機械学習講習会

第四回：誤差逆伝播法

2025/07/02

@Kobakos32

前回の復習

1. **ニューラルネットワークの基本構造**：線形層と活性化関数を交互に重ねた構造
   - $\text{Output} = \phi(W_n\phi(W_{n-1}\phi(\dots\phi(W_1x + b_1) + b_2)\dots) + b_n)$
2. **表現力**：万能近似定理により、十分な隠れニューロンがあればどんな連続関数でも近似可能
3. **高次元の利点**：次元の祝福により局所最適解が稀で、勾配降下法が有効

課題：複雑な関数の勾配をどう計算するか？

本日のゴール

1. **計算グラフ**を使った表現に触れる
2. **連鎖率 (Chain Rule)** を計算グラフを通じて理解する
3. **誤差逆伝播法**のアルゴリズムを理解する
4. ニューラルネットワークの各構成要素の微分を導出する

もくじ

1. なぜ誤差逆伝播法が必要なのか？

勾配降下法には微分が必要

以前学んだ勾配降下法では、損失関数 L の各パラメータに対する偏微分が必要でした：

$$w \leftarrow w - \alpha \frac{\partial L}{\partial w}$$

$$b \leftarrow b - \alpha \frac{\partial L}{\partial b}$$

線形回帰の場合：

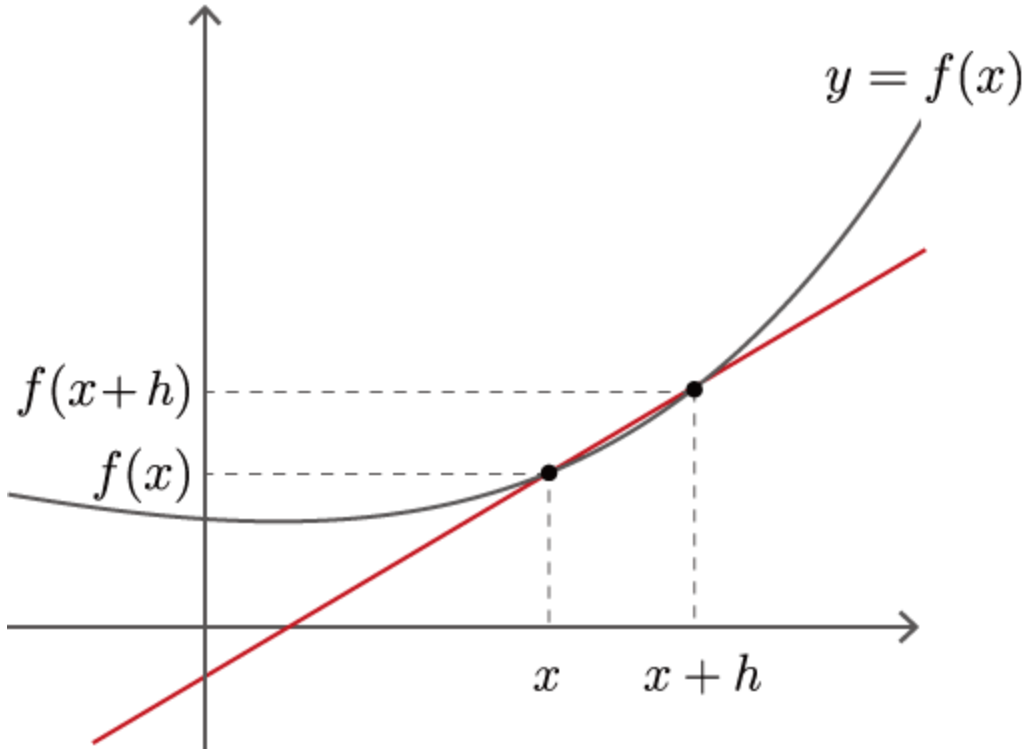
- パラメータ：w, b の2個のみ
- 損失関数：$L(w, b) = \frac{1}{N} \sum_{i=1}^{N} (y_i - (wx_i + b))^2$
- 微分：直接計算可能

