情報管理

第10回：ニューラルネットワークその１

今回の講義内容

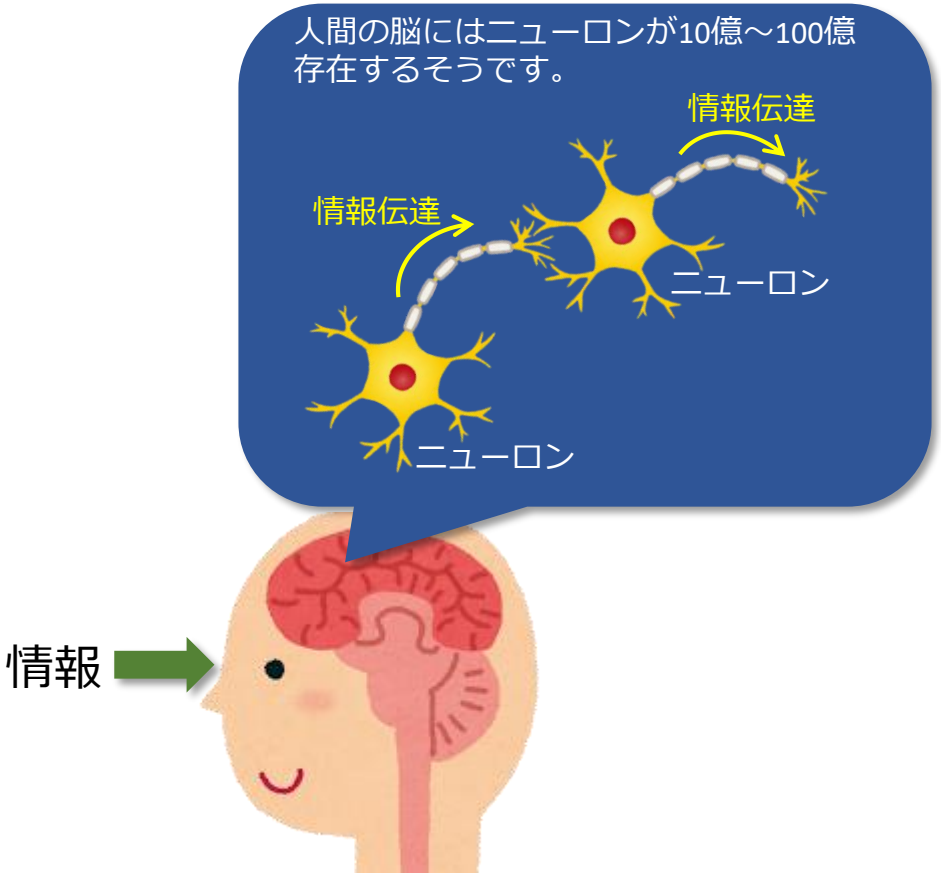
前回は教師「なし」クラスタリングの解説をしましたが,
今回は教師「あり」クラスタリングの話に戻り，近年の人工知能分野で
活発に研究されている「ニューラルネットワーク」について解説します。

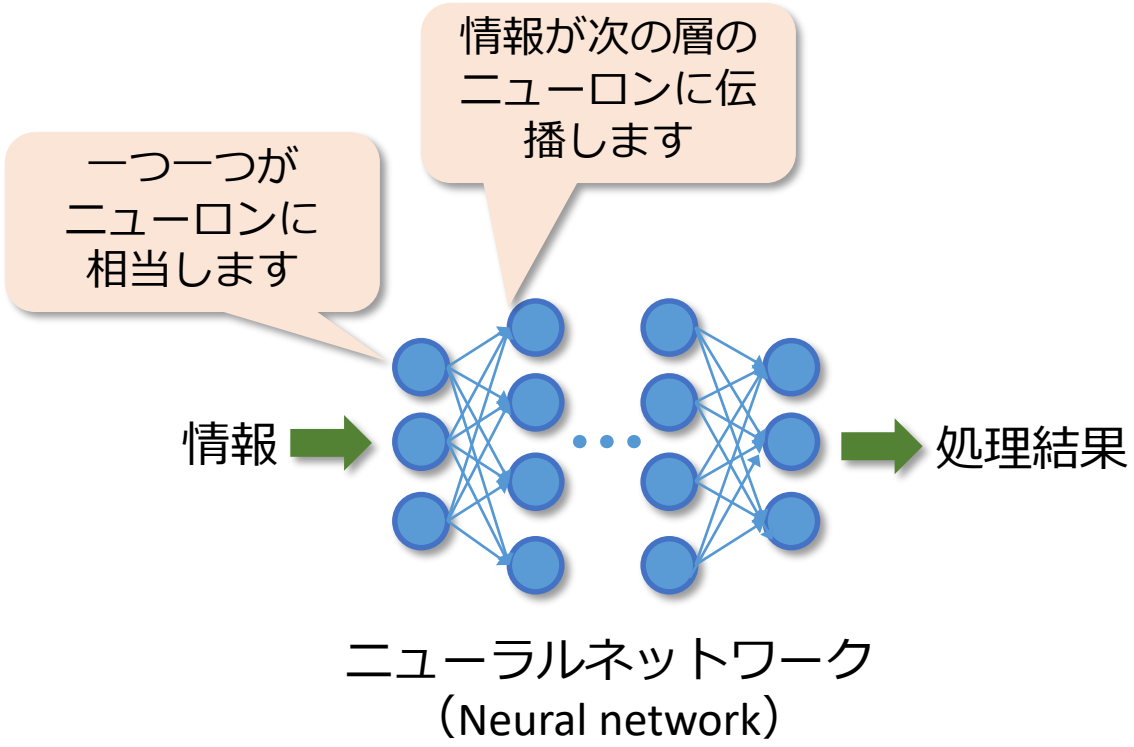
ニューラルネットワークは他の授業でも習っているかもしれませんが,
この授業では，第６回に説明したロジスティック回帰からの発展という
解釈でニューラルネットワークについて解説していきます。

ニューラルネットワークとは

ニューラルネットワークとは

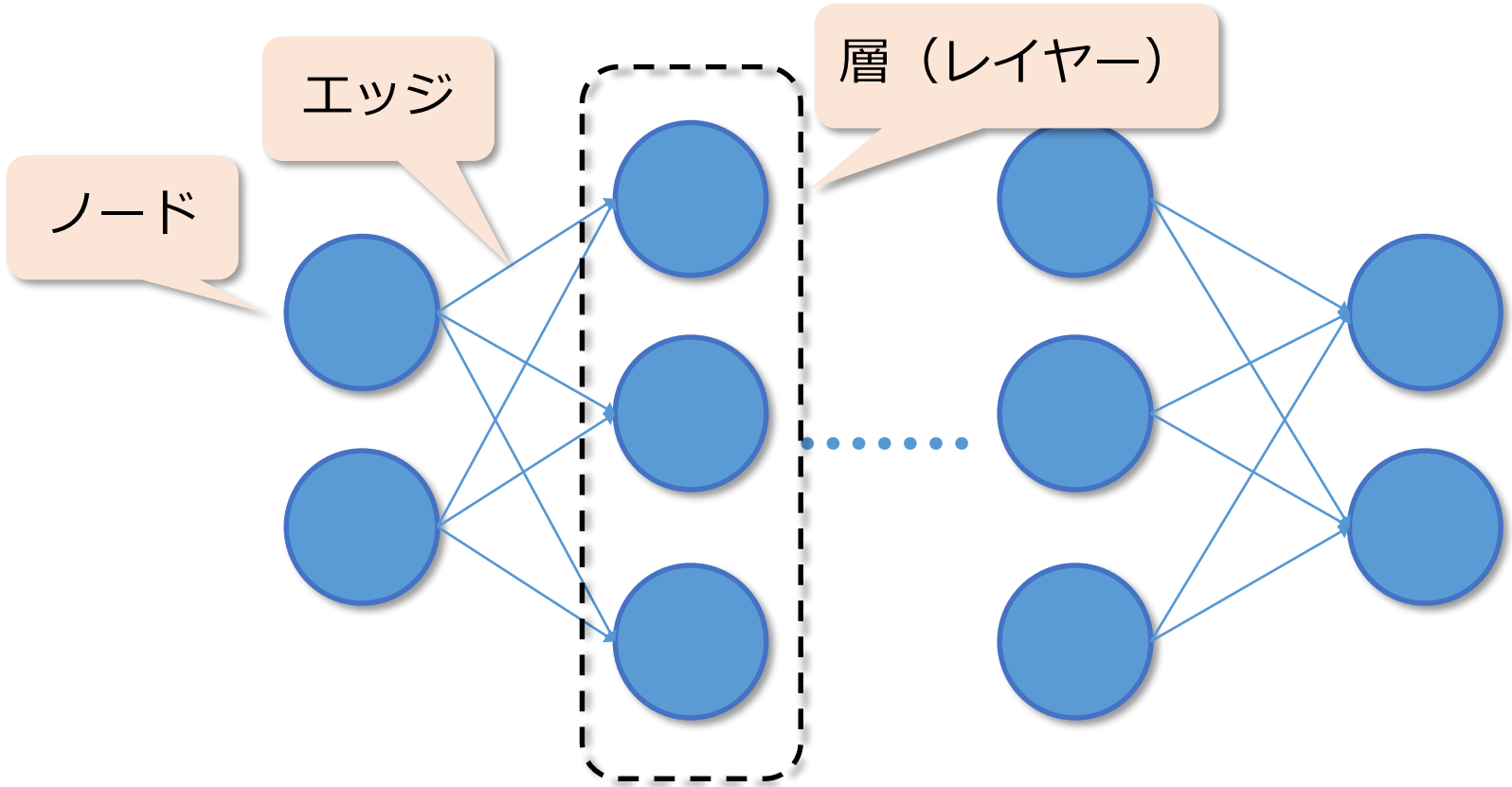
情報が脳の神経細胞（ニューロン）を伝播していくことで処理される様子を模擬したモデルです。





