データサイエンス『実践コース』

数理工学PBL

Day 1-2：深層学習

東北大学　小池　敦

機械学習

- コンピュータが
  データからそこに潜むパターンを
  学習することで
  未知のデータに対して判断を行うモデルを
  獲得すること

昔の機械

・決まった処理のみを実行できる

従来のコンピュータ

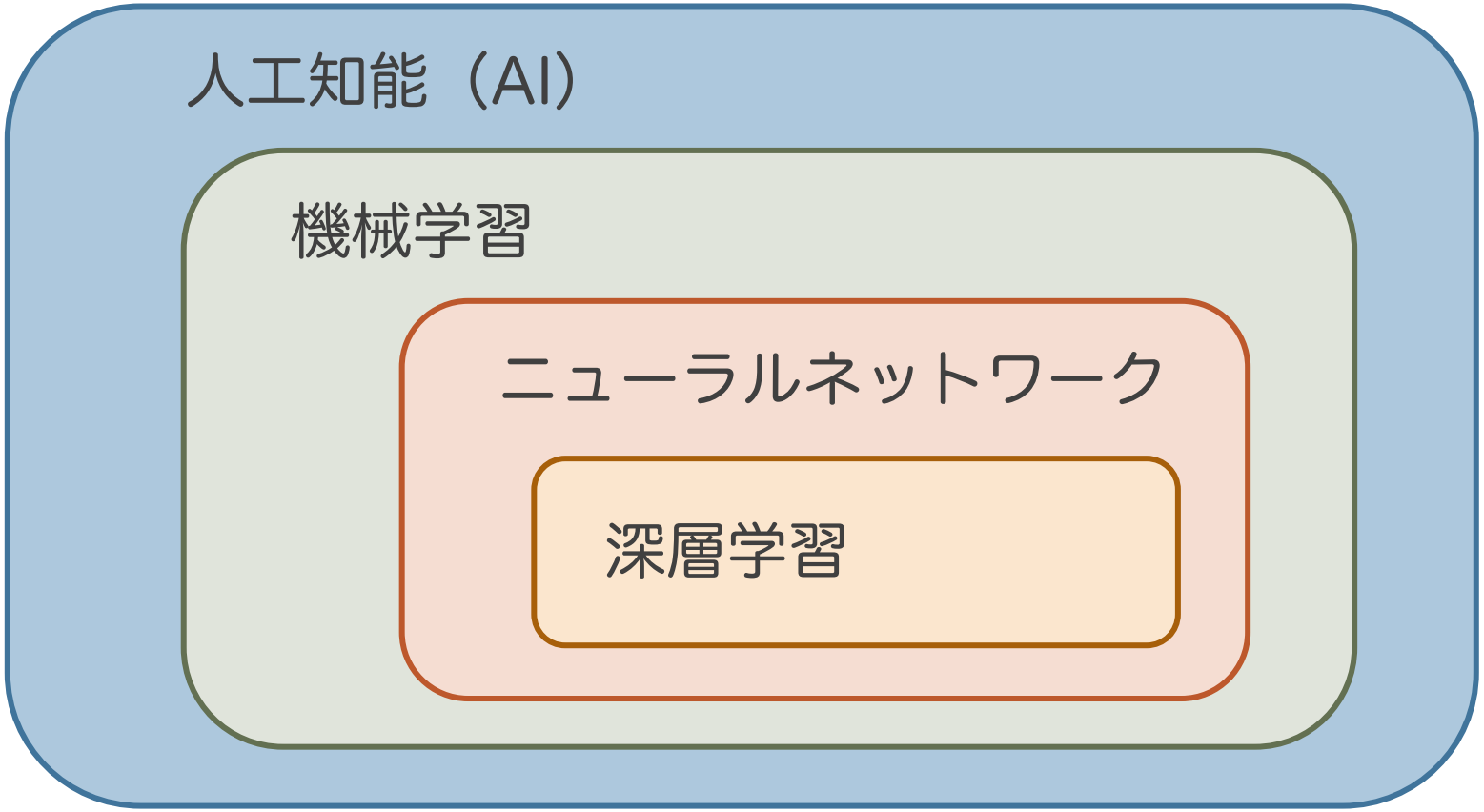
・プログラムを与えることで任意の処理を実行可能

1940年代

機械学習

・明示的にプログラムを与えなくても実行可能

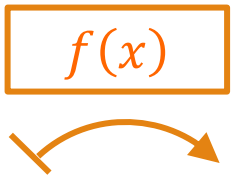
現在


人工知能（AI）
機械学習
ニューラルネットワーク
深層学習

機械学習の主なタイプ

- 教師あり学習（今回はこれ）
  - 「入力と出力」の集合から入出力間の関係性を学ぶ
- 教師なし学習
  - 入力の集合から、その特徴を見つけて何らかの処理を行う
- 強化学習
  - 明示的にデータを与えず、ソフトウェアが自ら試行錯誤することで適切な行動を学ぶ

