現場で使えるディープラーニング基礎講座

DAY2

SkillUP AI

前回の復習

- 前回は何を学びましたか？

ディープラーニング基礎 後半

目次

ミニバッチ学習

バッチ学習とミニバッチ学習

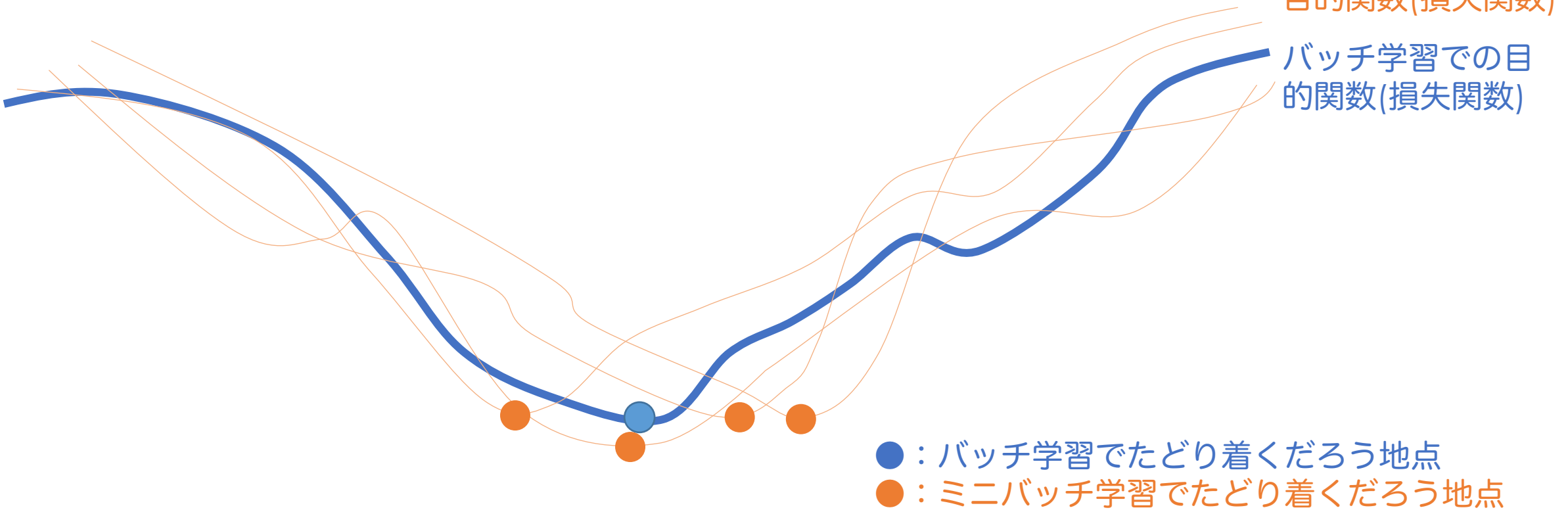
- バッチ学習とは、学習用データを一度に全て入力し、重みを更新する学習方法。
    - 最適化の目的関数は一定。
- ミニバッチ学習とは、学習用データを小さなバッチに分けて、その小さなバッチ毎に重みを更新していく学習方法。1つのバッチに含まれるデータ数をバッチサイズという。
    - 重みの更新は、ミニバッチを入れ替えながら行う。
    - 最適化の目的関数は、ミニバッチ毎に異なる。

ミニバッチ学習の特徴

- 一般的に、バッチサイズが大きいほど勾配推定が正確になる。逆に、小さなバッチサイズは、正則化の効果をもたらすことがある。
- バッチサイズが大きくなるとメモリ使用量が増えるため、容量の大きいデータを扱う場合は、メモリサイズがバッチサイズの制約になる。
- バッチサイズを小さくすると、更新回数が多くなるため、計算時間が長くなる。
- バッチサイズを大きくすると、重みの更新量が安定するので、学習係数を大きくできる。学習係数を大きくすると、学習が速く進む。
- ミニバッチ学習では、局所的最小解にトラップされてしまうリスクを低減できる。
    - 最小値探索の考え方は、勾配法を参照のこと。

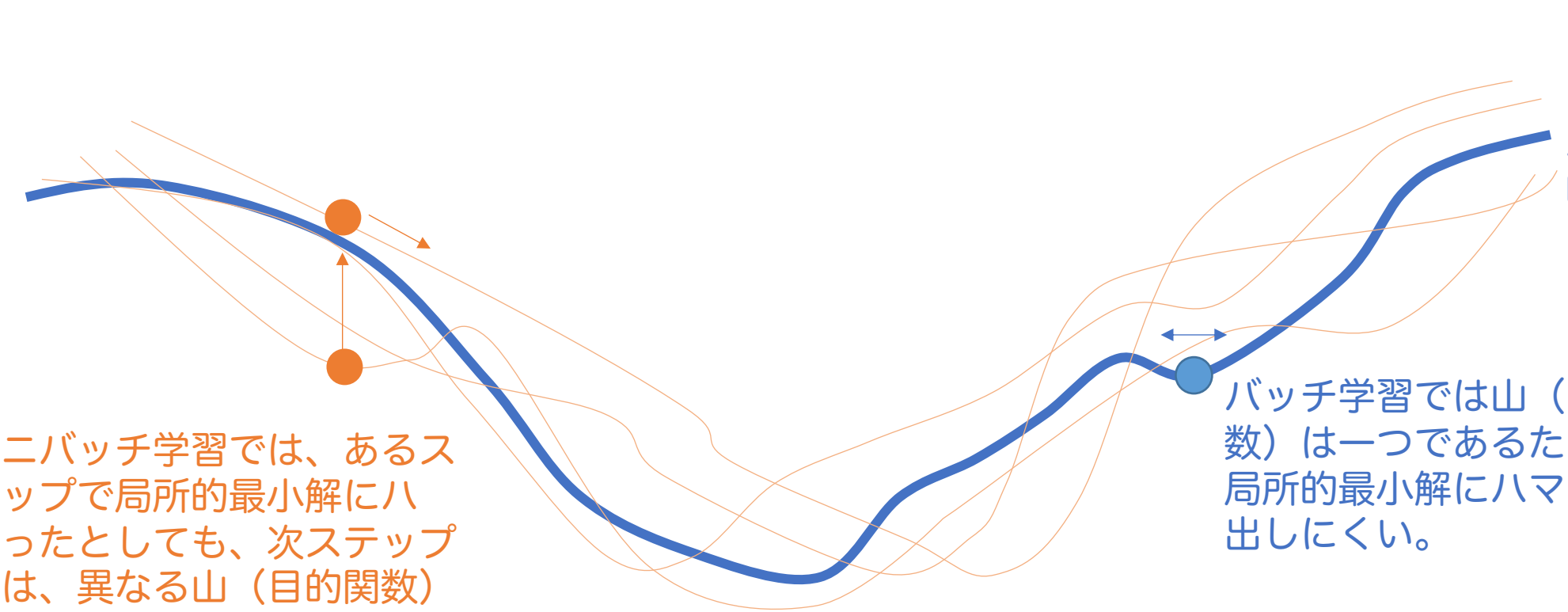
バッチ学習とミニバッチ学習の違い

- バッチ学習では、最適化の目的関数は一定。ミニバッチ学習では、最適化の目的関数は、ミニバッチ毎に異なる。
- どの地点での結果を採用するのが最もよいか？



バッチ学習とミニバッチ学習の違い

- ミニバッチ学習では、局所的最小解にトラップされてしまうリスクを低減できる。



ミニバッチ学習と損失関数

- バッチ学習の際は、一度にすべての学習用データの損失を算出し、その和(または平均)を最小化していく。例えば、100個の学習用データがある場合は、100個それぞれの損失を算出しその和(または平均)を求め、その和(または平均)を最小化していく。
- ミニバッチ学習の際は、ミニバッチ毎に損失の平均を求め、それを最小化するように学習を進める。例えば、バッチサイズが25の場合、 まず最初の25個についてそれぞれの損失を算出しその平均を求め、その平均を最小化するように計算する。次に、他の25個について、それぞれの損失を算出しその平均を求め、その平均を最小化するように計算する。これを繰り返す。
    - 全ミニバッチ を1順することを1エポックと数える。通常は、計算が収束するまでに、10エポック〜数百エポックくらい繰り返すことになる。

ミニバッチ学習におけるデータセットの与え方

- ミニバッチ学習は、一般的には非復元抽出によって行われることが多いが、必ずこうしなければならないというわけではなく、分析者がデータセットの与え方を工夫することもできる。ただし、工夫しても計算が上手くいくとは限らない。
- 工夫のしどころ。
  - 一般的には、エポック毎にシャッフルするが、シャッフルするタイミングを任意に変えてみる
  - 与えるミニバッチの順番を意図的に操作してみる
    - 例、出現頻度の少ないラベルのデータを先に学習させる
  - 抽出されるラベルの割合が一定になるように抽出してみる
  - 復元抽出にしてみる

[演習] ミニバッチ学習関数の実装

- 2_1_minibatch_trainee.ipynb
  - ミニバッチ学習関数を実装しましょう。
  - 復元抽出による方法と非復元抽出による方法を比較しましょう。

[グループワーク] ミニバッチ学習時のバッチサイズの決め方

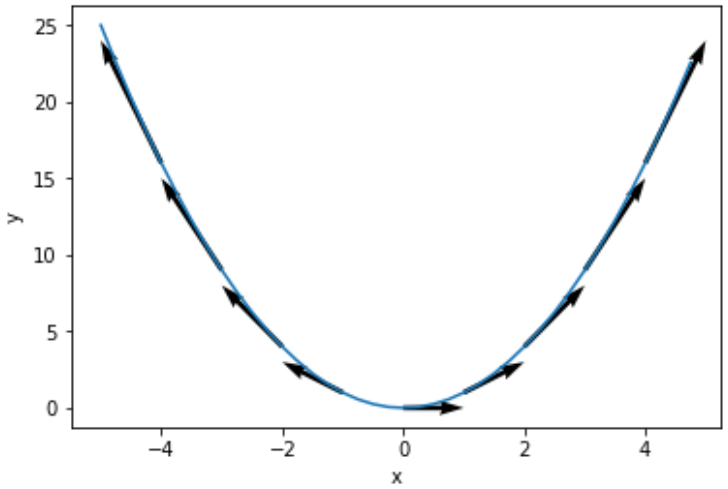
- 2~3名のグループに分かれて、ミニバッチ学習時におけるバッチサイズの決め方を話し合ってみましょう。(15分)
- 例えば、
  - バッチサイズを変えると、汎化誤差(テストデータでの誤差)は変化するでしょうか？
  - 初めて解く問題においては、何を参考にバッチサイズを決めればいいでしょうか？
  - バッチサイズは、どれくらいまで小さくしていいのでしょうか？
  - バッチサイズは、どれくらいまで大きくしていいのでしょうか？
  - 学習の途中で、バッチサイズを変更することは意味があるのでしょうか？
- 最後に、グループごとに発表していただきます。 (15分)

Any Questions ?