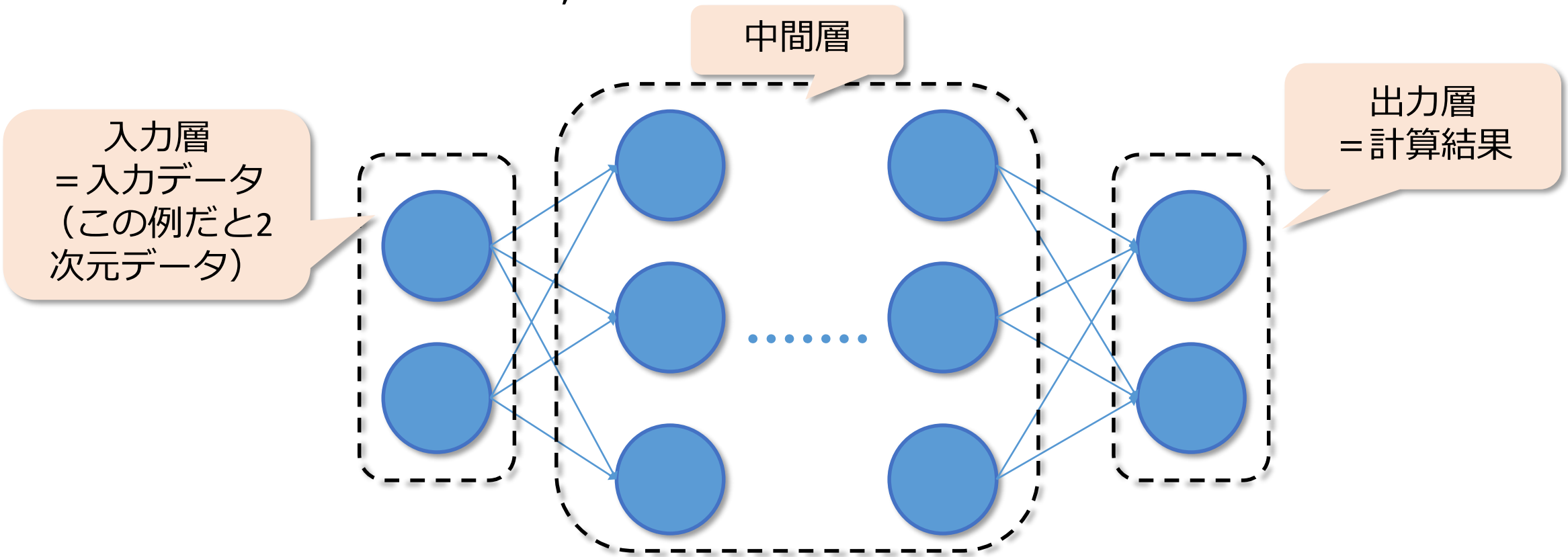
ニューラルネットワークとは

図の丸一つ一つを**ノード**と呼びます。
図のノードが縦に並んだまとまりを**層（レイヤー）**と呼びます。
ノード間をつなぐ矢印を**エッジ**と呼びます。



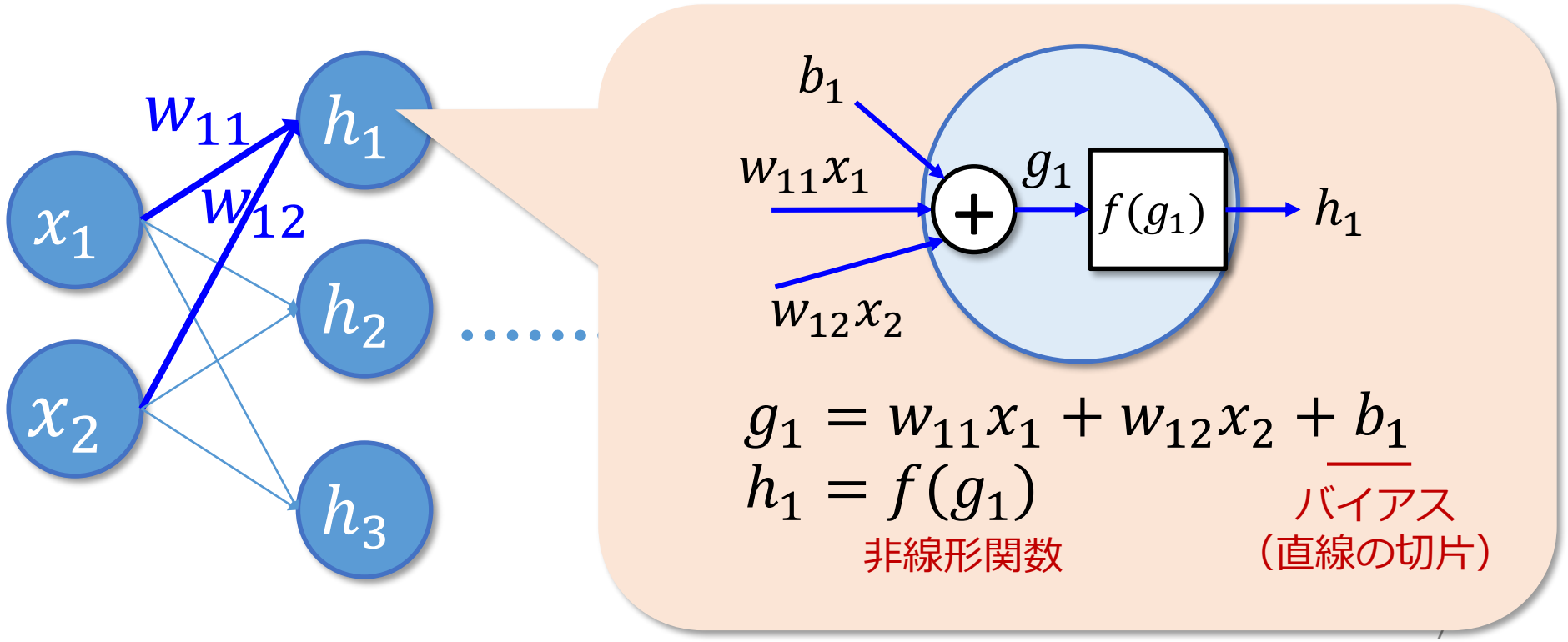
ニューラルネットワークとは

最初の層は**入力層**と呼ばれ，入力データのベクトルが格納されます。
最後の層は**出力層**と呼ばれ，計算結果のベクトルが格納されます。
間の層は**中間層**と呼ばれ，具体的な計算が行われます。