教師あり学習



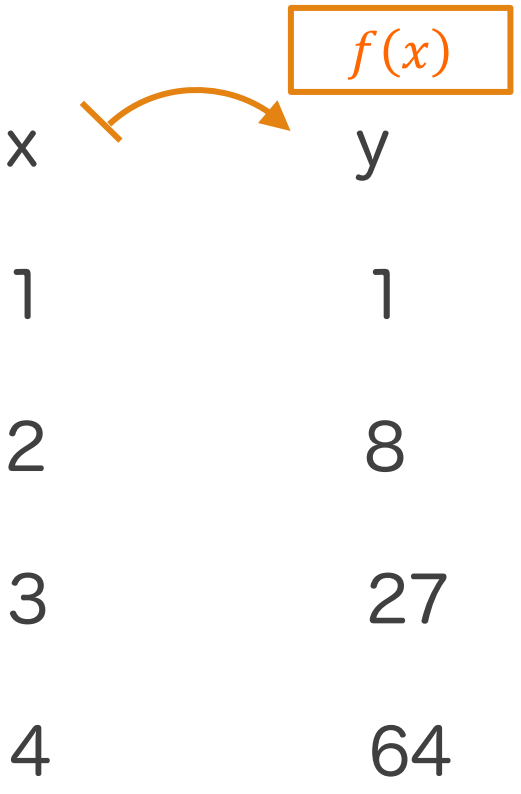
これを求めたい

x	y
1	1
2	8
3	27
4	64

考えてみよう

教師あり学習

- 教師あり学習におけるアプローチ



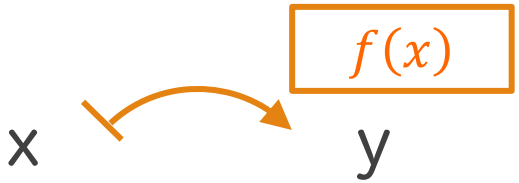
これを求めたい

$f(x)$の例：
$f(x) = x^3$

複数の$f(x)$が考えられるが
それっぽいものを選択する
（それっぽさについては
各種研究あり）

教師あり学習

- 一度関数が推定できれば、新しい入力に対しても出力を予測することができる ⇒ 汎化という



5　　？　　⇒ $f(x) = x^3$なので、125

教師あり学習の2つのフェーズ

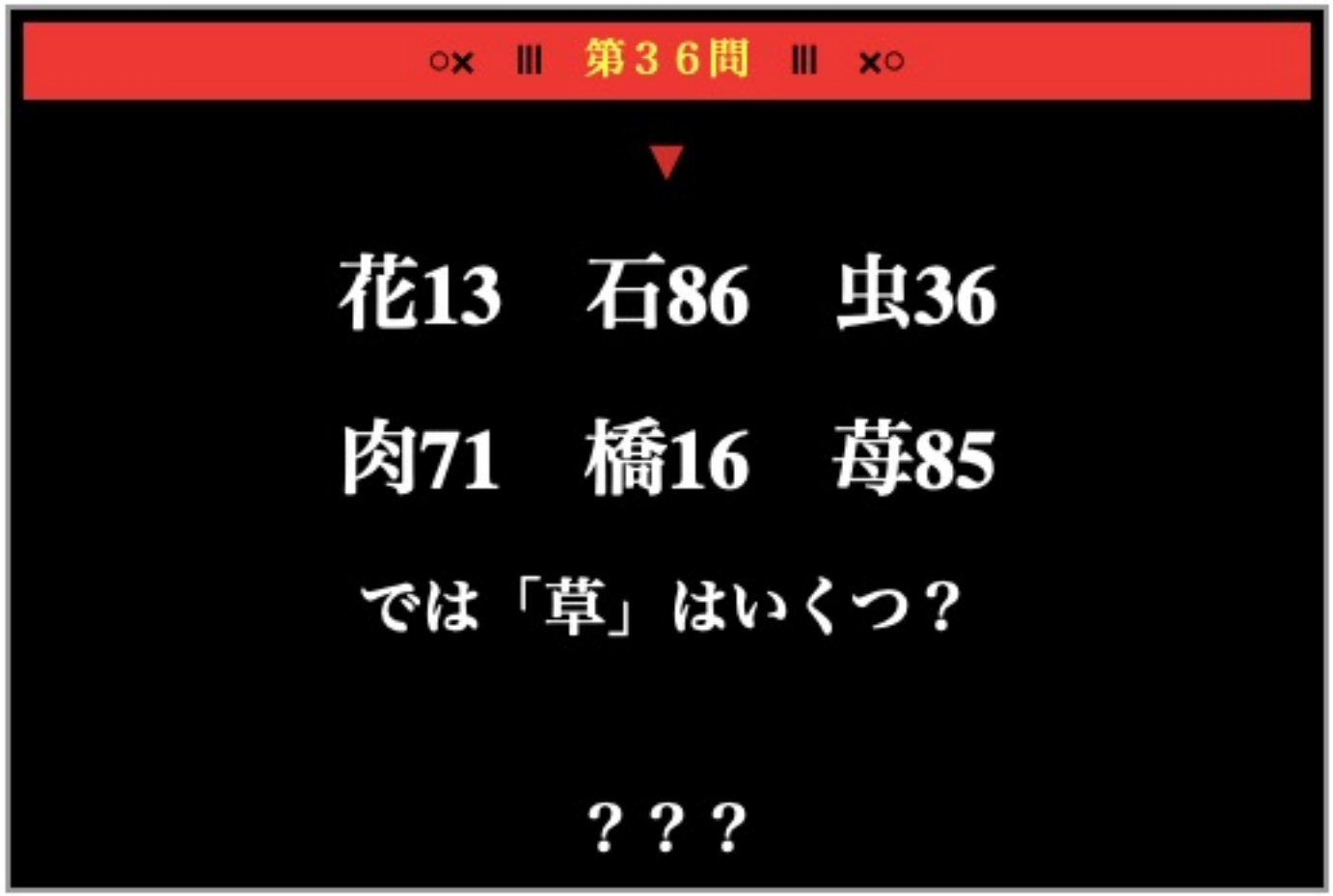
学習フェーズ

・「入力と出力」の集合からその関係を学習する

推論フェーズ

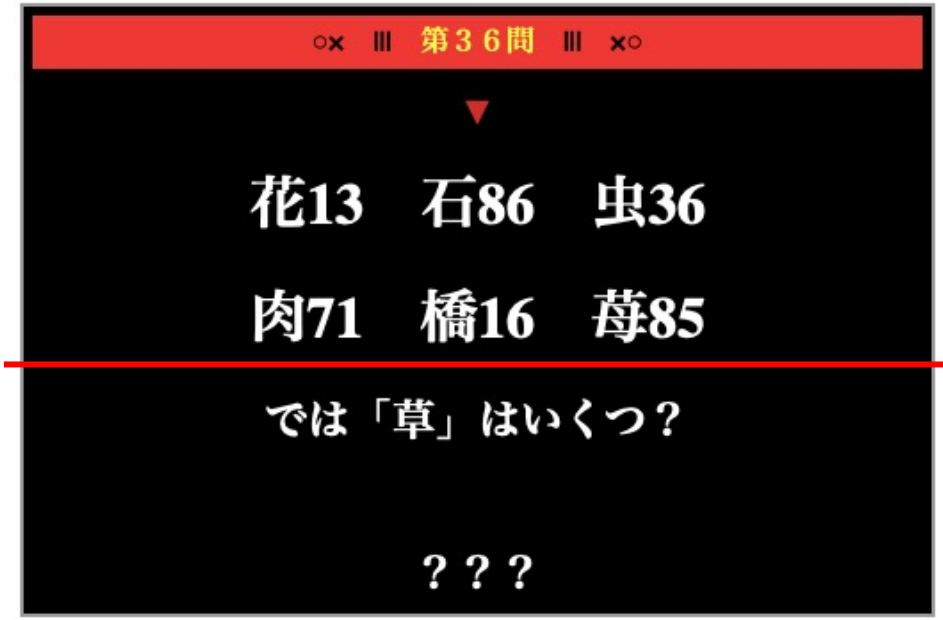
・未知の入力に対する出力を予測する

なぞなぞの例



おアソビ探偵団 なぞなぞ
https://www.tnc.ne.jp/oasobi/oasobi03/106nazonazo/09.html

なぞなぞの例



学習フェーズ：

$f(x) = 100 - [x$の語呂合わせ$]$
例: $f($花$) = 100 - 87 = 13$

推論フェーズ：

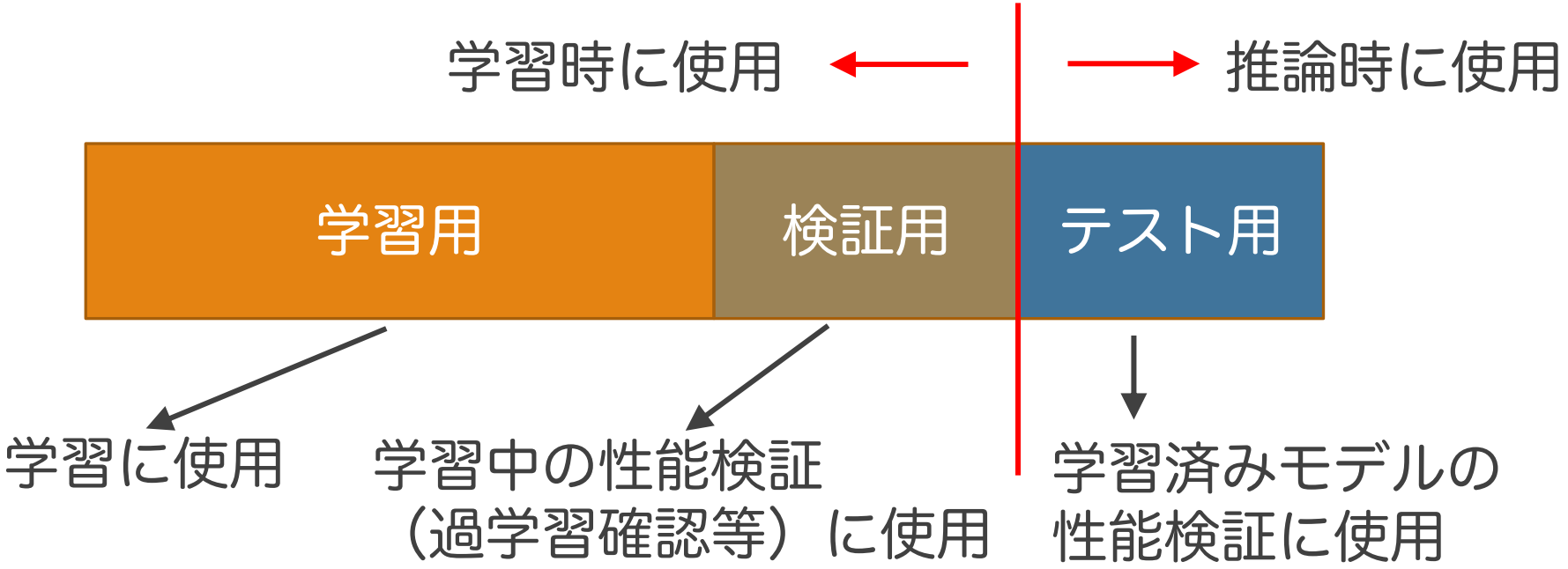
$f($草$) = 100 - 93 = 7$

今後は基本的に、学習フェーズに着目する

おアソビ探偵団 なぞなぞ
https://www.tnc.ne.jp/oasobi/oasobi03/106nazonazo/09.html

学習に使うデータ

- データを学習用とテスト用に分割する
- さらに学習用から検証用を分割することもある
- 各サンプルは入力とラベル（正解）からなる



教師あり学習

- もっと複雑な関数も学習できる

$f(x)$

画像 x → y ラベル

ねこ

いぬ

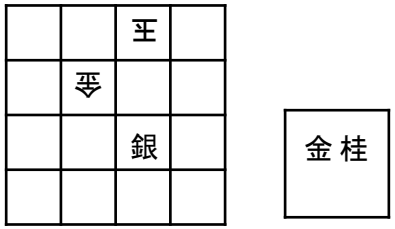
画像認識

教師あり学習

- もっと複雑な関数も学習できる

$f(x)$

将棋盤 x → y　最適手



３三桂

ゲームAI

深層学習

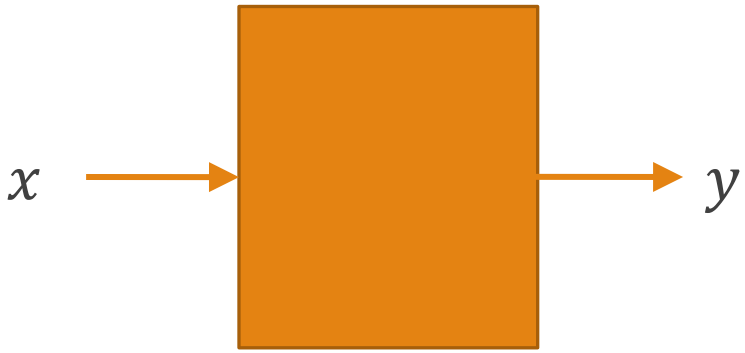
ニューラルネットワーク

- 脳の神経細胞（ニューロン）のネットワークを参考に作られた数理モデル
- 2010年ごろまではそれほど高性能ではなかった
- 2012年の画像認識コンテストILSVRCでディープラーニングを使用したチームが圧勝し，その頃から広く活用されるようになった

ニューラルネットワークの基礎

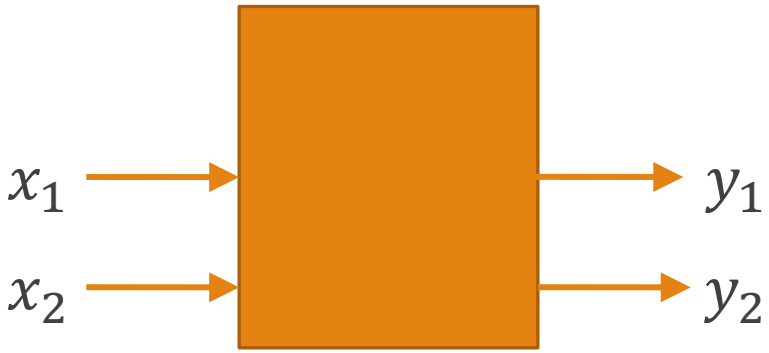
- 1ネットワークで入力xと出力yの関係を表す
  - 基本的には$y = f(x)$のような関数になる

$y = f(x)$



1入力1出力

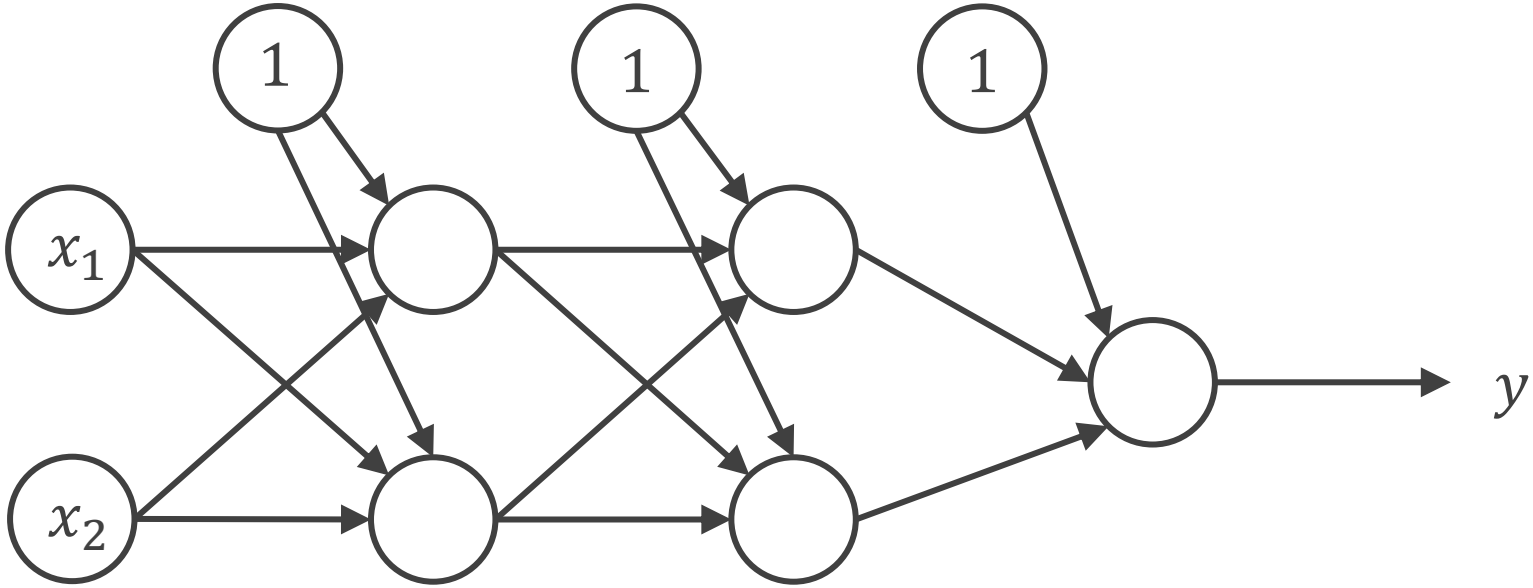
$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = f\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}\right)$$



