情報管理

第１１回：ニューラルネットワークその２

今回の講義内容

前回は「ニューラルネットワーク」について紹介しました。

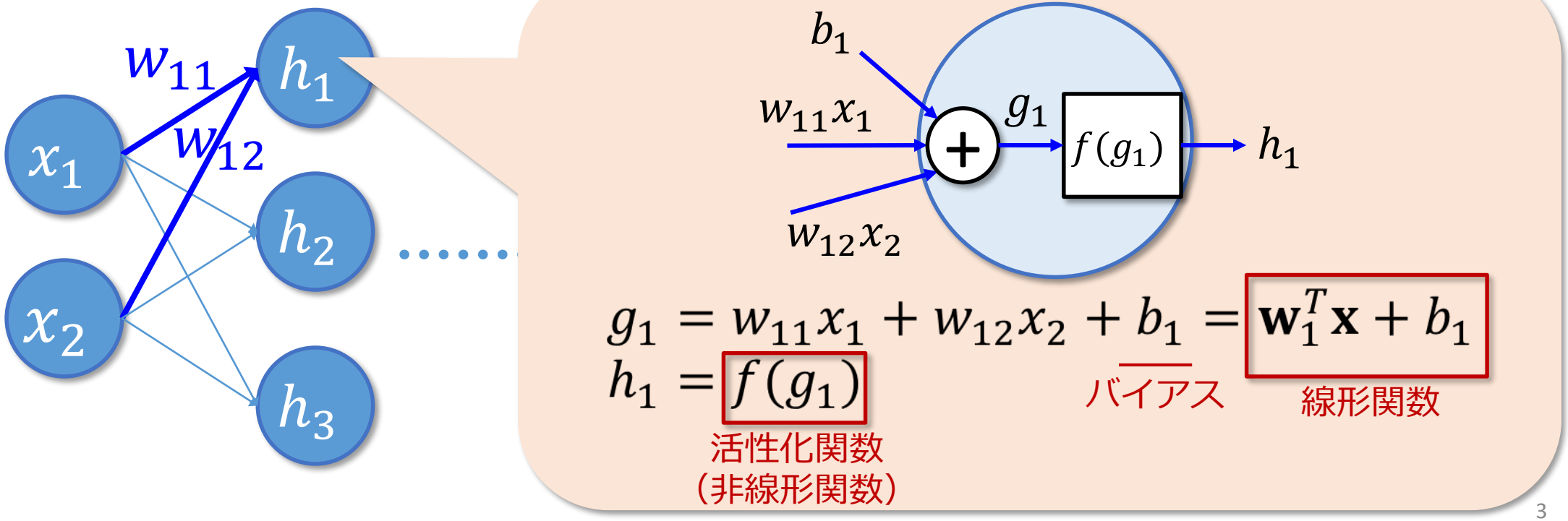
今回はニューラルネットワークについてさらに理解を深めるために,
その学習方法である誤差逆伝播法について解説します。