数値微分では？

数値微分：

微分を求めたいパラメーターを少しだけ変化させて損失関数を求め、差分を計算する

- パラメーターの数だけ順伝播が必要
- 厳密でない
- 誤差が大きくなる可能性



ニューラルネットワークの複雑さ

ニューラルネットワークの場合：

- パラメータ：数百万〜数十億個の重みとバイアス
- 損失関数：非常に複雑な合成関数

問題：すべてのパラメータに対する偏微分を独立に計算するのは現実的でない

例：3層ニューラルネットワーク

簡単な例でも、損失関数は以下のような複雑な合成関数になります：

$$L = \text{loss}(\phi(W_3\phi(W_2\phi(W_1x + b_1) + b_2) + b_3), y)$$

$\frac{\partial L}{\partial W_1}$ を計算するには...🤔

解決策：計算グラフと連鎖率を使った効率的な微分計算

計算グラフとは？

計算グラフ：数式の計算過程を有向グラフで表現したもの

- **ノード（節点）**：演算を表す
- **エッジ（辺）**：データの流れを表す

利点：

- 複雑な計算を段階的に分解
- 中間結果の保存と再利用
- 微分の効率的な計算（あとで）

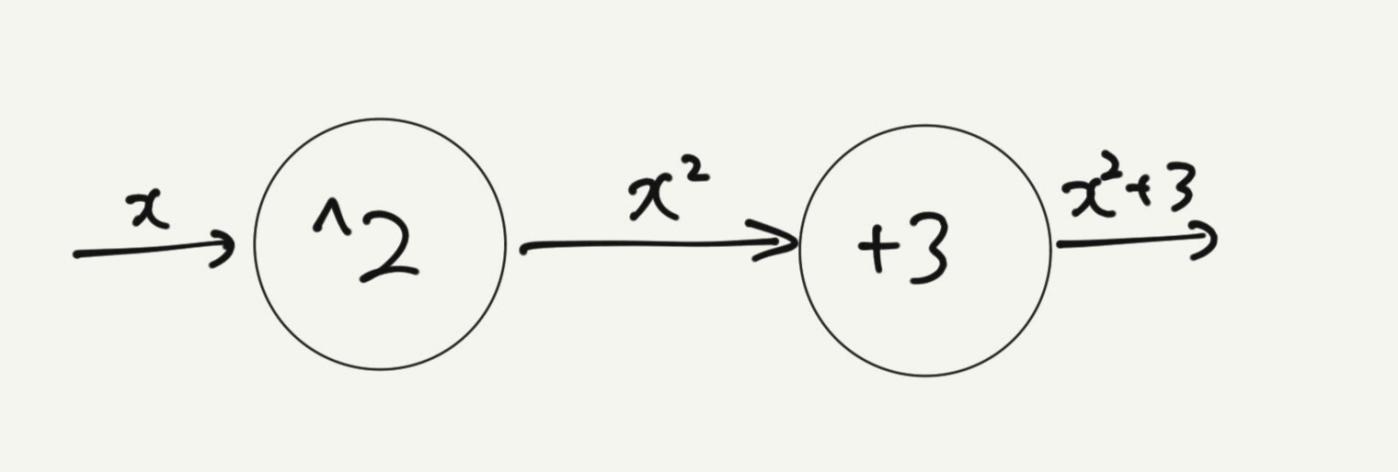
例1：$f(x) = x^2 + 3$

数式： $f(x) = x^2 + 3$

計算手順：

1. $a = x^2$
2. $f = a + 3$

計算グラフ：



例2(2変数)：$f(x, y) = (x + y) \times (x - y)$

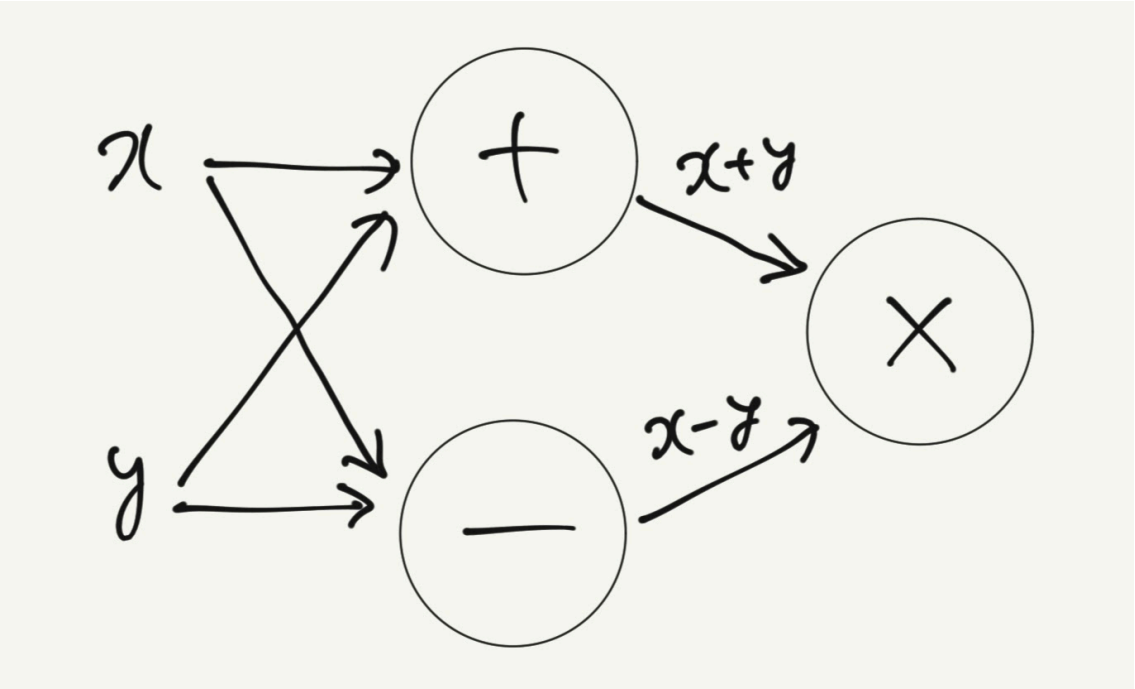
計算手順：

1. $a = x + y$
2. $b = x - y$
3. $f = a \times b$

$x = 3, y = 1$**の場合：**

- $a = 3 + 1 = 4$
- $b = 3 - 1 = 2$
- $f = 4 \times 2 = 8$

計算グラフ：



順伝播（Forward Propagation）