微分

微分値の意味の確認

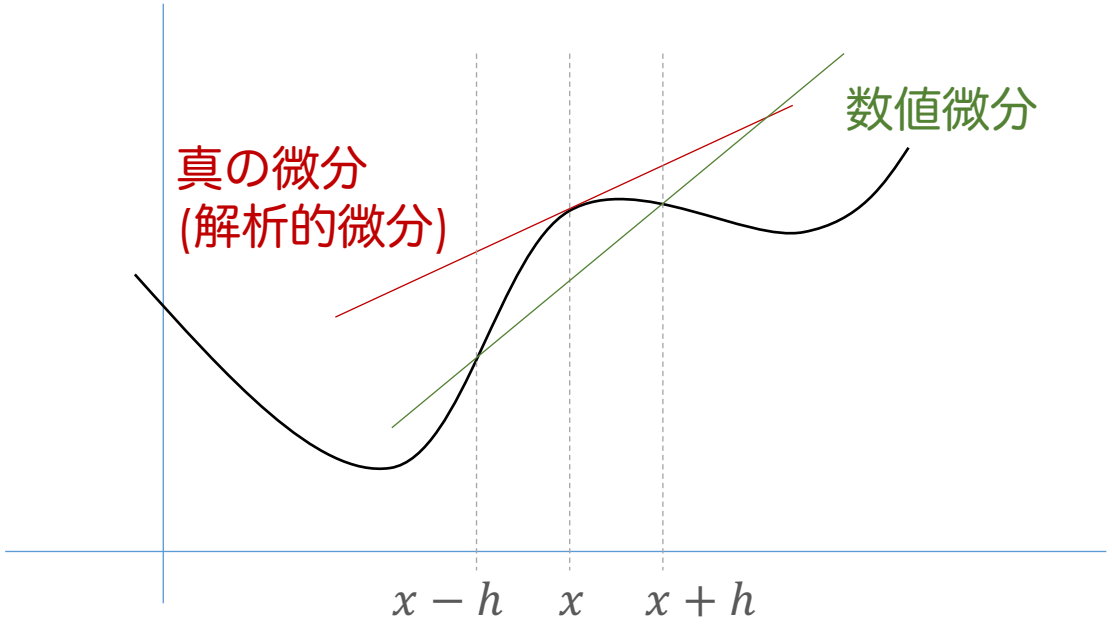
- 微分値の正負は、ある場所から少し動いた時に関数の値が大きくなる方向を表す。
- 微分値の値は、その方向に微小距離移動した時に、関数の値が大きくなる程度を意味する。
- 例えば、$f(x) = x^2$の$x = 1$における微分値は2であるが、x軸の正の方向に微小距離Δ移動したら関数の値が約2Δ大きくなるという意味になる。また同様に、$f(x) = x^2$の$x = -1$における微分値は-2であるが、x軸の負の方向に微小距離Δ移動したら関数の値が約2Δ大きくなるという意味になる。



微分値の意味

数値微分と解析的微分

- 数値微分とは、微小な差分によって微分を近似的に求めること。これに対し、式の展開によって微分を求めることを解析的に微分を求めるという。
- 解析的な微分には誤差が含まれない。
- 数値微分と解析的微分の違いを下図に示す。



- xにおける数値微分値は、ここでは微小な2hの範囲の傾きとして近似的に求めている。これを中心差分と呼ぶ。
- 図からわかるように、数値微分には誤差が含まれる。また、数値微分は解析的微分に比べ計算時間もかかる。よって、NNの学習では通常、解析的微分を用いる。
- 数値微分は勾配確認で用いる。勾配確認については後述する。

偏微分

- 偏微分とは、対象とする関数が多変数の状況で便利に使える微分。
- 例えば、$f(x,y)$という関数があった時に、yを定数とみなしxにだけ着目して微分することをxについて偏微分するという。同様に、xを定数とみなしyにだけ着目して微分することをyについて偏微分するという。
- 偏微分の例を以下に示す。

$$f(x,y) = 2x^2 + 3y$$

$$\frac{\partial f}{\partial x} = 4x$$

$$\frac{\partial f}{\partial y} = 3$$

勾配ベクトル

- 勾配ベクトルとは、偏微分をひとまとまりで表現したもの。
- 勾配ベクトルは、ある場所から少し動いた時に最も関数の値が大きくなる方向とその大きさを表す。逆に、マイナスの勾配ベクトルは、ある場所から少し動いた時に最も関数の値が小さくなる方向とその大きさを表す。
- 最急降下法では、マイナスの勾配ベクトルを用いて、関数の値が最も小さくなるところを探す。
- マイナスの勾配ベクトルは、あくまでも、関数の値が小さくなる方向を指し示すだけであって、関数の最も小さくなる場所を保証しているわけではない。
- 勾配ベクトルは、∇(ナブラ)記号を用いて表されることもある。
- 勾配ベクトルの例を以下に示す。

$$f(x,y) = 2x^2 + 3y$$

$$\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = (4x, 3)$$

[演習] 数値微分関数の実装

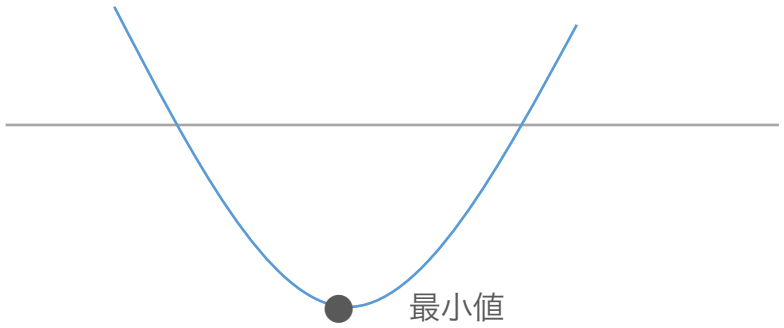
- 2_2_numerical_differentiation_trainee.ipynb
  - 数値微分関数を実装しましょう。

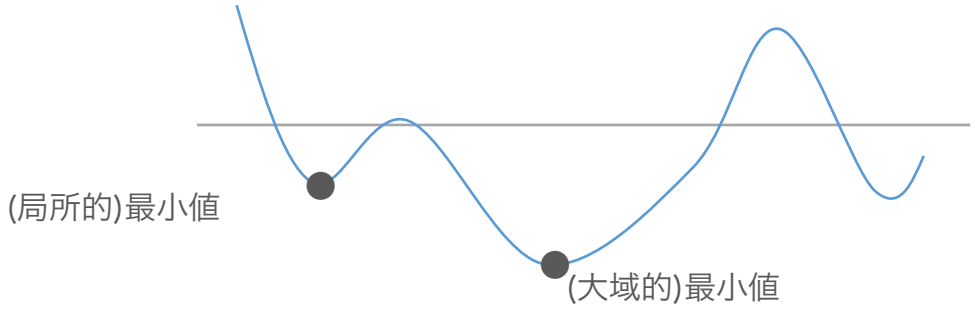
Any Questions ?

最急降下法

最小値の探索

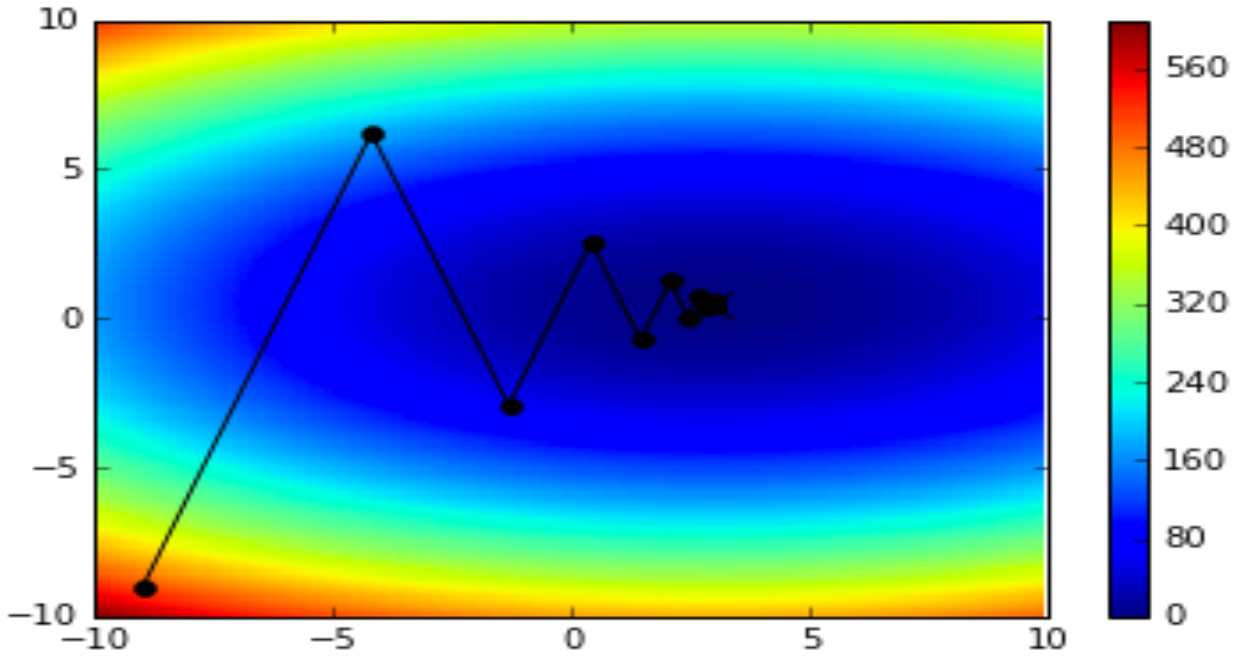
- 以下のような関数がある時、どうやって最小値を求めればいいでしょうか？





最急降下法

- 最急降下法とは、関数の勾配が最も急な方向に探索の方向を取りながら最小点にたどり着く方法。



最急降下法の例え

- [最急降下法の例え]
  - ある登山家が山で遭難しました。
  - 食料が尽き、一刻も早く下山したい状況です。
  - しかも、日が沈んでしまいました。
  - 懐中電灯は、持っていますが、電池が切れかけです。よって、電池は常時ONにできず、必要なときだけONにするしかありません。ここがポイント！
  - なるべく短い時間で山を下るにはどうすればいいでしょうか？

最急降下法における更新式

- 最急降下法によって最小値を探索する際の更新式を以下に示す。

$$\underline{x} = x - \eta \frac{\partial f}{\partial x}$$

更新後のx

$$\underline{y} = y - \eta \frac{\partial f}{\partial y}$$

更新後のy

$f(x, y)$：関数
x, y ：変数
η：学習率(1回あたりの更新の程度を制御する係数)

[演習] 最急降下法の実装

- 2_3_gradient_descent_trainee.ipynb
  - 最急降下法を実装しましょう。

Any Questions ?

