KYOTO UNIVERSITY

統計的モデリング基礎⑥
～ニューラルネットワーク～

鹿島久嗣
（情報学科 計算機科学コース）

DEPARTMENT OF INTELLIGENCE SCIENCE AND TECHNOLOGY

ロジスティック回帰

判別問題：
ダミー変数を従属変数として説明（予測）する問題

- データ（n 組の独立変数と従属変数）
  - 独立変数：$\left(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n)}\right)$
  - （ダミー）従属変数：$\left(y^{(1)}, y^{(2)}, \ldots, y^{(n)}\right), y^{(i)} \in \{+1, -1\}$

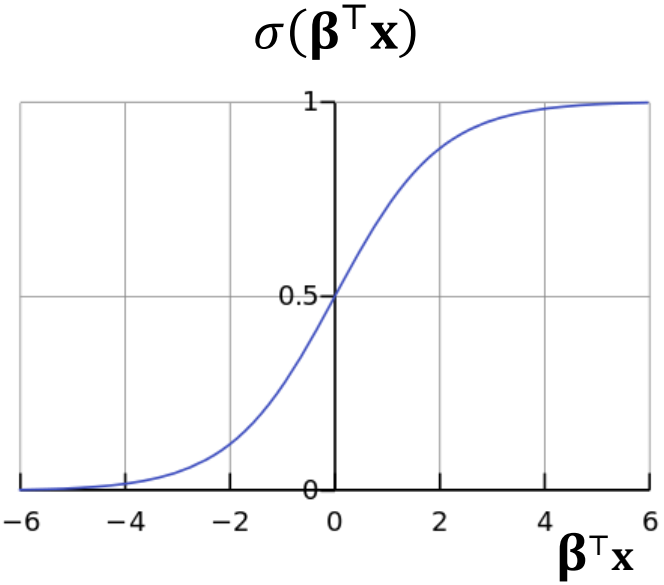
以降、表記上の利便性からダミー従属変数を
$\{0,1\}$ でなく $\{+1, -1\}$ と表記する
（本質的な違いはナシ）

ロジスティック回帰：ダミー変数を従属変数とするモデル

- 従属変数の値域が２値 $\{-1, +1\}$ であるモデル
- ロジスティック回帰モデルは$Y = +1$となる確率を与える：

$$P(Y = 1|\mathbf{x}; \boldsymbol{\beta}) = \frac{1}{1+\exp(-\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x})} = \sigma(\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x})$$

- σ：ロジスティック関数（$\sigma: \mathbb{R} \to (0,1)$）
  - 実数を確率に変換する



ロジスティック回帰のパラメータ推定：パラメータの繰り返し更新によって最適解を求める

- 最尤推定の目的関数（最大化）：

$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \log\left(1 + \exp\left(-y^{(i)}\boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}^{(i)}\right)\right)$$

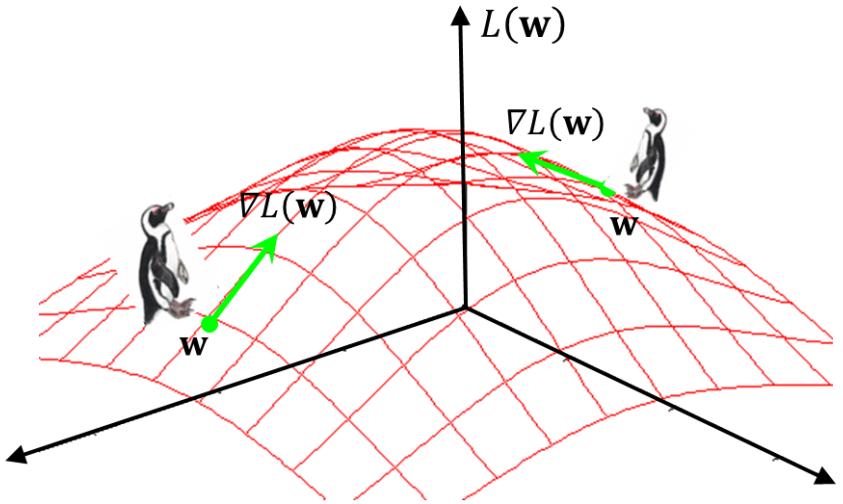
  - ＝ 負の対数尤度を目的関数として最小化
- 数値的な最適化手法によりパラメータを推定する
- パラメータの更新をくりかえす： $\boldsymbol{\beta}^{\mathrm{NEW}} \leftarrow \boldsymbol{\beta} + \mathbf{d}$

最急降下法※
最もシンプルな最適化手法

- 最急降下法：

$$\boldsymbol{\beta}^{\mathrm{NEW}} \leftarrow \boldsymbol{\beta} + \eta \nabla L(\boldsymbol{\beta})$$

  - 勾配$\nabla L(\boldsymbol{\beta})$ は最も急な（目的関数が最も増加する）向き
  - 学習率 η は線形探索で求める：



※ 最急降下法の名前は、最小化問題を想定しているが、ここでは最大化問題としているので降下じゃなくて上昇しているが、便宜上こう呼ぶことにする

ロジスティック回帰の勾配計算：比較的簡単に計算可能

- 対数尤度：$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \ln\left(1 + \exp\left(-y^{(i)}\boldsymbol{\beta}^{\top}\mathbf{x}^{(i)}\right)\right)$