計算グラフの左から右への計算を**順伝播**と呼びます。

- エッジをたどりながら、ノードに従って計算を進める
- ある入力についてのモデルの予測を計算することに対応

3. 合成関数の微分と連鎖率

合成関数の微分

一変数の場合：

$f(x) = g(h(x))$（でどっちも微分可能）のとき、

$$\frac{df}{dx} = \frac{dg}{dh} \cdot \frac{dh}{dx}$$

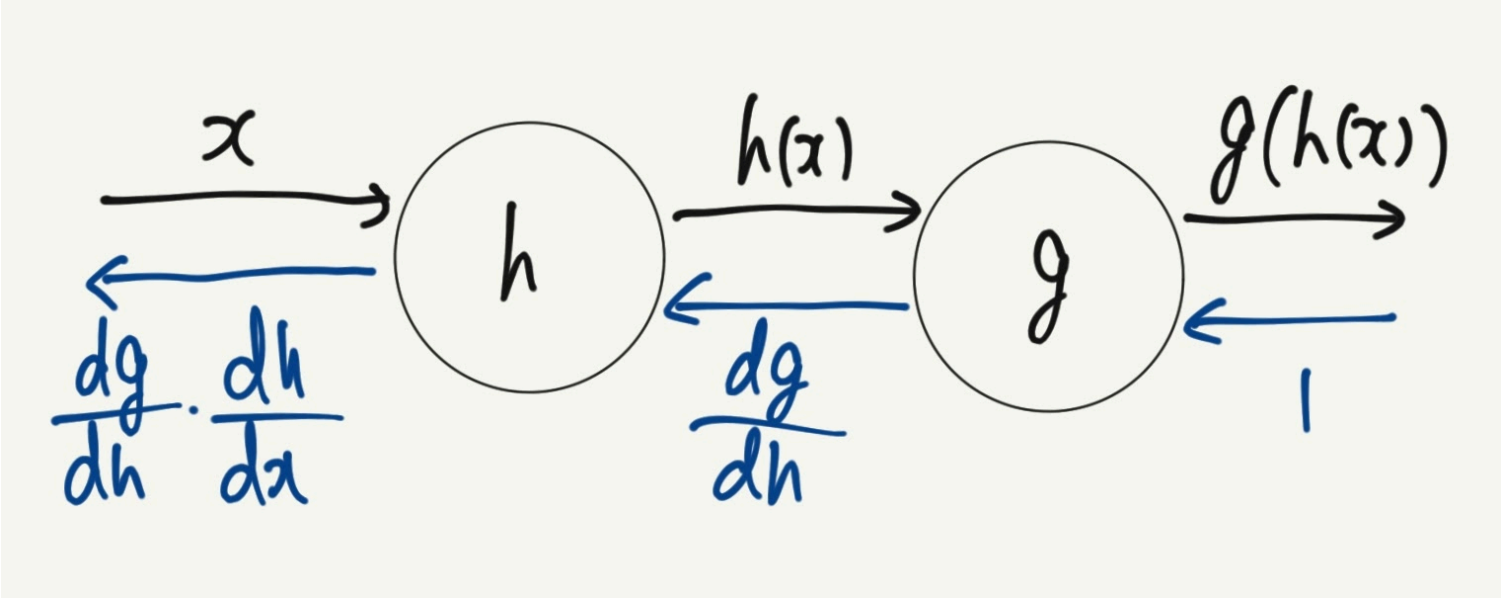
連鎖率の計算グラフによる可視化

順伝播：

- $x \to h(x) \to g(h(x)) = f(x)$

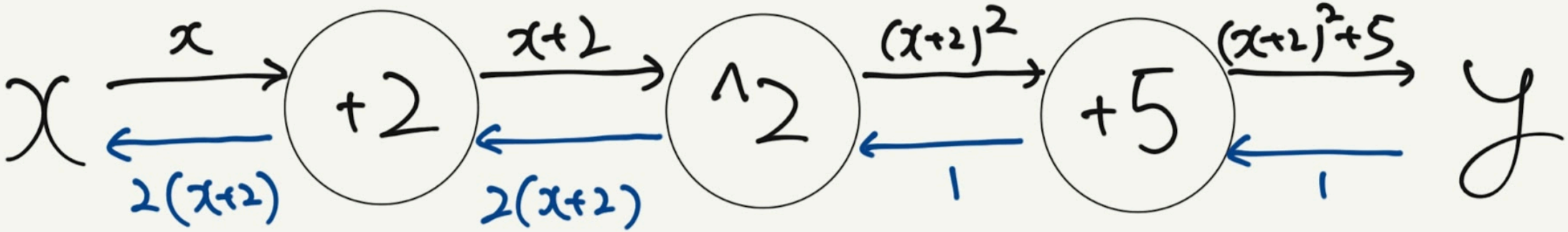
逆伝播（合成関数の微分）：

- $\frac{\partial f}{\partial x} = \frac{\partial g}{\partial h} \times \frac{\partial h}{\partial x}$



上流の微分に、そのノードの入力についての微分をかけることに対応

例：$y = (x+2)^2 + 5$ の微分



数式で確認：

$y = x^2 + 4x + 9$ なので、 $\frac{dy}{dx} = 2x + 4$ ✅

計算グラフとして考える利点

計算のモジュール化：

- 複雑な計算を小さな部品（モジュール）に分解
- 各モジュールは独立に計算可能
- グラフを定義すればあとは自動的に微分が計算可能

利点：一度実装すれば、どんな組み合わせでも再利用可能！

多変数での連鎖率

二変数の場合：

$f(x, y) = g(u(x, y), v(x, y))$ のとき、

$$\frac{\partial f}{\partial x} = \frac{\partial g}{\partial u} \cdot \frac{\partial u}{\partial x} + \frac{\partial g}{\partial v} \cdot \frac{\partial v}{\partial x}$$

