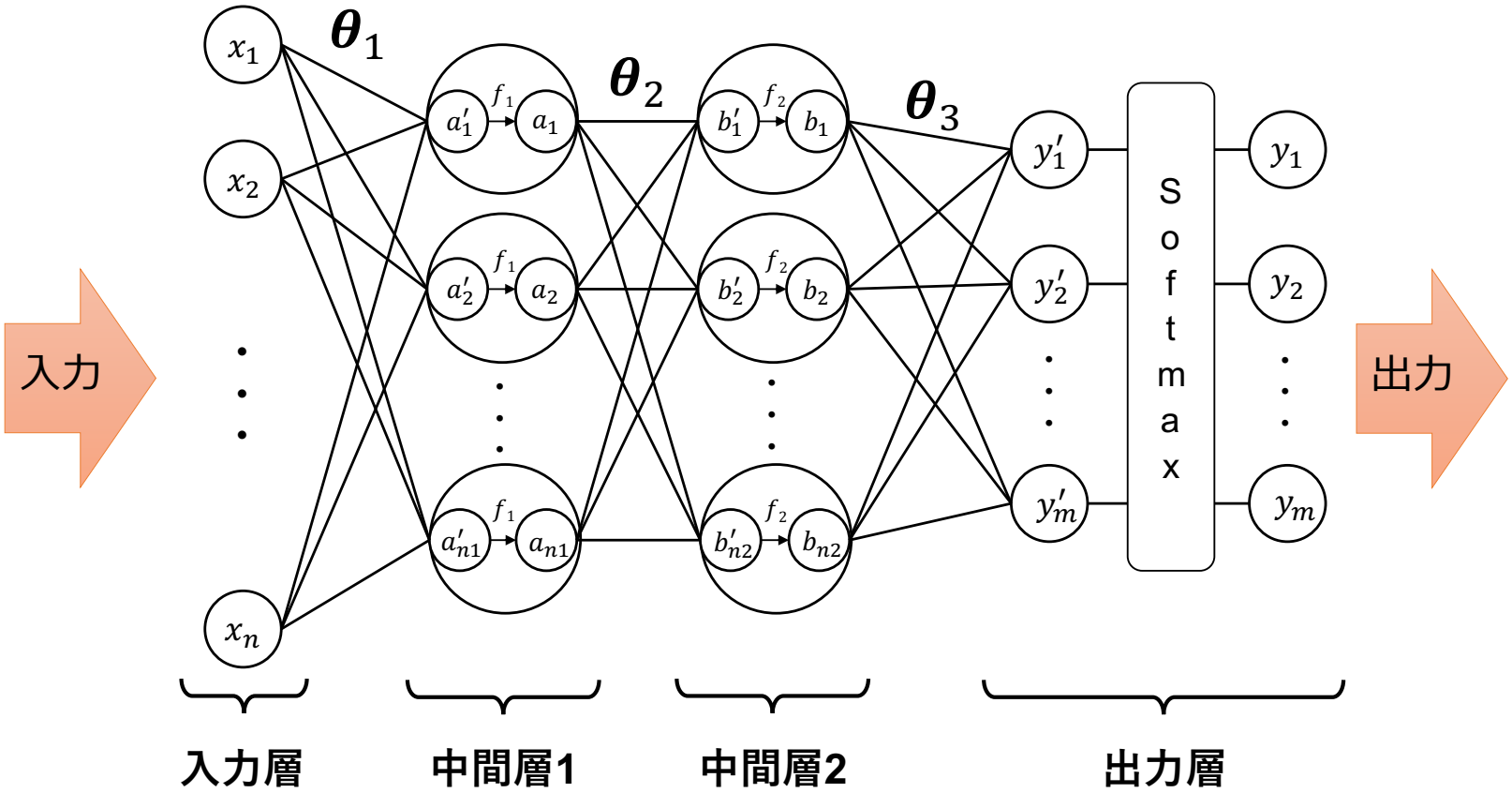
4 層の全結合型ネットワーク（入力層、中間層１，中間層２、出力層）を考える。



$x_1, x_2, \cdots, x_n, a'_1, \cdots, a_1, \cdots, b'_1, \cdots, y_m$ は一つの実数に相当する

