

# 分子のグラフ表現と機械学習

---

2020年10月20日(火)：日本化学会 第10回CSJ化学フェスタ2020  
データサイエンスの世界をのぞいてみませんか？

たきがわ いちがく

瀧川一学

<https://itakigawa.github.io/>

北海道大学 化学反応創成研究拠点 (WPI-ICReDD)

# 本日の内容

---

GNN(Graph Neural Networks)でグラフ表現された分子を  
入力にして機械学習するやり方をのぞいてみませんか？

- 機械学習とは？
- 分子のグラフ表現
- 準備：深層学習のココロ (forwardとbackward)
- Graph Neural Networksとは？
- 自分でやってみるための情報
- おまけ：  
最近の話 (計算限界、分子生成、事前学習、...)

# 本日の内容

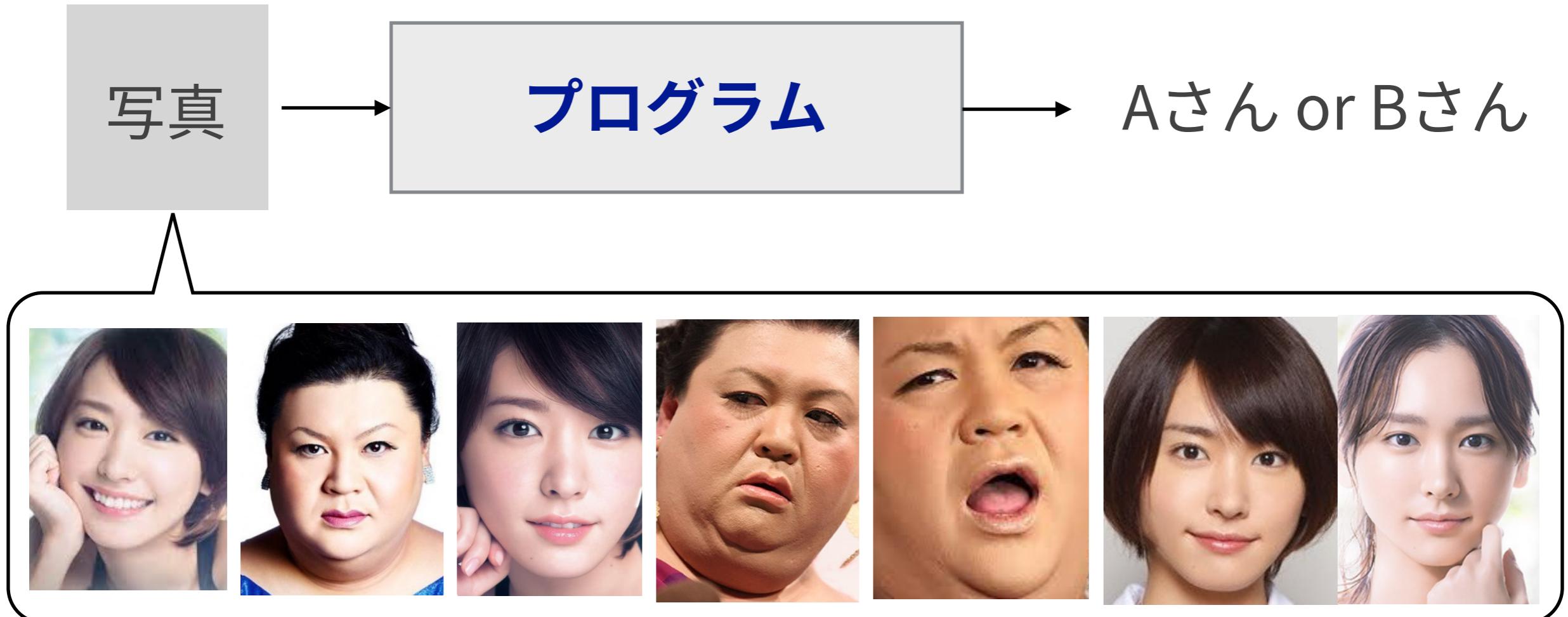
---

GNN(Graph Neural Networks)でグラフ表現された分子を  
入力にして機械学習するやり方をのぞいてみませんか？

- 機械学習とは？
- 分子のグラフ表現
- 準備：深層学習のココロ (forwardとbackward)
- Graph Neural Networksとは？
- 自分でやってみるための情報
- おまけ：  
最近の話 (計算限界、分子生成、事前学習、...)

# 機械学習の出番

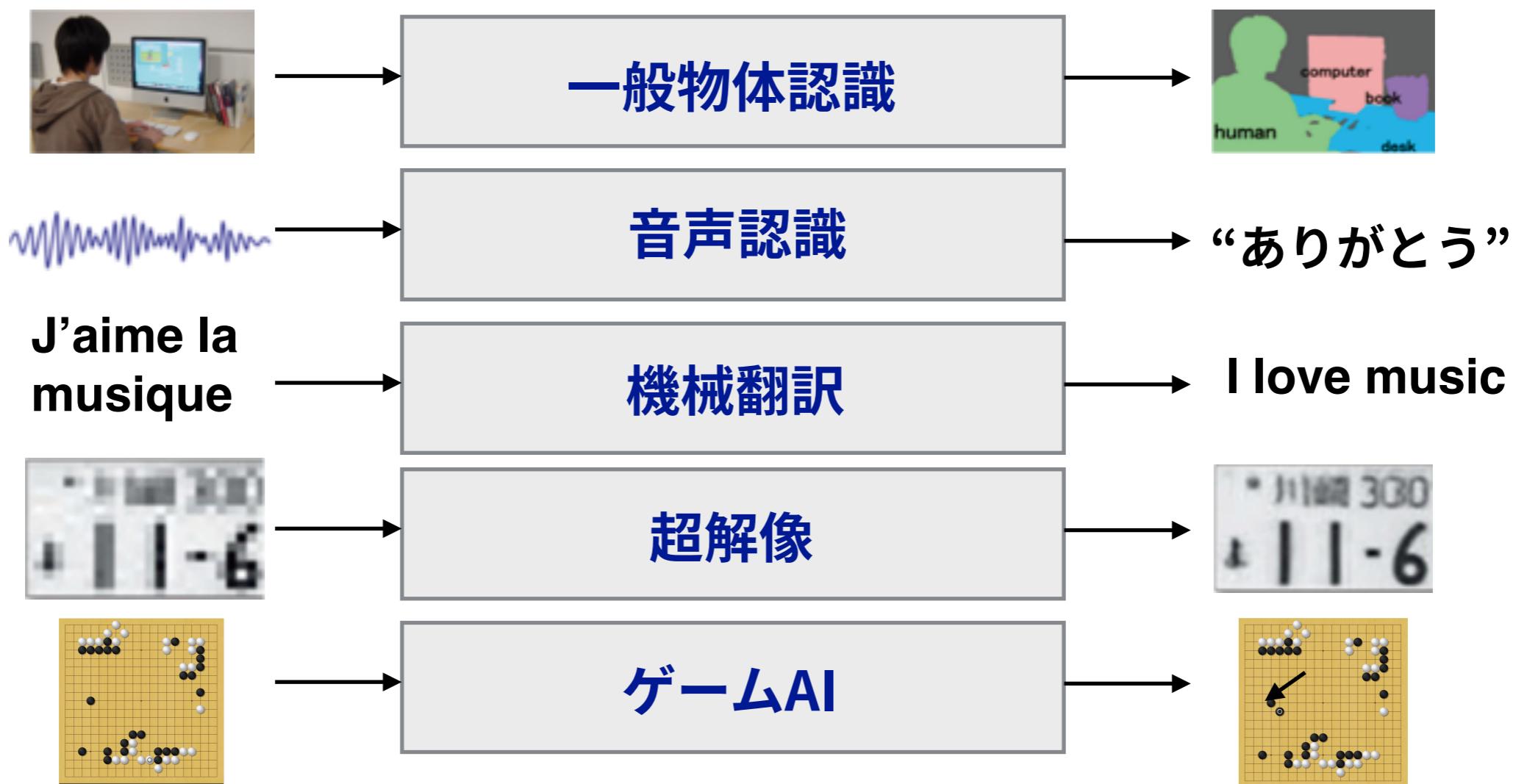
AさんとBさんの写真を分けるプログラムを作りたい



人間なら簡単に解けるけど手順は自分でも説明できない...

# 機械学習：新しいプログラミングのパラダイム

機械学習 = 多量の入出力の見本例からプログラムを自動的に作成する(いいかげんな)技術



# 今日の例：分子の活性や物性の予測 (QSAR/QSPR)



入出力の  
見本例

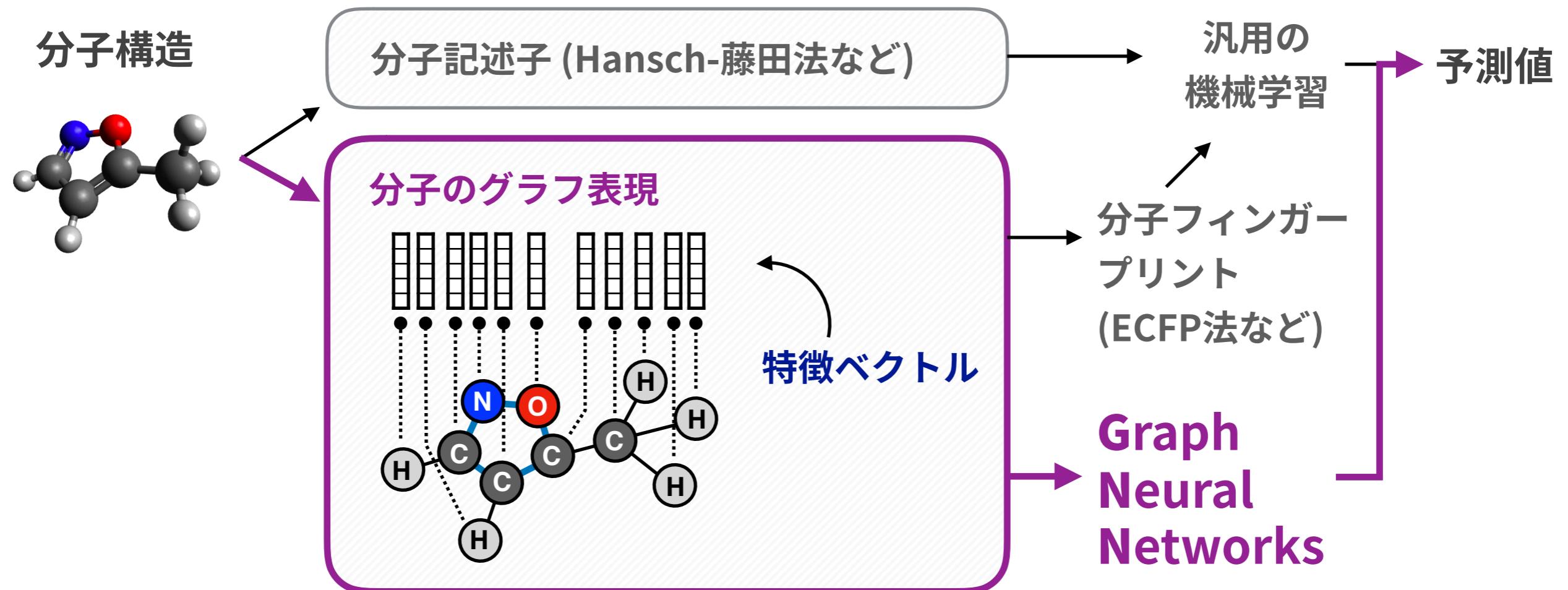


⋮

# 分子のグラフ表現とGraph Neural Networks (GNNs)

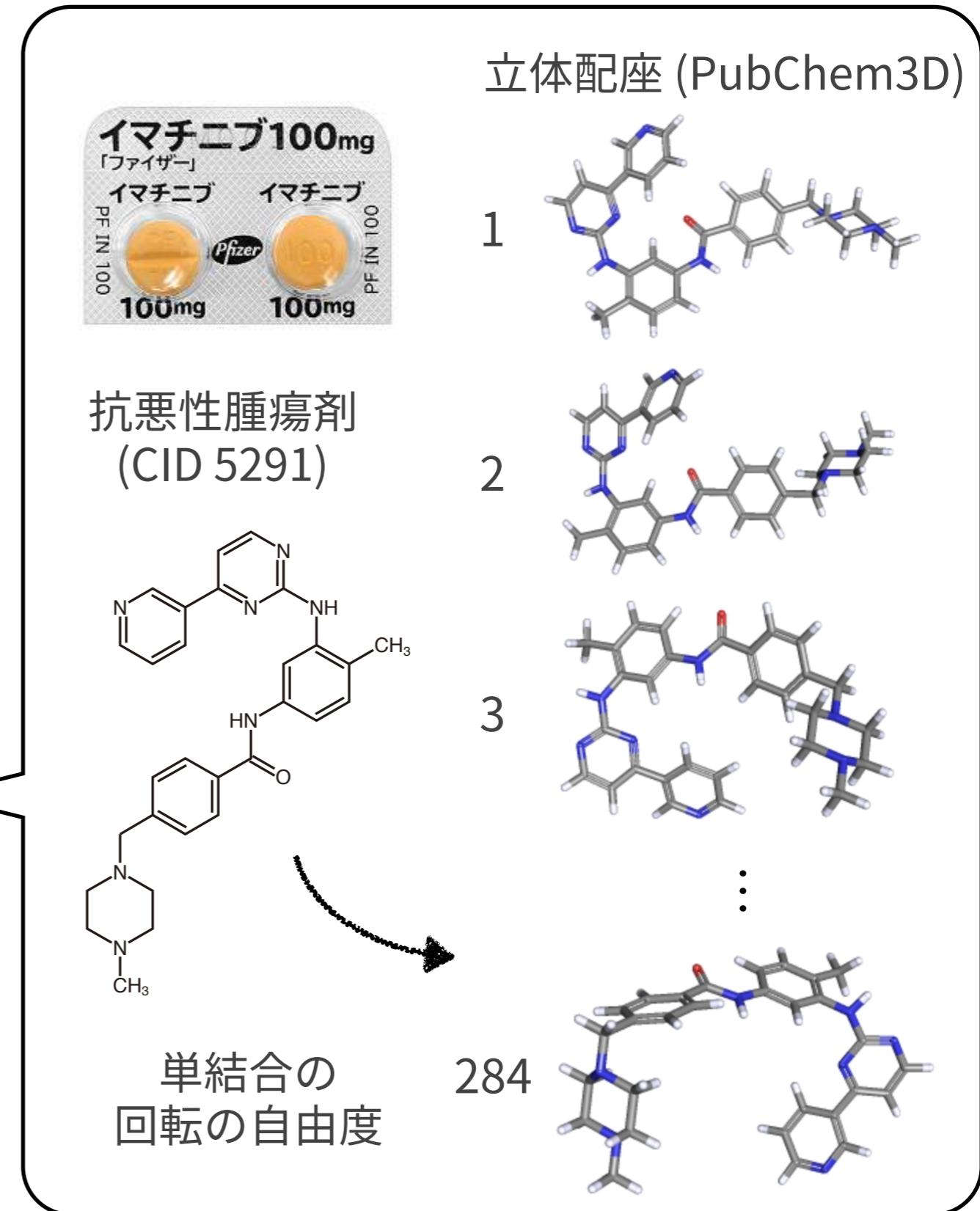
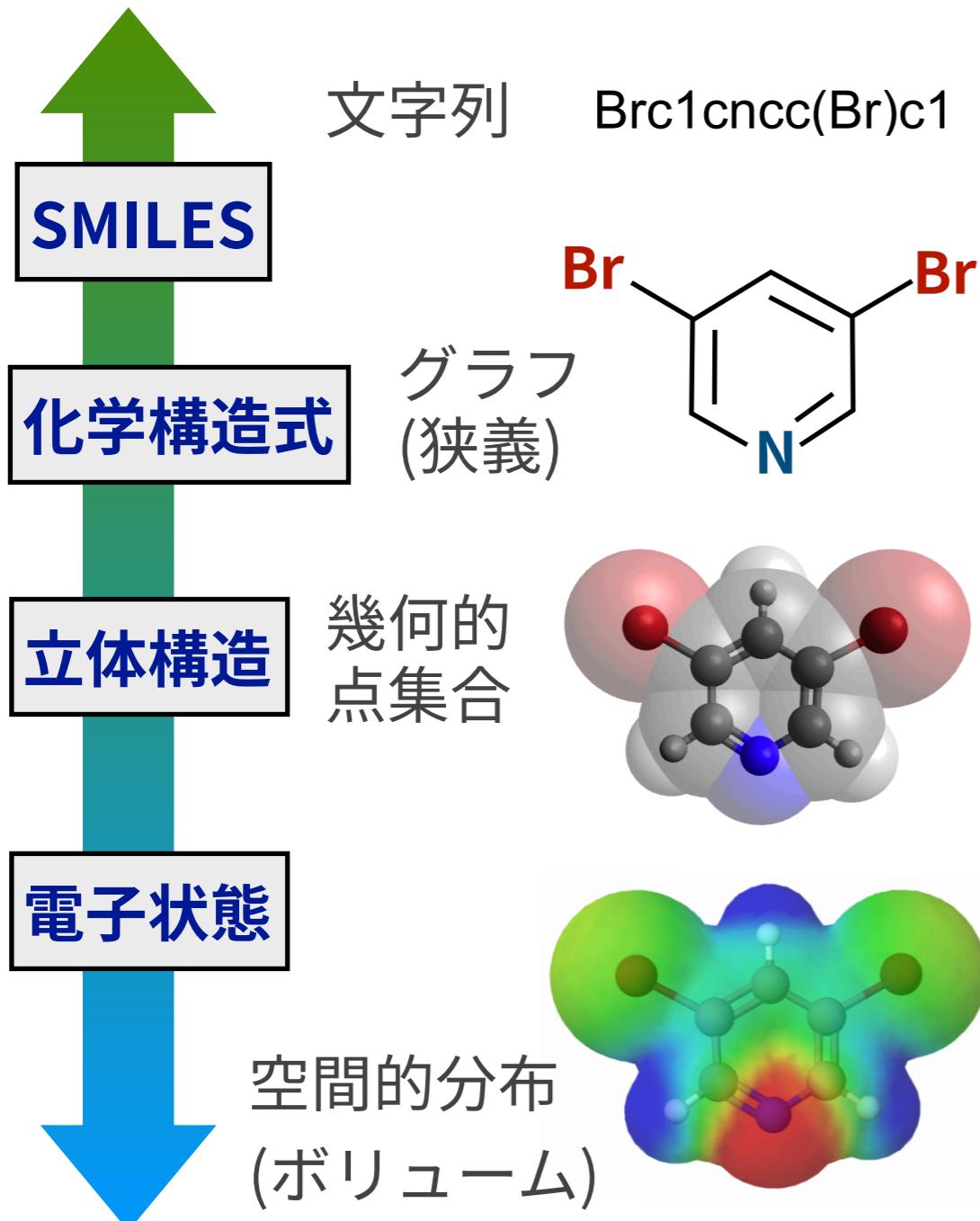


色々やり方あるけれど今日はGNNをのぞいてみませんか？



# 動機：GNNは広いモダリティを統一的に扱うことができる

GNNで扱えるモダリティは  
狭義の「グラフ」より広い



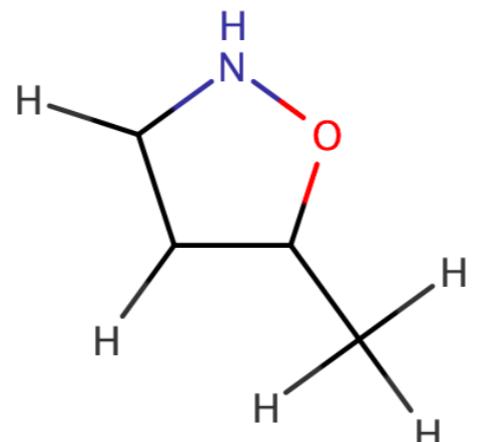
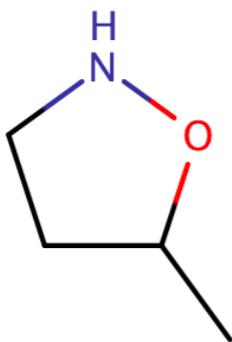
# 分子表現

1次元

CC1CCNO1

Canonical SMILES

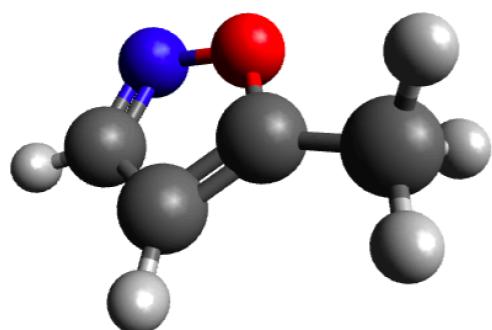
2次元



水素(Implicit)

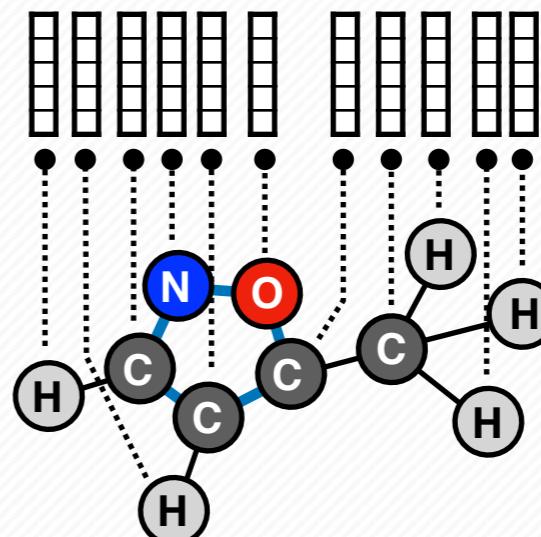
水素(Explicit)

3次元



MOL形式

分子のグラフ表現



特徴ベクトル

11	11	0	0	0	0	999
0.0264		1.4944		0.0287	C	
-0.0850		0.0111		0.0449	C	
-1.0957		-0.8588		-0.2308	C	
-0.5082		-2.1326		-0.0113	C	
0.7433		-2.0451		0.3632	N	
1.0156		-0.6777		0.3997	O	
0.8032		1.8213		-0.6702	H	
-0.9230		1.9401		-0.2734	H	
0.2923		1.8782		1.0190	H	
-2.1005		-0.6271		-0.5415	H	
-0.9559		-3.1113		-0.1136	H	
1	2	1	0	0	0	0
1	9	1	0	0	0	0
2	6	1	0	0	0	0
3	4	1	0	0	0	0
3	2	2	0	0	0	0
4	5	2	0	0	0	0
5	6	1	0	0	0	0
7	1	1	0	0	0	0
8	1	1	0	0	0	0
10	3	1	0	0	0	0
11	4	1	0	0	0	0

原子核のxyz

化学結合

# 実際の分子グラフデータを見てみよう

---

## QM9 Dataset (Ramakrishnan+ 2014)

133,885 分子：C, N, O, Fの組合せから成る9原子(H以外)までの分子の安定3D構造と15種の物性値を計算 (B3LYP/6-31G(2df,p) level)

SCIENTIFIC DATA 

OPEN  
SUBJECT CATEGORIES  
» Quantum chemistry  
» Density functional theory

Quantum chemistry structures and properties of 134 kilo molecules

Raghunathan Ramakrishnan<sup>1</sup>, Pavlo O. Dral<sup>2,3</sup>, Matthias Rupp<sup>1</sup> & O. Anatole von Lilienfeld<sup>1,4</sup>

*Sci Data* 1, 140022 (2014). <https://doi.org/10.1038/sdata.2014.22>

# 実際の分子グラフデータを見てみよう

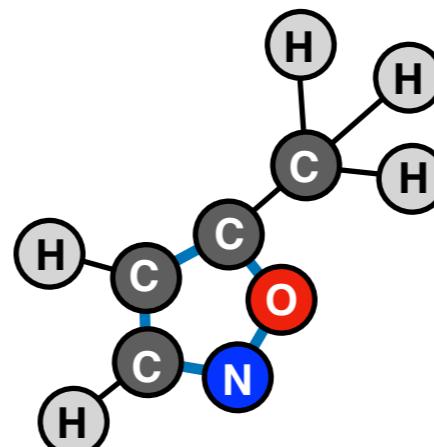
```
import matplotlib.pyplot as plt  
import networkx as nx  
import torch_geometric as pyg
```

```
data = pyg.datasets.QM9(root='/tmp/QM9')  
print('分子グラフの数:', len(data))
```