勾配法

勾配法

- 勾配法とは、勾配ベクトルを用いて、関数の最小値または最大値となる場所を探索する方法。
- NNの学習では、一般的に、確率的勾配降下法を用いる。

勾配法の分類

勾配法	勾配上昇法	最大値を求める勾配法の総称。 NNの学習では使わない。
	勾配降下法	最小値を求める勾配法の総称。 勾配降下法のうち、最も急な方向へ下る方法が最急降下法。
	確率的勾配降下法 (SGD)	無作為に選びだしたデータに対して行う勾配降下法。 NNの学習では主にこれを使う。

[演習] 勾配を求める関数の実装

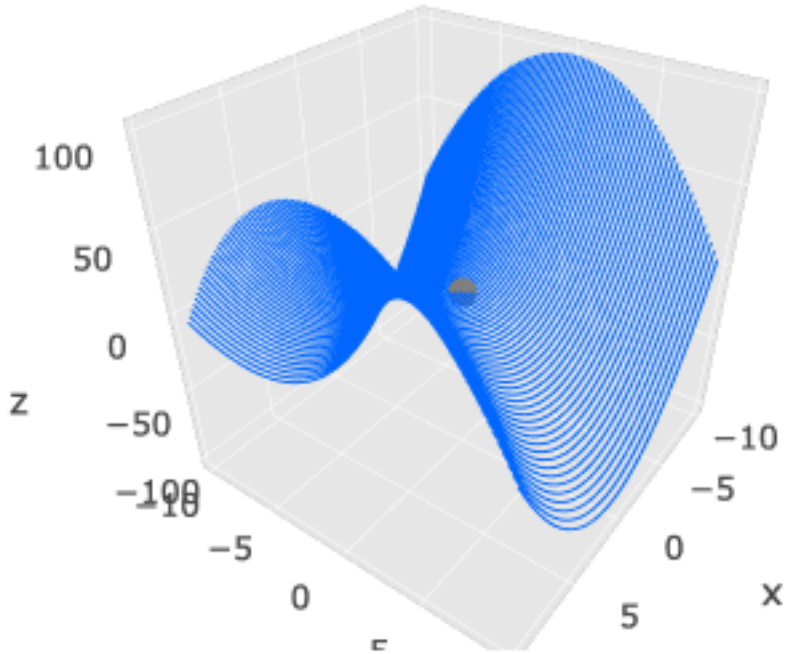
- 2_4_gradient_trainee.ipynb
  - 勾配を求める関数を実装しましょう。

[演習] 勾配を求める関数(numerical_gradient)を2次元へ拡張

- 2_5_gradient2D_trainee.ipynb
  - 勾配を求める関数(numerical_gradient)を2次元へ拡張しましょう。

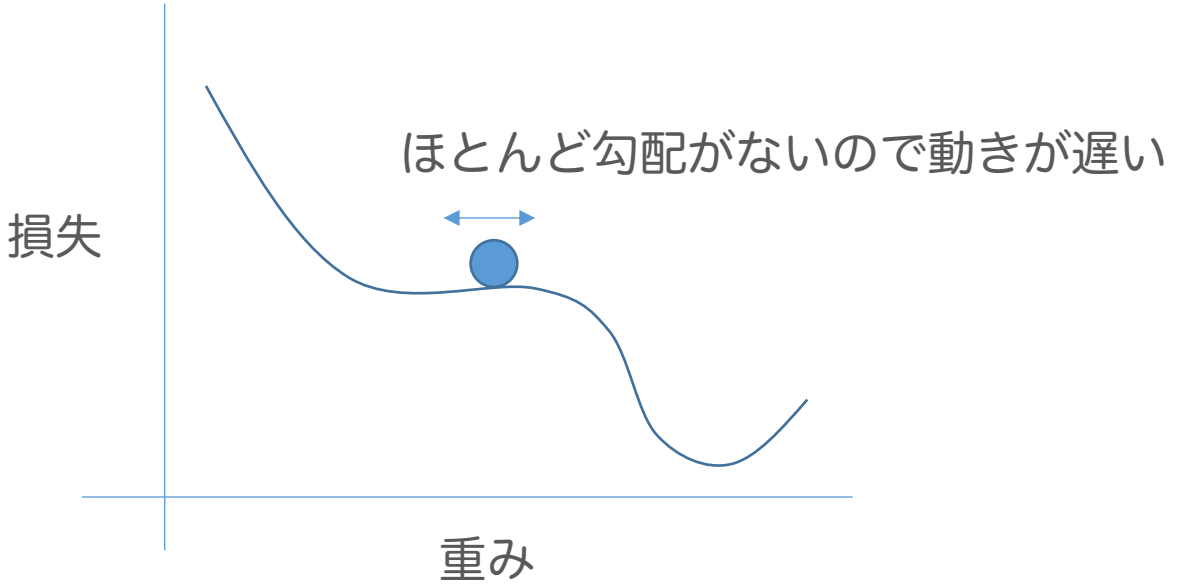
鞍点(あんてん)

- 鞍点（あんてん、saddle point）とは、大局的最小値でないにも関わらず勾配が0になる場所のこと。勾配が0になると最小値の探索が終了してしまう。
- 鞍点の例を以下に示す。



プラトー

- プラトー(plateau)とは、ほとんど勾配のない地帯のこと。学習が進まない停滞期に陥ることになる。
- プラトーの例を以下に示す。



[演習] 鞍点・プラトーの可視化

- 2_6_saddlepoint.ipynb
  - 鞍点を可視化しましょう。
  - プラトーを可視化しましょう。

[演習] NNにおいて勾配を求めるクラスの実装

- 2_7_simple_NeuralNetwork_trainee.ipynb
  - シンプルなNNにおいて勾配を求めるクラスを実装しましょう。
- 2_8_two_layer_NeuralNetwork_trainee.ipynb
  - 2層のNNにおいて勾配を求めるクラスを実装しましょう。

[演習] NNにおけるパラメータ更新部分の実装

- 2_9_update_weight_trainee.ipynb
  - NNにおけるパラメータ更新部分を実装しましょう。

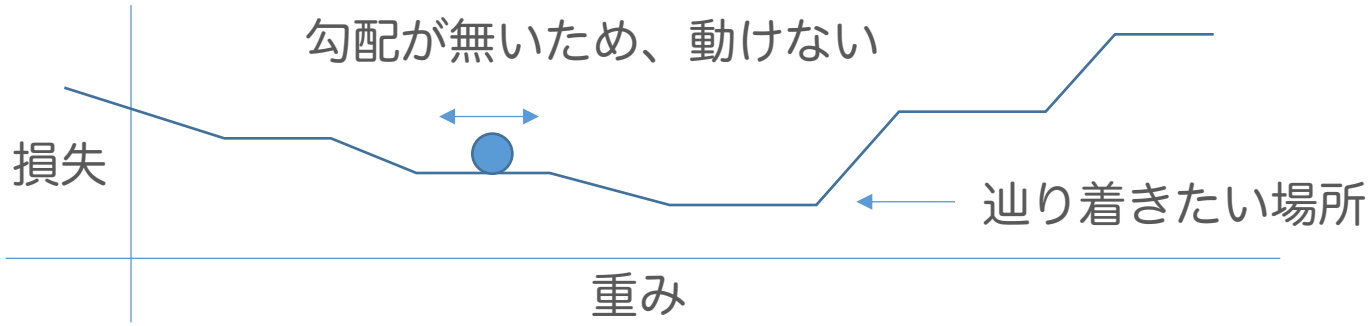
[問] 損失関数を設定する意味について考える

- 識別(クラス分類)問題において、誤識別率を損失関数に設定するとどうなるでしょうか？

[答] 損失関数を設定する意味について考える

- 例えば、誤識別率を損失関数に設定すると、学習中の損失は $\frac{6}{10}, \frac{5}{10}, \frac{4}{10}$ など離散的な値になり、勾配0の領域が多数発生することになる。
- これにより、計算が円滑に進まなくなる。

離散的な損失関数を用いた場合の探索イメージ



- クロスエントロピー誤差関数などの連続的な値を出力する損失関数に設定すると、損失の変化が滑らかになり、重みの探索が円滑になる。

[グループワーク] 鞍点やプラトーについて

- 鞍点やプラトーに入っているということは、どうやって知ることができるでしょうか？
- 鞍点やプラトーをうまく回避するにはどうすればいいでしょうか？　色々なアイデアを出してみましょう。
- 2~3名のグループに分かれて、上記テーマを話し合ってみましょう。(15分)
- 最後に、グループごとに発表していただきます。 (15分)