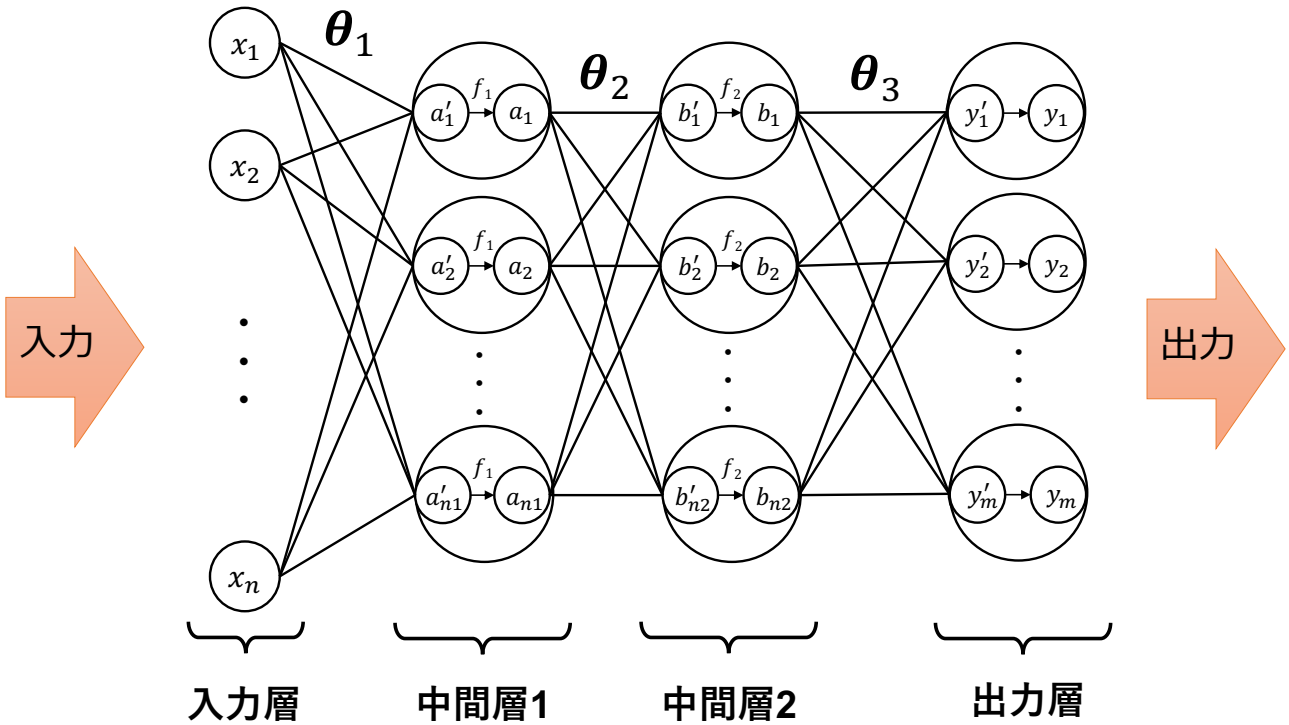
判別のネットワーク

４層の全結合型ネットワーク（入力層、中間層１，中間層２、出力層）を考える。



回帰の場合と出力層のみ変わる

各層を表すベクトル



入力層：	$\boldsymbol{x} = (x_1, x_2, \cdots, x_n)^T \in \mathbb{R}^n$	入力ベクトル
中間層１：	$\boldsymbol{a}' = (a'_1, a'_2, \cdots, a'_{n_1})^T \in \mathbb{R}^{n_1}$	中間層１の入力
	$\boldsymbol{a} = (a_1, a_2, \cdots, a_{n_1})^T \in \mathbb{R}^{n_1}$	中間層１の出力
中間層２：	$\boldsymbol{b}' = (b'_1, b'_2, \cdots, b'_{n_2})^T \in \mathbb{R}^{n_2}$	中間層２の入力
	$\boldsymbol{b} = (b_1, b_2, \cdots, b_{n_2})^T \in \mathbb{R}^{n_2}$	中間層２の出力
出力層：	$\boldsymbol{y}' = (y'_1, y'_2, \cdots, y'_m)^T \in \mathbb{R}^m$	出力層の入力
	$\boldsymbol{y} = (y_1, y_2, \cdots, y_m)^T \in \mathbb{R}^m$	出力ベクトル