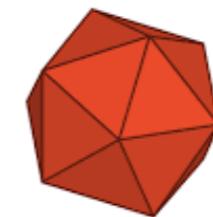
分子グラフの数: 130831

```
mol = data[717] ← 718番目の  
mol.name  
'gdb_727'
```

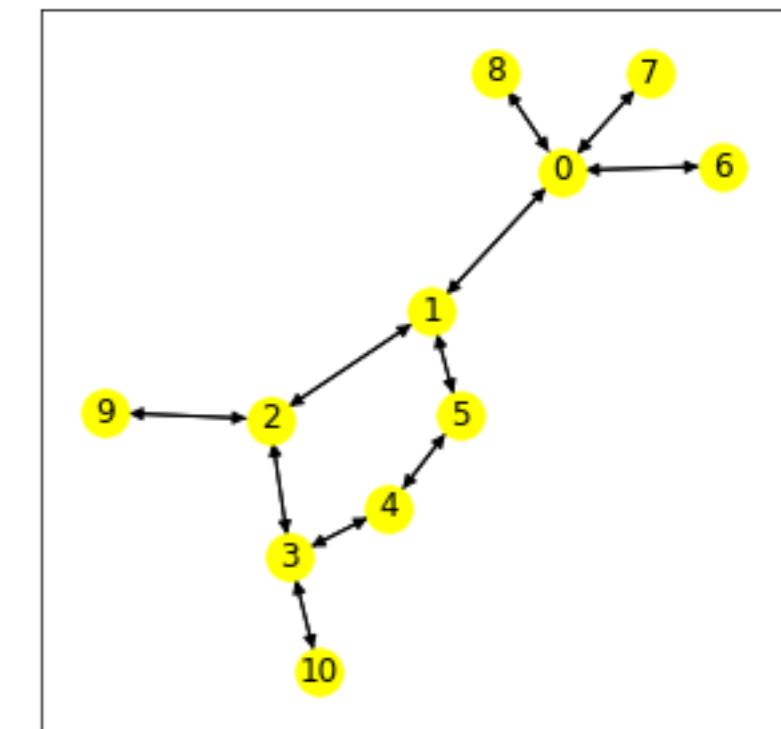
この分子の  
molを観察



```
g = pyg.utils.convert.to_networkx(mol)  
fig, ax = plt.subplots(figsize=(5, 5))  
nx.draw_networkx(g, node_color="yellow")  
fig.show()
```



PyTorch  
geometric



# 実際の分子グラフデータを見てみよう

mol.y

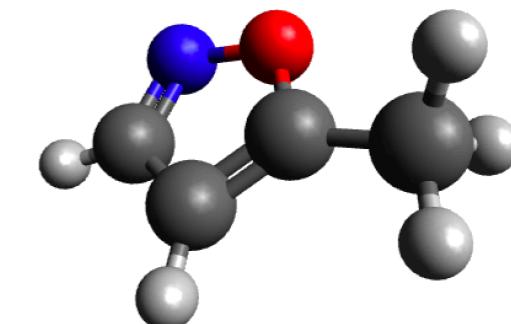
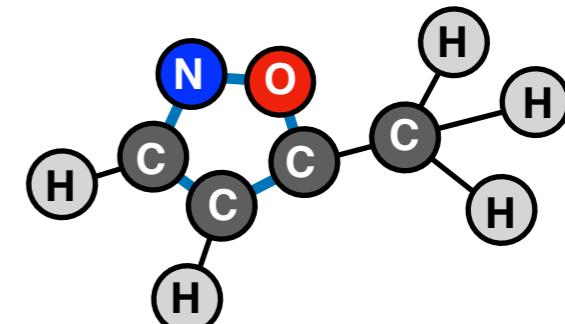
```
tensor([[ 3.1194e+00,  4.8060e+01, -6.8763e+00, -1.9320e-01,  6.6831e+00,
         4.8378e+02,  2.3288e+00, -7.7632e+03, -7.7631e+03, -7.7631e+03,
        -7.7640e+03,  1.8293e+01, -4.7811e+01, -4.8092e+01, -4.8349e+01,
       -4.4744e+01,  9.2803e+00,  3.5644e+00,  2.6172e+00]])
```

```
import pandas as pd
pd.Series(mol.y.numpy()[0])
```

0	3.119400	Dipole moment
1	48.060001	Isotropic polarizability
2	-6.876316	Highest occupied molecular orbital (HOMO) energy
3	-0.193201	Lowest unoccupied molecular orbital (LUMO) energy
4	6.683115	Gap between HOMO and LUMO
5	483.779388	Electronic spatial extent
6	2.328750	Zero point vibrational energy
7	-7763.241699	Internal energy at 0K
8	-7763.098633	Internal energy at 298.15K
9	-7763.073242	Enthalpy at 298.15K
10	-7764.022949	Free energy at 298.15K
11	18.292999	Heat capacity at 298.15K
12	-47.811089	Atomization energy at 0K
13	-48.092346	Atomization energy at 298.15K
14	-48.349361	Atomization enthalpy at 298.15K
15	-44.743744	Atomization free energy at 298.15K
16	9.280260	Rotational constant A
17	3.564360	Rotational constant B
18	2.617160	Rotational constant C

タテに表示

これらのDFT計算値を  
(DFTしないで)予測するタスク



# 実際の分子グラフデータを見てみよう

```
mol.edge_index
```

```
tensor([[ 0,  0,  0,  0,  1,  1,  1,  2,  2,  2,  3,  3,  3,  4,  4,  5,  5,  6,
         7,  8,  9, 10],
        [ 1,  6,  7,  8,  0,  2,  5,  1,  3,  9,  2,  4, 10,  3,  5,  1,  4,  0,
         0,  0,  2,  3]])
```

```
for id, (s, t) in enumerate(zip(*mol.edge_index)):
    print(f'Edge: {id}, node[{s}] -> node[{t}]')
```

```
Edge: 0, node[ 0 ] -> node[ 1 ]
```

```
Edge: 1, node[ 0 ] -> node[ 6 ]
```

```
Edge: 2, node[ 0 ] -> node[ 7 ]
```

```
Edge: 3, node[ 0 ] -> node[ 8 ]
```

```
Edge: 4, node[ 1 ] -> node[ 0 ]
```

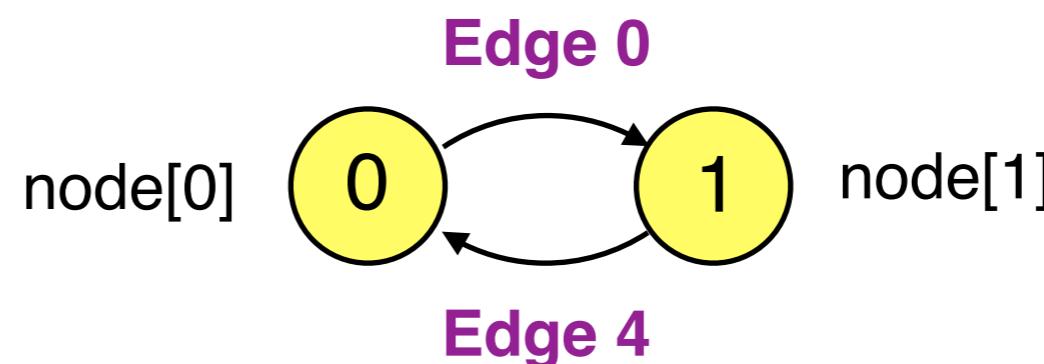
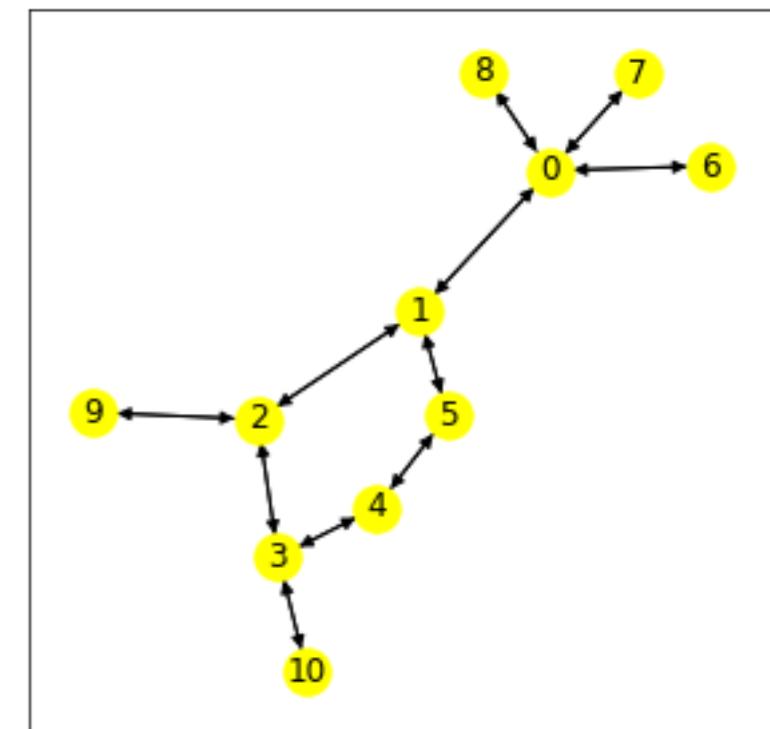
```
Edge: 5, node[ 1 ] -> node[ 2 ]
```

```
:
```

```
Edge: 20, node[ 9 ] -> node[ 2 ]
```

```
Edge: 21, node[10] -> node[ 3 ]
```

辺の向きも  
考慮できる



# 実際の分子グラフデータを見てみよう

mol.z 頂点の原子番号

```
tensor([6, 6, 6, 6, 7, 8, 1, 1, 1, 1, 1])
```

## mol.x 頂点の特徴ベクトル(11次元)

```
tensor([[0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 0., 3.],
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 7., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 8., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0.]])
```

## mol.pos 頂点の位置ベクトル (3次元)

```
tensor([[ 0.0264,  1.4944,  0.0287],  
       [-0.0850,  0.0111,  0.0449],  
       [-1.0957, -0.8588, -0.2308],  
       [-0.5082, -2.1326, -0.0113],  
       [ 0.7433, -2.0451,  0.3632],  
       [ 1.0156, -0.6777,  0.3997],  
       [ 0.8032,  1.8213, -0.6702],  
       [-0.9230,  1.9401, -0.2734],  
       [ 0.2923,  1.8782,  1.0190],  
       [-2.1005, -0.6271, -0.5415],  
       [-0.9559, -3.1113, -0.1136]])
```

## 辺の特徴ベクトル(4次元)

`mol.edge_attr`

Single  
Double  
Triple  
Aromatic

(one-hot  
encoding)

# 実際の分子グラフデータを見てみよう

mol.z 頂点の原子番号

```
tensor([6, 6, 6, 6, 7, 8, 1, 1, 1, 1])
```

mol.x 頂点の特徴ベクトル (11次元)

```
tensor([[0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 3.],
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 1.],
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0., 7., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 8., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.]])
```

mol.pos 頂点の位置ベクトル (3次元)

```
tensor([[ 0.0264,  1.4944,  0.0287],
       [-0.0850,  0.0111,  0.0449],
       [-1.0957, -0.8588, -0.2308],
       [-0.5082, -2.1326, -0.0113],
       [ 0.7433, -2.0451,  0.3632],
       [ 1.0156, -0.6777,  0.3997],
       [ 0.8032,  1.8213, -0.6702],
       [-0.9230,  1.9401, -0.2734],
       [ 0.2923,  1.8782,  1.0190],
       [-2.1005, -0.6271, -0.5415],
       [-0.9559, -3.1113, -0.1136]])
```

辺の特徴ベクトル (4次元)

mol.edge\_attr

```
tensor([[1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.]]))
```

Single  
Double  
Triple  
Aromatic

(one-hot  
encoding)

# 実際の分子グラフデータを見てみよう

mol.x

頂点の特徴ベクトル (11次元)

```
tensor([[0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 3.],  
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 0.],  
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 1.],  
       [0., 1., 0., 0., 0., 6., 0., 0., 0., 0., 1.],  
       [0., 0., 1., 0., 0., 7., 0., 0., 0., 0., 0.],  
       [0., 0., 0., 1., 0., 8., 0., 0., 0., 0., 0.],  
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
       [1., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0.]])
```

0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10

H C N O F  
(one-hot  
encoding)

原子番号  
(one-hot  
encoding)

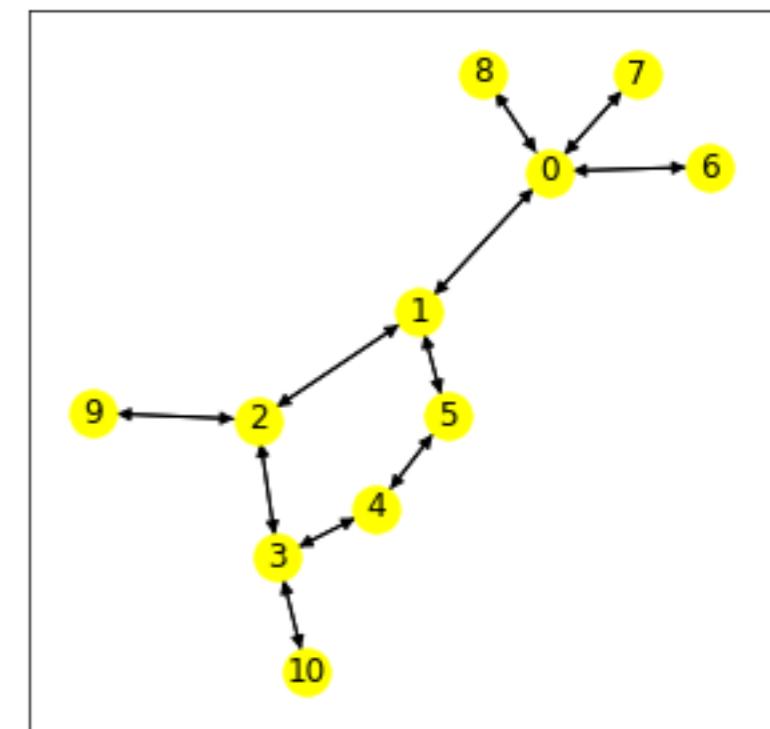
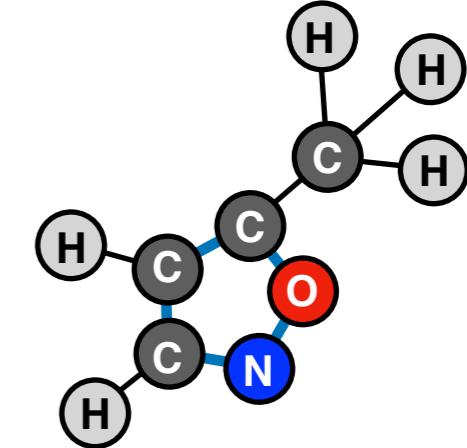
aromatic? (0/1)

sp? (0/1)

sp<sup>2</sup>? (0/1)

sp<sup>3</sup>? (0/1)

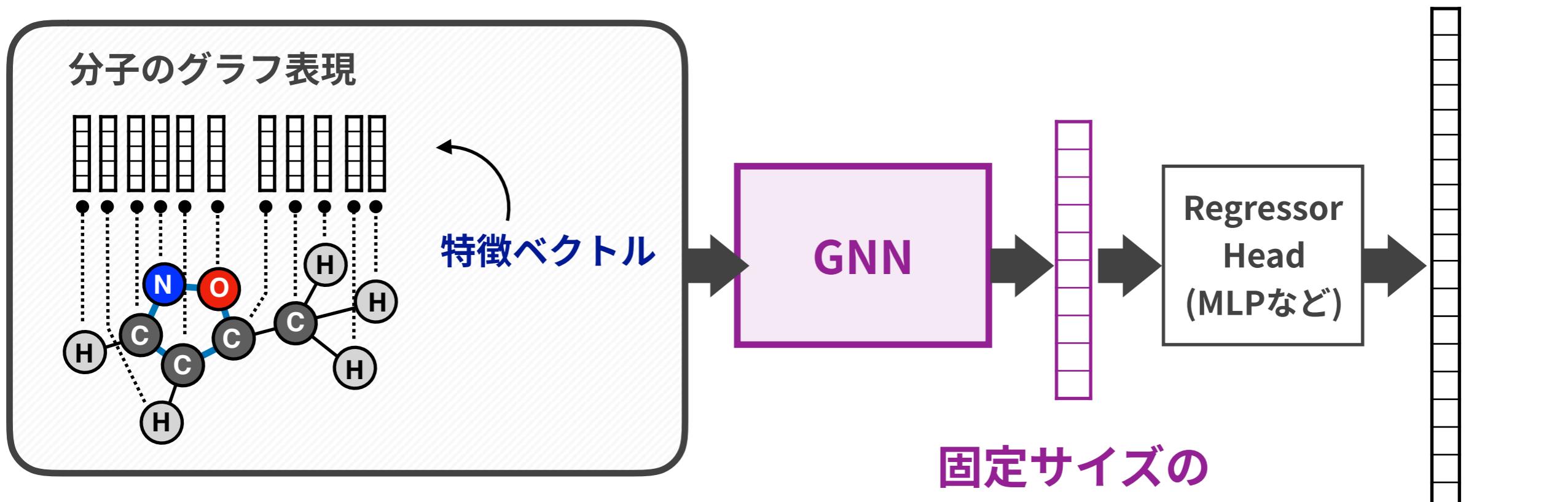
結合するHの数



これ以外にもいろいろ有りえることに注意

# GNNがやること

可変サイズの分子グラフから「固定サイズのベクトル表現」への変換を学習



**mol.x** 頂点の特徴ベクトル (11次元)

**mol.pos** 頂点の位置ベクトル (3次元)

**mol.edge\_attr** 辺の特徴ベクトル (4次元)

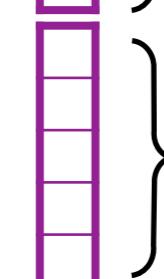
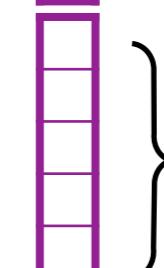
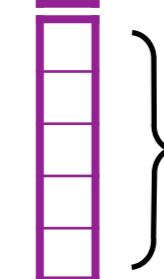
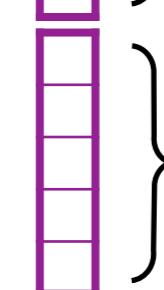
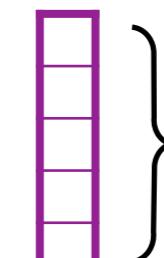
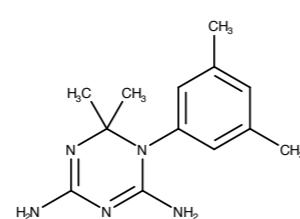
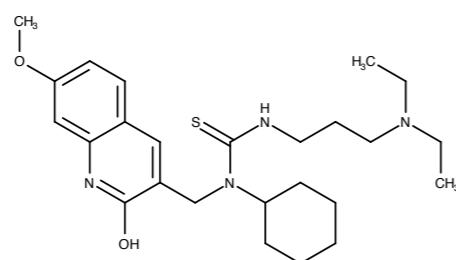
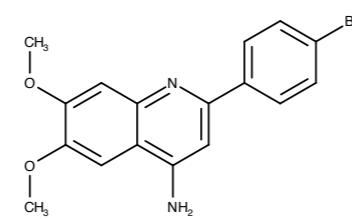
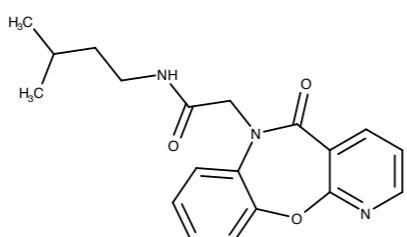
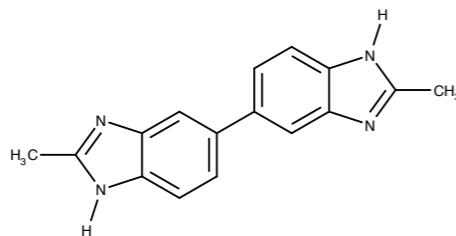
**mol.edge\_index** 辺と結合原子ペアの対応

**mol.y**  
DFT計算値  
(19次元)

# 核となる技術上の要件

違うサイズの分子グラフを入力しても  
いつも「固定サイズのベクトル表現」が出力されないといけない

頂点数や  
変数が異なる  
分子グラフ



いつも  
同じサイズの  
ベクトルを出力

# 本日の内容

---

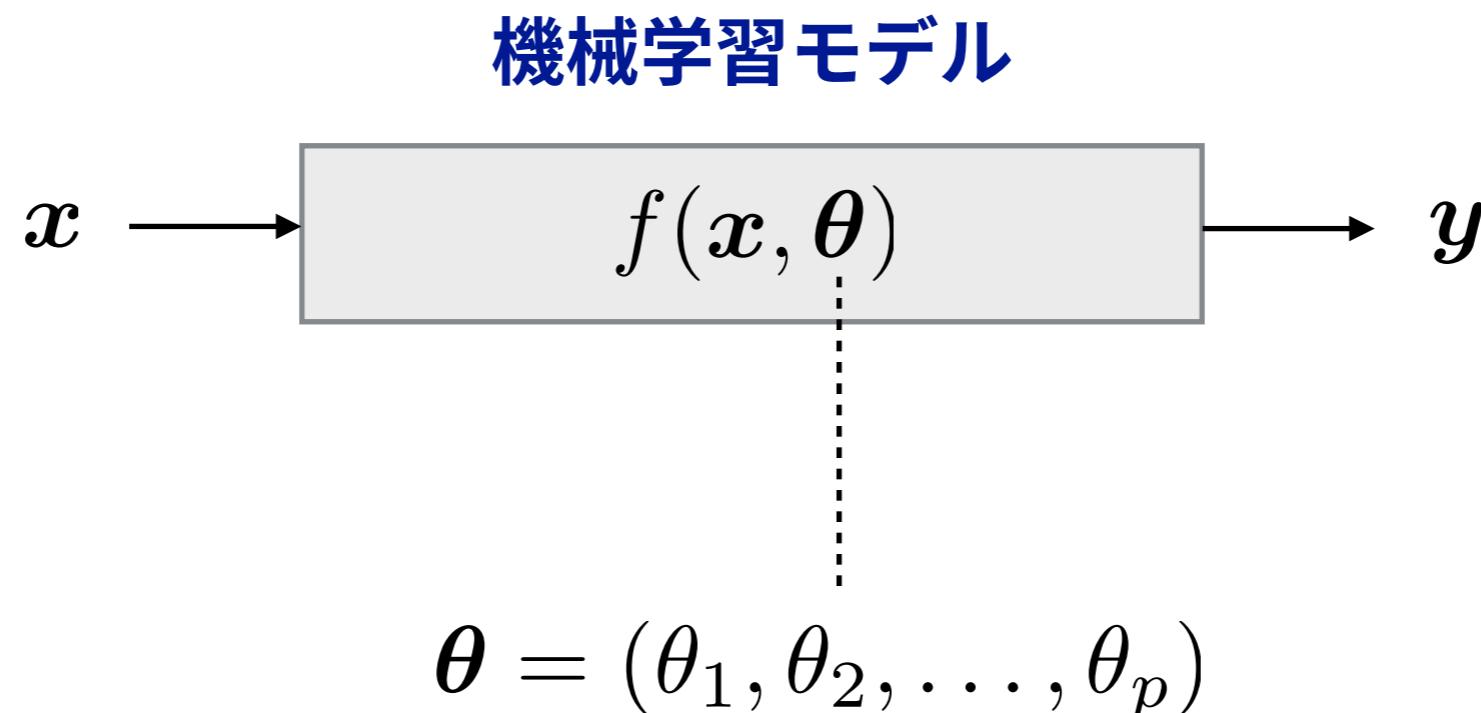
GNN(Graph Neural Networks)でグラフ表現された分子を  
入力にして機械学習するやり方をのぞいてみませんか？

- 機械学習とは？
- 分子のグラフ表現
- 準備：深層学習のココロ (forwardとbackward)
- Graph Neural Networksとは？
- 自分でやってみるための情報
- おまけ：  
最近の話 (計算限界、分子生成、事前学習、...)