2入力2出力
（入出力とも2次元ベクトル）

ニューラルネットワークの基礎

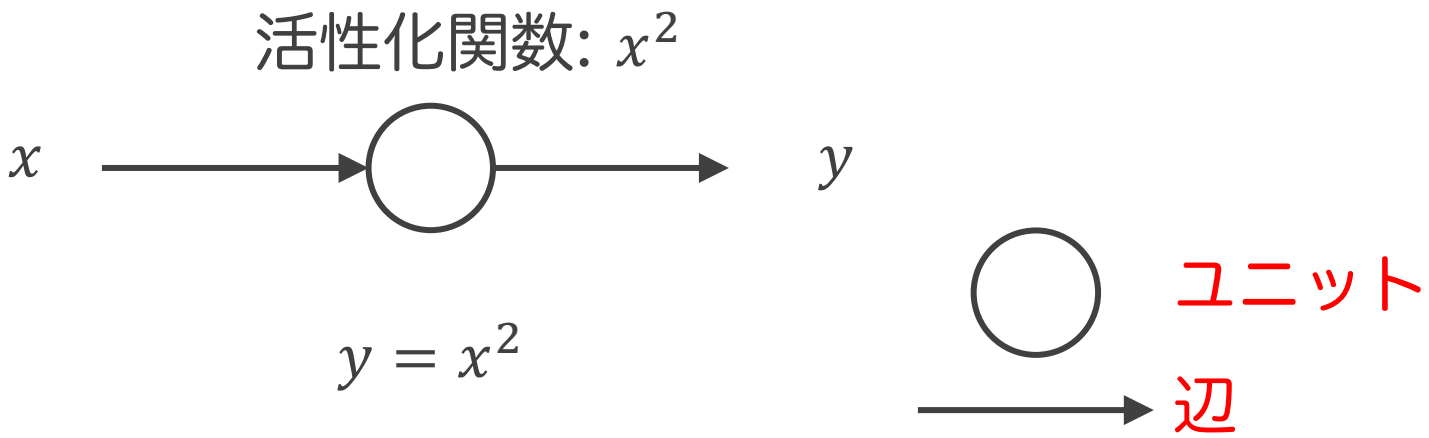
具体例



この意味を説明する

ニューラルネットワークの基礎

ユニットは関数を表す（活性化関数と呼ぶ）
活性化関数は1引数，1出力



※ 実際にはこんな活性化関数は使わない（具体例は後述）

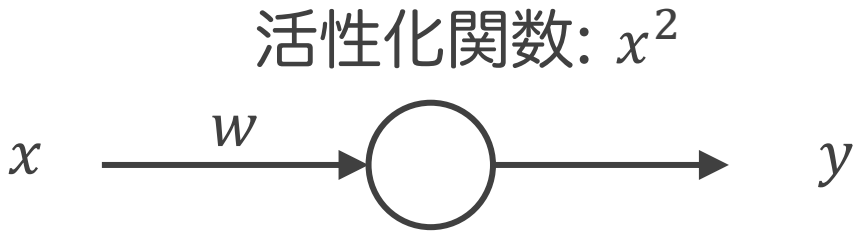
ニューラルネットワークの基礎

頂点への入力がない時は定数を返す関数
（ネットワーク内部に追加される定数はバイアス項と呼ばれる）

$$y = 1$$

ニューラルネットワークの基礎

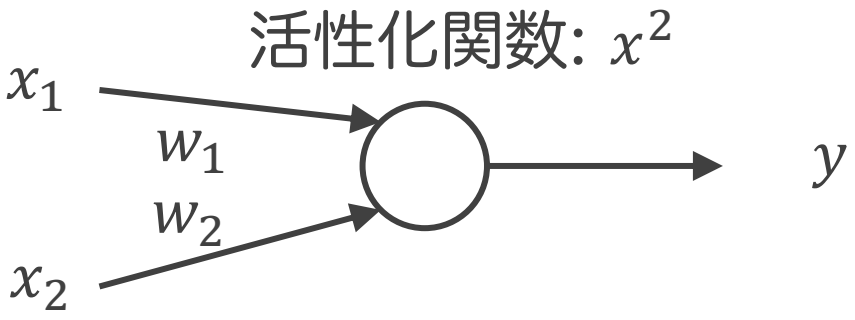
辺には重みを設定できる
重みの値で定数倍される



$$y = (wx)^2$$

ニューラルネットワークの基礎

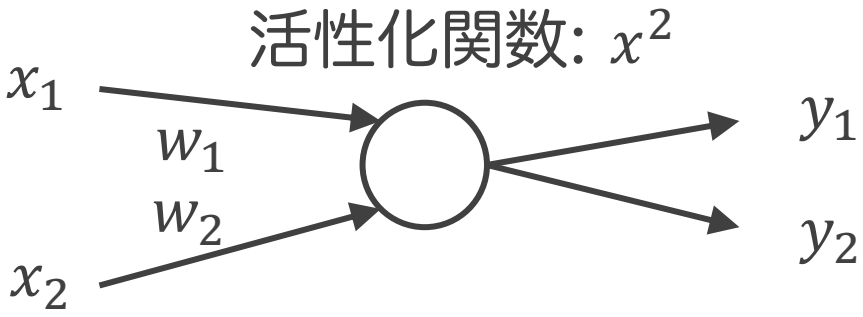
頂点への入力が複数ある時，それらは加算される



$$y = (w_1 x_1 + w_2 x_2)^2$$

ニューラルネットワークの基礎

頂点からの出力が複数ある時，それらは同じ値になる



$$y_1 = (w_1 x_1 + w_2 x_2)^2$$

$$y_2 = (w_1 x_1 + w_2 x_2)^2$$

ニューラルネットワークの基礎

活性化関数

- 活性化関数には正規化線形関数がよく使われる
  ReLU: $y = \max(0, x)$