[演習] 2層NNを学習するクラスの実装

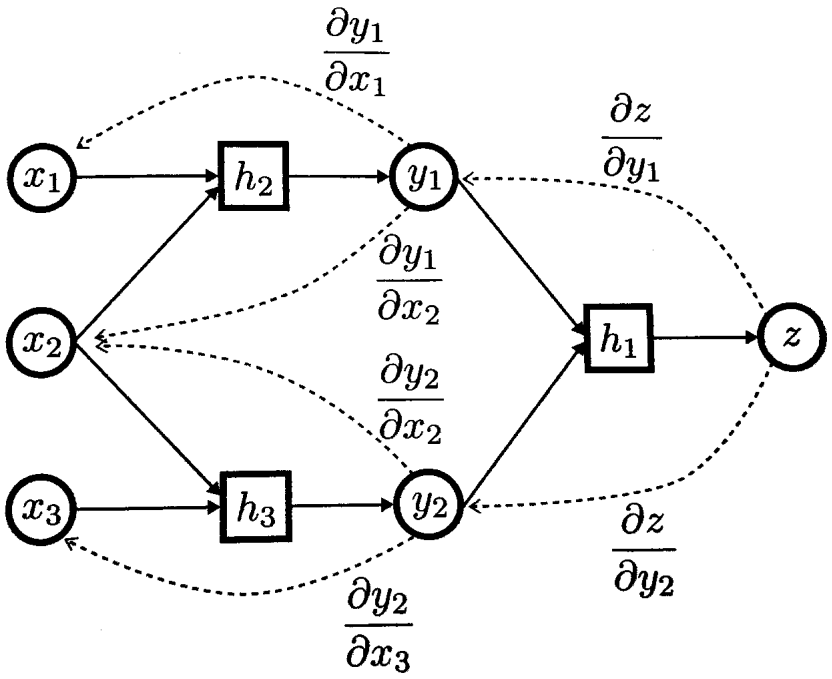
- 2_10_two_layer_NeuralNetwork_mnist_trainee.ipynb
  - 2層NNを学習するクラスを実装しましょう。
  - 以前に実装したミニバッチ学習を使ってください。
  - MNIST問題を解き、計算の進行と共に損失が小さくなっていくことを確認しましょう。
- 2_11_two_layer_NeuralNetwork_regression_trainee.ipynb
  - 2層NNを学習するクラスを実装しましょう。
  - 単純な回帰問題を解き、計算の進行と共に損失が小さくなっていくことを確認しましょう。

Any Questions ?

誤差逆伝播法

誤差逆伝播法

- 勾配降下法で最小値を求めるには各変数の微分値(偏微分値)が必要。
- 誤差逆伝播法は、出力層から入力層へと向かって導関数を伝播させていく方法。
- 連鎖律の原理を用いる。



$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1}\frac{\partial y_1}{\partial x_1}$$

誤差逆でんぱ法

- 誤差逆でんぱ法が正しい名称。誤差逆でんぱん法ではない。
- 漢字では、誤差逆伝播法が正しい表記。誤差逆伝搬法ではない。
- そもそも、伝搬(でんぱん)は伝播(でんぱ)の誤記誤読であり、もともと存在しない言葉らしい。現代では、いろんなところで使われているいるが。

連鎖律の原理とは

- 微分法において連鎖律（れんさりつ、英: chain rule）とは、複数の関数が合成された合成関数を微分するとき、その導関数がそれぞれの導関数の積で与えられるという原理のこと。

  Wikipedia https://ja.wikipedia.org/ 2018年1月5日 (金) 20:39 （日時はUTC）

連鎖律の原理

- [合成関数の微分に関する問題]
  - 以下の関数があります。

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

  - x_1に関するyの微分を求めましょう。

$$\frac{\partial y}{\partial x_1}$$

  - x_2に関するyの微分を求めましょう。

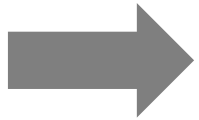
$$\frac{\partial y}{\partial x_2}$$

連鎖律の原理

- [合成関数の微分に関する問題]
  - 以下の式があります。

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

元々の式は、2つの関数が合成されたものと考える。



$$y = t^2$$
$$t = x_1 + x_2$$

これにより連鎖律の原理を適応できるようになる。

  - x_1に関するyの微分を求めましょう。

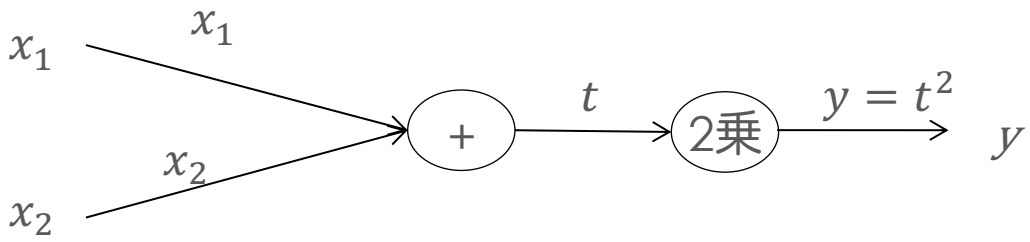
$$\frac{\partial y}{\partial x_1} = \frac{\partial y}{\partial t}\frac{\partial t}{\partial x_1} = 2t \times 1 = 2(x_1 + x_2)$$

  - x_2に関するyの微分を求めましょう。

$$\frac{\partial y}{\partial x_2} = \frac{\partial y}{\partial t}\frac{\partial t}{\partial x_2} = 2t \times 1 = 2(x_1 + x_2)$$

連鎖律の説明でよくでてくる合成微分の計算グラフ表現

- さきほどの合成関数の微分に関する問題をグラフで表現してみる。



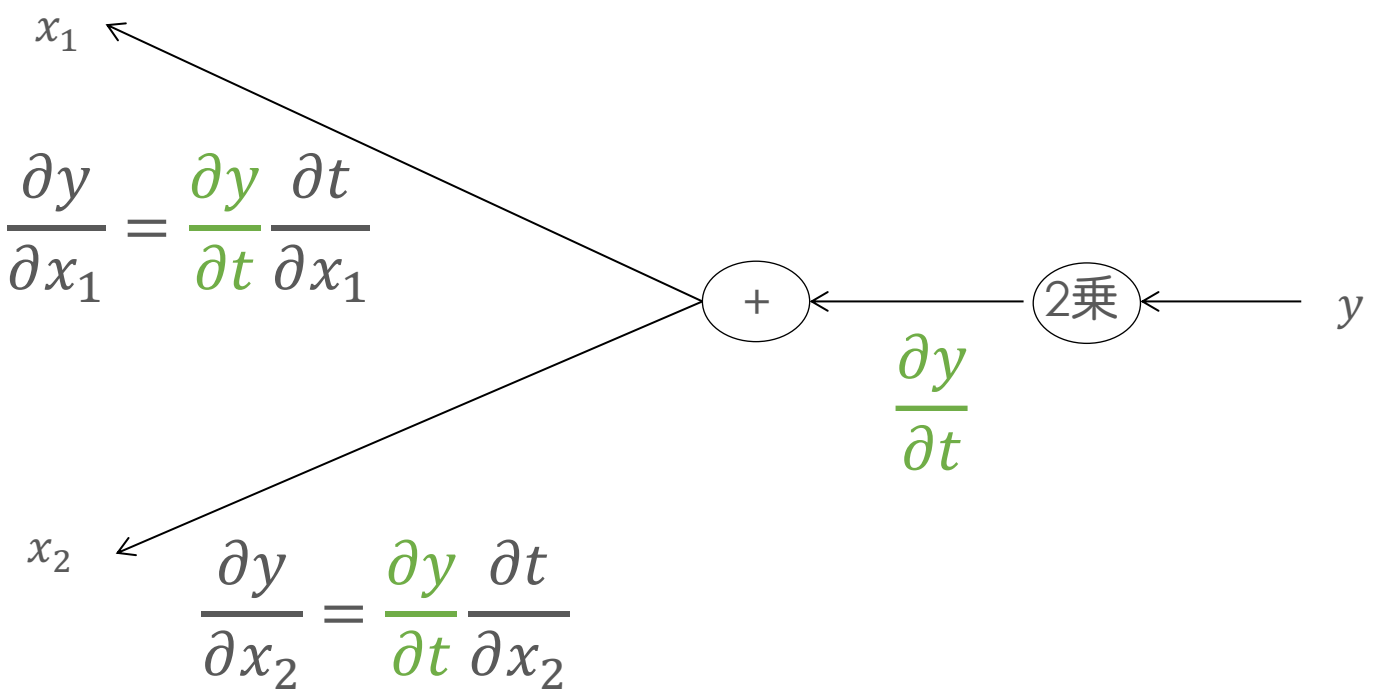
$$y = (x_1 + x_2)^2$$

- 知りたい微分は、

$$\frac{\partial y}{\partial x_1} \quad と \quad \frac{\partial y}{\partial x_2}$$