# 機械学習のモデル

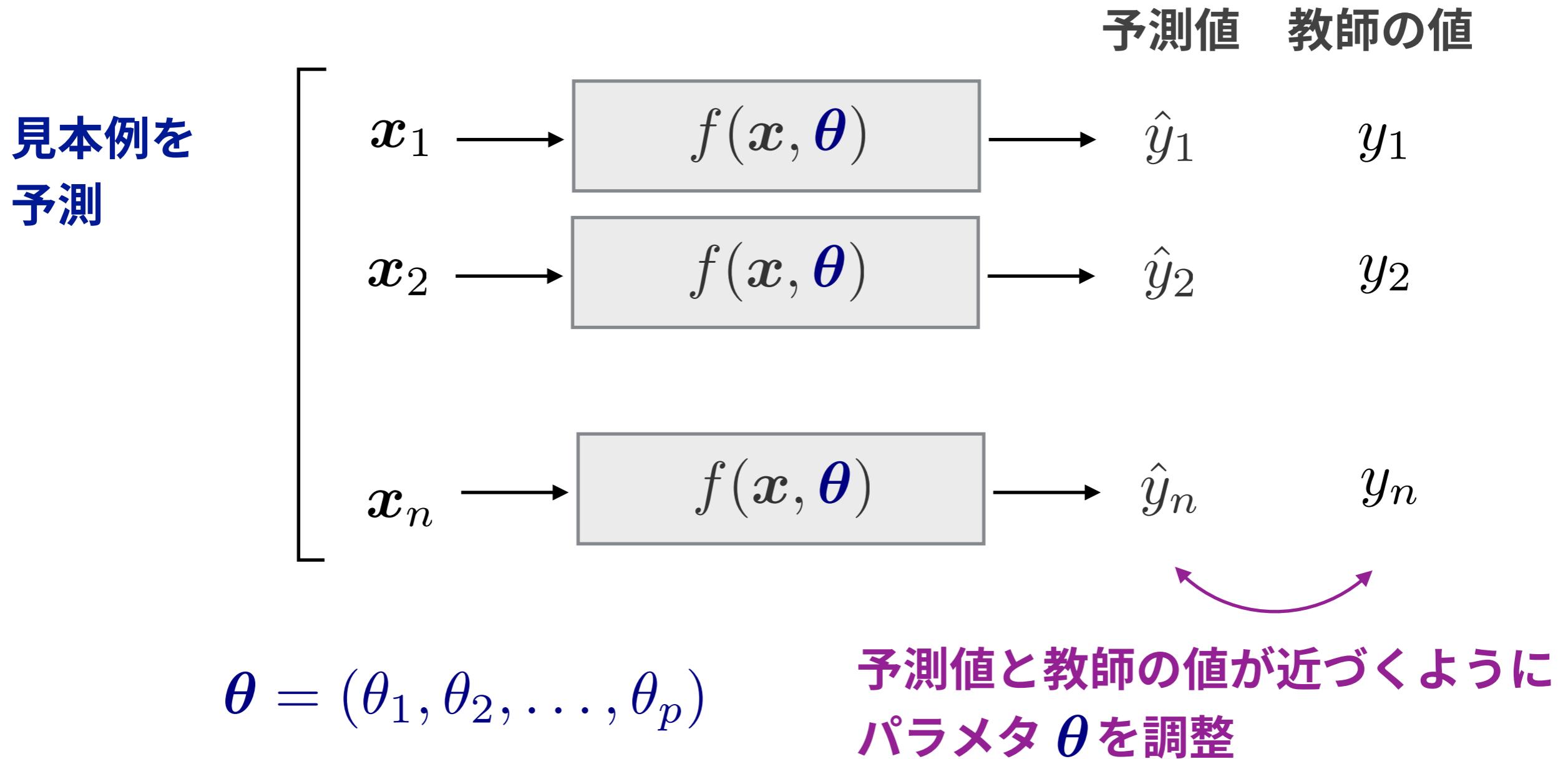
---

機械学習 = 多量の入出力の見本例からプログラムを自動的に作成する(いいかげんな)技術



多数のパラメタの値を動かして  
出力が望む値になるようにする

# モデルの学習 = パラメタの値を最適にする



# モデルの学習 = パラメタの値を最適にする

---

n個の見本の全体について  
予測値と教師の値が近づくように  
パラメタ  $\theta$  を調整

# モデルの学習 = パラメタの値を最適にする

n個の見本の全体について  
予測値と教師の値が近づくように  
パラメタ  $\theta$  を調整

関数  $\text{loss}(\theta)$  の値が小さくなる  
ようにパラメタ  $\theta$  を調整

$$\text{loss}(\theta) = \sum_{i=1}^n (\hat{y}_i - y_i)^2$$



# モデルの学習 = パラメタの値を最適にする

n個の見本の全体について  
予測値と教師の値が近づくように  
パラメタ  $\theta$  を調整

パラメタ調整のやりかた：

関数  $\text{loss}(\theta)$  の値が小さくなる  
ようにパラメタ  $\theta$  を調整

$$\text{loss}(\theta) = \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

- $\theta$  をランダム値で初期化
- $\text{loss}(\theta)$  の値が小さくなるように  $\theta$  をちょっとだけ変える

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix} + 0.01 \times \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \\ \vdots \\ \Delta\theta_p \end{bmatrix}$$

くりかえし

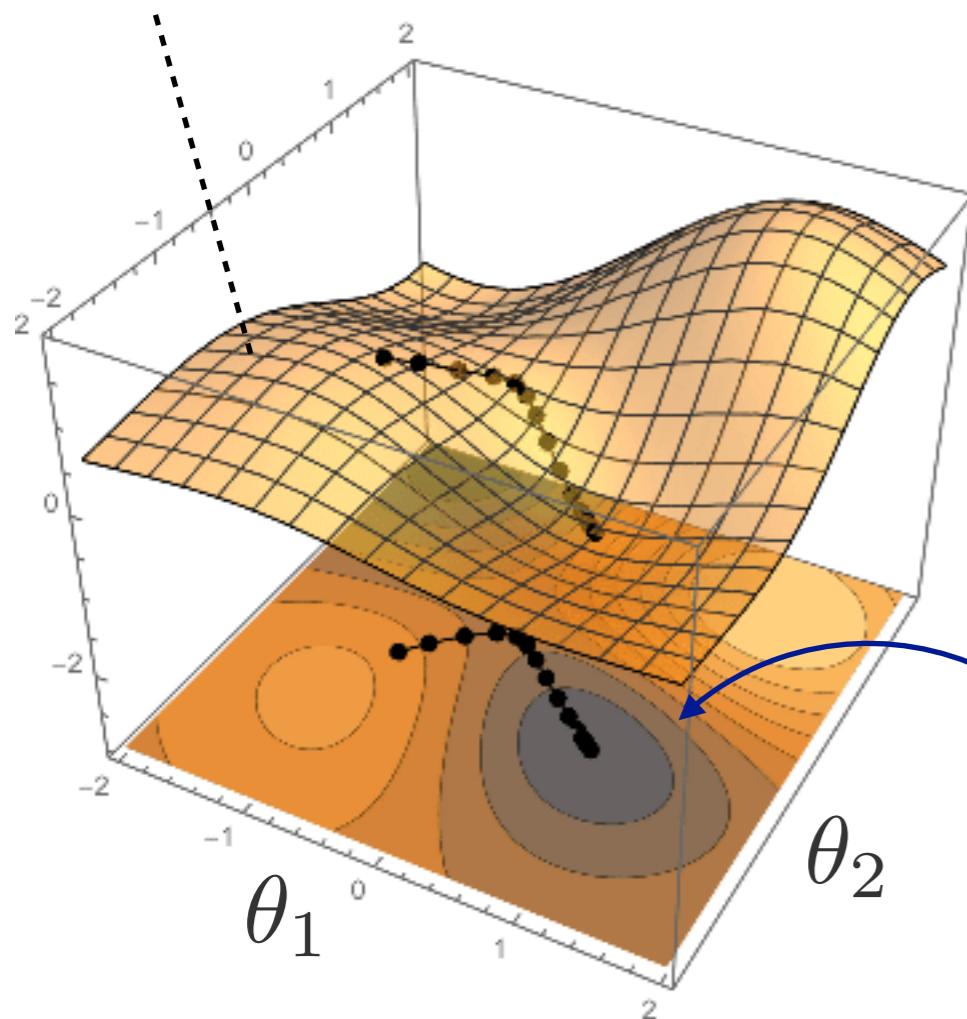
学習率  
(step size)

# 勾配降下法 (Gradient Descent)

- loss( $\theta$ ) の値が小さくなるように  $\theta$  をちょっとだけ変える

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} - 0.01 \times \begin{bmatrix} f'_1(\theta_1, \theta_2) \\ f'_2(\theta_1, \theta_2) \end{bmatrix}$$

loss( $\theta_1, \theta_2$ )



勾配ベクトル

$$f'_1(\theta_1, \theta_2) = \frac{\partial \text{loss}(\theta_1, \theta_2)}{\partial \theta_1}$$

$$f'_2(\theta_1, \theta_2) = \frac{\partial \text{loss}(\theta_1, \theta_2)}{\partial \theta_2}$$

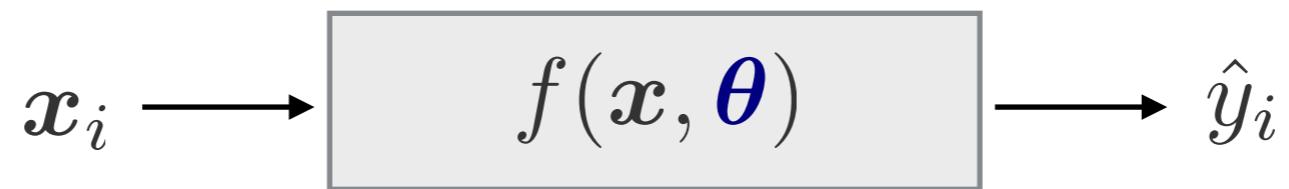
偏微分

$\theta_1$  や  $\theta_2$  を  
少し動かしたときの  
loss( $\theta_1, \theta_2$ ) の変化量

勾配ベクトルに比例する量をちょっと引くと  
最寄りの極小値に向かうことがわかっている

# 計算グラフと自動微分：複雑なモデルの勾配の計算

機械学習モデル



小さくしたい尺度

$$\text{loss}(\theta) = \sum_{i=1}^n (\hat{y}_i - y_i)^2$$

$$= \sum_{i=1}^n (f(x_i, \theta) - y_i)^2$$

勾配ベクトル

$$\begin{bmatrix} \partial \text{loss}(\theta) / \partial \theta_1 \\ \partial \text{loss}(\theta) / \partial \theta_2 \\ \vdots \\ \partial \text{loss}(\theta) / \partial \theta_p \end{bmatrix} \leftarrow \text{これをどう計算するかという話}$$

# 計算グラフ：素演算の合成関数のグラフ表現

---

例  $\text{loss} = (\hat{y} - y)^2$  予測モデル (特に意味はないToy Example)



$$\hat{y} = f(x_1, x_2, \theta_1, \theta_2, \theta_3)$$

$$= \theta_1 x_1 - \theta_2 \cos(\theta_3 + x_2)$$

# 計算グラフ：素演算の合成関数のグラフ表現

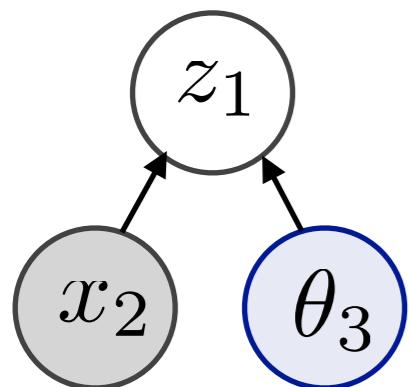
例  $\text{loss} = (\hat{y} - y)^2$  予測モデル (特に意味はないToy Example)



$$\hat{y} = f(x_1, x_2, \theta_1, \theta_2, \theta_3)$$

$$= \theta_1 x_1 - \theta_2 \cos(\underbrace{\theta_3 + x_2}_{z_1})$$

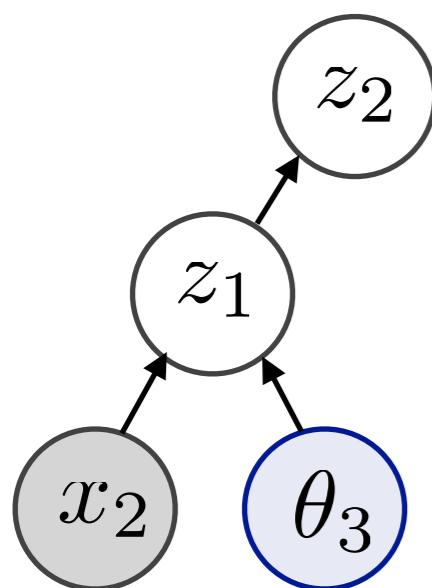
合成関数  $z_1 = \theta_3 + x_2$



# 計算グラフ：素演算の合成関数のグラフ表現

例  $\text{loss} = (\hat{y} - y)^2$  予測モデル (特に意味はないToy Example)

$$\begin{aligned}\hat{y} &= f(x_1, x_2, \theta_1, \theta_2, \theta_3) \\ &= \theta_1 x_1 - \theta_2 \cos(\underbrace{\theta_3 + x_2}_{z_1})\end{aligned}$$



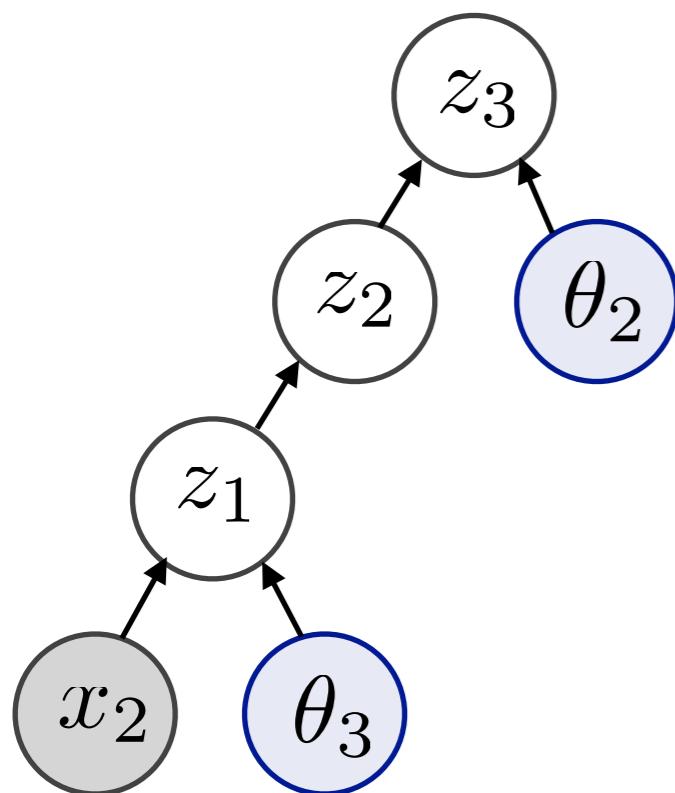
合成関数  $z_1 = \theta_3 + x_2$   
 $z_2 = \cos(z_1)$

# 計算グラフ：素演算の合成関数のグラフ表現

例  $\text{loss} = (\hat{y} - y)^2$  予測モデル (特に意味はないToy Example)

$$\begin{aligned}\hat{y} &= f(x_1, x_2, \theta_1, \theta_2, \theta_3) \\ &= \theta_1 x_1 - \theta_2 \cos(\theta_3 + x_2)\end{aligned}$$

$\underbrace{\phantom{\theta_1 x_1 - \theta_2 \cos(\theta_3 + x_2)}}_{z_1}$   
 $\underbrace{\phantom{\theta_1 x_1 - \theta_2 \cos(\theta_3 + x_2)}}_{z_2}$   
 $\underbrace{\phantom{\theta_1 x_1 - \theta_2 \cos(\theta_3 + x_2)}}_{z_3}$



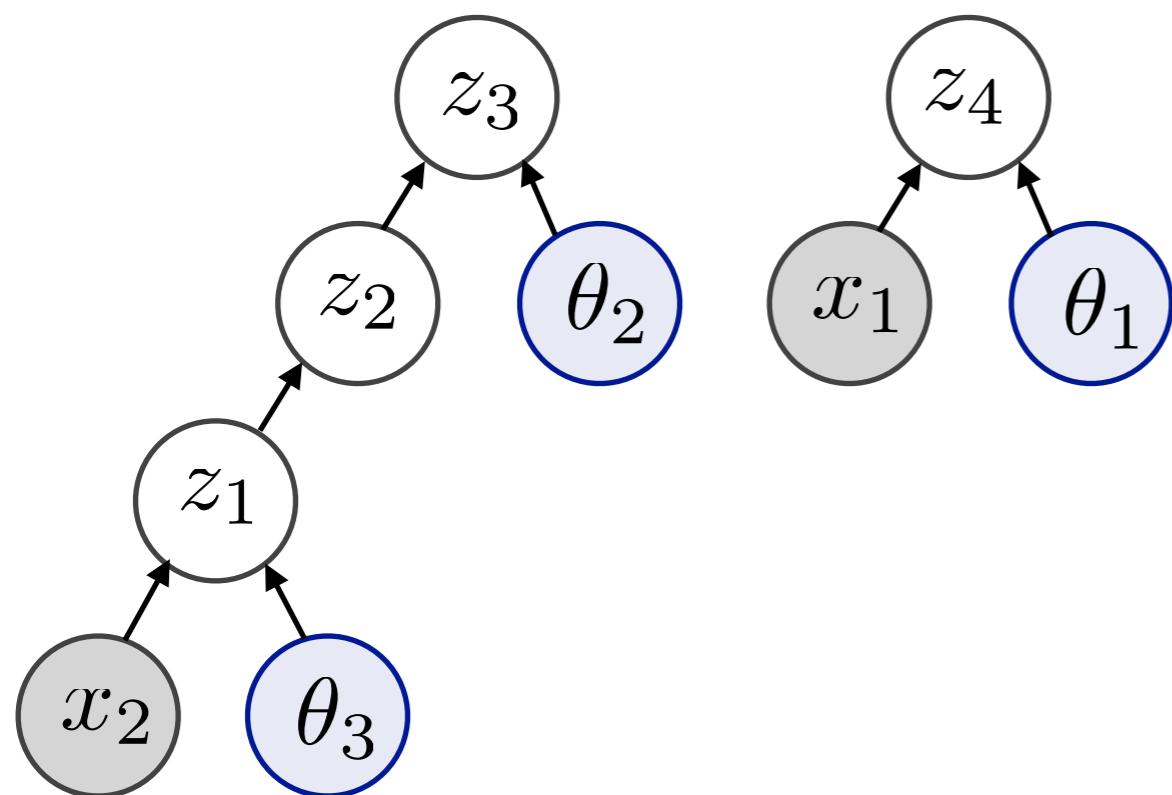
合成関数

$$\begin{aligned}z_1 &= \theta_3 + x_2 \\ z_2 &= \cos(z_1) \\ z_3 &= \theta_2 z_2\end{aligned}$$

# 計算グラフ：素演算の合成関数のグラフ表現

例  $\text{loss} = (\hat{y} - y)^2$  予測モデル (特に意味はないToy Example)

$$\begin{aligned}\hat{y} &= f(x_1, x_2, \theta_1, \theta_2, \theta_3) \\ &= \underbrace{\theta_1 x_1}_{z_4} - \theta_2 \cos(\underbrace{\theta_3 + x_2}_{z_1}) \\ &\quad \quad \quad \underbrace{\qquad\qquad\qquad}_{z_2} \\ &\quad \quad \quad \quad \quad \underbrace{\qquad\qquad\qquad}_{z_3}\end{aligned}$$



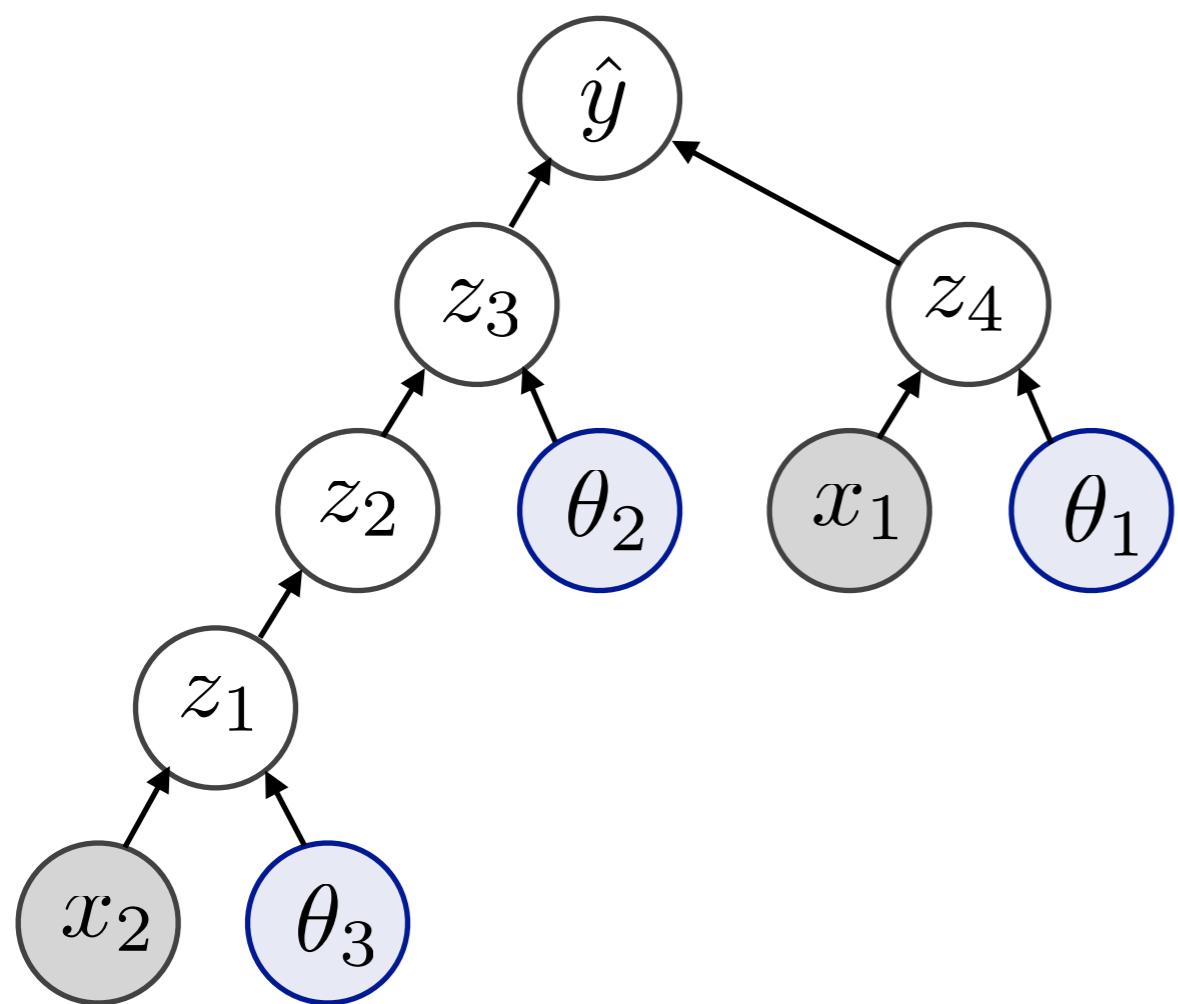
合成関数

$$\begin{aligned}z_1 &= \theta_3 + x_2 \\z_2 &= \cos(z_1) \\z_3 &= \theta_2 z_2 \\z_4 &= \theta_1 x_1\end{aligned}$$