前回のおさらい

ニューラルネットワークは，ノードの集まりである層と，層間をつなぐエッジからなるモデル。

層から層へ情報が伝播する際，線形変換と非線形変換が行われる。

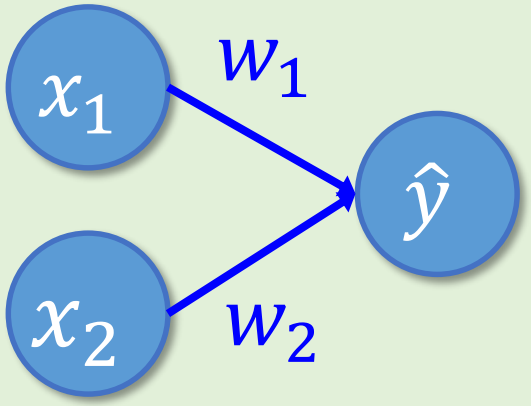
前回の補足：非線形関数のことを**活性化関数**と呼びます。



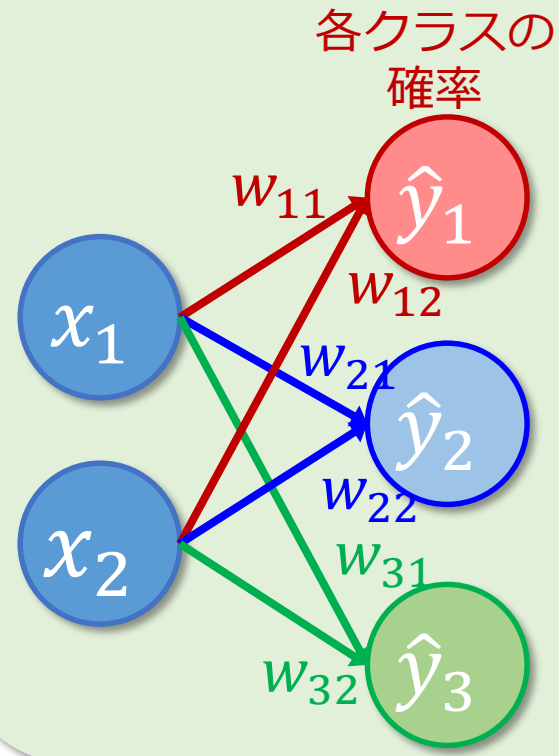
前回のおさらい

ロジスティック関数もニューラルネットワークで表現でき，
多クラス・非線形識別に拡張することが可能。

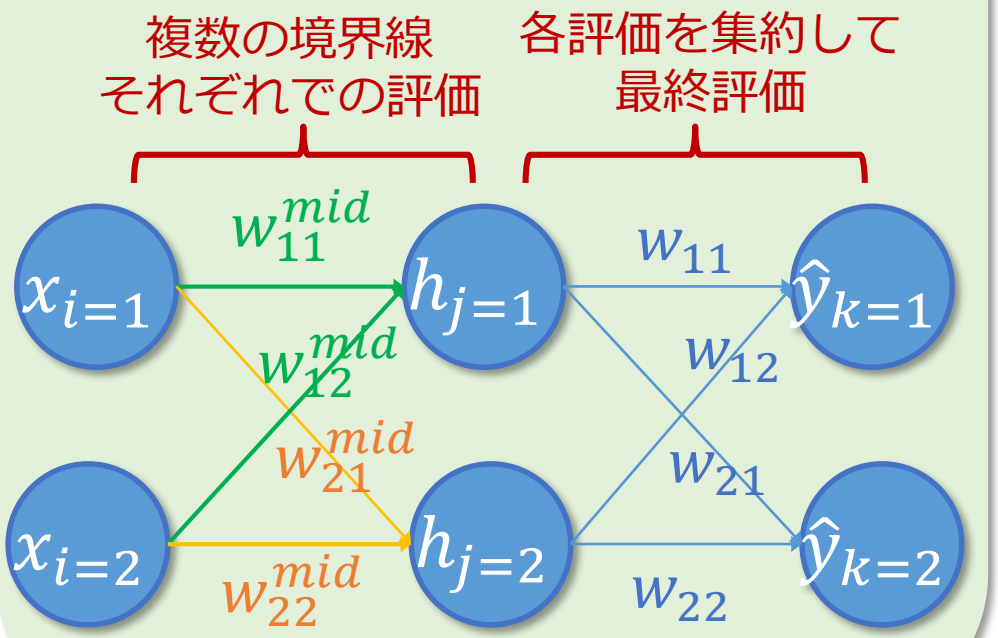
ロジスティック関数



出力層のノード数を増やせば
多クラス分類が可能



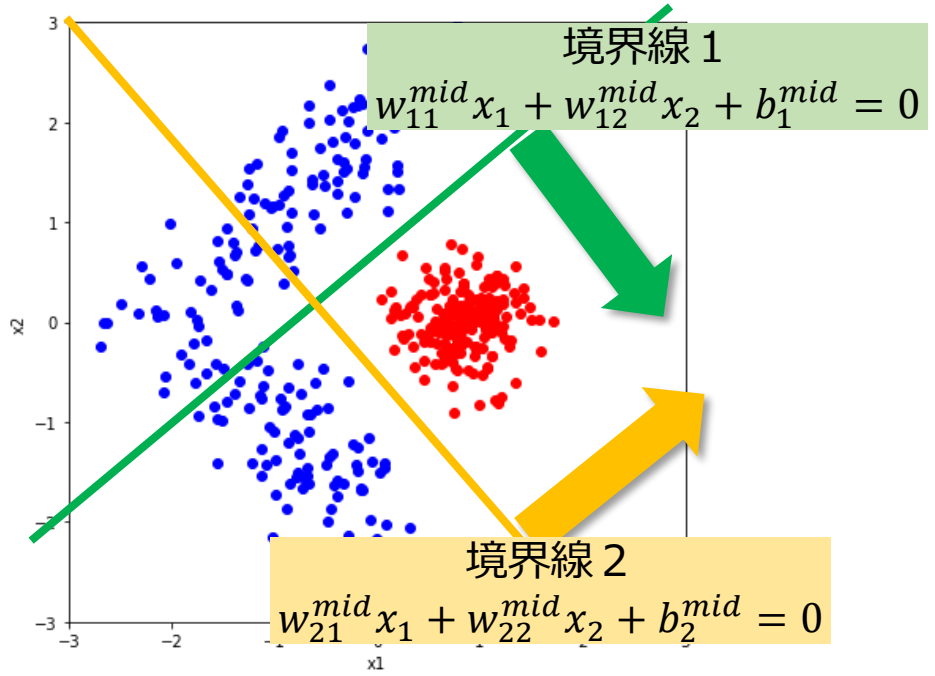
中間層を追加すれば
非線形な分離が可能

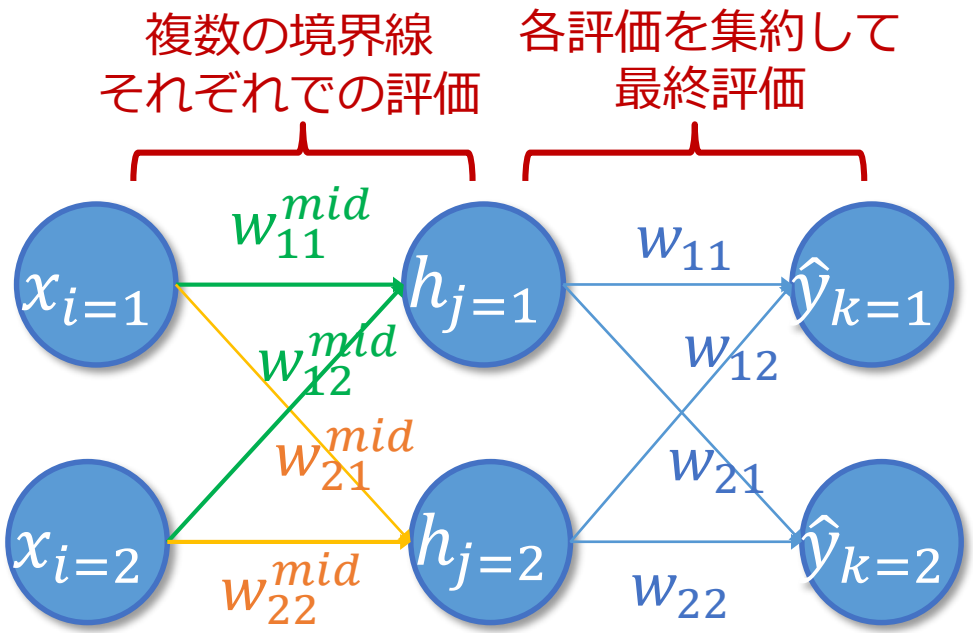


ニューラルネットワークの実装を確認

11_01_neural_network.ipynb を動かしながら，ニューラルネットワークの実装を確認しましょう。

おさらい：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

$$\hat{y}_k = f(z_k) = softmax(z_k)$$

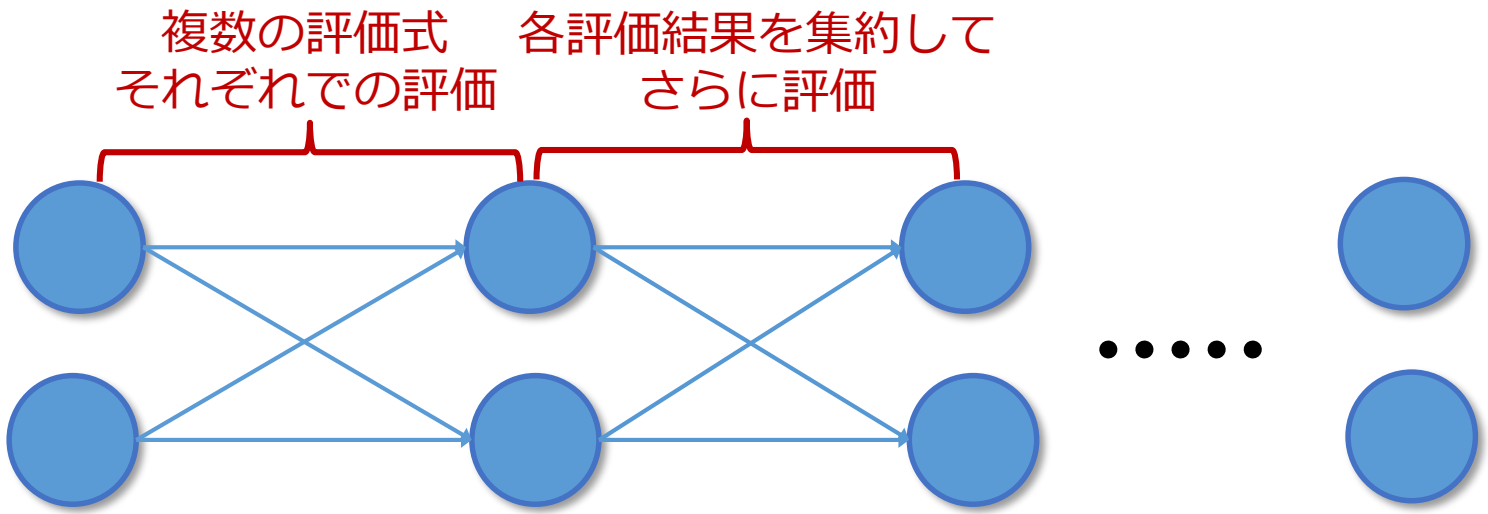
最終結果を出力する

深層ニューラルネットワーク

中間層を1層追加することで，複数の評価式による条件付けの効果が得られ，非線形な分離（区分的線形分離）が行えます。

