機械学習の基礎

2025/05/01

交通・都市・国土学研究室　古橋郁一・西尾舞紘

本日の発表内容

1. 機械学習分野の概観: 1分
2. Neural Networksの基礎: 25分
3. 様々な機械学習フレームワーク: 25分
4. Transformer: 20分

1. 機械学習分野の概観

機械学習分野の概観

AIが世間を席巻→交通研も！？

今回の理論談話会

M1　西尾（他分野）
Generative Topology Optimization: Exploring Diverse Solutions in Structural Design

M1　西尾（生成モデルの基礎）
Guided conditional diffusion for controllable traffic simulation

M1　門坂（選択モデル）
Deep neural networks for choice analysis: Enhancing behavioral regularity with gradient regularization

M1　堀（生成モデルの基礎）
Face2Diffusion for Fast and Editable Face Personalization

M1　堀（交通流・配分モデル）
End-to-end learning of user equilibrium with implicit neural networks

B4　水谷（生成モデルの基礎）
RoutesFormer: A sequence-based route choice Transformer for efficient path inference from sparse trajectories

B4　一山（最適化・制御の応用）
Learning to delegate for large-scale vehicle routing

機械学習分野の概観

倉澤修論：Vision+Route+Transformer

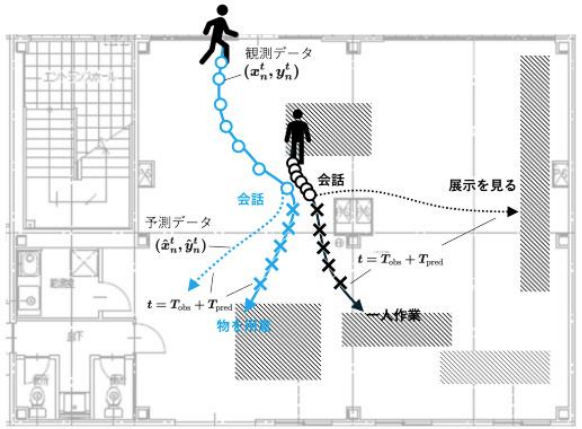


図 3.1　観測データから軌跡と活動を予測するフロー

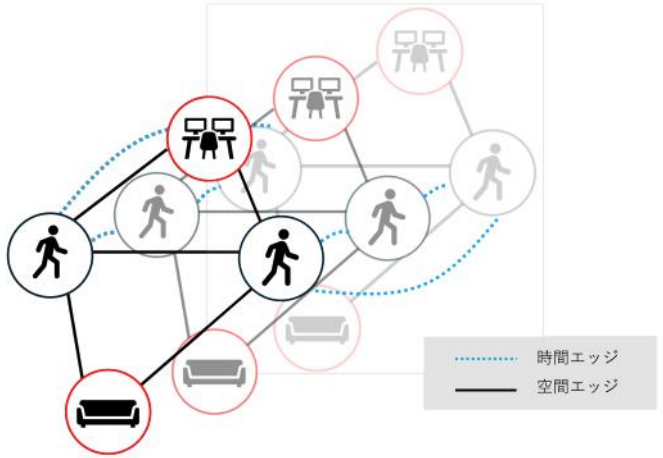
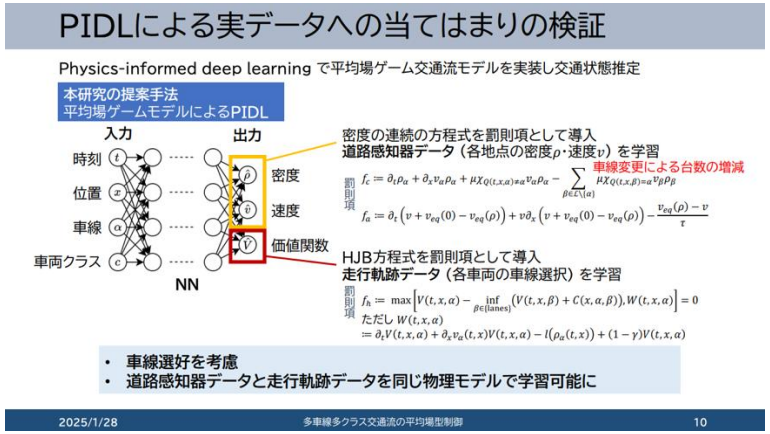
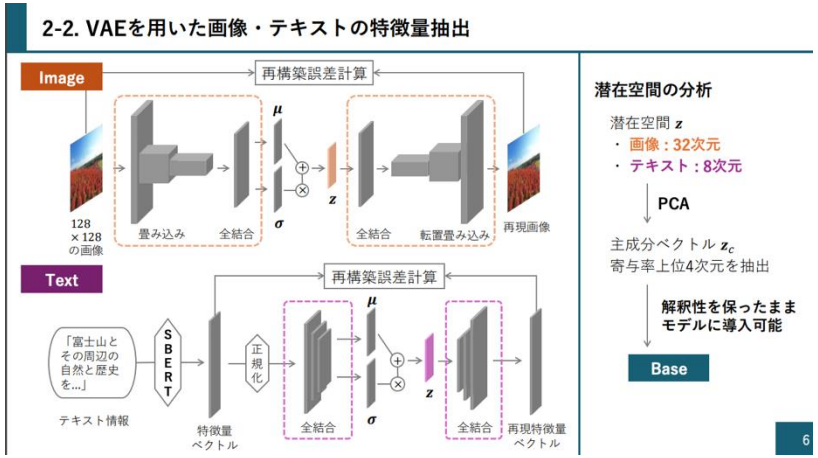