# 計算グラフ：素演算の合成関数のグラフ表現

例  $\text{loss} = (\hat{y} - y)^2$  予測モデル (特に意味はないToy Example)

$$\begin{aligned}\hat{y} &= f(x_1, x_2, \theta_1, \theta_2, \theta_3) \\ &= \underbrace{\theta_1 x_1}_{z_4} - \theta_2 \cos(\underbrace{\theta_3 + x_2}_{z_1}) \\ &\quad \quad \quad \underbrace{\quad\quad\quad}_{z_2} \\ &\quad \quad \quad \quad\quad \underbrace{\quad\quad\quad\quad\quad}_{z_3}\end{aligned}$$

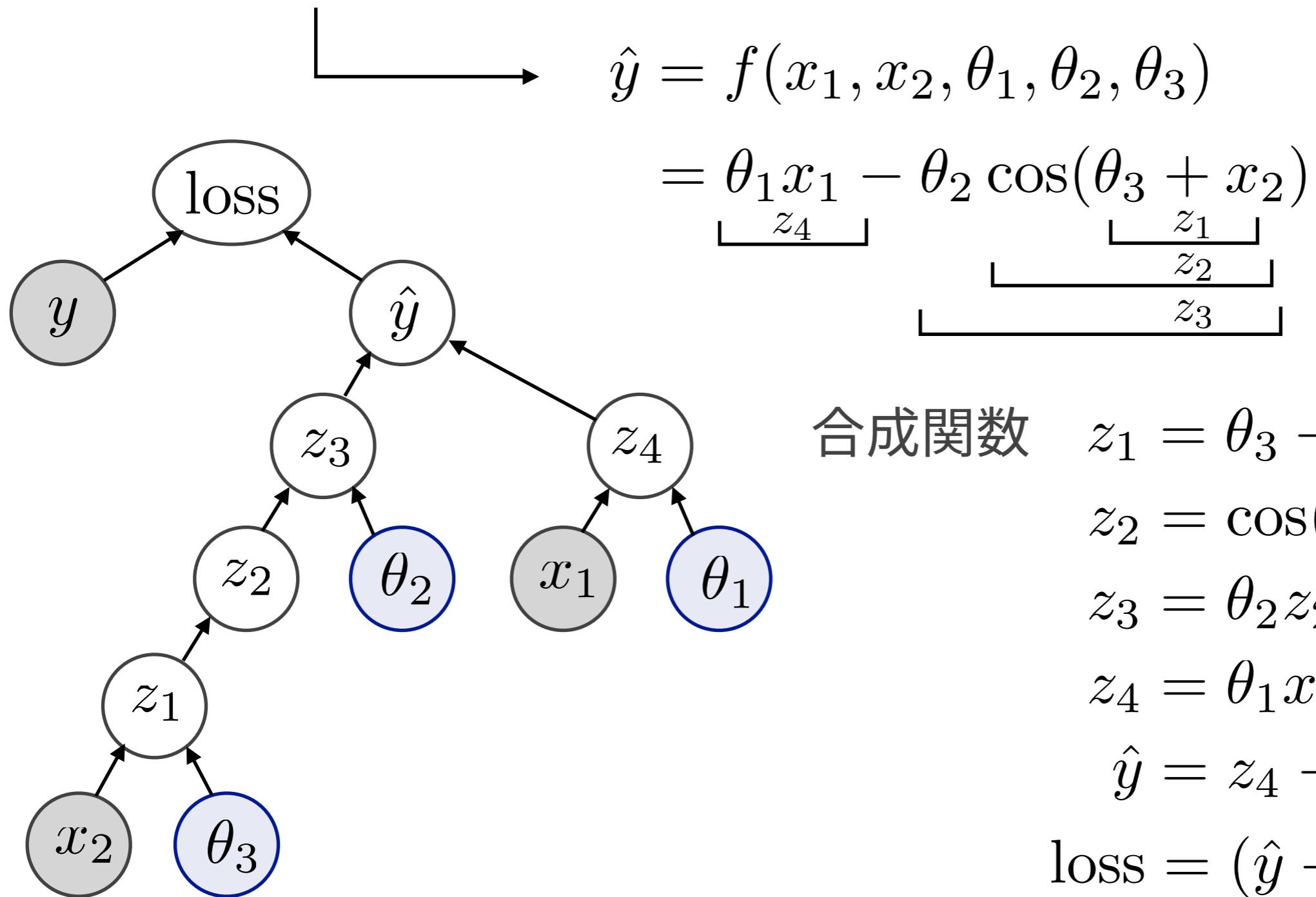


合成関数

$$\begin{aligned}z_1 &= \theta_3 + x_2 \\ z_2 &= \cos(z_1) \\ z_3 &= \theta_2 z_2 \\ z_4 &= \theta_1 x_1 \\ \hat{y} &= z_4 - z_3\end{aligned}$$

# 計算グラフ：素演算の合成関数のグラフ表現

例  $\text{loss} = (\hat{y} - y)^2$  予測モデル (特に意味はないToy Example)

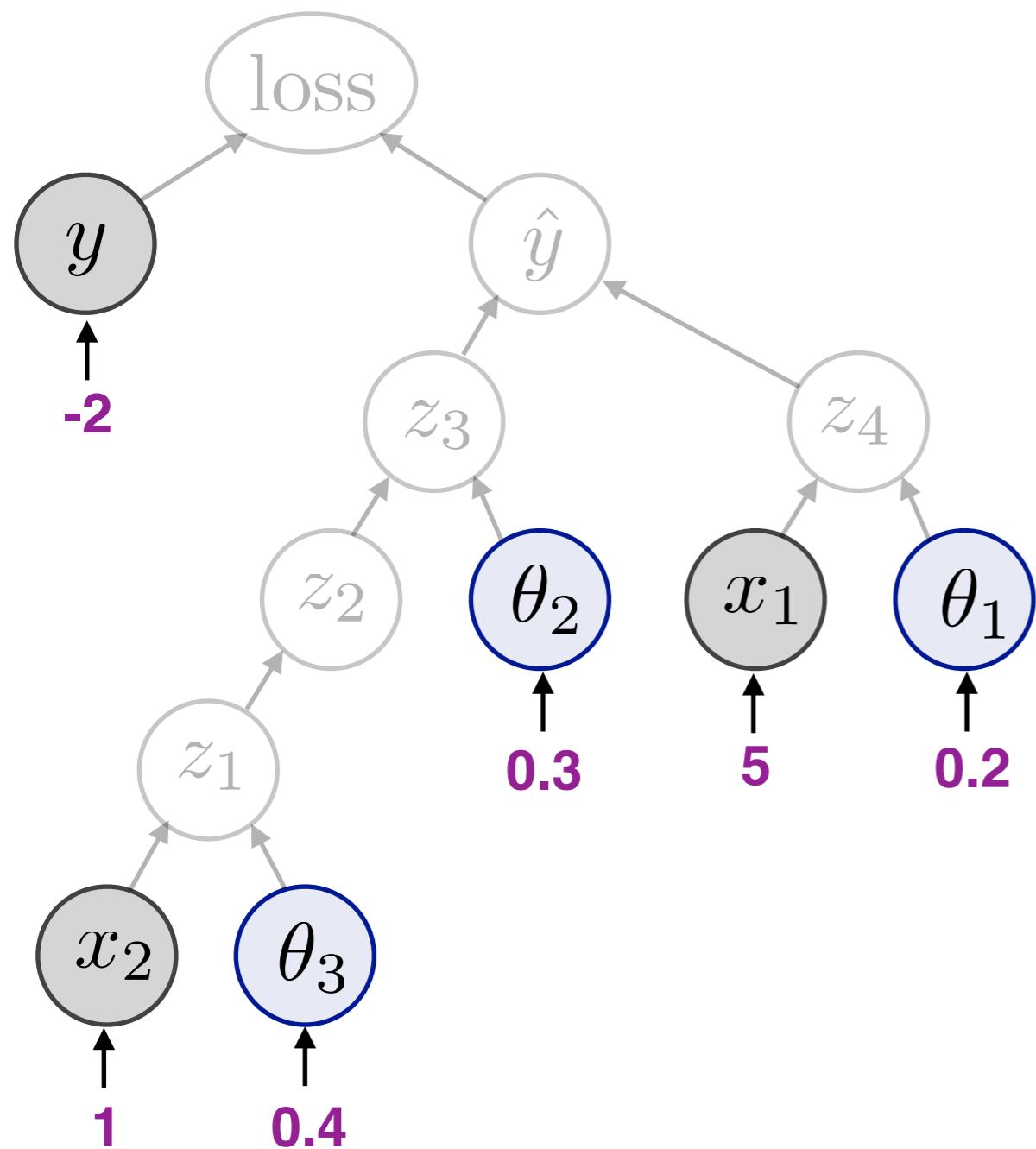


合成関数

$$\begin{aligned} z_1 &= \theta_3 + x_2 \\ z_2 &= \cos(z_1) \\ z_3 &= \theta_2 z_2 \\ z_4 &= \theta_1 x_1 \\ \hat{y} &= z_4 - z_3 \\ \text{loss} &= (\hat{y} - y)^2 \end{aligned}$$

# forward: 計算グラフの変数に値をセットしてみる

このとき loss や  $\hat{y}$  の値は？



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

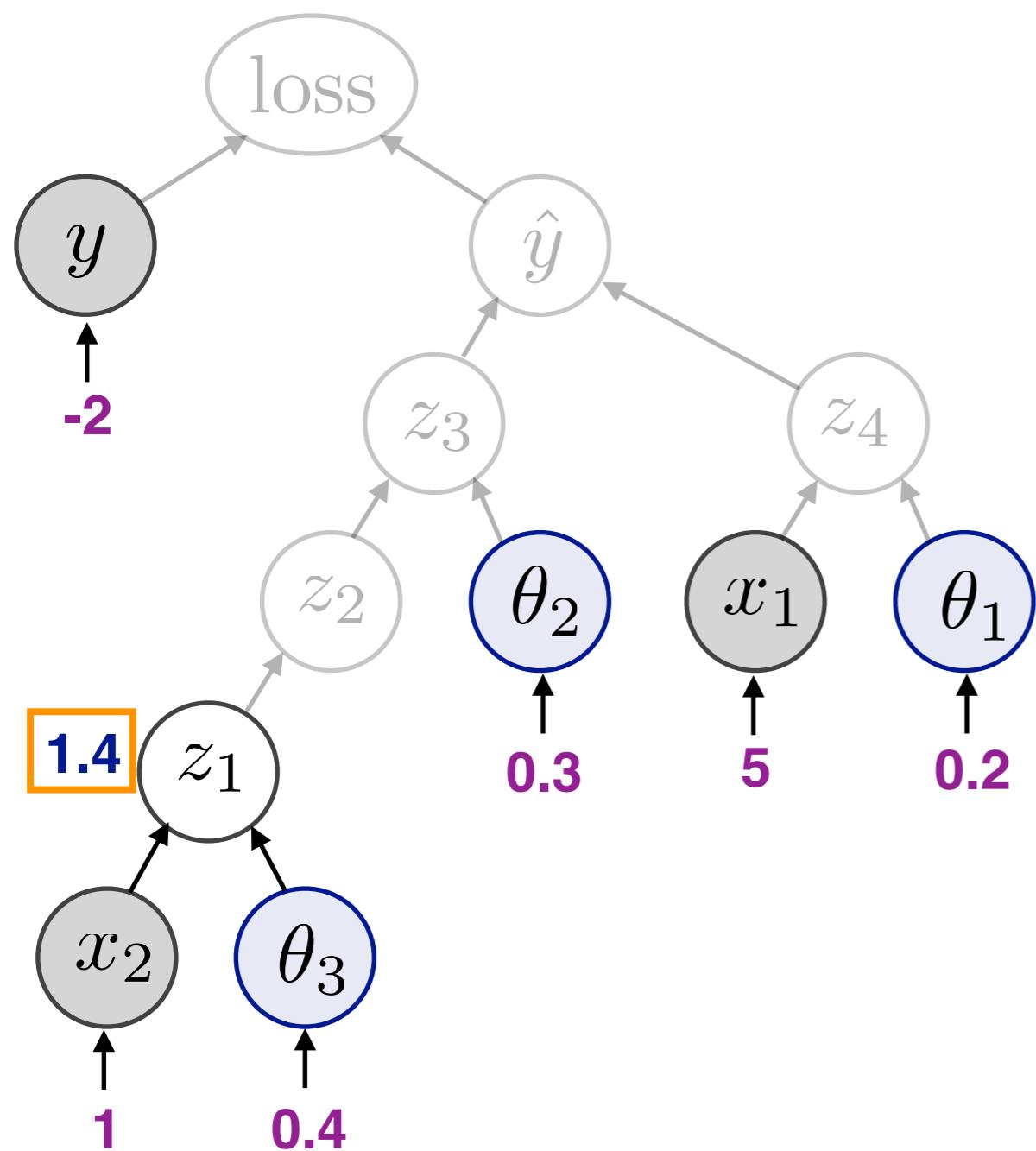
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forward: 計算グラフの変数に値をセットしてみる

このとき loss や  $\hat{y}$  の値は？



$$loss = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

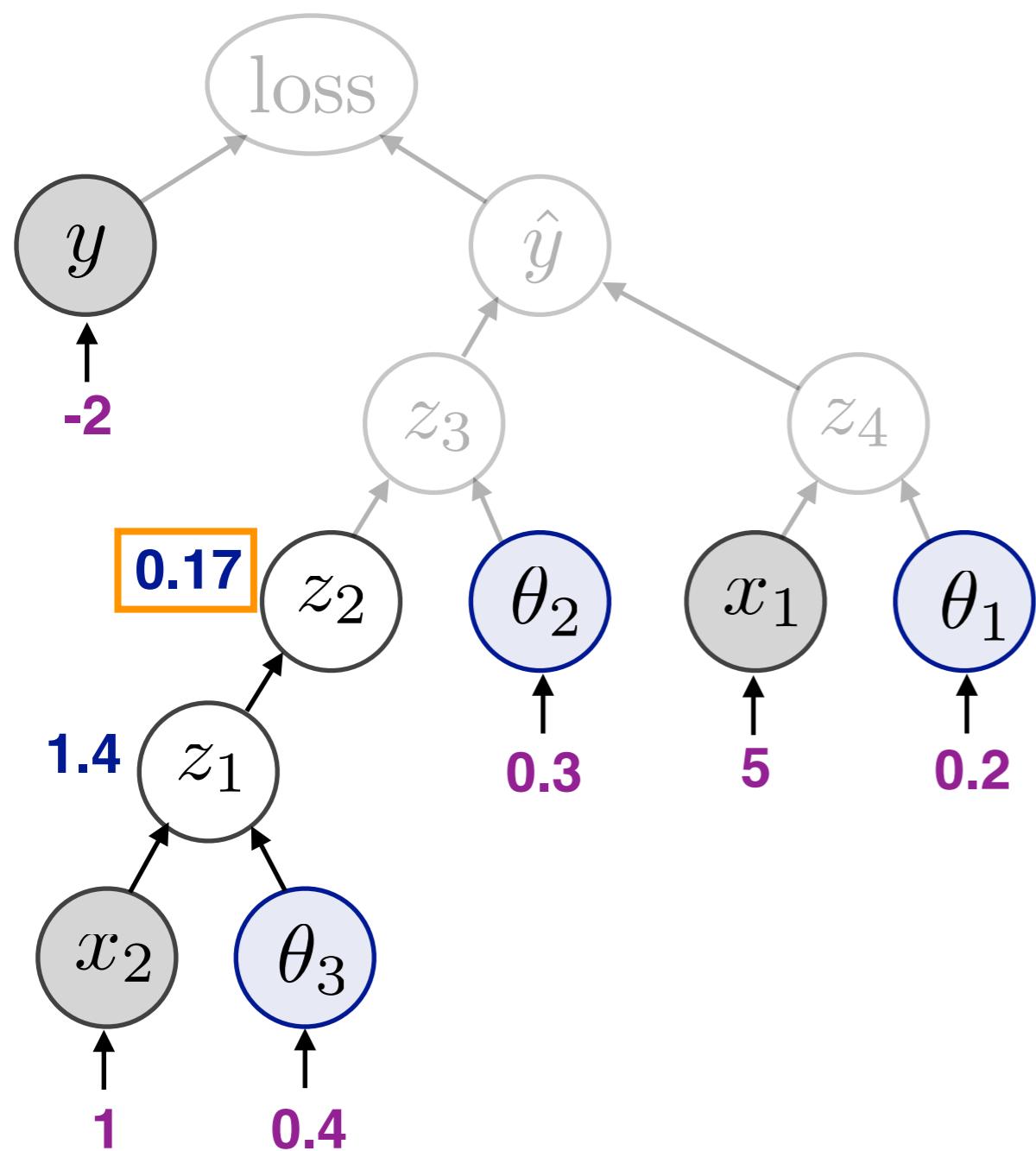
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forward: 計算グラフの変数に値をセットしてみる

このとき loss や  $\hat{y}$  の値は？



$$loss = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

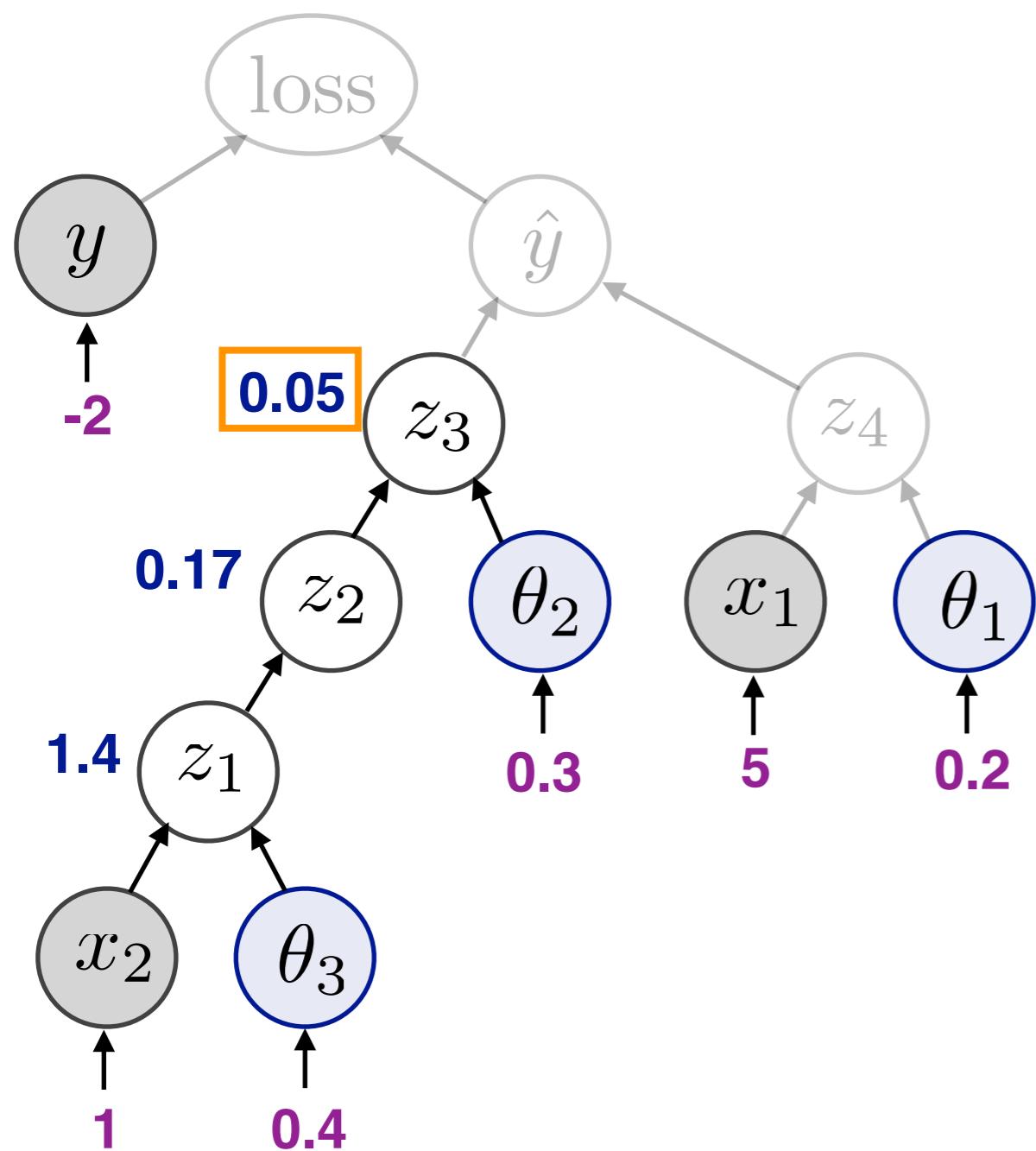
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forward: 計算グラフの変数に値をセットしてみる

このとき loss や  $\hat{y}$  の値は？



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

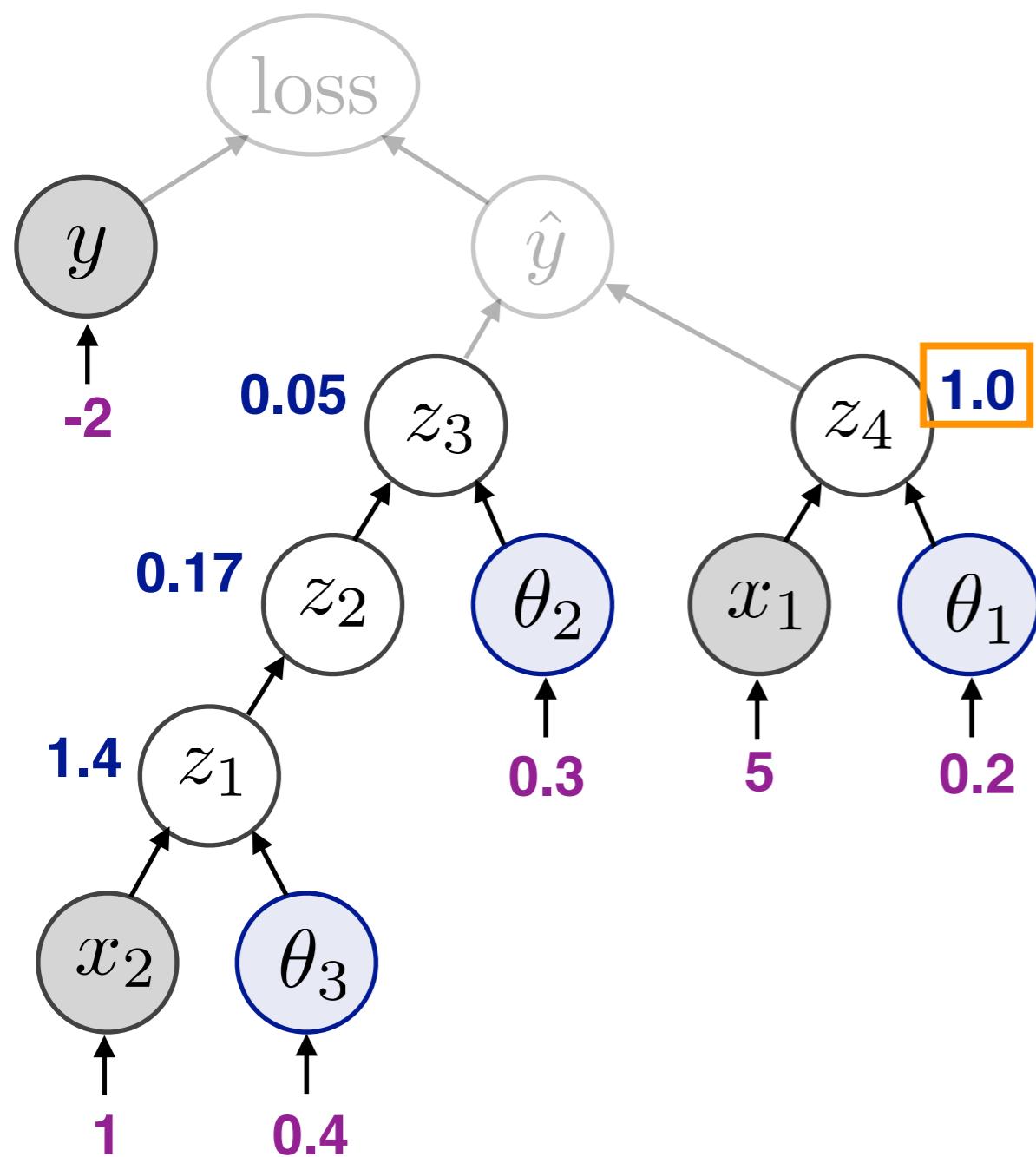
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forward: 計算グラフの変数に値をセットしてみる

