自然言語処理
―ニューラルネット―

https://satoyoshiharu.github.io/nlp/

100本ノック第7章と第8章の位置づけ

- １００本ノック課題集では、第7章が単語ベクトルで、第8章がニューラルネットとなっています。第7章の課題は、単語ベクトルがあるとしてこれを利用するだけなので、その順序になっています。
- しかし、単語ベクトルは、ニューラルネットを使って構築するものなので、学習する論理的な順序としては逆です。論理的な順序で第8章->第7章と進むことをお勧めします。
- 第7章には、機械学習のトピックであるクラスタリングと次元圧縮が含まれます。そのため、第7章の課題を済ませて機械学習をカバーしてから、第8章のニューラルネットの課題に取り組むのでも、OKです。

以下、ニューラルネットの基本的な部分の概念を説明します。ニューラルネットの入門書一冊に相当します。それをご自分のペースで咀嚼した後で、課題に取り組んでください。

自然言語処理:ニューラルネット
Deep Learning はじめの一歩

解説動画

神経細胞

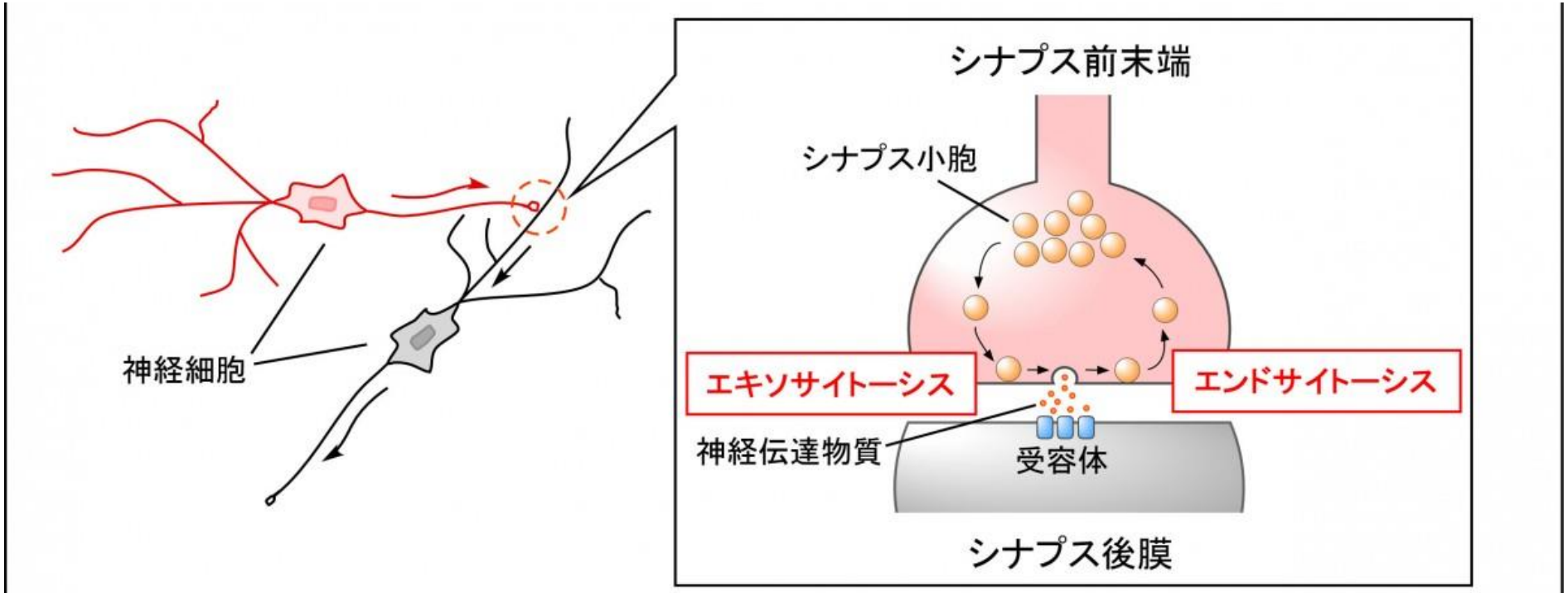
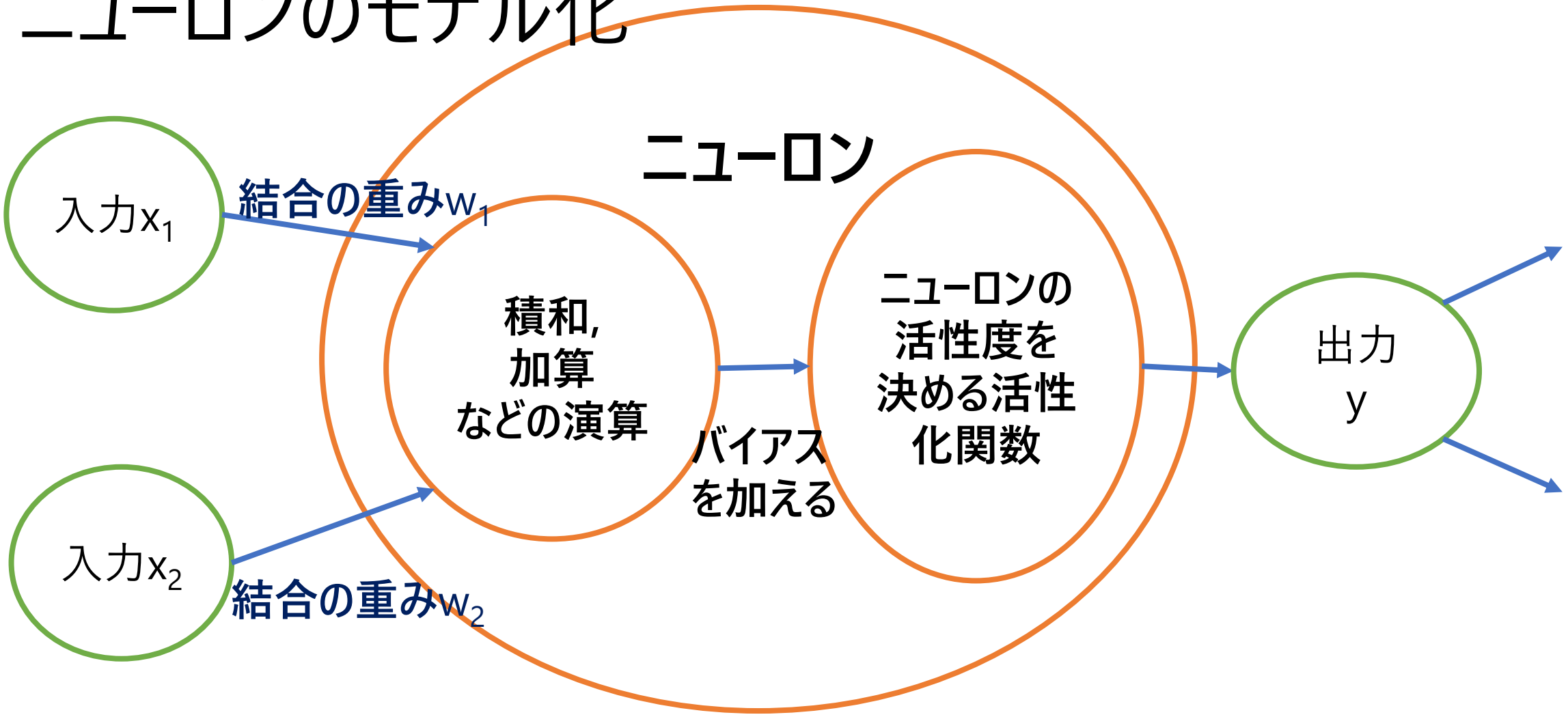

神経細胞
シナプス前末端
シナプス小胞
エキソサイトーシス
エンドサイトーシス
神経伝達物質
受容体
シナプス後膜

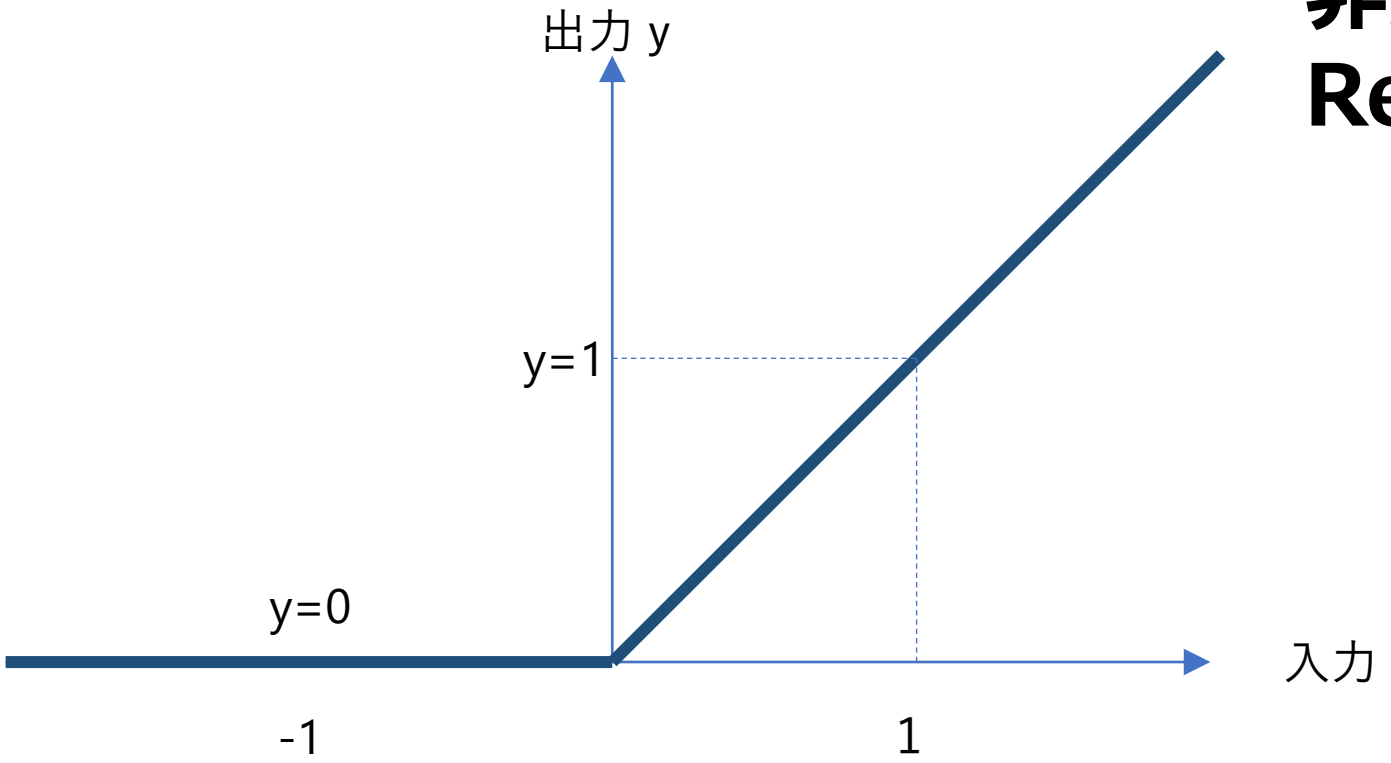
図1

ニューロンのモデル化


入力x_1
結合の重みw_1
ニューロン
積和,
加算
などの演算
バイアス
を加える
ニューロンの
活性度を
決める活性
化関数
出力
y
入力x_2
結合の重みw_2

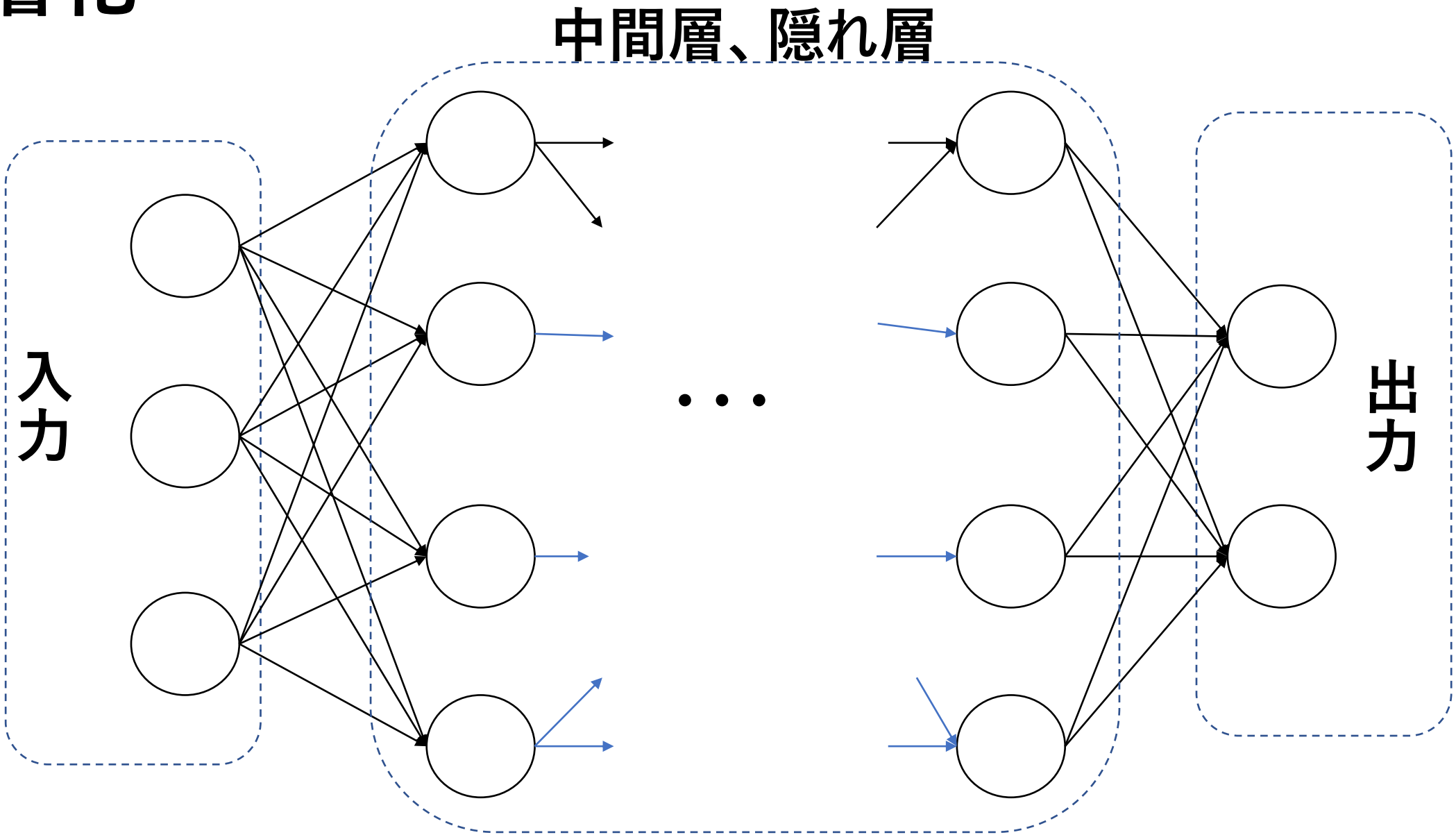
活性化(Activation)関数

非線形変換
ReLU



線形関数とは、f(x+y)=f(x)+f(y), f(ax)=af(x) であるような関数。
ReLUは**非**線形関数: ReLU(-1+1) = 0, ReLU(-1) + ReLU(1) = 0 + 1 = 1