重要：分岐がある場合は**和**になる

連鎖率の証明（簡単版）

$f(x,y) = g(u(x,y), v(x,y))$ について、xで偏微分すると：

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h}$$

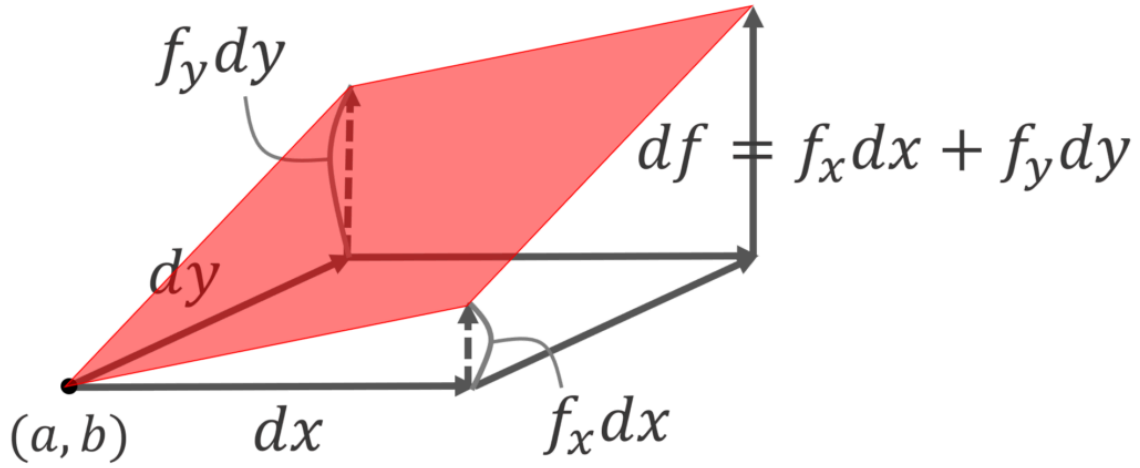
xがhだけ変化したとき、u, vの変化をそれぞれ$\Delta u, \Delta v$とすると：

$$\Delta f \approx \frac{\partial g}{\partial u} \Delta u + \frac{\partial g}{\partial v} \Delta v$$

両辺をhで割って極限を取ると：

$$\frac{\partial f}{\partial x} = \frac{\partial g}{\partial u} \cdot \lim_{h \to 0} \frac{\Delta u}{h} + \frac{\partial g}{\partial v} \cdot \lim_{h \to 0} \frac{\Delta v}{h}$$

$$= \frac{\partial g}{\partial u} \cdot \frac{\partial u}{\partial x} + \frac{\partial g}{\partial v} \cdot \frac{\partial v}{\partial x}$$



(記号は少し違いますがイメージとして)

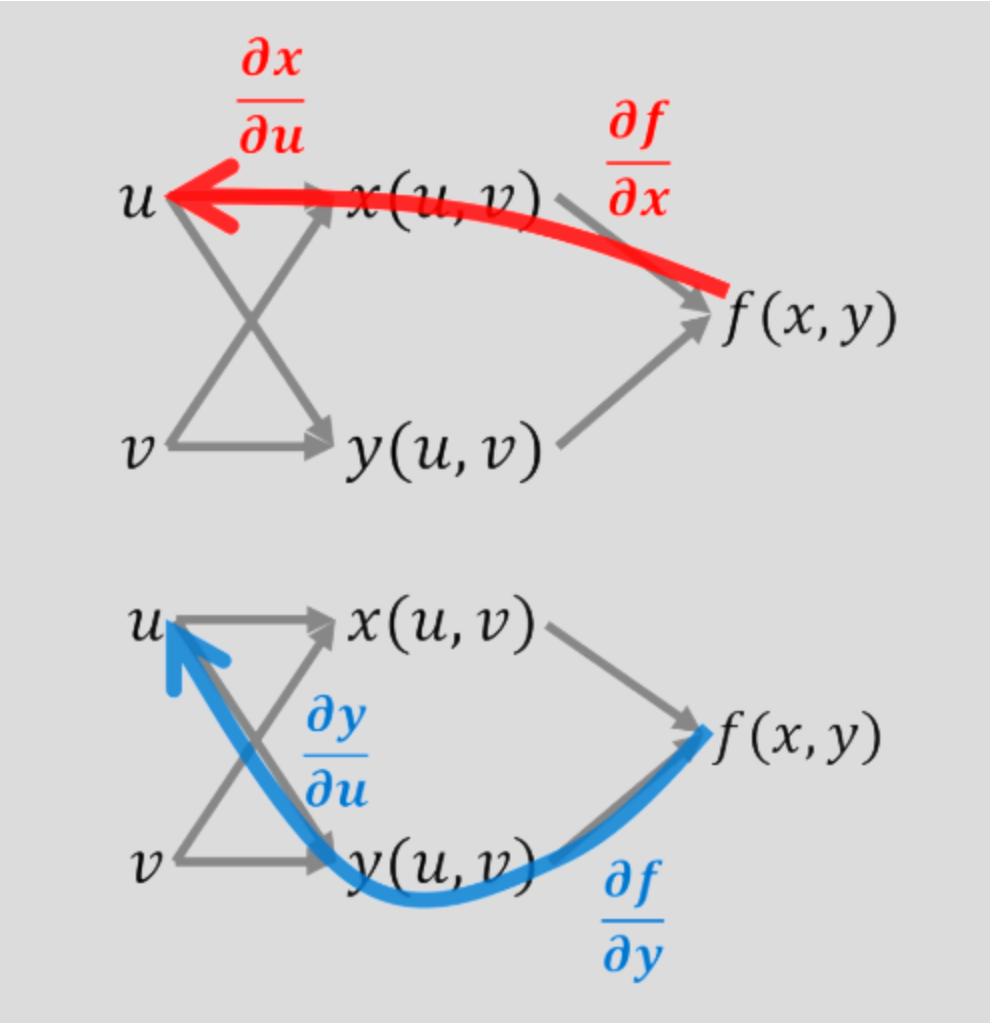
計算グラフでの連鎖率

計算グラフ上では、連鎖率は以下のルールで適用されます：

$$\frac{\partial f}{\partial x} = \frac{\partial g}{\partial u} \cdot \frac{\partial u}{\partial x} + \frac{\partial g}{\partial v} \cdot \frac{\partial v}{\partial x}$$