ベクトルの変換

$\boldsymbol{x} \in \mathbb{R}^n$ 入力層

↓ 層間の伝播

$\boldsymbol{a}' \in \mathbb{R}^{n_1}$ 中間層１の入力

↓ 中間層のノード内の変換

$\boldsymbol{a} \in \mathbb{R}^{n_1}$ 中間層１の出力

↓ 層間の伝播

$\boldsymbol{b}' \in \mathbb{R}^{n_2}$ 中間層２の入力

↓ 中間層のノード内の変換

$\boldsymbol{b} \in \mathbb{R}^{n_2}$ 中間層２の出力

↓ 層間の伝播

$\boldsymbol{y}' \in \mathbb{R}^m$ 出力層の入力

↓ 出力層の（ノード内の）変換

$\hat{\boldsymbol{y}} \in \mathbb{R}^m$ 出力層の出力

(1) 層間の伝播

層間の変換は線形変換（正確にはアフィン変換）

入力層から中間層１：　$\boldsymbol{x}$ を $\boldsymbol{a}'$ に変換

パラメタ $\boldsymbol{\theta}_1 := \{\boldsymbol{M}_1, \boldsymbol{v}_1\}$　ただし $\boldsymbol{M}_1 \in \mathbb{R}^{n_1 \times n}$, $\boldsymbol{v}_1 \in \mathbb{R}^{n_1}$

$$\boldsymbol{a}' = f_{\theta_1}(\boldsymbol{x}) := \boldsymbol{M}_1\boldsymbol{x}+\boldsymbol{v}_1$$

中間層１から中間層２：　$\boldsymbol{a}$ を $\boldsymbol{b}'$ に変換

パラメタ $\boldsymbol{\theta}_2 := \{\boldsymbol{M}_2, \boldsymbol{v}_2\}$　ただし $\boldsymbol{M}_2 \in \mathbb{R}^{n_2 \times n_1}$, $\boldsymbol{v}_2 \in \mathbb{R}^{n_2}$

$$\boldsymbol{b}' = f_{\theta_2}(\boldsymbol{a}) := \boldsymbol{M}_2\boldsymbol{a}+\boldsymbol{v}_2$$

中間層２から出力層：　$\boldsymbol{b}$ を $\boldsymbol{y}'$ に変換

パラメタ $\boldsymbol{\theta}_3 := \{\boldsymbol{M}_3, \boldsymbol{v}_3\}$　ただし $\boldsymbol{M}_3 \in \mathbb{R}^{m \times n_2}$, $\boldsymbol{v}_3 \in \mathbb{R}^{m}$

$$\boldsymbol{y}' = f_{\theta_3}(\boldsymbol{b}) := \boldsymbol{M}_3\boldsymbol{b}+\boldsymbol{v}_3$$

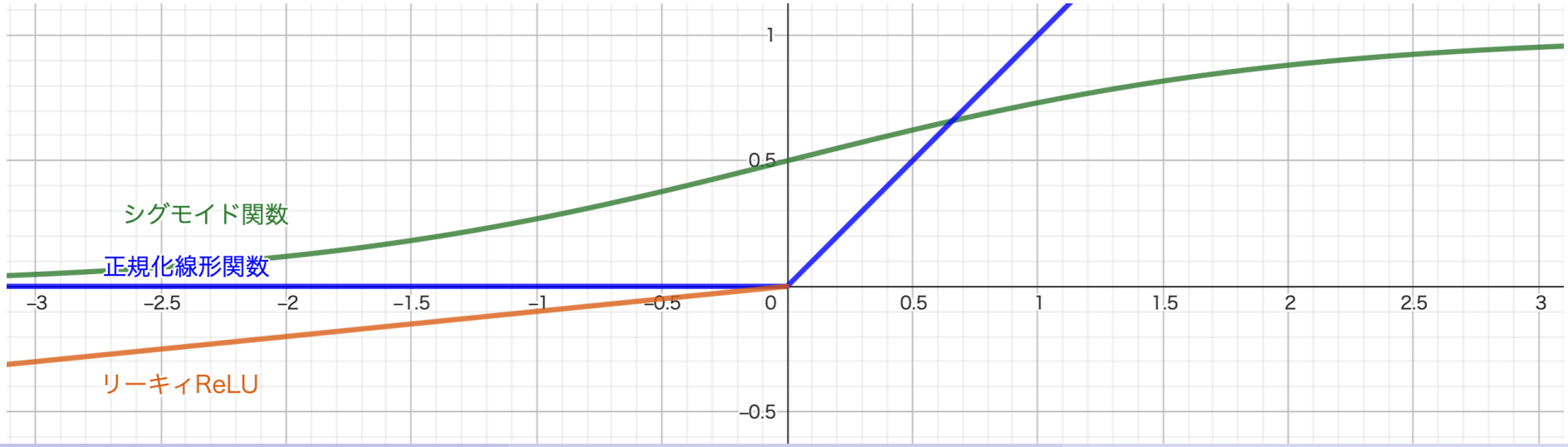
※ 同じ層の変数を結合する働きがある

(2) 中間層の活性化関数

ノード内での変換は非線形な活性化関数

例：

- シグモイド関数： $g(x) = \dfrac{1}{1+e^{-x}}$
- 正規化線形関数 (ReLU)： $g(x) = \max\{0, x\}$
- リーキィ ReLU： $g(x;\alpha) = \begin{cases} x & (x \geq 0) \\ \alpha x & (x < 0) \end{cases}$