多層化



Deep Learning は、入力に対する**非**線形変換を**何層**も繰り返して、複雑な入出力関係を表現する。

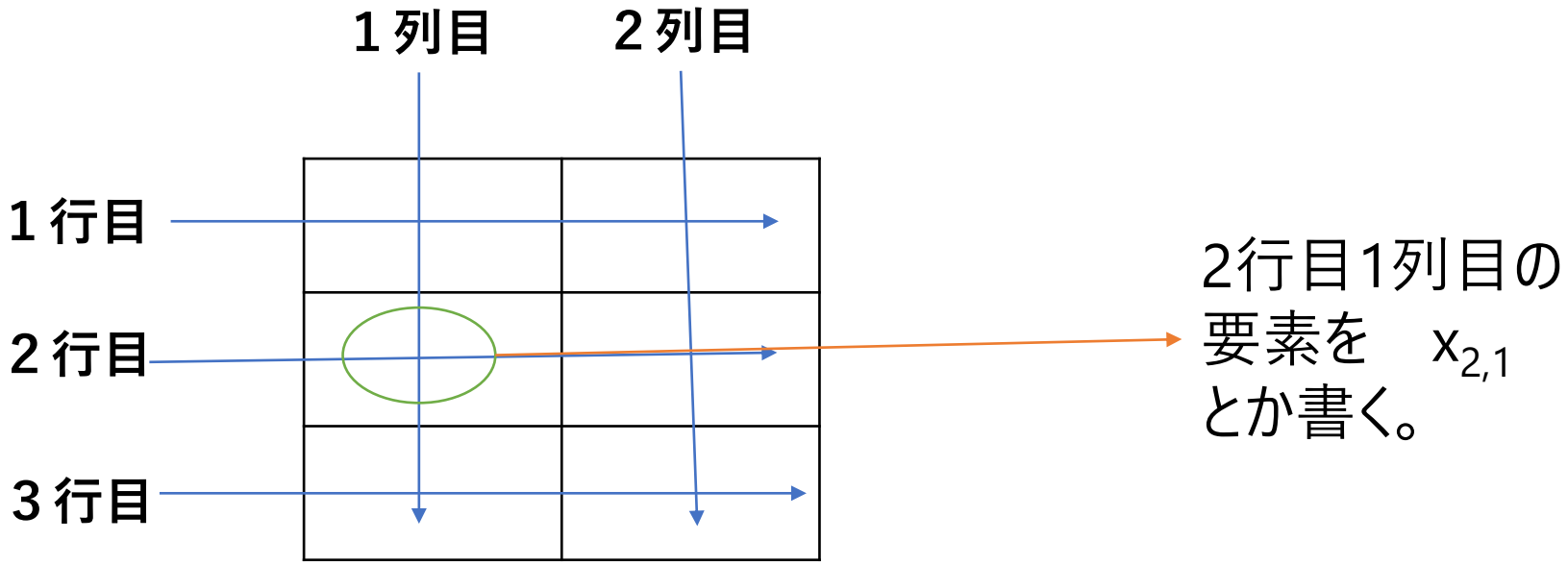
参考

- 「ゼロから作るDeep Learning」by 斉藤こうき、第2章、第3章
- 「Deep　Learningとは」by Sony 小林
- 内職が要らないくらい分かりやすいディープラーニング入門
- 線形変換（一時結合）は、何層重ねても、１個の線形変換と同じ。 -> 【大学数学】線形代数入門③(一次変換と演算の性質)【線形代数】　など
- 関数を重ねる効果 -> 【深層学習】関数 - なぜ「深さ」が AI を生み出しているのか？
- 活性化関数の選び方　-> 「様々な活性化関数」by Sony 小林

自然言語処理：ニューラルネット
全結合層
(別名：Affine層、リニア層)

解説動画

行列



行列の積

$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix} \begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$

行列の積

$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix} \begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$

行列の積

$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix} \begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$

行列の積

$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix} \begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$

行列の積

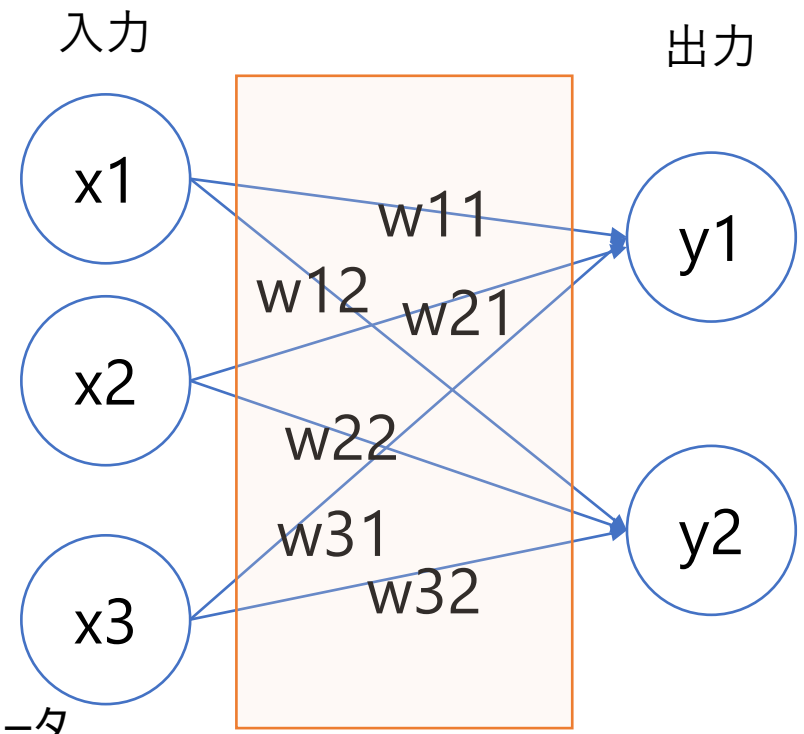
$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix} \begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$

行列の積

$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix} \begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$

行列の積

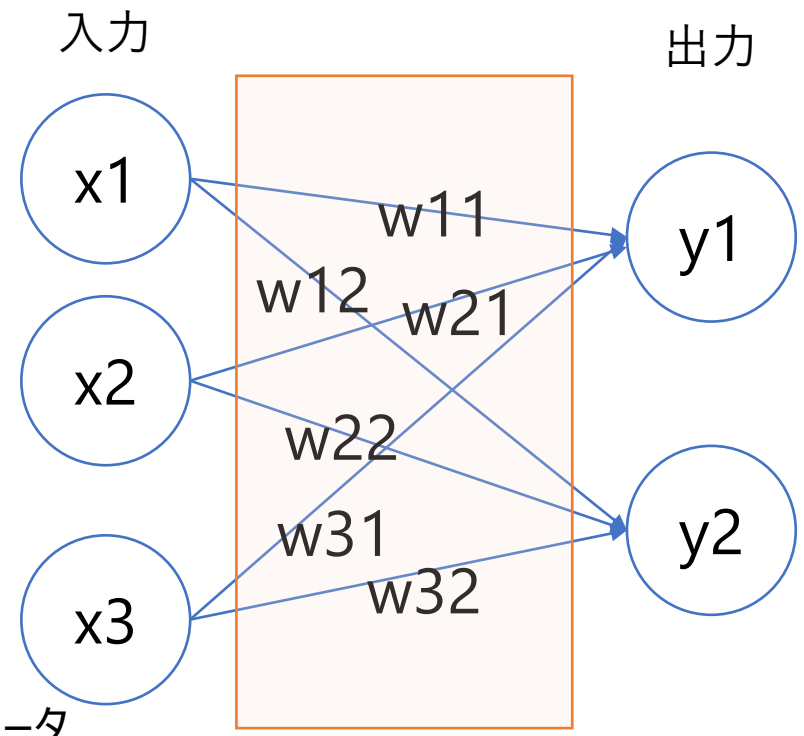
$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix} \begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$



結合重みパラメータ

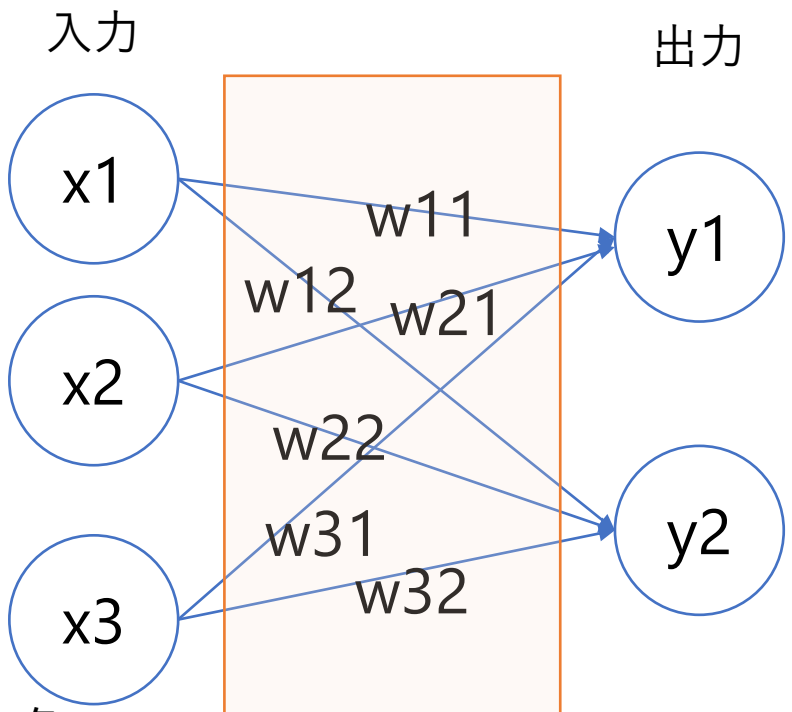
入力 出力 y1 y2

$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = (x1w11 + x2w21 + x3w31, x1w12 + x2w22 + x3w32)$$



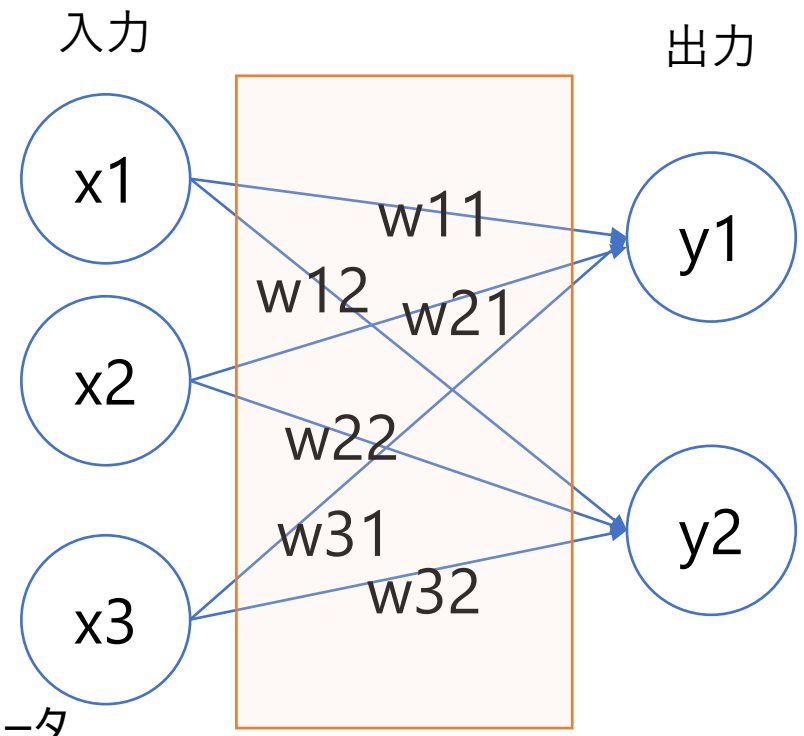
結合重みパラメータ

$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{matrix} y1 \\ (x1w11 + x2w21 + x3w31, \end{matrix} \begin{matrix} y2 \\ x1w12 + x2w22 + x3w32) \end{matrix}$$



結合重みパラメータ

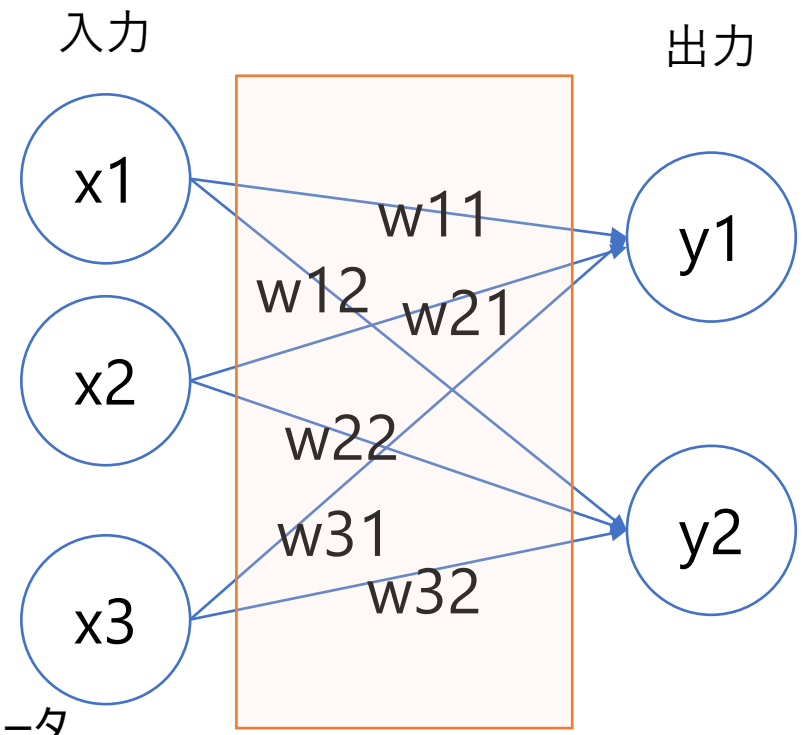
$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = (\overset{y1}{x1w11 + x2w21 + x3w31}, \overset{y2}{x1w12 + x2w22 + x3w32})$$



結合重みパラメータ

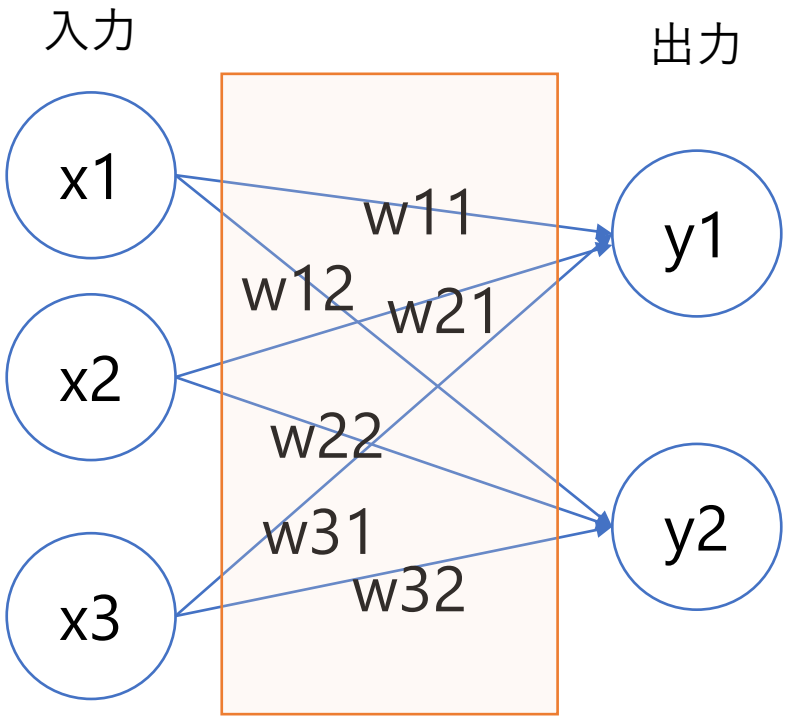
$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = (x1w11 + x2w21 + x3w31,\ x1w12 + x2w22 + x3w32)$$

y1 y2



結合重みパラメータ

$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = (\underset{y1}{x1w11 + x2w21 + x3w31}, \underset{y2}{x1w12 + x2w22 + x3w32})$$