このとき loss や  $\hat{y}$  の値は？



$$loss = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

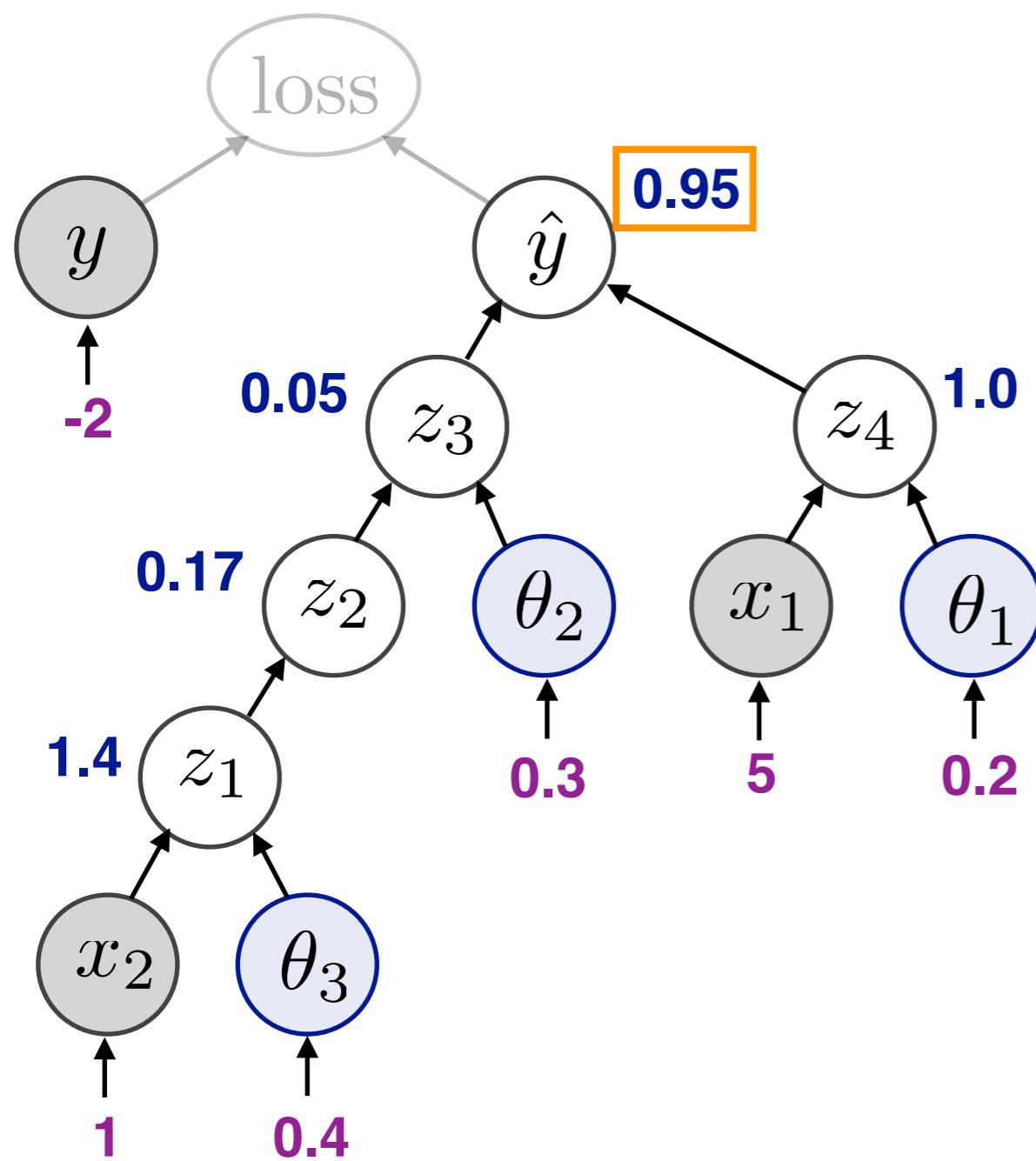
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forward: 計算グラフの変数に値をセットしてみる

このとき loss や  $\hat{y}$  の値は？



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

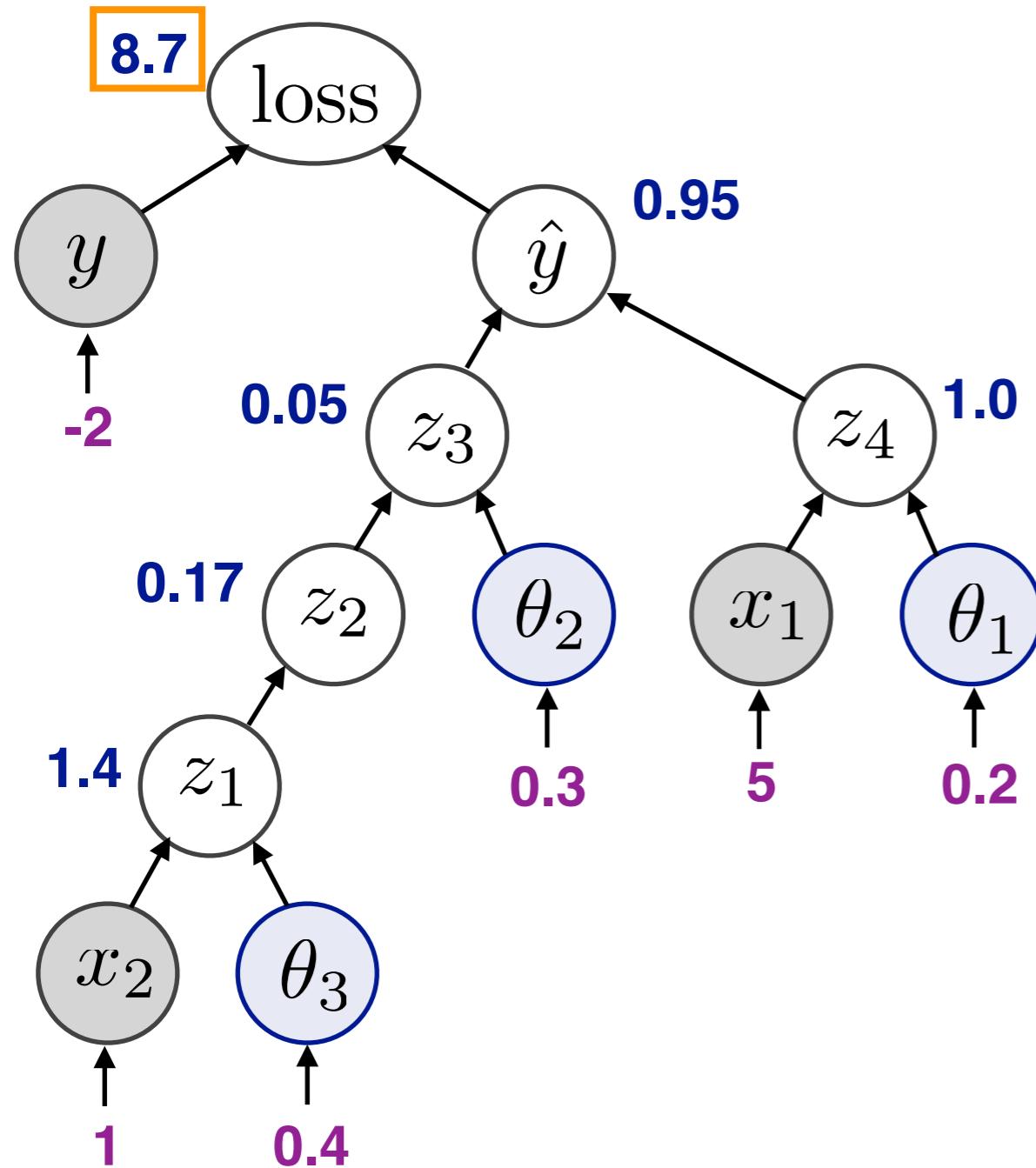
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forward: 計算グラフの変数に値をセットしてみる

このとき loss や  $\hat{y}$  の値は？



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

$$z_3 = \theta_2 z_2$$

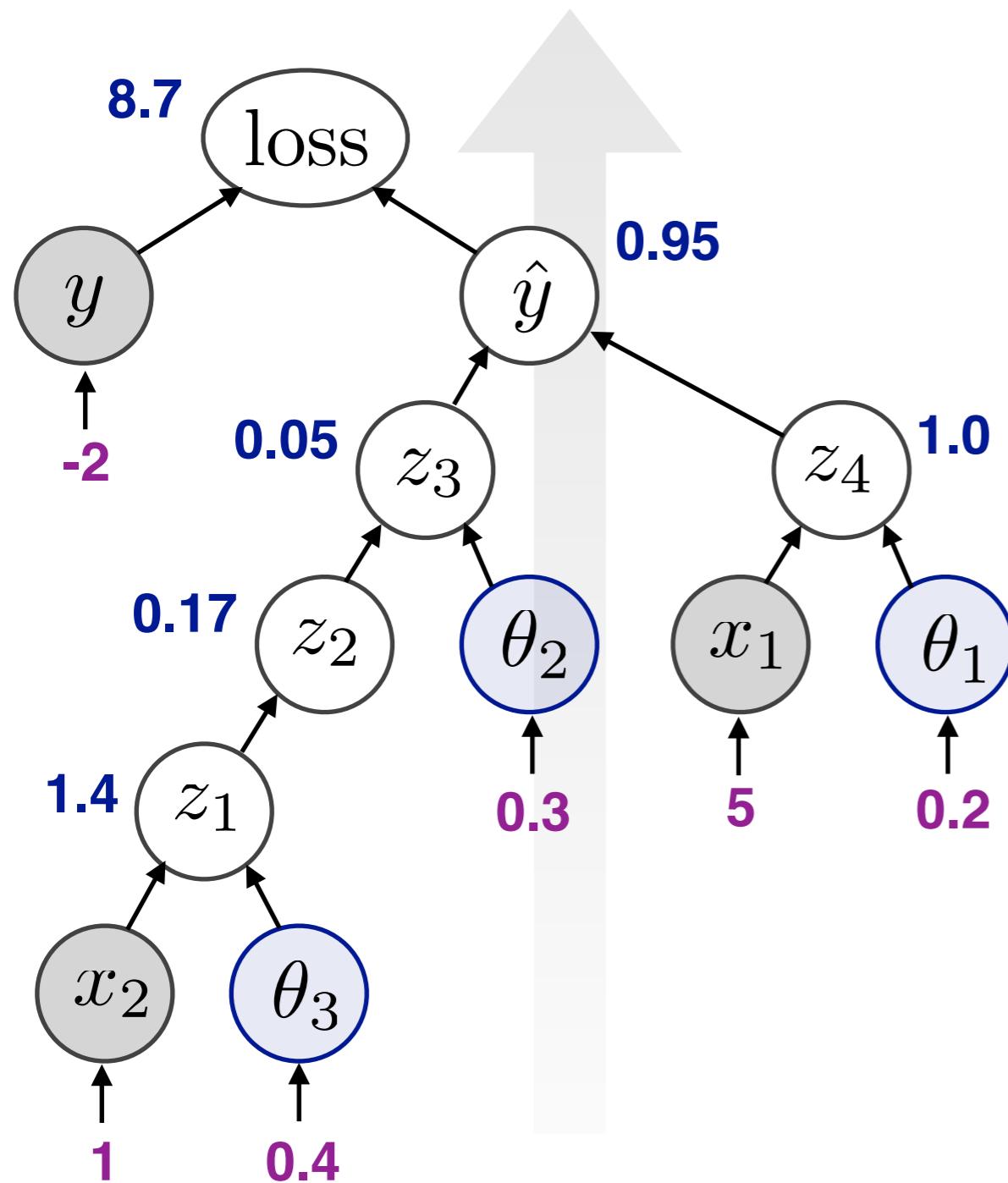
$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forward

---

このとき loss や  $\hat{y}$  の値は？



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

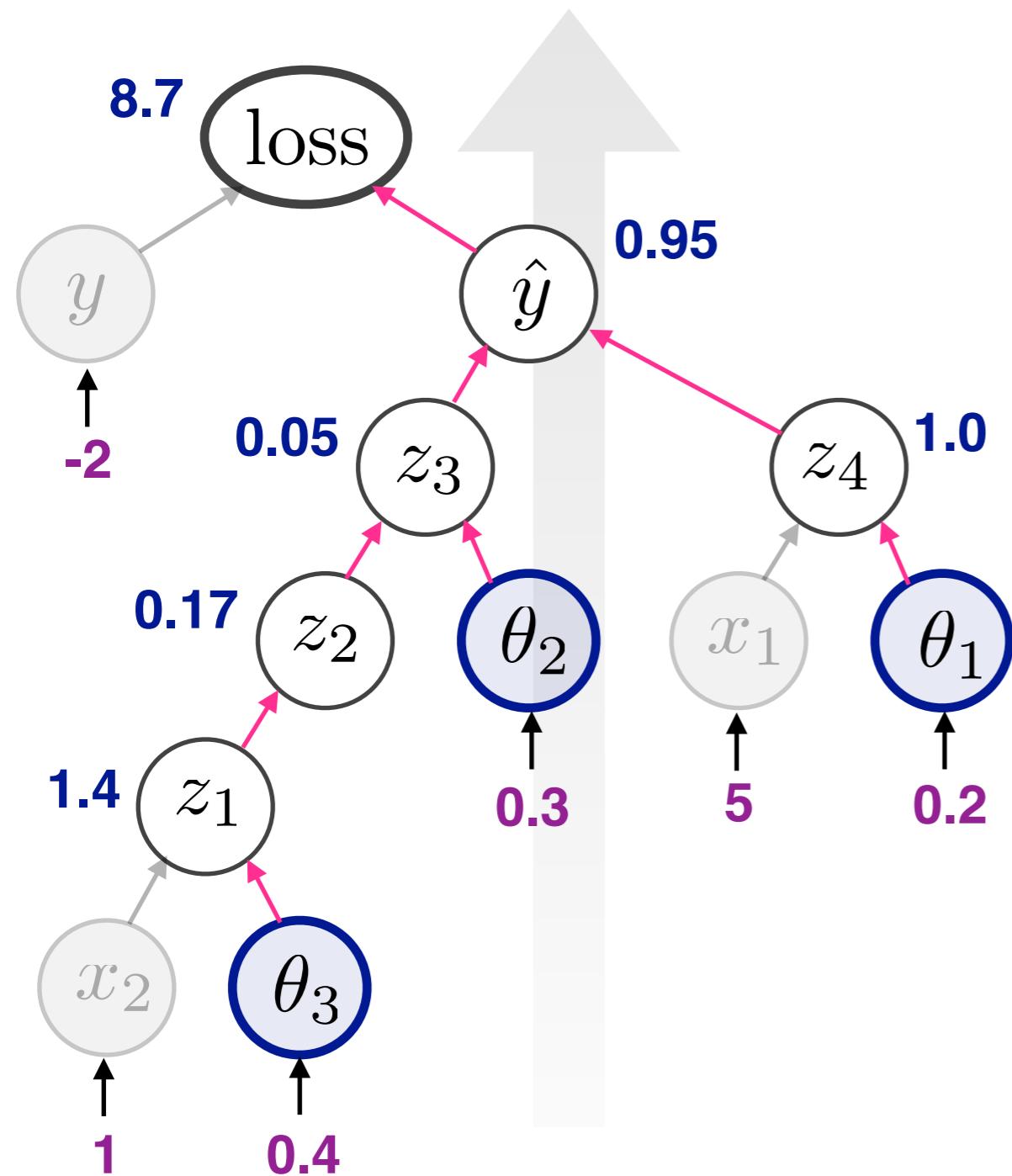
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forwardでついでにやること：偏微分値も計算

勾配を計算する変数はついでに各素演算の偏微分値も計算しておく



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

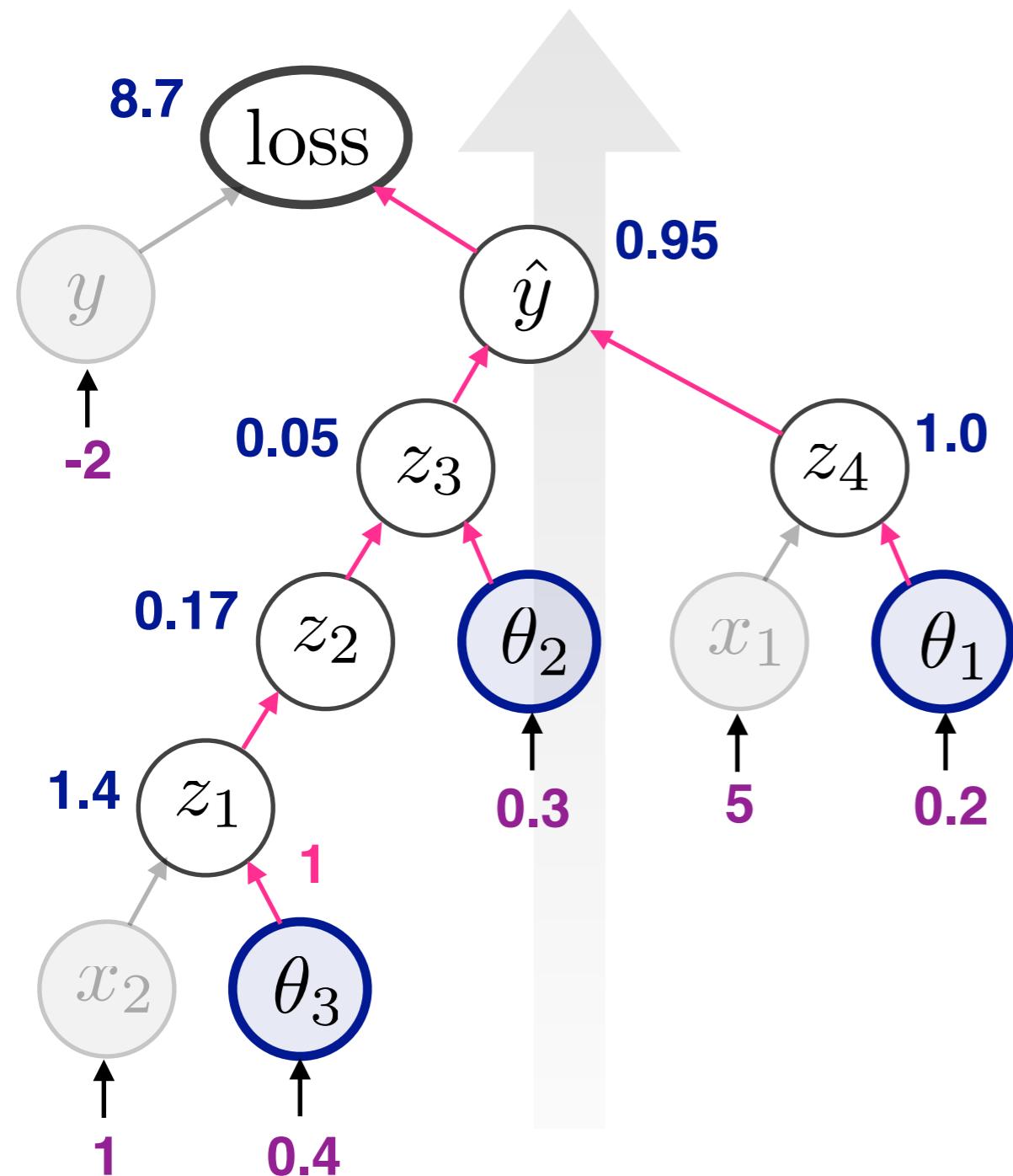
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2$$

# forwardでついでにやること：偏微分値も計算

勾配を計算する変数はついでに各素演算の偏微分値も計算しておく



$$loss = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

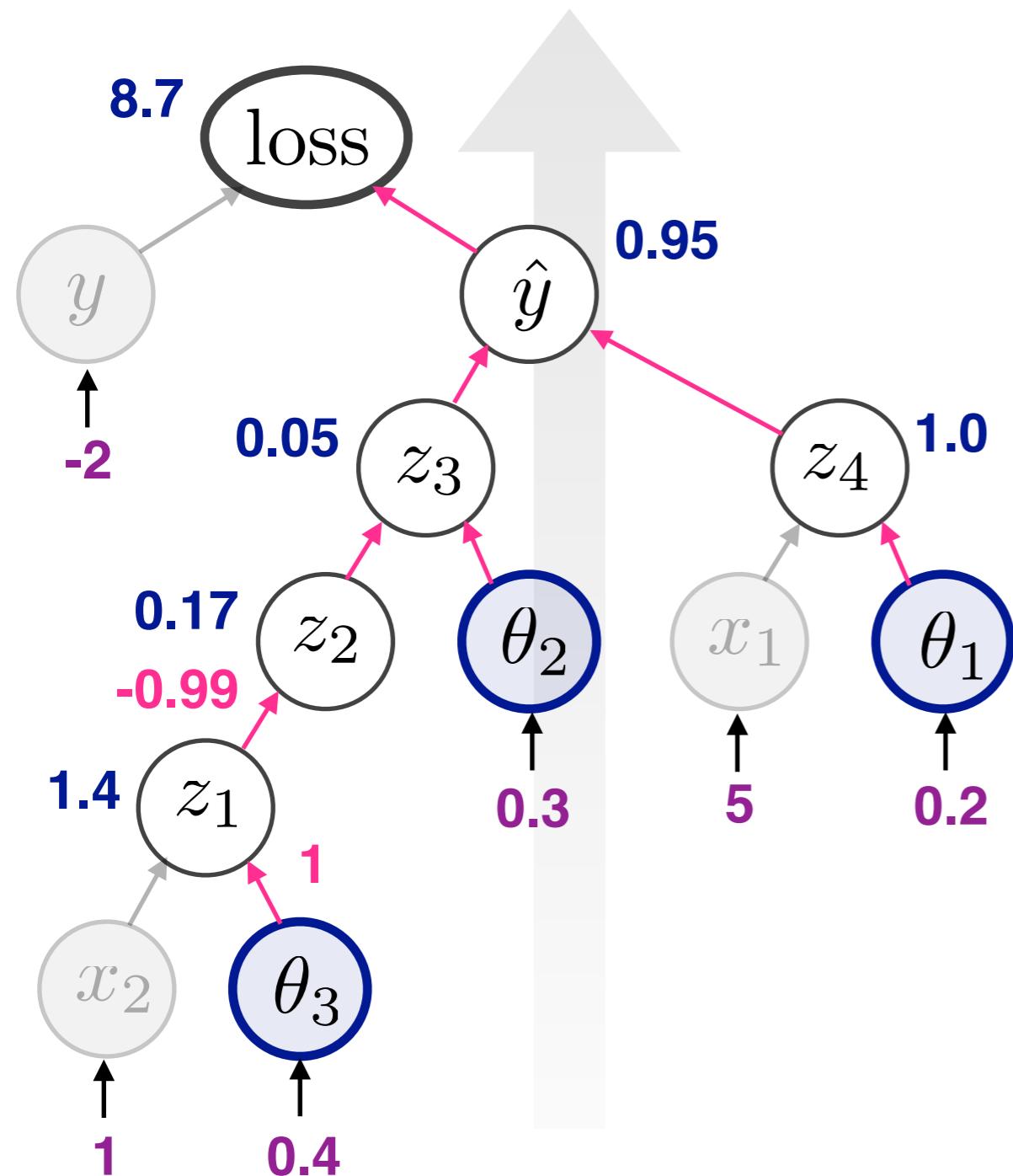
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