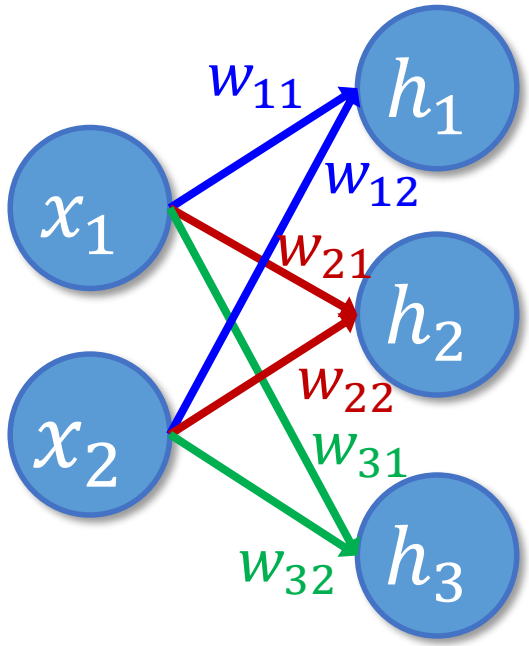
ニューラルネットワークとは

エッジには**重み**が存在し，エッジを通って次の層へ値が伝播する際，
伝播元の値の**重みづけ和**（線形変換）が計算され，さらに
非線形（直線でない）関数を通した値が伝播先のノードに保存されます。
つまり，線形変換と非線形変換が1回ずつ行われます。



ニューラルネットワークとは

層同士の情報伝達の式は，行列とベクトルを使って表現可能です。



$$\begin{bmatrix} g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} w_{11}x_1 + w_{12}x_2 + b_1 \\ w_{21}x_1 + w_{22}x_2 + b_2 \\ w_{31}x_1 + w_{32}x_2 + b_3 \end{bmatrix}$$

$$= \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

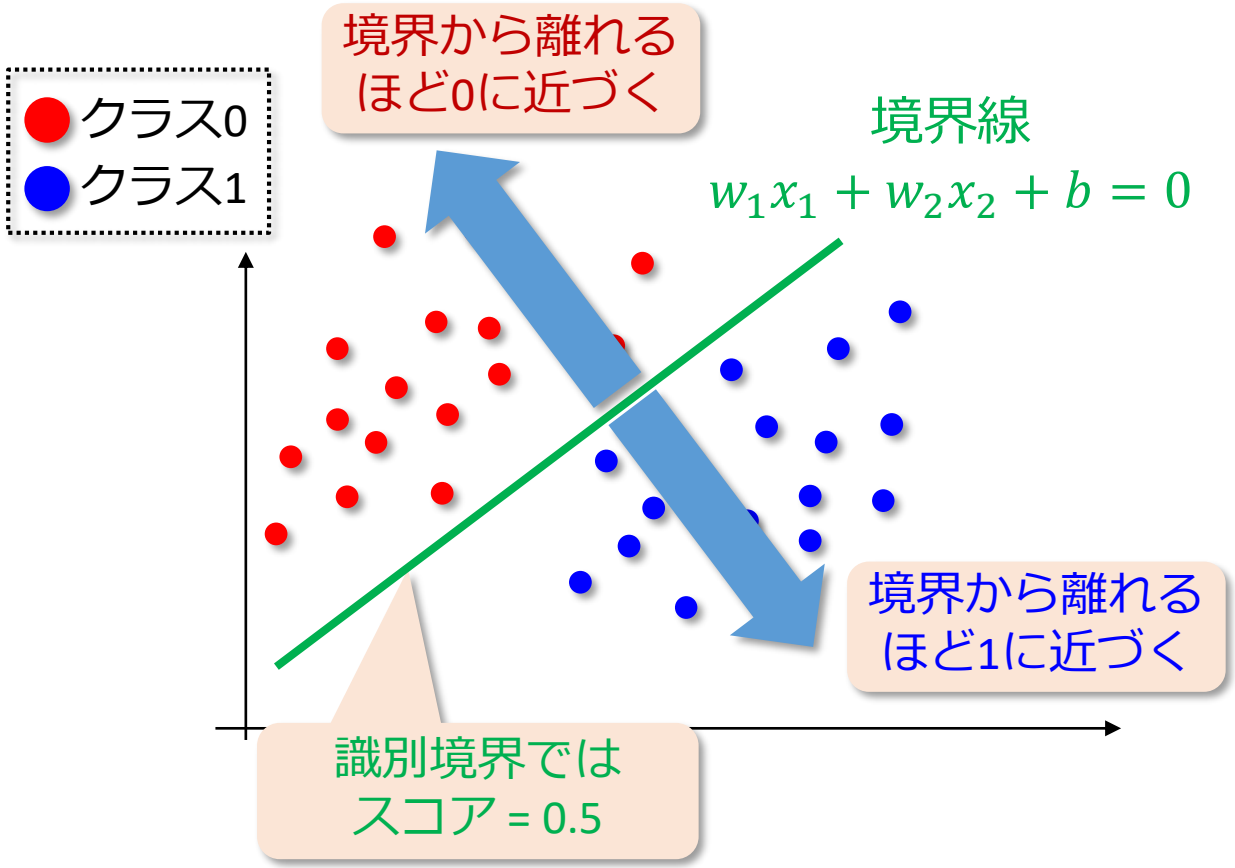
$$= \mathbf{Wx + b}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix} = \begin{bmatrix} f(g_1) \\ f(g_2) \\ f(g_3) \end{bmatrix}$$

ニューラルネットワークによる ロジスティック回帰の表現

ロジスティック回帰：おさらい

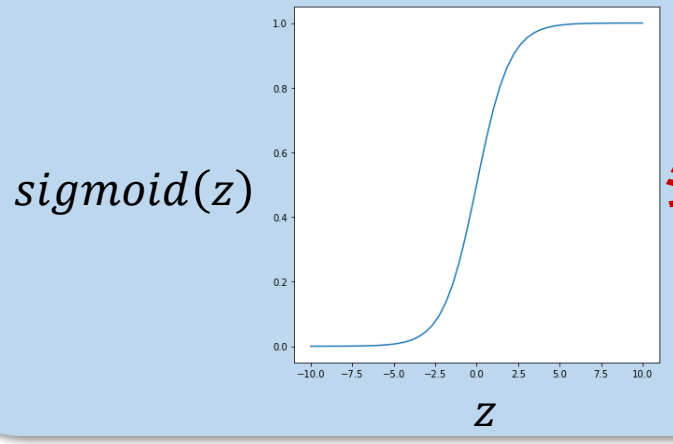
2種類のクラスに対して，それぞれのクラスラベルを0，1，としたとき，
あるサンプルの識別結果を 0 から 1 までの連続値スコアとして出力する。



2次元のサンプル (x_1, x_2)のロジスティック回帰結果を $\hat{y}$ とすると

$$z = w_1x_1 + w_2x_2 + b$$

$$\hat{y} = sigmoid(z) = \frac{1}{1 + e^{-z}}$$



シグモイド関数

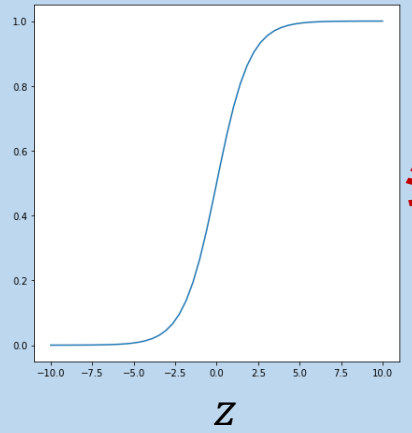
ロジスティック回帰をニューラルネットワークで表現

中間層が無く，出力層のノード数が１，非線形関数にシグモイド関数を使ったニューラルネットワークは，ロジスティック回帰と等価になります。

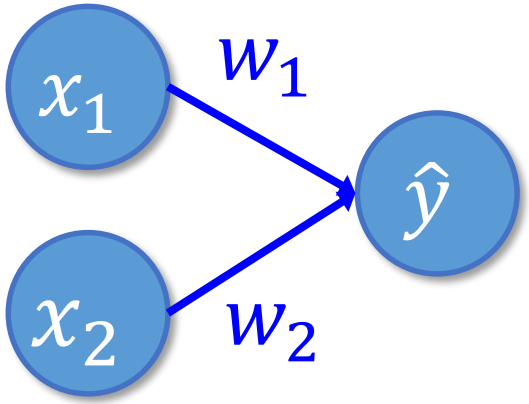
2次元のサンプル(x_1, x_2)のロジスティック回帰結果を$\hat{y}$とすると