ニューラルネットワークの基礎

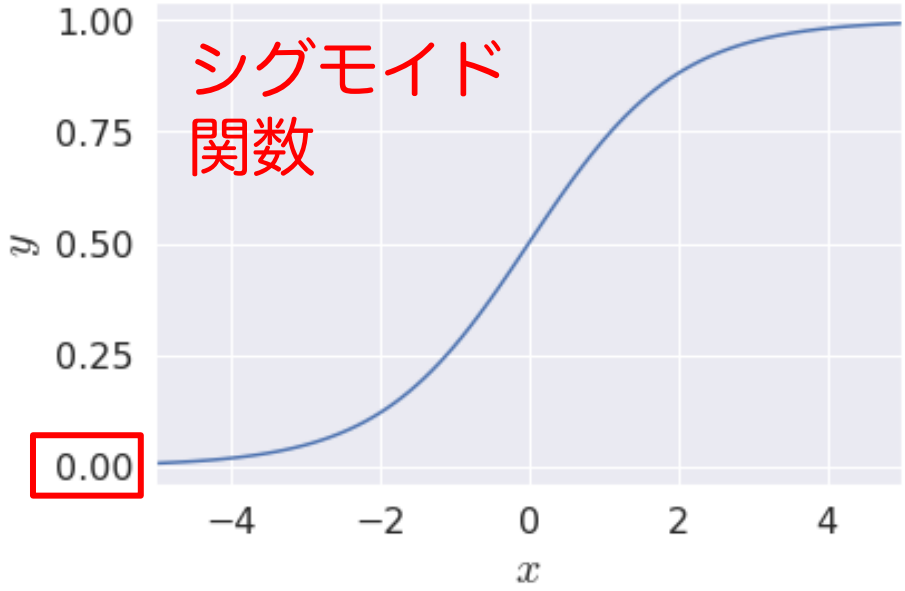
その他の活性化関数

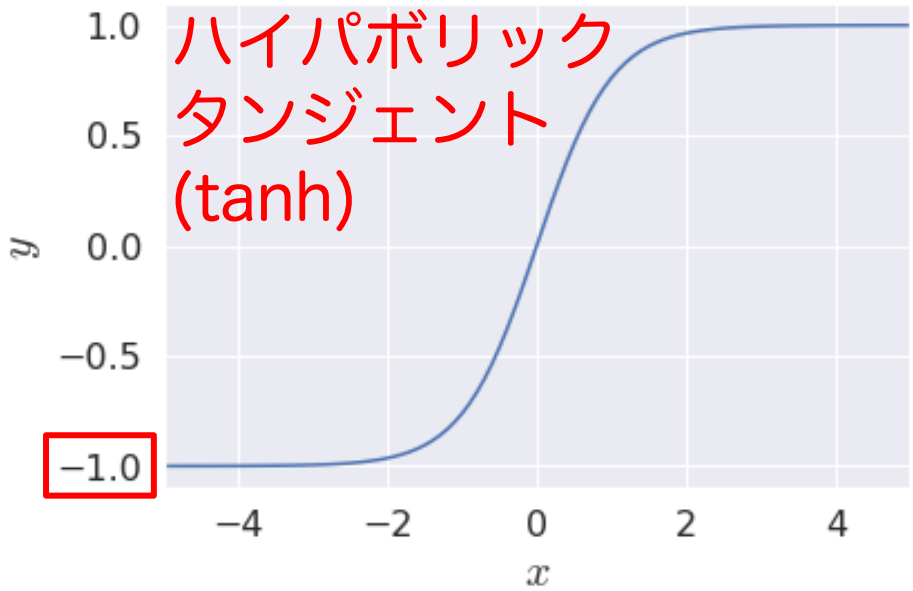
- シグモイド関数（sigmoid）

$$y = 1/(1 + e^{-x})$$

- ハイパボリックタンジェント（tanh）

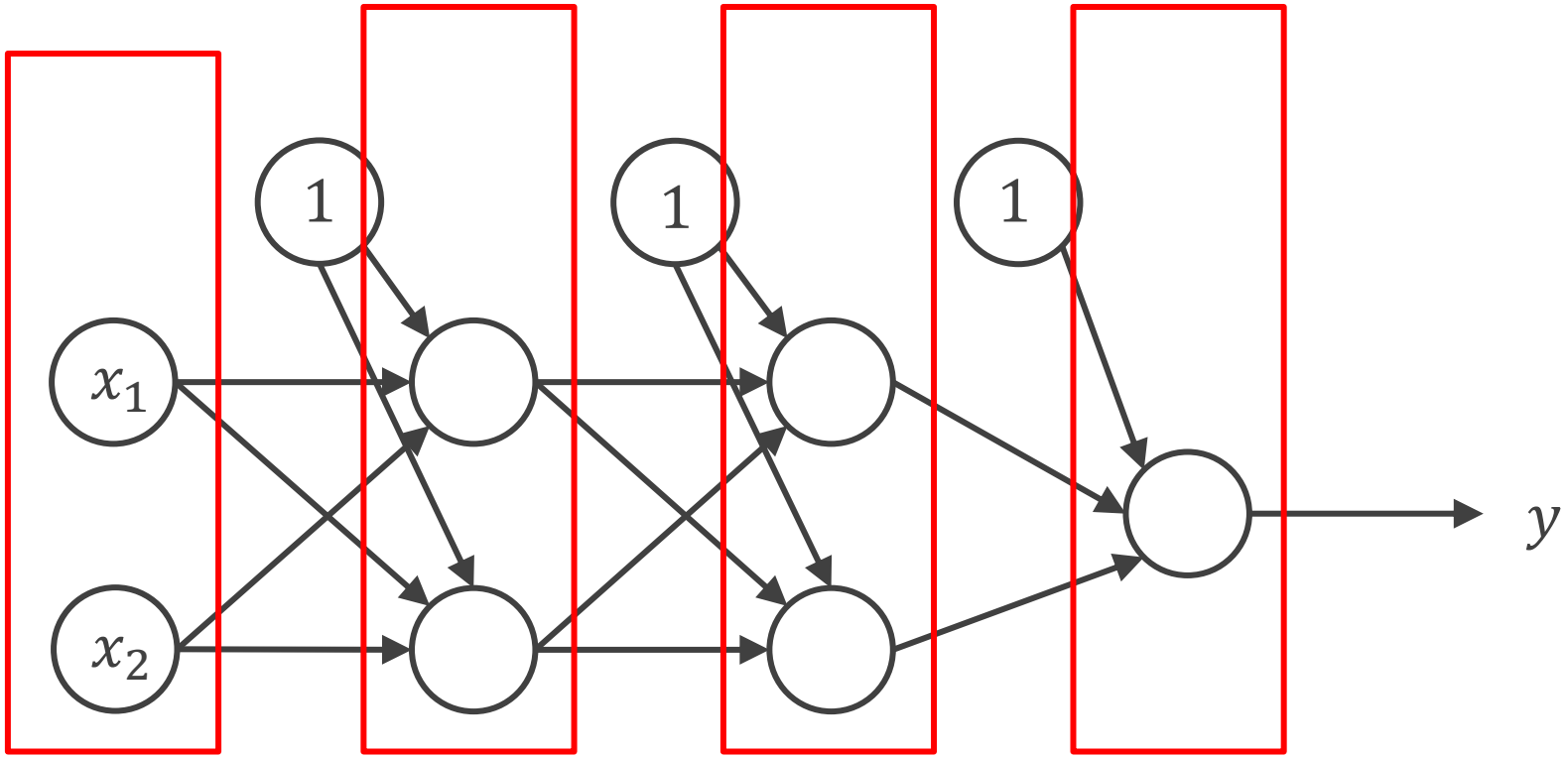
$$y = (e^{x} - e^{-x})/(e^{x} + e^{-x})$$





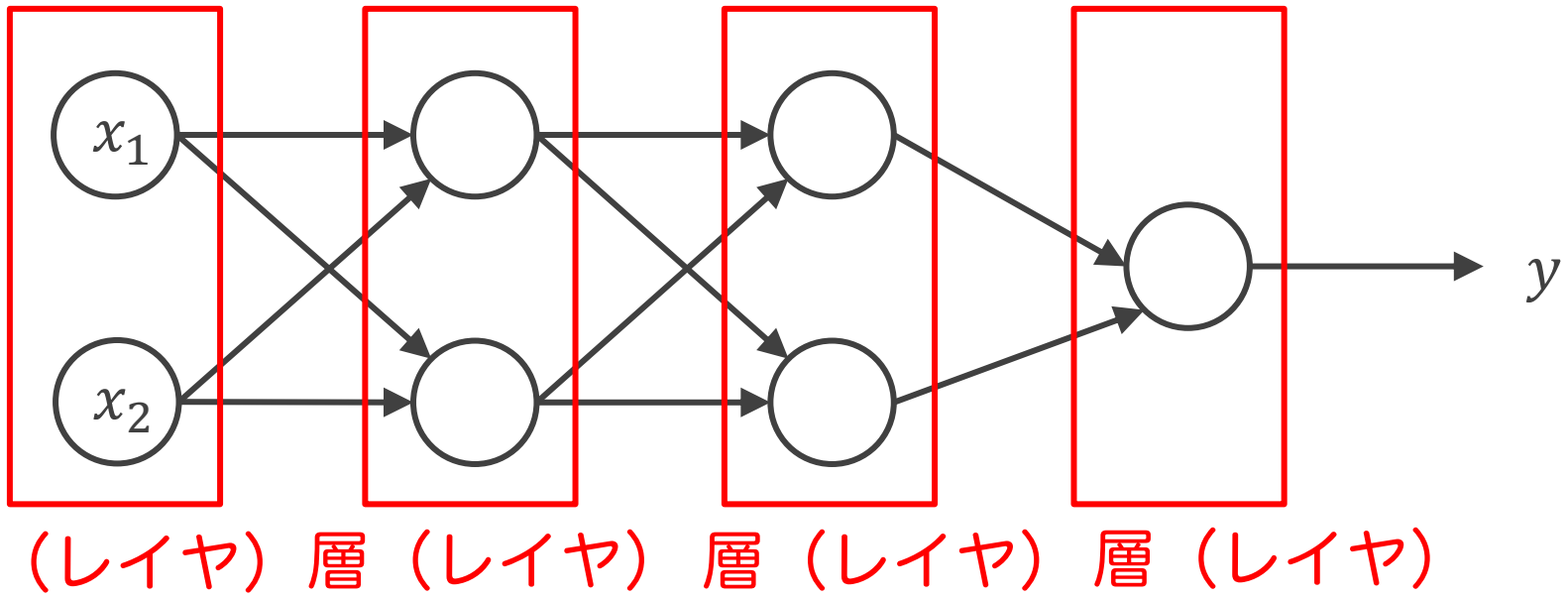
ニューラルネットワークの基礎

ユニットは一列に並べて管理される（同一の活性化関数）⇒ 層

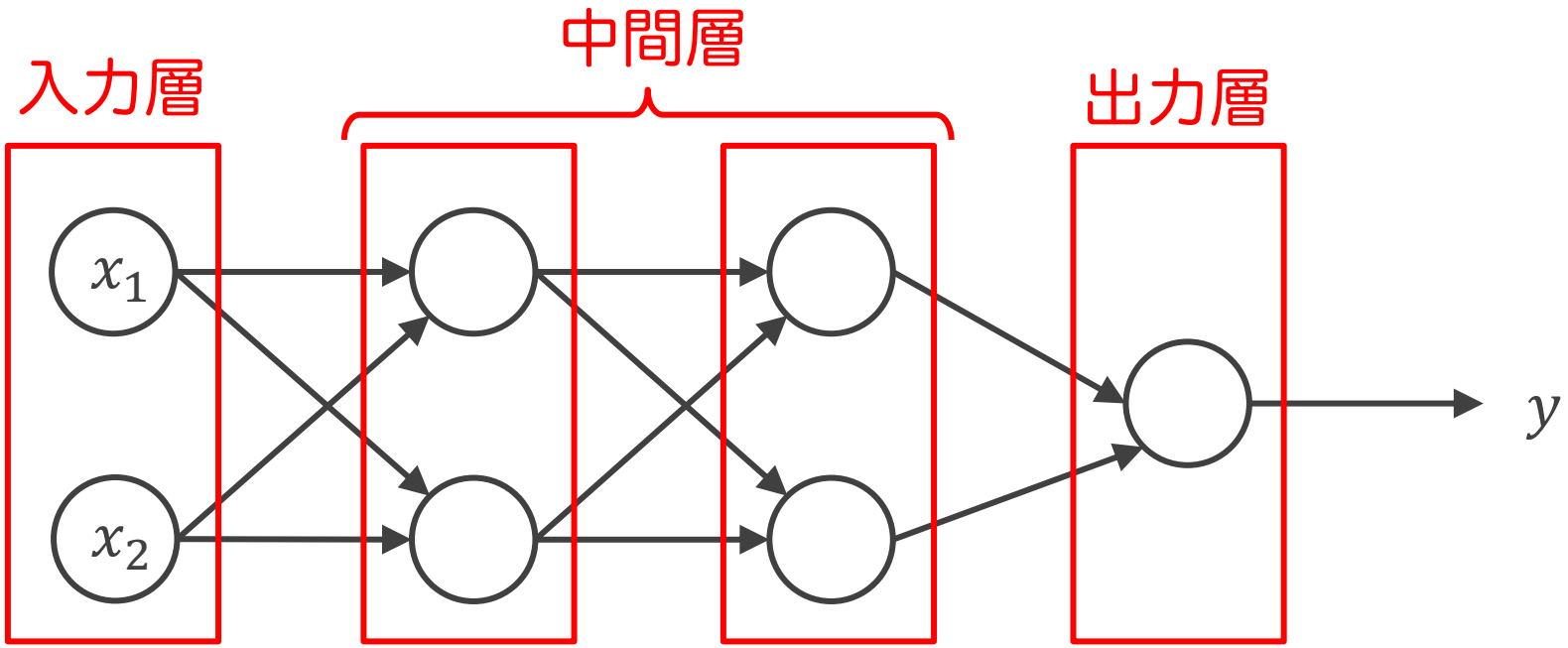


ニューラルネットワークの基礎

バイナリ項はしばしば省略して図示される



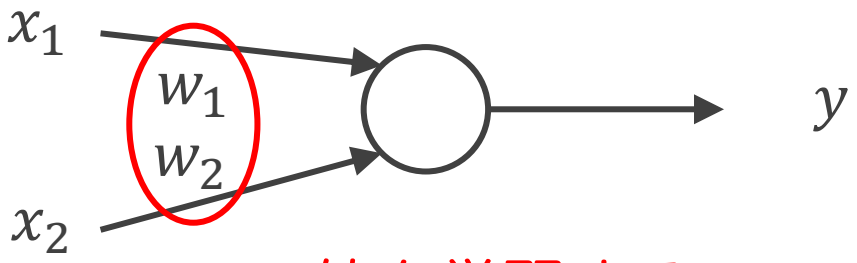
深層学習とは？



中間層の数が多いニューラルネット ⇒ 深層学習

ニューラルネットワークの学習

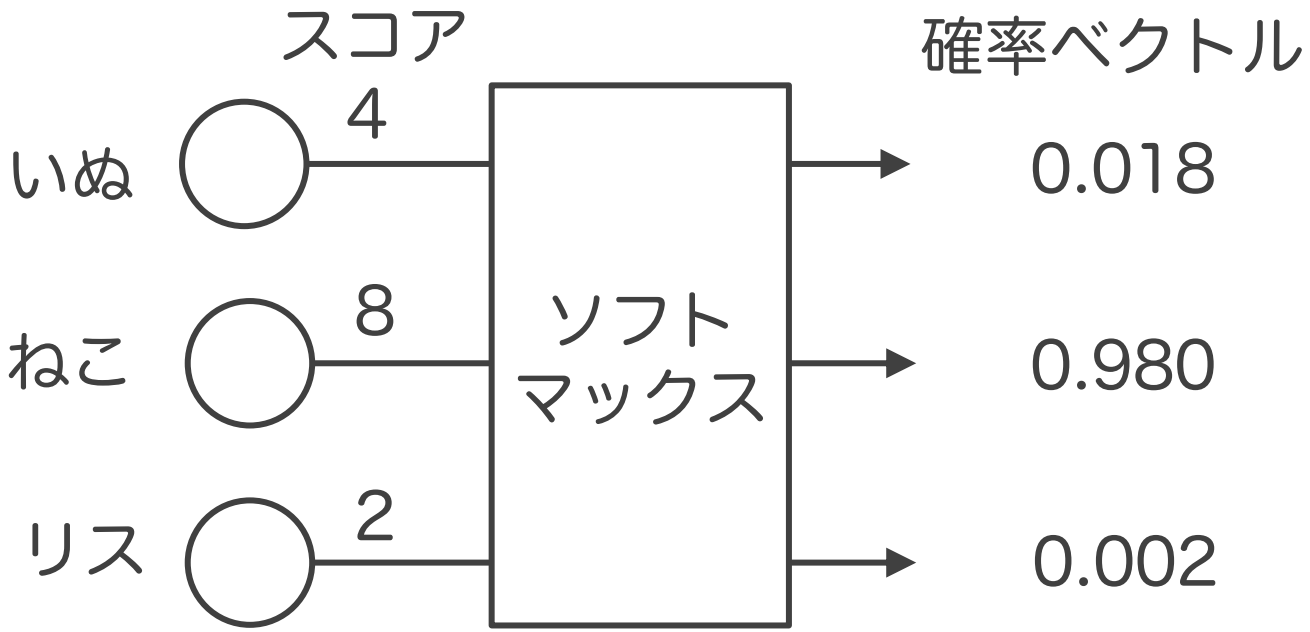
- 辺の重みを変えることでコスト関数を最小化する
- 活性化関数の部分は変えない
- 誤差最小を目指して繰り返し重みを変化させる
  - 1ターンのことをエポック(epoch)と呼ぶ



ソフトマックス関数

分類問題では，多くの場合，確率ベクトルが出力される.