図 3.4　SGAT-TCN モジュールにおける空間および時間的エッジの関係

西尾卒論：GNN

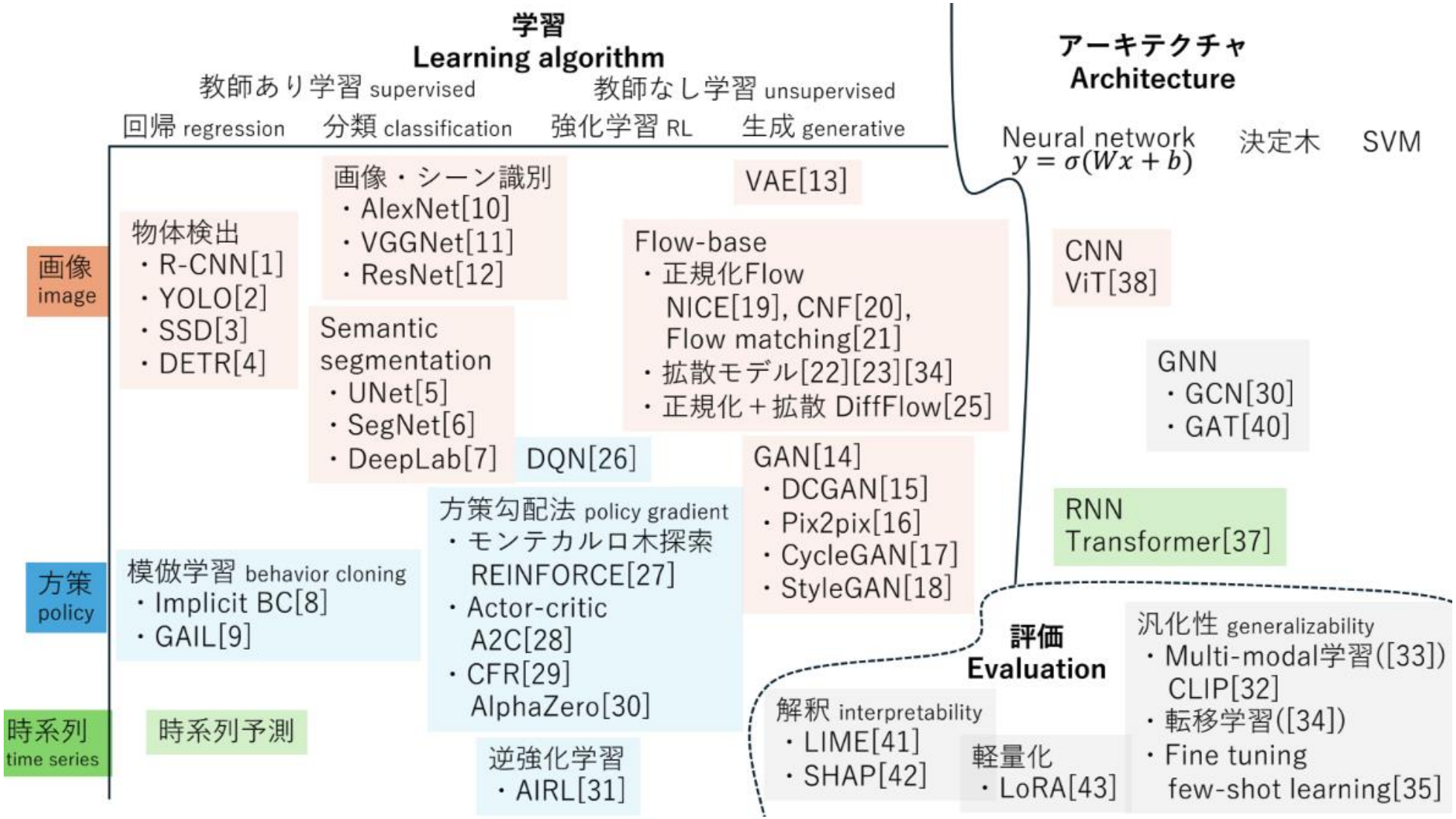


林修論：PINN



門阪卒論：行動モデル+VAE

機械学習を学ぼう！



小川さん, 理論談話会#1, 2024

2. Neural Networks

機械学習 ≒（めっちゃすごい）線形回帰
である

（復習）線形回帰/重回帰

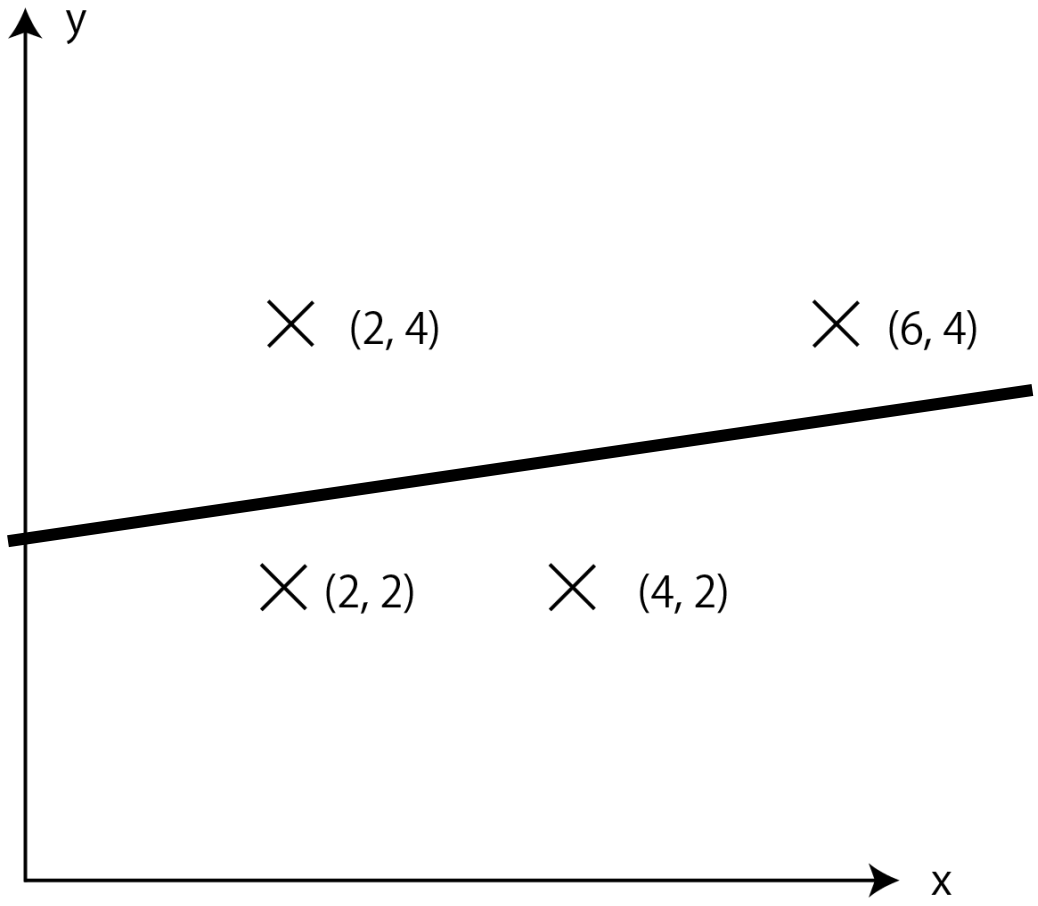
線形回帰

一つの説明変数xを使って、被説明変数yを予測する

$$\hat{y} = w_1 x + b$$

x: 説明変数
y: 被説明変数
w_1: 重み
b: バイアス

$$\text{minimize } MSE = \frac{1}{N}\sum_{i}^{N}(\hat{y}_i - \overline{y_i})$$



（復習）線形回帰/重回帰

線形回帰

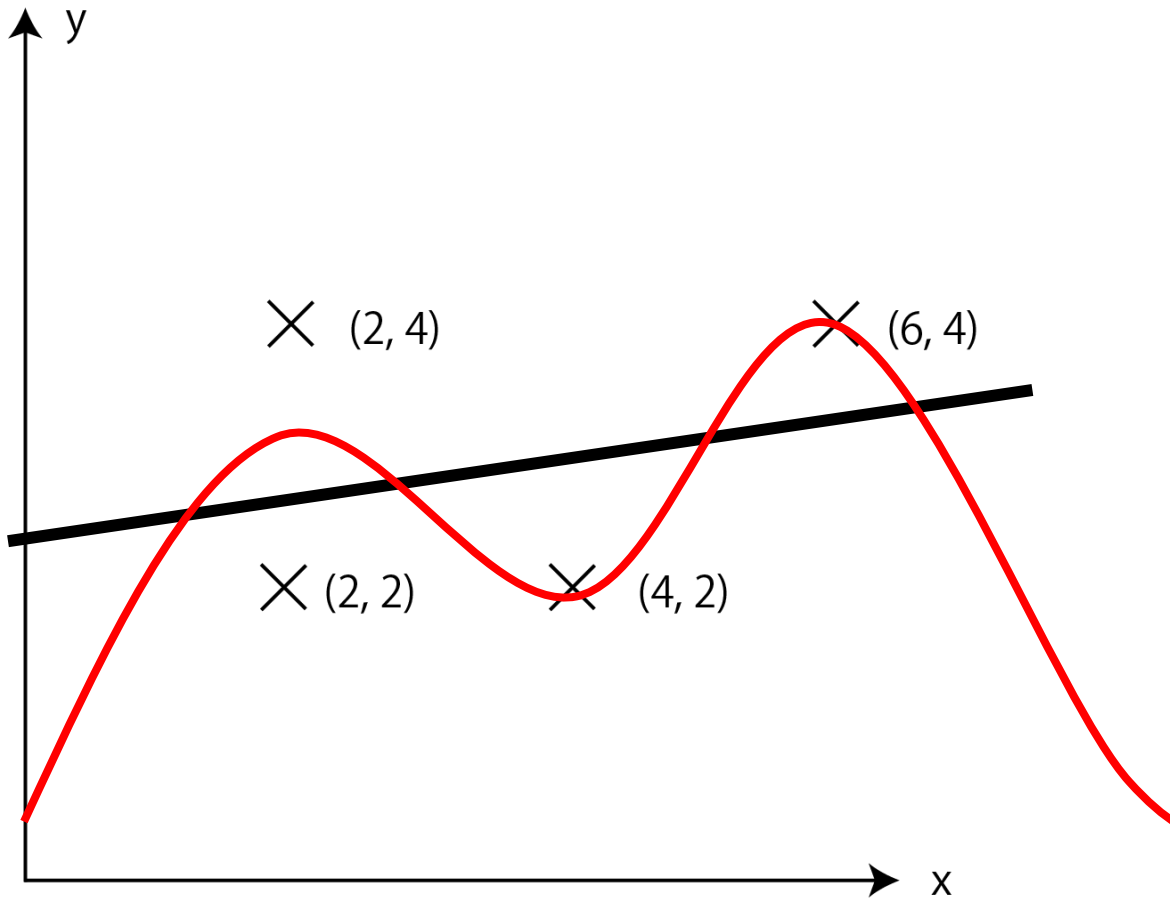
一つの説明変数xを使って、被説明変数yを予測する

$$\hat{y} = w_1 x + b$$

x: 説明変数
y: 被説明変数
w_1: 重み
b: バイアス

$$\text{minimize } MSE = \frac{1}{N}\sum_{i}^{N}(\hat{y}_i - \overline{y_i})$$

より複雑な関数で近似したい！？

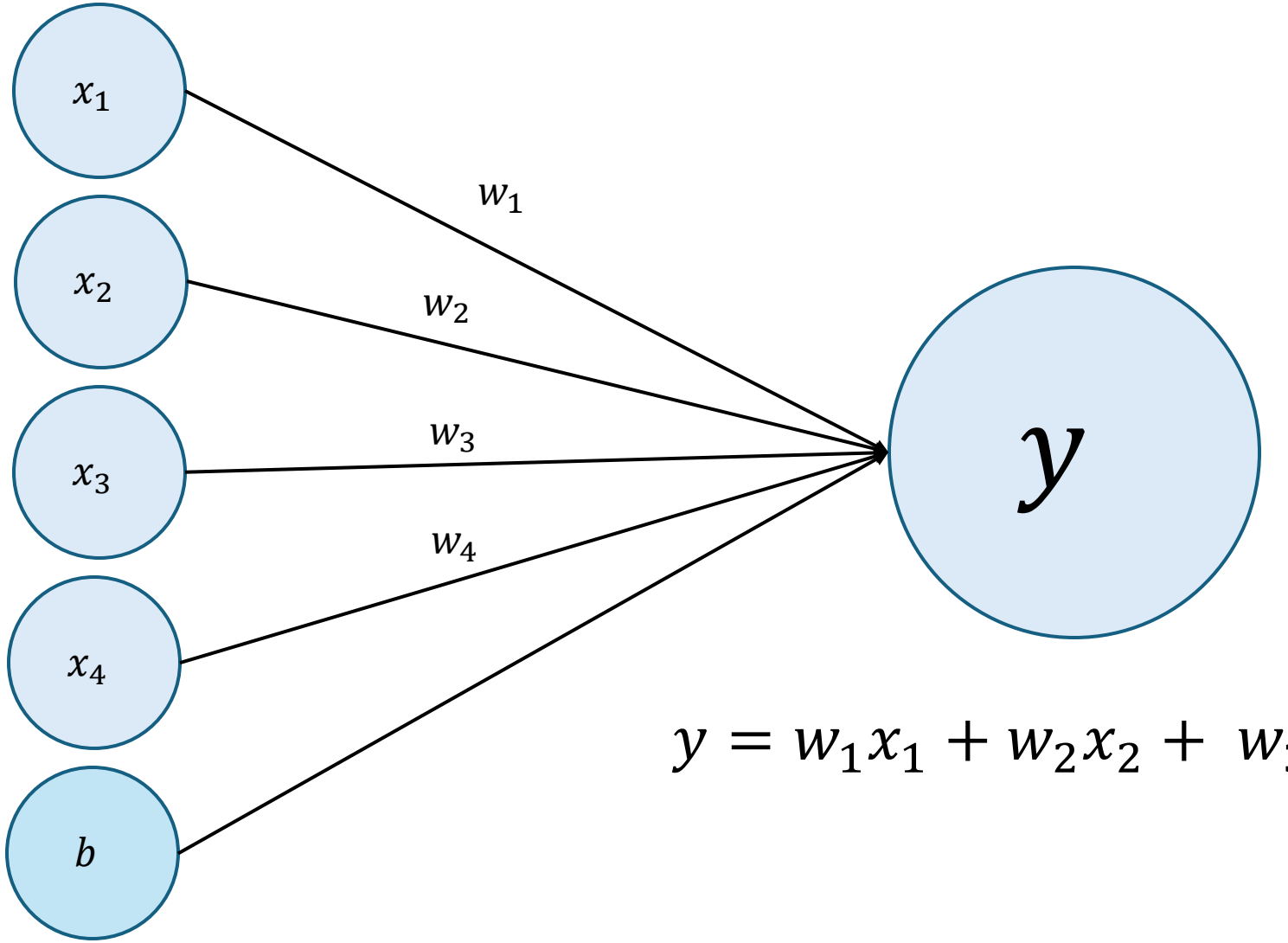


Neural Networksの全体像

- 基本構造
  - Unit
  - Weight
  - Bias
  - Activation Function
- 学習法
  - データの取り扱い
  - 誤差逆伝播法

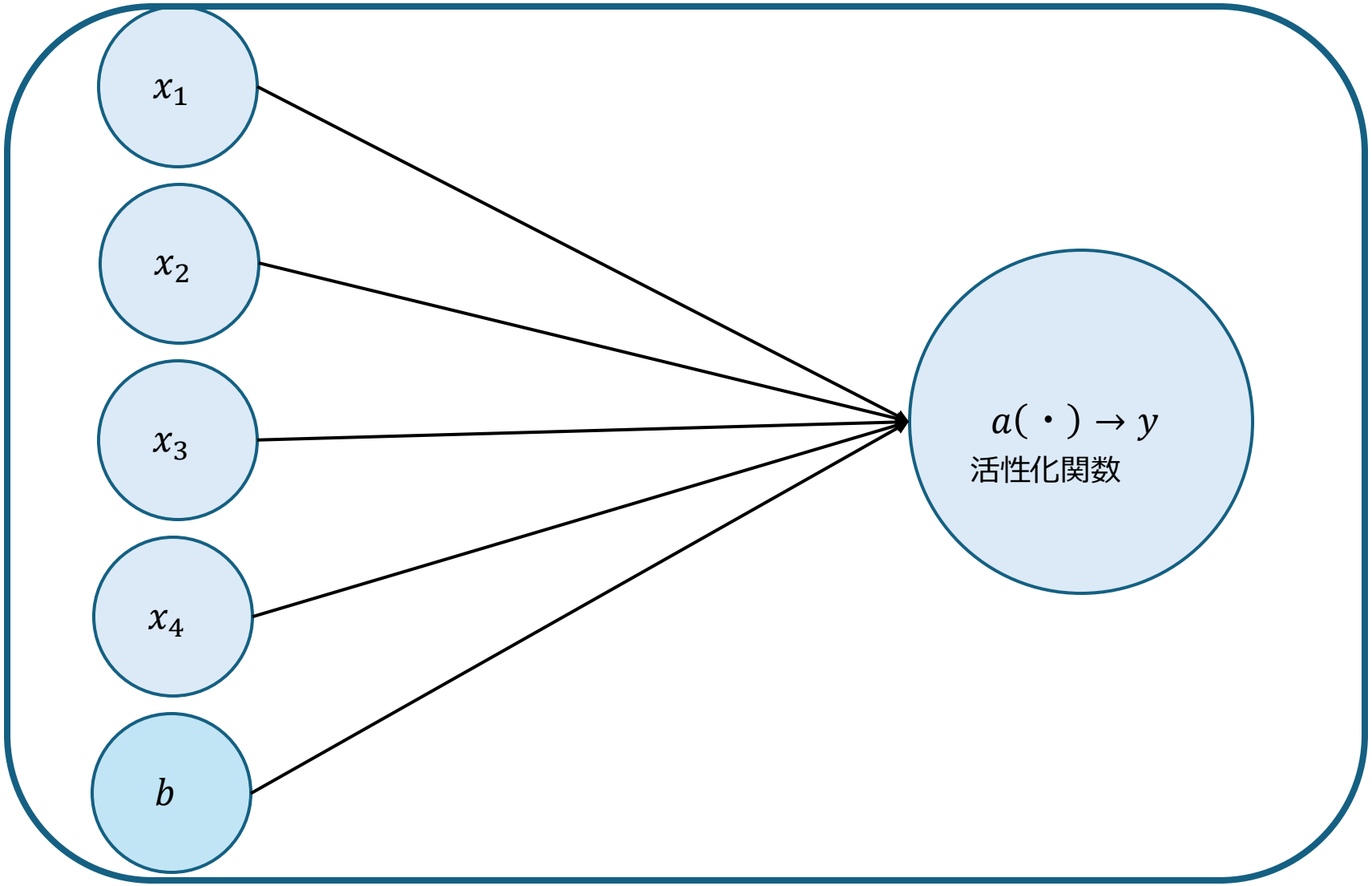
2.1 Neural Netの基本構造

Neural Networksの基本構造 – パーセプトロン



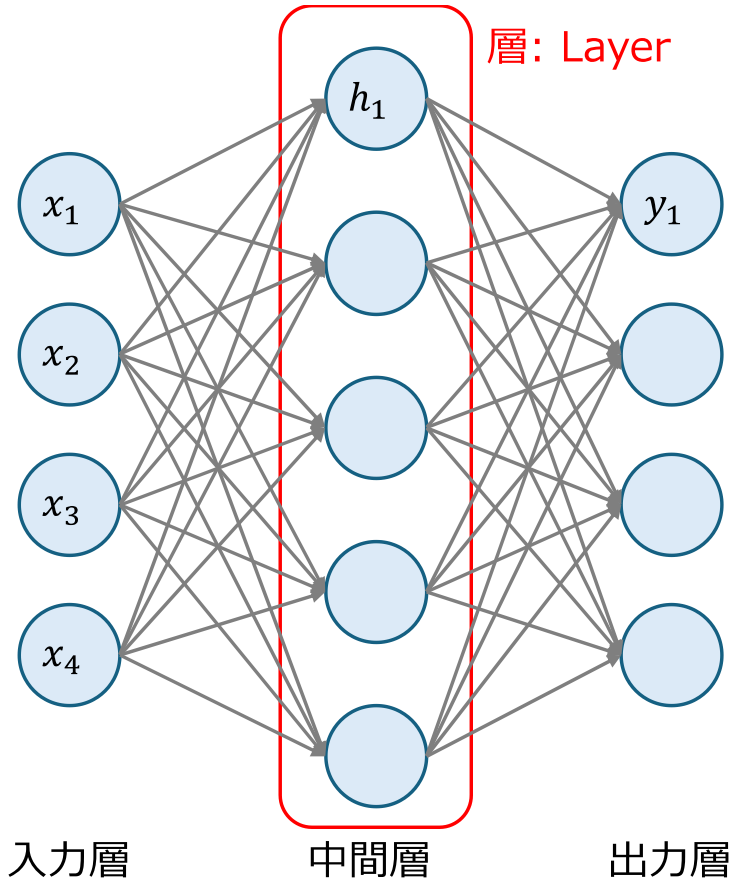
$$y = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$$

Neural Networksの基本構造 – パーセプトロン



この単位を
パーセプトロン

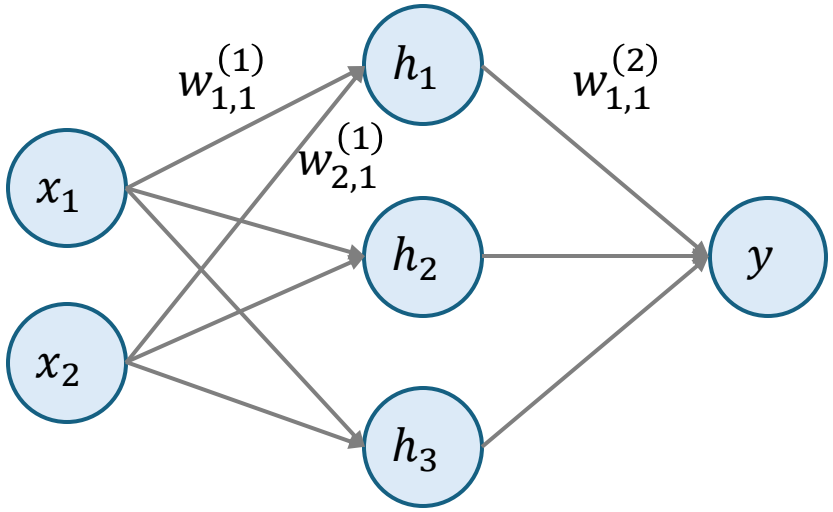
多層パーセプトロン



入力層 → 中間層: $h_1 = a\left(\sum_{i=1}^{4} w_i^{(1)} x_i\right)$

中間層 → 出力層: $y_1 = a\left(\sum_{i=1}^{5} w_i^{(2)} h_i\right)$

多層パーセプトロン ~計算してみよう~



$x_1 = 1, x_2 = 2$　　活性化関数: 層の平均 (ex. $h_1 = \frac{h_1'}{\sum_i h_i'}$)

第一層	h_1	h_2	h_3
x_1	$w_{1,1} = 1$	$w_{1,2} = 2$	$w_{1,3} = 3$
x_2	$w_{2,1} = 4$	$w_{2,2} = 5$	$w_{2,3} = 6$

第二層	y_1	y_2
h_1	$w_{1,1} = 1$	$w_{1,2} = 2$
h_2	$w_{2,1} = 3$	$w_{2,2} = 4$
h_3	$w_{3,1} = 5$	$w_{3,2} = 6$

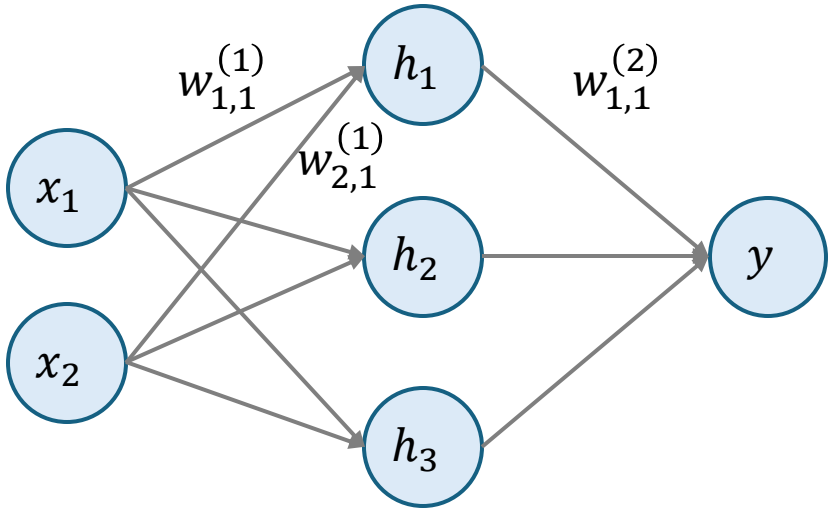
中間層

h_1~h_3を求める

活性化関数:

出力層

多層パーセプトロン ~計算してみよう~



$x_1 = 1, x_2 = 2$　　活性化関数: 層の平均 (ex. $h_1 = \frac{h_1'}{\sum_i h_i'}$)

第一層	h_1	h_2	h_3
x_1	$w_{1,1} = 1$	$w_{1,2} = 2$	$w_{1,3} = 3$
x_2	$w_{2,1} = 4$	$w_{2,2} = 5$	$w_{2,3} = 6$

第二層	y_1	y_2
h_1	$w_{1,1} = 1$	$w_{1,2} = 2$
h_2	$w_{2,1} = 3$	$w_{2,2} = 4$
h_3	$w_{3,1} = 5$	$w_{3,2} = 6$

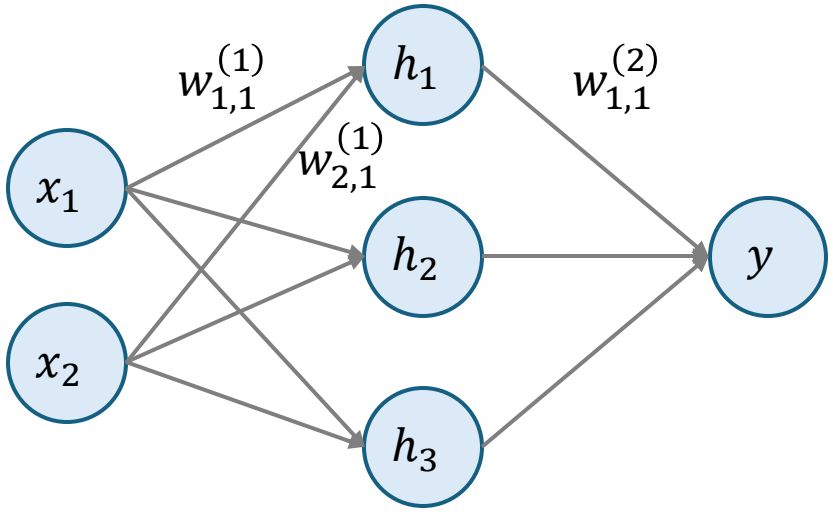
中間層

$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:
平均化する

出力層

多層パーセプトロン ~計算してみよう~



$x_1 = 1, x_2 = 2$　　活性化関数: 層の平均 (ex. $h_1 = \frac{h_1'}{\sum_i h_i'}$)

第一層	h_1	h_2	h_3
x_1	$w_{1,1} = 1$	$w_{1,2} = 2$	$w_{1,3} = 3$
x_2	$w_{2,1} = 4$	$w_{2,2} = 5$	$w_{2,3} = 6$

第二層	y_1	y_2
h_1	$w_{1,1} = 1$	$w_{1,2} = 2$
h_2	$w_{2,1} = 3$	$w_{2,2} = 4$
h_3	$w_{3,1} = 5$	$w_{3,2} = 6$

中間層

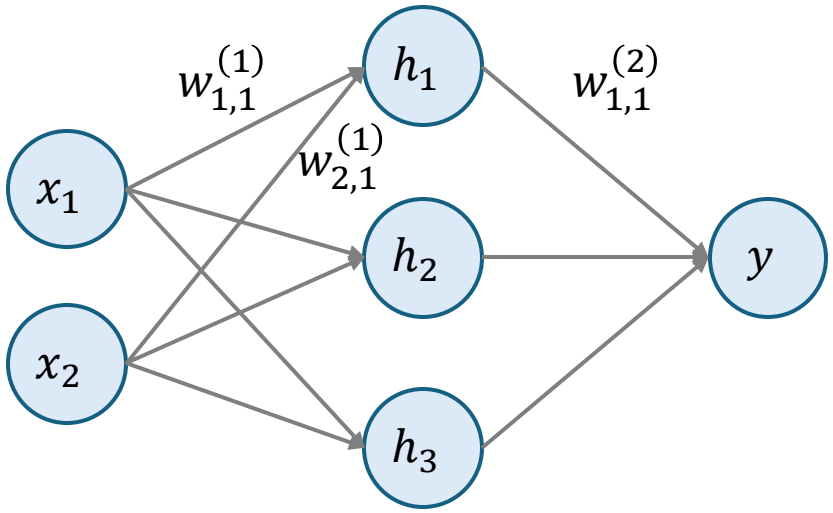
$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:

$h_1 = \frac{h_1'}{\sum_i h_i'} = \frac{9}{36}$
$h_2 = \frac{12}{36}$
$h_3 = \frac{15}{36}$

出力層

多層パーセプトロン ~計算してみよう~



$x_1 = 1, x_2 = 2$　　活性化関数: 層の平均 (ex. $h_1 = \frac{h_1'}{\sum_i h_i'}$)

第一層	h_1	h_2	h_3
x_1	$w_{1,1} = 1$	$w_{1,2} = 2$	$w_{1,3} = 3$
x_2	$w_{2,1} = 4$	$w_{2,2} = 5$	$w_{2,3} = 6$

第二層	y_1	y_2
h_1	$w_{1,1} = 1$	$w_{1,2} = 2$
h_2	$w_{2,1} = 3$	$w_{2,2} = 4$
h_3	$w_{3,1} = 5$	$w_{3,2} = 6$

中間層

$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:

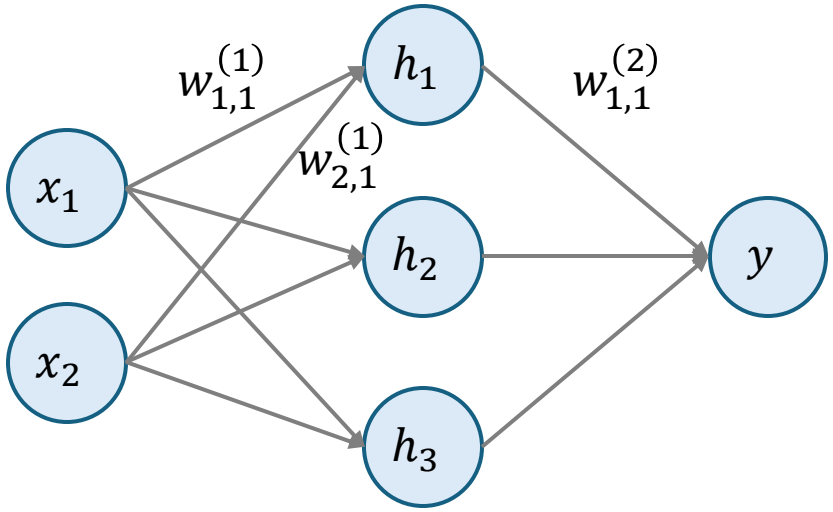
$h_1 = \frac{h_1'}{\sum_i h_i'} = \frac{9}{36}$
$h_2 = \frac{12}{36}$
$h_3 = \frac{15}{36}$

出力層

$y = \frac{120}{276} = \frac{10}{23}$

順伝播！

多層パーセプトロン ~計算してみよう~



$x_1 = 1, x_2 = 2$　　活性化関数: 層の平均 (ex. $h_1 = \frac{h_1'}{\sum_i h_i'}$)

第一層	h_1	h_2	h_3
x_1	$w_{1,1} = 1$	$w_{1,2} = 2$	$w_{1,3} = 3$
x_2	$w_{2,1} = 4$	$w_{2,2} = 5$	$w_{2,3} = 6$

第二層	y_1	y_2
h_1	$w_{1,1} = 1$	$w_{1,2} = 2$
h_2	$w_{2,1} = 3$	$w_{2,2} = 4$
h_3	$w_{3,1} = 5$	$w_{3,2} = 6$

中間層

$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:

$h_1 = \frac{h_1'}{\sum_i h_i'} = \frac{9}{36}$
$h_2 = \frac{12}{36}$
$h_3 = \frac{15}{36}$

出力層

$y = \frac{120}{276} = \frac{10}{23}$

※行列の掛け算！→GPUが重要

$$\boldsymbol{Y} = a(\boldsymbol{WX})$$

2.2 Neural Netの学習法

2.2.1 モデルの学習

線形回帰の場合→最小二乗法

$$y = ax + b$$

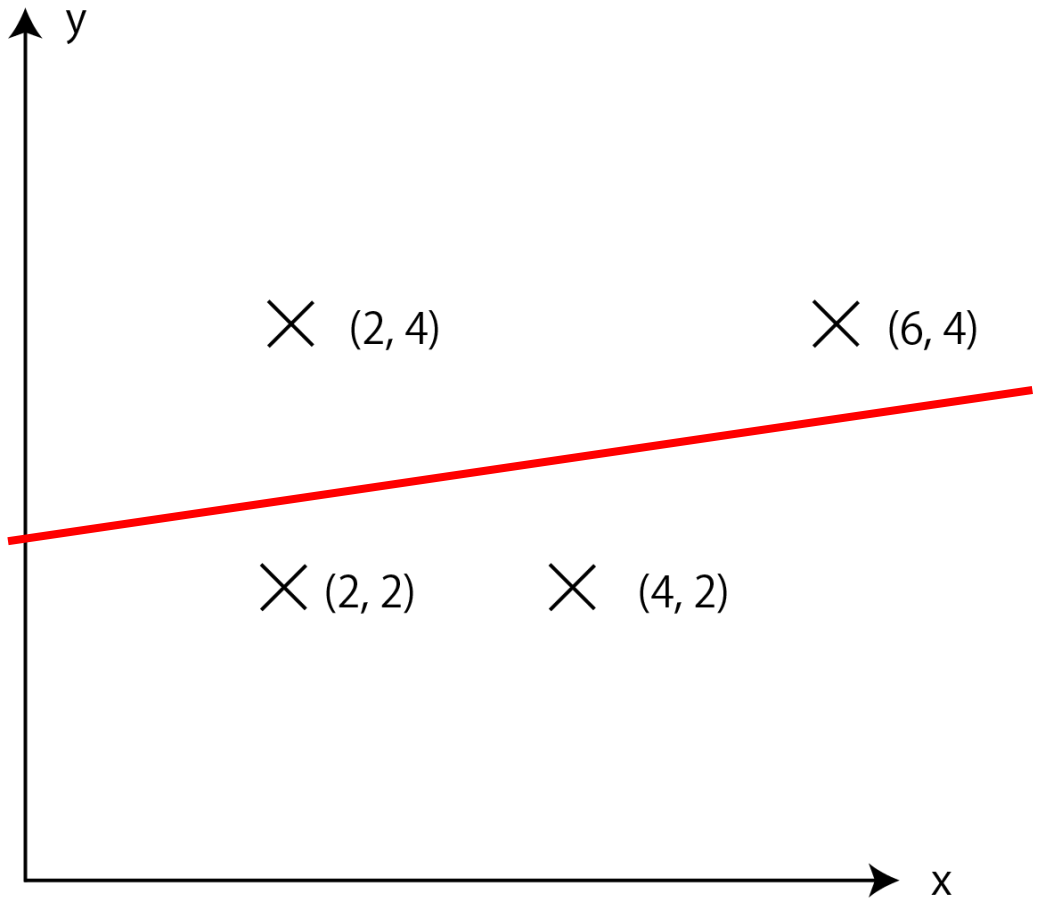
二乗誤差Lを最小化

$$L(a,b) = \sum_i^4 (y_i - y)^2 = \sum_i^4 (y_i - ax_i - b)^2$$

$$\begin{cases} \dfrac{\partial L}{\partial a} = -2\sum_i^4 x_i(y_i - ax_i - b) = 0 \\ \dfrac{\partial L}{\partial b} = -2\sum_i^4 (y_i - ax_i - b) = 0 \end{cases}$$

$$\begin{cases} a = \dfrac{\sum_i^4 (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^4 (x_i - \bar{x})^2} \\ b = \bar{y} - a\bar{x} \end{cases}$$

$$y = \frac{2}{11}x + \frac{26}{11}, L^* = \frac{440}{121}$$



2.2.1 モデルの学習

NNの場合

中間層

$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:

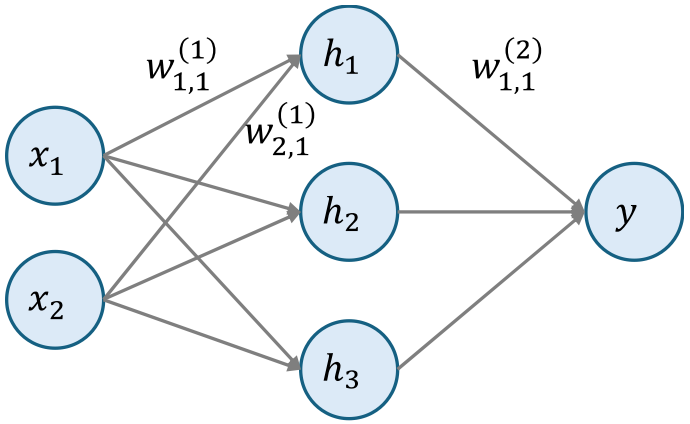
$h_1 = \frac{h_1'}{\sum_i h_i'} = \frac{9}{36}$
$h_2 = \frac{12}{36}$
$h_3 = \frac{15}{36}$

出力層

$y = \frac{120}{276} = \frac{10}{23}$

真値:1

乖離 $= 1 - \frac{10}{23} = \frac{13}{23}$



→どうやって最適化する？

Q. NNの学習の難しさは？

2.2.1 モデルの学習

NNの場合

中間層

$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:

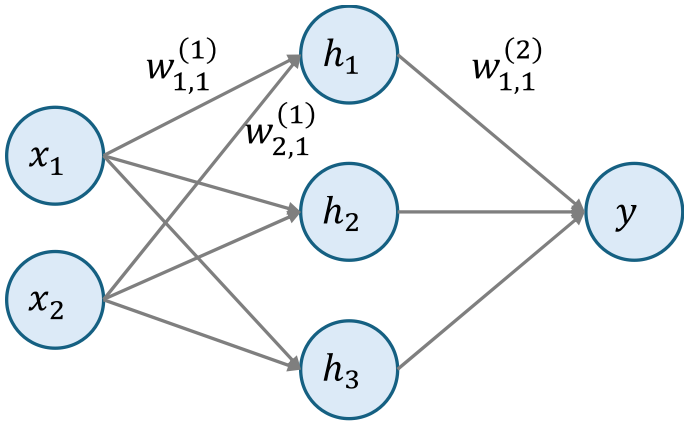
$h_1 = \frac{h_1'}{\sum_i h_i'} = \frac{9}{36}$
$h_2 = \frac{12}{36}$
$h_3 = \frac{15}{36}$