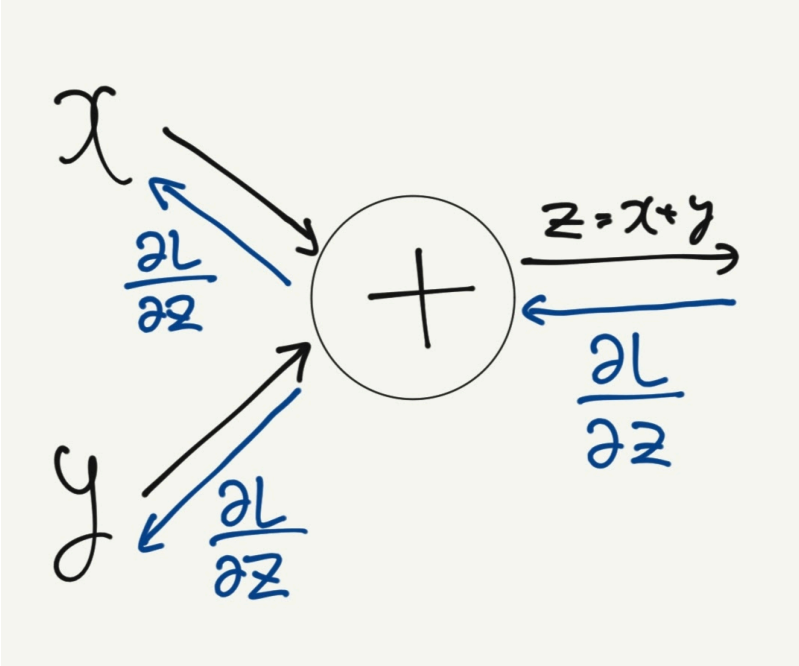
1. **直列接続**：微分を掛ける
2. **分岐**：微分を足す

分岐は和になる！！（大事なので二回）



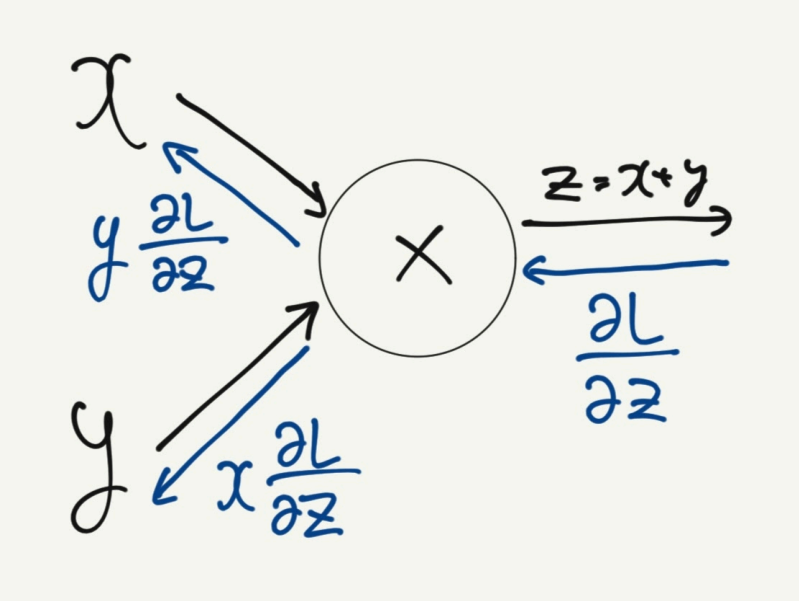
主な計算の逆伝播

- $z = x + y$
- $\frac{\partial z}{\partial x} = 1, \frac{\partial z}{\partial y} = 1$



上流の勾配をそのまま下流に伝える

- $z = x \times y$
- $\frac{\partial z}{\partial x} = y, \frac{\partial z}{\partial y} = x$

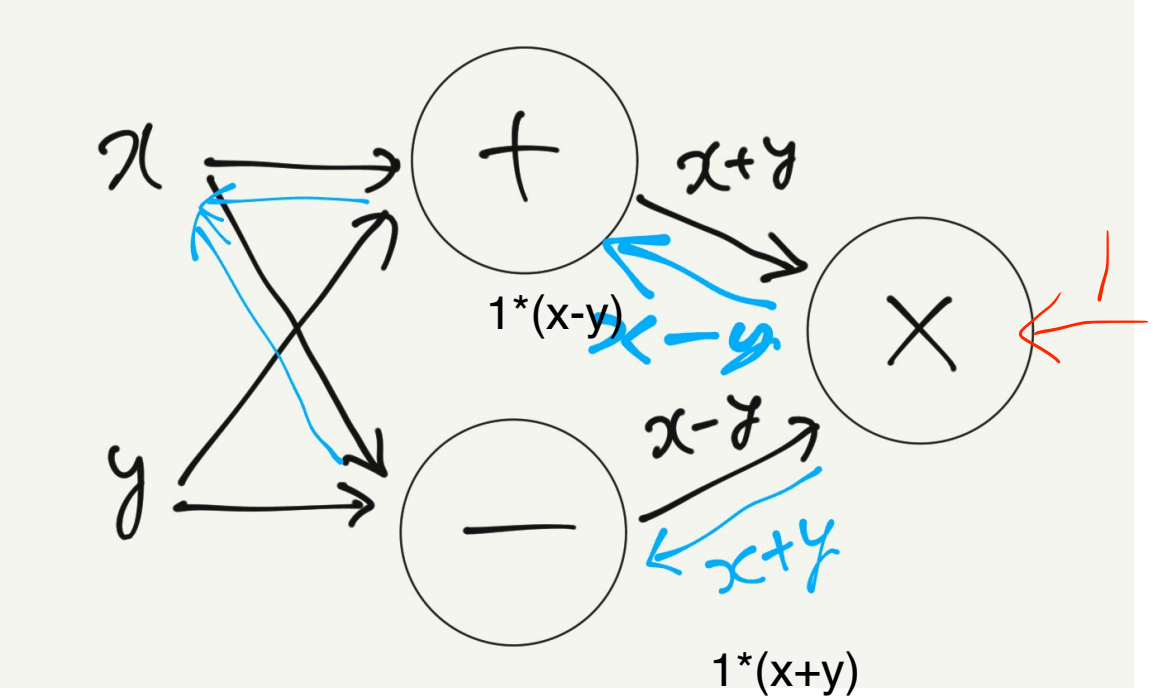


上流の勾配に他方の値を掛けて下流に伝える

例：$f(x, y) = (x + y) \times (x - y)$ の偏微分

順伝播：

- $a = x + y$
- $b = x - y$
- $f = a \times b$



逆伝播：

- $\frac{\partial f}{\partial a} = b = x - y$
- $\frac{\partial f}{\partial b} = a = x + y$