(2) 中間層の活性化関数

- 中間層の活性化関数 $g : \mathbb{R} \to \mathbb{R}$
- 入力をベクトルにした関数 $g : \mathbb{R}^n \to \mathbb{R}^n$

$$g(\boldsymbol{x}) := \begin{pmatrix} g(x_1) \\ g(x_2) \\ \vdots \\ g(x_n) \end{pmatrix} \in \mathbb{R}^n$$

変数ごとに活性化関数 g を適用する

変数間の結合はしない

中間層１： $\boldsymbol{a}'$ を $\boldsymbol{a}$ に変換

$$\boldsymbol{a} = g_1(\boldsymbol{a}')$$

中間層２： $\boldsymbol{b}'$ を $\boldsymbol{b}$ に変換

$$\boldsymbol{b} = g_2(\boldsymbol{b}')$$

(3) 出力層の活性化関数

回帰の場合： 恒等写像を使う（何もしない）

$$\hat{\boldsymbol{y}} = h(\boldsymbol{y}') := \boldsymbol{y}'$$

判別の場合： ソフトマックス関数を使う。

$$\hat{\boldsymbol{y}} = h(\boldsymbol{y}') := \frac{1}{\sum_{i \in [m]} \exp\{y'_i\}} \begin{pmatrix} \exp\{y'_1\} \\ \exp\{y'_2\} \\ \vdots \\ \exp\{y'_m\} \end{pmatrix}$$

- $\hat{\boldsymbol{y}}$ は各要素が 0 以上 1 以下で和が 1 となるため確率とみなせる
- 中間層の活性化関数のようにノード内で閉じた変換ではないことに注意

予測器

予測器

回帰・判別関数　　$f : \mathbb{R}^n \to \mathbb{R}^m$

$$\hat{\boldsymbol{y}} = f(\boldsymbol{x}) := h(f_{\theta_3}(g_2(f_{\theta_2}(g_1(f_{\theta_1}(\boldsymbol{x}))))))$$

パラメタは $\boldsymbol{\theta} := \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3\}$

単純な関数の合成で複雑な予測器 f を表現している。

- 単純で扱いやすい関数

  $f_{\theta_1}, f_{\theta_2}, f_{\theta_3}$：　全変数に作用、線形

  g_1, g_2：　　各変数に作用、非線形
- 複雑で扱いにくい関数

  f：　　全変数に作用、非線形

ネットワークの層数が増えれば、f_θ, g_i が増えるだけ

誤差関数

目標 $\boldsymbol{y}$ と予測値 $\hat{\boldsymbol{y}}$ との誤差を定義する。

誤差関数

回帰の場合： ２乗誤差

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) := \frac{1}{2}\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|^2$$

判別の場合： クロスエントロピー誤差

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) := \sum_{i \in [m]} -y_i \log \hat{y}_i$$

クロスエントロピー誤差

- $y = 1$ のクラスに対する予測確率 $\hat{y}_i$ が 1 のとき誤差は 0 で、予測確率が 0 に近いほど大きな誤差となっている。
- 2 クラス判別のクロスエントロピー誤差の拡張
  $L(y, \hat{y}) := y \log \hat{y} + (1 - y) \log(1 - \hat{y})$

学習

データ：$\{(\boldsymbol{x}_d, \boldsymbol{y}_d)\}_{d\in[D]}$

学習

パラメタは $\boldsymbol{\theta} := \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3\}$

$$\min_{\boldsymbol{\theta}} \frac{1}{D} \sum_{d\in[D]} L(\boldsymbol{y}_d, h(f_{\theta_3}(g_2(f_{\theta_2}(g_1(f_{\theta_1}(\boldsymbol{x}_d)))))))$$

最適化の変数が $\boldsymbol{\theta}$ であることを明確にするため、次の関数 [1] を考える。

$$F(\boldsymbol{\theta}; d) := L(\boldsymbol{y}_d, h(f_{\theta_3}(g_2(f_{\theta_2}(g_1(f_{\theta_1}(\boldsymbol{x}_d)))))))$$

最適化問題

$$\min_{\boldsymbol{\theta}} \frac{1}{D} \sum_{d\in[D]} F(\boldsymbol{\theta}; d)$$

[1] 独立変数は $\boldsymbol{\theta}$ であるが、d（より正確には $\boldsymbol{x}_d$ と y_d）にも依存している関数