出力層

$y = \frac{120}{276} = \frac{10}{23}$

真値:1

乖離 $= 1 - \frac{10}{23} = \frac{13}{23}$



→どうやって最適化する？

Q. NNの学習の難しさは？

パラメータ数が多すぎる！

線形回帰

↓ 2

2.2.1 モデルの学習

NNの場合

中間層

$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:

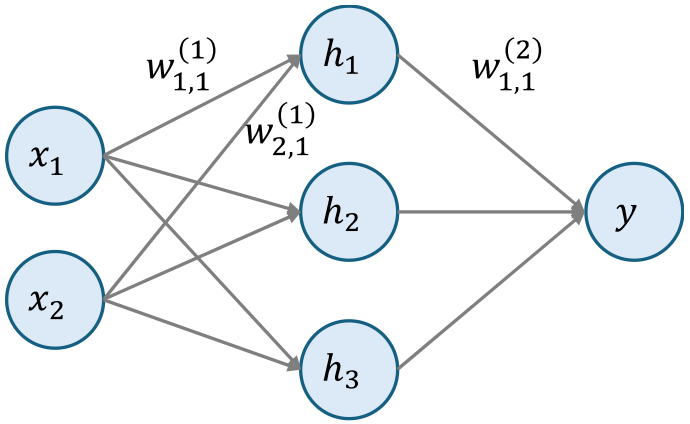
$h_1 = \frac{h_1'}{\sum_i h_i'} = \frac{9}{36}$
$h_2 = \frac{12}{36}$
$h_3 = \frac{15}{36}$

出力層

$y = \frac{120}{276} = \frac{10}{23}$

真値:1

乖離 $= 1 - \frac{10}{23} = \frac{13}{23}$



→どうやって最適化する？

Q. NNの学習の難しさは？

パラメータ数が多すぎる！

線形回帰　例題

2　9

2.2.1 モデルの学習

NNの場合

中間層

$h_1' = w_{1,1}x_1 + w_{2,1}x_2 = 1 \times 1 + 4 \times 2 = 9$
$h_2' = 12$
$h_3' = 15$

活性化関数:

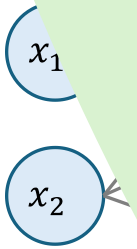
$h_1 = \frac{h_1'}{\sum_i h_i'} = \frac{9}{36}$
$h_2 = \frac{12}{36}$
$h_3 = \frac{15}{36}$

出力層

$y = \frac{120}{276} = \frac{10}{23}$

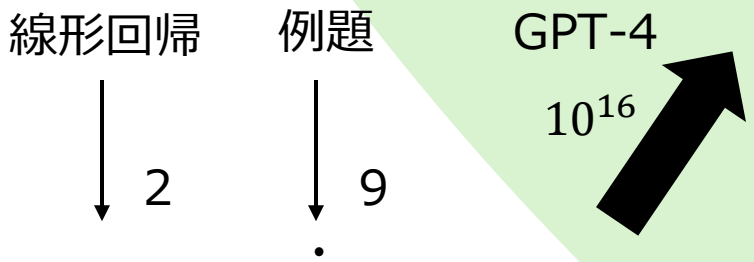
真値:1

乖離 $= 1 - \frac{10}{23} = \frac{13}{23}$



Q. NNの学習の難しさは？

パラメータ数が多すぎる！



Q. Neural Networksの学習における難しさは？

学習 = 重みパラメータを求める

線形回帰ならすぐ解が求まるが…

Q. Neural Networksの学習における難しさは？

学習 = 重みパラメータを求める

線形回帰ならすぐ解が求まるが…

最小二乗法を思い返す

二乗誤差Lを最小化

$$L(a,b) = \sum_i^4 (y_i - y)^2 = \sum_i^4 (y_i - ax_i - b)^2$$

$$\begin{cases} \frac{\partial L}{\partial a} = -2\sum_i^4 x_i(y_i - ax_i - b) = 0 \\ \frac{\partial L}{\partial b} = -2\sum_i^4 (y_i - ax_i - b) = 0 \end{cases}$$

←じっくり眺める

$$\begin{cases} a = \frac{\sum_i^4 (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^4 (x_i - \bar{x})^2} \\ b = \bar{y} - a\bar{x} \end{cases}$$

$$y = \frac{2}{11}x + \frac{26}{11}, L^* = \frac{440}{121}$$

Q. Neural Networksの学習における難しさは？

学習 = 重みパラメータを求める

線形回帰ならすぐ解が求まるが…

最小二乗法を思い返す

二乗誤差Lを最小化

$$L(a,b) = \sum_i^4 (y_i - y)^2 = \sum_i^4 (y_i - ax_i - b)^2$$

$$\begin{cases} \frac{\partial L}{\partial a} = -2\sum_i^4 x_i(y_i - ax_i - b) = 0 \\ \frac{\partial L}{\partial b} = -2\sum_i^4 (y_i - ax_i - b) = 0 \end{cases}$$

$$\begin{cases} a = \frac{\sum_i^4 (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^4 (x_i - \bar{x})^2} \\ b = \bar{y} - a\bar{x} \end{cases}$$

$$y = \frac{2}{11}x + \frac{26}{11}, L^* = \frac{440}{121}$$

- 💡データとの乖離(損失関数)を定義しよう！
- 💡損失関数が最小化される方向に重みを調整しよう！

↑聞いたことがある話

Q. Neural Networksの学習における難しさは？

学習 = 重みパラメータを求める

線形回帰ならすぐ解が求まるが…

最小二乗法を思い返す

二乗誤差Lを最小化

$$L(a,b) = \sum_i^4 (y_i - y)^2 = \sum_i^4 (y_i - ax_i - b)^2$$

$$\begin{cases} \dfrac{\partial L}{\partial a} = -2\sum_i^4 x_i(y_i - ax_i - b) = 0 \\ \dfrac{\partial L}{\partial b} = -2\sum_i^4 (y_i - ax_i - b) = 0 \end{cases}$$

$$\begin{cases} a = \dfrac{\sum_i^4 (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^4 (x_i - \bar{x})^2} \\ b = \bar{y} - a\bar{x} \end{cases}$$

$$y = \frac{2}{11}x + \frac{26}{11}, L^* = \frac{440}{121}$$

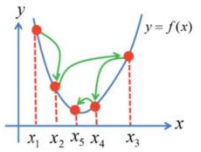
- データとの乖離(損失関数)を定義しよう！
- 損失関数が最小化される方向に重みを調整しよう！

制約なし非線形最適化問題の数値解法

増田さん, スタートアップ最適化回

ステップA：降下方向の探索
どの方向に向かえば目的関数が減少するか

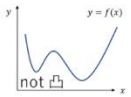
ステップB：ステップサイズの探索
降下方向にどこまで進めるか

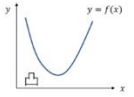


ステップAとステップBを繰り返すことで局所最適解を見つける

凸計画問題では，局所最適解が大域的最適解に一致する→解の一意性

目的関数が凸関数で，実行可能領域が凸集合であるような最適化問題





fが凸関数
⇔任意のa, bに対して，
$\lambda f(a) + (1-\lambda) f(b) \geq f(\lambda a + (1-\lambda) b)\ \forall \lambda \in [0,1]$

→関数上の任意の2点をとって線分を引いた時，その線が必ず元の関数より上に来る

最急降下法

- 各反復において、 $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}(\boldsymbol{x}^{(k)})$と更新する。
  ステップ幅　探索方向
- 更新先での目的関数値は、 $f\left(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}(\boldsymbol{x}^{(k)})\right) \approx f(\boldsymbol{x}^{(k)}) + \alpha \nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}(\boldsymbol{x}^{(k)})$
- $\nabla f(\boldsymbol{x}^{(k)})^T \boldsymbol{d}(\boldsymbol{x}^{(k)}) < 0$ならば、十分小さいステップ幅$\alpha$に対して、 $f\left(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{d}(\boldsymbol{x}^{(k)})\right) < f(\boldsymbol{x}^{(k)})$
- 最急降下法では、 $\boldsymbol{d}(\boldsymbol{x}^{(k)}) = -\nabla f(\boldsymbol{x}^{(k)})$ (勾配の逆方向)と定める。
- 非線形最適化では、目的関数の形を工夫して、凸最適化に落とし込んで勾配降下法で解くことを考えてみる！

28

Q. Neural Networksの学習における難しさは？

学習 = 重みパラメータを求める

線形回帰ならすぐ解が求まるが…

最小二乗法を思い返す

二乗誤差Lを最小化

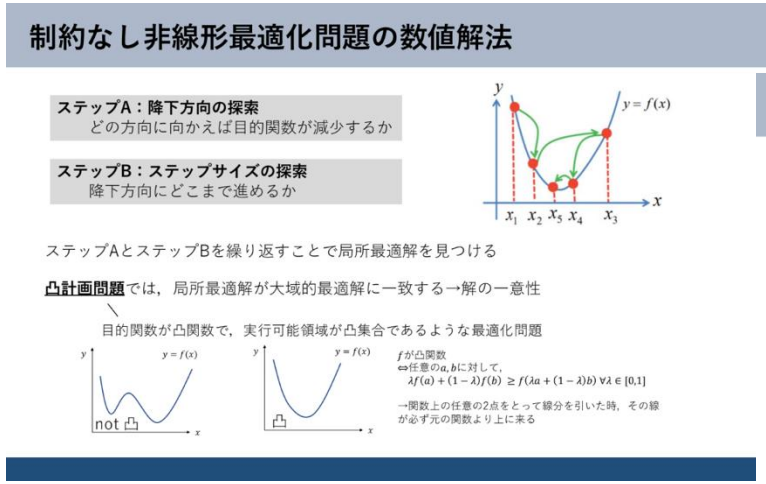
$$L(a,b) = \sum_i^4 (y_i - y)^2 = \sum_i^4 (y_i - ax_i - b)^2$$

$$\begin{cases} \dfrac{\partial L}{\partial a} = -2\sum_i^4 x_i(y_i - ax_i - b) = 0 \\ \dfrac{\partial L}{\partial b} = -2\sum_i^4 (y_i - ax_i - b) = 0 \end{cases}$$

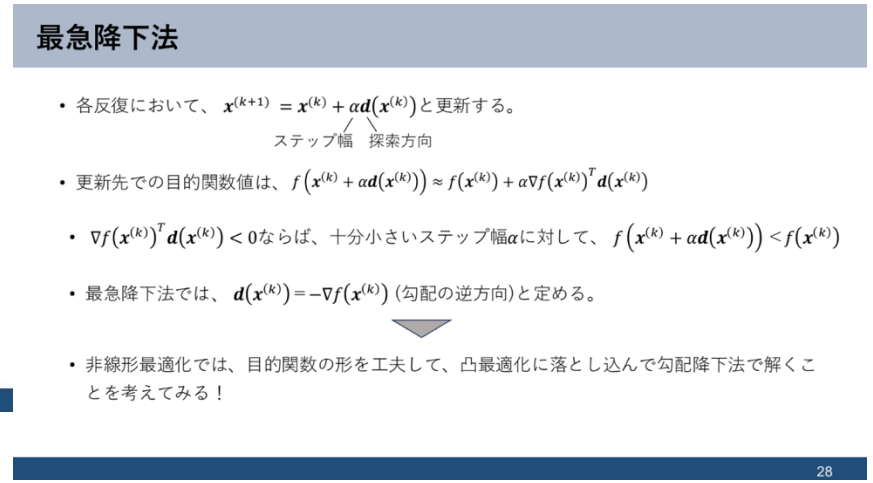
$$\begin{cases} a = \dfrac{\sum_i^4 (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^4 (x_i - \bar{x})^2} \\ b = \bar{y} - a\bar{x} \end{cases}$$

$$y = \frac{2}{11}x + \frac{26}{11}, L^* = \frac{440}{121}$$

💡データとの乖離(損失関数)を定義しよう！
💡損失関数が最小化される方向に重みを調整しよう！



増田さん, スタートアップ最適化回

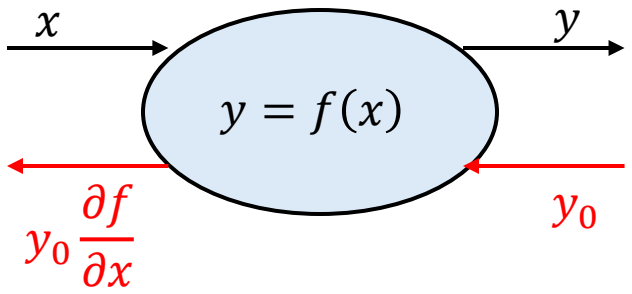


ということは、微分が必要…！？

計算グラフ

https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

計算グラフ: 計算過程をグラフ表現したもの

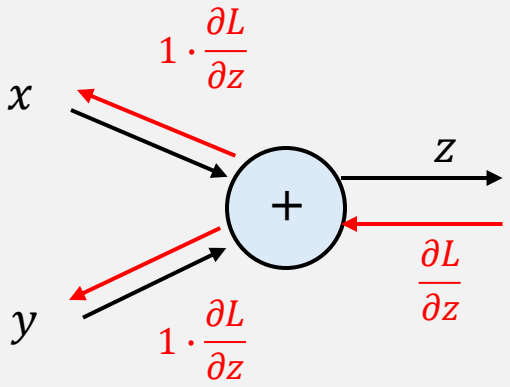


損失関数が減少する方向に重みパラメータを更新したい

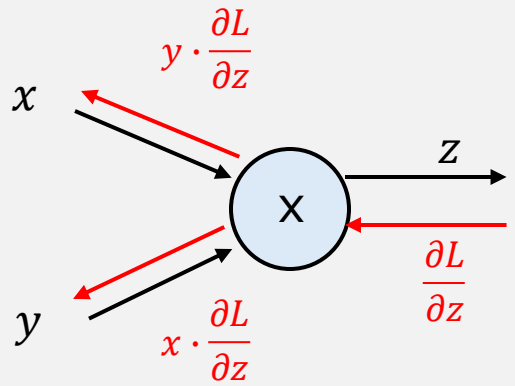
→ $\frac{\partial L}{\partial w}$を考える

計算グラフのルール

①加算



②乗算



※これは微分の連鎖律に基づく

計算グラフ

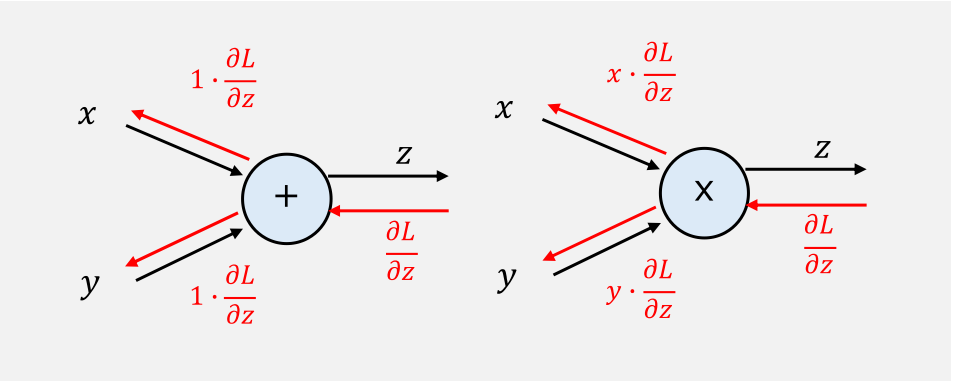
https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