入力は3個

結合重みパラメータ

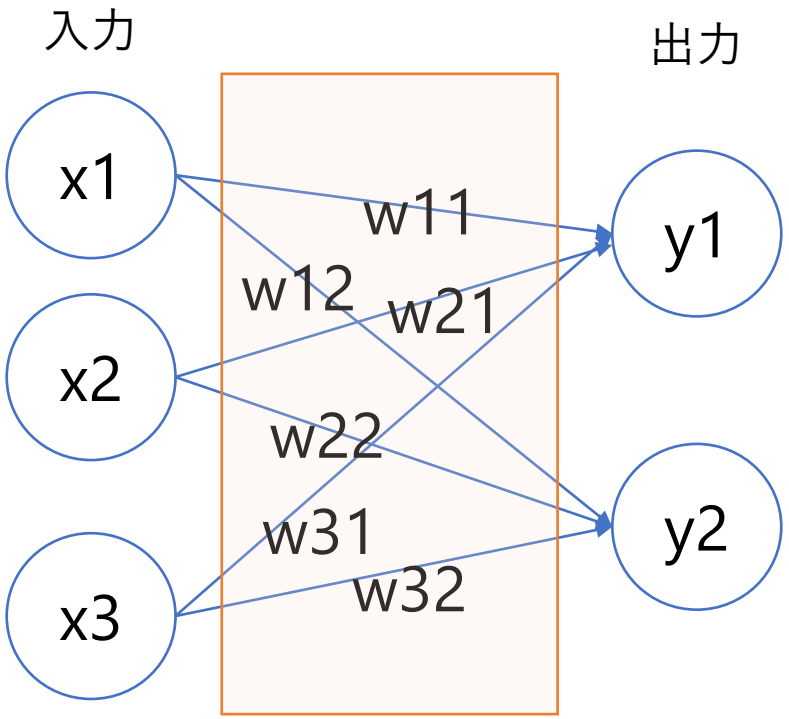
出力は2個

$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = (\underset{y1}{x1w11 + x2w21 + x3w31}, \underset{y2}{x1w12 + x2w22 + x3w32})$$

1行 3列

3行 2列

1行 2列



入力は要素3個のベクトル

出力は要素2個のベクトル

結合重みパラメータ

$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{matrix} y1 & y2 \\ (x1w11 + x2w21 + x3w31,\ x1w12 + x2w22 + x3w32) \end{matrix}$$

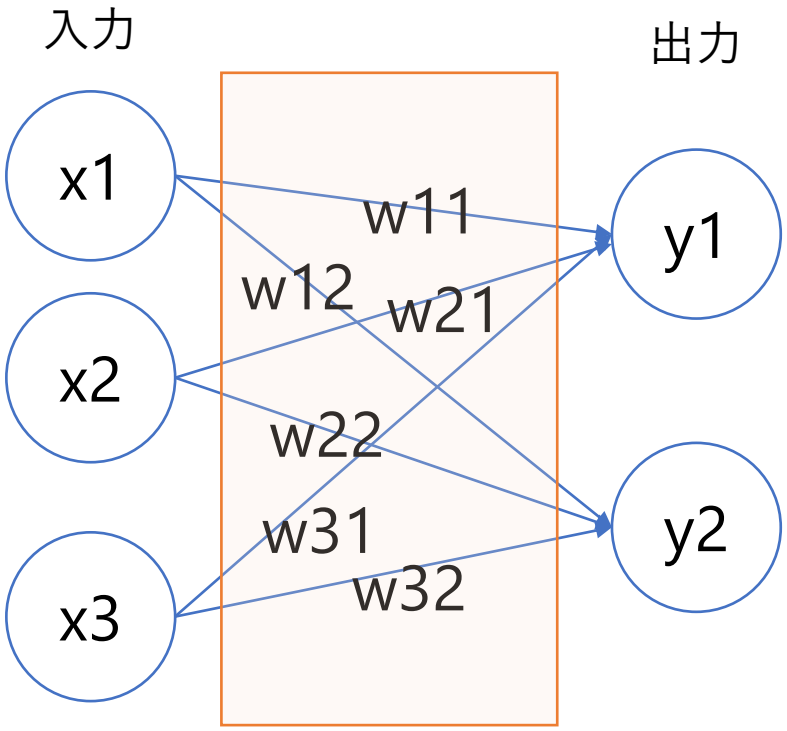
1行 3列

3行 2列

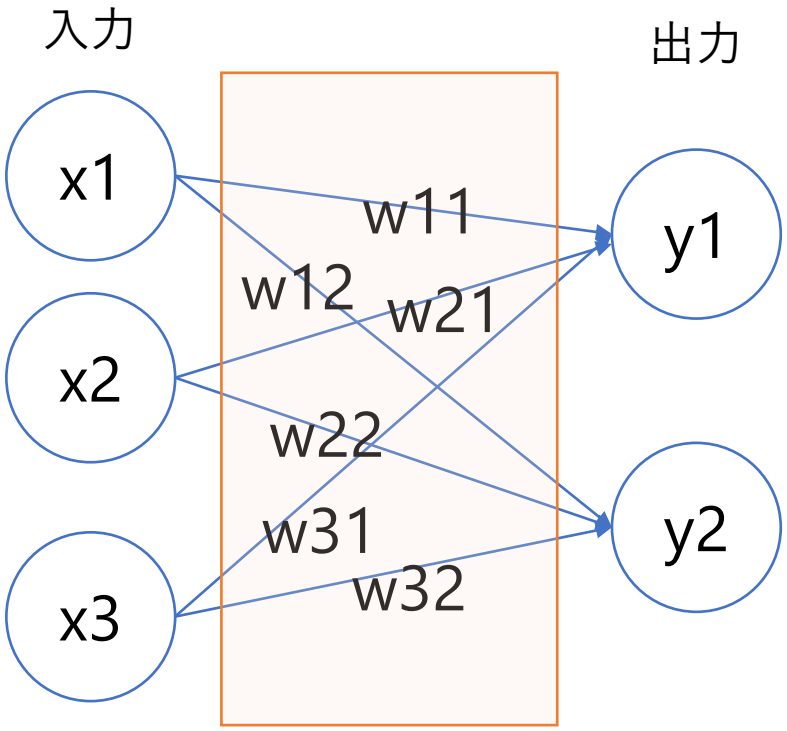
一致

1行 2列

全結合層は、結合の重み行列によって、入力から何らかの特徴を取り出す



結合の重みを、正解が得られるように学習することで、望ましい特徴抽出をさせる



ミニバッチ：Training/Test最小単位

$$(x1, x2, x3)\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = (x1w11 + x2w21 + x3w31, x1w12 + x2w22 + x3w32)$$

データを何個分かまとめる

ミニバッチ
3 個

$$\begin{pmatrix} x11 & x12 & x13 \\ x21 & x22 & x23 \\ x31 & x32 & x33 \end{pmatrix}\begin{pmatrix} w11 & w12 \\ w21 & w22 \\ w31 & w32 \end{pmatrix} = \begin{pmatrix} x11w11 + x12w21 + x13w31 & x11w12 + x12w22 + x13w32 \\ x21w11 + x22w21 + x23w31 & x21w12 + x22w22 + x23w32 \\ x31w11 + x32w21 + x33w31 & x31w12 + x32w22 + x33w32 \end{pmatrix}$$

3個分の
3要素の入力

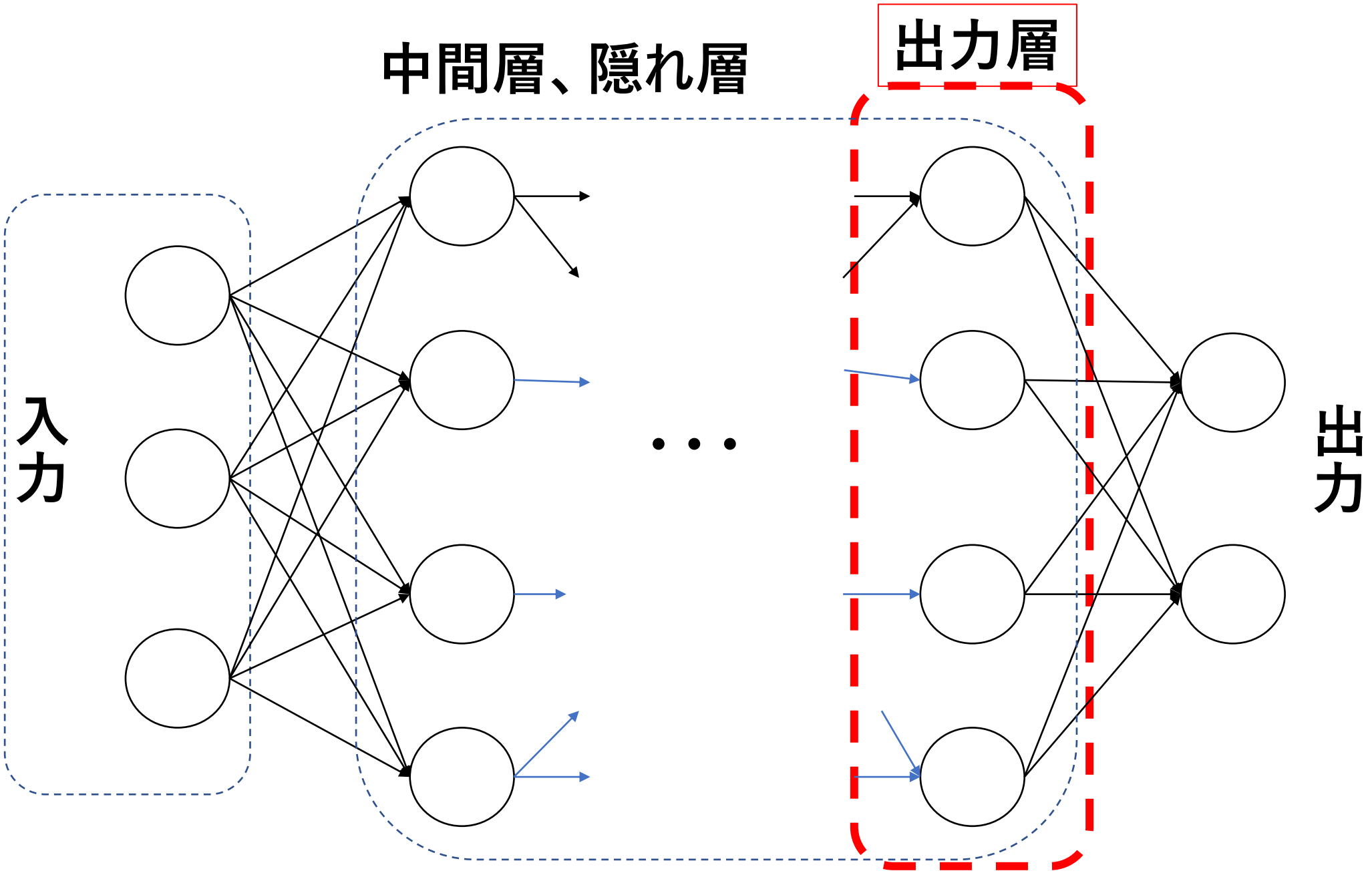
3個を2個へ変換する結合重み

3個分の
2 要素の出力

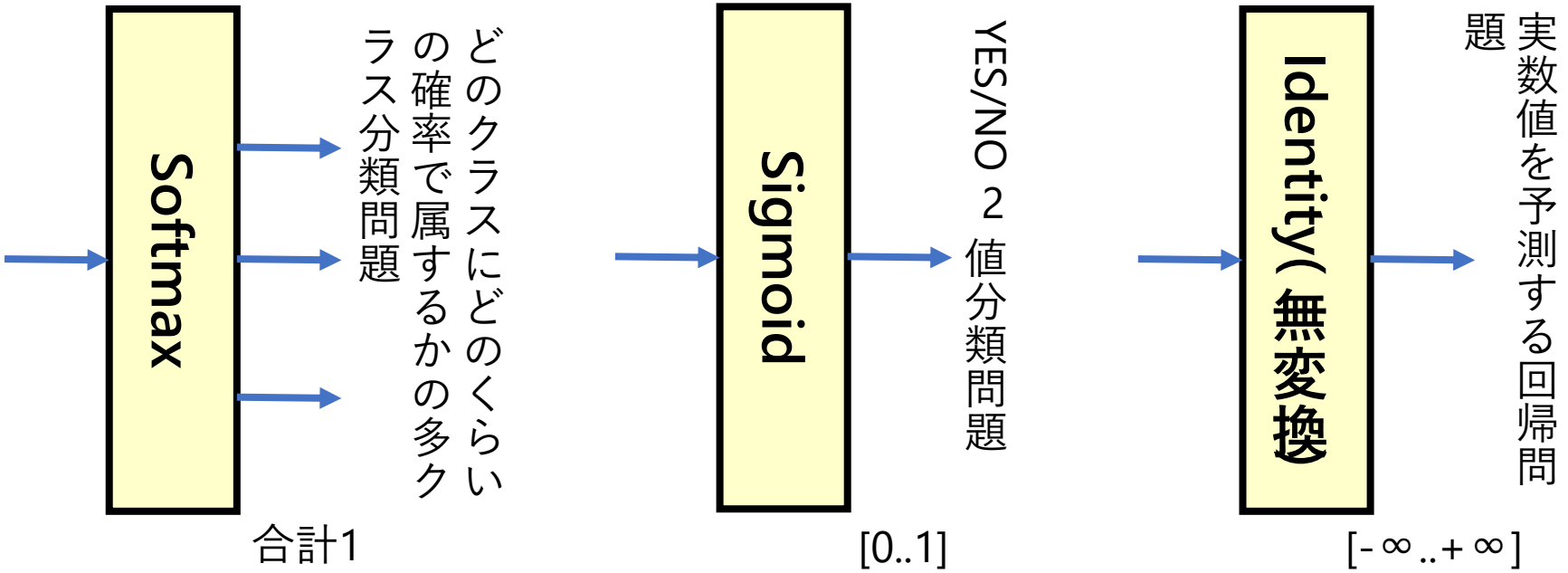
GPUの能力活用

自然言語処理：ニューラルネット
Softmax出力層

解説動画


中間層、隠れ層
出力層
入力
出力

出力層は望ましい最終形に変換する

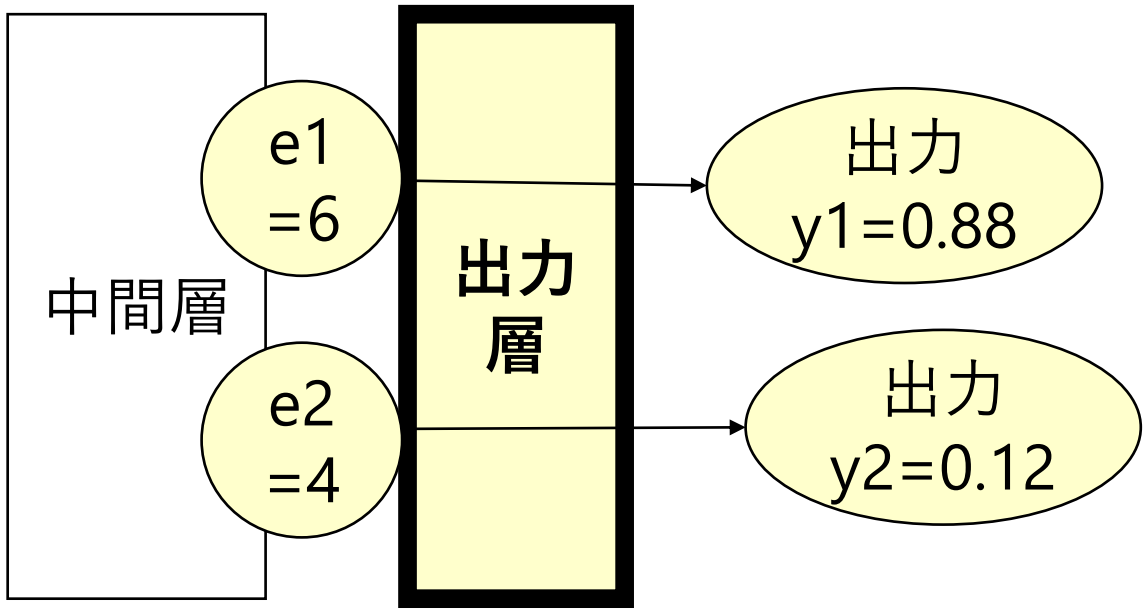


Softmax
合計1になるように変換する

入力(中間層の出力)