学習アルゴリズム

勾配降下法

- 非線形最適化問題を解く最も基本的な解法である最急降下法のこと
- 探索方向は目的関数の勾配の -1 倍したもの
- 更新式は次の通り。

$$\boldsymbol{\theta}_{k+1} := \boldsymbol{\theta}_k - \alpha \left(\frac{1}{D} \sum_{d \in [D]} \nabla F(\boldsymbol{\theta}; d) \right)$$

ステップサイズ $\alpha > 0$ は学習率と呼ばれる。

勾配 $\nabla F(\boldsymbol{\theta}; d)$ の計算

- $F(\boldsymbol{\theta}; d)$ は複雑な関数だが、扱いやすい関数の合成関数
- 誤差逆伝播法によって効率よく計算することが可能（後述）

確率的勾配降下法

勾配降下法の問題点

- データ数 D が多くなると全ての勾配の計算に時間がかかる

$$\boldsymbol{\theta}_{k+1} := \boldsymbol{\theta}_k - \alpha \left(\frac{1}{D} \sum_{d \in [D]} \nabla F(\boldsymbol{\theta}; d) \right)$$

- あまり良くない局所最適解に収束してしまう

確率的勾配降下法 (stochastic gradient descent method: SGD)

- ニューラルネットワークや深層学習に対する最も基本的な学習アルゴリズム
- 基本的な枠組みは勾配降下法と同じで、違いは勾配を近似計算するところ

勾配の近似計算

- 全勾配の代わりに、データの１部分（ミニバッチ）の勾配を使う。
- ミニバッチを $S \subset \{1, 2, \cdots, D\}$ とすると、更新式は次の通り。

$$\boldsymbol{\theta}_{k+1} := \boldsymbol{\theta}_k - \alpha \left(\frac{1}{|S|} \sum_{d \in S} \nabla F(\boldsymbol{\theta}; d) \right)$$

- 各反復でのミニバッチの選び方（非復元抽出）
  バッチサイズを bs とする
  - 1 反復目では、D 個のデータからランダムに bs 個を抜き出し、それをミニバッチとする。
  - 2 反復目以降は、残ったデータからランダムに bs 個抜き出しミニバッチとする。
  - データ全体を使いきったら、また D 個の全データを戻して上の作業を繰り返す。

  D 個の全データを使い切る期間をエポックと呼ぶ。

誤差逆伝播法 1

予測器： $F(\boldsymbol{\theta}; d) := L(\boldsymbol{y}_d, h(f_{\theta_3}(g_2(f_{\theta_2}(g_1(f_{\theta_1}(\boldsymbol{x}_d)))))))$

$$F \overset{L}{\leftarrow} \hat{\boldsymbol{y}} \overset{h}{\leftarrow} \boldsymbol{y}' \overset{f_{\theta_3}}{\leftarrow} \boldsymbol{b} \overset{g_2}{\leftarrow} \boldsymbol{b}' \overset{f_{\theta_2}}{\leftarrow} \boldsymbol{a} \overset{g_1}{\leftarrow} \boldsymbol{a}' \overset{f_{\theta_1}}{\leftarrow} \boldsymbol{x}_d$$

まず、勾配 $\nabla F(\boldsymbol{\theta}_1; d)$ すなわち $\left(\dfrac{\partial F}{\partial \boldsymbol{\theta}_1}\right)^T$ について考える。連鎖律より

$$\frac{\partial F}{\partial \boldsymbol{\theta}_1} = \frac{\partial F}{\partial \boldsymbol{y}'} \frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{b}} \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{b}'} \frac{\partial \boldsymbol{b}'}{\partial \boldsymbol{a}} \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{a}'} \frac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1} \in \mathbb{R}^{1 \times |\boldsymbol{\theta}_1|}$$

$$\frac{\partial F}{\partial \boldsymbol{y}'} \in \mathbb{R}^{1 \times m}, \quad \frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{b}} \in \mathbb{R}^{m \times n_2}, \quad \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{b}'} \in \mathbb{R}^{n_2 \times n_2},$$

$$\frac{\partial \boldsymbol{b}'}{\partial \boldsymbol{a}} \in \mathbb{R}^{n_2 \times n_1}, \quad \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{a}'} \in \mathbb{R}^{n_1 \times n_1}, \quad \frac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1} \in \mathbb{R}^{n_1 \times |\boldsymbol{\theta}_1|}$$

誤差逆伝播法 2

$\dfrac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1} \in \mathbb{R}^{n_1 \times |\boldsymbol{\theta}_1|}$ 疎行列（後述）