出力層の活性化関数をソフトマックス関数にすることで
スコアから確率ベクトルに変換できる.



ソフトマックス関数の計算

スコア$\boldsymbol{s}$ ⟶ e^{s} ⟶ ソフトマックス $\mathrm{softmax}(\boldsymbol{s})$

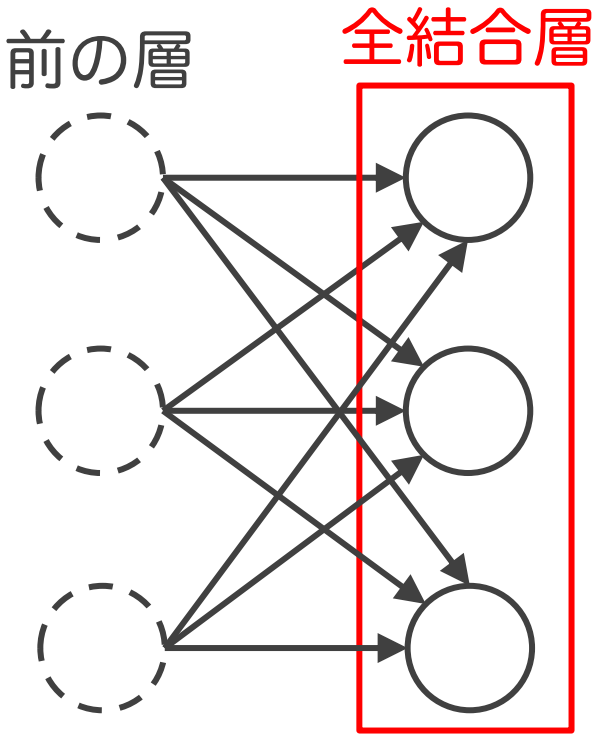
いぬ　$s_1 = 4$　$e^{s_1} = 54.6$　$p_1 = \dfrac{e^{s_1}}{e^{s_1} + e^{s_2} + e^{s_3}} = 0.018$

ねこ　$s_2 = 8$　$e^{s_2} = 2981.0$　$p_2 = \dfrac{e^{s_2}}{e^{s_1} + e^{s_2} + e^{s_3}} = 0.980$

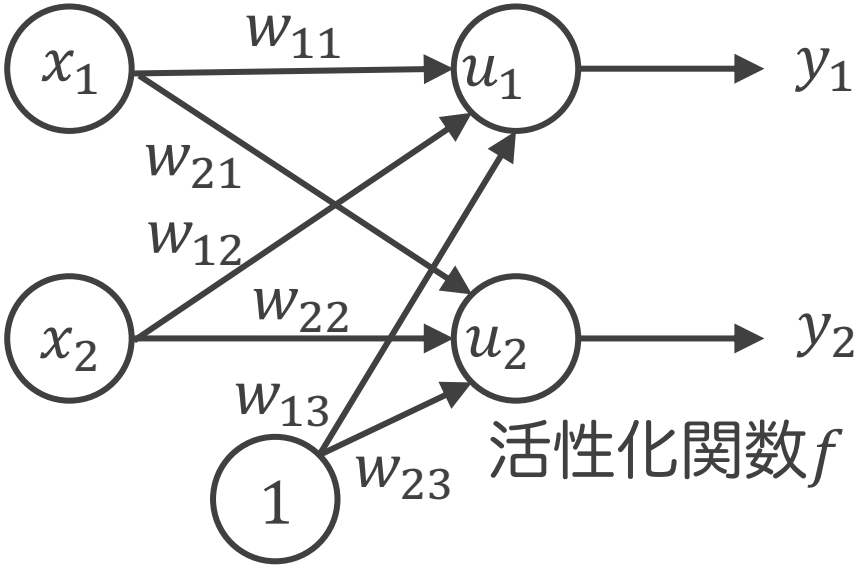
リス　$s_3 = 2$　$e^{s_3} = 7.4$　$p_3 = \dfrac{e^{s_3}}{e^{s_1} + e^{s_2} + e^{s_3}} = 0.002$

全結合層（Dense）

- 各ユニットが前の層の全ユニットと結合



全結合層の行列表現



$$[u_1 \quad u_2] = [x_1 \quad x_2 \quad 1] \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix}$$

$f([u_1 \quad u_2]) = [f(u_1) \quad f(u_2)]$ と定義すると

$$[y_1 \quad y_2] = f\left([x_1 \quad x_2 \quad 1] \begin{bmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{bmatrix} \right)$$