- $$\frac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = -\sum_{i=1}^{n} \frac{1}{1+\exp\left(-y^{(i)}\boldsymbol{\beta}^{\top}\mathbf{x}^{(i)}\right)} \frac{\partial\left(1+\exp\left(-y^{(i)}\boldsymbol{\beta}^{\top}\mathbf{x}^{(i)}\right)\right)}{\partial \boldsymbol{\beta}}$$

$$= \sum_{i=1}^{n} \frac{1}{1 + \exp\left(-y^{(i)}\boldsymbol{\beta}^{\top}\mathbf{x}^{(i)}\right)} \exp\left(-y^{(i)}\boldsymbol{\beta}^{\top}\mathbf{x}^{(i)}\right) y^{(i)}\mathbf{x}^{(i)}$$

$$= \sum_{i=1}^{n} \left(1 - \frac{1}{1+\exp\left(-\boldsymbol{\beta}^{\top}\mathbf{x}^{(i)}\right)}\right) y^{(i)}\mathbf{x}^{(i)} = \sum_{i=1}^{n} \frac{y^{(i)}\mathbf{x}^{(i)}}{1+\exp\left(\boldsymbol{\beta}^{\top}\mathbf{x}^{(i)}\right)}$$

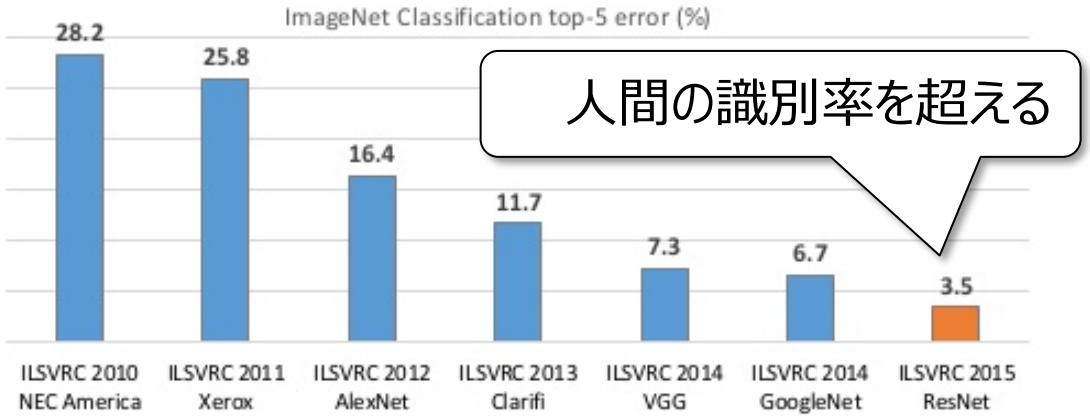
現在のパラメータでのモデルが与える確率

ニューラルネットワーク

深層学習（ディープラーニング）の出現：機械による認識の大幅な精度向上

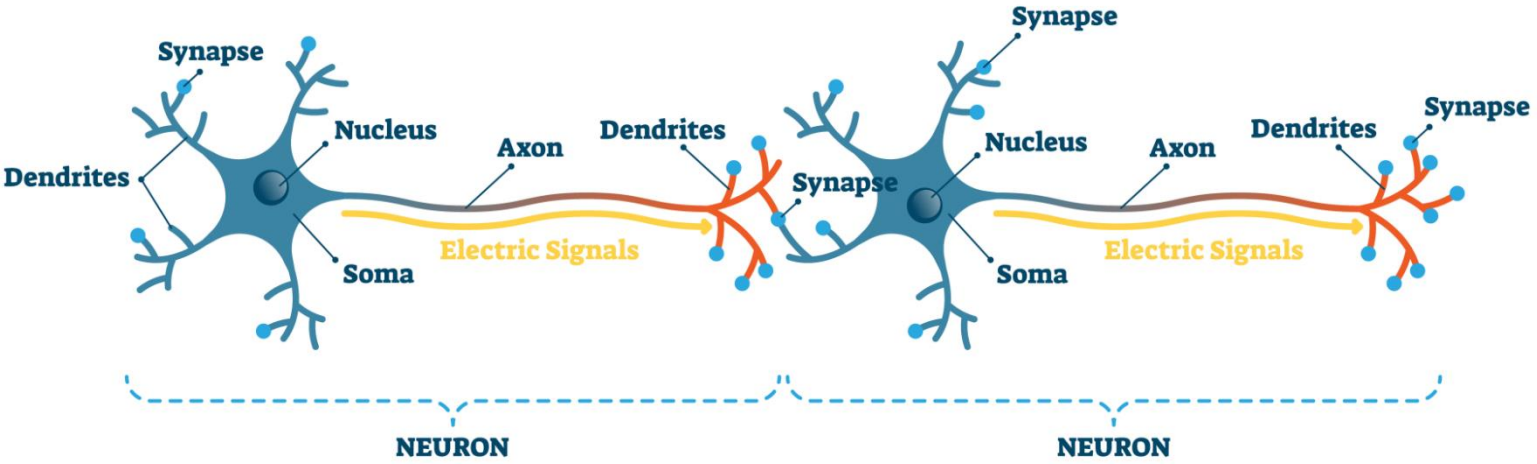
- ニューラルネットワーク：
  1980年代に盛んに研究がされていたがその後下火に
- 画像識別で10%以上の記録更新、一躍注目を浴びる → 畳み込みニューラルネットがデファクト・スタンダードに
- Big Techを筆頭に数々の企業が深層学習に大きな投資
- 研究・開発のトレンドも深層学習が中心に