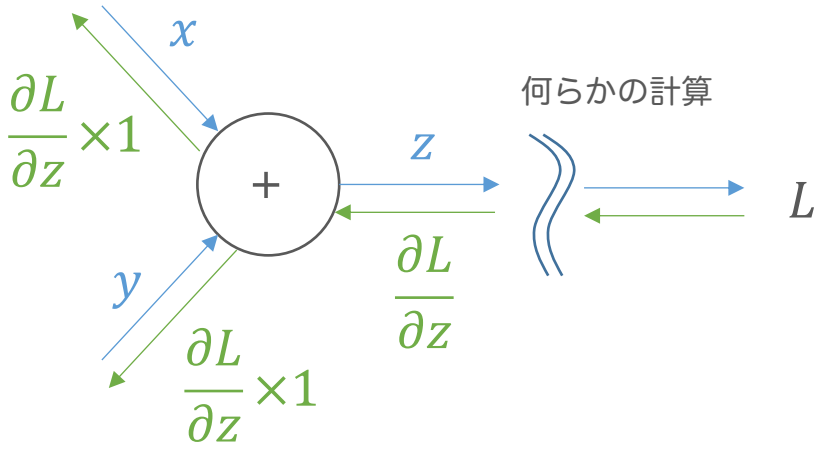
連鎖律の説明でよくでてくる合成微分の計算グラフ表現

- グラフに合成関数の微分の計算を載せると、勾配が逆に伝わっていることがわかる。



加算ノードの逆伝播

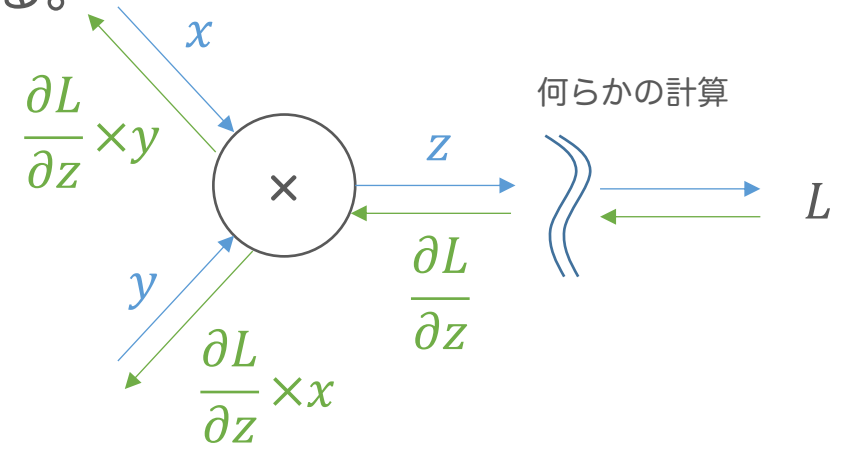
- $z = x + y$ という数式について考える。
- x, yそれぞれについてのzの偏微分は1 になる。 $\frac{\partial z}{\partial x} = 1, \quad \frac{\partial z}{\partial y} = 1$
- 順伝播の最終値がLの場合、勾配法で必要なのは、$\partial L / \partial x$と$\partial L / \partial y$。
- 連鎖律によって、 $\partial L / \partial x$と$\partial L / \partial y$を求める。 $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x} = \frac{\partial L}{\partial z} \times 1, \quad \frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y} = \frac{\partial L}{\partial z} \times 1$
- よって、加算ノードでは、上流(出力)側の勾配がそのまま下流(入力)側に伝わることになる。



※L は、スカラー。
以降も同様

乗算ノードの逆伝播

- $z = x \times y$という数式について考える。
- x, yそれぞれについてのzの偏微分は右記になる。 $\frac{\partial z}{\partial x} = y, \quad \frac{\partial z}{\partial y} = x$
- 順伝播の最終値がLの場合、勾配法で必要なのは、$\partial L / \partial x$と$\partial L / \partial y$。
- 連鎖律によって、 $\partial L / \partial x$と$\partial L / \partial y$を求める。 $\frac{\partial L}{\partial x} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial x} = \frac{\partial L}{\partial z} \times y, \quad \frac{\partial L}{\partial y} = \frac{\partial L}{\partial z}\frac{\partial z}{\partial y} = \frac{\partial L}{\partial z} \times x$
- よって、乗算ノードでは、上流(出力)側の勾配にxとyをひっくり返した値をかけたものが下流(入力)側に伝わることになる。

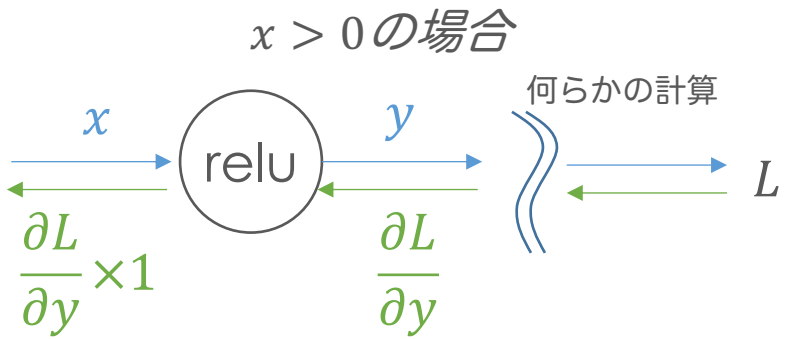


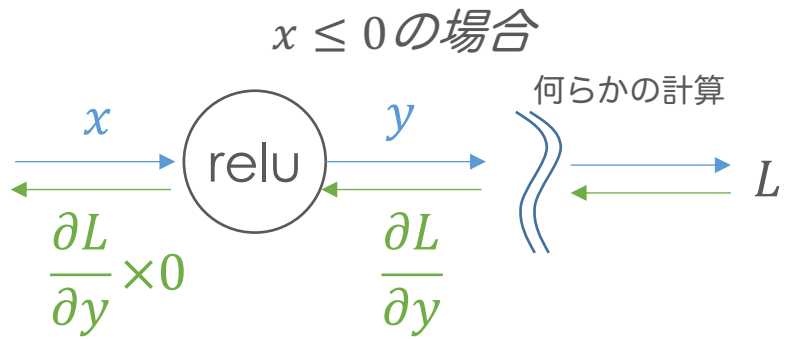
ReLUレイヤの逆伝播

- 活性化関数として使われるReLUは、右記の計算を行う。 $y = \begin{cases} x\ (x > 0) \\ 0\ (x \leq 0) \end{cases}$
- xに関するyの微分は、右記になる。 $\frac{\partial y}{\partial x} = \begin{cases} 1\ (x > 0) \\ 0\ (x \leq 0) \end{cases}$
- 順伝播の最終値がLの場合、勾配法で必要なのは、$\partial L / \partial x$。
- 連鎖律によって、 $\partial L / \partial x$を求める。

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial L}{\partial y} \times 1\ (x > 0),$$

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial L}{\partial y} \times 0 (x \leq 0)$$





ReLUレイヤの逆伝播

- ReLU関数において、 xに関するyの微分は以下になる。

$$\frac{\partial y}{\partial x} = \begin{cases} 1 \ (x > 0) \\ 0 \ (x \leq 0) \end{cases}$$

- $(x > 0)$の場合、ReLU関数の微分(実際には勾配)は1である。逆伝播計算では勾配が何度も掛け合わされていくが、勾配が1であれば、何度掛け合わされても1であるため、勾配が消失することがない。

シグモイドレイヤの逆伝播

- 活性化関数として使われるシグモイド関数は、右記の通り。 $y = \frac{1}{1 + \exp(-x)}$
- xに関するyの微分は、以下のようになる。

$t = 1 + \exp(-x)$ とおくと、$y = \frac{1}{t} = t^{-1}$

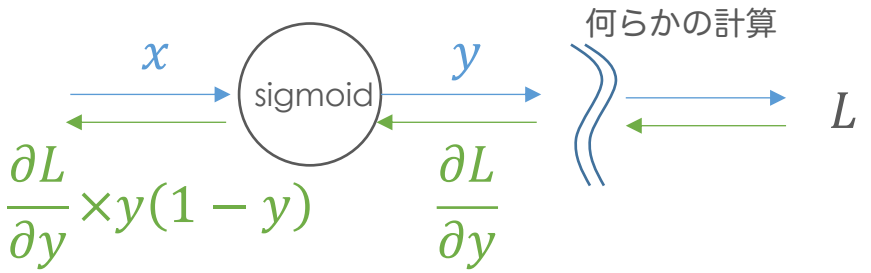
$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial t}\frac{\partial t}{\partial x} = -t^{-2} \times -\exp(-x) = y^2 \times \exp(-x)$$

$$= \left(\frac{1}{1 + \exp(-x)}\right)^2 \times \exp(-x) = \frac{1}{1 + \exp(-x)} \times \frac{\exp(-x)}{1 + \exp(-x)}$$

$$= y\left(\frac{1 + \exp(-x) - 1}{1 + \exp(-x)}\right) = y(1 - y)$$

- 順伝搬の最終値がLの場合、勾配法で必要なのは、$\partial L / \partial x$。
- 連鎖律によって、 $\partial L / \partial x$を求める。

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial L}{\partial y} \times y(1 - y)$$



シグモイドレイヤの逆伝播

- シグモイド関数において、xに関するyの微分は、以下のようになる。

$$\frac{\partial y}{\partial x} = y(1-y)$$

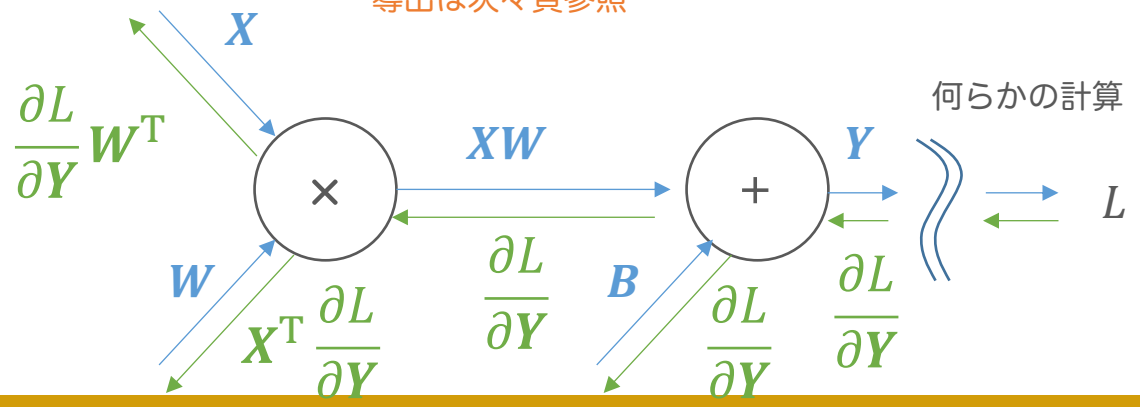
- y の最大値は1(正確には1にはならない)であるため、 $y(1-y)$の最大値は0.25となる。
- 逆伝播では勾配が何度も掛け合わされていくため、仮に最大の0.25をとったとしても、入力層に近づくにつれ値がどんどん小さくなる。これを勾配消失という。
- シグモイド関数を活性化関数に用いると、勾配消失が起きやすくなる。

アフィン(Affine)レイヤの逆伝播

- 通常のNNにおける中間層レイヤについて考える。ここでは、アフィンレイヤと呼ぶことにする。アフィンとは幾何学用語。行列積にバイアスを足すことがアフィン変換に相当する。
- 通常のNNにおける中間層レイヤでは、右記のように結合される。$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W} + \boldsymbol{B}$
- $\boldsymbol{X}$についての$\boldsymbol{Y}$の偏微分は右記になる。 $\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}} = \boldsymbol{W}^{\mathrm{T}}$ 導出は次々頁参照
- 順伝播の最終値がLの場合、勾配法で必要なのは、$\partial L/\partial \boldsymbol{X}$と$\partial L/\partial \boldsymbol{W}$。
- 連鎖律によって、 $\partial L/\partial \boldsymbol{X}$と$\partial L/\partial \boldsymbol{W}$を求める。 $\frac{\partial L}{\partial \boldsymbol{X}} = \frac{\partial L}{\partial \boldsymbol{Y}}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}} = \frac{\partial L}{\partial \boldsymbol{Y}}\boldsymbol{W}^{\mathrm{T}}$, $\frac{\partial L}{\partial \boldsymbol{W}} = \boldsymbol{X}^{\mathrm{T}}\frac{\partial L}{\partial \boldsymbol{Y}}$