身近な例:

（問題）

りんご100円を2個, みかん150円を3個買う

りんご・みかんの値段が値上がりするとき、合計金額Lへの影響は？



計算グラフ

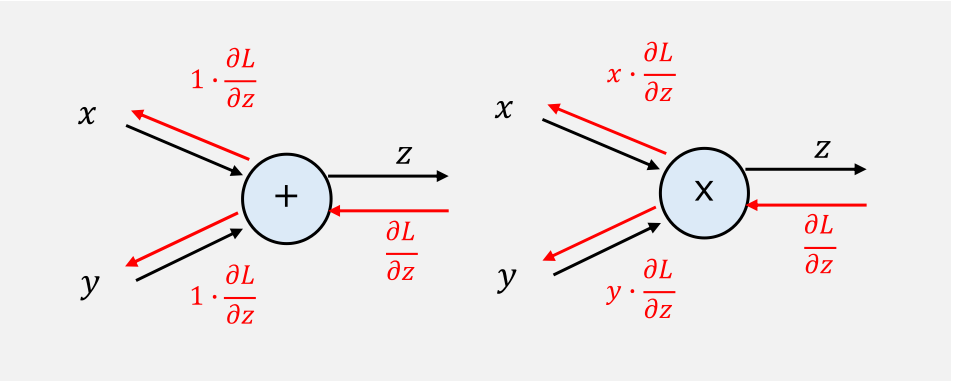
https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

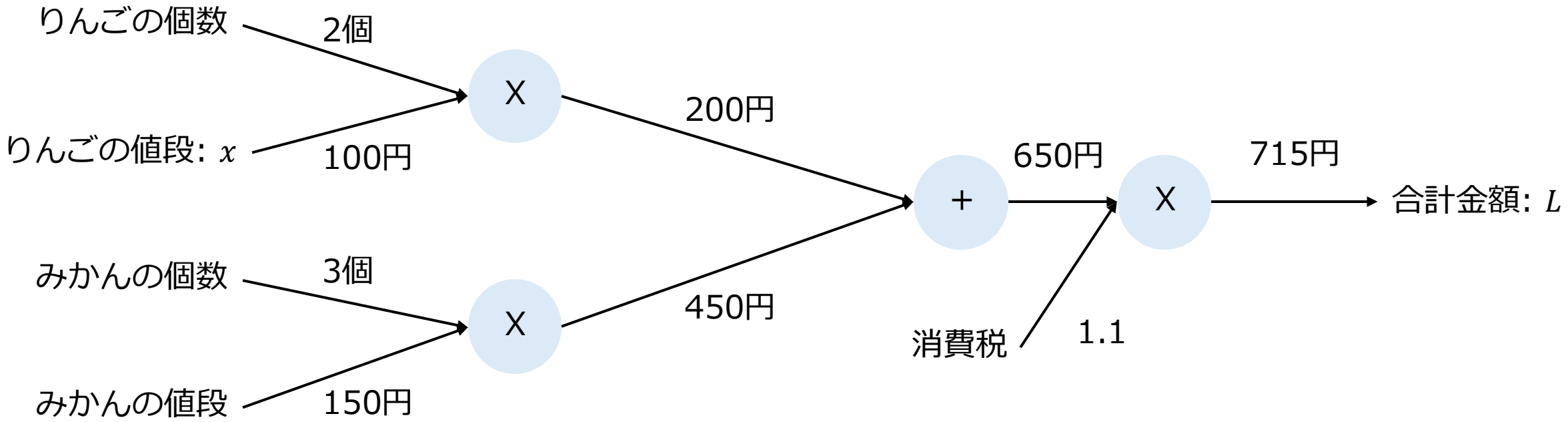
身近な例:

（問題）

りんご100円を2個, みかん150円を3個買う

りんご・みかんの値段が値上がりするとき、合計金額Lへの影響は？





計算グラフ

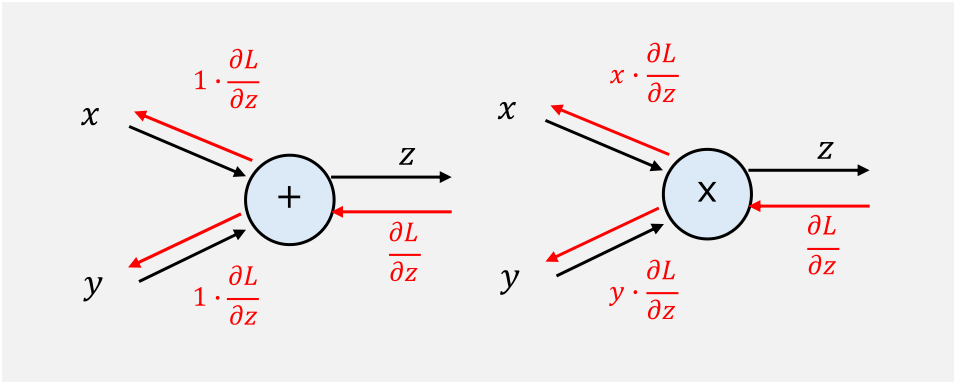
https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

身近な例:

（問題）

りんご100円を2個, みかん150円を3個買う

りんご・みかんの値段が値上がりするとき、合計金額Lへの影響は？



$\frac{\partial L}{\partial x}$ **を求めたい！**

りんごの個数

2個

りんごの値段: x

100円

X

200円

みかんの個数

3個

みかんの値段

150円

X

450円

+

650円

X

715円

合計金額: L

1

消費税

1.1

つまり、**2.2倍の影響**がある

= 100円値上がり→合計金額は220円上がる

計算グラフ

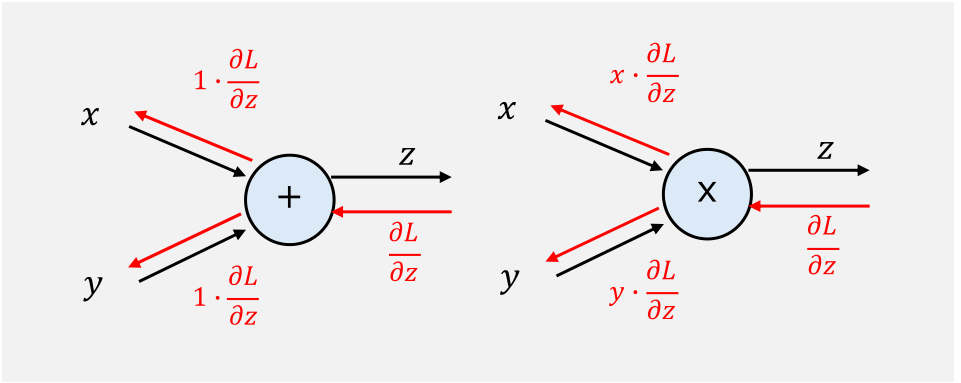
https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

身近な例:

（問題）

りんご100円を2個, みかん150円を3個買う

りんご・みかんの値段が値上がりするとき、合計金額Lへの影響は？



$\frac{\partial L}{\partial x}$ **を求めたい！**

りんごの個数 2個

りんごの値段: x 100円

X → 200円

みかんの個数 3個

みかんの値段 150円

X → 450円

+ 650円 → X 715円 → 合計金額: L

1.1 / 1

消費税 1.1

つまり、**2.2倍の影響**がある

= 100円値上がり→合計金額は220円上がる

計算グラフ

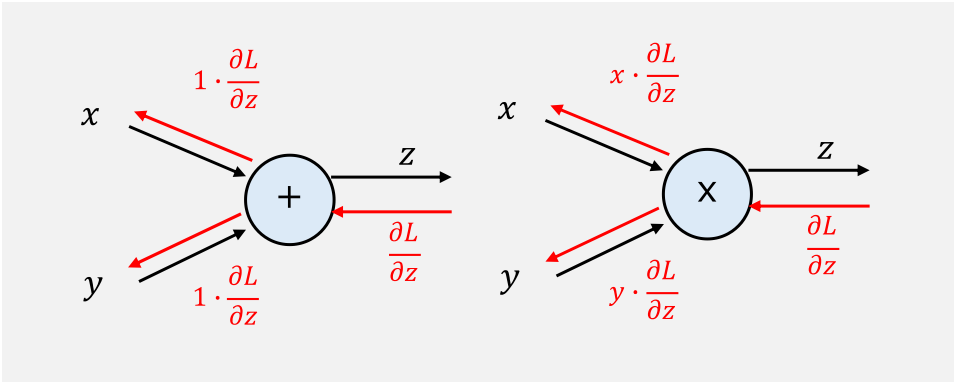
https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

身近な例:

（問題）

りんご100円を2個, みかん150円を3個買う

りんご・みかんの値段が値上がりするとき、合計金額Lへの影響は？



$\frac{\partial L}{\partial x}$ **を求めたい！**

りんごの個数

2個

りんごの値段: x

100円

X

200円

1.1

みかんの個数

3個

みかんの値段

150円

X

450円

+

650円

1.1

消費税

1.1

X

715円

1

合計金額: L

つまり、**2.2倍の影響**がある

= 100円値上がり→合計金額は220円上がる

計算グラフ

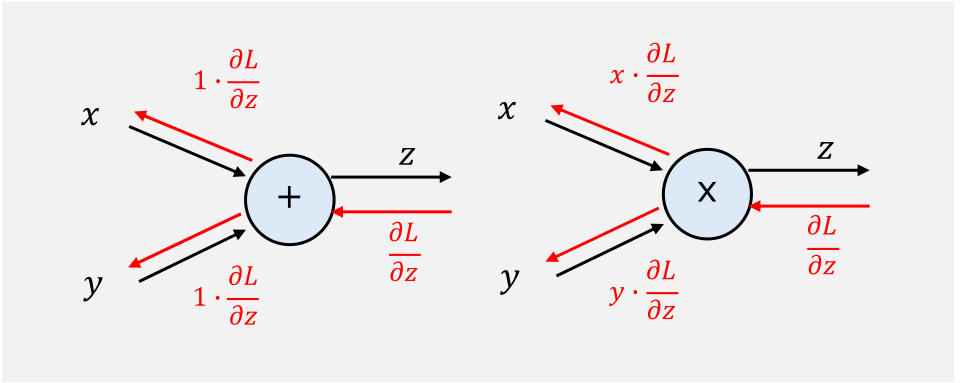
https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

身近な例:

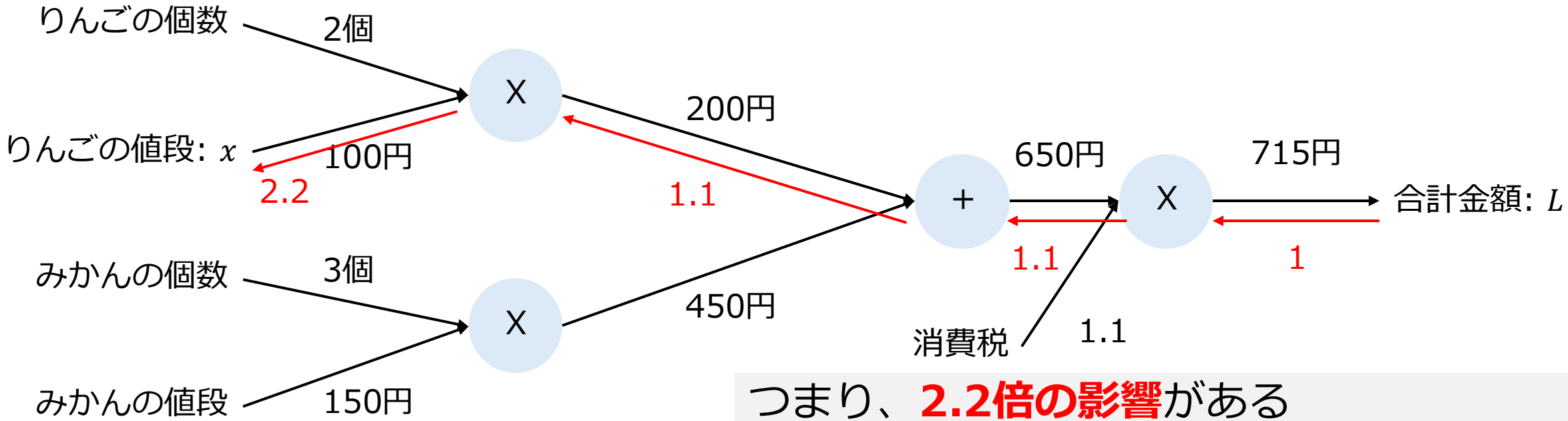
（問題）

りんご100円を2個, みかん150円を3個買う

りんご・みかんの値段が値上がりするとき、合計金額Lへの影響は？



$\frac{\partial L}{\partial x}$ **を求めたい！**



つまり、**2.2倍の影響**がある

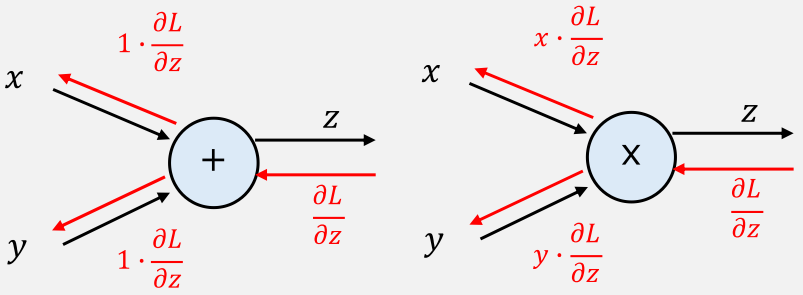
ex. 100円値上がり→合計金額は220円上がる

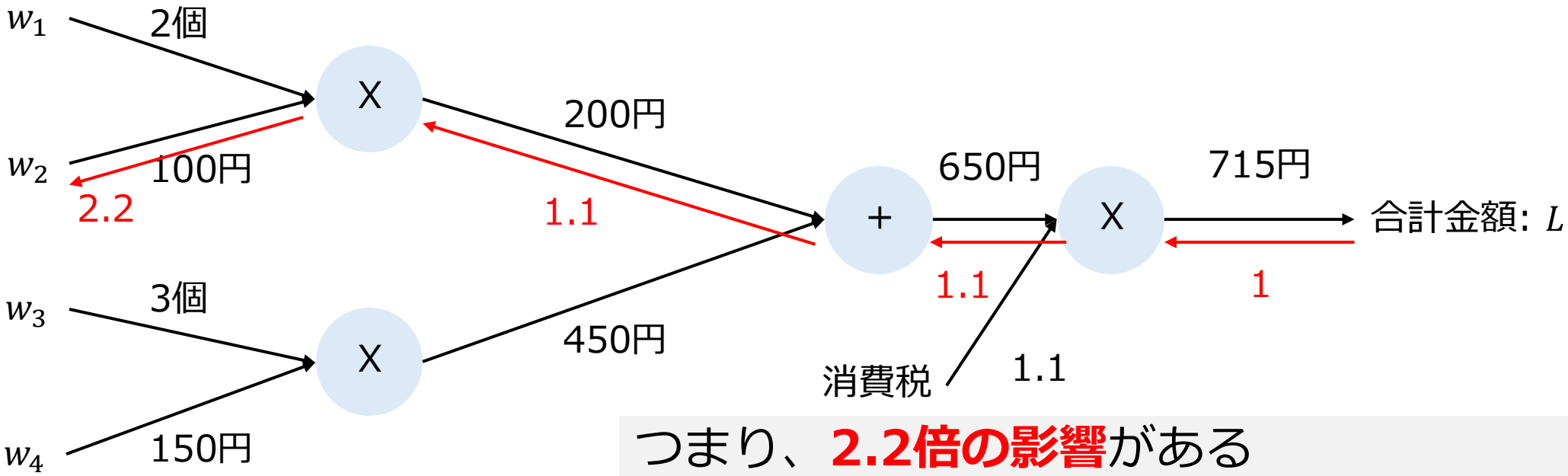
計算グラフ

https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b

計算グラフの凄さ

- 実際にはりんご、みかんを**重み**に置き換える
- 中間値を保存することで、逆から辿れば勾配が計算できる
  →**誤差逆伝播**と呼ぶ
- Pytorch, Tensorflowでは勾配を保存し、**自動微分**