Microsoftの発表資料より抜粋



（本物の）ニューラルネットワーク：多数の神経細胞が繋がった脳のモデル

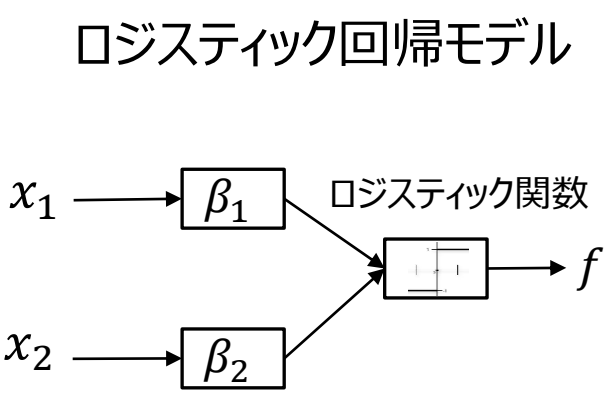
- 多数の神経細胞(neuron)がネットワーク状に繋がった脳の神経回路（を模した計算モデル）
- 静止電位にある神経細胞に他の細胞から興奮性の刺激電流が入れば電位が上昇し、ある閾値を超えると活動電位と呼ばれるピーク性の電位変化を起こす

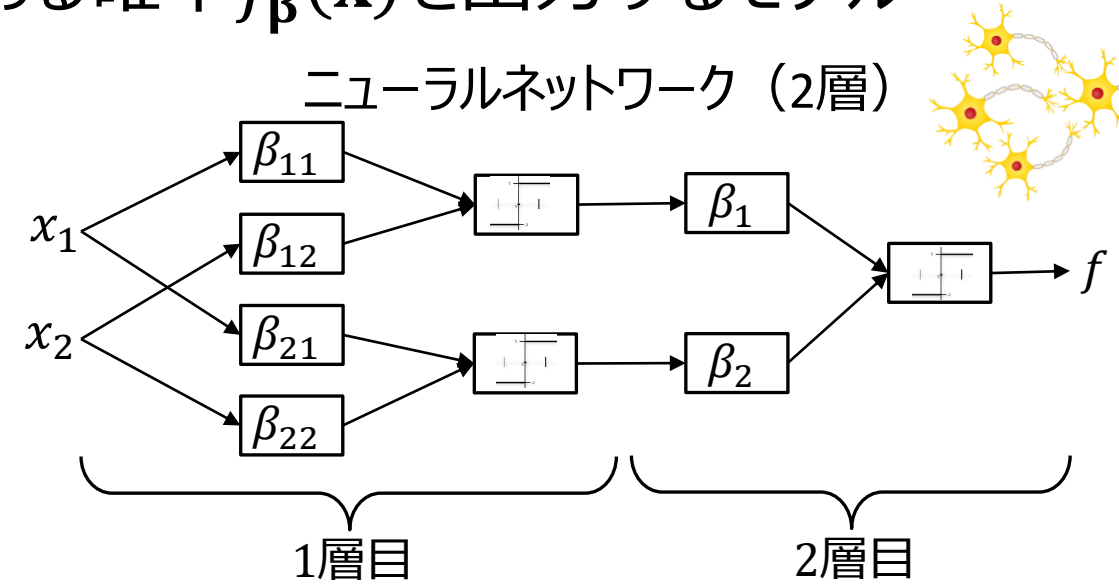


https://www.simplypsychology.org/synapse.html

ニューラルネットワーク：（ざっくりいえば）ロジスティック回帰モデルを連結したもの

- ニューラルネットワークはロジスティック回帰モデルを連結したもの
  - 複数のロジスティック回帰モデルの出力が、別のロジスティック回帰モデルの入力になる
  - ロジスティック関数（非線形）によりモデルに非線形性を導入
  - 両者ともに、$y = +1$である確率$f_{\boldsymbol{\beta}}(\mathbf{x})$を出力するモデル

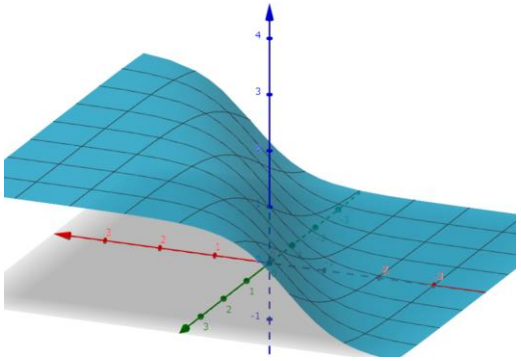


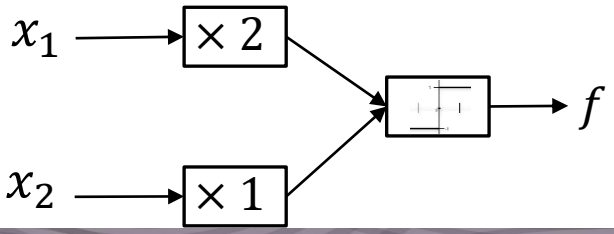


ニューラルネットワークの非線形性の例：ロジスティック回帰を2層積むと非線形分類が可能

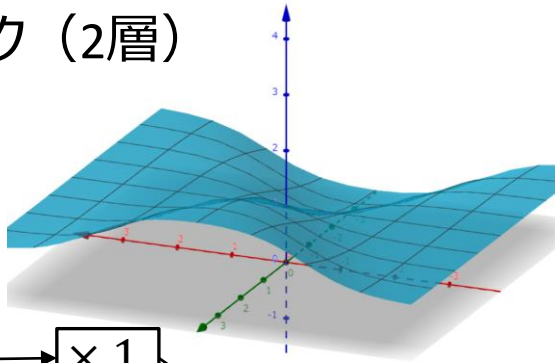
- ロジスティック回帰は1層では線形判別しかできない（AND/OR）
- 2層以上積むことで非線形の表現力を獲得（XOR）
- 任意の入出力関数を表現できる （ユニバーサル近似定理）