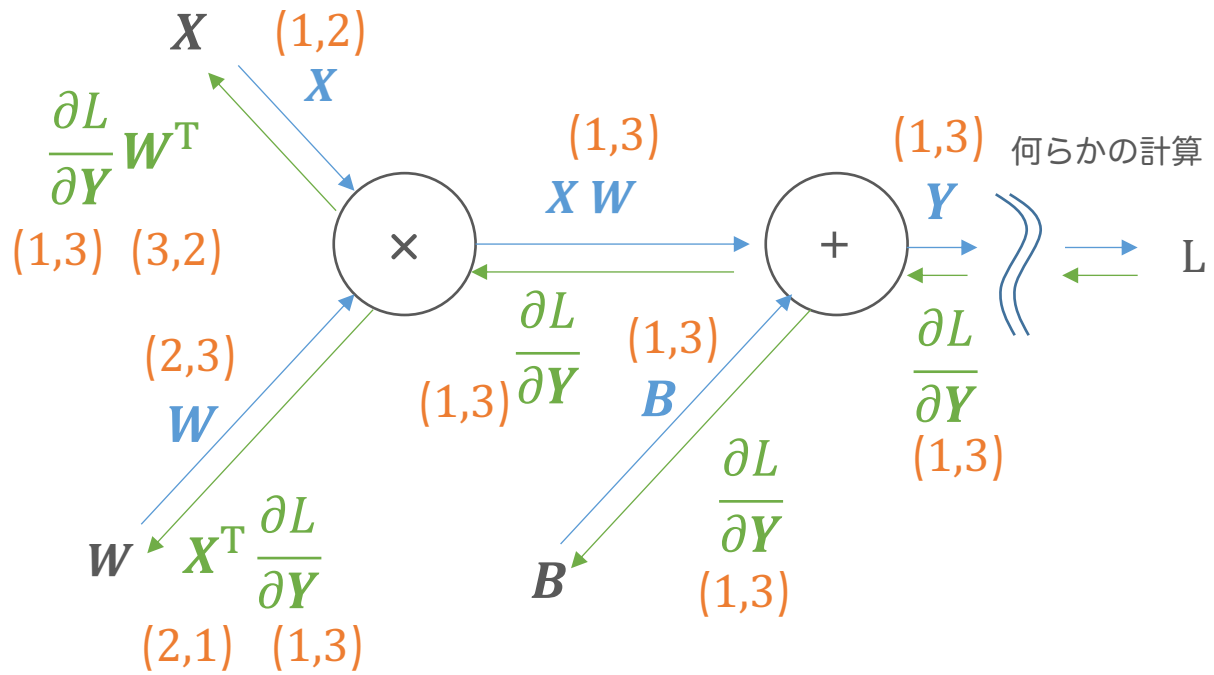
導出は次々頁参照

- よって、乗算ノードでは、上流(出力)側の勾配に$\boldsymbol{X}$と$\boldsymbol{Y}$をひっくり返した値をかけたものが下流(入力)側に伝わることになる。



アフィンレイヤの逆伝播

- ここでのX, B, Y, Wは、ベクトルではなく行列で考える。
- 入力が2次元、出力が3次元の場合の例を以下に示す。



アフィンレイヤの逆伝播

$\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{B}, \boldsymbol{W}$は、ベクトルではなく行列

$$\boldsymbol{Y} = \boldsymbol{XW} + \boldsymbol{B} = (x_1 \quad x_2)\begin{pmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \end{pmatrix} + \boldsymbol{B} = \begin{pmatrix} w_{11}x_1 + w_{21}x_2 \\ w_{12}x_1 + w_{22}x_2 \\ w_{13}x_1 + w_{23}x_2 \end{pmatrix}^{\mathrm{T}} + (b_1 \quad b_2 \quad b_3 \quad)$$

$$\boldsymbol{Y} = (y_1 \;\; y_2 \;\; y_3) = \begin{pmatrix} w_{11}x_1 + w_{21}x_2 + b_1 \\ w_{12}x_1 + w_{22}x_2 + b_2 \\ w_{13}x_1 + w_{23}x_2 + b_3 \end{pmatrix}^{\mathrm{T}}$$

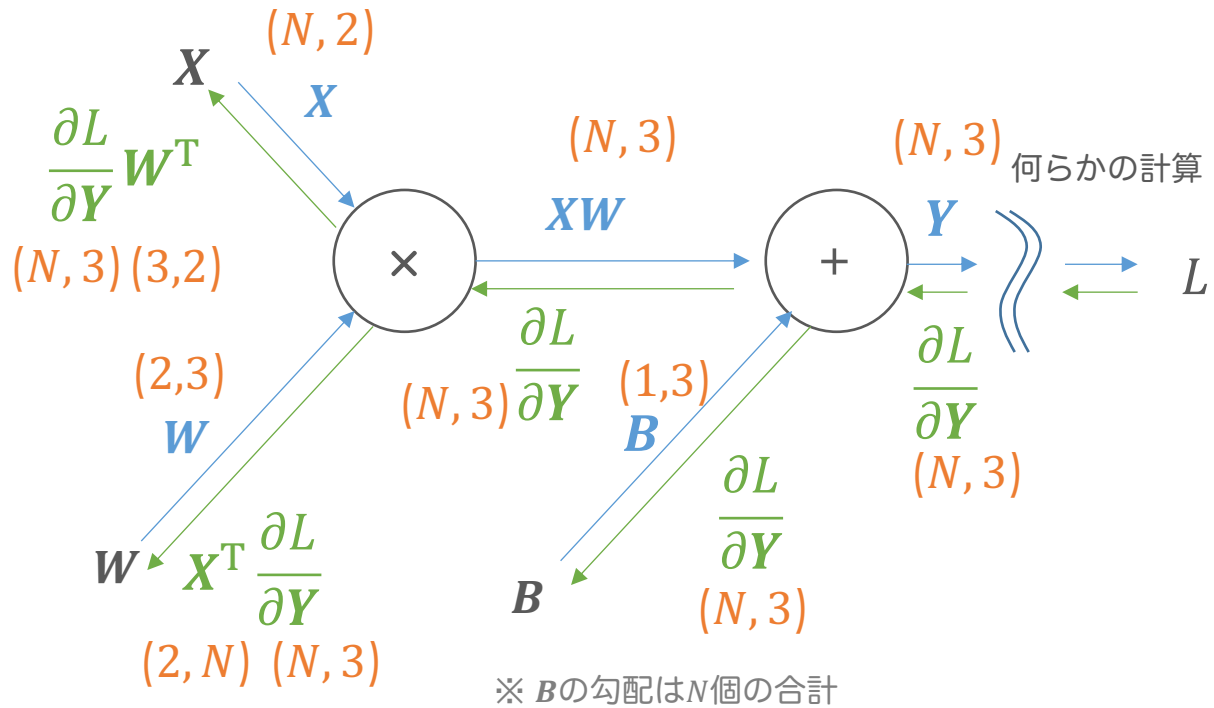
$$\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \\ \frac{\partial y_3}{\partial x_1} & \frac{\partial y_3}{\partial x_2} \end{pmatrix} = \begin{pmatrix} w_{11} & w_{21} \\ w_{12} & w_{22} \\ w_{13} & w_{23} \end{pmatrix} = \boldsymbol{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \boldsymbol{X}} = \frac{\partial L}{\partial \boldsymbol{Y}}\frac{\partial \boldsymbol{Y}}{\partial \boldsymbol{X}} = \frac{\partial L}{\partial \boldsymbol{Y}}\mathbf{W}^{\mathrm{T}}$$

$$\frac{\partial L}{\partial \boldsymbol{W}} = \begin{pmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{12}} & \frac{\partial L}{\partial w_{13}} \\ \frac{\partial L}{\partial w_{21}} & \frac{\partial L}{\partial w_{22}} & \frac{\partial L}{\partial w_{23}} \end{pmatrix} = \begin{pmatrix} \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial w_{11}} & \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial w_{12}} & \frac{\partial L}{\partial y_3}\frac{\partial y_3}{\partial w_{13}} \\ \frac{\partial L}{\partial y_1}\frac{\partial y_1}{\partial w_{21}} & \frac{\partial L}{\partial y_2}\frac{\partial y_2}{\partial w_{22}} & \frac{\partial L}{\partial y_3}\frac{\partial y_3}{\partial w_{23}} \end{pmatrix} = \begin{pmatrix} \frac{\partial L}{\partial y_1}x_1 & \frac{\partial L}{\partial y_2}x_1 & \frac{\partial L}{\partial y_3}x_1 \\ \frac{\partial L}{\partial y_1}x_2 & \frac{\partial L}{\partial y_2}x_2 & \frac{\partial L}{\partial y_3}x_2 \end{pmatrix} = \boldsymbol{X}^{\mathrm{T}}\frac{\partial L}{\partial \boldsymbol{Y}}$$

アフィンレイヤの逆伝播(バッチ対応版)

- バッチに対応させるアフィンレイヤは、入力がN次元になる。



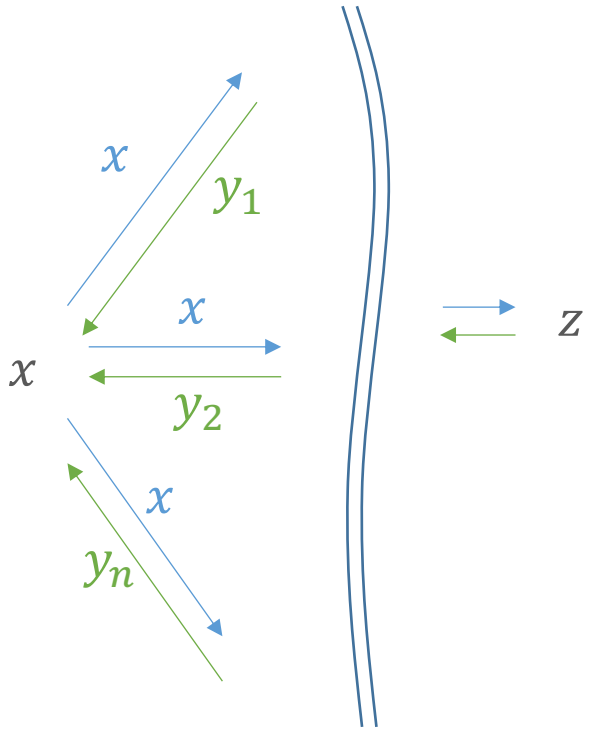
※ $\boldsymbol{B}$の勾配はN個の合計

※ $\boldsymbol{W}$の勾配も$\boldsymbol{B}$の勾配と同様、 N個の合計になっている。この合計は行列計算の中で行われる。

アフィンレイヤの逆伝播(バッチ対応版)