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{a}'} = \begin{pmatrix} g_1'(a_1') & & \\ & \ddots & \\ & & g_1'(a_{n_1}') \end{pmatrix} \in \mathbb{R}^{n_1 \times n_1}$$

なぜなら $\boldsymbol{a} = g_1(\boldsymbol{a}')$

$$\frac{\partial \boldsymbol{b}'}{\partial \boldsymbol{a}} = \boldsymbol{M}_2 \in \mathbb{R}^{n_2 \times n_1}$$

なぜなら $\boldsymbol{b}' = \boldsymbol{M}_2 \boldsymbol{a} + \boldsymbol{v}_2$

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{b}'} = \begin{pmatrix} g_2'(b_1') & & \\ & \ddots & \\ & & g_2'(b_{n_2}') \end{pmatrix} \in \mathbb{R}^{n_2 \times n_2}$$

なぜなら $\boldsymbol{b} = g_2(\boldsymbol{b}')$

$$\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{b}} = \boldsymbol{M}_3 \in \mathbb{R}^{m \times n_2}$$

なぜなら $\boldsymbol{y}' = \boldsymbol{M}_3 \boldsymbol{b} + \boldsymbol{v}_3$

$$\frac{\partial F}{\partial \boldsymbol{y}'} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T \in \mathbb{R}^{1 \times m}$$

次スライド

式変形

回帰の場合： $\dfrac{\partial F}{\partial \boldsymbol{y}'} = \dfrac{\partial}{\partial \hat{\boldsymbol{y}}} \dfrac{1}{2} \|\hat{\boldsymbol{y}} - \boldsymbol{y}_d\|^2 = (\hat{\boldsymbol{y}} - \boldsymbol{y}_d)^T$

判別の場合： $\boldsymbol{y}$ は one-hot ベクトルであるが、k 番目の要素が 1 であるとする。

$$F = \sum_{i \in [m]} -y_i \log \hat{y}_i = -\log \hat{y}_k = -\log \frac{\exp\{y'_k\}}{\sum_{i \in [m]} \exp\{y'_i\}} = \log \sum_{i \in [m]} \exp\{y'_i\} \; - \; y'_k$$

k と k 以外で分けて微分すると、

$$\frac{\partial F}{\partial y'_k} = \frac{\exp\{y'_k\}}{\sum_{i \in [m]} \exp\{y'_i\}} - 1 = \hat{y}_k - 1,$$

$$j \neq k, \quad \frac{\partial F}{\partial y'_j} = \frac{\exp\{y'_j\}}{\sum_{i \in [m]} \exp\{y'_i\}} = \hat{y}_j$$

よって、 $\dfrac{\partial F}{\partial \boldsymbol{y}'} = (\hat{\boldsymbol{y}} - \boldsymbol{y})^T$ $\cdots\cdots$ 微分は簡単に計算できる

誤差逆伝播法 3

$$\frac{\partial F}{\partial \boldsymbol{\theta}_1} = \frac{\partial F}{\partial \boldsymbol{y}'}\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{b}}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{b}'}\frac{\partial \boldsymbol{b}'}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{a}'}\frac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1}$$

各層のノード数を全て n として切片項 $\boldsymbol{v}$ を除いたとき、

$$\frac{\partial F}{\partial \boldsymbol{y}'} \in \mathbb{R}^{1\times n}, \qquad \frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{b}}, \frac{\partial \boldsymbol{b}}{\partial \boldsymbol{b}'}, \frac{\partial \boldsymbol{b}'}{\partial \boldsymbol{a}}, \frac{\partial \boldsymbol{a}}{\partial \boldsymbol{a}'} \in \mathbb{R}^{n\times n}, \qquad \frac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1} \in \mathbb{R}^{n\times n^2}$$

計算量 [2]

- 入力の方から計算（連鎖律の右から計算）
  $n \times n$ のベクトルと $n \times n^2$ の行列の積： $O(n^4)$
- 出力の方から計算（連鎖律の左から計算）
  $1 \times n$ のベクトルと $n \times n$ の行列の積： $O(n^2)$

  ※　最後の $\dfrac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1}$ の掛け算は疎性を使うと簡単に計算可能（次スライド）

[2] 厳密に言うと対角行列との掛け算の部分は簡単にできるので無視してよい

誤差逆伝播法 4

行列 $\dfrac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1}$ の疎性（非ゼロ要素の割合）

行と列の意味

- $\boldsymbol{\theta}_1$ は $n \times n$ の行列を適当な順で並び替えたベクトル
- $\boldsymbol{a}'$ は中間層１の入力となるベクトル

$\boldsymbol{\theta}_1$ の一つの成分に対して、影響を与える $\boldsymbol{a}'$ は一つの成分

$\Longrightarrow$ $\dfrac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1}$ の各列に非ゼロ要素は一つ