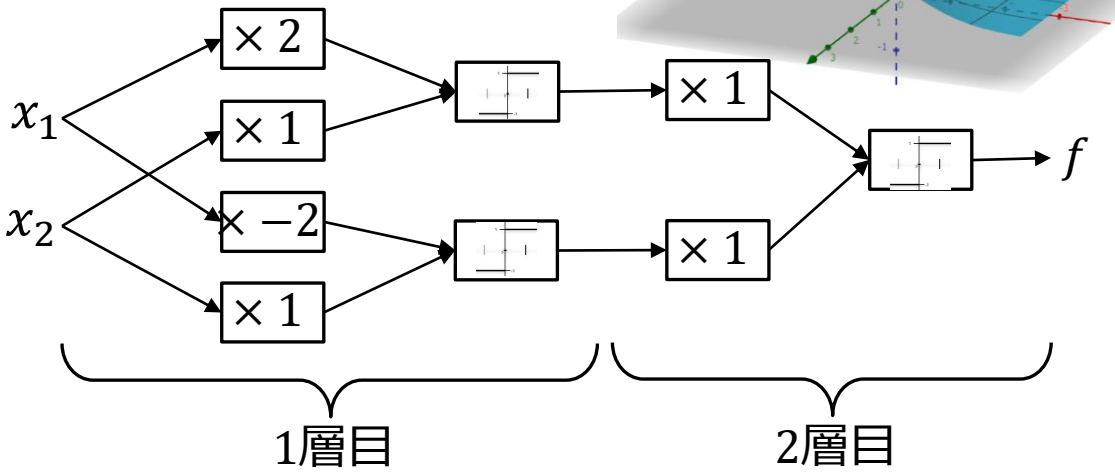
ロジスティック回帰モデル





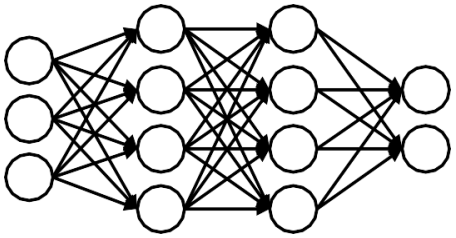
ニューラルネットワーク（2層）





深層学習：
多層化による高い表現力の獲得

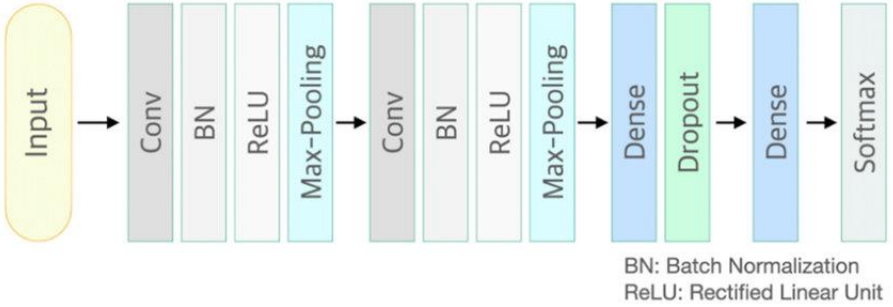
- 深層ニューラルネットワーク（DNN）：
  多数のニューロンを用いて層状に積み重ねることで、
  入力データの複雑な変換を実現したモデル
  - ニューラルネットを（かなり）多層化したもの
  - 積み重ねることで表現力が増し、複雑な判断が可能に
    - 深層学習（ディープラーニング）と呼ばれるゆえん
- コンピュータの性能向上や、方法論の進歩によって、複雑なモデルの学習が可能になってきた

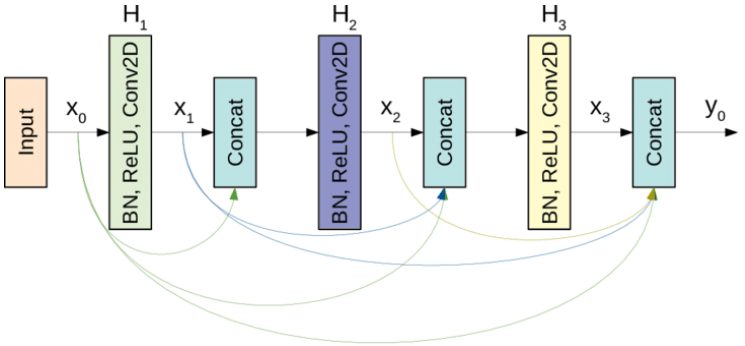


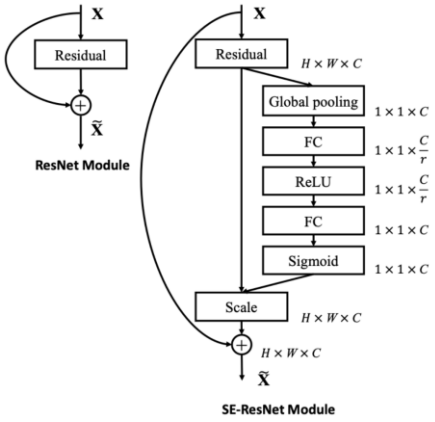
$$x \to \mathrm{B}_1 x \to g_1 = \sigma(\mathrm{B}_1 x)$$
$$g_1 \to \mathrm{B}_2 g_1 \to g_2 = \sigma(\mathrm{B}_2 g_1)$$
$$g_2 \to \mathrm{B}_3 g_2 \to y = \sigma(\mathrm{B}_3 g_2)$$