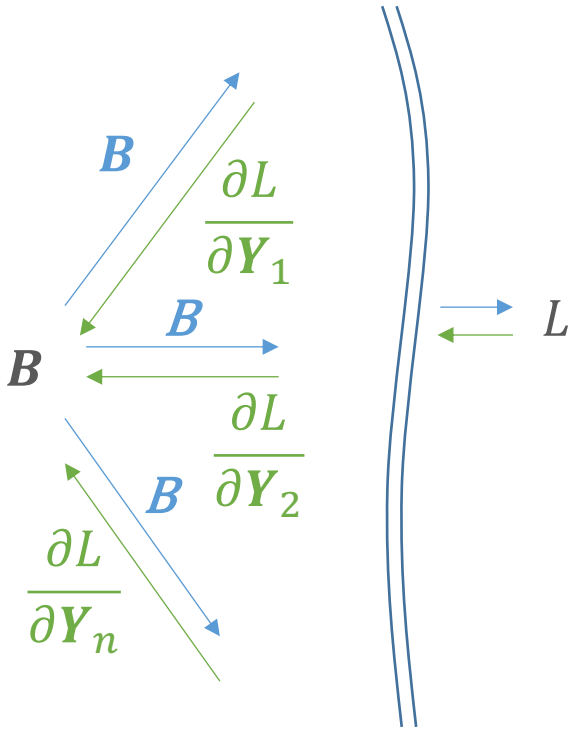
- $\boldsymbol{B}$の勾配がN個の合計になる理由

連鎖率のルール



$$\frac{\partial z}{\partial x} = y_1 + y_2 + \dots y_n$$

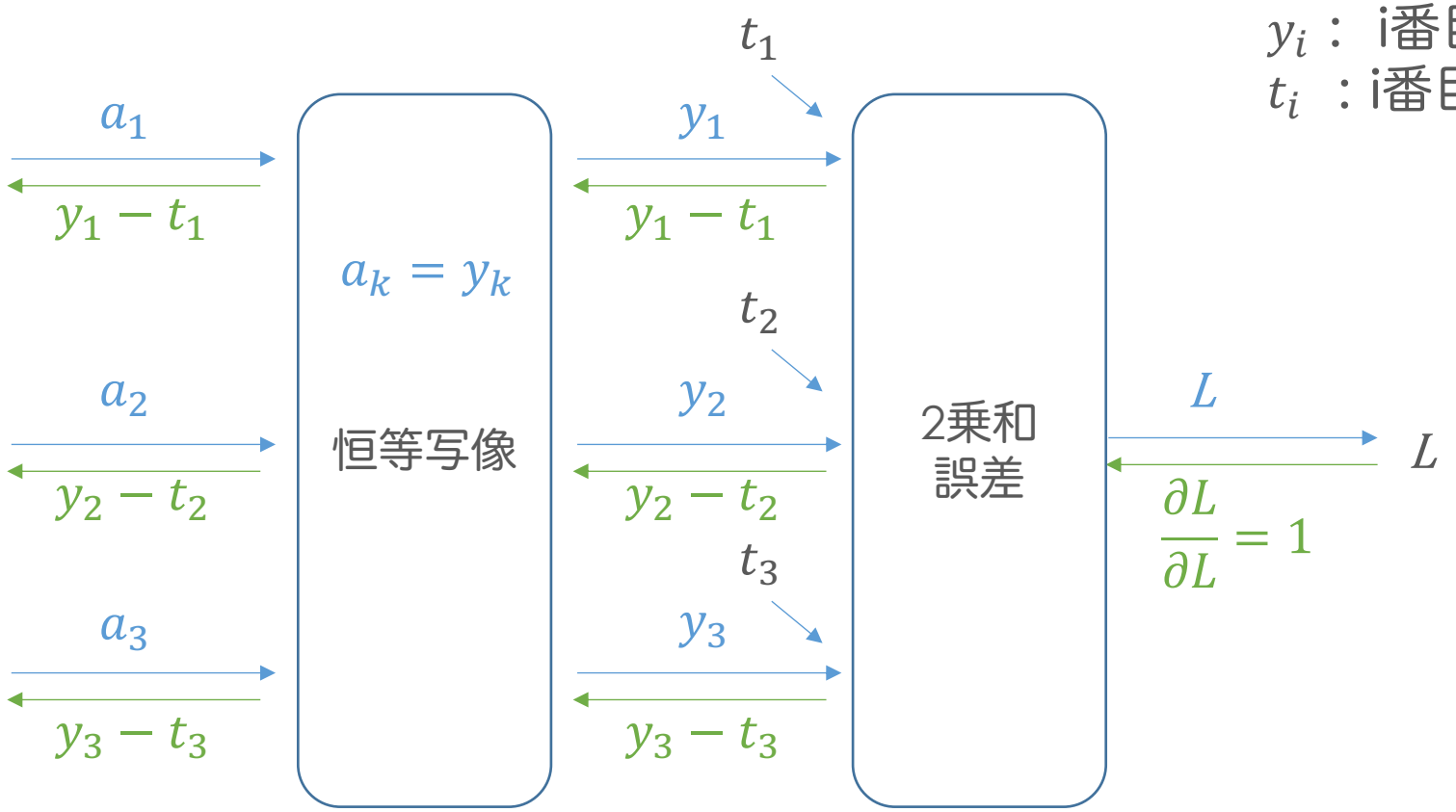
バッチ数Nの時の$\boldsymbol{B}$の勾配



$$\frac{\partial L}{\partial \boldsymbol{B}} = \frac{\partial L}{\partial \boldsymbol{Y}_1} + \frac{\partial L}{\partial \boldsymbol{Y}_2} + \cdots \frac{\partial L}{\partial \boldsymbol{Y}_n}$$

2乗和誤差レイヤの逆伝播

- 2乗和誤差レイヤでの逆伝播は、以下のようになる。



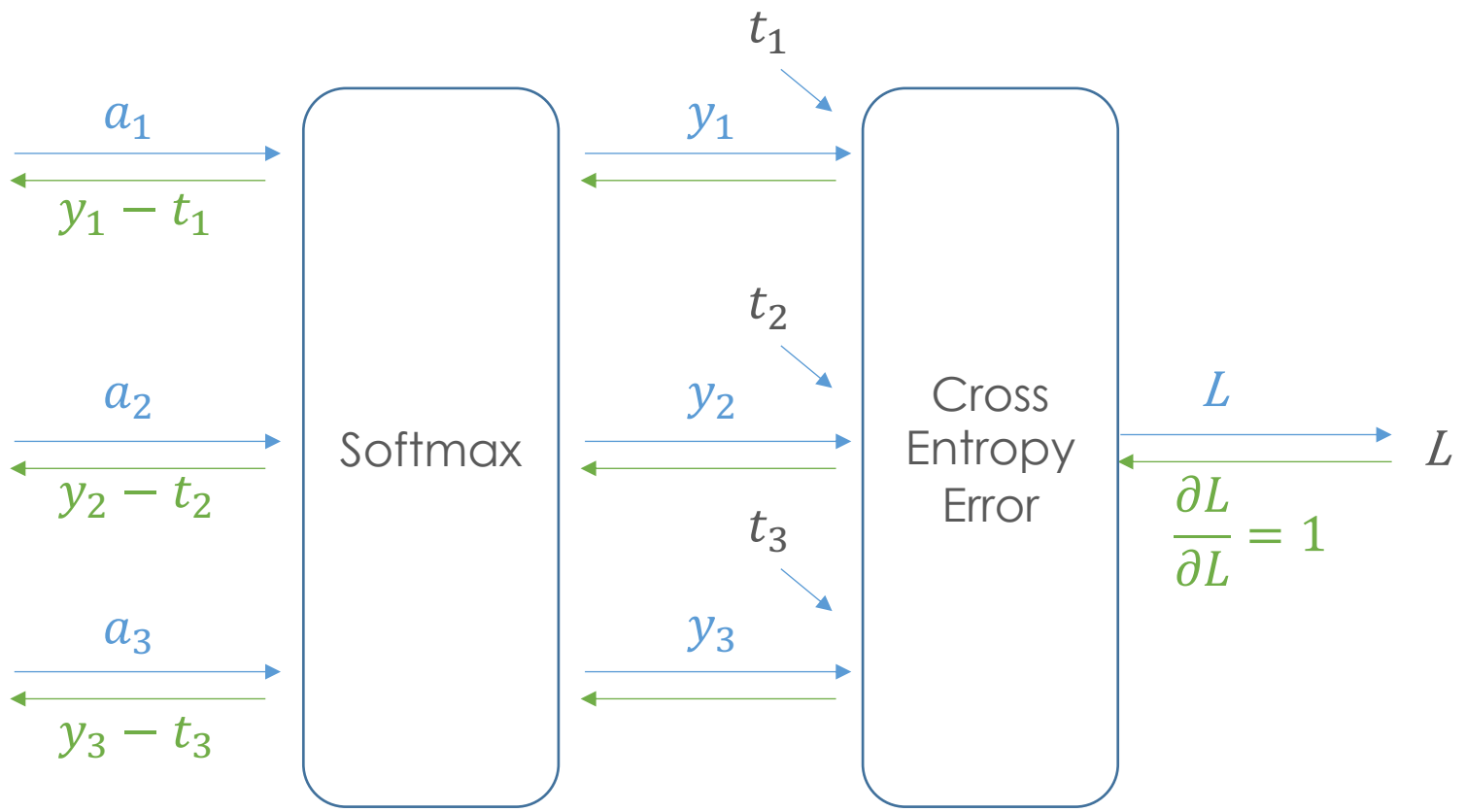
2乗和誤差

$$L = \frac{1}{2}\sum_{i=1}^{k}(y_i - t_i)^2$$

L：損失
k：出力層のノード数
y_i： i番目のノードの出力値
t_i ：i番目のノードの正解値

Softmax_with_Lossレイヤの逆伝播(結果のみ表示)