$\Longrightarrow$ $\boldsymbol{v} \in \mathbb{R}^n$ に対し、$\boldsymbol{v}^T \dfrac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1}$ の計算に必要な計算量は $O(n^2)$

（非ゼロの部分だけ計算すればよい）

疎性があるため行列とベクトルの掛け算は簡単

誤差逆伝播法 5

同様に、 $\dfrac{\partial F}{\partial \boldsymbol{\theta}_2}, \dfrac{\partial F}{\partial \boldsymbol{\theta}_3}$ も連鎖律を使って計算できる。

$$\frac{\partial F}{\partial \boldsymbol{\theta}_1} = \frac{\partial F}{\partial \boldsymbol{y}'}\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{b}}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{b}'}\frac{\partial \boldsymbol{b}'}{\partial \boldsymbol{a}}\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{a}'}\frac{\partial \boldsymbol{a}'}{\partial \boldsymbol{\theta}_1}$$

$$\frac{\partial F}{\partial \boldsymbol{\theta}_2} = \frac{\partial F}{\partial \boldsymbol{y}'}\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{b}}\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{b}'}\frac{\partial \boldsymbol{b}'}{\partial \boldsymbol{\theta}_2}$$

$$\frac{\partial F}{\partial \boldsymbol{\theta}_3} = \frac{\partial F}{\partial \boldsymbol{y}'}\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{\theta}_3}$$

出力の方から順に計算すると、同じ計算が出てくるため省略できる

勾配の計算に必要な計算量　（s：中間層の層数、n：各層の（平均）ノード数）

- 入力の方から順に計算（連鎖律の右から計算）： $O(s^2n^4)$
- 出力の方から順に計算（連鎖律の左から計算）： $O(sn^2)$

出力の方から計算する手法を誤差逆伝播法（バックプロパゲーション）という。

パラメタの数 $O(sn^2)$ 個に比例した計算量で勾配の計算が可能で高速。

学習率

- 勾配降下法（最急降下法）は、学習率（ステップサイズ）の制御が難しい
- ちなみに、Newton 法はステップサイズを 1 にしてよい
  （Newton 法の導出過程から分かる）
- Newton 法を部分的・近似的に取り入れて安定性を向上させる研究などがある

k 反復目の学習率 α_k の例（$\dot{\alpha}, \ddot{\alpha}, K$ は定数）

- $\alpha_k = \begin{cases} \dfrac{k}{K}\ddot{\alpha} + \left(1 - \dfrac{k}{K}\right)\dot{\alpha} & (k \leq K), \\ \ddot{\alpha} & (k > K), \end{cases}$
- $\alpha_k = \dfrac{\dot{\alpha}}{k}$,
- $\alpha_k = \dfrac{\dot{\alpha}}{\sqrt{k}}$.

など

探索方向の改良

確率的勾配降下法による近似勾配

$$\boldsymbol{v} := \frac{1}{|S|} \sum_{d \in S} \nabla F(\boldsymbol{\theta}; d) \quad \approx \nabla F(\boldsymbol{\theta})$$

確率的勾配降下法での探索方向

$$\boldsymbol{d} := -\boldsymbol{v}$$

深層学習で確率的勾配降下法を使う場合は、
探索方向を各成分ごとにスケーリングして収束性を向上させる。

- RMSprop
- AdaDelta
- Adam
- AdaBelief　　など

そのとき、指数移動平均を使うことが多い

指数移動平均

時系列 $a_1, a_2, \cdots, a_k \in \mathbb{R}$ の重み付き平均を考える

- 最近（k が大きい）のものを重視する
- 過去を遡るほど $\rho \in [0, 1)$ 倍の減衰させる

- 指数移動平均 s は次の値となる。

$$\begin{aligned} s &= \frac{1-\rho}{1-\rho^k}\left(\rho^{k-1}a_1 + \rho^{k-2}a_2 + \cdots + \rho a_{k-1} + a_k\right) \\ &\approx (1-\rho)\left(\rho^{k-1}a_1 + \rho^{k-2}a_2 + \cdots + \rho a_{k-1} + a_k\right) \end{aligned}$$

$1-\rho^k$ は、k が大きいとき 1 に近い

a が増えていく状況で、その時点までの指数移動平均の逐次更新は簡単

1. 初期値として、$s := 0$ とする。
2. 新たな a が来たときに、次のように更新する。

$$s := \rho s + (1-\rho)a$$

RMSprop

確率的勾配降下法による近似勾配： $\boldsymbol{v} := \dfrac{1}{|S|}\displaystyle\sum_{d\in S}\nabla F(\boldsymbol{\theta};d) \approx \nabla F(\boldsymbol{\theta})$