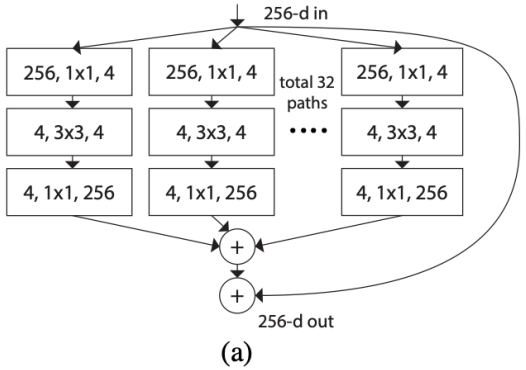
深層ニューラルネットワークの構造：
高い表現力と汎用学習法による自由度の高いモデリング

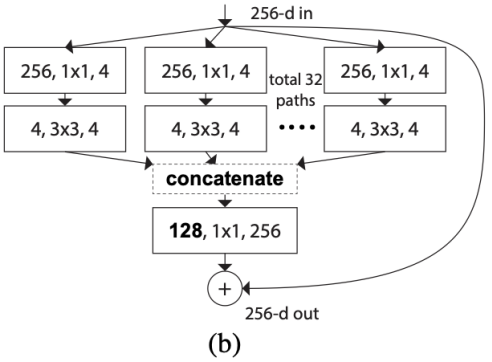
- 「自動微分＋勾配法」による統一的なパラメタ調整により層をブロックとして組み合わせ自由自在なモデリング（≒配線芸）が可能
  - 全結合層，活性化層，畳み込み層，プーリング層， バッチ正規化層，ドロップアウト層など

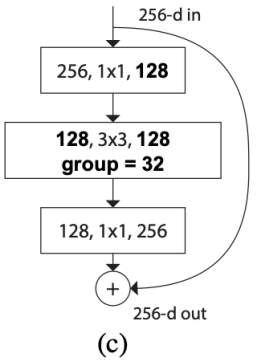












ニューラルネットワークのパラメータ推定

ニューラルネットワークのパラメータ推定：最急降下法を適用するために勾配の計算が必要

- 対数尤度関数$L(\boldsymbol{\beta})$を最大化するパラメータ$\boldsymbol{\beta}$を求める：

$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n}\left(\delta\left(y^{(i)}=1\right)\log f_{\boldsymbol{\beta}}\left(x^{(i)}\right)+\delta\left(y^{(i)}=-1\right)\log\left(1-f_{\boldsymbol{\beta}}\left(x^{(i)}\right)\right)\right)$$

  - $f_{\boldsymbol{\beta}}\left(x^{(i)}\right)$は$x^{(i)}$に対するニューラルネットの出力
    （$y^{(i)}=1$である確率）

- 勾配$\nabla L(\boldsymbol{\beta})=\dfrac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$が計算できれば最急降下法を適用できる：

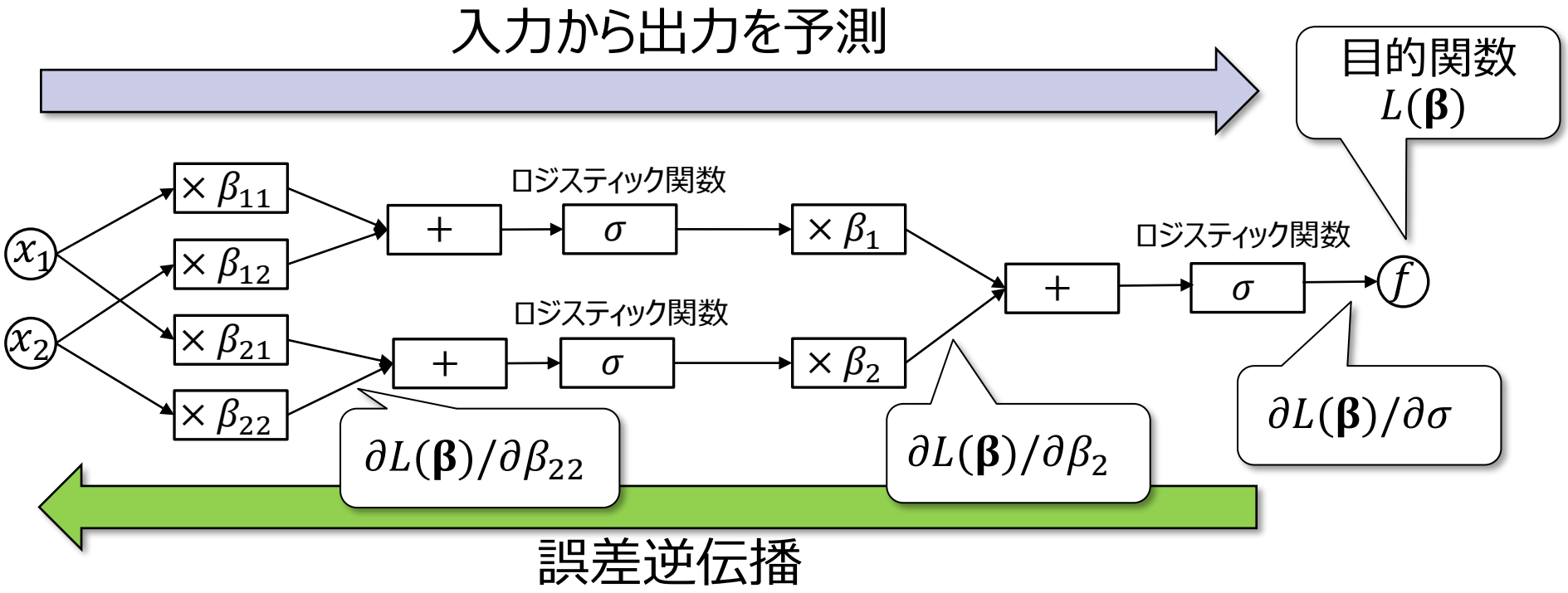
$$\boldsymbol{\beta}^{\mathrm{NEW}} \leftarrow \boldsymbol{\beta}+\eta\nabla L(\boldsymbol{\beta})$$