$$y_i = \frac{e^i}{\sum_j e^j}$$

出力



出力値を合計値で割れば合計１になるのだが、後の処理のため（逆伝播計算が楽になる）、e の出力値乗にする。

Softmaxの処理

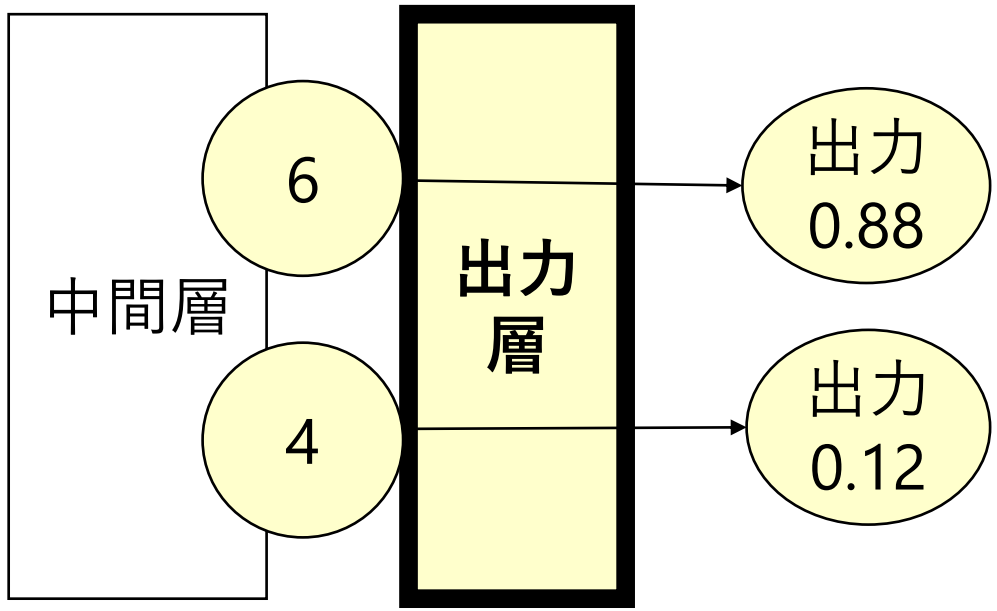
入力

$$y_i = \frac{e^{i}}{\sum_j e^{j}}$$

出力

出力層への入力が 6 と 4 とすると、
$e^6 = 403$
$e^4 = 55$
$e^6+e^4=458$
なので、
$e^6/(e^6+e^4)=0.88$
$e^4/(e^6+e^4)=0.12$
これらの和は 1