最終結果：

- $\frac{\partial f}{\partial x} = \frac{\partial f}{\partial a} \cdot \frac{\partial a}{\partial x} + \frac{\partial f}{\partial b} \cdot \frac{\partial b}{\partial x} = (x - y) \cdot 1 + (x + y) \cdot 1 = 2x$
- $\frac{\partial f}{\partial y} = \frac{\partial f}{\partial a} \cdot \frac{\partial a}{\partial y} + \frac{\partial f}{\partial b} \cdot \frac{\partial b}{\partial y} = (x - y) \cdot 1 + (x + y) \cdot (-1) = -2y$

4. 誤差逆伝播法のアルゴリズム

逆伝播（Backward Propagation）

計算グラフの右から左へとたどりながが微分を計算していく過程を**逆伝播**と呼びます。

- 出力から入力に向かって微分を計算
- 順伝播で計算した中間結果を利用
- 連鎖率を使って効率的に計算
- 微分係数を**アルゴリズミックに**計算可能

誤差逆伝播法のアルゴリズム

ステップ1：順伝播

- 入力から出力まで計算
- 中間結果を保存

ステップ2：逆伝播

- 損失関数から開始
- 連鎖率を使って各ノードの勾配を計算
- 前の層に勾配を伝播

逆伝播の計算ルール

各演算ノードでの逆伝播の計算：

初期化：一番最後のノードに入力する勾配として1を設定（$\frac{\partial L}{\partial L} = 1$）
計算ノード：上流からの勾配を受け取り、自分の入力に対する微分をかけて下流に伝える
分岐：上流からの勾配を合計

これをパラメーターのノードにたどり着くまで行う

5. ニューラルネットワークの構成要素の微分

線形層の微分

線形層：$z = Wx + b$

$z \in \mathbb{R}^m, W \in \mathbb{R}^{m \times n}, x \in \mathbb{R}^n, b \in \mathbb{R}^m$

行列積を要素ごとに書くと以下のようになります

$$z_i = \sum_j W_{ij} x_j + b_i$$

線形層の微分（成分表示）

$z_i = \sum_j W_{ij} x_j + b_i$について：

重みの勾配：

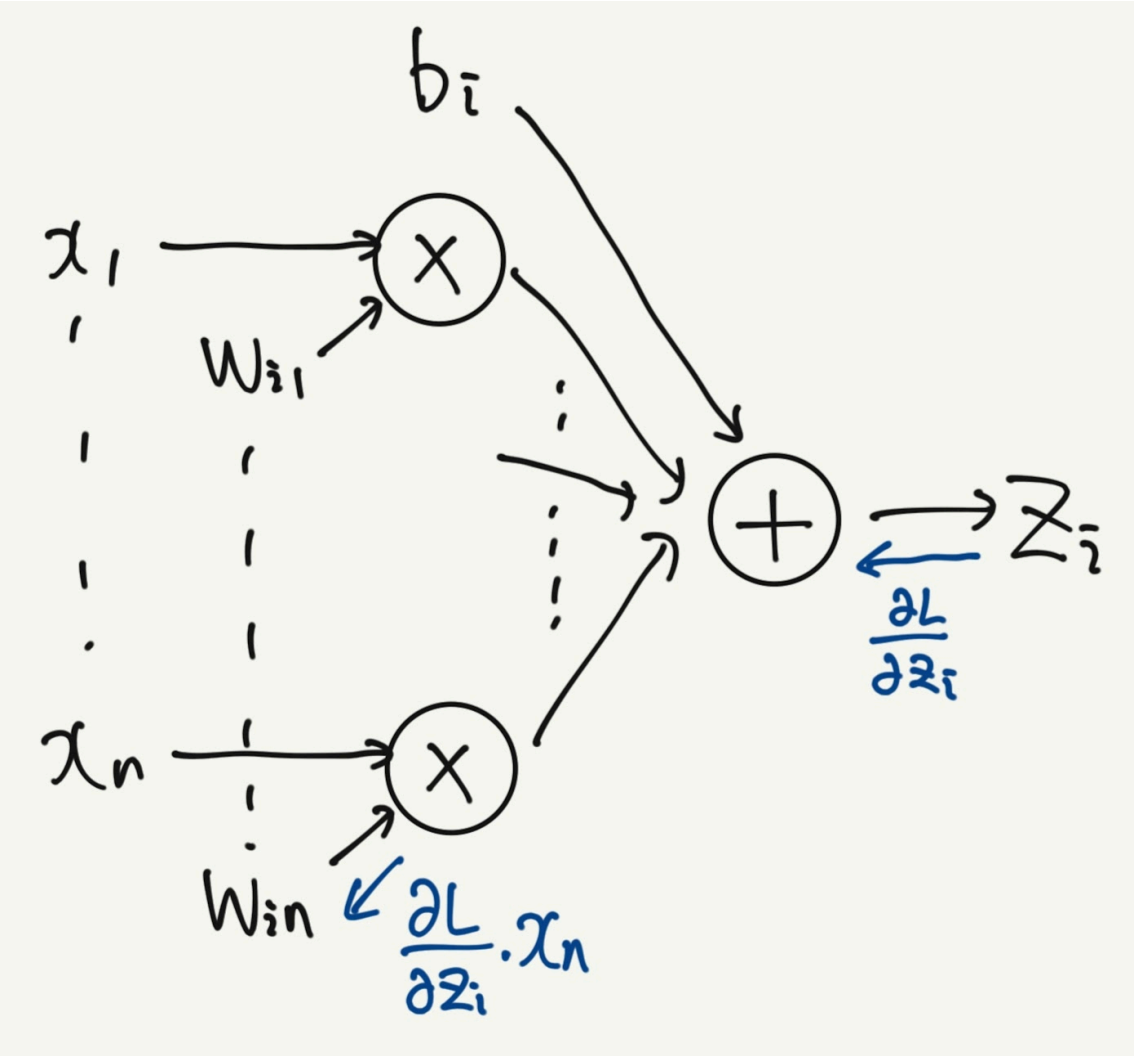
$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}} = \frac{\partial L}{\partial z_i} \cdot x_j$$