さらに中間層を追加すると，条件付けの結果をさらに評価することになるため，より複雑なモデルが作成できるはずです。

中間層を多く備えたニューラルネットワークを「**深層ニューラルネットワーク**」と呼びます。



･･･でも層が増えるとパラメータも増えるから，学習が複雑で実装が大変なんじゃないの？

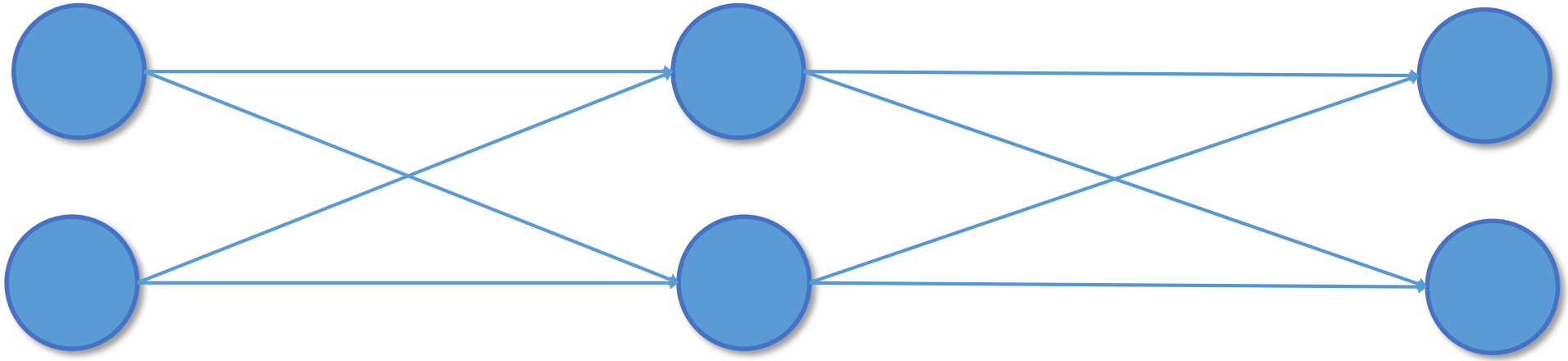
➡ **誤差逆伝播法**を使うことで，複雑なニューラルネットワークも簡単に学習を実装できます。

誤差逆伝播法による
ニューラルネットワークの学習

合成関数の偏微分：連鎖律

ニューラルネットワークは線形関数と非線形関数の計算を交互に行うので，
複数の関数の合成関数で表現されることになります。

勾配を求めるためには，合成関数の偏微分を理解する必要があります。



$$\mathbf{x} \rightarrow \mathbf{g^1} = f^1_{\text{線形}}(\mathbf{x}) \rightarrow \boldsymbol{h^1} = f^1_{\text{非線形}}(\mathbf{g^1}) \rightarrow \mathbf{g^2} = f^2_{\text{線形}}(\boldsymbol{h^1}) \rightarrow \boldsymbol{h^2} = f^2_{\text{非線形}}(\mathbf{g^2})$$

$$\boldsymbol{h^2} = f^2_{\text{非線形}}\left(f^2_{\text{線形}}\left(f^1_{\text{非線形}}\left(f^1_{\text{線形}}(\mathbf{x})\right)\right)\right)$$