合計1になる数値は、あるクラスに属する確率と解釈できる



クラス0に属する確率

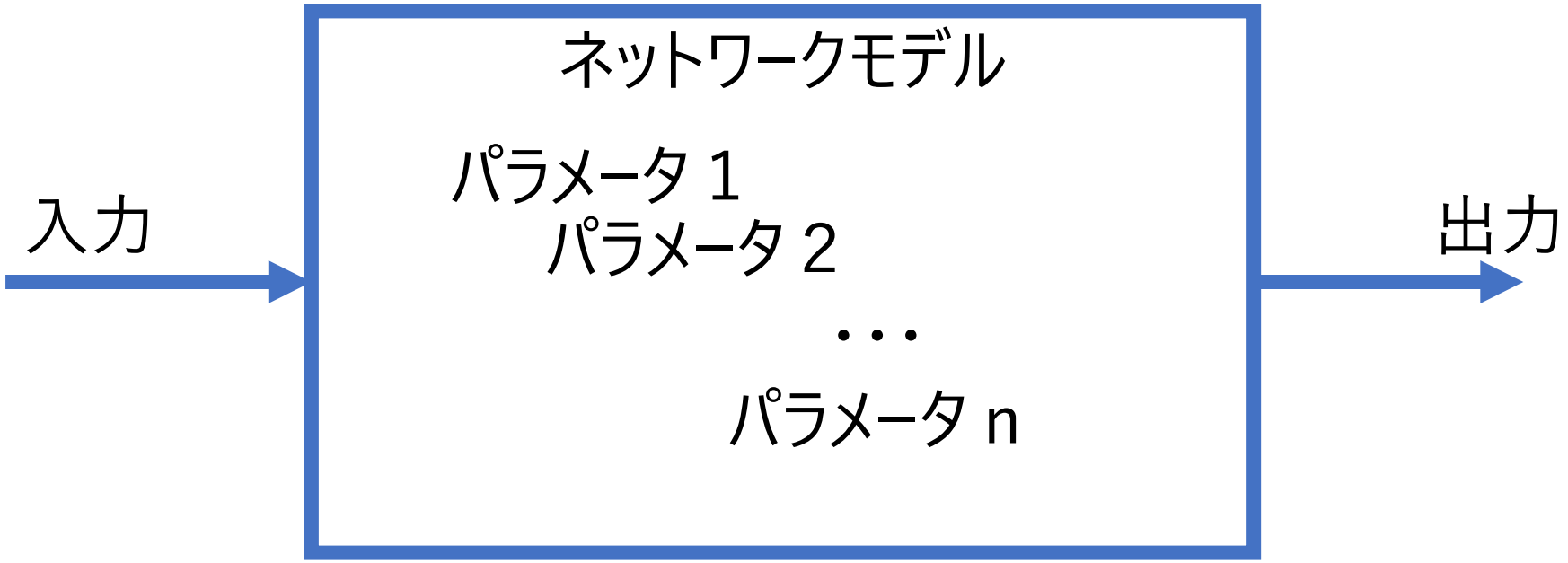
クラス1に属する確率

自然言語処理：ニューラルネット

クロスエントロピー・ロス関数

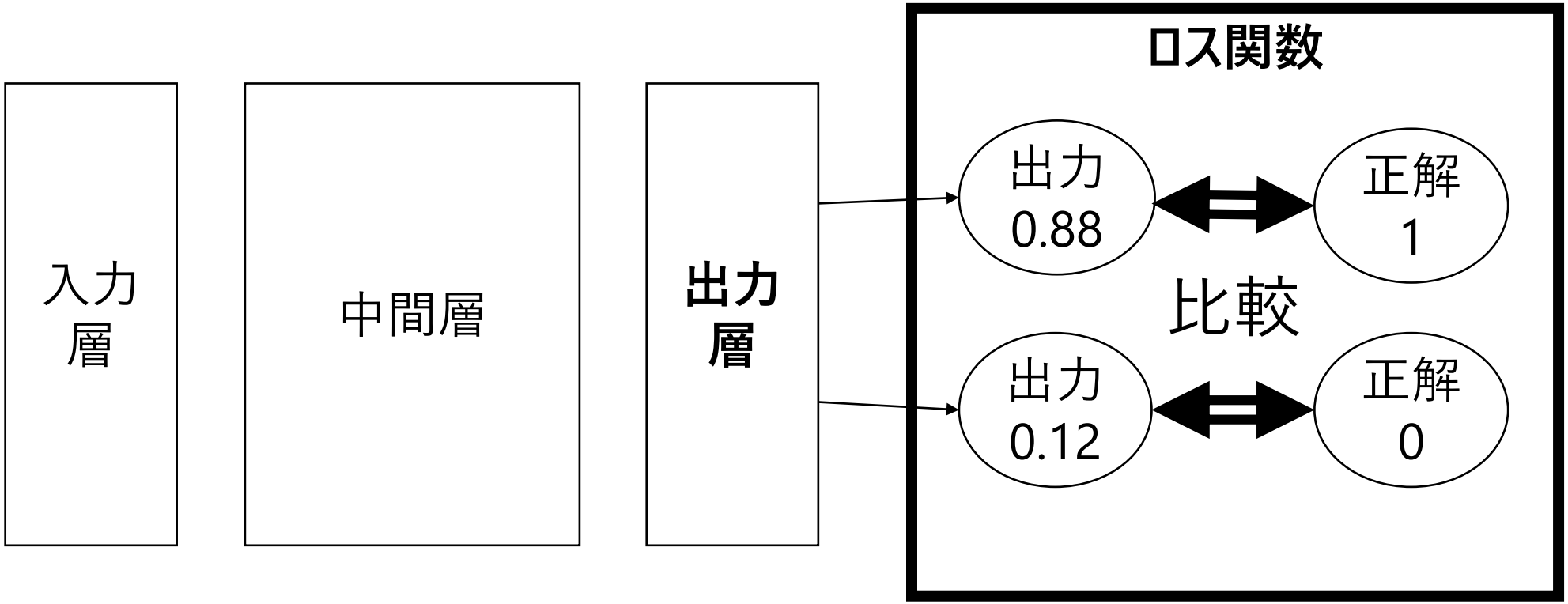
解説動画

ネットワークモデルの学習とは、パラメータ（ニューロン結合の重みなど）を最適な値にすること



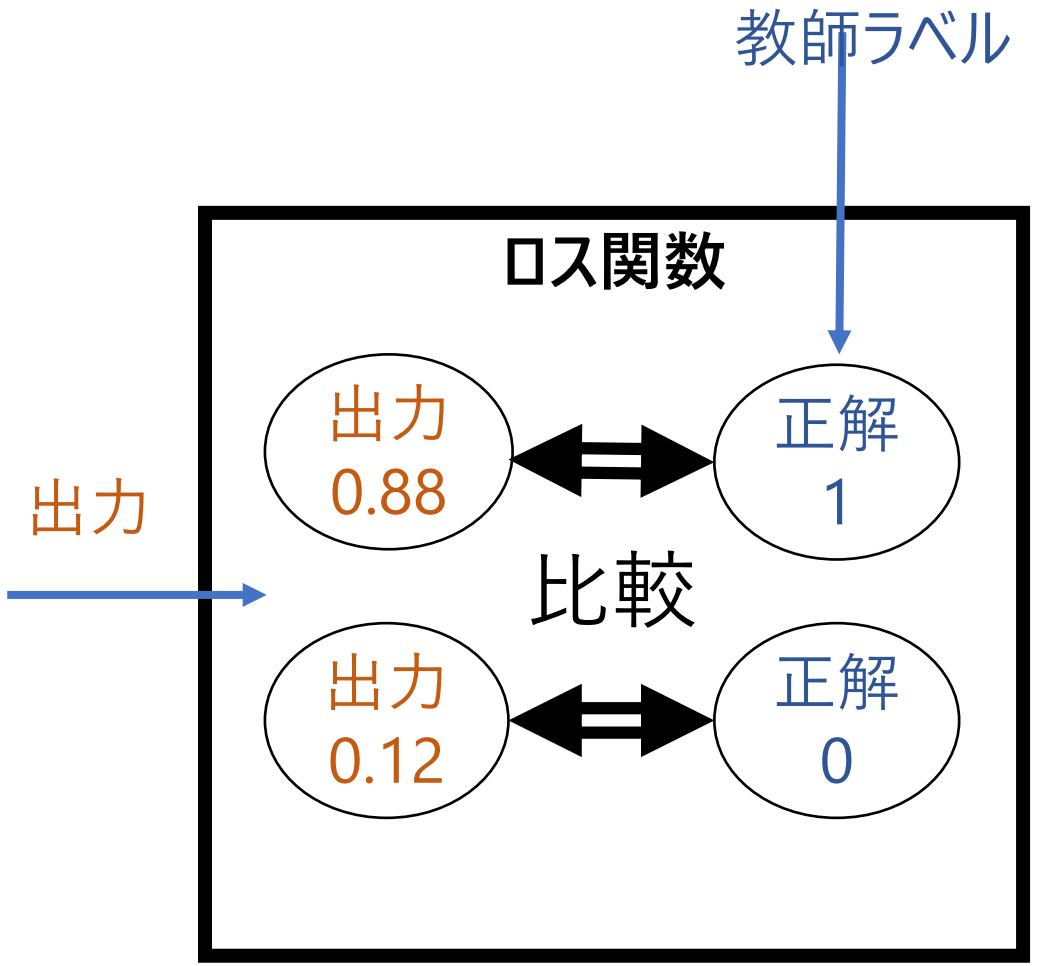
損失（ロス）関数

正解と、モデルの出力層の結果とを比べて、どれだけ違うかを量化する

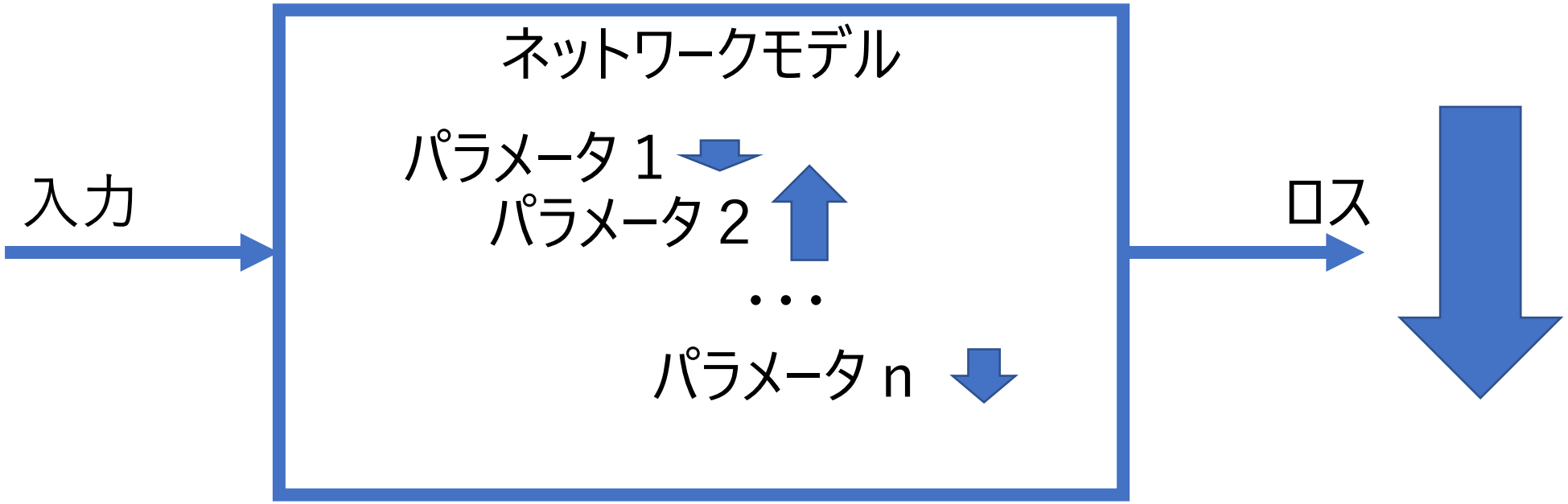


ロス関数の性質

1. 常に正である
2. 出力が正解から遠いほど大きな値になり、出力が正解に近いほど小さな値になる。



学習は、**ロス**（クロスエントロピー誤差など）が小さくなるように、パラメータを変更することで行う。

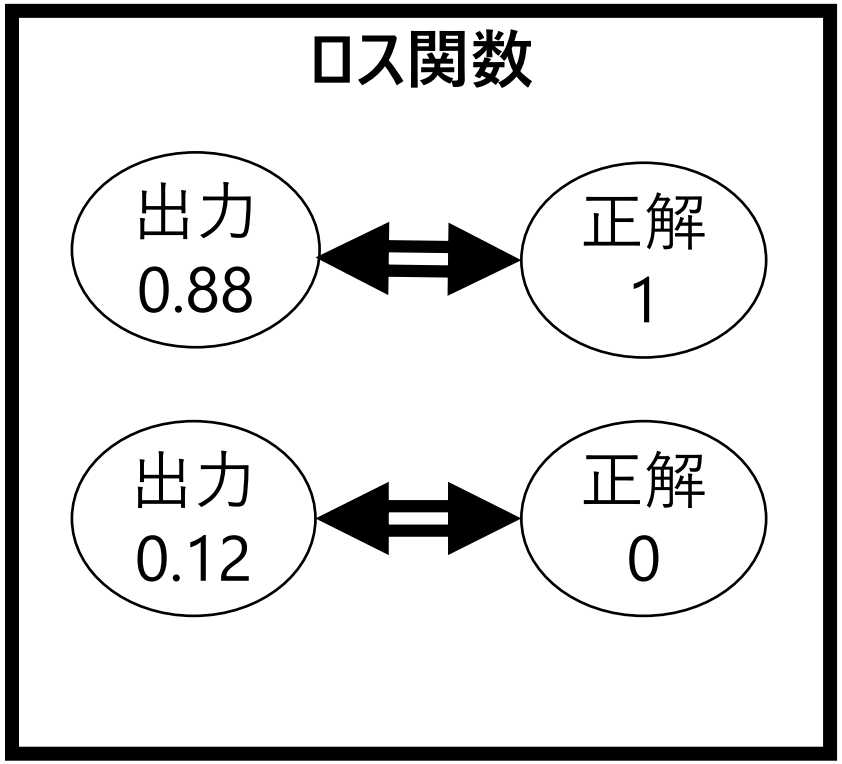


クロスエントロピー誤差

対応する正解ラベル

$$E = -\sum_{k} t_k \log y_k$$

k 番目の出力



対数

$$x = b^p \Leftrightarrow p = \log_b x$$

事象(確率p)の情報量 $-\log_2 p$

- コインの裏表
  - 表が出る確率pは1/2。=> 表という特定事象によって増える情報量は、$-\log_2(1/2)=-(\log_2 1-\log_2 2)=-(0-1)=1$　（$2^0=1, 2^1=2$）
  - 表裏は、１ビットで表現できる。１ビットで、裏表が特定できる。
- トランプのマーク
  - ハートが出る確率pは1/4。=> ハートという特定事象によって増える情報量は、$-\log_2(1/4)=-(\log_2 1-\log_2 4)=0+2=2$　（$2^0=1, 2^2=4$）
  - トランプのマークは、２ビットで表現できる。２ビットで、マークが特定できる。
- ルーレットを回して、東・西・南・北・南東・南西・北西・北東のいずれか
  - 確率pは、1/8。=>　東という特定事象によって増える情報量は、$-\log_2(1/8)=-(\log_2 1-\log_2 8)=0+3=3$　（$2^0=1, 2^3=8$）
  - 8方向の方角は、3ビットで表現できる。３ビットで方角が特定できる。

確率分布のエントロピー（平均情報量）

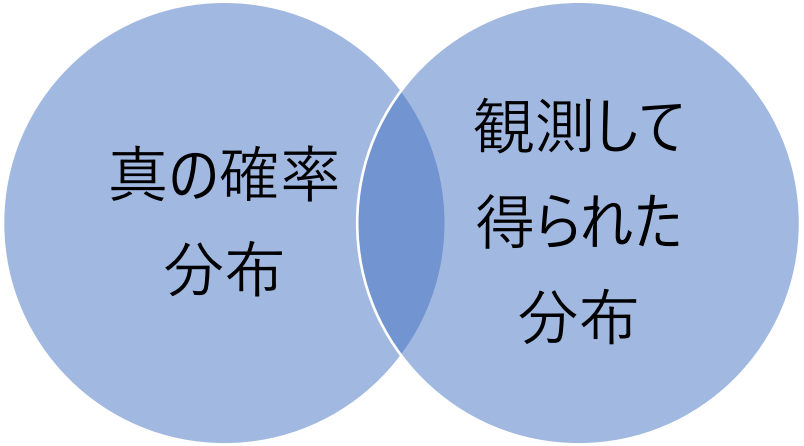
$$-\Sigma p(x)\log p(x)$$

事象 x の起きる確率　　事象 x の情報量

算数での平均は足して個数で割る。その代わりに、起きる確率で重みづけたものを足して、情報量の平均とする

コインの裏表の確率分布pは、p(表)＝0.5、p(裏)=0.5。一方、確率0.5の情報量は$-\log_2(1/2)=1$だった。このpのエントロピーは、-(0.5*(-1)+0.5*(-1))=1。

クロスエントロピー（交差平均情報量）

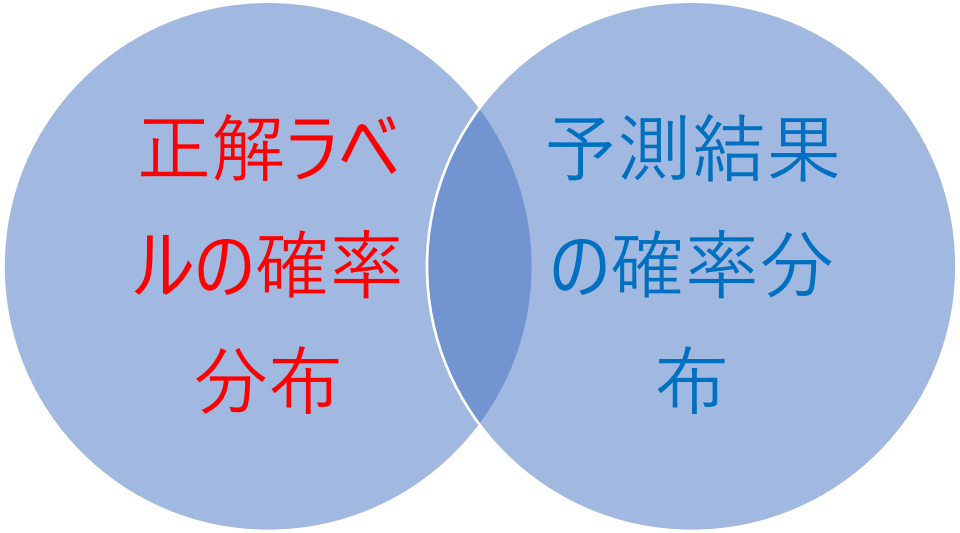


$$-\Sigma\ p(x)\ \log q(x)$$

真の確率で 重みづけ

x を観測して得られた情報量

クロスエントロピー



真の確率分布（重みづけ）

モデルが正しいと仮定して観測して得られた確率分布（評価対象）

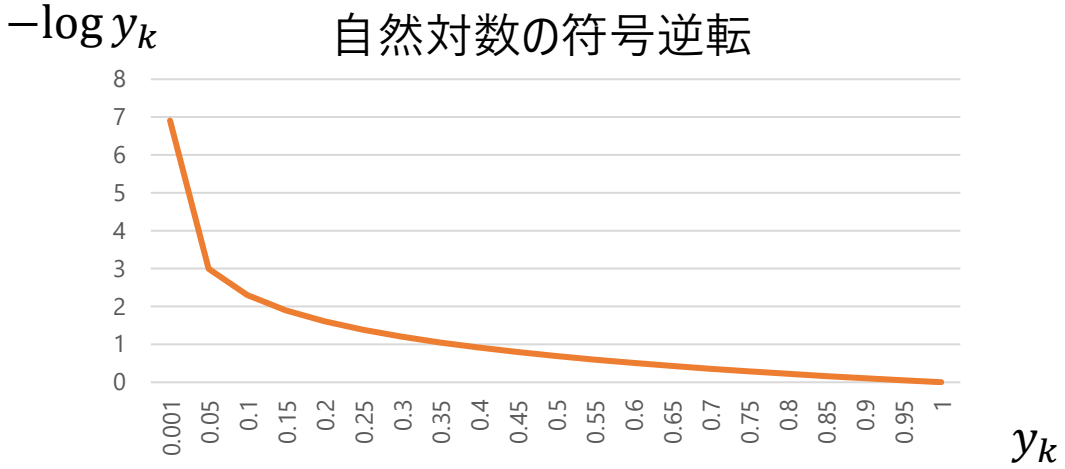
$$-\Sigma \; p(x) \; \log q(x)$$

クロスエントロピー誤差：２値の場合

対応する正解ラベル

$$E = -\sum_k t_k \log y_k$$

k番目の出力

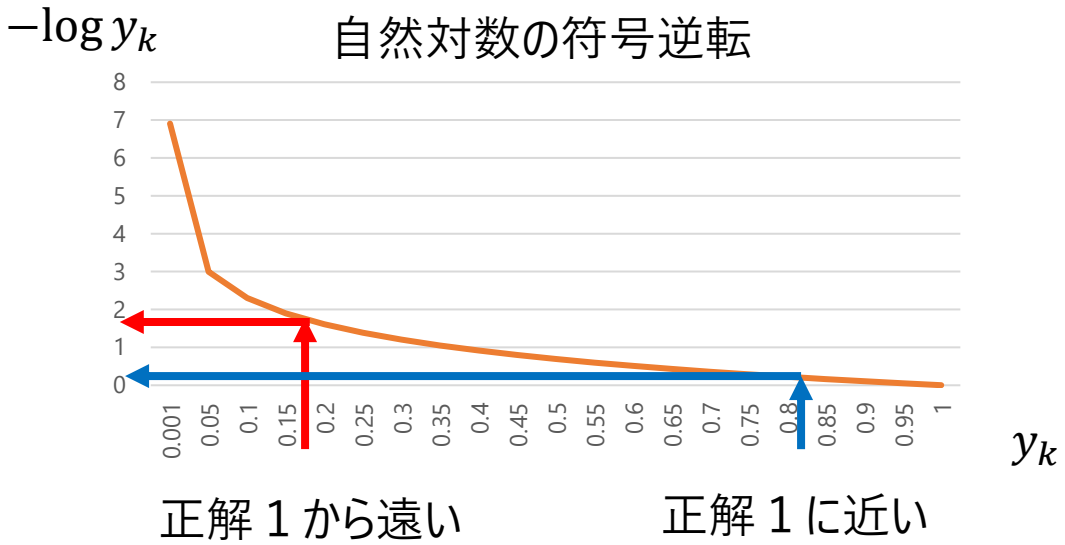


2値問題で、正解のときのラベルが１で、そうでないときは0とします。
すると、上の式のtkは正解の時だけ１で間違いの時は０なので、
正解の時の出力の対数を足しこむだけの計算になります。
最後に符号を逆転していることに注意。