$$z_1 = \theta_3 + x_2 \quad \partial z_1 / \partial \theta_3 = 1$$

# forwardについてやること：偏微分値も計算

勾配を計算する変数はついでに各素演算の偏微分値も計算しておく



$$loss = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

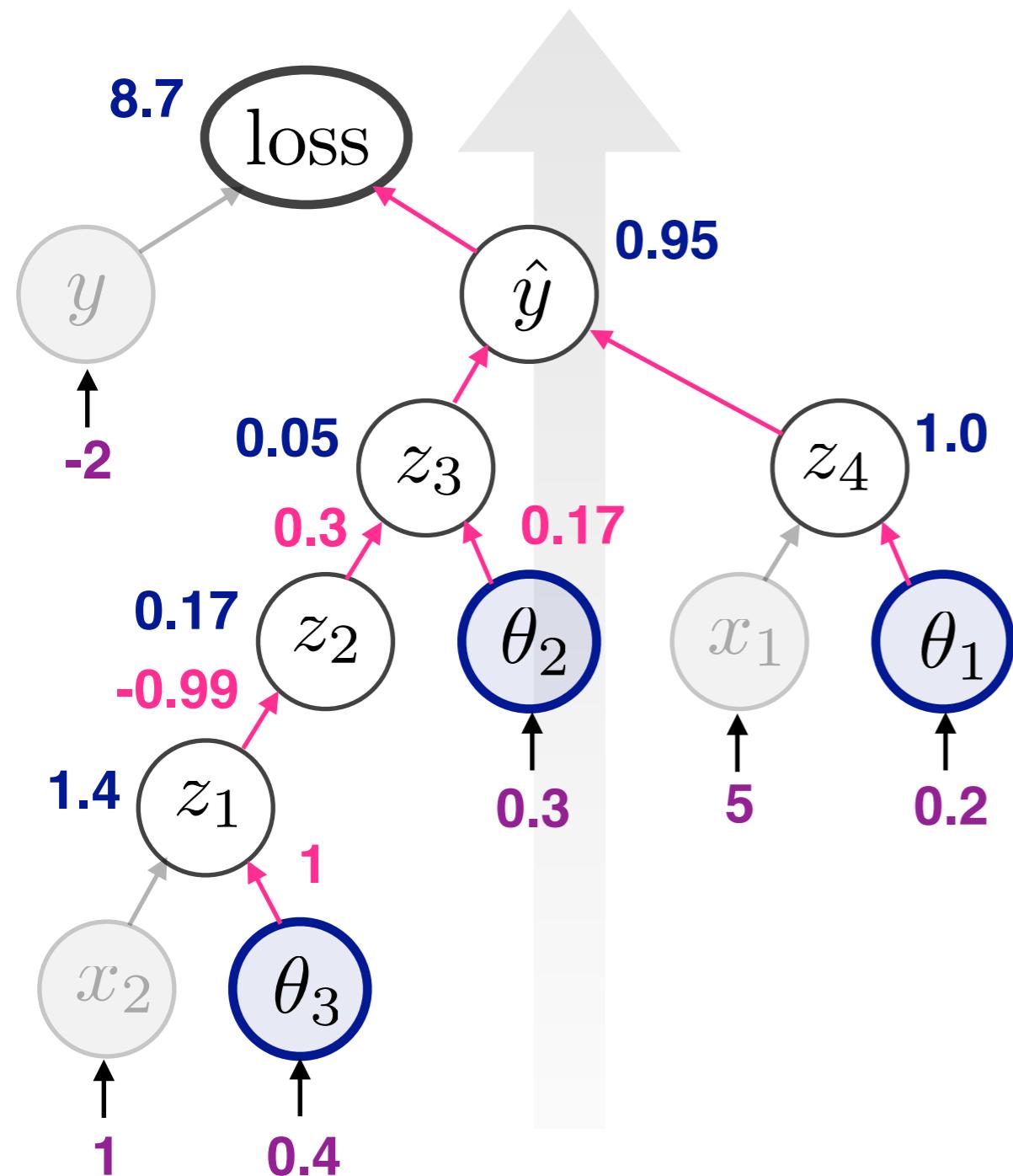
$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1) \quad \partial z_2 / \partial z_1 = -\sin(z_1)$$

$$z_1 = \theta_3 + x_2 \quad \partial z_1 / \partial \theta_3 = 1$$

# forwardについてやること：偏微分値も計算

勾配を計算する変数はついでに各素演算の偏微分値も計算しておく



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

$$z_3 = \theta_2 z_2$$

$$z_2 = \cos(z_1)$$

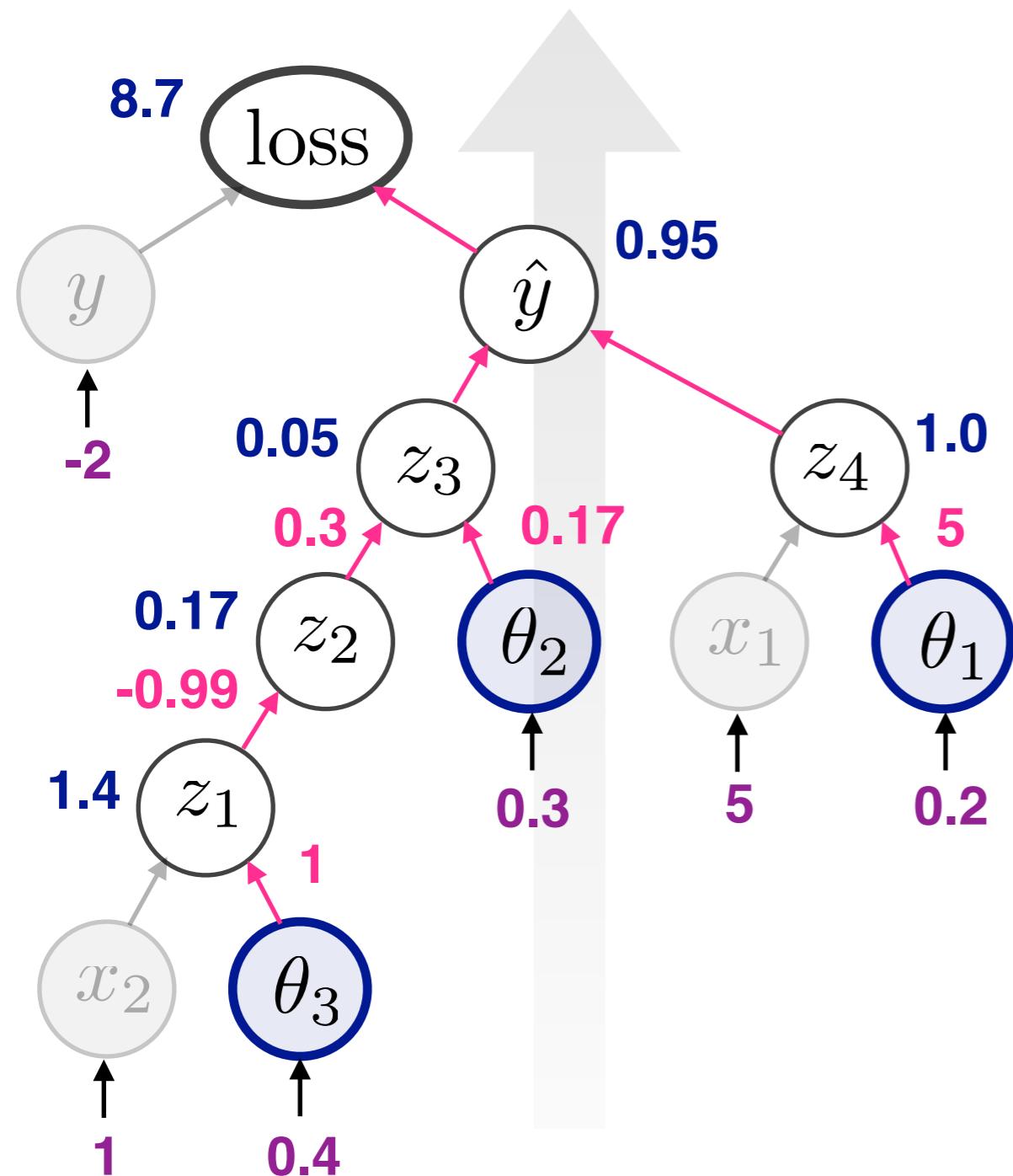
$$z_1 = \theta_3 + x_2 \quad \partial z_1 / \partial \theta_3 = 1$$

$$\begin{cases} \partial z_3 / \partial z_2 = \theta_2 \\ \partial z_3 / \partial \theta_2 = z_2 \end{cases}$$

$$\partial z_2 / \partial z_1 = -\sin(z_1)$$

# forwardについてやること：偏微分値も計算

勾配を計算する変数はついでに各素演算の偏微分値も計算しておく



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3$$

$$z_4 = \theta_1 x_1$$

$$\partial z_4 / \partial \theta_1 = x_1$$

$$z_3 = \theta_2 z_2$$

$$\begin{cases} \partial z_3 / \partial z_2 = \theta_2 \\ \partial z_3 / \partial \theta_2 = z_2 \end{cases}$$

$$z_2 = \cos(z_1)$$

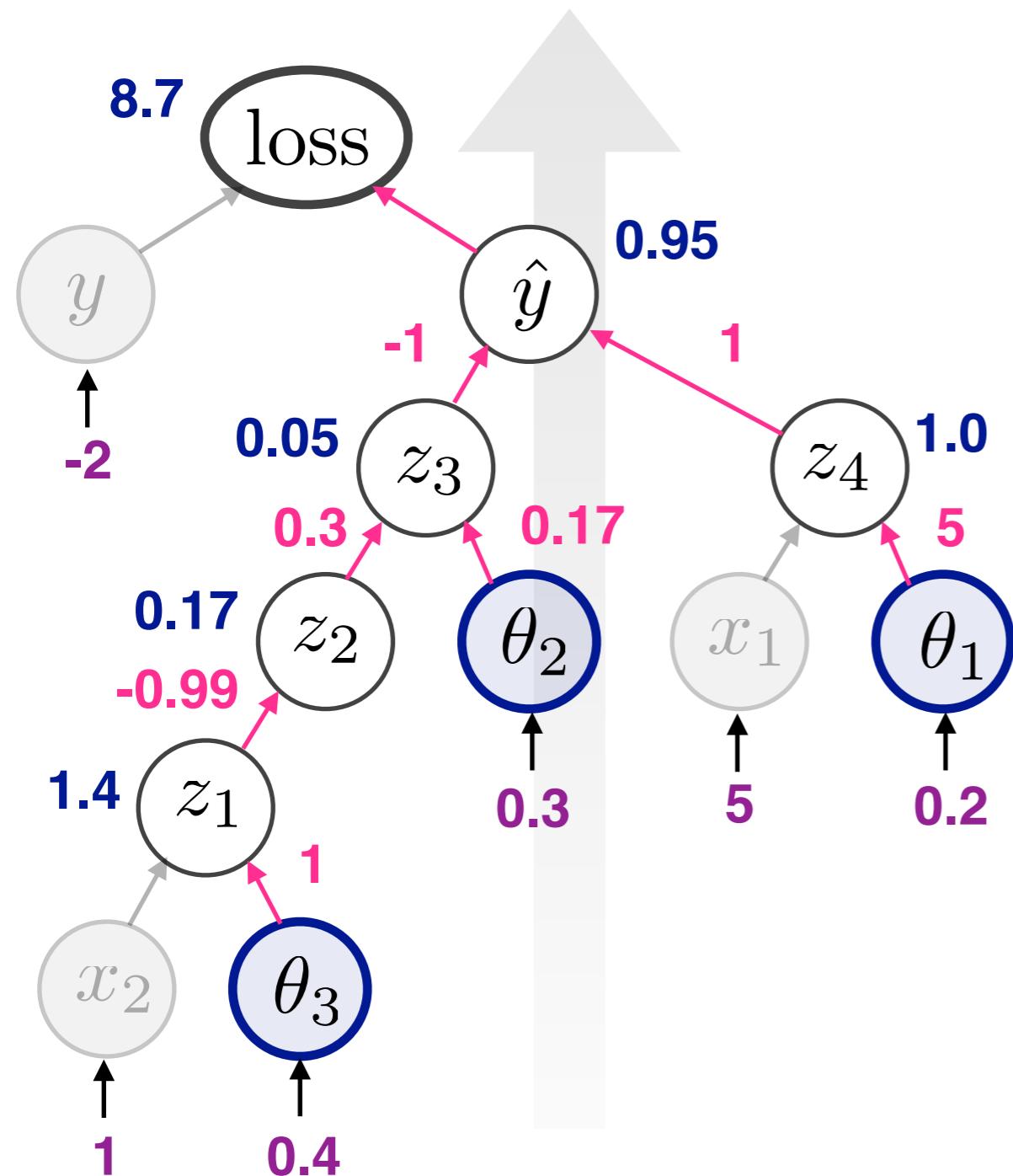
$$\partial z_2 / \partial z_1 = -\sin(z_1)$$

$$z_1 = \theta_3 + x_2$$

$$\partial z_1 / \partial \theta_3 = 1$$

# forwardでついでにやること：偏微分値も計算

勾配を計算する変数はついでに各素演算の偏微分値も計算しておく



$$\text{loss} = (\hat{y} - y)^2$$

$$\hat{y} = z_4 - z_3 \quad \begin{cases} \partial \hat{y} / \partial z_3 = -1 \\ \partial \hat{y} / \partial z_4 = 1 \end{cases}$$

$$z_4 = \theta_1 x_1 \quad \partial z_4 / \partial \theta_1 = x_1$$

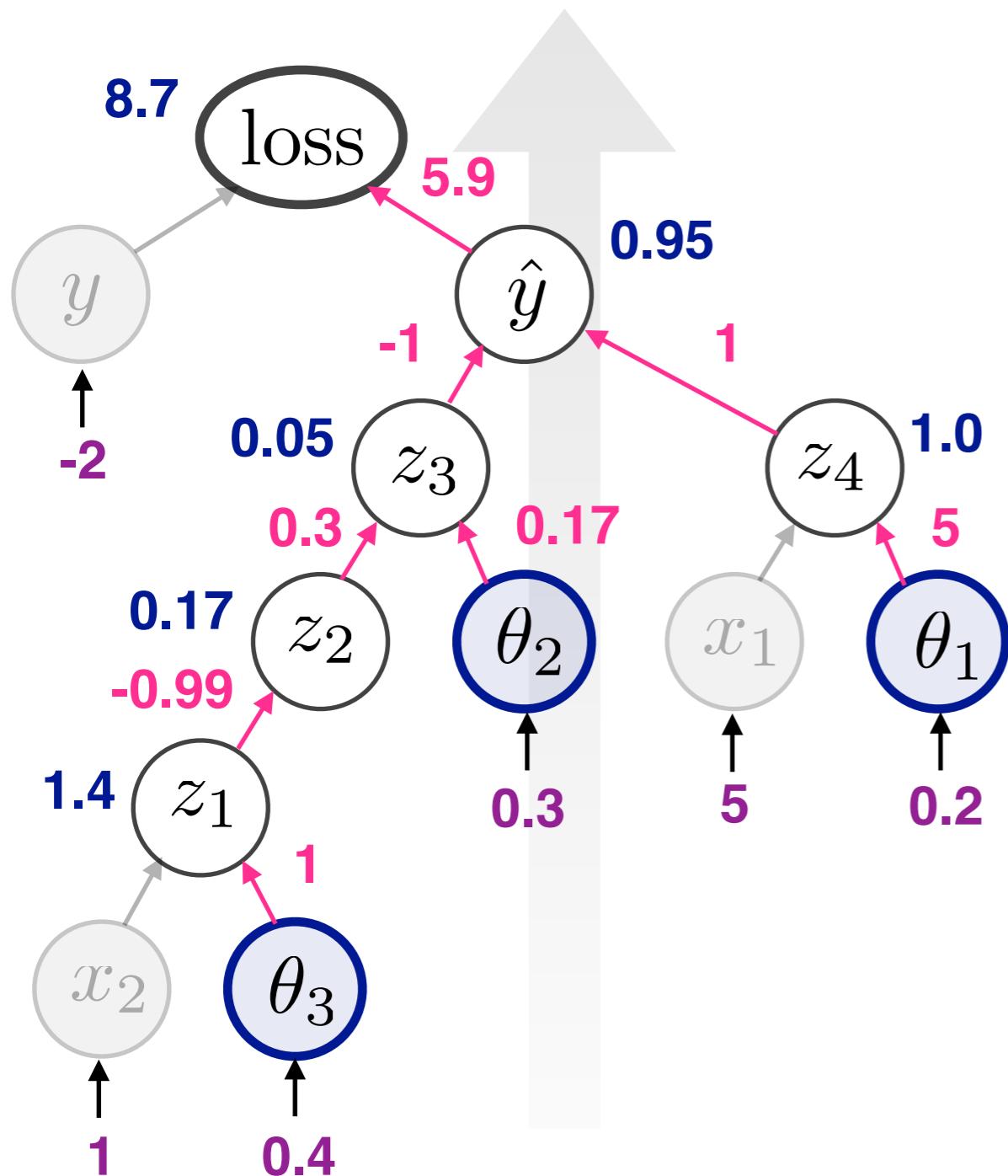
$$z_3 = \theta_2 z_2 \quad \begin{cases} \partial z_3 / \partial z_2 = \theta_2 \\ \partial z_3 / \partial \theta_2 = z_2 \end{cases}$$

$$z_2 = \cos(z_1) \quad \partial z_2 / \partial z_1 = -\sin(z_1)$$

$$z_1 = \theta_3 + x_2 \quad \partial z_1 / \partial \theta_3 = 1$$

# forwardでついでにやること：偏微分値も計算

勾配を計算する変数はついでに各素演算の偏微分値も計算しておく



$$loss = (\hat{y} - y)^2 \quad \partial loss / \partial \hat{y} = 2(\hat{y} - y)$$

$$\hat{y} = z_4 - z_3 \quad \begin{cases} \partial \hat{y} / \partial z_3 = -1 \\ \partial \hat{y} / \partial z_4 = 1 \end{cases}$$

$$z_4 = \theta_1 x_1 \quad \partial z_4 / \partial \theta_1 = x_1$$

$$z_3 = \theta_2 z_2 \quad \begin{cases} \partial z_3 / \partial z_2 = \theta_2 \\ \partial z_3 / \partial \theta_2 = z_2 \end{cases}$$

$$z_2 = \cos(z_1) \quad \partial z_2 / \partial z_1 = -\sin(z_1)$$

$$z_1 = \theta_3 + x_2 \quad \partial z_1 / \partial \theta_3 = 1$$

# forwardについてやること：偏微分値も計算

合成に使える素演算は予め決めてそれぞれの微分公式も持つておく

## 変数の合成に使える素演算の例

- 加減乗除

`torch.add`, `torch.sub`, `torch.mul`, `torch.div`, ..

- 三角関数

`torch.sin`, `torch.cos`, `torch.tan`, `torch.asin`, ..

- 指数・対数・べき

`torch.pow`, `torch.log`, `torch.square`, `torch.sqrt`, ..

- 行列演算

`torch.matmul`, `torch.dot`, `torch.inverse`, `torch.det`, ..

- 操作

`torch.sum`, `torch.mean`, `torch.max`, `torch.argmin`, ..

$$\text{loss} = (\hat{y} - y)^2 \quad \partial \text{loss} / \partial \hat{y} = 2(\hat{y} - y)$$

$$\hat{y} = z_4 - z_3 \quad \begin{cases} \partial \hat{y} / \partial z_3 = -1 \\ \partial \hat{y} / \partial z_4 = 1 \end{cases}$$

$$z_4 = \theta_1 x_1 \quad \partial z_4 / \partial \theta_1 = x_1$$

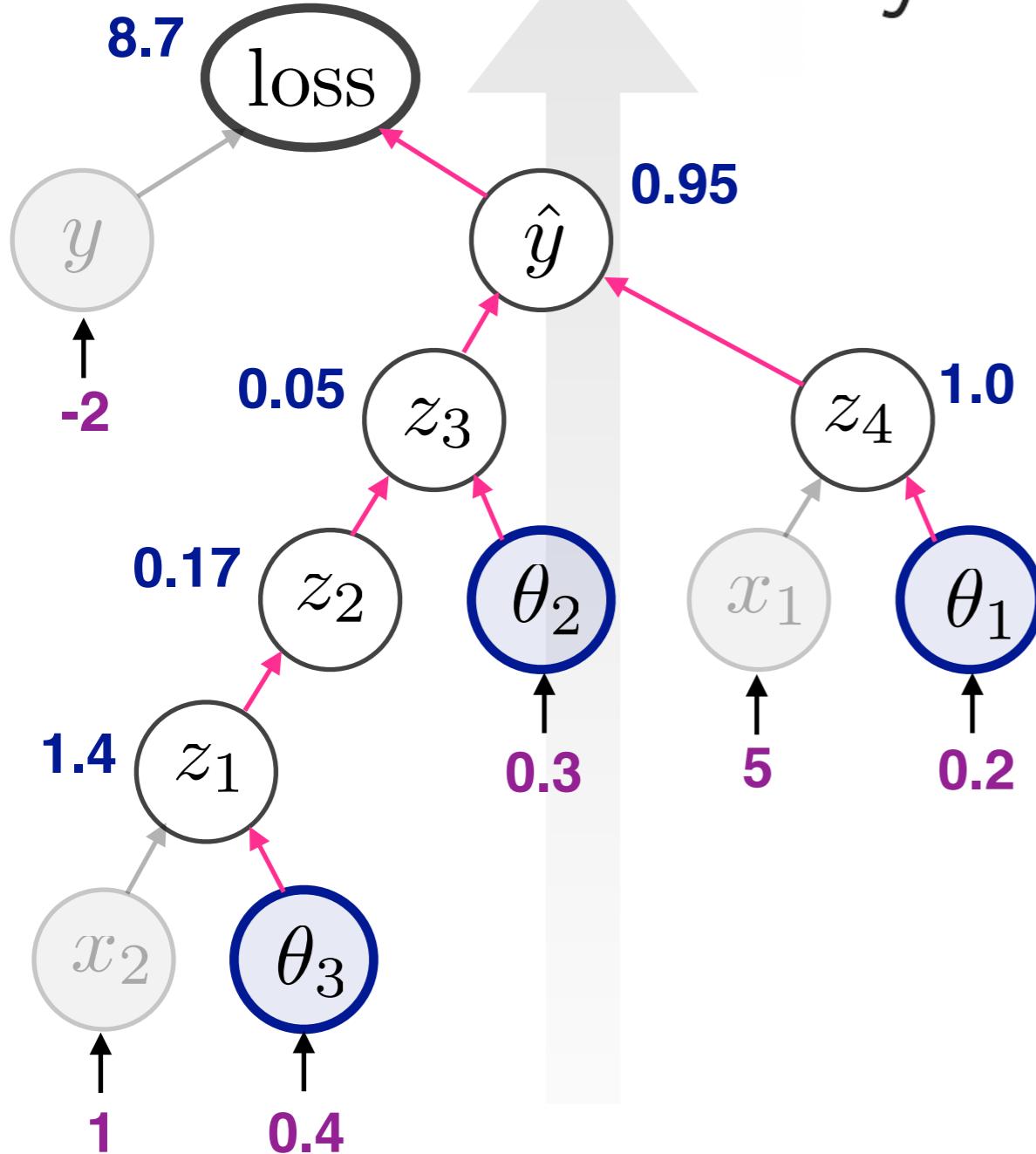
$$z_3 = \theta_2 z_2 \quad \begin{cases} \partial z_3 / \partial z_2 = \theta_2 \\ \partial z_3 / \partial \theta_2 = z_2 \end{cases}$$