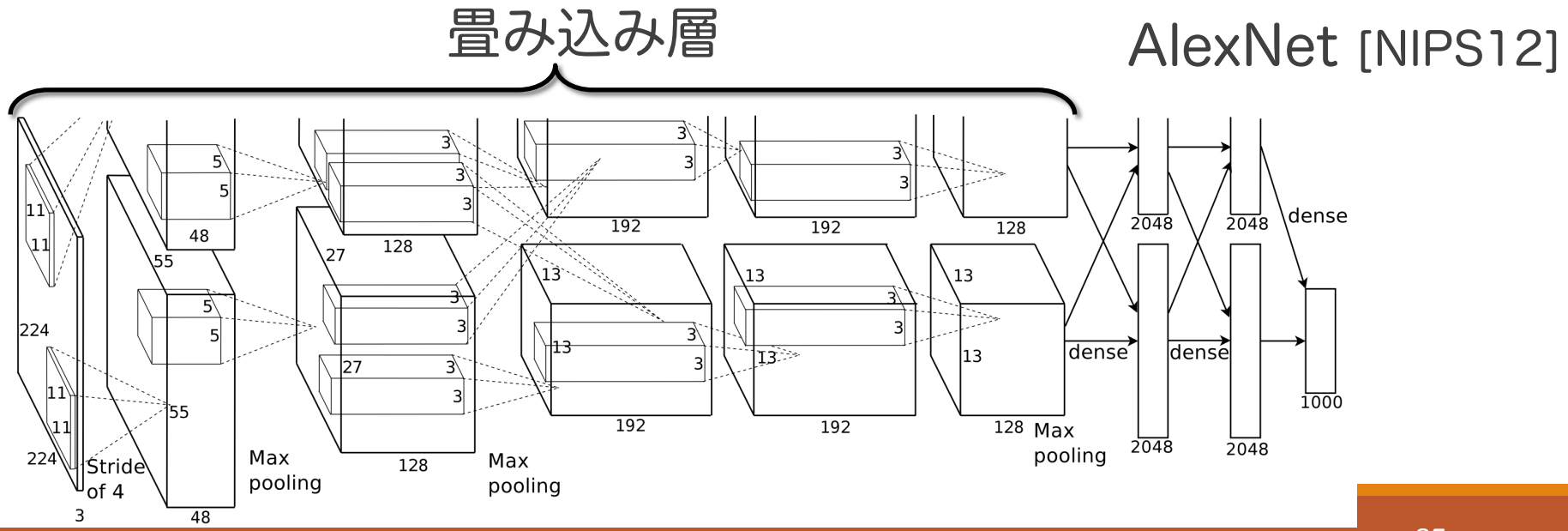
$[u_1 \quad u_2]$ は活性化関数 f への入力値

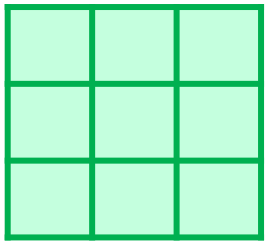
ニューラルネットワークは
行列積と活性化関数の繰り返し
として記述できる

画像処理のための ニューラルネットワーク （畳み込みニューラルネットワーク）

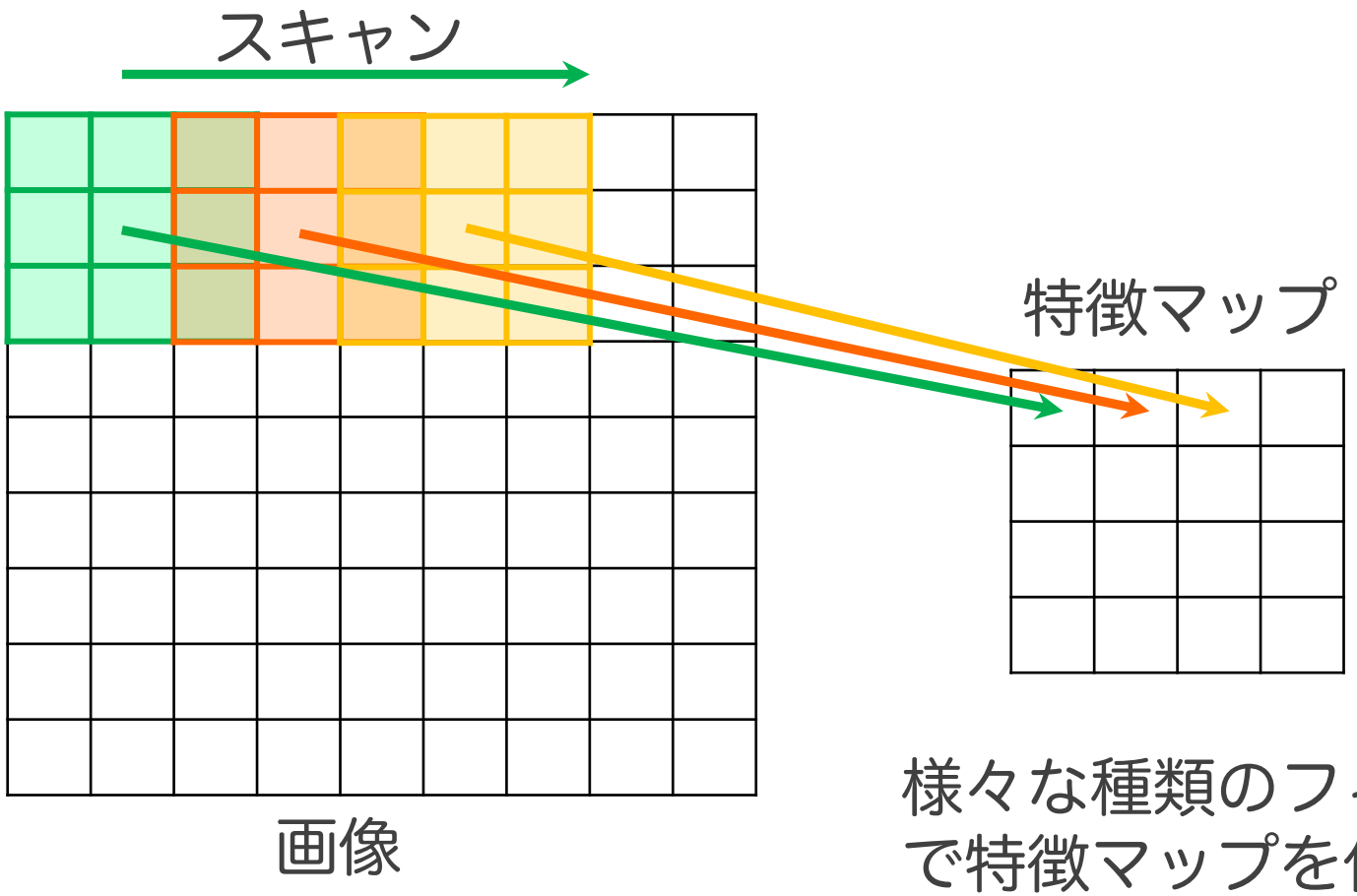
畳み込みニューラルネットワーク（CNN）

- 畳み込み層を備えたニューラルネットワーク
- 畳み込み層
  - 画像をスキャンして，画像のパターンを検出する





フィルタ（特定の画像パターンの検出用）



様々な種類のフィルタ
で特徴マップを作る

フィルタ詳細

- フィルタにより得られる値

画像の画素値　フィルタ

この処理により画像のパターンを抽出できることを具体例で説明する

縦じまの検出

縦じま画像

1	0	1	0
1	0	1	0
1	0	1	0
1	0	1	0

1	-1	1	-1
1	-1	1	-1
1	-1	1	-1
1	-1	1	-1

縦じま検出用
フィルタ

→ 8

パターンが検出
されると絶対値が
大きな値が得られる

横じま画像

1	1	1	1
0	0	0	0
1	1	1	1
0	0	0	0

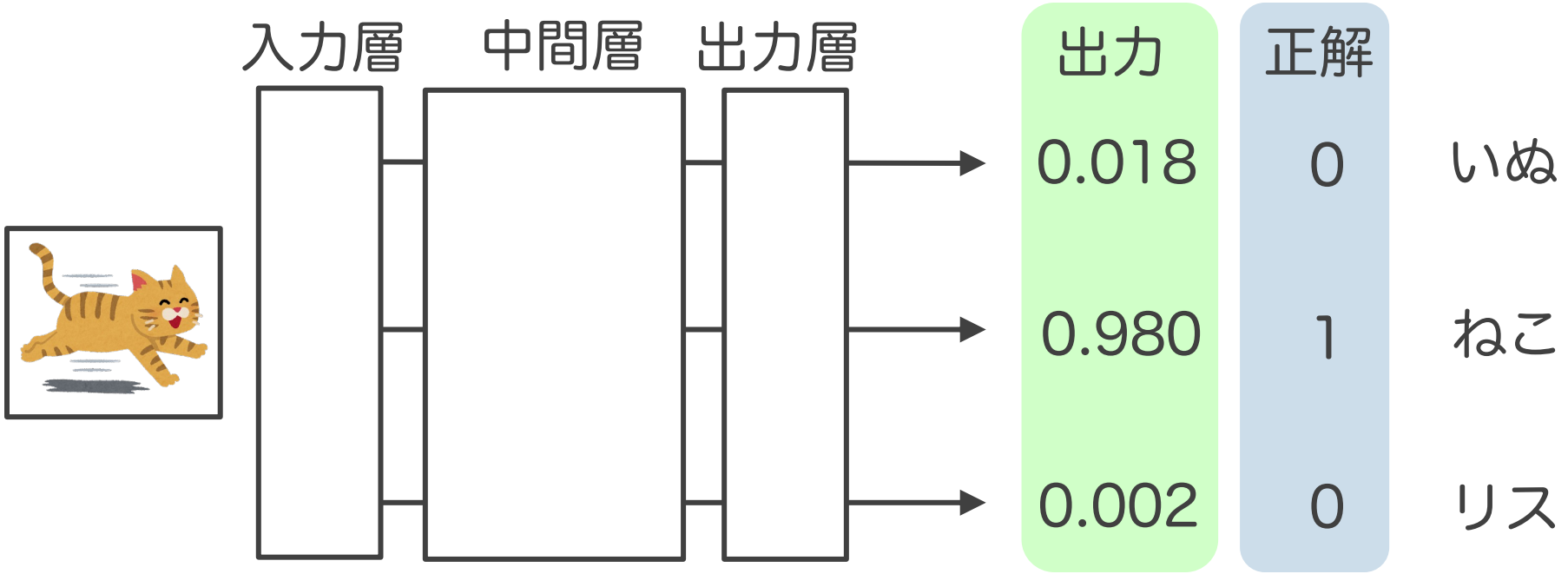
1	-1	1	-1
1	-1	1	-1
1	-1	1	-1
1	-1	1	-1

→ 0

ネットワークの学習

ニューラルネットワークの学習

- ネットワークが所望の出力をするように繰り返しネットワークを調整する

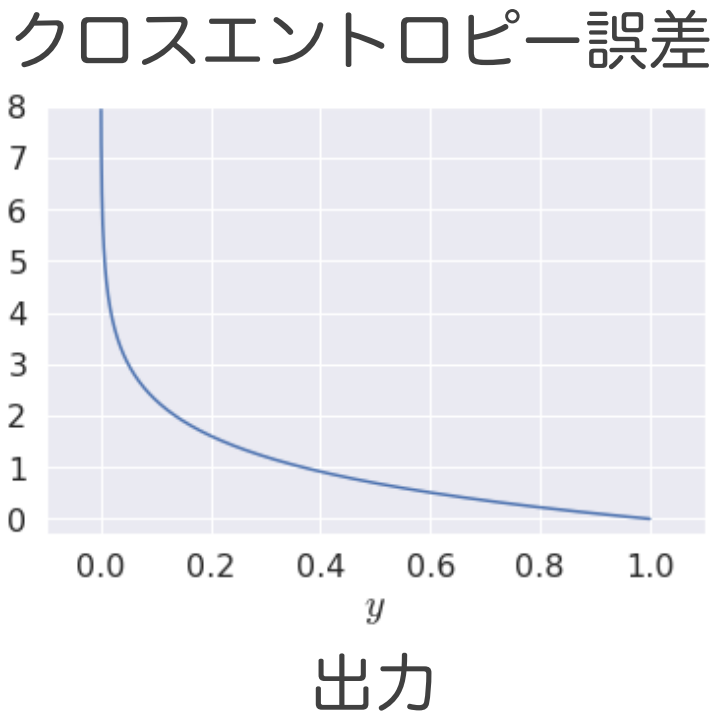


ニューラルネットワークの学習

- クロスエントロピー誤差
  ⇒ 分類問題で使用される誤差基準

正解に対応する項目が
1でないことに対し，
ペナルティ

出力	正解
0.018	0
0.980	1
0.002	0



ニューラルネットワークの学習

- 最適化基準：全サンプルに対する誤差の合計を最小化する

誤差関数：クロスエントロピー誤差
　　平均二乗誤差　など

辺の重みの学習

- 全部の辺の重みを個別に調整する
  - 重みの初期値は辺ごとにランダムに散らす（重要）
  - 学習時，複数の辺で連携したりはしない
  - コスト関数を辺の重みで偏微分し，その逆方向に変化させる
  - ある辺の重みwのt回目の更新の値をw_tとすると
    は以下のように計算される（勾配法の場合）

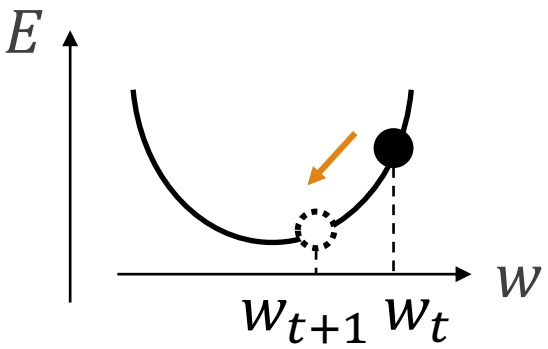
$$w_{t+1} \leftarrow w_t - \alpha \cdot \left.\frac{\partial E}{\partial w}\right|_{w=w_t}$$

  - Eはコスト関数，　αは学習率と呼ばれユーザが事前に決める値

辺の重みの更新例

$$w_{t+1} \leftarrow w_t - \alpha \cdot \left.\frac{\partial E}{\partial w}\right|_{w=w_t}$$

コスト関数



傾きが正なので
負の方向性に更新

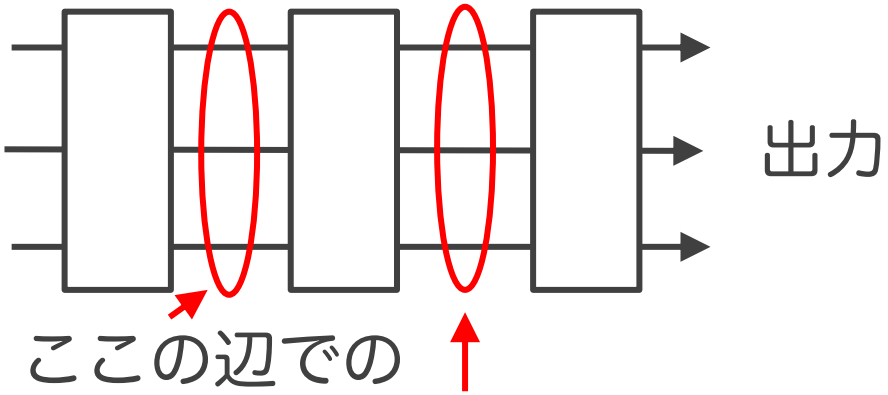
辺の重みの学習

- 重みの更新方法（更新量の調整など）として複数のアルゴリズムがある
  - RMSProp
  - AdaGrad
  - Adam など

⇒ Adam がよく使われる

誤差逆伝播法

- 「誤差関数を各辺の重みで偏微分する」計算を高速に行う手法
- 出力層の誤差（誤差の偏微分）を入力層方向に伝播
  ⇒ 誤差逆伝播法（バックプロパゲーション）と呼ぶ