$$z = w_1 x_1 + w_2 x_2 + b$$

$$\hat{y} = sigmoid(z) = \frac{1}{1 + e^{-z}}$$



シグモイド関数

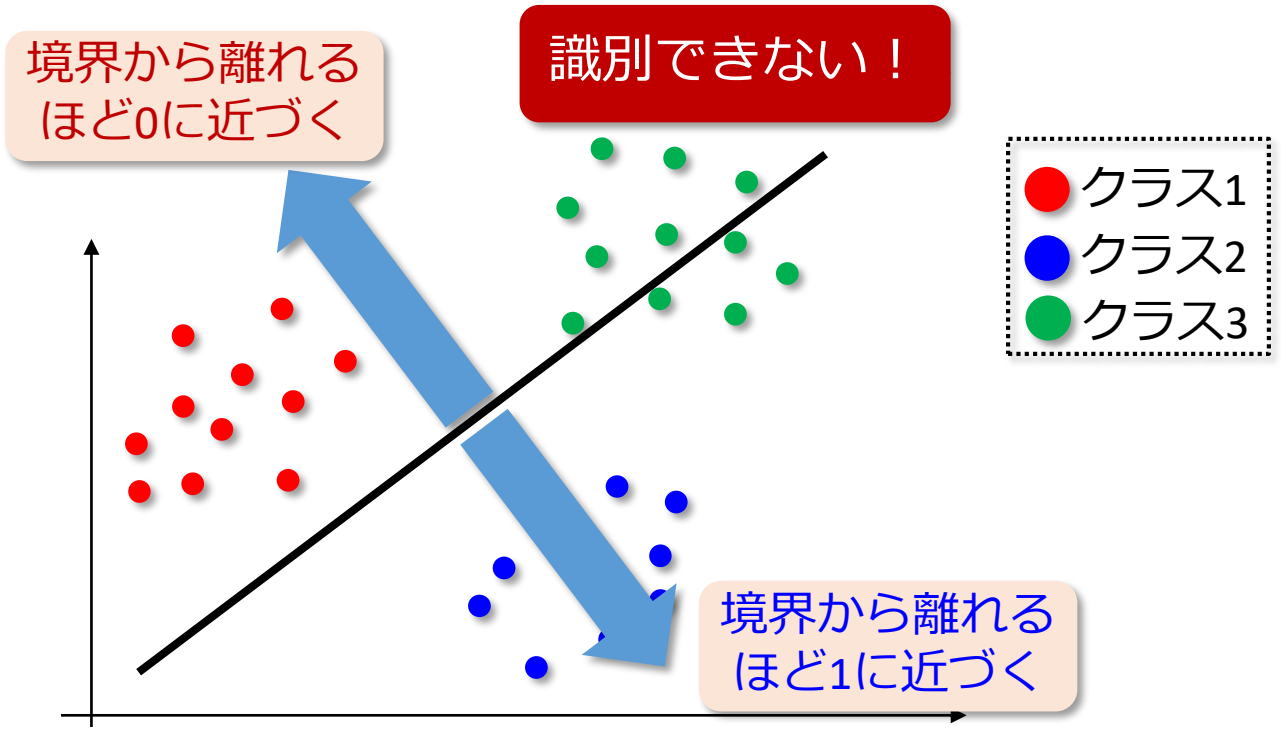


$$z = w_1 x_1 + w_2 x_2 + b$$
$$\hat{y} = f(z) = sigmoid(z)$$

ロジスティック回帰から
ニューラルネットへの拡張

ロジスティック回帰の多クラス分類への拡張

ロジスティック回帰では，単一の線形関数を使い，その出力が0.5以上だとクラス1，0.5未満だとクラス0と識別していました。
この方法では，3クラス以上（多クラス）の識別ができません。