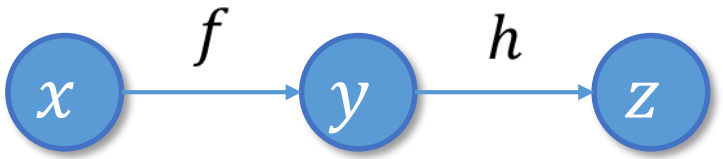
合成関数の偏微分：連鎖律

合成関数を偏微分する際のルールとして，**連鎖律** があります。

連鎖律１（基本形）

$y = f(x)$，$z = h(y)$としたとき，$\partial z/\partial x$ は以下のように計算できる。

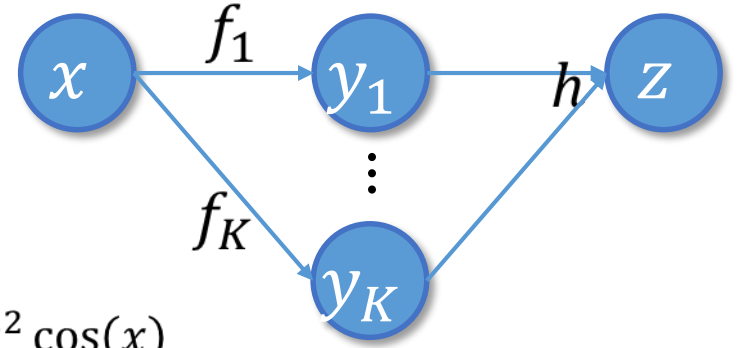
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}$$



連鎖律２（多変数関数の場合）

$y_1 = f_1(x), y_2 = f_2(x), \ldots, y_K = f_K(x)$，$z = h(y_1, y_2, \ldots, y_K)$としたとき，
$\partial z/\partial x$ は以下のように計算できる。

$$\frac{\partial z}{\partial x} = \sum_{k=1}^{K} \frac{\partial z}{\partial y_k}\frac{\partial y_k}{\partial x}$$



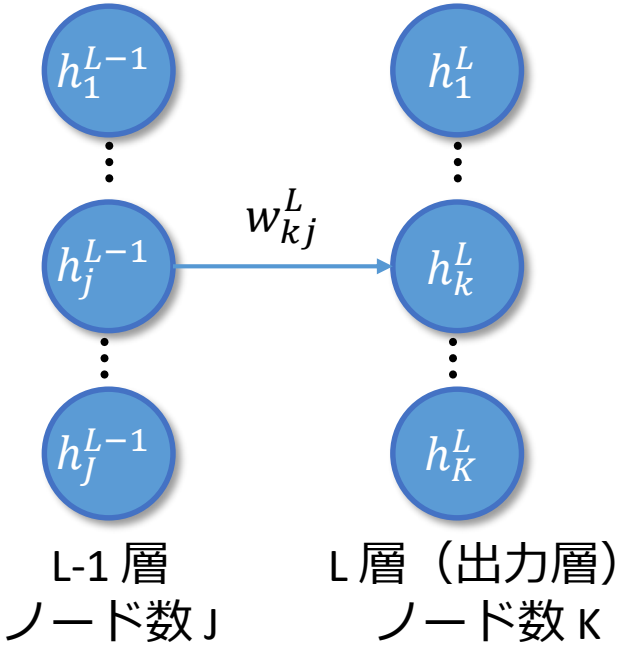
例 ： $\frac{\partial(x^2 \sin(x))}{\partial x} = \frac{\partial(x^2 \sin(x))}{\partial x^2}\frac{\partial x^2}{\partial x} + \frac{\partial(x^2 \sin(x))}{\partial \sin(x)}\frac{\partial \sin(x)}{\partial x} = 2xsin(x) + x^2 \cos(x)$

連鎖律を使って出力層の重みの勾配を計算

出力層（L 層目）の重み行列を$\mathbf{W}^L$，バイアスを$\mathbf{b}^L$，一つ前の層の出力を$\boldsymbol{h}^{L-1}$としたとき，出力層のk 番目のノードの出力 h_k^L は以下の式で計算されます。

$$g_k^L = \sum_{j=1}^{J} w_{kj}^L h_j^{L-1} + b_k^L$$

$$h_k^L = softmax\left(g_k^L\right) = \frac{e^{g_k^L}}{\sum_c e^{g_c^L}}$$



また，損失関数にはクロスエントロピーを使用します。

$$L_{ce} = -\sum_k y_k \log h_k^L$$

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）

連鎖律を使って出力層の重みの勾配を計算

連鎖律 1 を使って勾配を導出します。

$$g_k^L = \sum_{j=1}^{J} w_{kj}^L h_j^{L-1} + b_k^L$$
$$h_k^L = softmax(g_k^L) = \frac{e^{g_k^L}}{\sum_c e^{g_c^L}}$$
$$L_{ce} = -\sum_k y_k \log h_k^L$$

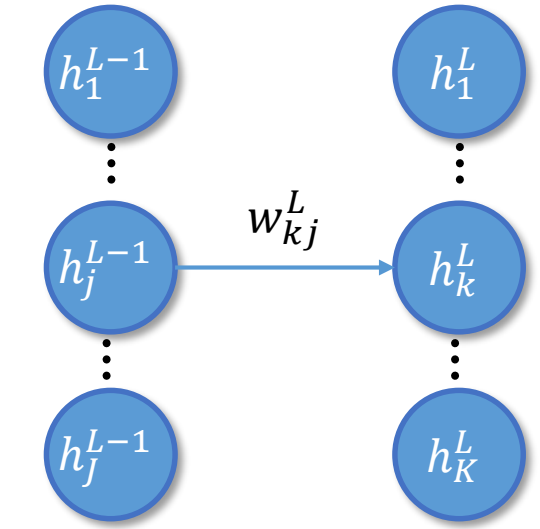
$$\frac{\partial L}{\partial w_{kj}^L} = \frac{\partial L_{ce}}{\partial g_k^L}\frac{\partial g_k^L}{\partial w_{kj}^L} = (h_k^L - y_k)h_j^{L-1}$$