クロスエントロピー誤差：ロス関数の性質

$$E = -\sum_{k} \log y_k$$

正解ラベルが 1 のときの出力

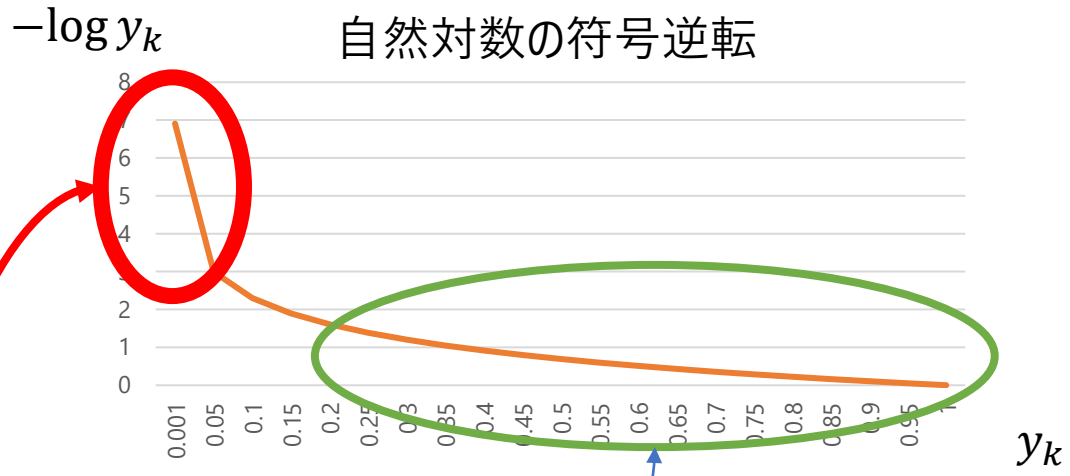


1．常に正。
2．データが正解 1 に近いほど小さな値、遠いほど大きな値。

クロスエントロピー誤差：学習向き

$$E = -\sum_{k} \log y_k$$

正解ラベルが１のときの出力



出力が正解から遠いほど、ロスはとても大きくなり、間違えたときのペナルティが大きい。

正解に近づいたら慎重に最適点を探す

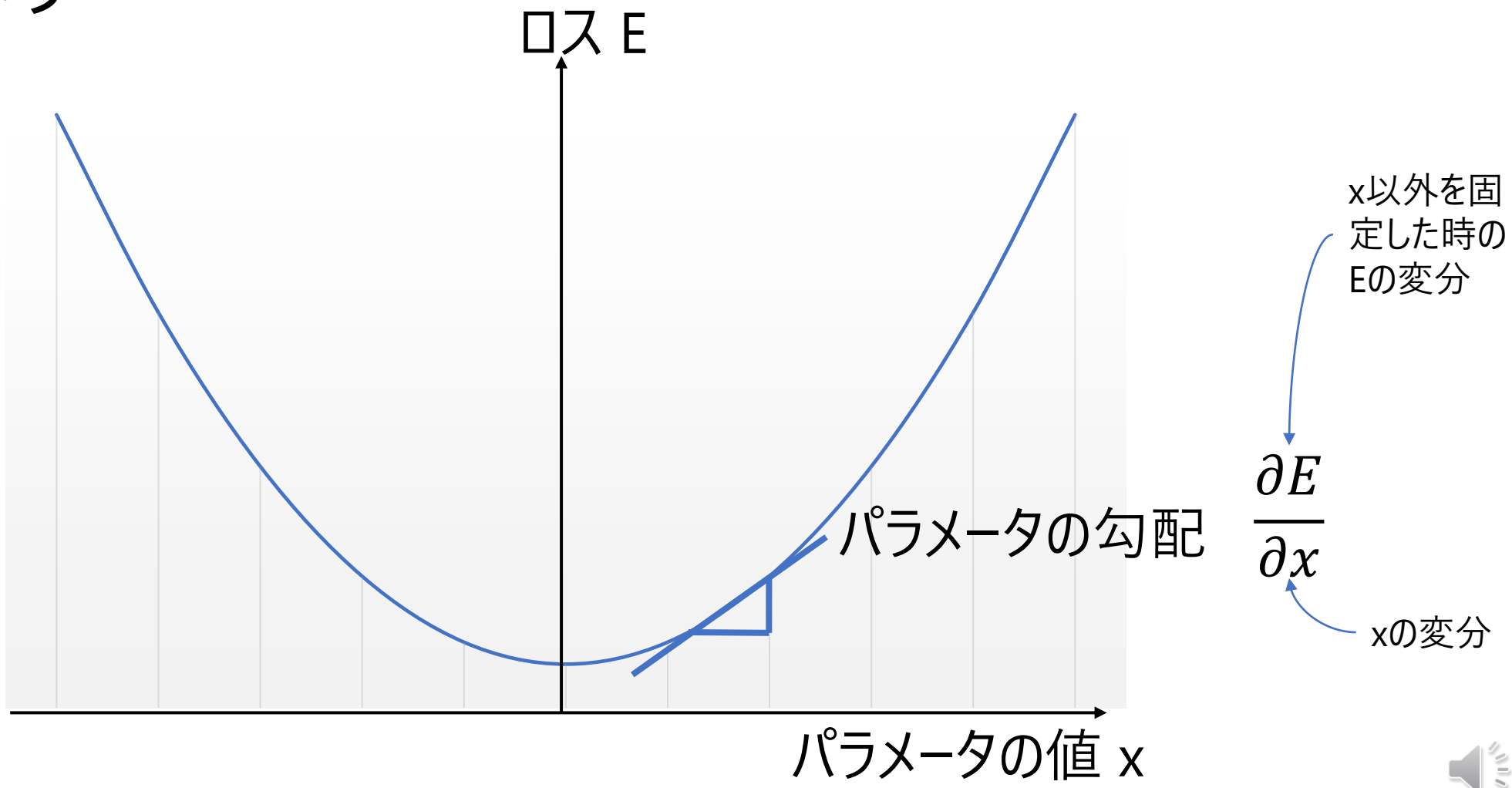
クロスエントロピー資料

- 交差エントロピーの導出
- 情報理論を視覚的に理解する
- ニューラルネットワークと深層学習、ニューラルネットワークの学習の改善

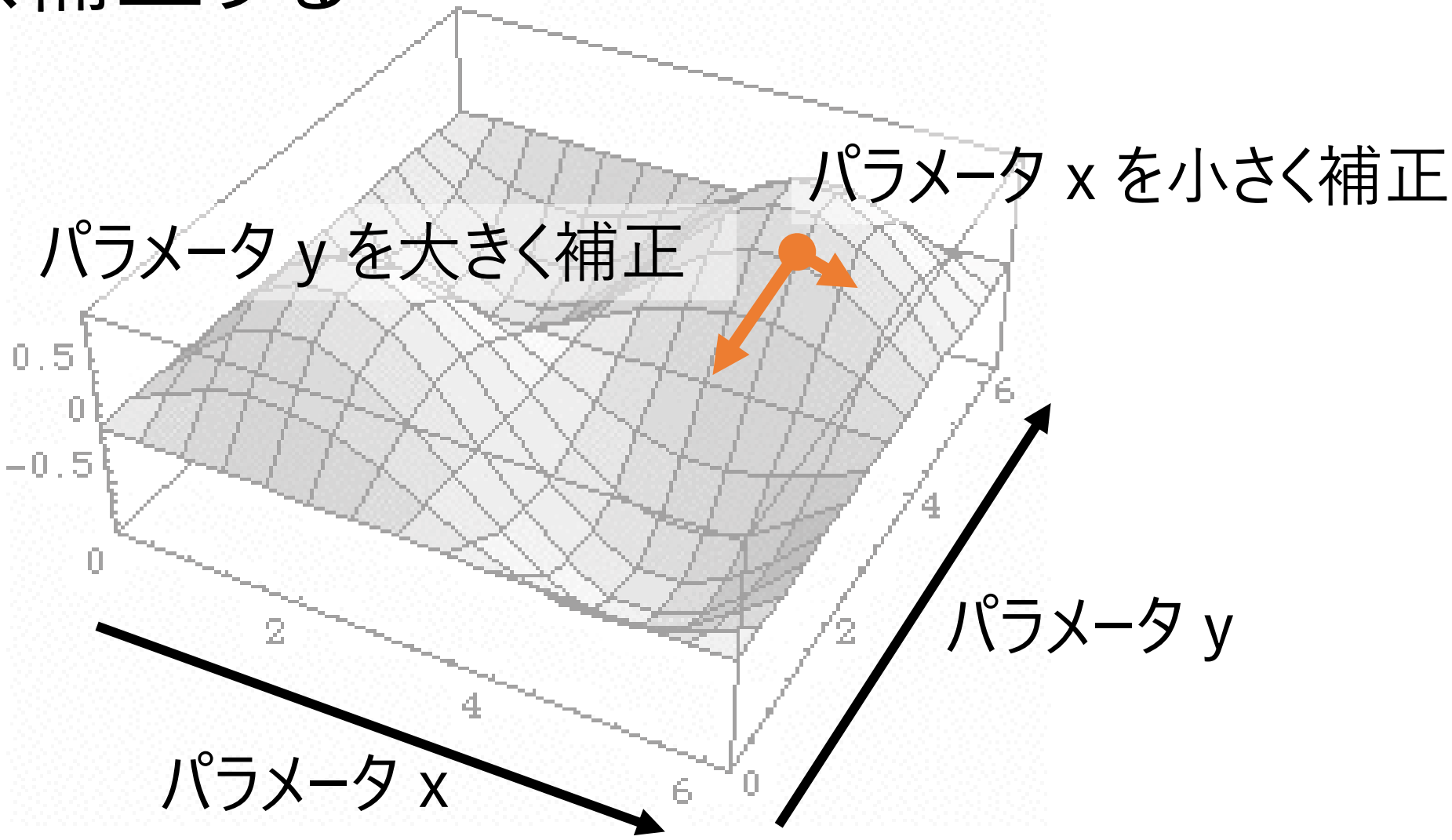
自然言語処理：ニューラルネット
逆伝播学習の仕組み

解説動画

あるパラメータをどう、どれだけ変更するか？　あるパラメータを少し変えたときにロスがどれだけ変わるかを、勾配という

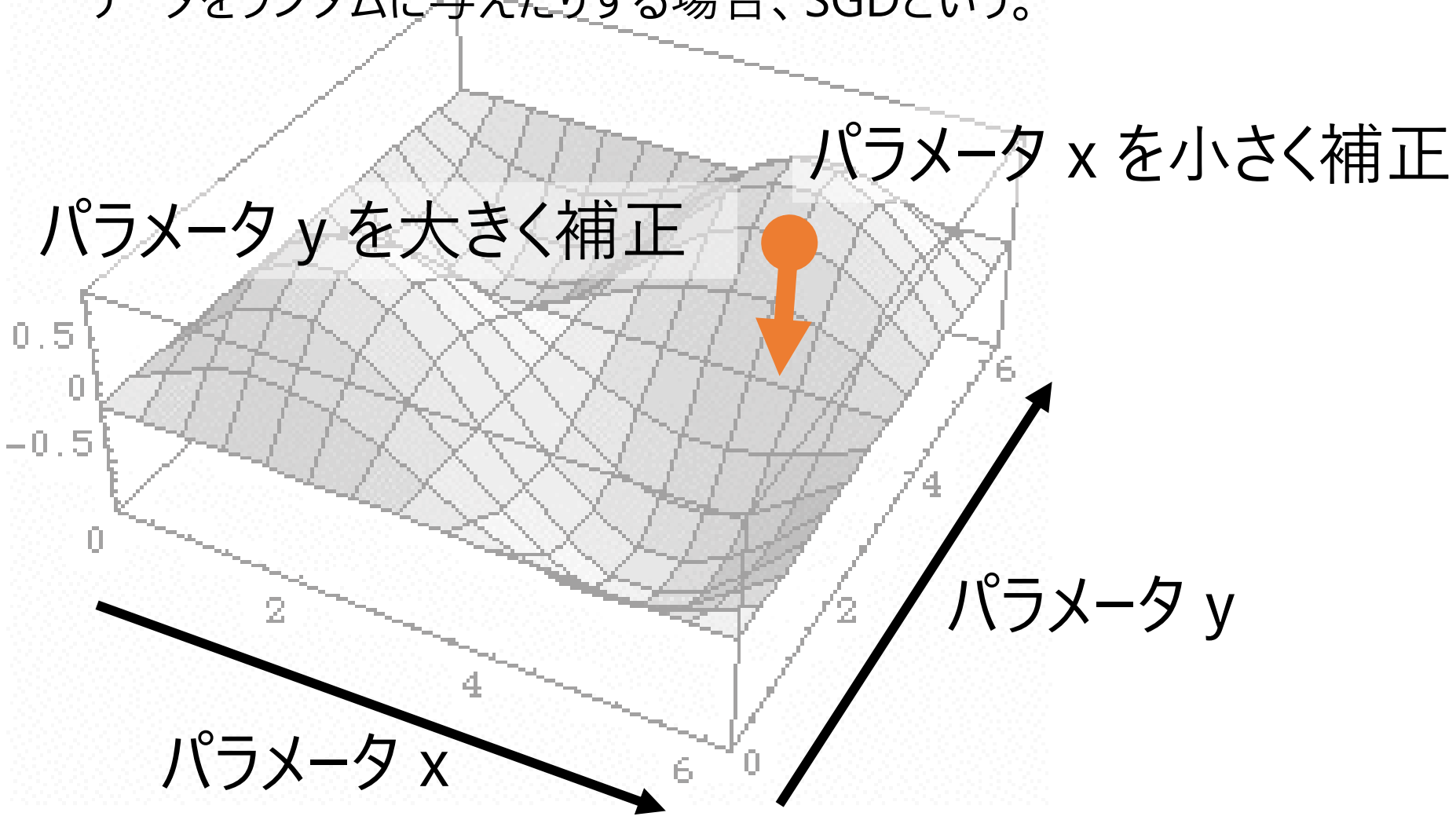


ロスが最も減るように、勾配の大きなパラメータほど強く補正する

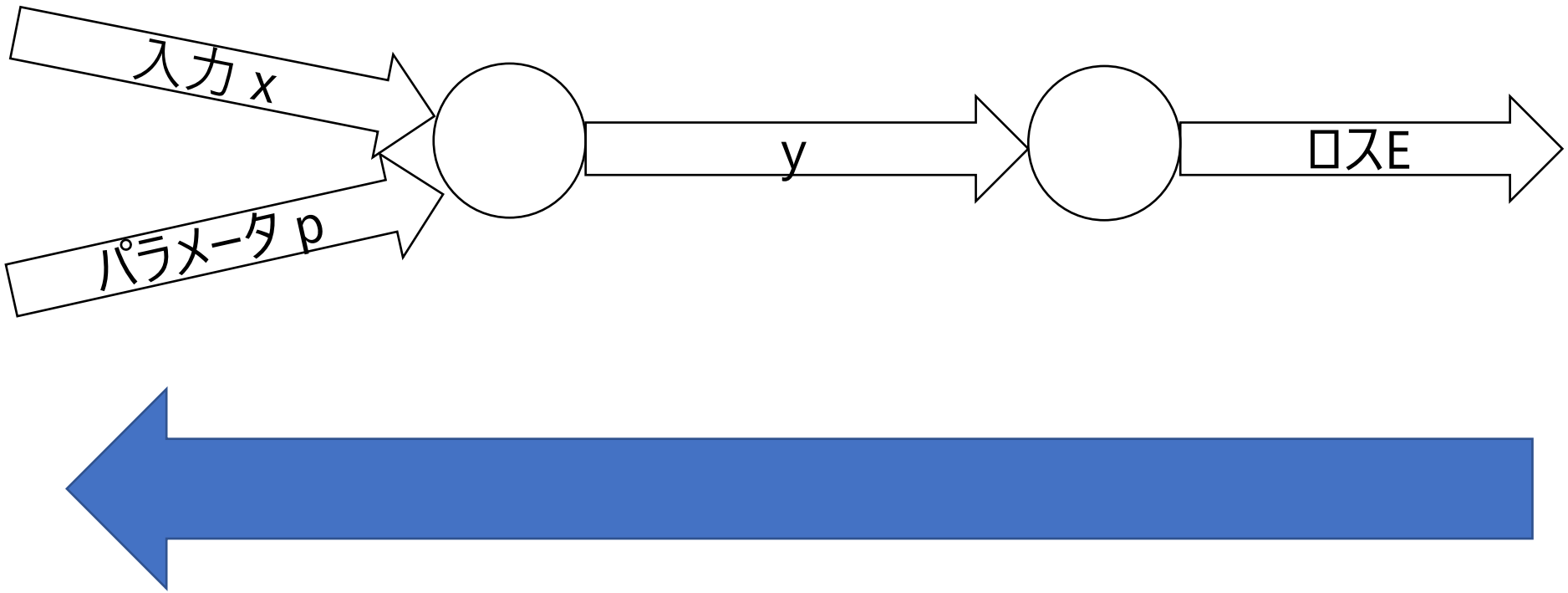


勾配降下法(Gradient Descent)