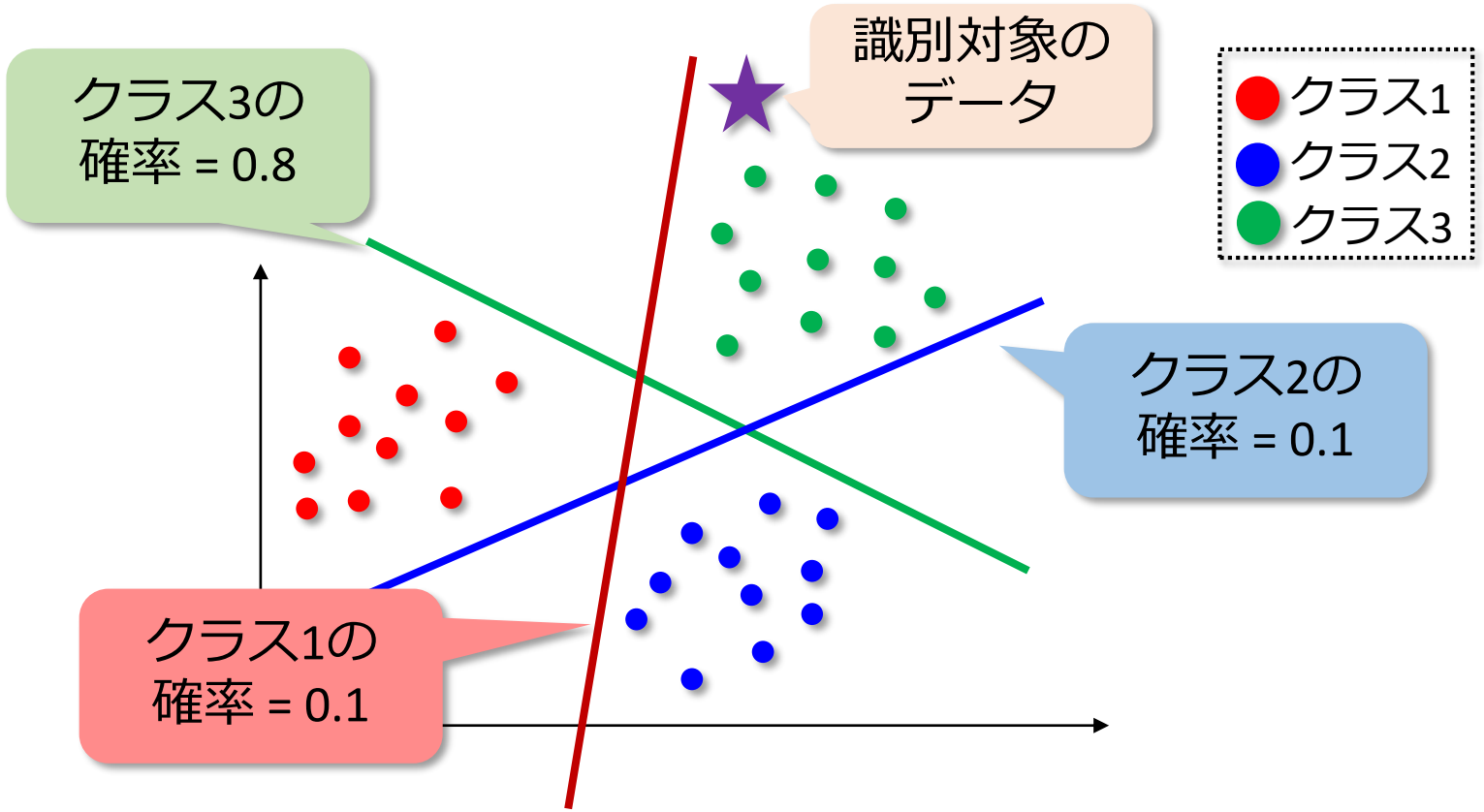


ロジスティック回帰の多クラス分類への拡張

そこで，線形関数を一つではなく，３つ（クラス数）使うことを考えます。

各線形関数は，それぞれが担当するクラスの確率を計算します。

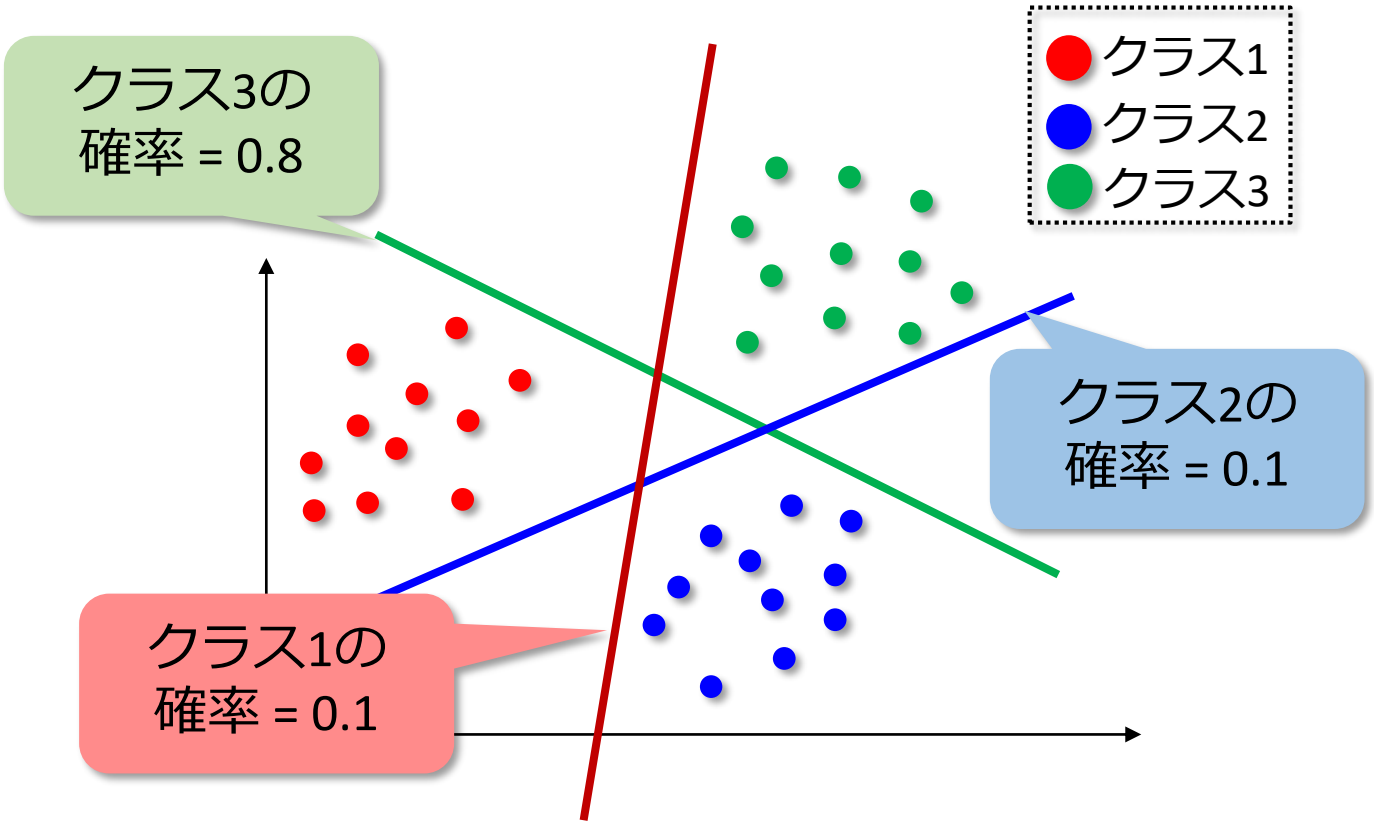
そして，確率が最大となるクラスを出力することで，多クラス識別ができます。

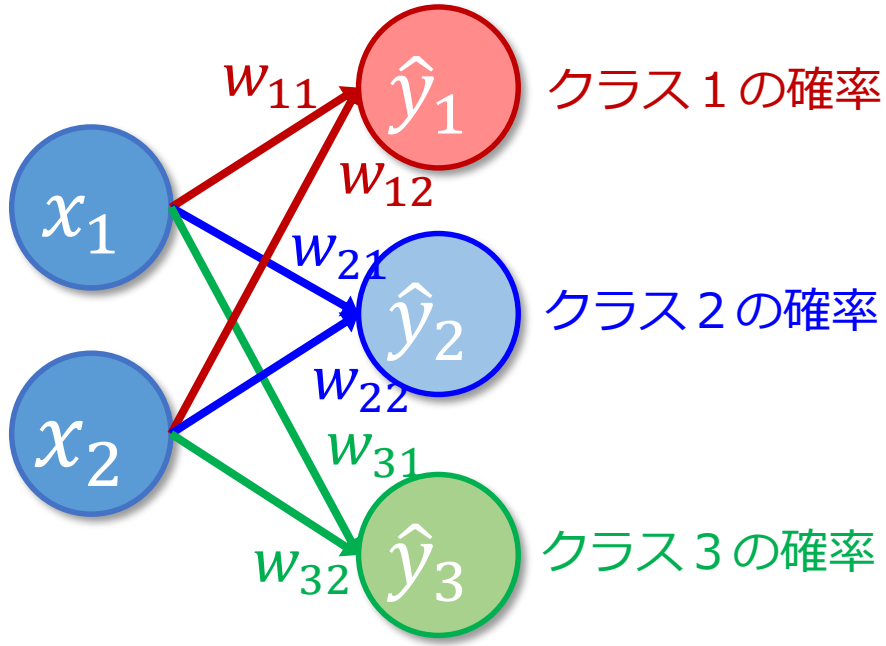


ロジスティック回帰の多クラス分類への拡張

これをニューラルネットワークで表現すると，以下のようになります。

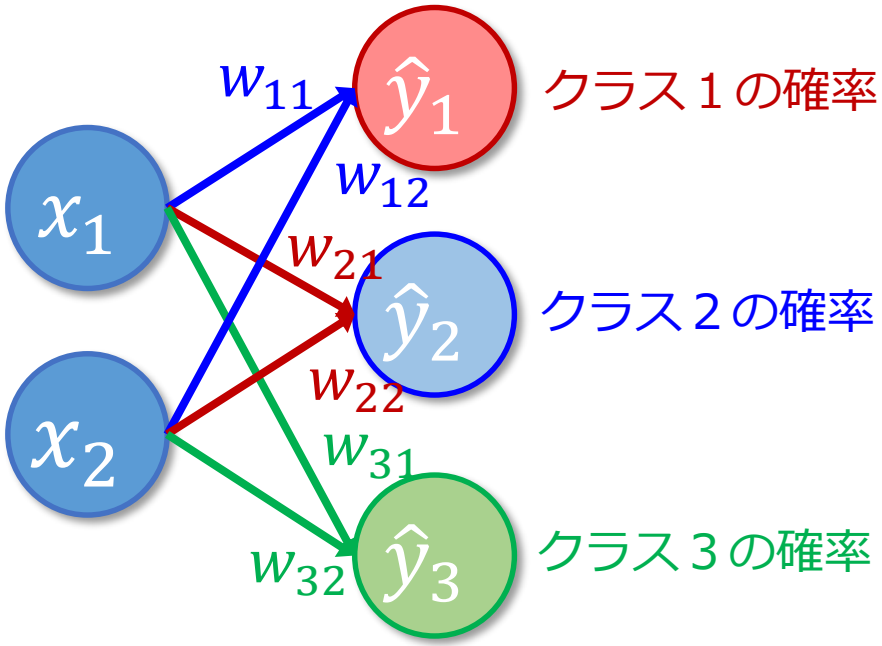
・・・しかし，これはまだ不完全です。





$$z_k = w_{k1}x_1 + w_{k2}x_2 + b_k$$
$$\hat{y}_k = f(z_k) = sigmoid(z_k)$$
$$(k = 1, 2, 3)$$