$$\frac{\partial L}{\partial b_k^L} = \frac{\partial L_{ce}}{\partial g_k^L}\frac{\partial g_k^L}{\partial b_k^L} = (h_k^L - y_k)$$

$$\frac{\partial L_{ce}}{\partial g_k^L} = \frac{\partial}{\partial g_k^L}\left[-\sum_k y_k \log h_k^L\right] = \frac{\partial}{\partial g_k^L}\left[-\sum_k y_k \log \frac{e^{g_k^L}}{\sum_c e^{g_c^L}}\right]$$
$$= \frac{\partial}{\partial g_k^L}\left[-\sum_k y_k g_k^L + \sum_k y_k \log \sum_c e^{g_c^L}\right] = \left(-y_k + \underbrace{\frac{e^{g_k^L}}{\sum_c e^{g_c^L}}}_{=h_k^L}\underbrace{\sum_k y_k}_{=1}\right)$$
$$= -y_k + h_k^L$$

$$\frac{\partial g_k^L}{\partial w_{kj}^L} = \frac{\partial \sum_{j=1}^{J} w_{kj}^L h_j^{L-1} + b_k^L}{\partial w_{kj}^L} = h_j^{L-1}$$

$$\frac{\partial g_k^L}{\partial b_k^L} = \frac{\partial \sum_{j=1}^{J} w_{kj}^L h_j^{L-1} + b_k^L}{\partial w_{kj}^L} = 1$$



L-1 層
ノード数 J

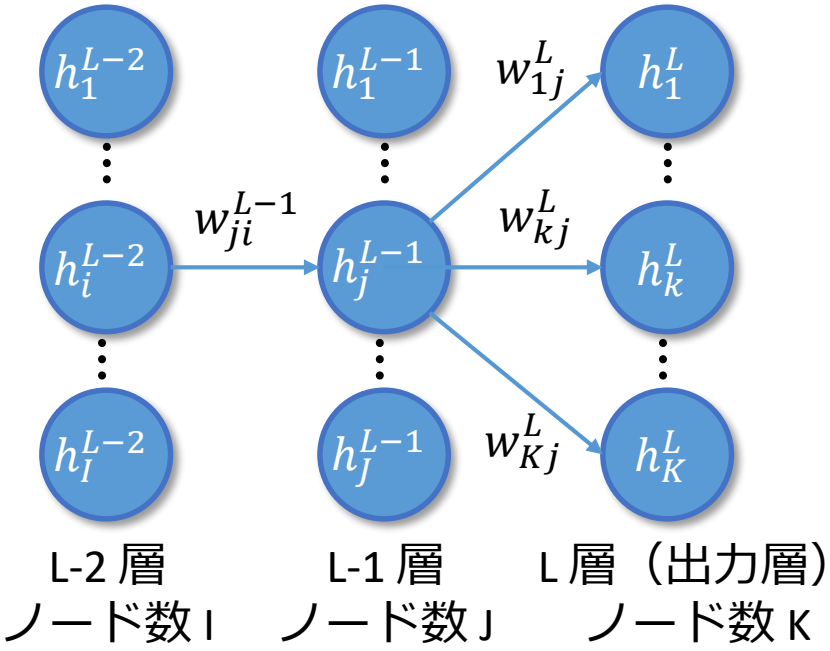
L 層（出力層）
ノード数 K

連鎖律を使って中間層の重みの勾配を計算

$L-1$ 層目の重み行列を$\mathbf{W}^{L-1}$，バイアスを$\mathbf{b}^{L-1}$，一つ前の層の出力を$\boldsymbol{h}^{L-2}$
としたとき，$L-1$層のj 番目のノードの出力 h_j^{L-1} は以下の式で計算されます。

$$g_j^{L-1} = \sum_{i=1}^{I} w_{ji}^{L-1} h_i^{L-2} + b_j^{L-1}$$

$$h_j^{L-1} = sigmoid\left(g_j^{L-1}\right) = \frac{1}{1+e^{-g_j^{L-1}}}$$



出力層のときと同様に，連鎖律１を使って展開します。

$$\frac{\partial L_{ce}}{\partial w_{ji}^{L-1}} = \frac{\partial L_{ce}}{\partial g_j^{L-1}} \frac{\partial g_j^{L-1}}{\partial w_{ji}^{L-1}} \qquad \frac{\partial L_{ce}}{\partial b_j^{L-1}} = \frac{\partial L_{ce}}{\partial g_j^{L-1}} \frac{\partial g_j^{L-1}}{\partial b_j^{L-1}}$$

連鎖律を使って中間層の重みの勾配を計算

$$\frac{\partial L_{ce}}{\partial w_{ji}^{L-1}} = \frac{\partial L_{ce}}{\partial g_j^{L-1}} \frac{\partial g_j^{L-1}}{\partial w_{ji}^{L-1}} \qquad \frac{\partial L_{ce}}{\partial b_j^{L-1}} = \frac{\partial L_{ce}}{\partial g_j^{L-1}} \frac{\partial g_j^{L-1}}{\partial b_j^{L-1}}$$