  - 実際は確率的最適化やミニバッチを用いることも多い

ニューラルネットワークのパラメータ推定法：

誤差逆伝播法（自動微分）による勾配法でパラメータ推定

- 対数尤度$L(\boldsymbol{\beta})$をパラメータで微分できれば勾配法で推定できる
- 誤差逆伝播法（自動微分）：層を遡って微分計算



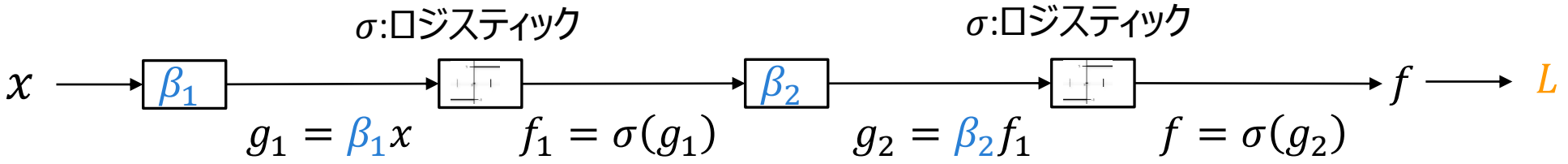
誤差逆伝播法：勾配を再帰的に効率的に計算できる

- 誤差逆伝播法：$L(\boldsymbol{\beta})$の勾配$\nabla L(\boldsymbol{\beta}) = \dfrac{\partial L(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}}$を計算する方法

- 1次元の場合の例：　注：このモデルは説明のための例であって実用的にはあまり意味はない
  - fから「後ろ向きに」遡っていく計算（微分の連鎖率）
  - $\dfrac{\partial L}{\partial \beta_2} = \dfrac{\partial L}{\partial f} \cdot \dfrac{\partial f}{\partial g_2} \cdot \dfrac{\partial g_2}{\partial \beta_2}$
  - $\dfrac{\partial L}{\partial \beta_1} = \dfrac{\partial L}{\partial f} \cdot \dfrac{\partial f}{\partial g_2} \cdot \dfrac{\partial g_2}{\partial f_1} \cdot \dfrac{\partial f_1}{\partial g_1} \cdot \dfrac{\partial g_1}{\partial \beta_1}$

共通なので層をまたいで使いまわし可能

$$L(\boldsymbol{\beta}) = -\sum_{i=1}^{n} \Big(\delta\big(y^{(i)} = 1\big) \log f\big(x^{(i)}\big) + \delta\big(y^{(i)} = -1\big) \log\Big(1 - f\big(x^{(i)}\big)\Big) \Big)$$

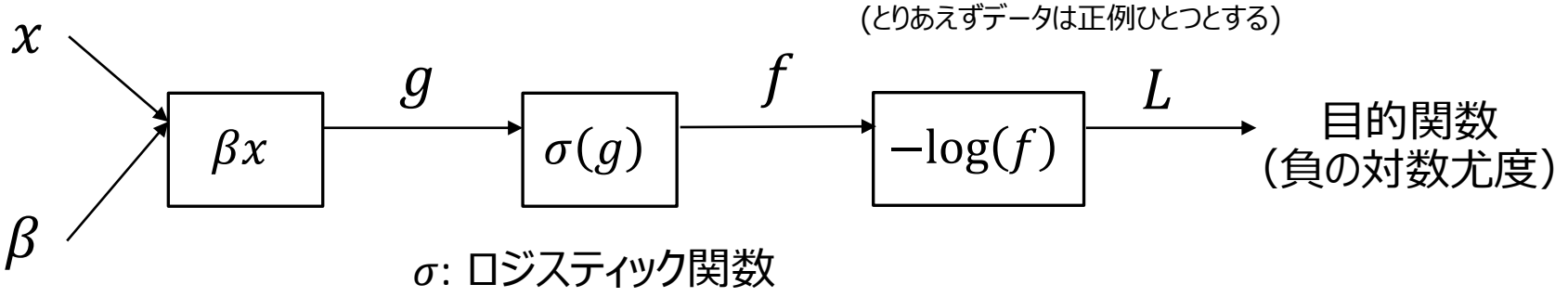


計算グラフと自動微分

計算グラフ：
ニューラルネットの入力から出力までの計算を図示

- 計算グラフ：関数の入出力の間の関係を、単純な計算ユニットをつないで表したもの
- 計算グラフをたどりながら、入力に順番に単純な操作（重み付き和やロジスティック関数の適用など）を適用していくと、出力が得られる
- ロジスティック回帰の計算グラフ：

ロジスティック回帰の出力： $f = \sigma(\beta x)$
最適化問題の目的関数：$L = -\log f$
(とりあえずデータは正例ひとつとする)