$$z_2 = \cos(z_1) \quad \partial z_2 / \partial z_1 = -\sin(z_1)$$

$$z_1 = \theta_3 + x_2 \quad \partial z_1 / \partial \theta_3 = 1$$

# forward

PyTorch



```
import torch
```

```
y = torch.tensor(-2)
x1 = torch.tensor(5)
x2 = torch.tensor(1)
theta1 = torch.tensor(0.2, requires_grad=True)
theta2 = torch.tensor(0.3, requires_grad=True)
theta3 = torch.tensor(0.4, requires_grad=True)
```

```
z1 = theta3 + x2
z2 = torch.cos(z1)
z3 = theta2 * z2
z4 = theta1 * x1
y_hat = z4 - z3

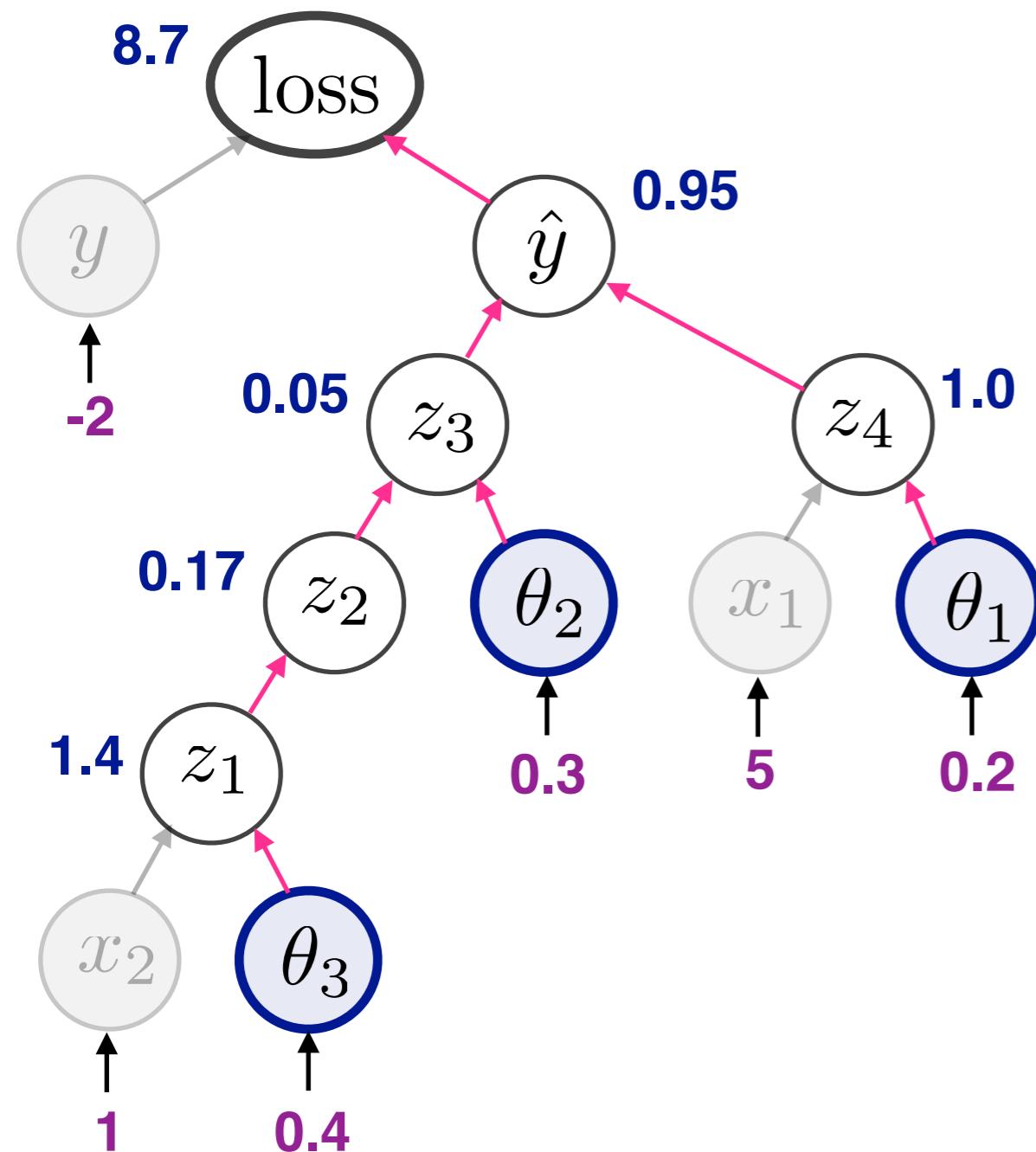
loss = (y_hat-y)**2
```

```
z1, z2, z3, z4, y_hat, loss
```

```
(tensor(1.4000, grad_fn=<AddBackward0>),
 tensor(0.1700, grad_fn=<CosBackward>),
 tensor(0.0510, grad_fn=<MulBackward0>),
 tensor(1., grad_fn=<MulBackward0>),
 tensor(0.9490, grad_fn=<SubBackward0>),
 tensor(8.6967, grad_fn=<PowBackward0>))
```

# 自動微分(リバースモード) "Back Propagation"

本当に計算したいのは勾配を計算する変数に関する偏微分



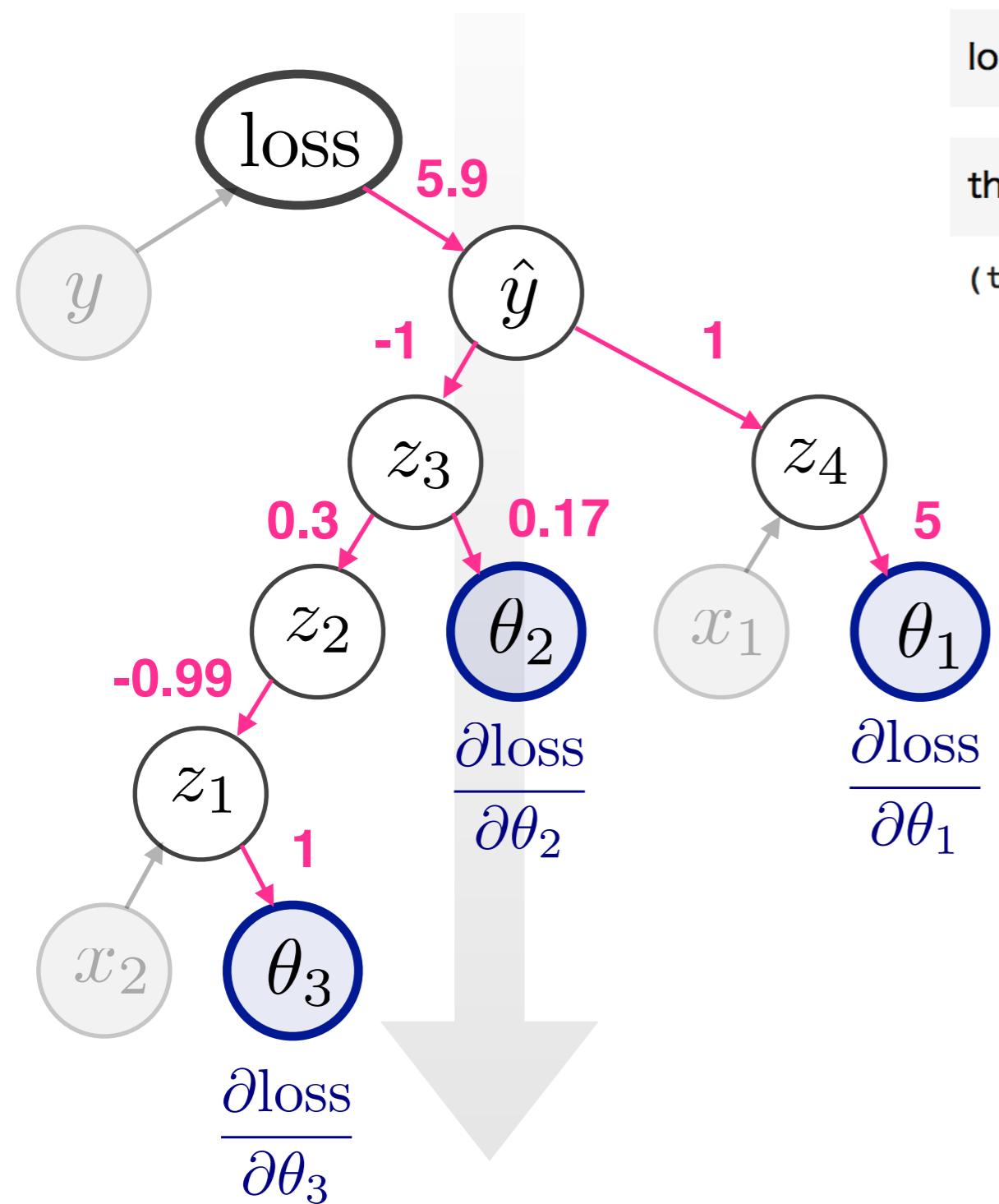
勾配ベクトル = 
$$\begin{bmatrix} \frac{\partial \text{loss}}{\partial \theta_1} \\ \frac{\partial \text{loss}}{\partial \theta_2} \\ \frac{\partial \text{loss}}{\partial \theta_3} \end{bmatrix}$$

$\theta_1, \theta_2, \theta_3$  を少し動かしたときの loss の変化量 (勾配法が使いたい)

?

# backward

合成関数の微分の連鎖律より、計算グラフを使って簡単に計算できる！



loss.backward()

theta1.grad, theta2.grad, theta3.grad

(tensor(29.4901), tensor(-1.0025), tensor(1.7437))

$$\frac{\partial loss}{\partial \theta_1} = \frac{\partial loss}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_4} \times \frac{\partial z_4}{\partial \theta_1}$$

$$5.9 \times 1 \times 5 = 29.5$$

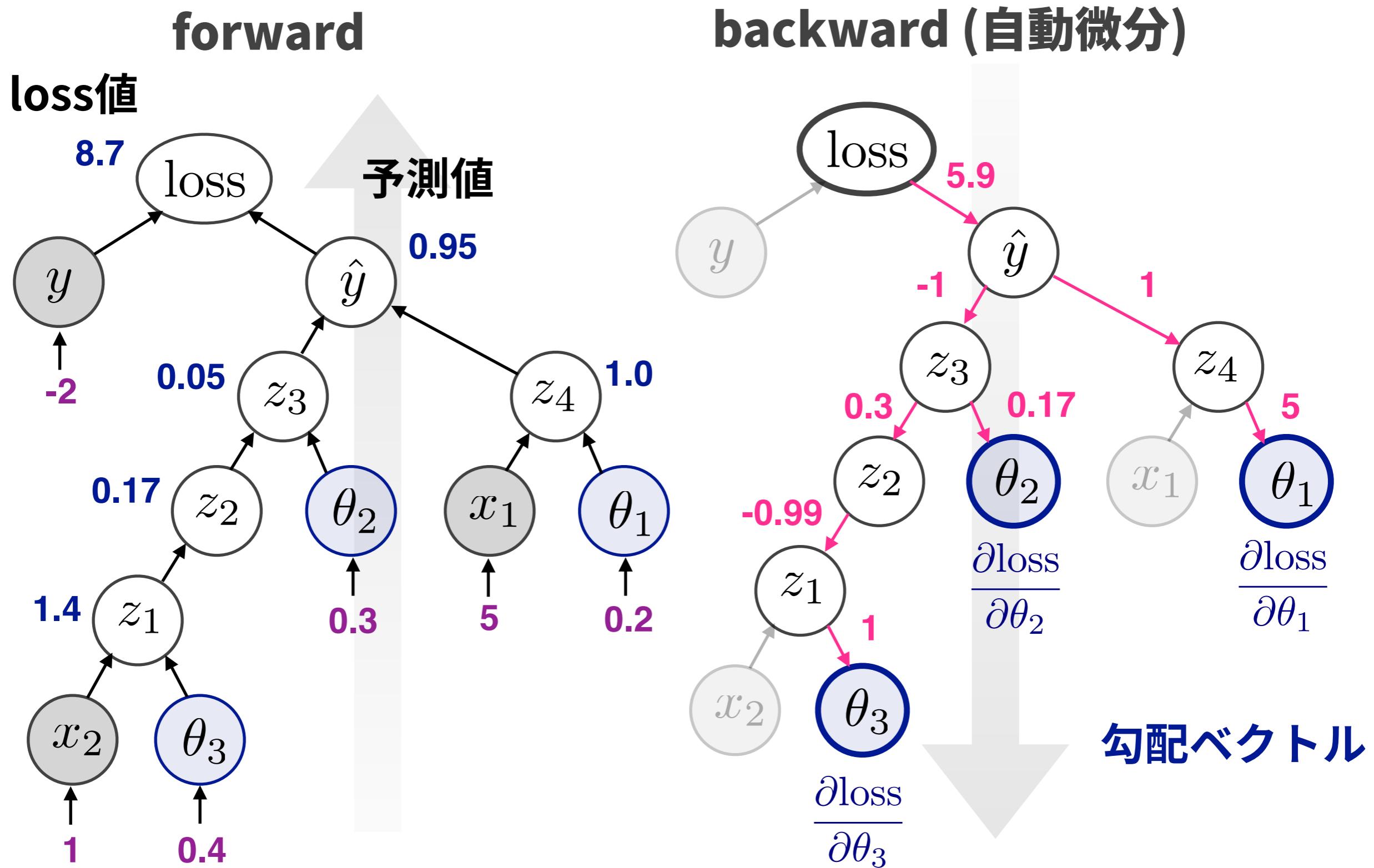
$$\frac{\partial loss}{\partial \theta_2} = \frac{\partial loss}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_3} \times \frac{\partial z_3}{\partial \theta_2}$$

$$5.9 \times -1 \times 0.17 = -1.00$$

$$\frac{\partial loss}{\partial \theta_3} = \frac{\partial loss}{\partial \hat{y}} \times \frac{\partial \hat{y}}{\partial z_3} \times \frac{\partial z_3}{\partial z_2} \times \frac{\partial z_2}{\partial z_1} \times \frac{\partial z_1}{\partial \theta_3}$$

$$5.9 \times -1 \times 0.3 \times -0.99 \times 1 = 1.75$$

# プログラムとは究極的には素演算の何らかの合成関数！



# 深層学習のココロ："微分可能な"プログラミング

プログラムの不確定な箇所はパラメタにしておき後で機械学習

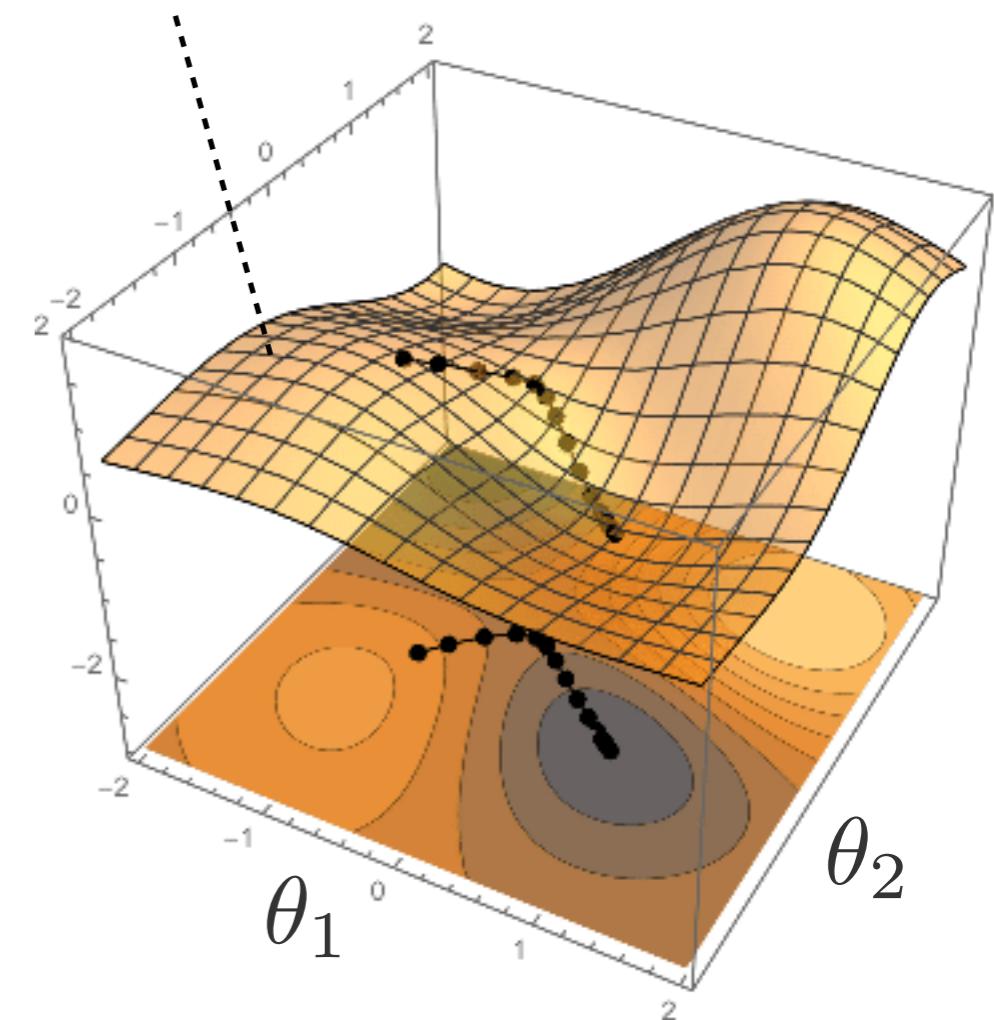
どんなプログラムも素演算の合成関数(計算グラフ)になってさえ  
いれば自動微分+勾配降下法でパラメタを最適にできる！

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix} \leftarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_p \end{bmatrix} - \eta \times \begin{bmatrix} \partial \text{loss}(\theta) / \partial \theta_1 \\ \partial \text{loss}(\theta) / \partial \theta_2 \\ \vdots \\ \partial \text{loss}(\theta) / \partial \theta_p \end{bmatrix}$$

勾配ベクトル

自動微分で計算

$\text{loss}(\theta_1, \theta_2)$



# 本日の内容

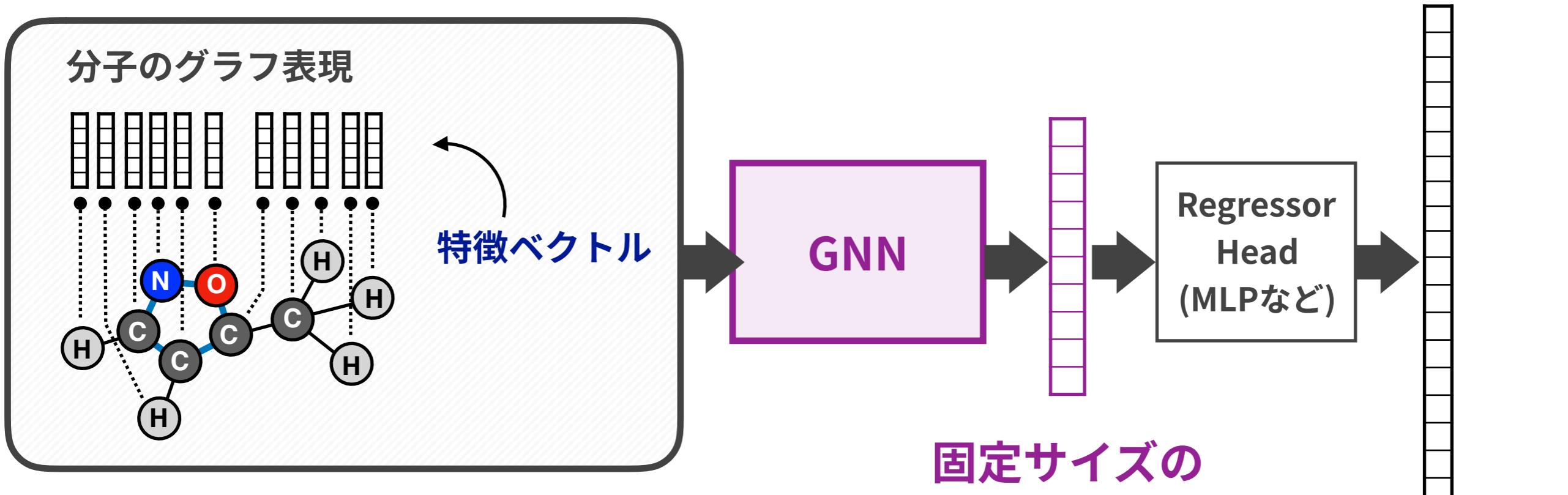
---

GNN(Graph Neural Networks)でグラフ表現された分子を  
入力にして機械学習するやり方をのぞいてみませんか？

- 機械学習とは？
- 分子のグラフ表現
- 準備：深層学習のココロ (forwardとbackward)
- Graph Neural Networksとは？
- 自分でやってみるための情報
- おまけ：  
最近の話 (計算限界、分子生成、事前学習、...)