つまり、**2.2倍の影響**がある
ex. 100円値上がり→合計金額は220円上がる

3. さまざまな機械学習フレームワーク

注) 本章はあくまで初学者の理解を目的にしているため、厳密な議論を避けています

MLP （Multi-layer Perceptron）の限界

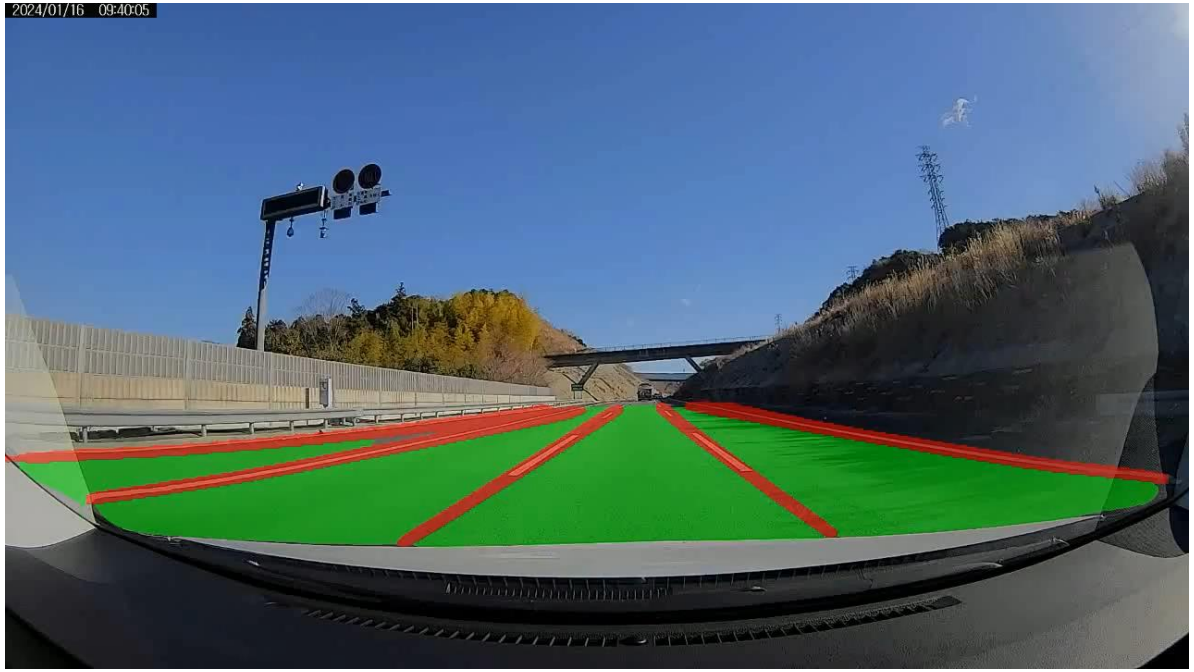
考えてみよう！

例えば…

DeepL

Google Translate

例えば…



MLPの限界

限界①: 局所構造の見落とし

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]

ここのペアを学習したい

MLP

✕

[“しています”, “専攻”, “は”, “私”, “社会基盤”, “を”]

MLPの限界

限界①: 局所構造の見落とし

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]

ここのペアを学習したい

MLP

✕

[“しています”, “専攻”, “は”, “私”, “社会基盤”, “を”]

Convolutional NN

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]

MLP　MLP　MLP　MLP

[“私”, “は”, “社会基盤”, “を”, “専攻”, “しています”]

パラメータ数爆発なども…
パラメータ数 =層数 × 隠れ次元 × 入力次元

MLPの限界

限界①: 局所構造の見落とし

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]

ここのペアを学習したい

MLP

✕

[“しています”, “専攻”, “は”, “私”, “社会基盤”, “を”]

Convolutional NN

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]

MLP MLP MLP MLP

[“私”, “は”, “社会基盤”, “を”, “専攻”, “しています”]

パラメータ数爆発なども…
パラメータ数 =層数 × 隠れ次元 × 入力次元

限界②: 長期依存性を見落とす

[“私”, “は”, “社会基盤”, “を”, “専攻”, “しています”]

CNN

[“I”, “am”, “civil”, “engineering”, ”major”]

例えば↓を扱いたい

”**The decision**, which I reached after months of consulting with faculty advisors who specialize in transportation planning and after volunteering on a community project to rebuild flood-damaged roads, **is that I am majoring in civil engineering.**” (36words)

MLPの限界

限界①: 局所構造の見落とし

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]

ここのペアを学習したい

MLP

[“しています”, “専攻”, “は”, “私”, “社会基盤”, “を”]

Convolutional NN

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]

MLP MLP MLP MLP

[“私”, “は”, “社会基盤”, “を”, “専攻”, “しています”]

パラメータ数爆発なども…
パラメータ数 =層数 × 隠れ次元 × 入力次元

限界②: 長期依存性を見落とす

[“私”, “は”, “社会基盤”, “を”, “専攻”, “しています”]

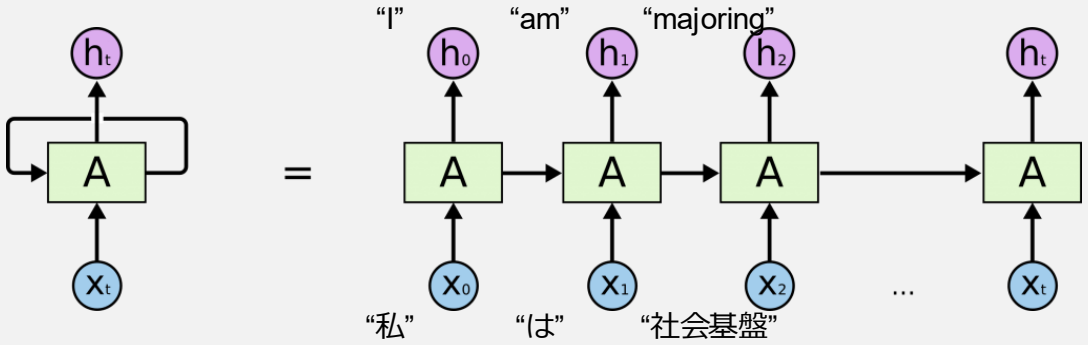
CNN

[“I”, “am”, “civil”, “engineering”, ”major”]

例えば↓を扱いたい

”**The decision**, which I reached after months of consulting with faculty advisors who specialize in transportation planning and after volunteering on a community project to rebuild flood-damaged roads, **is that I am majoring in civil engineering.**” (36words)

Recurrent NN

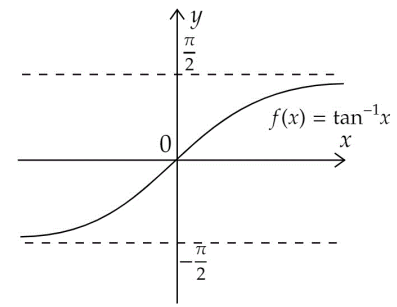


RNNのさらなる課題

例えば↓を扱いたい

”The decision, which I reached after months of consulting with faculty advisors who specialize in transportation planning and after volunteering on a community project to rebuild flood-damaged roads, is that I am majoring in civil engineering.” (36words)

RNNの課題: 勾配消失

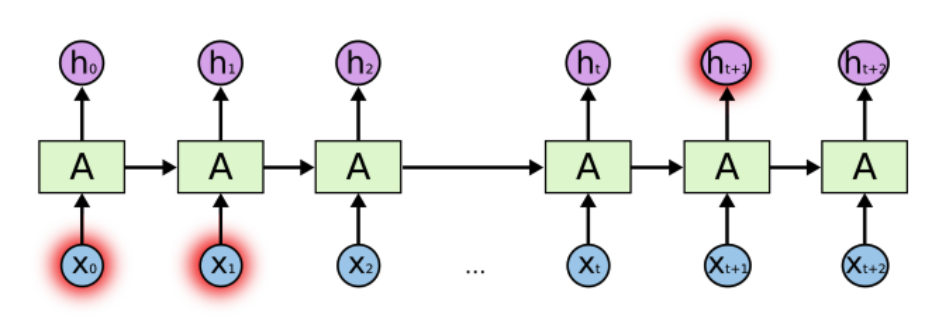


各ブロックごとに勾配を計算
→ブロック数が大きくなると勾配が0に漸近

※Tanhの勾配は0~1
勾配をかけ続けると…

$$0.5^{36} = \frac{1}{6.8 \times 10^{10}}$$

RNN Block

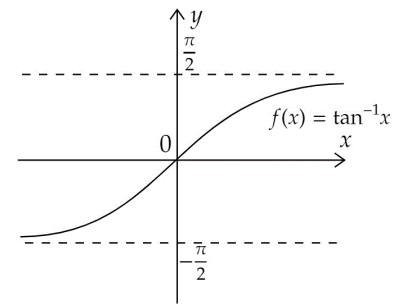


RNNのさらなる課題

例えば↓を扱いたい

"The decision, which I reached after months of consulting with faculty advisors who specialize in transportation planning and after volunteering on a community project to rebuild flood-damaged roads, is that I am majoring in civil engineering." (36words)

RNNの課題: 勾配消失

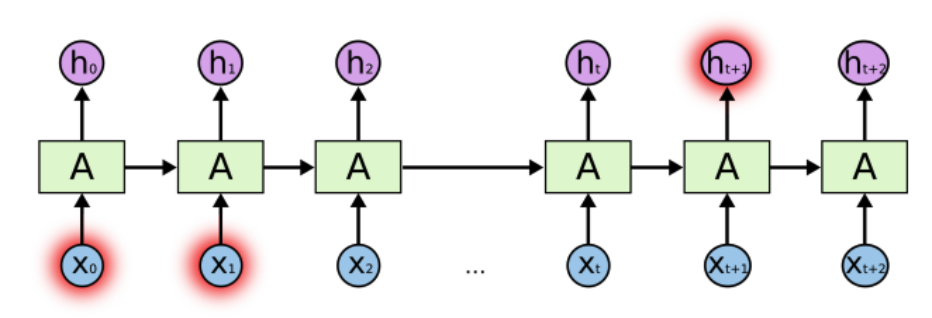


各ブロックごとに勾配を計算
→ブロック数が大きくなると勾配が0に漸近

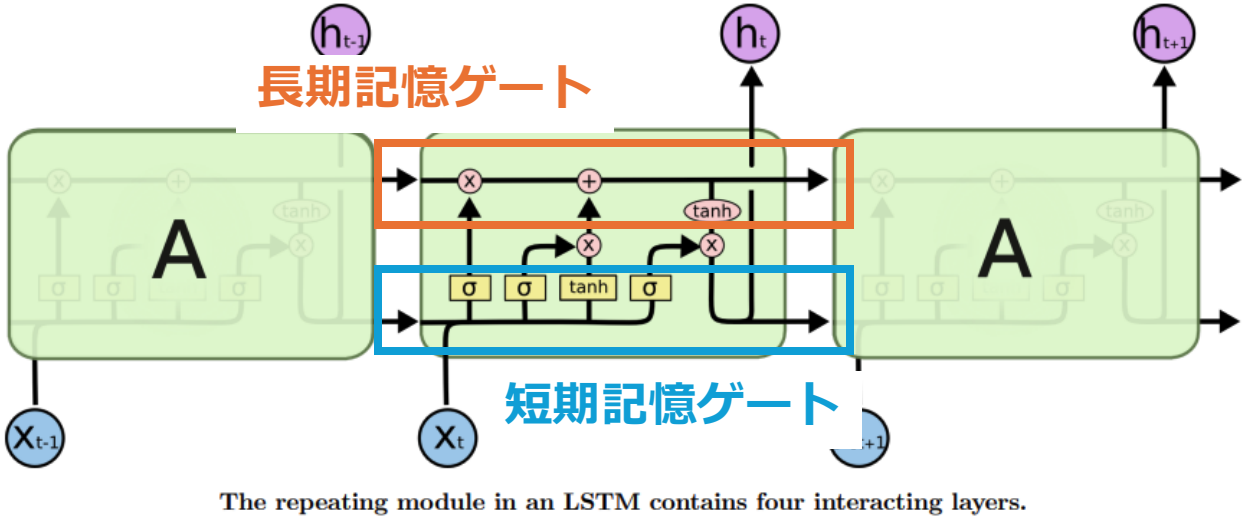
※Tanhの勾配は0~1
勾配をかけ続けると…

$$0.5^{36} = \frac{1}{6.8 \times 10^{10}}$$

RNN Block



LSTM; Long-Short Term Memory



The repeating module in an LSTM contains four interacting layers.