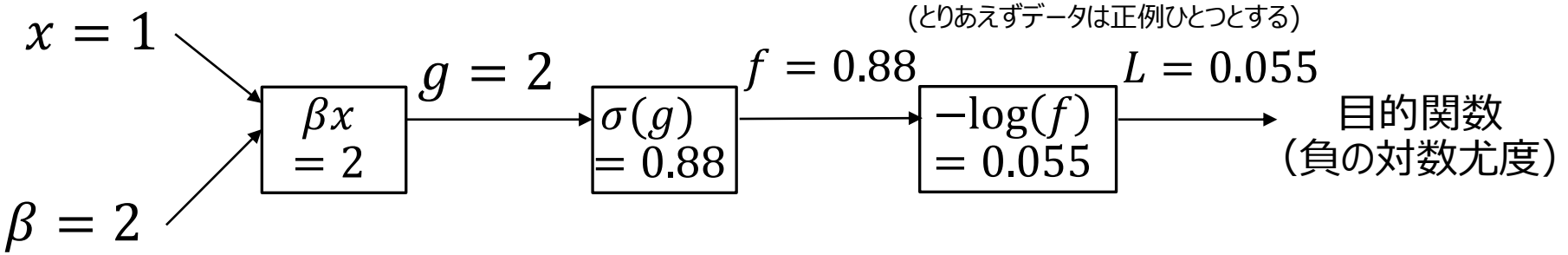


σ: ロジスティック関数

計算グラフ：
ニューラルネットの入力から出力までの計算を図示

- 計算グラフ：関数の入出力の間の関係を、単純な計算ユニットをつないで表したもの
- 前向き計算：計算グラフをたどりながら、入力に順番に単純な操作（重み付き和やロジスティック関数の適用など）を適用していくと、出力が得られる
- ロジスティック回帰の（前向き）計算グラフ：

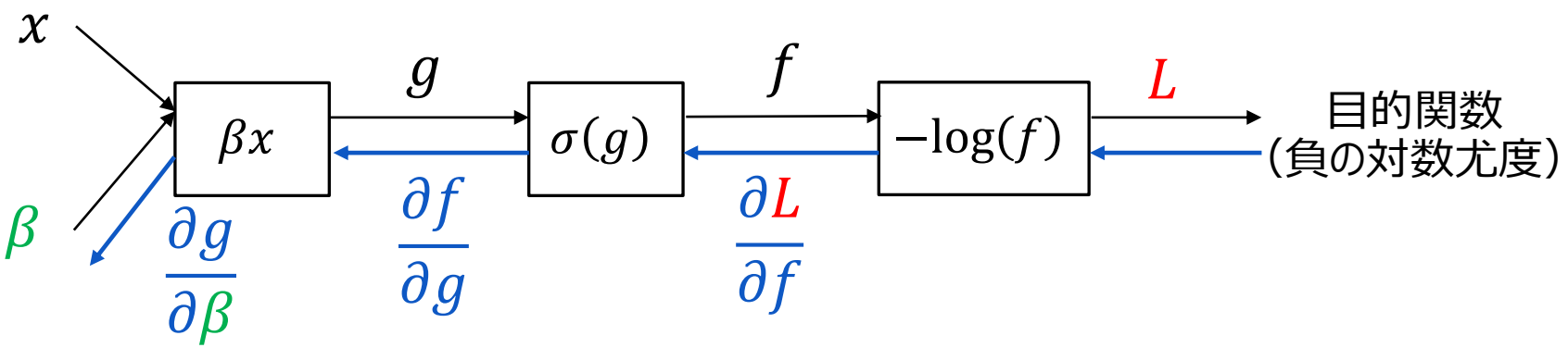
ロジスティック回帰の出力： $f = \sigma(\beta x)$
最適化問題の目的関数：$L = -\log f$
(とりあえずデータは正例ひとつとする)



σ: シグモイド関数

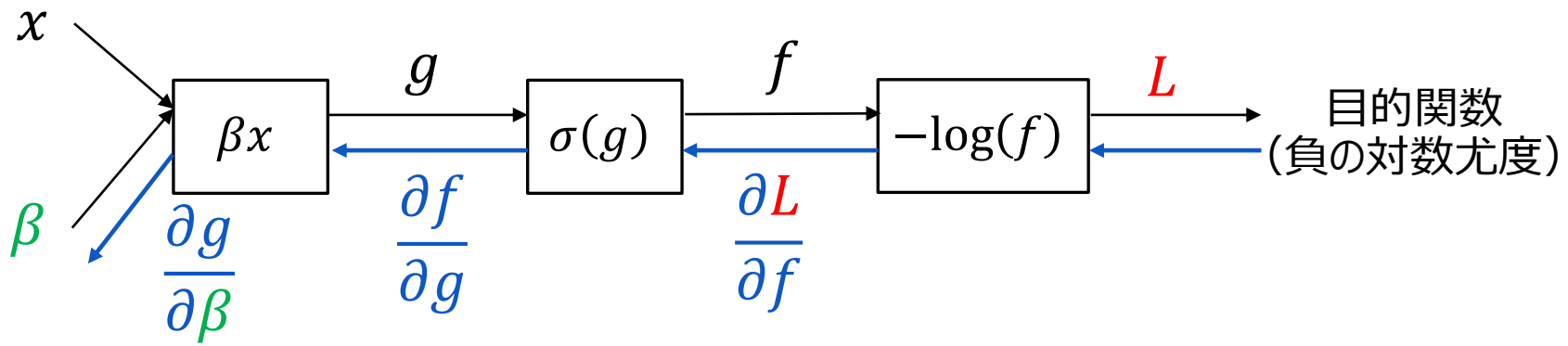
計算グラフ上での自動微分：
計算グラフを出力から逆向きにたどることで勾配計算

- 勾配計算：$\partial L/\partial \beta$を求めたい
- 計算グラフ上でLとβは遠い
- 後ろ向き計算：
  計算グラフを逆向きにたどりながら、微分を計算する
  - ロジスティック回帰の場合：$\frac{\partial L}{\partial \beta} = \frac{\partial g}{\partial \beta} \cdot \frac{\partial f}{\partial g} \cdot \frac{\partial L}{\partial f}$



自動微分のポイント：
微分可能な計算ユニット

- 自動微分：計算グラフを逆向きにたどりながら（微分の連鎖律によって）微分を計算する
- 各ユニットは、入力について微分可能である必要がある
  - $\frac{\partial f}{\partial g} = \frac{\partial \sigma(g)}{\partial g} = \sigma(g)\big(1 - \sigma(g)\big)$ （ロジスティック関数の微分）
  - $\frac{\partial g}{\partial \beta} = \frac{\partial \beta x}{\partial \beta} = x$