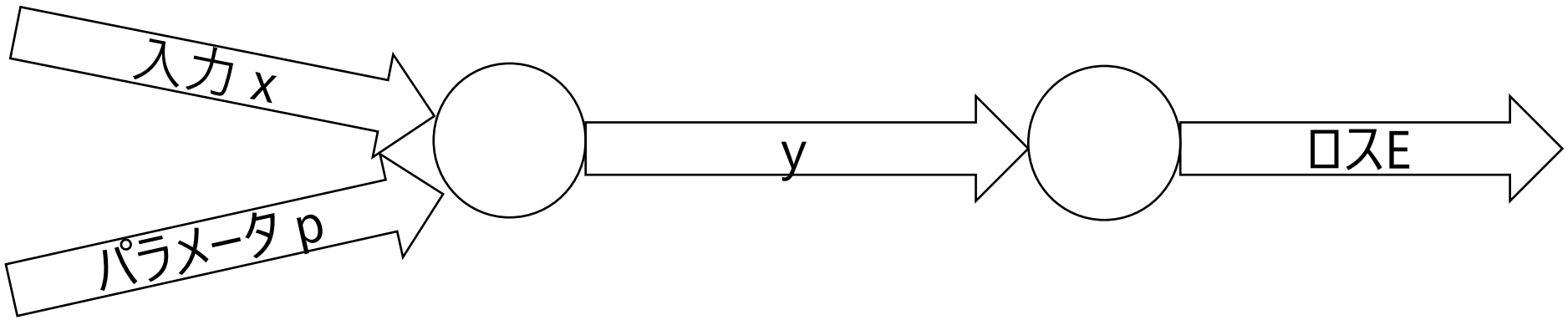
データをランダムに与えたりする場合、SGDという。



逆伝播：各パラメータのロスに対する勾配を、計算過程に沿って逆方向に求めていく

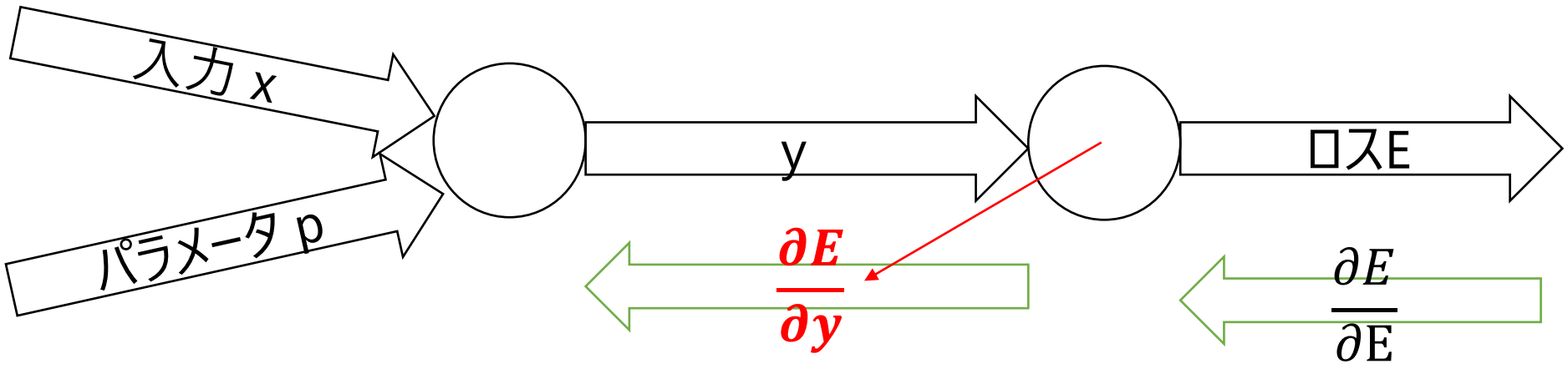


逆伝播：各パラメータのロスに対する勾配を、局所的な微分計算で求めていく



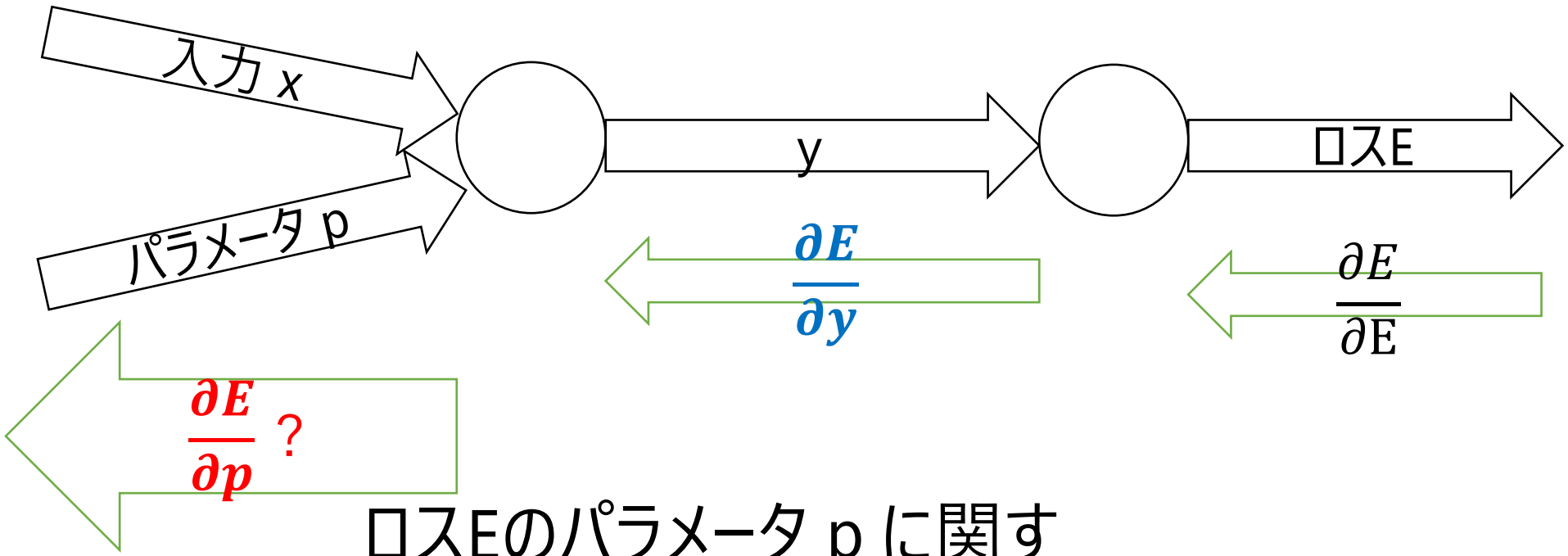
局所的な微分計算のみで効率よく

逆伝播：各パラメータの勾配を、局所的な微分計算で求めていく



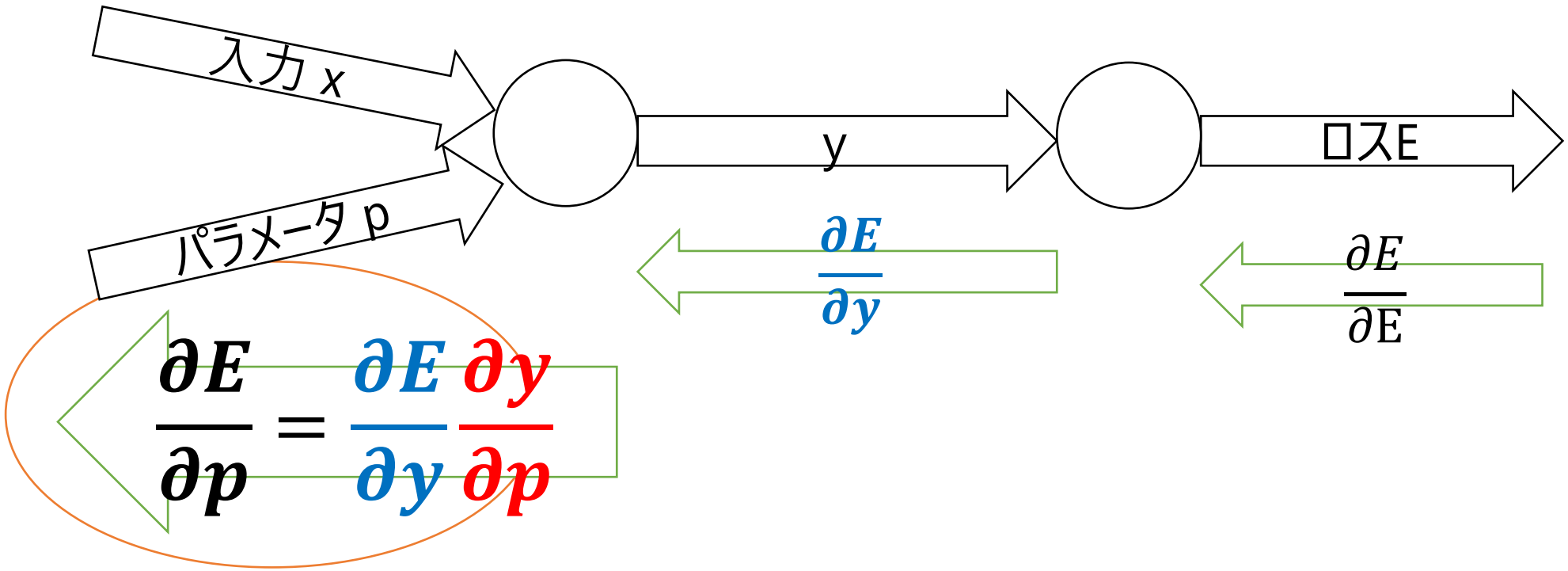
ロスEの y に関する傾き

逆伝播：各パラメータの勾配を、局所的な微分計算で求めていく

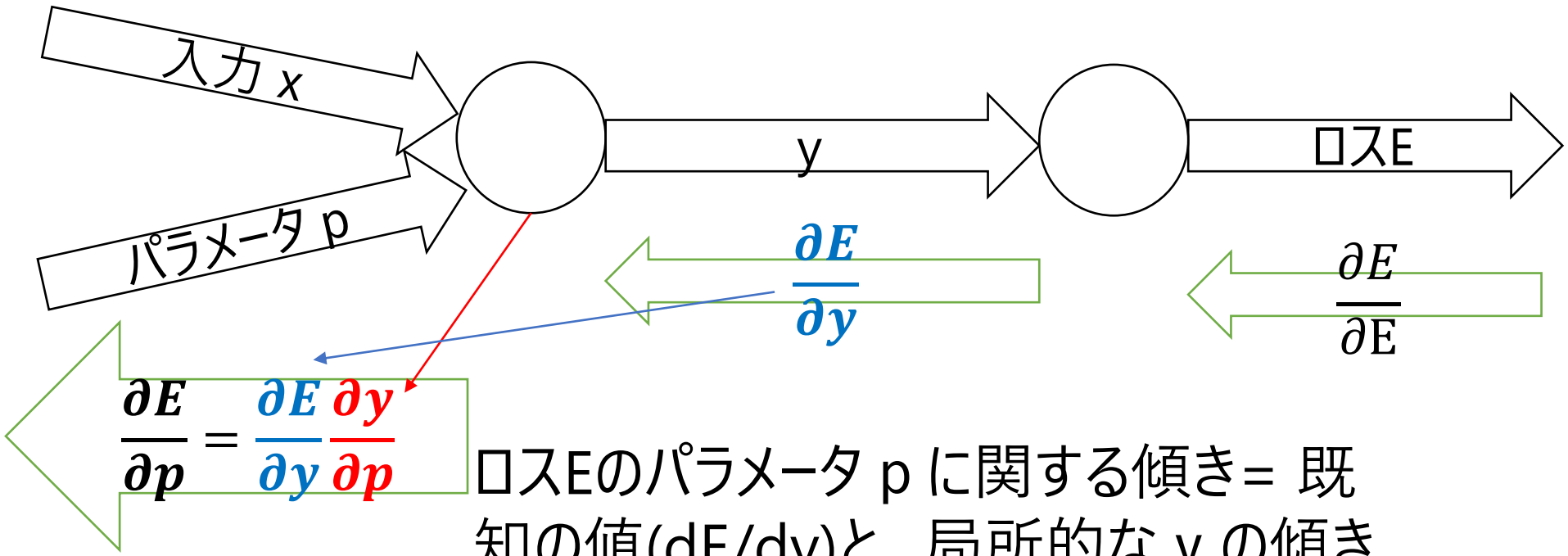


ロスEのパラメータ p に関する傾き???

逆伝播：合成関数の微分のチェインルール

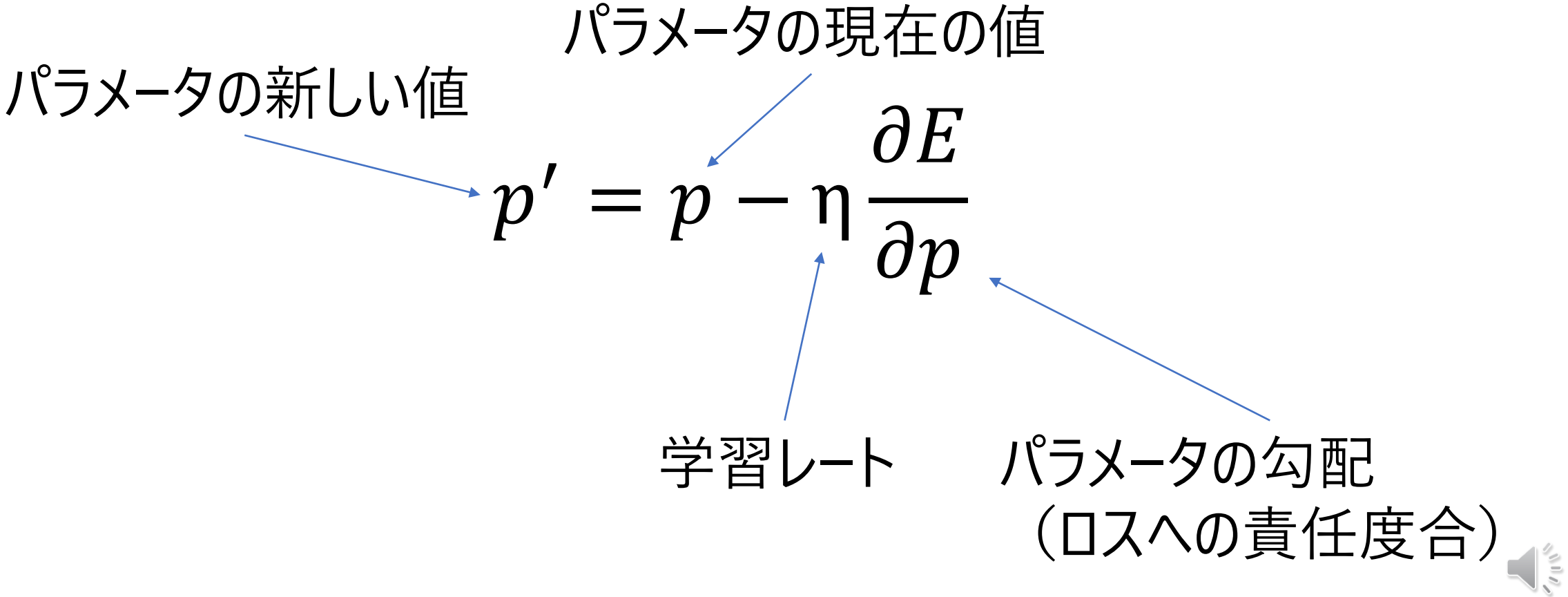


逆伝播：各パラメータの勾配を、局所的な微分計算で求めていく

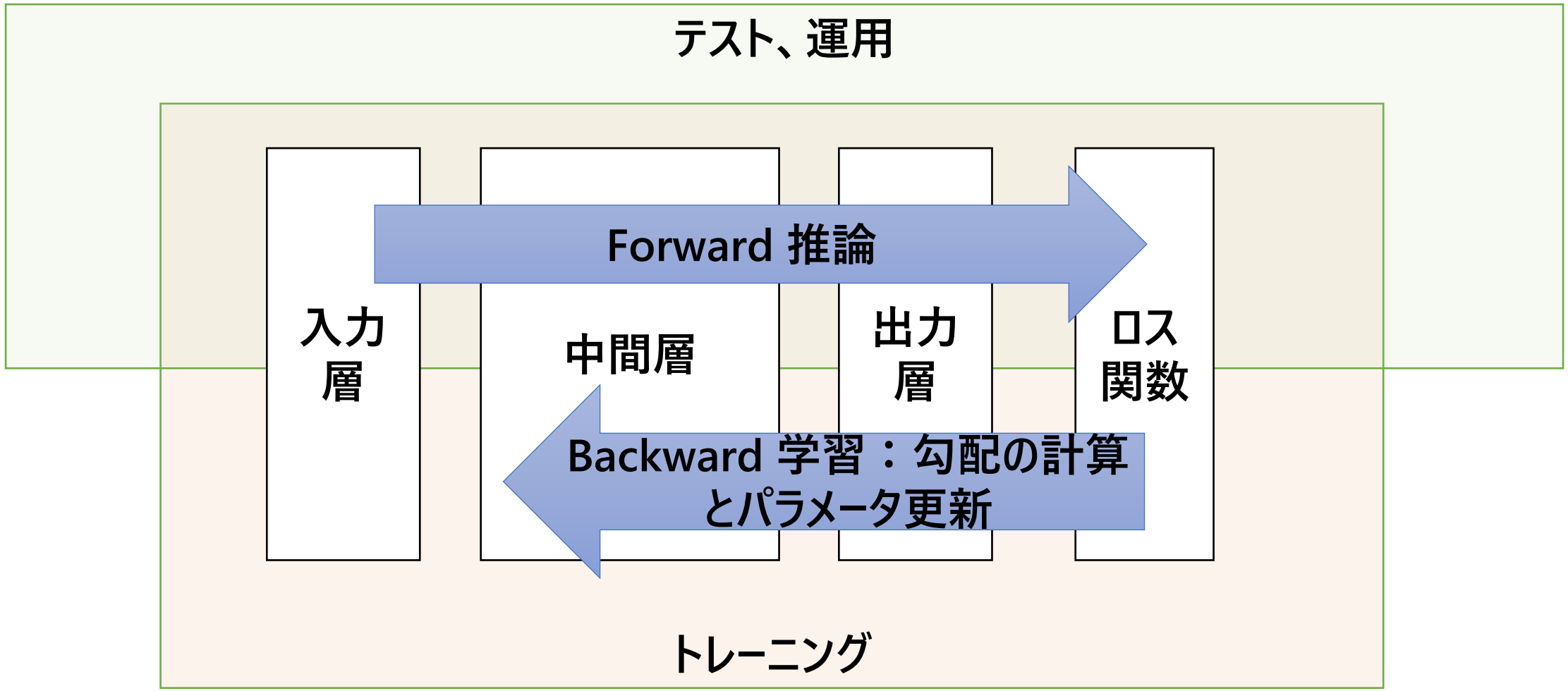


ロスEのパラメータ p に関する傾き= 既知の値(dE/dy)と、局所的な y の傾き計算で得られる

各パラメータの勾配を決めたら、勾配に応じてパラメータを更新する



ネットワークの学習
バックプロパゲーション（逆伝播）



PyTorchのAutograd

PyTorchなどの最新のフレームワークは、Chainerの考えを吸収し、実行時に計算過程を記録し、それをもとにバックプロパゲーションを自動的に行うことをライブラリでサポートしている。