長期依存性に強い！
ということで一斉を風靡

次なる課題: Seq2Seq

翻訳タスクの特異性→可変長

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]: 6単語

[“私”, “は”, “社”, “会”, “基”, “盤”, “を”, “専”, “攻”, “し”, “て”, “い”, “ま”, “す”]: 14文字

→ルールベース, Word2Vec

次なる課題: Seq2Seq

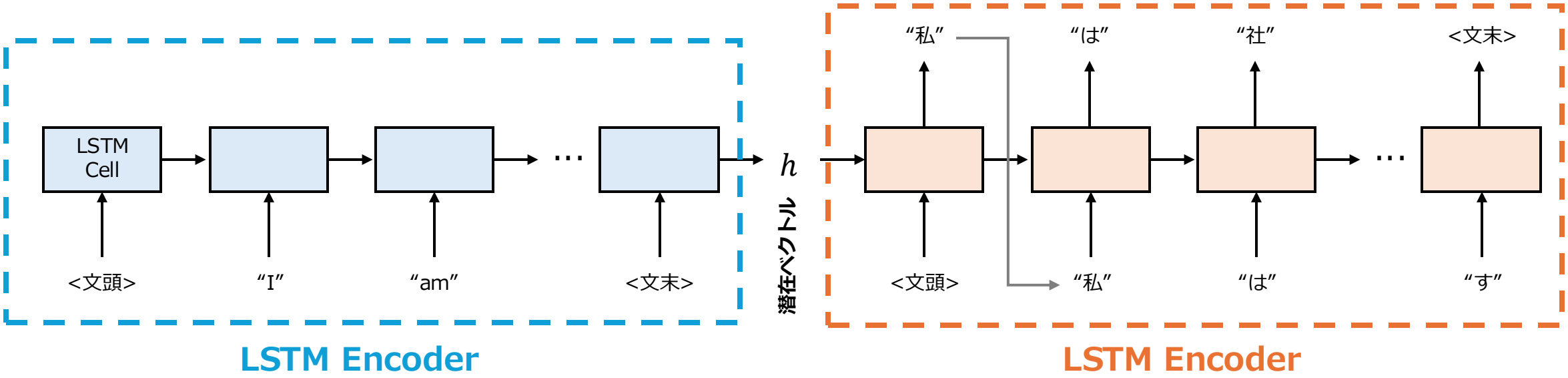
翻訳タスクの特異性→可変長

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]: 6単語

[“私”, “は”, “社”, “会”, “基”, “盤”, “を”, “専”, “攻”, “し”, “て”, “い”, “ま”, “す”]: 14文字

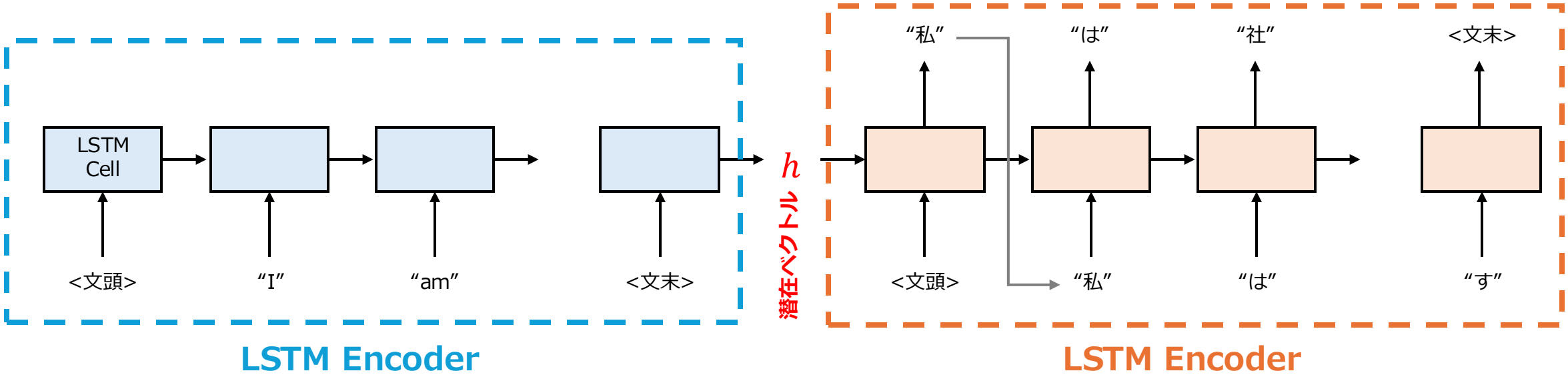
→ルールベース, Word2Vec

文同士の対応を考える**Seq2Seq**モデル(Sutskever, 2014)が提案



https://arxiv.org/pdf/1409.3215

長文処理タスクの再来



❌潜在ベクトル次元は固定

” I am majoring in civil engineering.”

→ [0.3, 0.11, 0.2, 0.9]

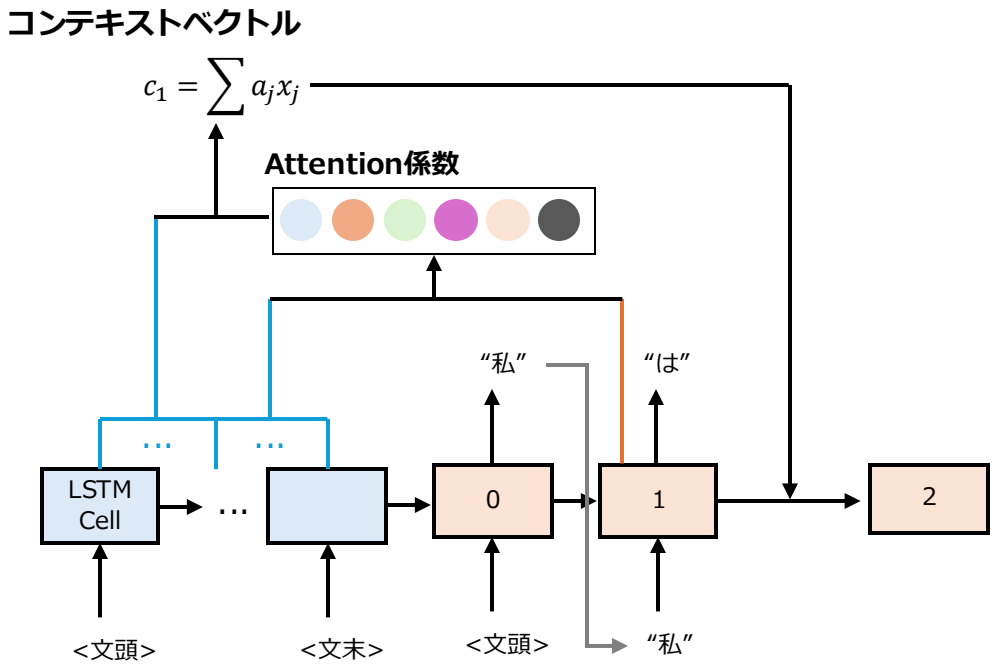
”The decision, which I reached after months of consulting with faculty advisors who specialize in transportation planning and after volunteering on a community project to rebuild flood-damaged roads, is that I am majoring in civil engineering.”

→ [0.21, 0.22, 0.24, 0.42]

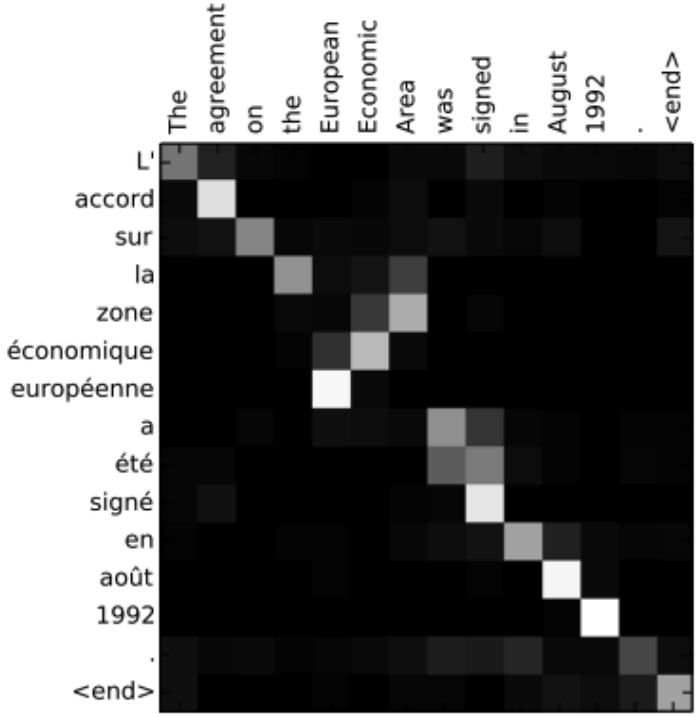
❌長文で表現力が落ちる

長文処理タスクの再来

長文で表現力が落ちる→Attention機構



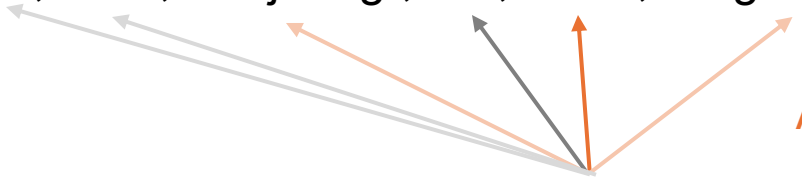
Attention Map (英→仏)



Bahdanau et al., 2015

各単語がどこに注目しているか？

[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]



Attention 👀

出力: “社会基盤”

…Transformer前夜

4. Transformer

Attention Is All You Need

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez*† University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin*‡ illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles, by over 2 BLEU. On the WMT 2014 English-to-French translation task, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature.

1 Introduction

Recurrent neural networks, long short-term memory [12] and gated recurrent [7] neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation [29, 2, 5]. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures [31, 21, 13].

*Equal contribution. Listing order is random. Jakob proposed replacing RNNs with self-attention and started the effort to evaluate this idea. Ashish, with Illia, designed and implemented the first Transformer models and has been crucially involved in every aspect of this work. Noam proposed scaled dot-product attention, multi-head attention and the parameter-free position representation and became the other person involved in nearly every detail. Niki designed, implemented, tuned and evaluated countless model variants in our original codebase and tensor2tensor. Llion also experimented with novel model variants, was responsible for our initial codebase, and efficient inference and visualizations. Lukasz and Aidan spent countless long days designing various parts of and implementing tensor2tensor, replacing our earlier codebase, greatly improving results and massively accelerating our research.

†Work performed while at Google Brain.

‡Work performed while at Google Research.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Attention is all you need

A Vaswani, N Shazeer, N Parmar… - Advances in neural …, 2017 - proceedings.neurips.cc

… to attend to **all** positions in the decoder up to and including that position. **We need** to prevent … **We** implement this inside of scaled dot-product **attention** by masking out (setting to −∞) …

☆ 保存　99 引用　被引用数: 172113　関連記事　全 73 バージョン　»