ここで，g_j^{L-1}の値は出力層 $\mathbf{W}^L$ を通って K 個のノードに伝播しているため，

$\frac{\partial L_{ce}}{\partial g_j^{L-1}}$ は連鎖律２（多変数関数）を使って展開します。

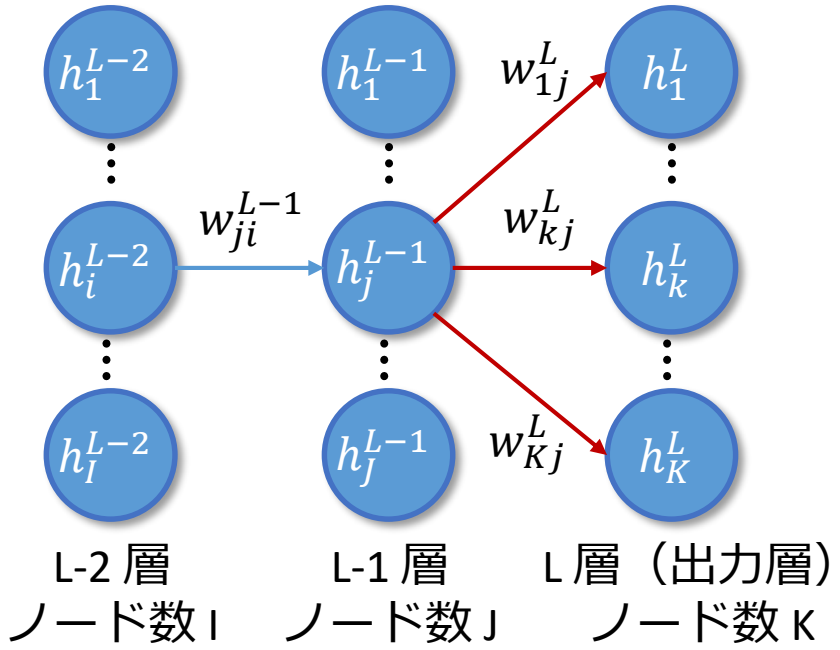
$$\frac{\partial L_{ce}}{\partial g_j^{L-1}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_k^L} \frac{\partial g_k^L}{\partial h_j^{L-1}} \right) \frac{\partial h_j^{L-1}}{\partial g_j^{L-1}}$$

$$\frac{\partial g_k^L}{\partial h_j^{L-1}} = \frac{\partial \sum_{j=1}^{J} w_{kj}^L h_j^L + b_k^L}{\partial h_j^{L-1}} = w_{kj}^L$$

$$\frac{\partial h_j^{L-1}}{\partial g_j^{L-1}} = \frac{\partial sigmoid\left(g_j^{L-1}\right)}{\partial g_j^{L-1}} = \left(1 - h_j^{L-1}\right) h_j^{L-1}$$

よって

$$\frac{\partial L_{ce}}{\partial g_j^{L-1}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_k^L} w_{kj}^L \right) \left(1 - h_j^{L-1}\right) h_j^{L-1}$$



連鎖律を使って中間層の重みの勾配を計算

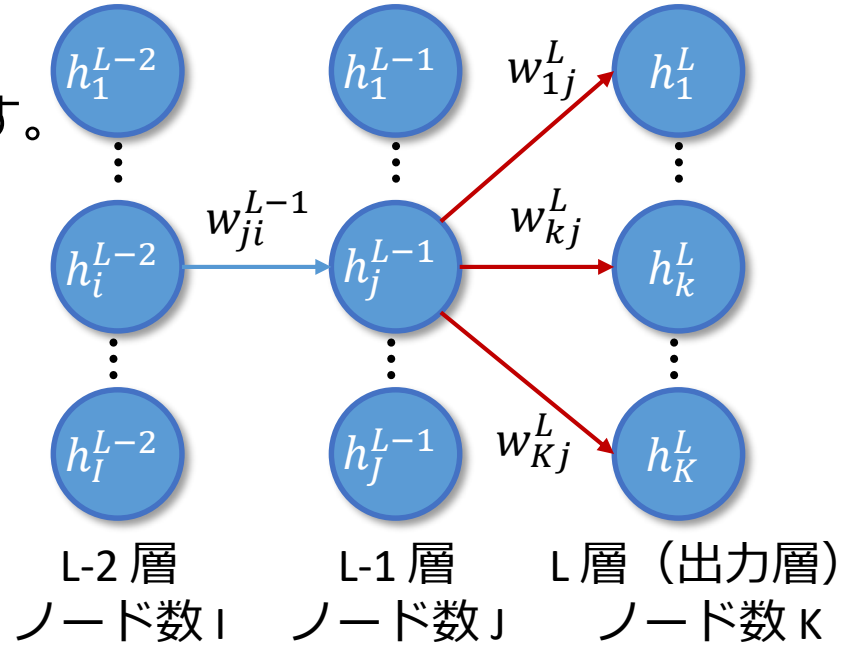
$$\frac{\partial L_{ce}}{\partial g_j^{L-1}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_k^L} w_{kj}^L \right) (1 - h_j^{L-1}) h_j^{L-1}$$

ここで，$\frac{\partial L_{ce}}{\partial g_k^L}$ は p.12 で求めた $(h_k^L - y_k)$ です。つまり**計算済みの項**です。

以上をまとめると，$L-1$ 層目の勾配は以下のようになります。

$$\frac{\partial L_{ce}}{\partial w_{ji}^{L-1}} = \frac{\partial L_{ce}}{\partial g_j^{L-1}} \frac{\partial g_j^{L-1}}{\partial w_{ji}^{L-1}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_k^L} w_{kj}^L \right) (1 - h_j^{L-1}) h_j^{L-1} h_i^{L-2}$$

$$\frac{\partial L_{ce}}{\partial b_j^{L-1}} = \frac{\partial L_{ce}}{\partial g_j^{L-1}} \frac{\partial g_j^{L-1}}{\partial b_j^{L-1}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_k^L} w_{kj}^L \right) (1 - h_j^{L-1}) h_j^{L-1}$$