逆伝播参考資料

- 「ゼロから作るDeep Learning」、第4章、第5章
- ニューラルネットワークの学習の仕組み
- ニューラルネットワークと深層学習、逆伝播の仕組み
- 誤差逆伝播法を宇宙一わかりやすく解説してみる | ロボット ...
- 連鎖律の原理で、誤差を「後ろ」に伝えよう
- 高卒でもわかる機械学習 (5) 誤差逆伝播法 その1
- Deep Learning精度向上テクニック：様々な最適化手法 #1

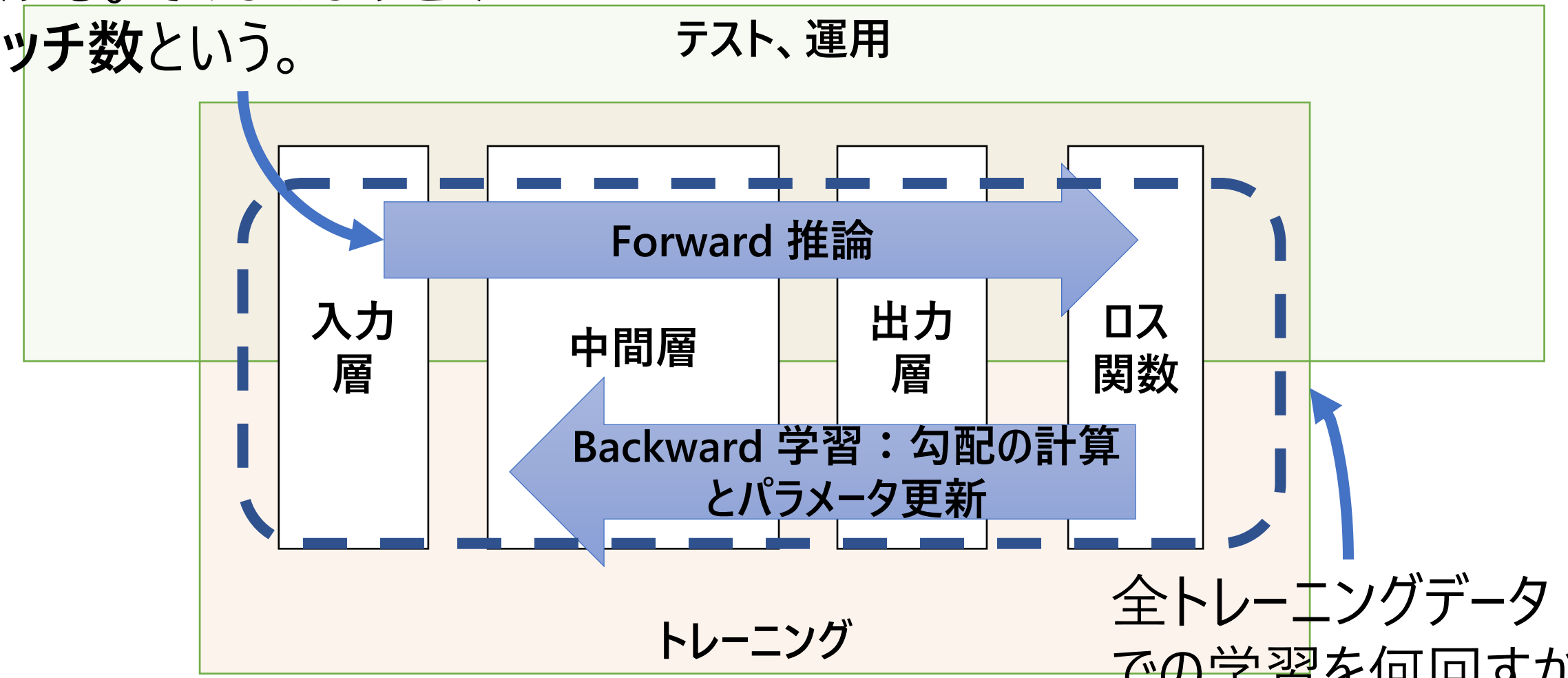
PyTorch事始め

課題：PyTorch 事始め

- 動的翻訳ができるChromeを使ってください。
- https://pytorch.org/ へ行ってください。
- 右クリックで、日本語へ翻訳。以下同様。
- 上のメニューからチュートリアル > ページ内の基本を学ぶ > 1. テンソルへ。
- Google Colabで実行を別タブでクリック。Colabページを翻訳すると、コードまで翻訳されてしまいますので、元のページの日本語訳説明を読みながら、Colabページを英語のままで、含まれるコードセクションを実行していきましょう。
- PyTorchのテンソルの動きを確認していきましょう。

エポック、ミニバッチ

データを何個かまとめて投入する。そのまとまりを**ミニバッチ数**という。



全トレーニングデータでの学習を何回すかを**エポック数**という

100本ノック第8章課題70～78と79

- 100本ノックの第8章は、中でも秀逸の課題群です。ニューラルネットの概念がステップバイステップで積み上げ式に学べます。

- ニューラルネットのコードに初めて触れる場合、最初から書こうとするのは無理です。まず、単純なサンプルを読み、慣れる必要があります。「100本ノック」の8章の課題 の70から78の回答例の、ポイントの部分だけ穴埋めにしています。「NLP、ニューラルネット.ipynb」というノートのサンプルを完成させてください。ノートには解説やコメントを入れています。

100本ノック第8章課題70～78と79

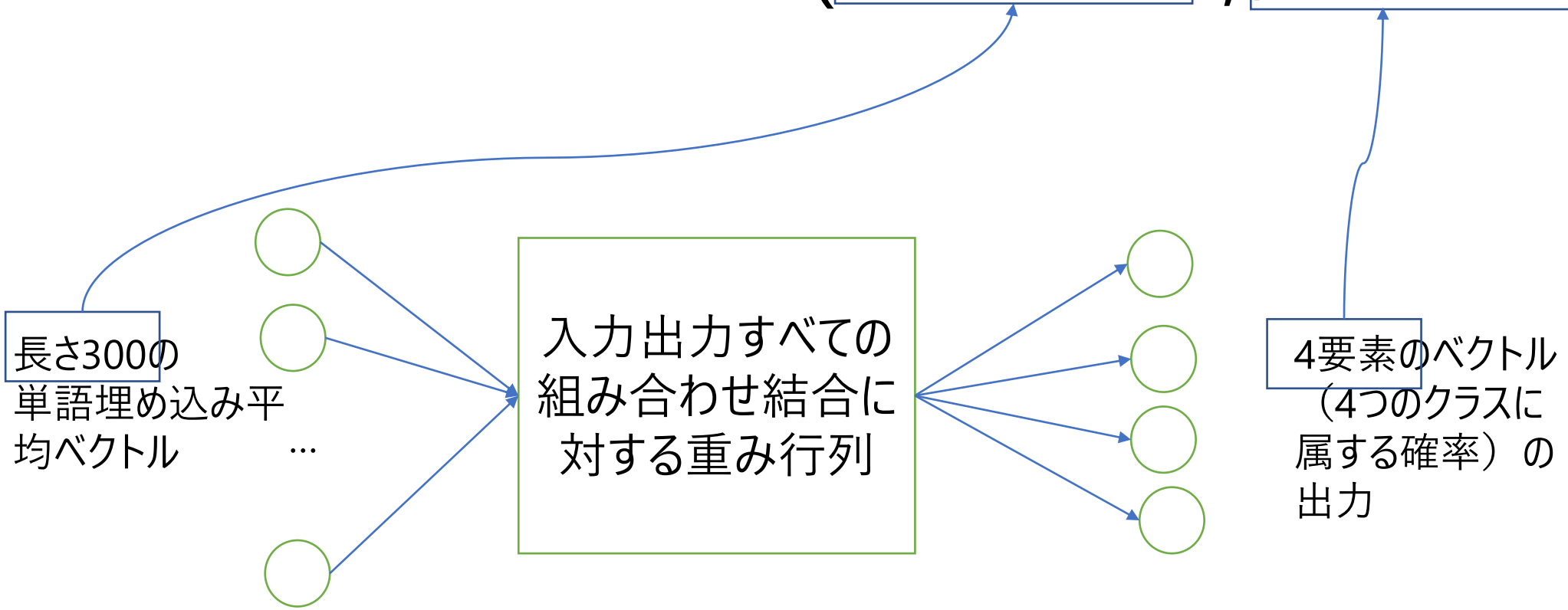
- 最後の79は、いろいろいじってみよという自由課題になっています。チャレンジしてみましょう。講師のコードも入れておきますが、いろいろいじってみてください。変更後のコードと、実行ログを残してください。この世界は、数理科学というより、まだ経験科学であるという実感を持てると思います。
- この章の到達点は、テキストを分類するタスクなので、応用が広いです。
- ここで、ニューラルネットの大枠は習得済み。ニューラルネットの入門書を一冊理解したことに相当します。

100本ノック第8章課題70～78と79

- あと、CNN、RNN、Attention、Transformerは、ネットアーキテクチャのデザインパターンを習得していくような感じです。第8章が終われば、もう自力でそれらのDeep Learningの技術を深めていくことができます。
- なお、自然言語アプリを目指す方は、第7章の「単語ベクトル」は必須です。

課題を解く参考

71：torch.nn.Linear(入力サイズ,出力サイズ)



実際には、データ件数（ミニバッチの数)分、まとめて計算され、
入力も出力も、1次元ベクトルではなく、ベクトルを横（行）、データを縦（列）に並べた行列となる

呼び出し可能オブジェクト：*outputs = model(inputs)* は、forward呼び出しに化ける

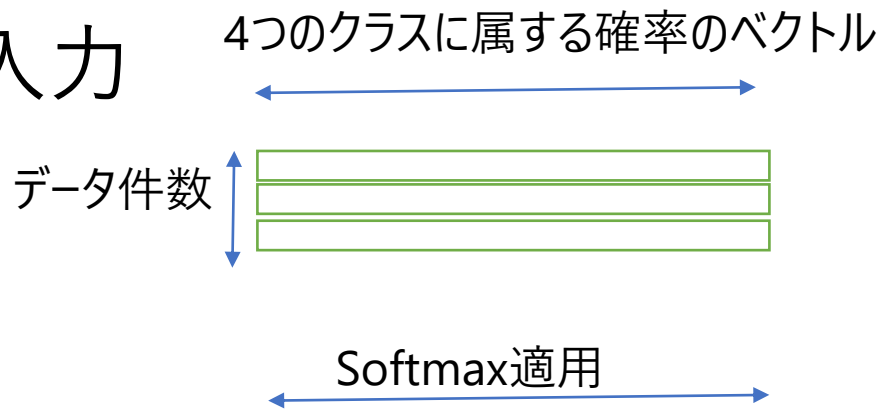
Pytorchは、Pythonの呼び出し可能オブジェクトの機能を利用している

- object.__call__(self[, args...])
  - Called when the instance is "called" as a function; if this method is defined, x(arg1, arg2, ...) roughly translates to type(x).__call__(x, arg1, ...).
  - オブジェクト()　と書くと、クラス.__call__(インスタンス、引数...)　に化ける

- Pytorchのコードを見ると
  - moduleクラスの__call__メソッドは、中で forward 呼び出しを行っている。

torch.softmax(x, dim=1)

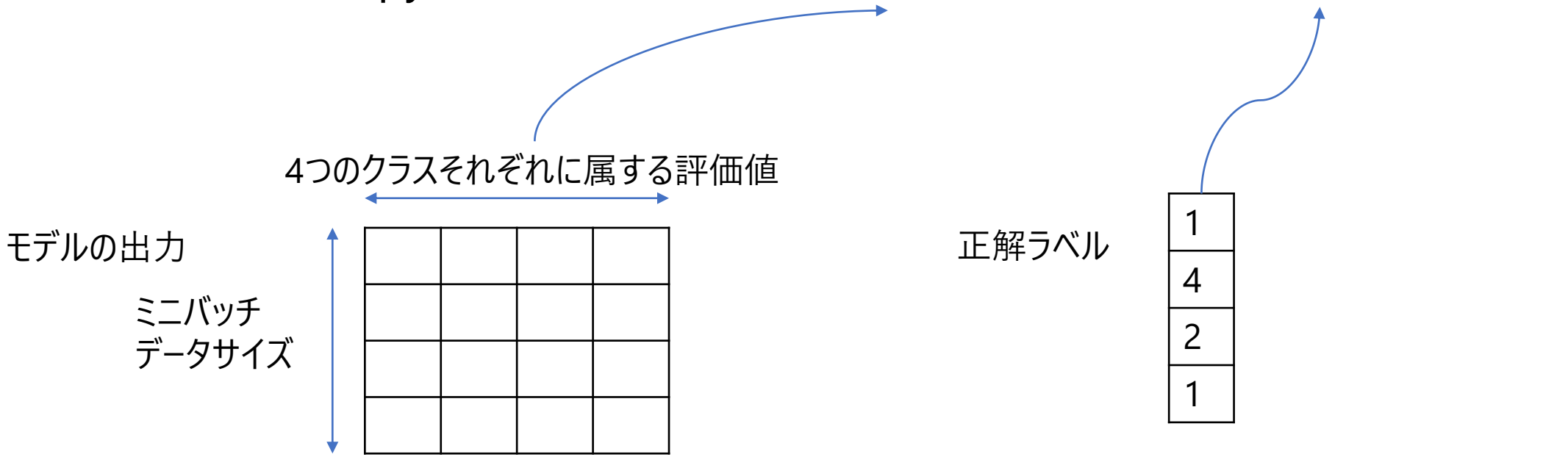
入力



softmaxをとる次元：dim=0がミニバッチ、dim= 1 が4要素のベクトル。dim=1とすることで、4個の値に対して合計1になるように変換する

72. torch.nn.CrossEntropyLoss()

loss = CrossEntropyLossのインスタンス(モデルの出力, 正解ラベルの1次元テンソル)



確認クイズ

- 確認クイズをやってください。