ミニバッチ学習

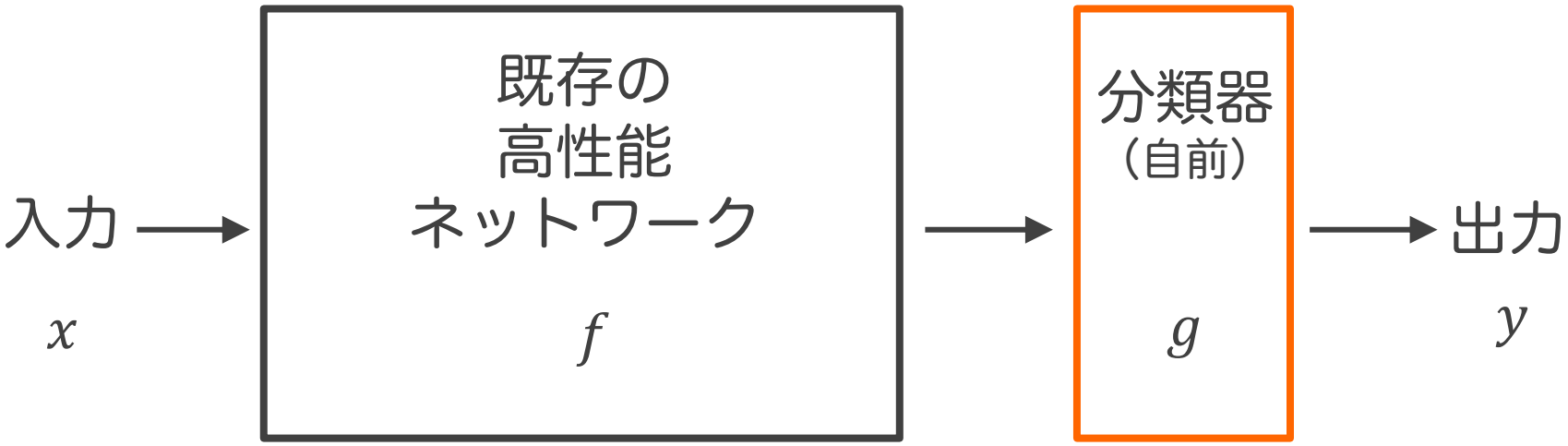
- 機械学習モデルの学習をする際，誤差を求めるには，すべてのサンプルに対する誤差を計算する必要がある
- これは時間がかかるのでやめる
  ⇒ ミニバッチ学習
  - 一部のサンプルのみから求めた誤差を用いてパラメータを更新する
  - その際のサンプルのサイズをバッチサイズと呼ぶ

転移学習

転移学習

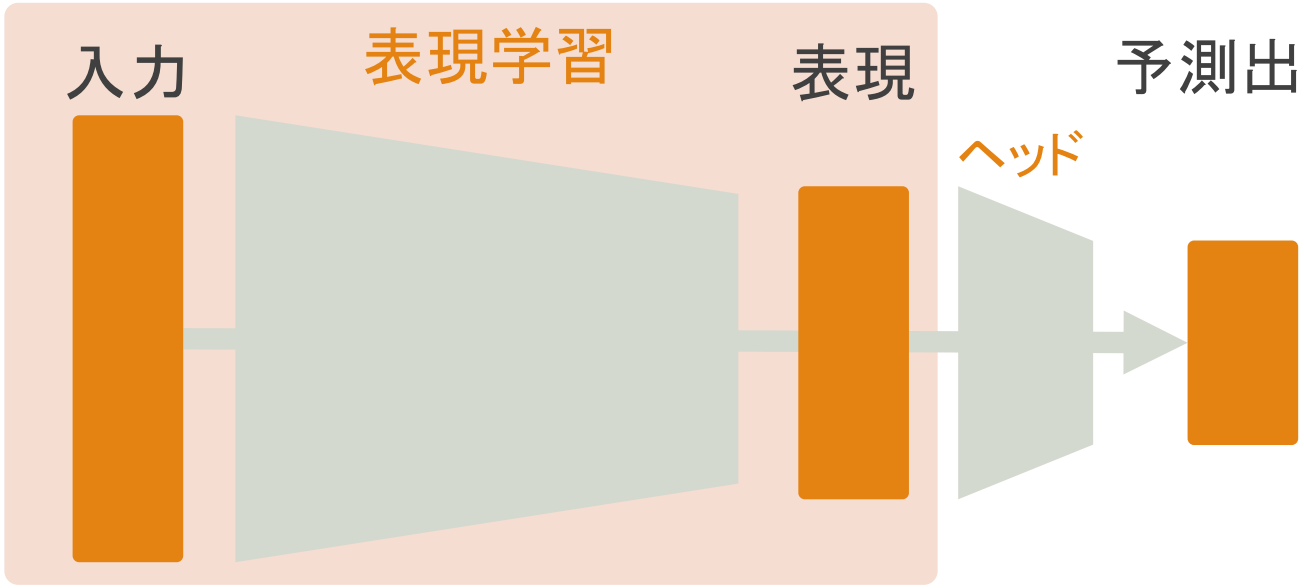
- 学習済みのニューラルネットワークを活用して自分用のネットワークを作る
- 画像分類での活用例
  - 画像分類では前半の層で画像のパターンを認識し，最終層付近で出力を計算する
  - そこで，前半部分に学習済みネットワークを使う
  - 出力生成部分だけを手持ちデータを使って学習する
  - 学習済み部分も手持ちデータで再学習
    ⇒ ファインチューニング

転移学習



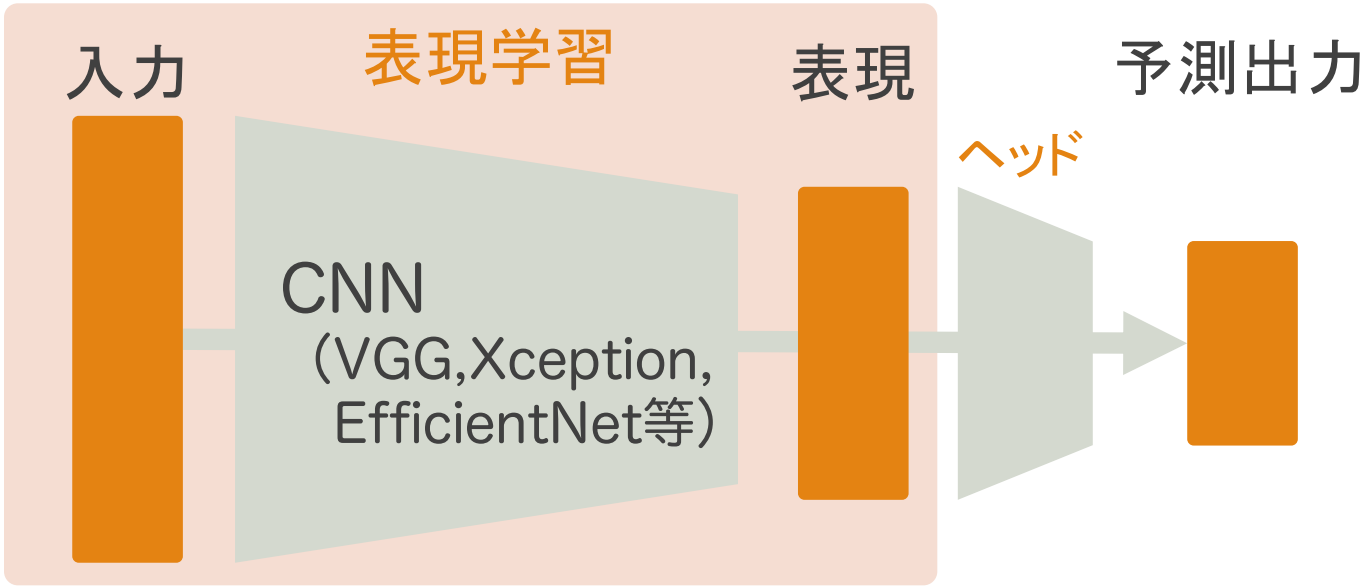
$$y = g \circ f(x)$$

深層学習による予測モデルの典型的な構造



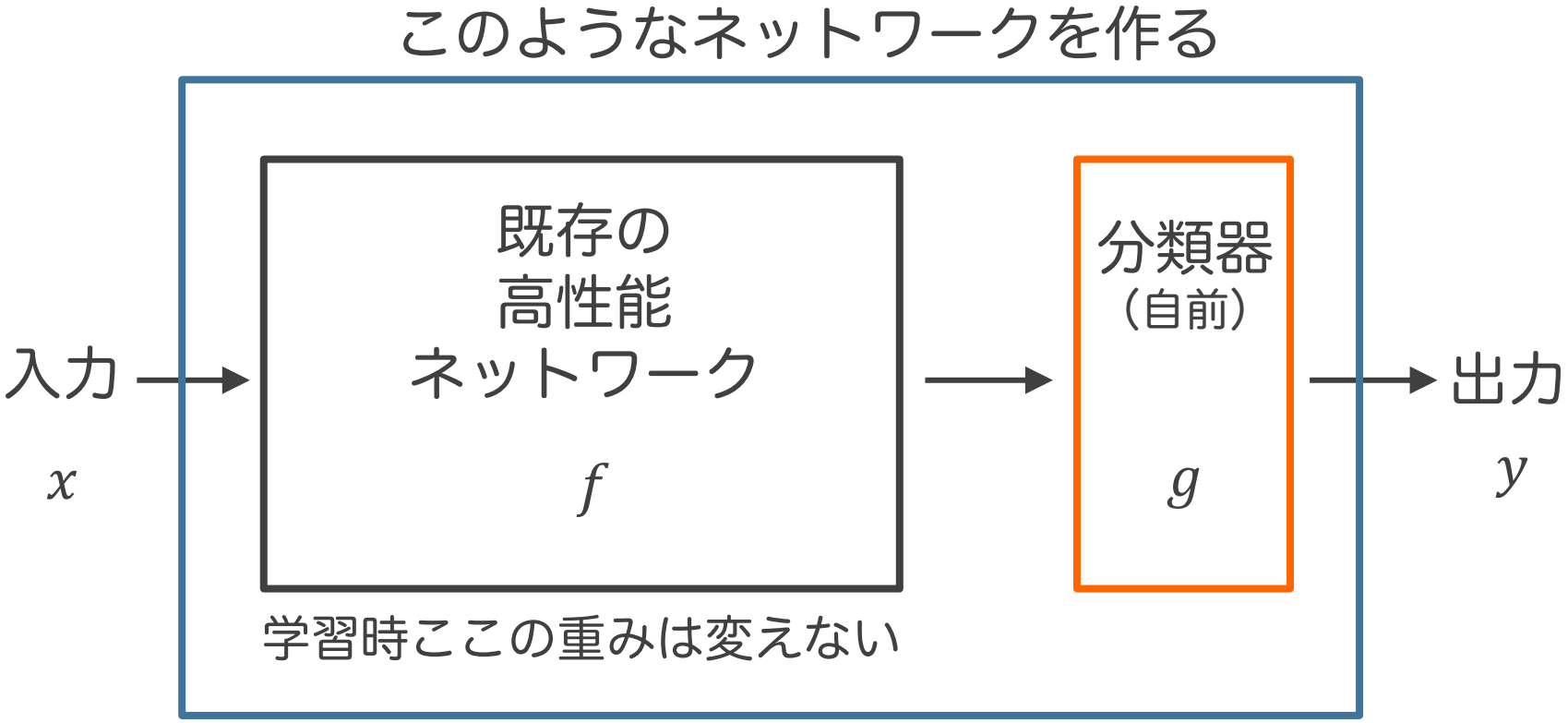
※ 表現学習部分は既存モデルを使いまわせる
（重み（内部パラメータ）はそのまま使う（転移学習） or ファインチューニング）
※ 対照学習と呼ばれる表現学習のための自己教師あり学習手法も知られる