Vaswani

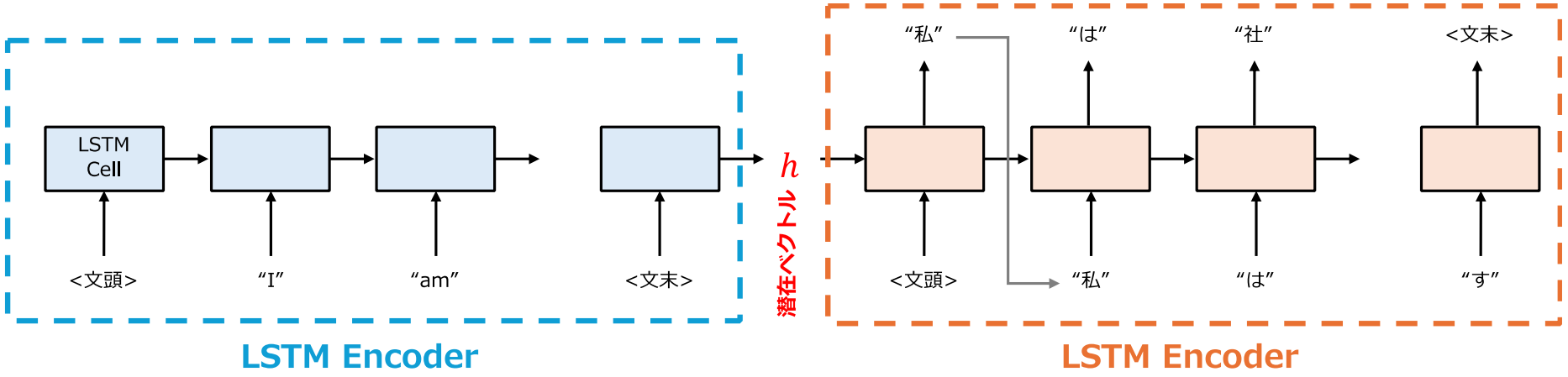
Shazeer

Parmar

DeepMind

Transformerの何がすごい？

RNNの課題: 直列計算



[“I”, “am”, “majoring”, “in”, “civil”, “engineering”]: 6単語

[“私”, “は”, “社”, “会”, “基”, “盤”, “を”, “専”, “攻”, “し”, “て”, “い”, “ま”, “す”]: 14文字

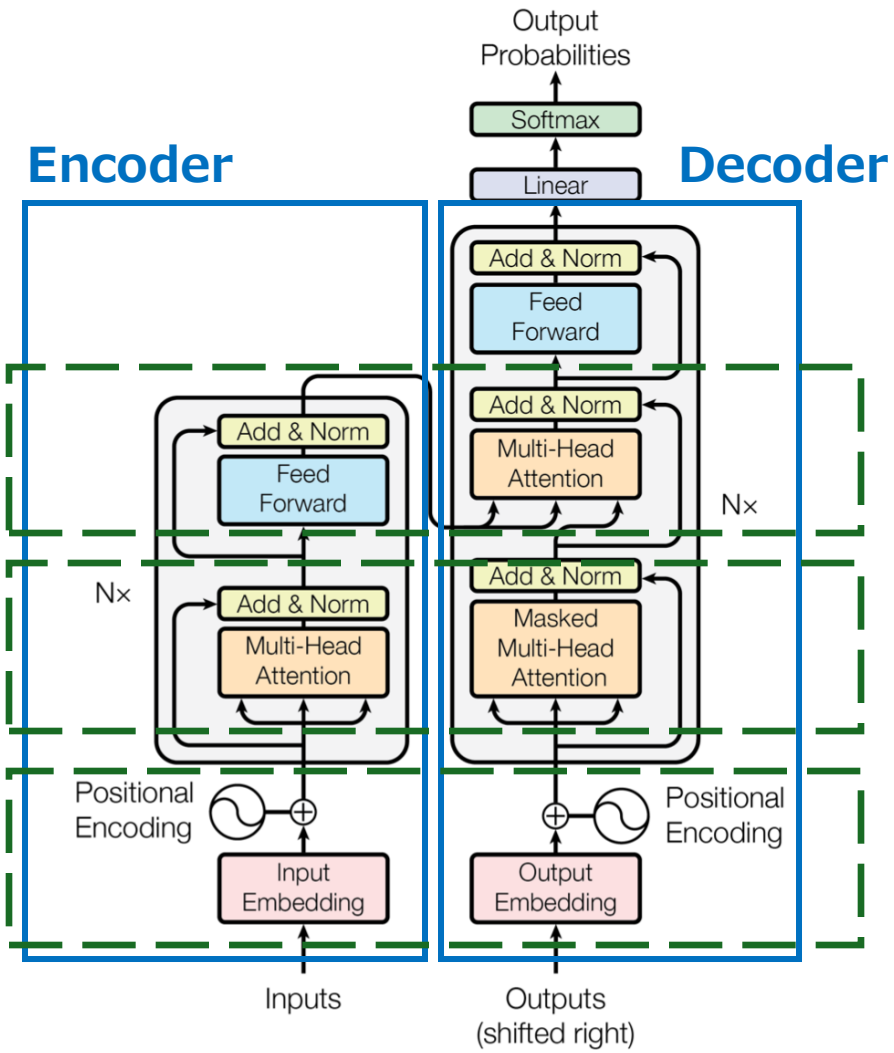
20回

すごいポイント: 入力長に対して並列化可能！！

①: Multi-head Attention

②: Self-Attention

アーキテクチャの全体像



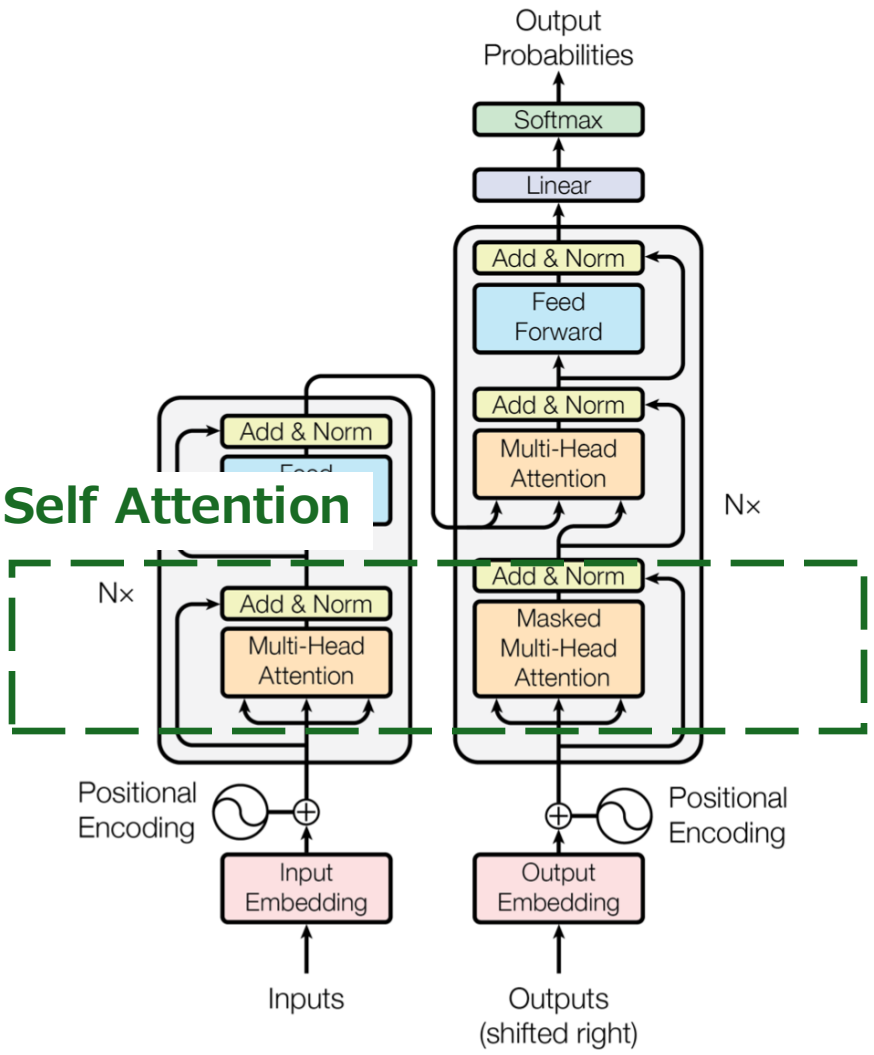
Attention②
- Inputs ✕ Outputs

Attention①
- Inputs/Outputsそれぞれ

Positional Encoding
- 単語の位置情報を追加
- トークン化

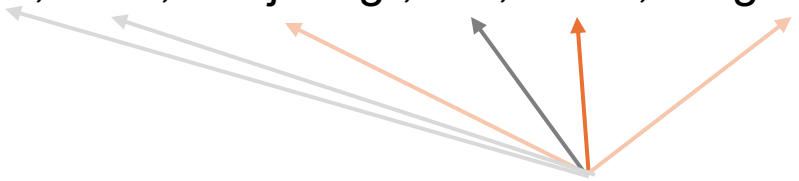
※Attention のみで計算なのでAll you Needなんですね～

Self-Attention



これまでのAttention機構: 系列ごとに計算

["I", "am", "majoring", "in", "civil", "engineering"]



出力: "社会基盤"

Self Attention: **単語**ごとに計算

["I", "am", "majoring", "in", "civil", "engineering"]

["I", "am", "majoring", "in", "civil", "engineering"]

Self-Attention

（計算例）
["I", "am", "majoring", "in", "civil", "engineering"]を計算

①エンコード: $X = \left[\binom{2}{1}, \binom{1}{2}, \dots, \binom{9}{6}\right]$ ←位置エンコード

②重み行列: W^Q, W^K, W^Vより、

- Query: $Q = XW^Q = [\mathbf{1}, 2, \dots, 6]$
- Key: $K = XW^K = [1, \mathbf{2}, \dots, 6]$
- Value: $V = XW^V = [1, 2, \dots, 6]$

を計算**（線形変換）**

補足）Q, K, Vの意味合い
Query: 他の単語への関心度
Key: 参照値
Value: 実際の単語の内容

Self-Attention

（計算例）
["I", "am", "majoring", "in", "civil", "engineering"]を計算

①エンコード: $X = \left[\binom{2}{1}, \binom{1}{2}, \ldots, \binom{9}{6}\right]$ ←位置エンコード

②重み行列: W^Q, W^K, W^Vより、
- Query: $Q = XW^Q = [\mathbf{1}, 2, \ldots, 6]$
- Key: $K = XW^K = [1, \mathbf{2}, \ldots, 6]$
- Value: $V = XW^V = [1, 2, \ldots, 6]$

を計算**（線形変換）**

補足）Q, K, Vの意味合い
Query: 他の単語への関心度
Key: 参照値
Value: 実際の単語の内容

③スケーリング:
例えば i=1: "I", j=2: "am"を計算
（ d_k はKeyの潜在次元=1とする）

$$\text{Attention: } e_{ij} = \frac{Q_i K_j}{\sqrt{d_k}} = 1 \cdot 2 = 2$$

④ソフトマックスで正規化:

$$\alpha_{ij} = \text{softmax}(e_{ij}) = 0.2$$

⑤重み付和を出力

$$O_i = \sum_j \alpha_{ij} V_j$$

Multi-head Attention

重み行列: W^Q, W^K, W^Vより、
- Query: $Q = XW^Q = [1, 2, \dots, 6]$
- Key: $K = XW^K = [1, 2, \dots, 6]$
- Value: $V = XW^V = [1, 2, \dots, 6]$

を計算**（線形変換）**

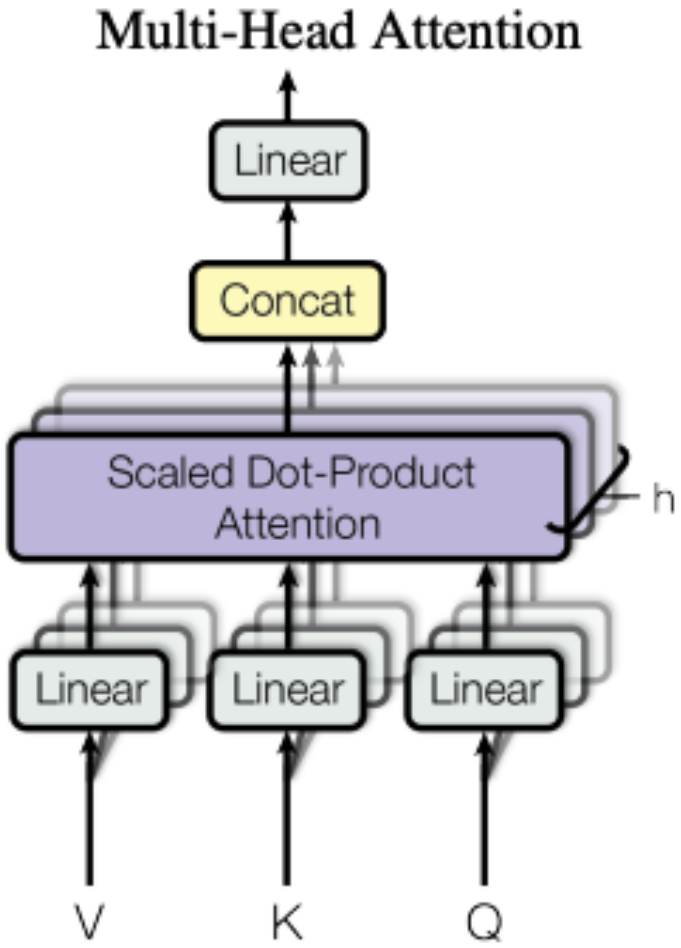
線形変換なので表現力に乏しい

→**Multi-head Attention**

複数のSelf-Attentionを同時に計算して結合
- 近くにAttentionを向けるHead
- 遠くにAttentionを向けるHead

みたいなことが表現可能

そしてまたもや並列化可能！



応用例

Routeformer (Qiu et al., 2024)（水谷理論談話会）

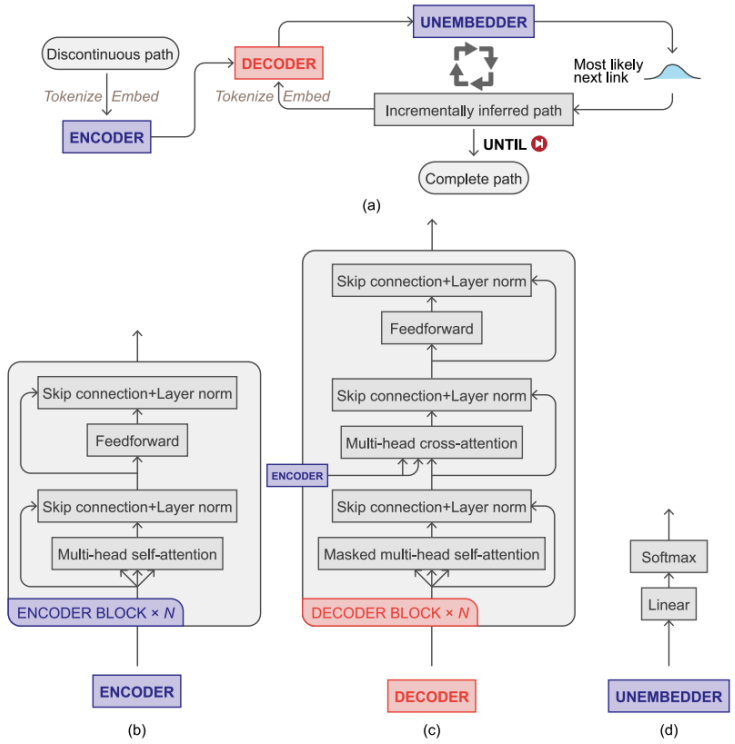


Fig. 3. Encoder–Decoder model architecture overview.

倉澤修論(2024)

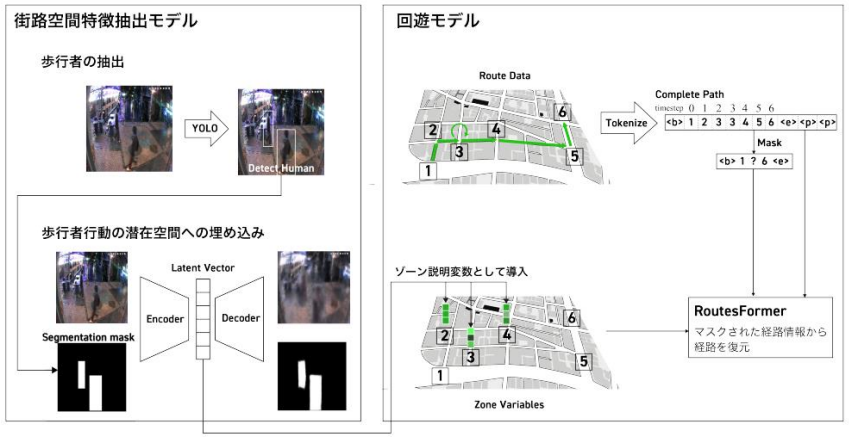


図 3.1: モデル全体のフレームワーク

などなど

課題

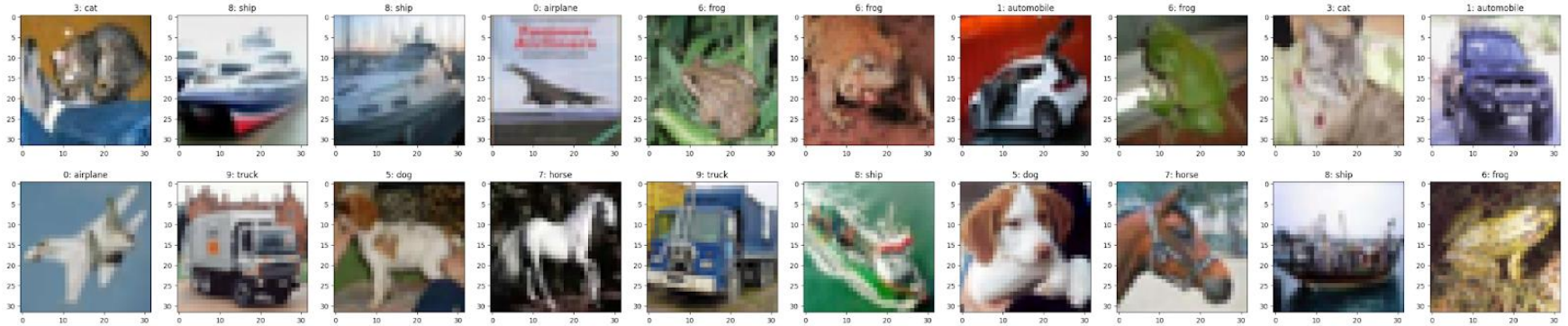
課題

必須課題+任意課題一つ以上

必須課題：画像の分類

250501_01_PytorchによるNNとCNNの実装練習.ipynb を参考に、以下の課題に取り組んでください。

1. 使用するデータセットを CIFAR-10 から MNIST に変更し、モデル構造もそれに合わせて調整してください。MNIST：https://pytorch.org/vision/stable/generated/torchvision.datasets.MNIST.html#torchvision.datasets.MNIST　https://www.kaggle.com/code/hojjatk/read-mnist-dataset
2. 学習率やバッチサイズ、エポック数などのハイパーパラメータを工夫しながら、訓練誤差と検証誤差の学習曲線を描いてください。参考：https://axa.biopapyrus.jp/machine-learning/model-evaluation/learning-curves.html
3. テスト結果として、画像とともに正解ラベル・予測ラベルを表示し、分類結果を可視化してください。



任意課題１・２：画像の分類（続き）

1. 精度向上を目指して、データの追加や別のモデルの導入（例：ViT）、データ拡張（既存の画像を加工）など自由に工夫してみましょう。

2. 自分でデータセットを作成し、その分類予測に挑戦してみましょう。

任意課題３：機械学習の活用事例を調べる

3. 実際の研究で、機械学習がどのように使われているかを論文を探して調べてみましょう。

調べる例

- 何をしている研究か、どのような目的で機械学習が使われているか？
- 使用されている機械学習の手法は？
  例：CNN、RNN、GNN、Transformer、クラスタリング など
- 学習に使われたデータの種類や特徴は？
  例：画像／テキスト／GPS／動画など
- その研究における機械学習の強み・課題は？

Appendix.

詳しく学びたい方

書籍（研究室にあるもの, 簡単順）

- ゼロから作るDeep Learning
- Kaggleで勝つデータ分析の技術
- 深層学習
- パターン認識と機械学習

学会（情報系は学会投稿文化が盛ん）

- 人工知能研究センター@関西学院大
- AAAI, NeuralIPS, ICMLなど