バイアスの勾配：

$$\frac{\partial L}{\partial b_i} = \frac{\partial L}{\partial z_i} \cdot 1 = \frac{\partial L}{\partial z_i}$$

入力の勾配（下流に渡すのに必要）：

$$\frac{\partial L}{\partial x_j} = \sum_i \frac{\partial L}{\partial z_i} \cdot \frac{\partial z_i}{\partial x_j} = \sum_i \frac{\partial L}{\partial z_i} \cdot W_{ij}$$



線形層の微分（行列表示）

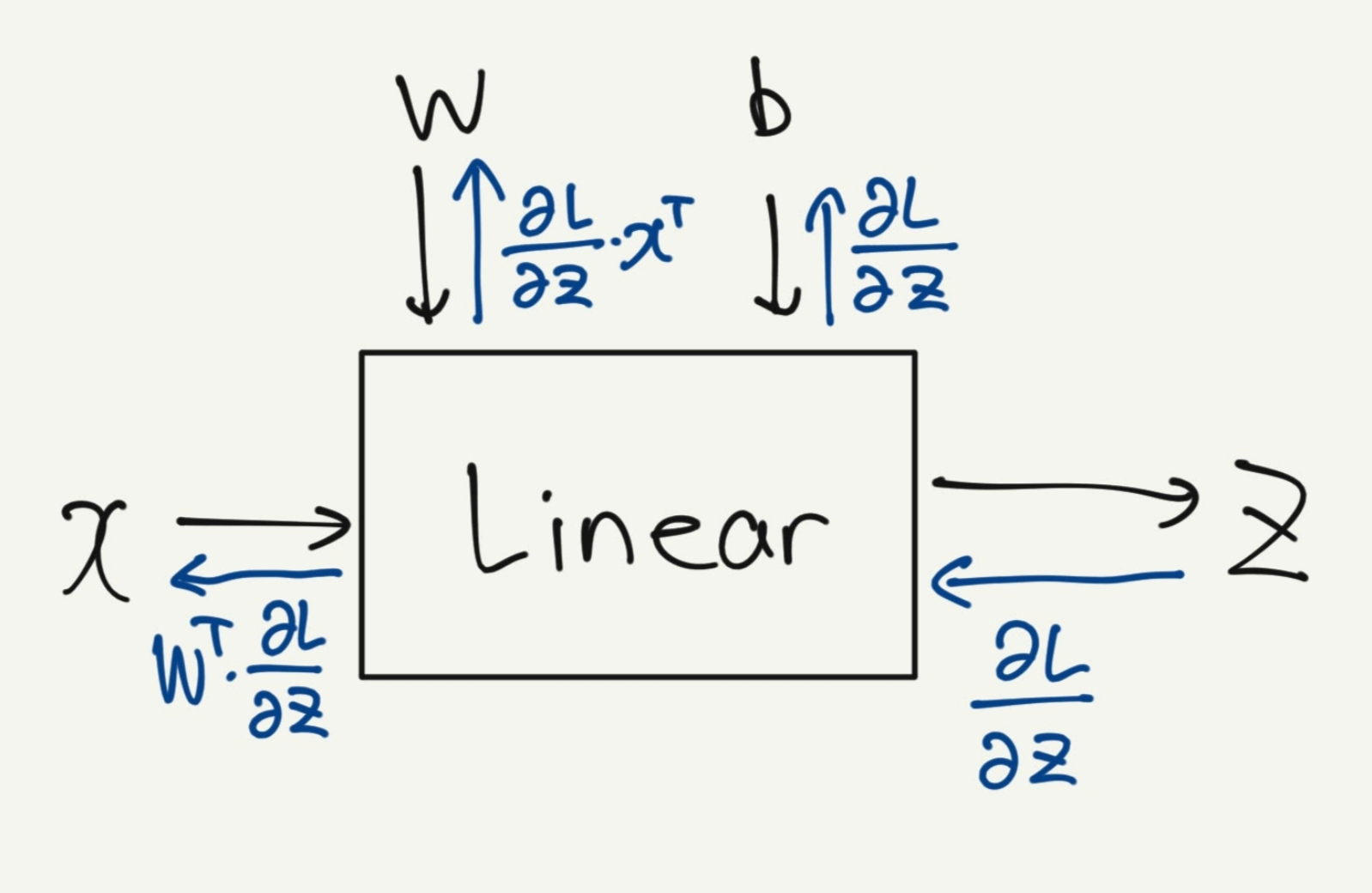
$$\frac{\partial L}{\partial W} = \begin{bmatrix} \frac{\partial L}{\partial W_{11}} & \cdots & \frac{\partial L}{\partial W_{1n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial W_{m1}} & \cdots & \frac{\partial L}{\partial W_{mn}} \end{bmatrix} = \begin{bmatrix} \frac{\partial L}{\partial z_1} x_1 & \cdots & \frac{\partial L}{\partial z_1} x_n \\ \vdots & \ddots & \vdots \\ \frac{\partial L}{\partial z_m} x_1 & \cdots & \frac{\partial L}{\partial z_m} x_n \end{bmatrix} = \frac{\partial L}{\partial z} \cdot x^T$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial z}$$

$$\frac{\partial L}{\partial x} = W^T \cdot \frac{\partial L}{\partial z}$$

(行列を使って書き直しただけです)