ロジスティック回帰の多クラス分類への拡張

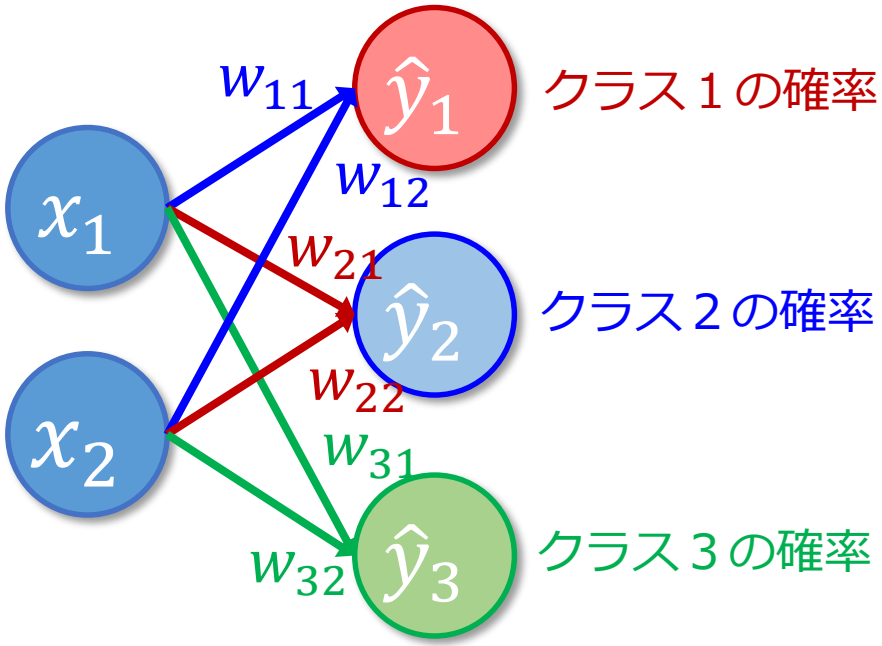


$$z_k = w_{k1}x_1 + w_{k2}x_2 + b_k$$
$$\hat{y}_k = f(z_k) = sigmoid(z_k)$$
$$(k = 1, 2, 3)$$

なぜ上のモデル化は不完全？

→ $\hat{y}_k = f(z_k) = sigmoid(z_k)$ はクラスｋごと独立に計算しているため，
クラス1，クラス2，クラス3の確率 $\hat{y}_1, \hat{y}_2, \hat{y}_3$ を足しても１になりません。
つまり，確率の性質を満たしていないからです。

ロジスティック回帰の多クラス分類への拡張



$$z_k = w_{k1}x_1 + w_{k2}x_2 + b_k$$
$$\hat{y}_k = f(z_k) = \boldsymbol{softmax(z_k)}$$
$$(k = 1, 2, 3)$$

ではどうすれば良いの？

→ シグモイド関数の代わりに，総和が１になることを保証する**ソフトマックス関数**を使います。

$$softmax(z_k) = \frac{e^{z_k}}{\sum_c e^{z_c}}$$

シグモイド関数とソフトマックス関数の関係

ソフトマックス関数において，クラス数が 2 で，かつ一方のクラスの z が 0 のとき，シグモイド関数と等価になります。

言い換えると，ソフトマックス関数はシグモイド関数を多クラス分類できるように一般化したものとなります。

例：$z_1 = 0$として，z_0のソフトマックス関数を計算

$$softmax(z_0) = \frac{e^{z_0}}{\sum_c e^{z_c}} = \frac{e^{z_0}}{e^{z_0} + e^{z_1}}$$
$$= \frac{1}{1 + \frac{e^{z_1}}{e^{z_0}}} = \frac{1}{1 + e^{z_1 - z_0}}$$
$$= \frac{1}{1 + e^{-z_0}}$$
$$= sigmoid(z_0)$$

多クラス回帰の学習

ここでは，損失関数としてクロスエントロピーを使用します。
クロスエントロピーは二つの確率（y_kと$\hat{y}_k$）の近さを測る指標で，
以下の式で定義されます。

$$L_{ce} = -\sum_{k} y_k \log \hat{y}_k \rightarrow minimize$$

$\hat{y}_k$：識別器が出力したクラスｋの確率
y_k：クラスｋの実際の確率（kが正解クラスの場合は1，不正解クラスの場合は0）

多クラス回帰の学習

ロジスティック回帰のときと同じく，勾配降下法を使って学習します。

勾配降下法

$$\hat{W}_{next} \leftarrow \hat{W}_{old} - \mu \frac{\partial L}{\partial W} \qquad \hat{b}_{next} \leftarrow \hat{b}_{old} - \mu \frac{\partial L}{\partial b}$$

合成関数の偏微分の公式を使って，$\frac{\partial L}{\partial w_k}$ および $\frac{\partial L}{\partial b_k}$を計算します。

$$\frac{\partial L}{\partial w_k} = \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial w_k} \qquad \frac{\partial L}{\partial b_k} = \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial b_k}$$

計算すると，以下の勾配の式が得られます（導出は次頁）。

$$\frac{\partial L}{\partial w_{ki}} = (-y_k + \hat{y}_k)x_i \qquad \frac{\partial L}{\partial b_k} = -y_k + \hat{y}_k$$

（勾配式の導出）

$$\frac{\partial L}{\partial w_k} = \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial w_k} \qquad \frac{\partial L}{\partial b_k} = \frac{\partial L}{\partial z_k}\frac{\partial z_k}{\partial b_k}$$

$$z_k = w_{k1}x_1 + w_{k2}x_2 + b_k$$
$$\hat{y}_k = f(z_k) = softmax(z_k)$$

$$\frac{\partial L}{\partial z_k} = \frac{\partial}{\partial z_k}\left[-\sum_k y_k \log \hat{y}_k\right] = \frac{\partial}{\partial z_k}\left[-\sum_k y_k \log \frac{e^{z_k}}{\sum_c e^{z_c}}\right]$$

$$= \frac{\partial}{\partial z_k}\left[-\sum_k y_k z_k + \sum_k y_k \log \sum_c e^{z_c}\right] = \left(-y_k + \frac{e^{z_k}}{\sum_c e^{z_c}} \underbrace{\sum_k y_k}_{=1}\right)$$

$$= -y_k + \hat{y}_k$$

$$\frac{\partial z_k}{\partial w_{ki}} = \frac{\partial}{\partial w_k} w_{k1}x_{i=1} + w_{k2}x_{i=2} + b_k = x_i \qquad \frac{\partial z_k}{\partial b_k} = \frac{\partial}{\partial w_k} w_{k1}x_{i=1} + w_{k2}x_{i=2} + b_k = 1$$

$$\frac{\partial L}{\partial w_{ki}} = (-y_k + \hat{y}_k)x_i \qquad \frac{\partial L}{\partial b_k} = -y_k + \hat{y}_k$$

多クラス識別モデルの動作を確認しよう

10_01_softmax_3class.ipynb を動かして，ソフトマックス関数による多クラス識別モデルの動作を確認しましょう。

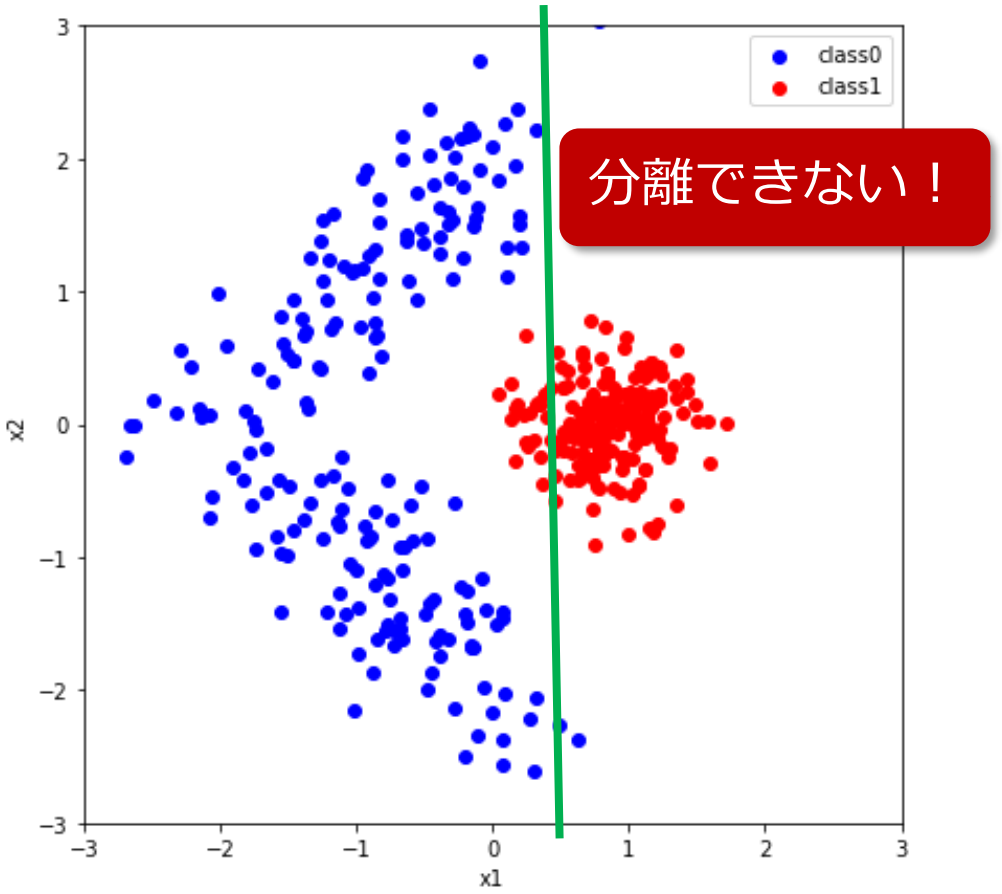
また，10_02_softmax_2class.ipynbを動かして，この方法が2クラス分類でも使えることを確認しましょう。