連鎖律を使って中間層の重みの勾配を計算

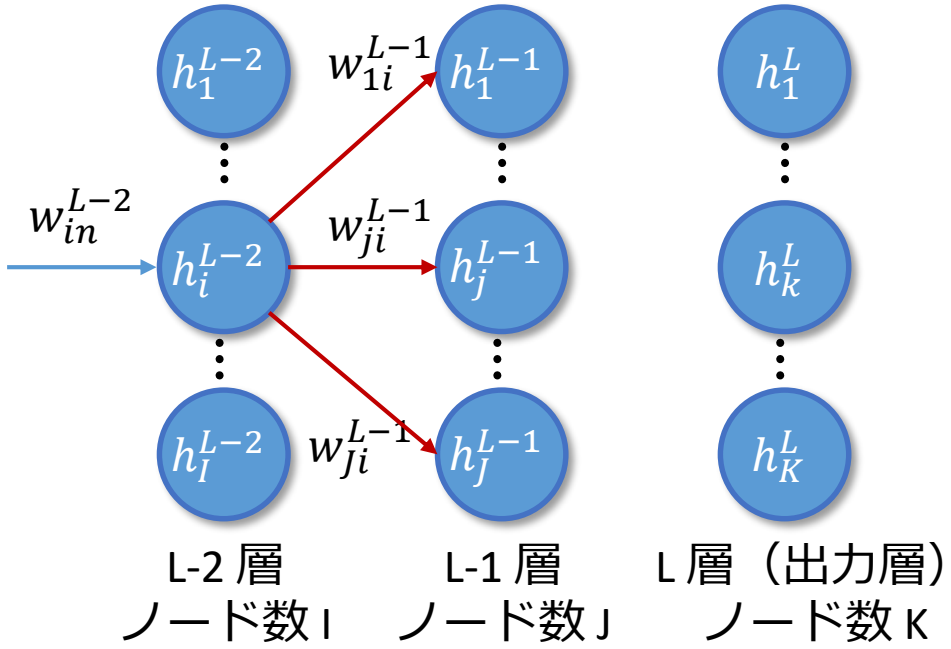
$L-2$ 層目の勾配は以下のようになります。

$$\frac{\partial L_{ce}}{\partial w_{in}^{L-2}} = \frac{\partial L_{ce}}{\partial g_i^{L-2}} \frac{\partial g_i^{L-2}}{\partial w_{in}^{L-2}}$$

$$= \sum_{j=1}^{J} \left(\frac{\partial L_{ce}}{\partial g_j^{L-1}} w_{ji}^{L-1} \right) \left(1 - h_i^{L-2}\right) h_i^{L-2} h_n^{L-3}$$

$$\frac{\partial L_{ce}}{\partial b_i^{L-2}} = \frac{\partial L_{ce}}{\partial g_i^{L-2}} \frac{\partial g_i^{L-2}}{\partial b_i^{L-2}}$$

$$= \sum_{j=1}^{J} \left(\frac{\partial L_{ce}}{\partial g_j^{L-1}} w_{ji}^{L-1} \right) \left(1 - h_i^{L-2}\right) h_i^{L-2}$$



$\frac{\partial L_{ce}}{\partial g_j^{L-1}}$ は 前ページで**計算済みの項**です。

誤差逆伝播法

以上の式をまとめると，以下のようになります。
（損失関数にクロスエントロピー，出力層の活性化関数にソフトマックス関数，
中間層の活性化関数にシグモイド関数を用いている場合）

出力層（L層）の勾配

$$\frac{\partial L}{\partial w_{kj}^{L}} = \frac{\partial L_{ce}}{\partial g_{k}^{L}} \frac{\partial g_{k}^{L}}{\partial w_{kj}^{L}} = \left(h_{k}^{L} - y_{k}\right) h_{j}^{L-1}$$

$$\frac{\partial L}{\partial b_{k}^{L}} = \frac{\partial L_{ce}}{\partial g_{k}^{L}} \frac{\partial g_{k}^{L}}{\partial b_{k}^{L}} = \left(h_{k}^{L} - y_{k}\right)$$

中間層（l層）の勾配

$$\frac{\partial L_{ce}}{\partial w_{ji}^{l}} = \frac{\partial L_{ce}}{\partial g_{j}^{l}} \frac{\partial g_{j}^{l}}{\partial w_{ji}^{l}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_{k}^{l+1}} w_{kj}^{l+1} \right) \left(1 - h_{j}^{l}\right) h_{j}^{l} h_{i}^{l-1}$$

$$\frac{\partial L_{ce}}{\partial b_{j}^{l}} = \frac{\partial L_{ce}}{\partial g_{j}^{l}} \frac{\partial g_{j}^{l}}{\partial b_{j}^{l}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_{k}^{l+1}} w_{kj}^{l+1} \right) \left(1 - h_{j}^{l}\right) h_{j}^{l}$$

誤差逆伝播法

$\frac{\partial L_{ce}}{\partial g_j^{l+1}}$ は一つ後ろの層で計算済みの項なので，後ろの層から順番に勾配を計算すれば，その計算結果を前の層の勾配計算時に使いまわすことができます。これを**誤差逆伝播法**と呼びます。

出力層

$$\frac{\partial L_{ce}}{\partial w_{kj}^L} = \frac{\partial L_{ce}}{\partial g_k^L}\frac{\partial g_k^L}{\partial w_{kj}^L} = \left(h_k^L - y_k\right)h_j^{L-1}$$

L-1 層

$$\frac{\partial L_{ce}}{\partial w_{ji}^{L-1}} = \frac{\partial L_{ce}}{\partial g_j^{L-1}}\frac{\partial g_j^{L-1}}{\partial w_{ji}^{L-1}} = \sum_{k=1}^{K}\left(\frac{\partial L_{ce}}{\partial g_k^L}w_{kj}^L\right)\left(1-h_j^{L-1}\right)h_j^{L-1}h_i^{L-2}$$

L-2 層

$$\frac{\partial L_{ce}}{\partial w_{in}^{L-2}} = \frac{\partial L_{ce}}{\partial g_i^{L-2}}\frac{\partial g_i^{L-2}}{\partial w_{in}^{L-2}} = \sum_{j=1}^{J}\left(\frac{\partial L_{ce}}{\partial g_j^{L-1}}w_{ji}^{L-1}\right)\left(1-h_i^{L-2}\right)h_i^{L-2}h_n^{L-3}$$

⋮

⋮

誤差逆伝播法を使ったニューラルネットワークの学習

11_02_neural_network2.ipynb を使って，誤差逆伝播法を使って任意の層の数を設定可能なニューラルネットワークの実装を確認しましょう。

各層の重み行列やバイアスとその勾配の計算が，forループを使ってシンプルに実装されている点に注目してください.