前のスライドの結果を使えば、線形層をモジュール化できる



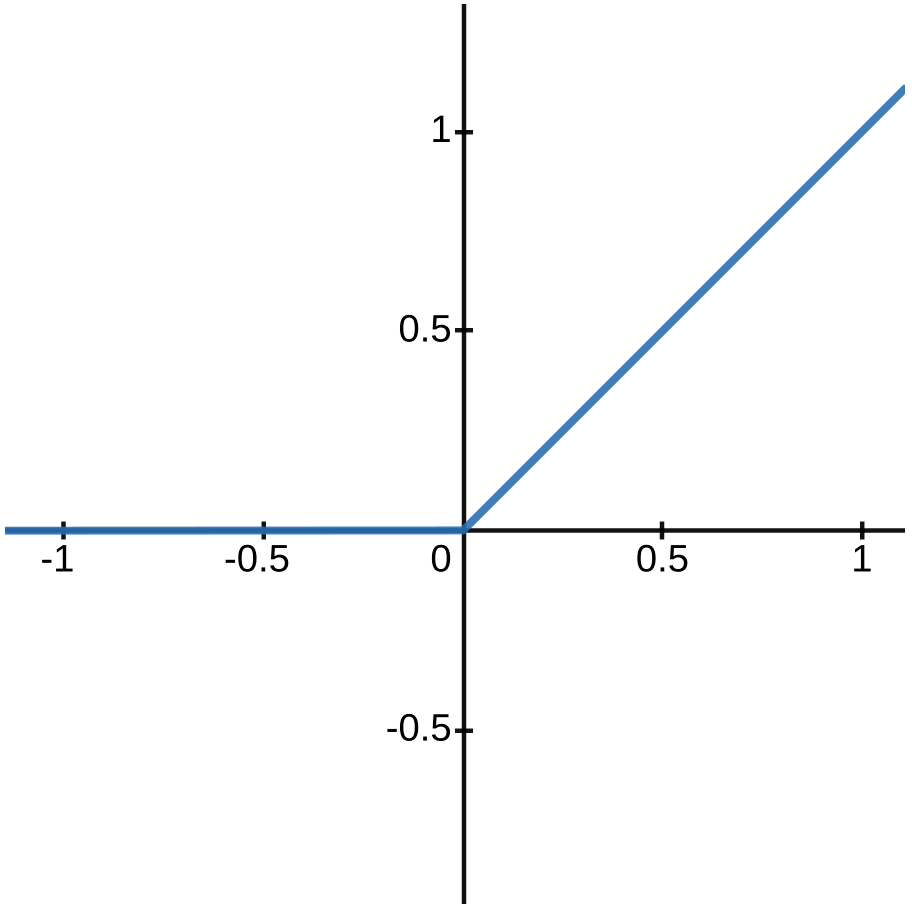
活性化関数の微分

ReLU関数

$$\mathrm{ReLU}(z) = \max(0, z)$$

$$\frac{d}{dz}\mathrm{ReLU}(z) = \begin{cases} 1 & (z_i > 0) \\ 0 & (z_i \leq 0) \end{cases}$$

入力が正の時はそのまま勾配を流し、負の時は勾配を0にする



Wikimedia Commons

損失関数の微分

交差エントロピー損失

$$L = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

シグモイド出力と組み合わせると、**ロジットzに対する微分**が非常にシンプル：

$$\frac{\partial L}{\partial z} = \hat{y} - y$$

誤差逆伝播と呼ばれる所以

6. 実際の使われ方

深層学習フレームワーク

現代の深層学習フレームワークはすべて自動微分を提供：

PyTorch：

```
import torch
x = torch.tensor([2.0], requires_grad=True)
y = x**2 + 3
y.backward()
print(x.grad)  # dy/dx = 2x = 4.0
```

ライブラリが基本的な関数についての微分を覚えている

-> それを組み合わせたものの微分も自動的に計算可能

自動微分の利点

1. **正確性**：連鎖率を使った数学的に正確な微分を求める
2. **効率性**：順伝播とほぼ同じ計算量で逆伝播が可能
3. **柔軟性**：任意*の計算グラフに対応

すべてのノードの勾配が定義されていれば

計算効率の比較

数値微分：n回の順伝播が必要（nはパラメータ数）
解析的微分：微分を数式として求め値を代入、複雑なモデルでは非現実的
誤差逆伝播：1回の順伝播 + 1回の逆伝播

パラメータが1億個のモデルでも、2回の伝播で全勾配を取得！

7. まとめ

本日のまとめ

1. **計算グラフ**：複雑な計算をモジュール化
2. **連鎖率**：合成関数の微分法の多変数版、直列は積、分岐は和のルール
3. **誤差逆伝播法**：順伝播で値を計算し、逆伝播で勾配を効率的に計算
4. **NNの構成要素の微分**：線形層、活性化関数、損失関数の微分がどうなるか
5. **自動微分**：ライブラリが自動的に微分を計算

誤差逆伝播法の意義

誤差逆伝播法により、以下が可能になりました：

- **大規模ニューラルネットワーク**の効率的な学習
- **深層学習**の実用化

現代のAIブームの技術的基盤の一つ！

次回予告

第五回：実装と実験

- ニューラルネットワークをPyTorchを用いて実装
- MNISTデータセットでの手書き数字認識

今日学んだ理論を実際のコードで体験しましょう！