画像分類



画像　　画像パターン情報　　分類結果

転移学習の学習方法



$$y = g \circ f(x)$$

過学習への対処

過学習

- 学習データに対してだけは高性能だが,
  未知のデータに対して性能が低い状態
- 過学習の検出
  - 検証用データで性能評価を行う
    ⇒ 学習用と検証用で性能差が大きければ過学習疑い
- 過学習への対処
  - データ数を増やす
  - モデルに正則化（過学習抑制処理）を入れる

※ 深層学習ではモデルを複雑にしても過学習が起きない

ドロップアウト

- ニューラルネットワークにおける正則化手法
- 学習時ランダムにユニットを無効化する
- 例えばある層のドロップアウト率が0.5なら
  - 層の出力ベクトルのうち，半分の要素は0
  - 推論時は出力の値を0.5倍にする
- なんでこんなことするの？
  - 開発者のHintonいわく，銀行の窓口の対応者をいつも変えることで不正が起きにくくなるのと一緒

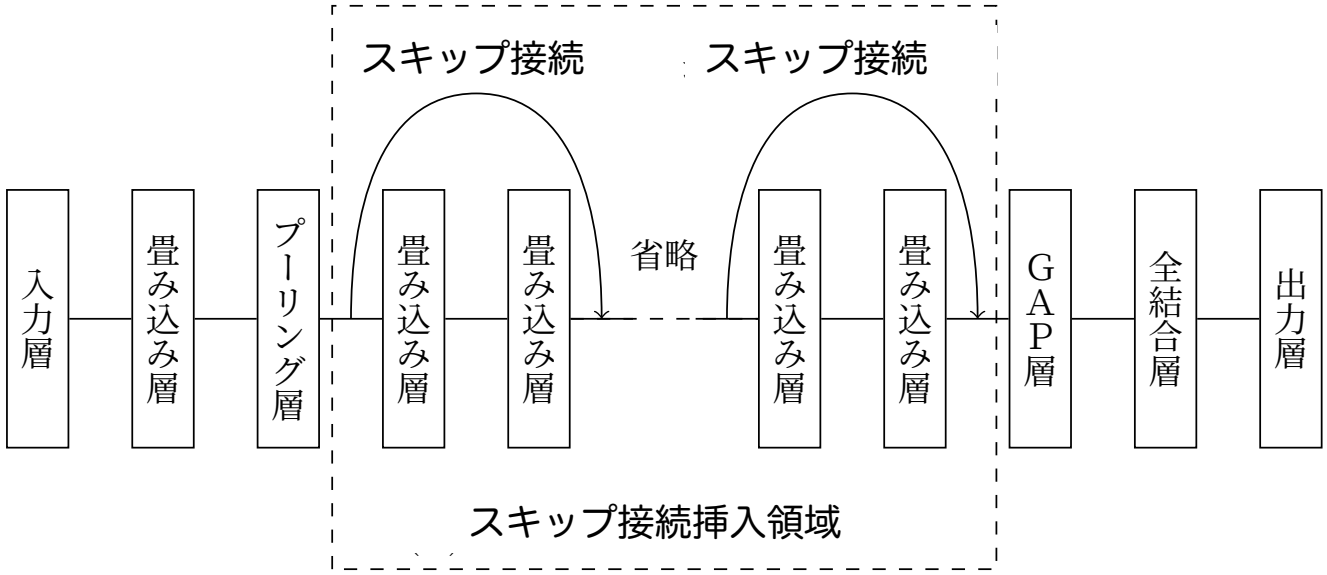
畳み込みニューラルネットワークの進展

代表的なCNN

- AlexNet（2012年）
  - CNNで画像分類を高精度に行えることを示した
  - GPU利用で高速化
- VGGNet（2014年）
  - 既存技術の組合せで性能アップ
  - 層の数によりVGG16とVGG19がある
  - シンプルな構造のため転移学習に向く
- GoogLeNet（2014年）：Inceptionモジュール

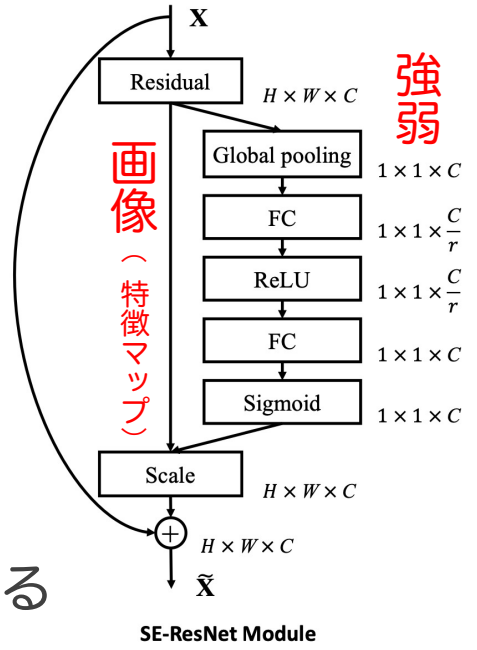
ResNet（残差ネットワーク）（2015年）

- 層の数をこれまでの20層程度から152層に！1202層でも動く！
- 「スキップ接続」により層を深くできる
- スキップ接続のアイデア
  - 各層において，$f(x) = 0$でなく，
    恒等写像（何もしない関数）から学習を開始する
    → 学習が進まない層があっても悪影響がない（何もしないだけ）



SENet（2017年）

- 圧縮興奮機構を導入
  - 自然言語処理のセルフアテンションに類似する処理
  - フィルタ外の情報を参照できるようになった
- アイデア
  - 他のフィルタの値と自フィルタの値の関係を学習することにより，他フィルタの値に応じて自フィルタの値を小さくする



SE-ResNet Module

直感的なイメージ：
目や口のパターンがない時，鼻のパターンの値も小さくする

Jie Hu et al. Squeeze-and-Excitation Networks. CVPR2018.
https://arxiv.org/abs/1709.01507

その他

- ニューラルアーキテクチャ探索（NAS）
  - EfficientNet（2019年）：
    NASで良いネットワークを発見．当時最高性能
  - EfficientNetV2（2021年）：
    色々工夫してさらに最高性能
- その他
  - Vision Transformer（2020年）：
    自然言語処理のTransformerを画像処理でも利用

実装

深層学習プラットフォーム

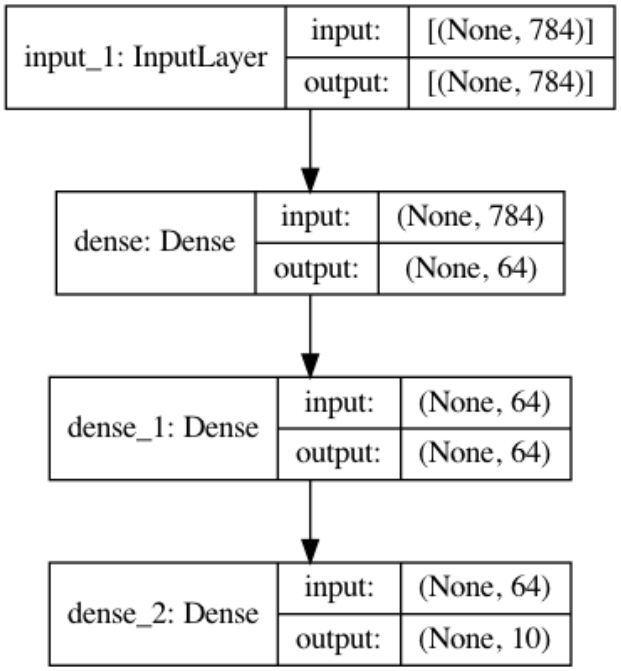
- TensorFlowとPyTorchが2強
  - 日本のPFN製のChainerは開発中止
  - 一番簡単に使えると言われているKerasはTensorFlowに吸収された
  - 最近だとJAXも使われている
- 本講座では，TensorFlowの中のKerasを使用する

Keras利用の流れ

1. 準備：モデルを記述後コンパイル
   - model.compile()
2. 学習
   - model.fit()
3. 推論
   - model.predict()
   - 正解率を測定する時は，model.evaluate()

Kerasの3つの使い方

1. Sequential model：
   入力層から順に各層を記述



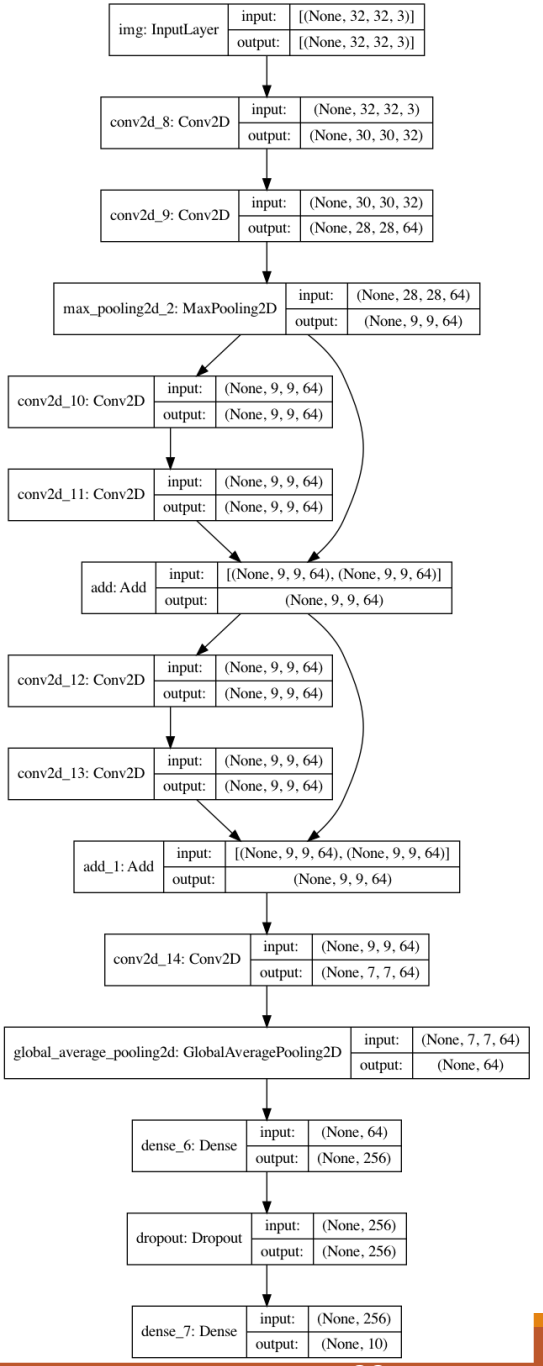
Sequential modelで作成できる
ネットワークの例

Keras functional API
https://keras.io/guides/functional_api/

Kerasの3つの使い方

2. Functional API：
   層の間の接続情報も自分で記述
   （多くのケースではこれで十分）

   Functional APIで作成できる
   ネットワークの例



3. Model subclassing：
   内部処理を自分でカスタマイズ

Keras functional API / models
https://keras.io/guides/functional_api/　https://keras.io/api/models/

深層学習の実行環境

- GPUを使うと大幅に高速化できる
  - 深層学習の主要な処理は行列計算であるが,
    GPUは行列計算が得意なため
  - 「学習」と「推論」の両方にGPUを使えるが,
    「学習」は時間がかかるため特に効果が大きい
- 今回はGoogle ColaboratoryでGPU環境を使用する