また，11_03_digit_classification.ipynb を使って，ニューラルネットワークによる手書き文字認識の動作を確認しましょう。

実装が簡単 ≠ 学習が簡単 – 勾配消失問題について

中間層の勾配は，それより後ろの層の勾配が積算されて計算されます。

そのため，後ろの層の勾配が1より小さい場合，1より小さい数が何度も掛けられることで，勾配が0になってしまう**勾配消失問題**が発生します。勾配が0になると，学習が行えません。

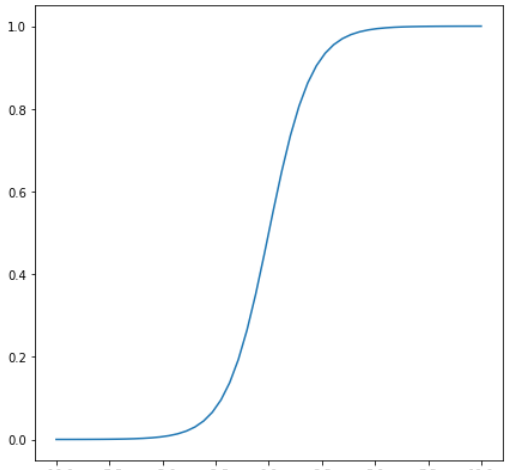
$$\frac{\partial L_{ce}}{\partial w_{ji}^{l}} = \frac{\partial L_{ce}}{\partial g_{j}^{l}} \frac{\partial g_{j}^{l}}{\partial w_{ji}^{l}} = \sum_{k=1}^{K} \left(\frac{\partial L_{ce}}{\partial g_{k}^{l+1}} w_{kj}^{l+1} \right) (1 - h_{j}^{l}) h_{j}^{l} h_{i}^{l-1}$$

例えばシグモイド関数の微分は最大でも0.25と小さい値なので，勾配消失が起きやすいです。

そのため近年では，勾配消失が起こりにくい別の活性化関数が使われたりしています。

シグモイド関数　$h = \frac{1}{1 + e^{-g}}$

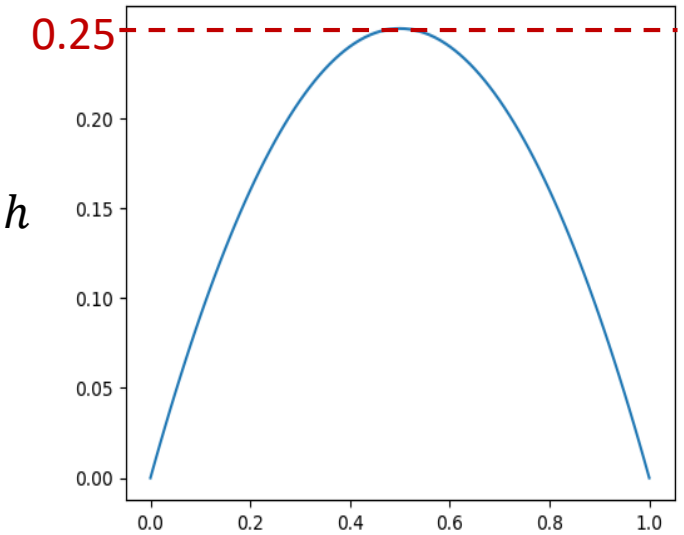
$h = sigmoid(g)$



g

シグモイド関数の微分　$(1 - h)h$

$(1 - h)h$

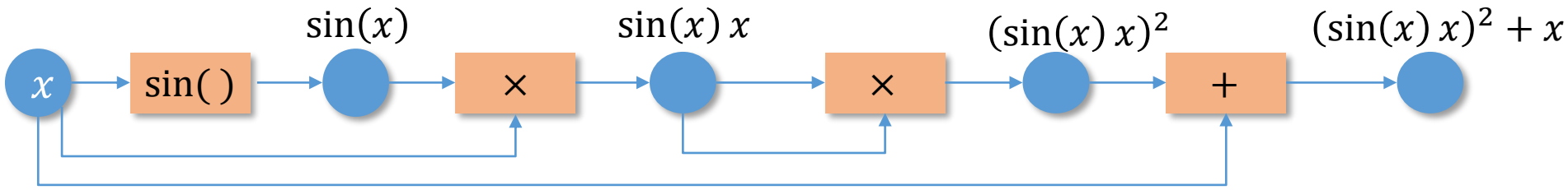


h

自動微分について

ニューラルネットワークに限らず，大抵の計算は合成関数で表現できます。

例：$(\sin(x)\,x)^2 + x$



このような式も，sin関数と掛け算，足し算の微分値だけ定義しておけば，
誤差逆伝播の要領で自動的に微分を計算することが可能です。
これを**自動微分**と呼びます。
Pythonには自動微分により簡単に深層ニューラルネットワークを構築できる
ライブラリが存在します(Pytorch，Tensorflowなど)。
これにより，深層ニューラルネットワークが手軽に扱えるようになったことが，
近年の人工知能の発展の大きな要因の一つになっています。

おわりに

今回は誤差逆伝播法について解説しました。

深層ニューラルネットワークが流行った要因として，もちろん性能が高いこともありますが，誤差逆伝播法＋自動微分のおかげで，どんな複雑なモデルでも，容易に実装・学習できるという点も大きいです。

そのため，誤差逆伝播法は人工知能分野における大発明と言っても良いでしょう。

といっても，誤差逆伝播法だけで深層ニューラルネットワークが学習できるようになったわけではなく，勾配消失問題の回避方法など，本講義では紹介しきれない沢山のテクニックの積み重ねによって実現されています。

（誤差逆伝播法が提案されたのが1986年で，深層ニューラルネットワークが実現したのが2006年頃です。）

レポート課題

ニューラルネットワークを用いて回帰を行え（識別ではない）。

第１１回ファイル一式に含まれる，“report11.ipynb” に，用いるデータの説明や，ニューラルネットワークを用いた回帰モデルの実装方法について説明しているので，参考にすること。

レポート提出期限：7/19(火) AM10:30，ipynbファイルをhtmlファイルに変換して提出