中間層の導入

線形分離不可能なデータ

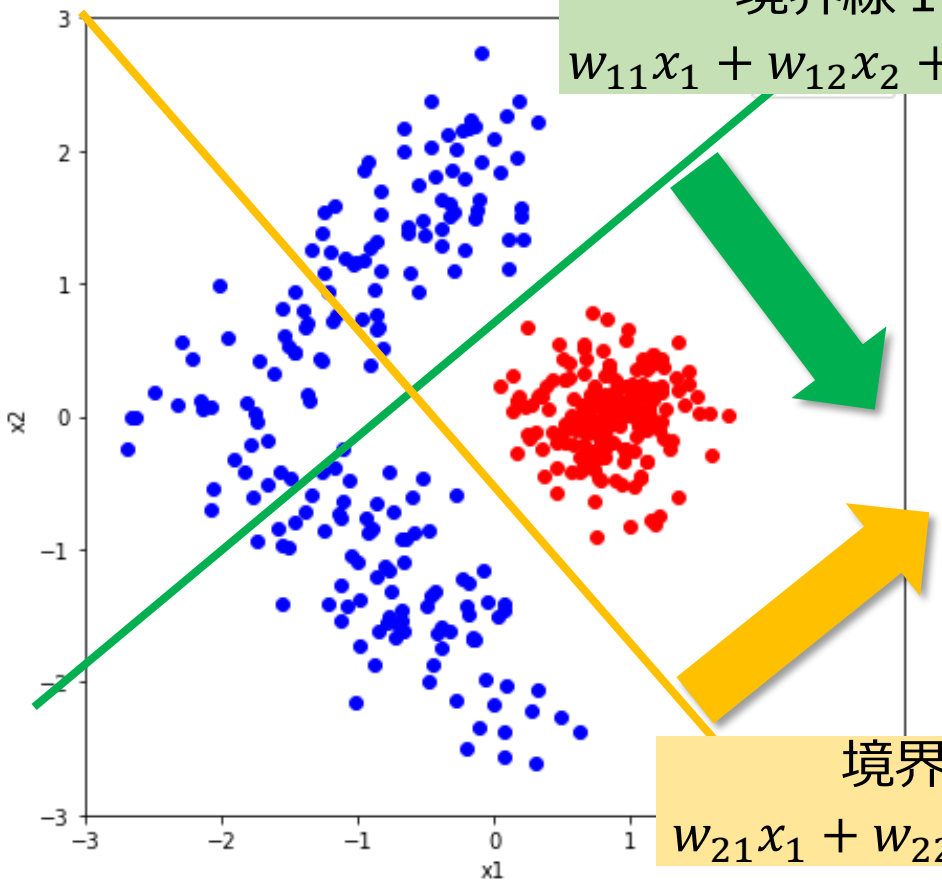
ロジスティック回帰では，直線（線形関数）で分離できないクラスは100%の精度で識別することができません。