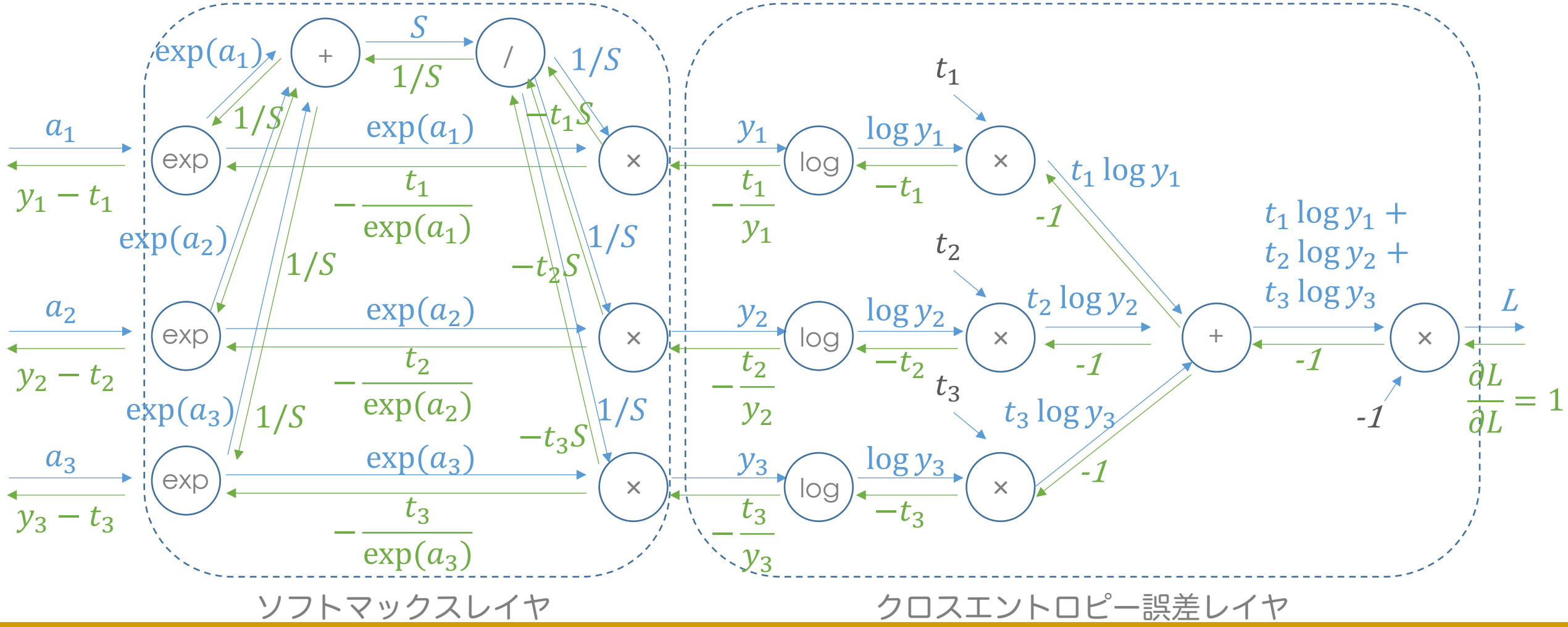
- ソフトマックス関数とクロスエントロピー誤差関数をまとめて実装する。



Softmax_with_Lossレイヤの逆伝播(詳細版)

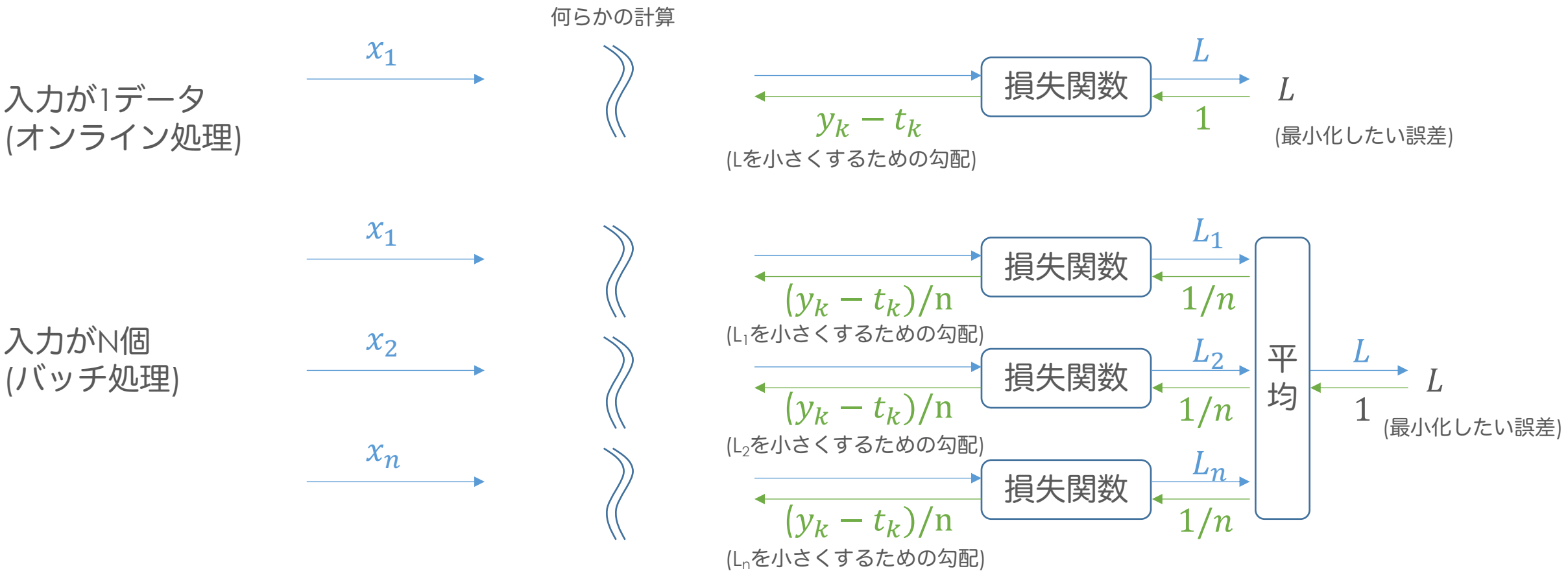
$$S = \exp(a_1) + \exp(a_2) + \exp(a_3)$$

- ソフトマックス関数とクロスエントロピー誤差関数をまとめて実装する。



損失関数レイヤにおける誤差逆伝播計算時の注意点

- バッチ処理において、損失関数レイヤでの誤差逆伝播を行うとき、バッチ数で割る必要がある。



[演習] 各種レイヤの実装

- 2_12_layers_online_trainee.ipynb
  - 加算レイヤを実装しましょう。
  - 乗算レイヤを実装しましょう。
  - ReLUレイヤを実装しましょう。
  - シグモイドレイヤを実装しましょう。
- 2_13_layers_batch.ipynb
  - 2_12_layers_online_trainee.ipynbで実装した各種レイヤがバッチ処理に対応していることを確認しましょう。

[演習] アフィンレイヤの実装

- 2_14_affine_trainee.ipynb
  - アフィンレイヤを実装しましょう。

[演習] Softmax_with_Lossレイヤの実装

- 2_15_softmax_with_loss_trainee.ipynb
  - Softmax_with_Lossレイヤを実装しましょう。

[演習] 誤差逆伝播に対応したNNの実装

- 2_16_NN_with_backpropagation_trainee.ipynb
  - 誤差逆伝播に対応したNNを実装しましょう。

誤差逆伝播法の勾配確認

- 誤差逆伝播法を用いると、微分を効率的に求めることができる。
- よって、NNの計算では、数値微分は使わずに、誤差逆伝播法を用いる。
- ただし、誤差逆伝播法の実装は複雑になるためミスが起きやすい。
- そこで、数値微分の結果と誤差逆伝播法の結果を比較することによって、誤差逆伝播法の実装が正しいことを確認することが行われる。
- この作業のことを勾配確認という。

[演習] 勾配の確認

- 2_17_check_gradient.ipynb
  - 数値微分で求めた勾配と誤差逆伝播で求めた勾配の差が小さいことを確認しましょう。

[演習] 誤差逆伝播を使った学習の実装

- 2_18_train_with_backpropagation.ipynb
  - 誤差逆伝播を使った学習を実装しましょう。

Any Questions ?

[グループワーク]

- 2乗和誤差レイヤでの逆伝播は、 $y_k - t_k$になることを確認しました。
- 本当に$y_k - t_k$になるかどうか、詳細な計算グラフを描いて確認しましょう。
  - 計算グラフのノードは、加算ノードや乗算ノード以外のものを用いても構いません。
  - ただしそれらノードは、微分可能なものである必要があります。
  - 例えば、2乗ノード、逆数ノード、減算ノードなどが考えられます。
- 2~3名のグループに分かれて、上記テーマに取り組んでみましょう。(15分)
- 最後に、グループごとに発表していただきます。 (15分)

[グループワーク]

- 活性化関数として使われるtanhの逆伝播はどのようになるでしょうか？
- 詳細な計算グラフを描いて確認しましょう。
  - 計算グラフのノードは、加算ノードや乗算ノード以外のものを用いても構いません。
  - ただしそれらノードは、微分可能なものである必要があります。
  - 例えば、2乗ノード、逆数ノード、減算ノードなどが考えられます。
- 最後に、 $y = \tanh(x)$ とした場合、 $\mathrm{d}y/\mathrm{d}x$がyを使って表せることを確認しましょう。
- 2~3名のグループに分かれて、上記テーマに取り組んでみましょう。(30分)
- 最後に、グループごとに発表していただきます。 (15分)

Any Questions ?

講座の時間が余ったら

- 今回の復習をします。
- 次回の予習をします。