要素ごとに、過去の勾配の平均で割って探索方向をつくる。

- 過去の勾配が大きい成分： 移動量が小さくなる
- 過去の勾配が小さい成分： 移動量が大きくなる

各成分をスケーリングした探索方向 $\boldsymbol{d}$

$$\forall i,\quad d_i := -\frac{1}{\sqrt{\text{過去 } v_i \text{ の２乗平均}}}\, v_i$$

RMSprop

1. $\forall i,\quad s_i := \rho s_i + (1-\rho)v_i^2$ 近似勾配の各要素の２乗の移動平均
2. $\forall i,\quad d_i := -\dfrac{1}{\sqrt{s_i}} v_i$

実際の RMSprop では、これに若干の補正を加えて計算をする。

AdaDelta

Newton 法

$$\boldsymbol{d} := -\left[\nabla^2 F(\boldsymbol{\theta})\right]^{-1} \nabla F(\boldsymbol{\theta})$$

$\left[\nabla^2 F(\boldsymbol{\theta})\right]^{-1}$ を計算するのは難しい。$\nabla^2 F(\boldsymbol{\theta})$ は対角行列であるとすると

$$\forall i, \quad d_i = -\frac{1}{\left[\nabla^2 F(\boldsymbol{\theta})\right]_{ii}} \left[\nabla F(\boldsymbol{\theta})\right]_i$$

$\left[\nabla^2 F(\boldsymbol{\theta}_k)\right]_{ii}$ を（無理やり）近似計算

- $\left[\nabla^2 F(\boldsymbol{\theta}_k)\right]_{ii}$ は $\boldsymbol{\theta}_k$ に依存しないとする。
- 過去の探索方向 $[\boldsymbol{d}_k]_i$ と近似勾配ベクトル $[\boldsymbol{v}_k]_i$ を用い近似

$$\left[\nabla^2 F(\boldsymbol{\theta}_k)\right]_{ii} \approx \frac{\text{過去の } v_i \text{ の平均}}{\text{過去の } d_i \text{ の平均}}$$

- 平均には２乗の移動平均の平方根を用いる。

AdaDelta

すると、各成分のスケーリングによって得られた探索方向 $\boldsymbol{d}$ は

$$\forall i, \quad d_i := -\frac{\sqrt{\text{過去 } d_i \text{ の２乗平均}}}{\sqrt{\text{過去 } v_i \text{ の２乗平均}}} v_i$$

AdaDelta

1. $\forall i, \quad s_i := \rho s_i + (1-\rho) v_i^2$ 近似勾配の各要素の２乗の移動平均
2. $\forall i, \quad t_i := \rho t_i + (1-\rho) d_i^2$ 探索方向の各要素の２乗の移動平均
3. $\forall i, \quad d_i := -\frac{\sqrt{t_i}}{\sqrt{s_i}} v_i$

実際の AdaDelta では、これに若干の補正を加えて計算をする。

Adam

- 探索方向を過去の探索方向で補正する。
  これまで全ての近似勾配 $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_k\}$ の移動平均を取る。

$$\boldsymbol{u} = (1-\rho)\left(\rho^{k-1}\boldsymbol{v}_1 + \rho^{k-2}\boldsymbol{v}_2 + \cdots + \rho\boldsymbol{v}_{k-1} + \boldsymbol{v}_k\right)$$

- 各成分のスケーリングは RMSprop と同じ。
- ２つの移動平均を計算するとき、$1-\rho^k$ で割る補正を加える。

すると、各成分のスケーリングによって得られた探索方向 $\boldsymbol{d}$ は

$$\forall i, \quad d_i := - \text{係数の補正} \frac{1}{\sqrt{\text{過去 } v_i \text{ の２乗平均}}} \text{過去 } v_i \text{ の平均}$$

Adam

Adam

1. $\forall i,\ \ s_i := \rho_1 s_i + (1-\rho_1) v_i^2$ 近似勾配の各要素の２乗の移動平均
2. $\forall i,\ \ u_i := \rho_2 u_i + (1-\rho_2) v_i$ 近似勾配の各要素の移動平均
3. $\forall i,\ \ d_i := -\dfrac{\sqrt{1-\rho_2^k}}{1-\rho_1^k} \dfrac{1}{\sqrt{s_i}} u_i$

実際の Adam では、これに若干の補正を加えて計算をする。

経験的に Adam は収束が早いことが知られており、よく使用される

ニューラルネットワークの判別面（２クラス分類）