このようなデータに対しては，非線形な関数を使って分離を行う必要があります。



非線形関数の例：区分的線形分離

非線形関数の作り方は色々ありますが，ここでは，複数の直線を組み合わせることで非線形関数を作ることを考えます。



境界線 1

$w_{11}x_1 + w_{12}x_2 + b_1 = 0$

境界線 2

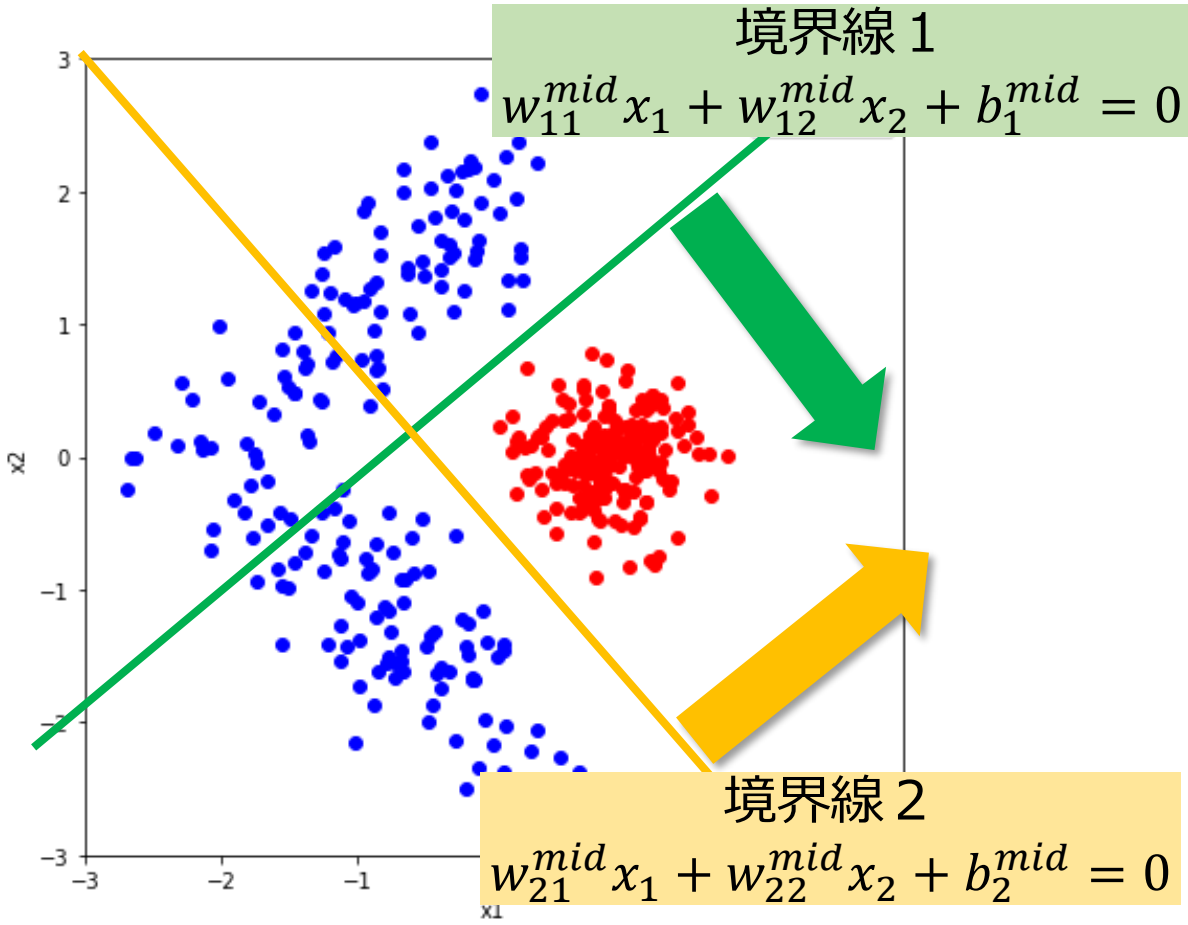
$w_{21}x_1 + w_{22}x_2 + b_2 = 0$

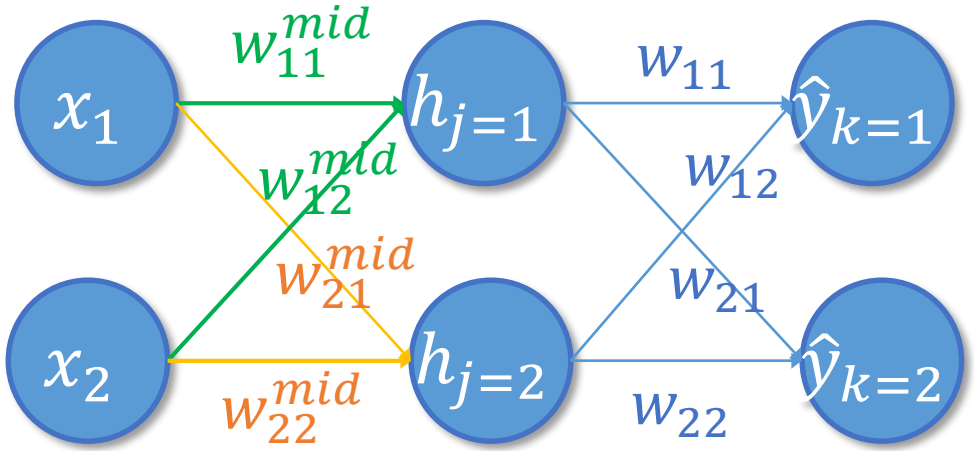
境界線1より下，かつ境界線2より上なら，赤のクラス

→ if $w_{11}x_1 + w_{12}x_2 + b_1 < 0$ and
$w_{21}x_1 + w_{22}x_2 + b_2 > 0$:
class = ‘red’
else:
class = ‘blue’

中間層を使った区分的線形分離の表現

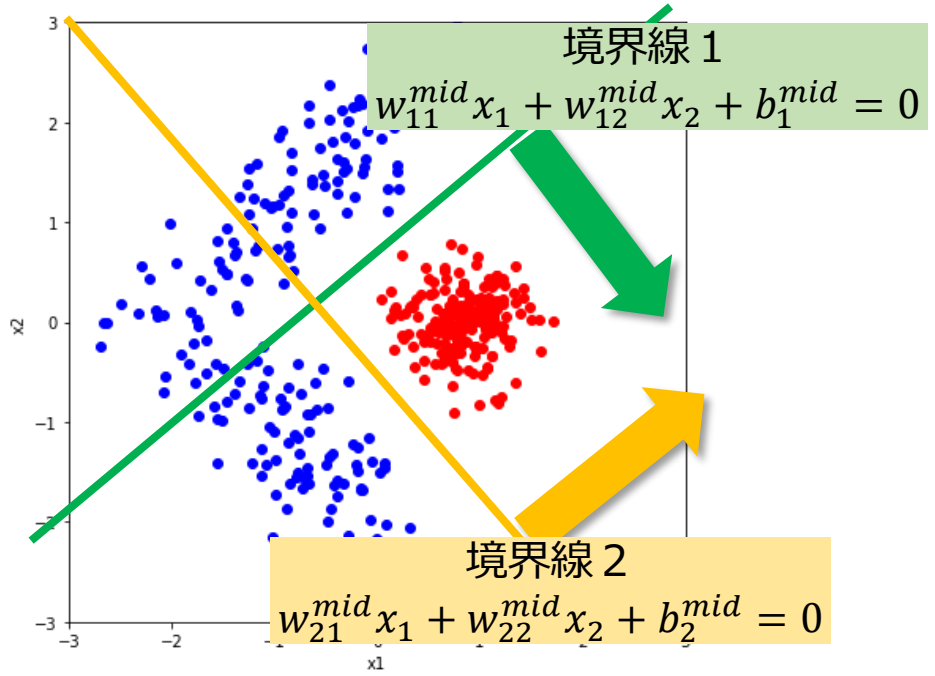
このような区分的線形分離は，**非線形関数にシグモイド関数を使った中間層を導入**することで表現可能です。

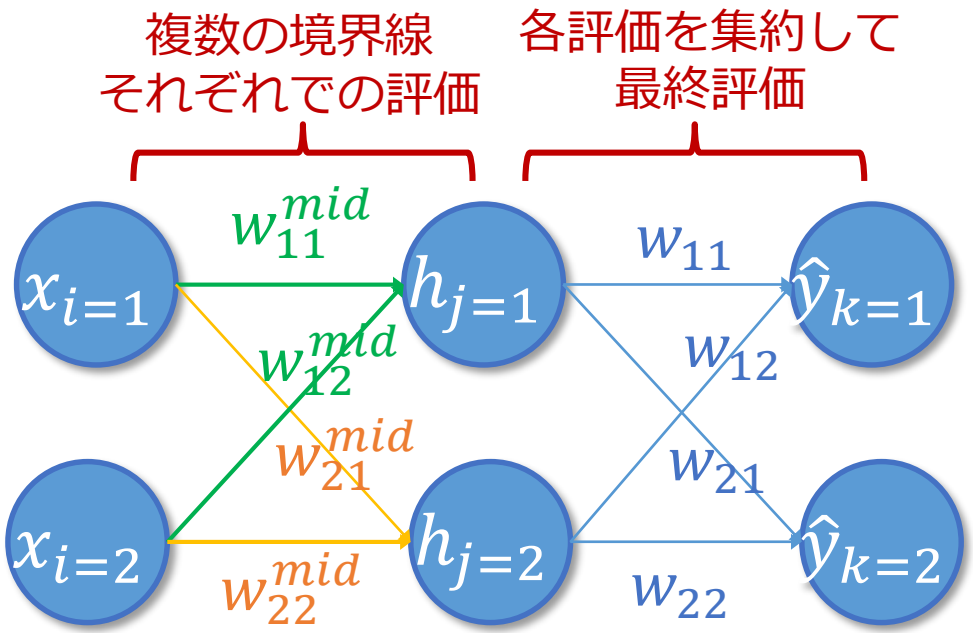




$$g_j = w_{j1}^{mid}x_1 + w_{j2}^{mid}x_2 + b_k^{mid}$$
$$h_j = sigmoid(g_j)$$
$$z_k = w_{k1}h_1 + w_{k2}h_2 + b_k$$
$$\hat{y}_k = f(z_k) = softmax(z_k)$$

中間層を使った区分的線形分離の表現





$$g_j = w_{j1}^{mid} x_1 + w_{j2}^{mid} x_2 + b_k^{mid}$$ それぞれの境界線に対して計算を実施

$$h_j = f(g_j) = sigmoid(g_j)$$ g_j が負なら0に，正なら1に近づく → 条件文のフラグに相当

$$z_k = w_{k1} h_1 + w_{k2} h_2 + b_k$$ 各境界線での評価をさらに集約する

$$\hat{y}_k = f(z_k) = softmax(z_k)$$ 最終結果を出力する

中間層つきニューラルネットの学習

これまで同様，勾配降下法を使って学習します。

クロスエントロピー損失を用いた時の，出力層と中間層の重みおよびバイアスに対する勾配は以下のように計算されます。

（導出は次回に解説します。）

出力層の勾配

$$\frac{\partial L}{\partial w_{kj}} = (-y_k + \hat{y}_k)h_j \qquad \frac{\partial L}{\partial b_k} = -y_k + \hat{y}_k$$

中間層の勾配

$$\frac{\partial L}{\partial w_{ji}^{mod}} = \sum_k \left[(-y_k + \hat{y}_k)w_{kj}\right]\left(1 - h_j\right)h_j x_i \qquad \frac{\partial L}{\partial b_j^{mod}} = \sum_k \left[(-y_k + \hat{y}_k)w_{kj}\right]\left(1 - h_j\right)h_j$$

ニューラルネットワークによる非線形識別を動作確認しよう

10_03_neural_network.ipynb を動かして，中間層つきニューラルネットワークの動作を確認しましょう。

おわりに

今回は，ロジスティック回帰を改良する形で，ニューラルネットワークの導入について話しました。

次回は，ニューラルネットワークの学習方法について，さらに詳しく解説していきます。

レポート課題

第１０回ファイル一式に含まれる，“2class_data_nl2.csv” に対して
ニューラルネットワークを用いて識別を行え。

10_03_neural_network.ipynb の実装などを改良し，可能な限り高精度で識別を行え。

提出ファイルには，少なくとも識別正解率と識別境界の可視化を載せること。

- 課題のメインは 10_03_neural_network.ipynb の改良なので，デフォルト設定での検証は特に行わなくて良い。
- 参考まで，100%の正解率で識別できることは確認済みです。

レポート提出期限：7/12(火) AM10:30，ipynbファイルをhtmlファイルに変換して提出