ニューラルネットワーク推定のポイント：
微分可能なユニットを組みあわせて自動微分にまかせる

- ニューラルネット推定法の汎用性

1. ネットワークを「計算グラフ」で記述する
   - 各ユニットはパラメータや入力について微分可能とする
2. 誤差逆伝播で自動的に勾配が計算できる
   （自動微分）

深いニューラルネットワークの推定

深層学習による微分可能プログラミング：
微分可能ユニットによるネットワーク設計と自動微分

1. 各層の出力が入力とパラメータについて微分可能な（簡単な形で微分が書ける）計算ユニットを選ぶ
2. ユニットを結合してネットワークを設計
3. あとは自動微分（誤差逆伝播）によって自動的にパラメータ調整されるのに任せる
   - 学習は少量のデータ（ミニバッチ）に対して、勾配法の1ステップを適用（確率的勾配法）

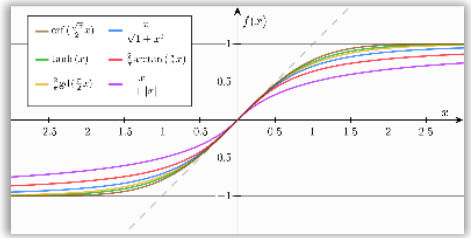
- 問題へのアプローチの変化：
  「解きやすい問題設定とアルゴリズムの開発」から →
  「微分可能ユニットの設計とネットワーク設計」に移行

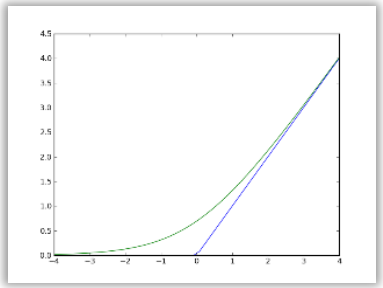
微分消失問題：
深いネットワークの学習における課題

- 微分消失問題：深いネットワークでは誤差逆伝播の過程で微分がどんどん弱くなる
- 自動微分では、各ユニットの微分の積で、勾配を計算
  - ロジスティック回帰の場合： $\frac{\partial L}{\partial w} = \frac{\partial g}{\partial w} \cdot \frac{\partial f}{\partial g} \cdot \frac{\partial L}{\partial f}$
- 深いネットワークではこの積の項数が多くなる
- ひとつひとつの微分値が小さいと全体としてどんどん値が小さくなるため、入出力の間の距離が長いほど、勾配が小さくなり、学習が進まない

正規化線形ユニット、スキップ接続：微分消失問題への対処

- 活性化関数の工夫による対処：
  正規化線形ユニットReLU（ランプ関数） $\max(0, x)$





シグモイド関数　　ReLU

  - ReLUの微分値は1（あるいは0）であるため、微分が消えにくい

- スキップ接続の利用

$$z_{l+1} = z_l + f(z_l)$$

- 深層ニューラルネットワークの学習に有利

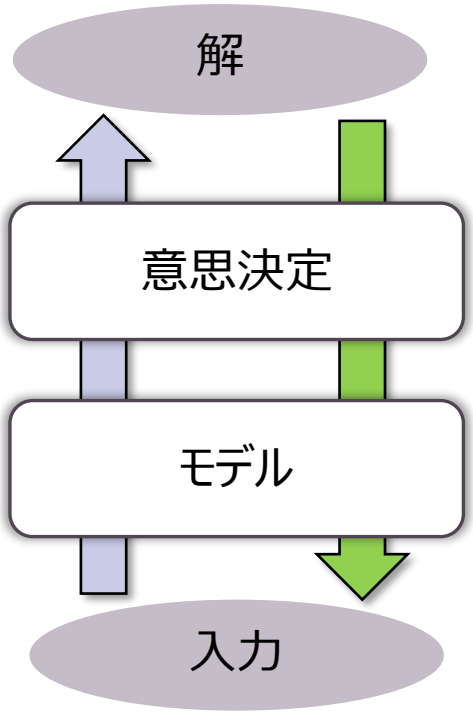
多数のパラメータをもつ深層モデルの学習：構造に制限を入れることで事前知識を導入

- 通常の機械学習モデルでは、人間のもつ事前知識を特徴量（独立変数）定義の形で入れる
- 深層学習では、知識をネットワーク構造に対する制限という形で入れる（「帰納バイアス」の一種）
- 例：CNN（畳み込みニューラルネットワーク）
  - 画像識別では「局所的なパターン」を集めて認識を行う
    - 画像識別ではパターンの絶対的出現位置はあまり関係ない
  - 「局所的なパターン」は画像の位置によらず共通で使いまわす

深層学習の世界観：
モデル化と意思決定を一気通貫で実現したい

- 通常は2ステップの：
  - 推定（対象のモデル化）
  - 最適化（モデルに基づく意思決定）

  を一気通貫で実現する世界観



- 学習可能（≒微分可能）な意思決定
  - 単純な更新操作の繰り返しで書けるような意思決定
  - 解が、所望の状態に近づくようにパラメータを更新
    - 解の誤差を定義して、これを最小化する

まとめ：
ニューラルネットワーク

1. ニューラルネットワーク：（乱暴に言えば）ロジスティック回帰を積んだもの
2. ニューラルネットワークの学習：勾配法（SGD、ミニバッチ）で学習
3. 計算グラフと自動微分（誤差逆伝播）：単純な微分可能ユニットで構成された計算グラフ上で、勾配を効率的に、体系的に計算可能
4. 深層学習モデルの学習：ReLUユニットによる微分消失問題への対処