# GNN = 下記の変換を何らかの合成関数でデザインしたもの

可変サイズの分子グラフから「固定サイズのベクトル表現」への変換を学習



**mol.x** 頂点の特徴ベクトル (11次元)

**mol.pos** 頂点の位置ベクトル (3次元)

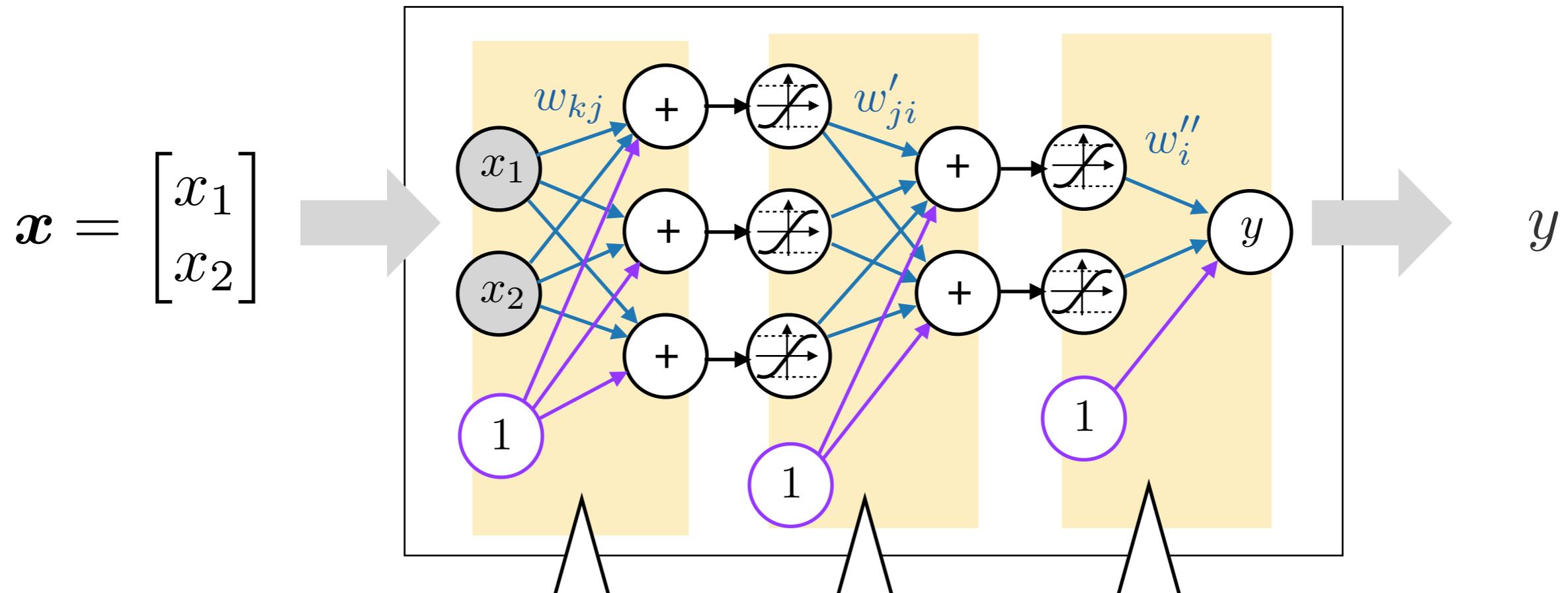
**mol.edge\_attr** 辺の特徴ベクトル (4次元)

**mol.edge\_index** 辺と結合原子ペアの対応

**mol.y**  
DFT計算値  
(19次元)

# 基本道具：MLP (多層パーセプトロン)

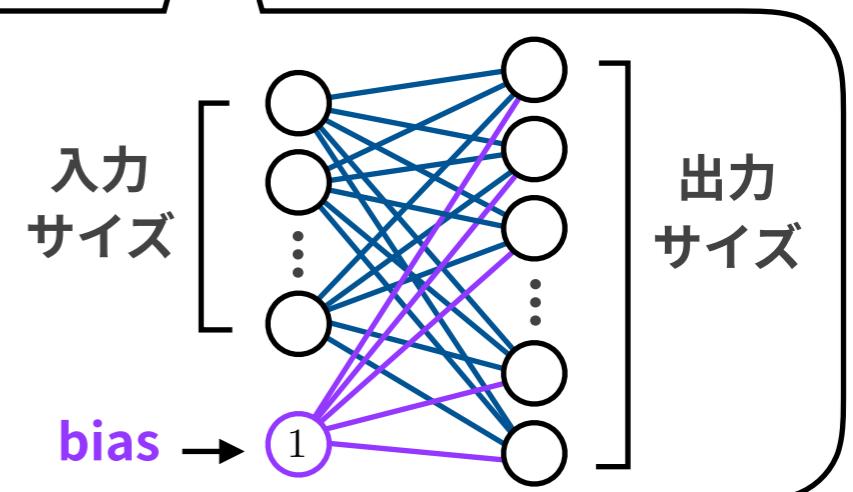
```
import torch.nn as nn  
nn.Sequential(nn.Linear(2, 3), nn.ReLU(), nn.Linear(3, 2), nn.ReLU(), nn.Linear(2, 1))
```



## torch.nn.Linear

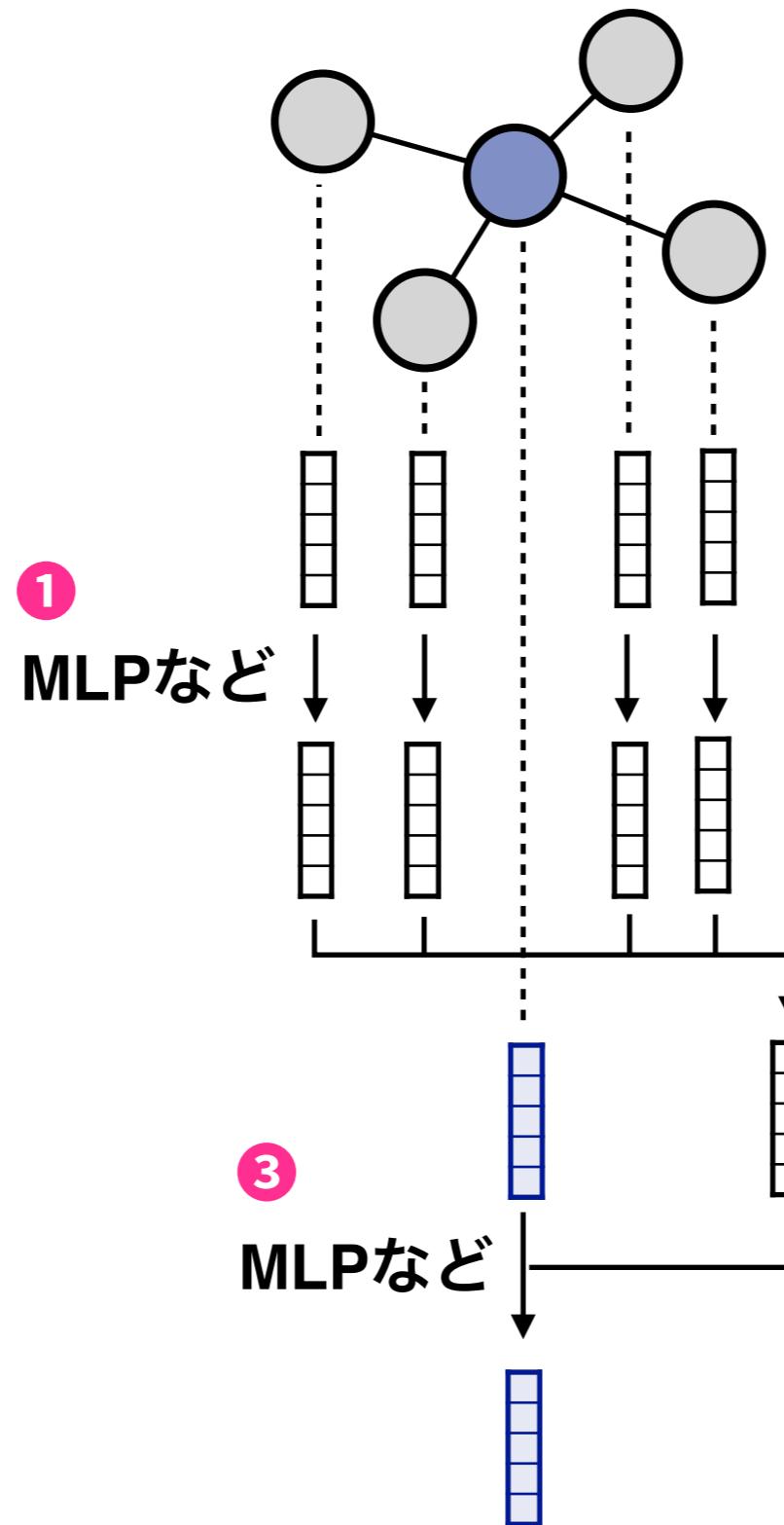
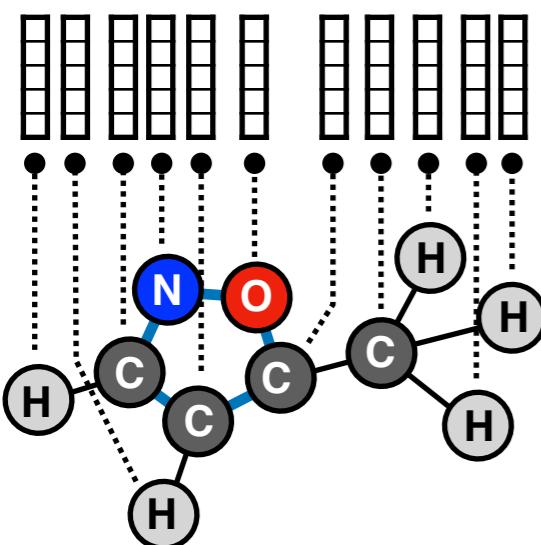
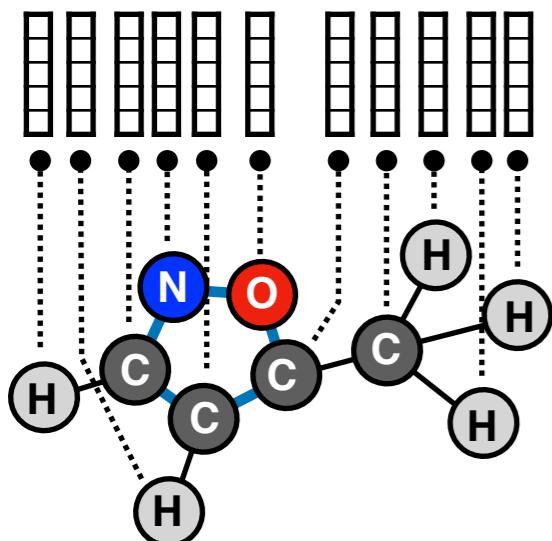
- **in\_features**
- **out\_features**
- **bias (True/False)**

入力サイズ  
出力サイズ  
**bias** の有無



# Message Passing (or Neighborhood Aggregation)

各特徴ベクトルの更新



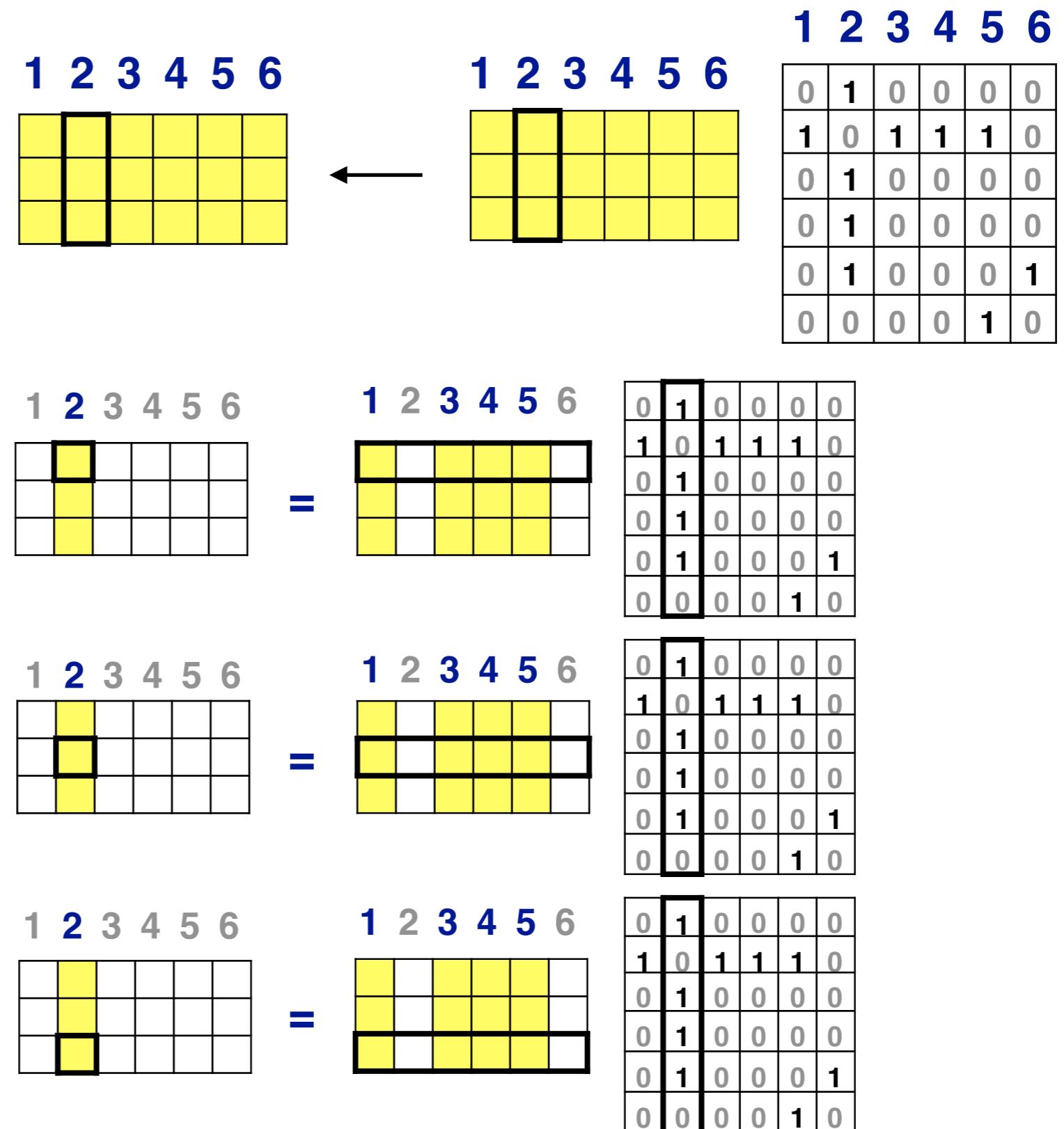
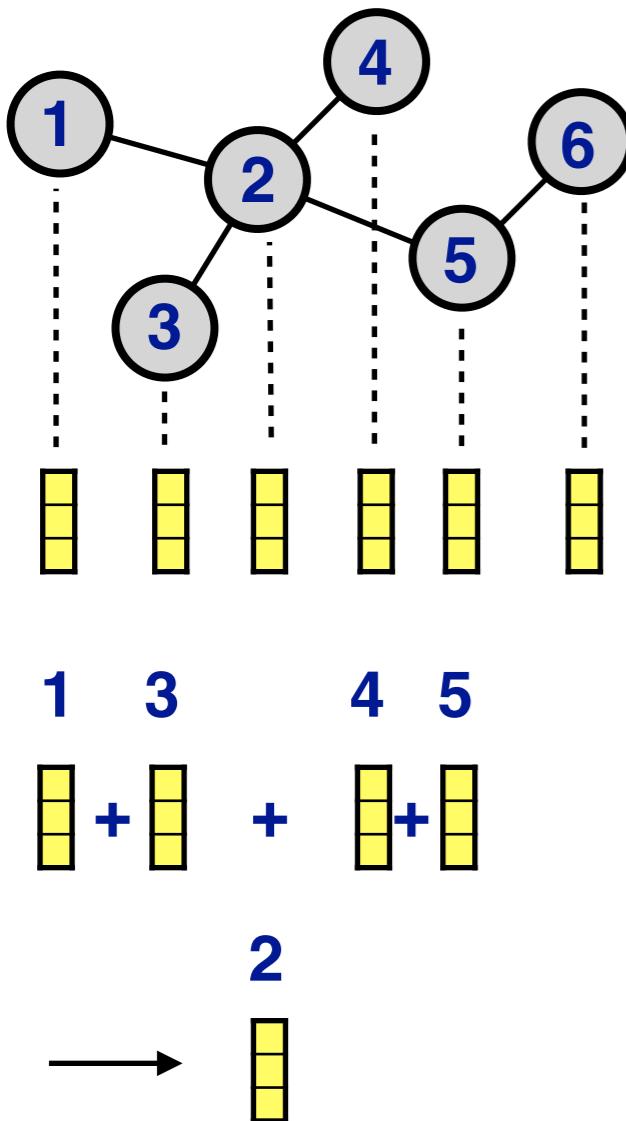
`torch_geometric.nn.  
MessagePassing`

$$\mathbf{x}'_i = \gamma_{\Theta} \left( \mathbf{x}_i, \square_{j \in \mathcal{N}(i)} \phi_{\Theta} \left( \mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{j,i} \right) \right)$$



②  
**順番に依存しない集約操作  
(sum, mean or max)**

# Message Passing (or Neighborhood Aggregation)



隣接行列などを使っても  
集約する操作は書ける

# Message Passing (or Neighborhood Aggregation)

---

- `torch_geometric.nn.GCNConv` (Kipf and Welling, ICML2017)

$$\mathbf{X}' = \hat{\mathbf{D}}^{-1/2} \hat{\mathbf{A}} \hat{\mathbf{D}}^{-1/2} \mathbf{X} \Theta$$

- `torch_geometric.nn.SAGEConv` (Hamilton et al, NIPS2017)

$$\mathbf{x}'_i = \mathbf{W}_1 \mathbf{x}_i + \mathbf{W}_2 \cdot \text{mean}_{j \in \mathcal{N}(i)} \mathbf{x}_j$$

- `torch_geometric.nn.GraphConv` (Morris et al, AAAI2019)

$$\mathbf{x}'_i = \Theta_1 \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \Theta_2 \mathbf{x}_j$$

- `torch_geometric.nn.GINConv` (Xu et al, ICLR2019)

$$\mathbf{x}'_i = h_\Theta \left( (1 + \epsilon) \cdot \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \right) \quad \text{or} \quad \mathbf{X}' = h_\Theta ((\mathbf{A} + (1 + \epsilon) \cdot \mathbf{I}) \cdot \mathbf{X})$$

# Message Passing (or Neighborhood Aggregation)

- `torch_geometric.nn.SchNet` (Schütt et al, NIPS2017)

$$\mathbf{x}'_i = \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j \odot h_{\Theta}(\exp(-\gamma(\mathbf{e}_{j,i} - \mu)))$$

○ 要素積

→ xyzでの距離を辺重みにして集約時に活用する

- `torch_geometric.nn.GATConv` (Veličković et al., ICLR2018)

$$\mathbf{x}'_i = \alpha_{i,i} \Theta \mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j} \Theta \mathbf{x}_j$$

$$\alpha_{i,j} = \frac{\exp(\text{LeakyReLU}(\mathbf{a}^\top [\Theta \mathbf{x}_i \parallel \Theta \mathbf{x}_j]))}{\sum_{k \in \mathcal{N}(i) \cup \{i\}} \exp(\text{LeakyReLU}(\mathbf{a}^\top [\Theta \mathbf{x}_i \parallel \Theta \mathbf{x}_k]))}$$

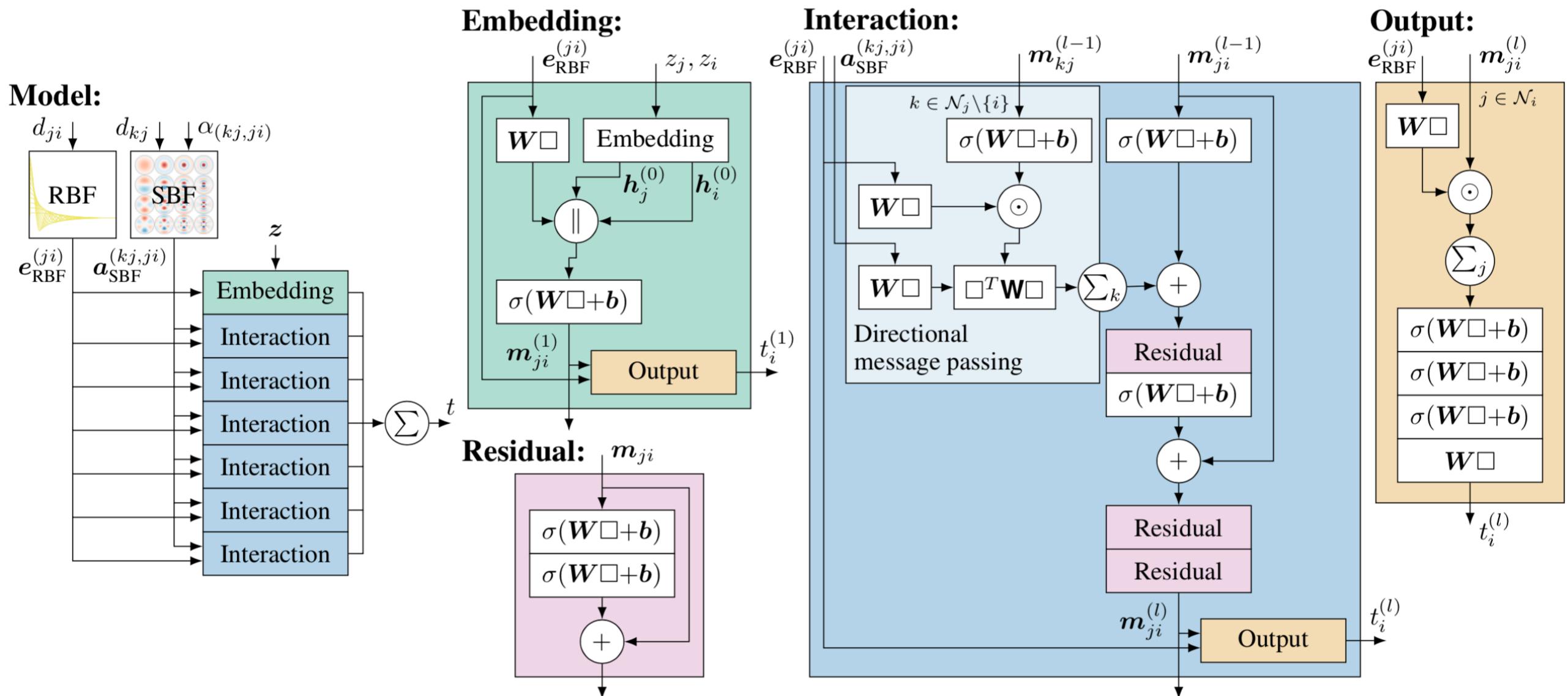
|| concat

→ 集約操作はシンプルだがアテンション機構で重みを適応的に

# Message Passing (or Neighborhood Aggregation)

- `torch_geometric.nn.DimeNet` (Klicpera et al, ICLR2020)

xyzに対して化学的なドメイン知識を活用して操作を作り込みまくった例



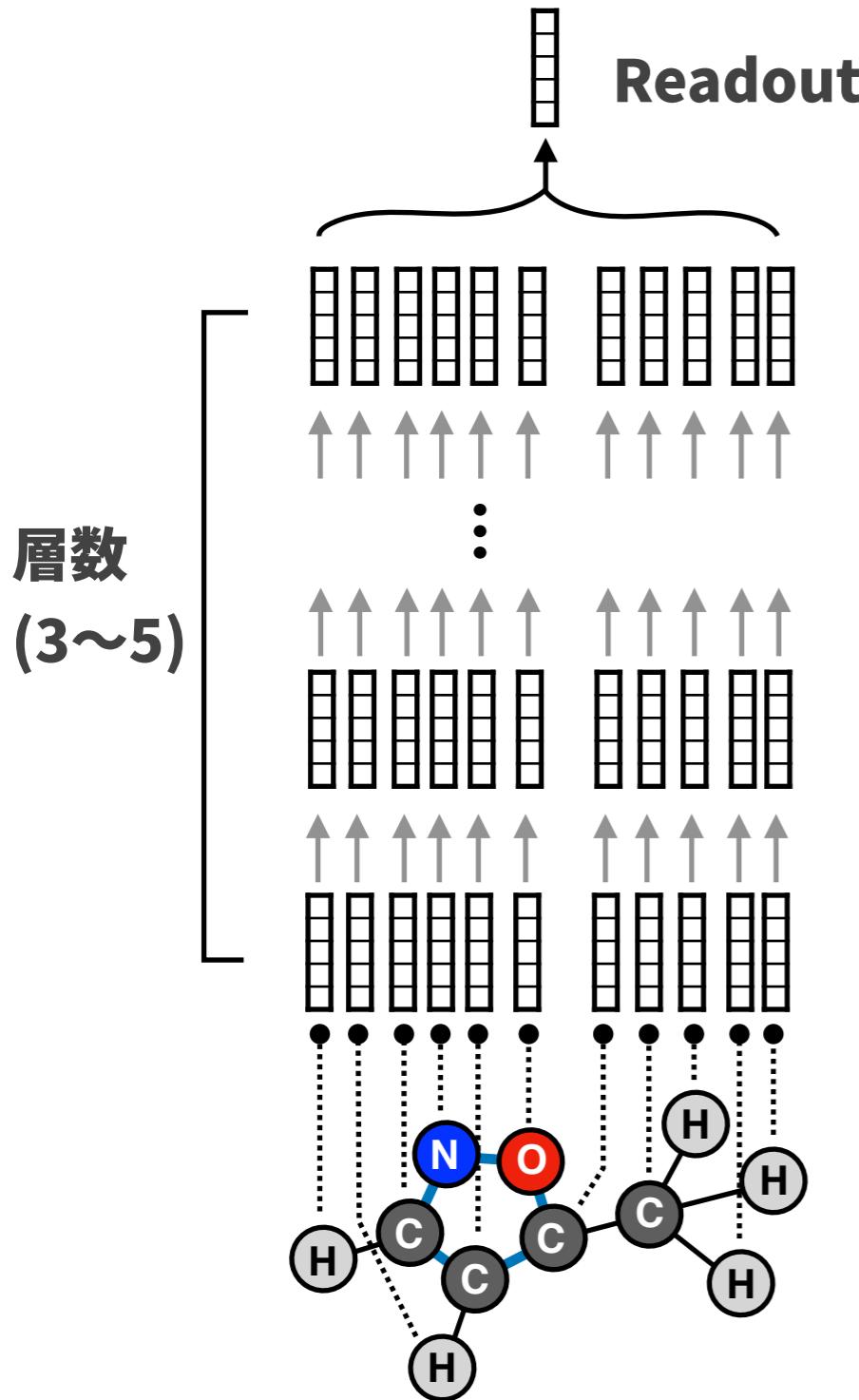
# Message Passing (or Neighborhood Aggregation)

最初に例に出したQM9データでは従来手法を優越する予測精度

Table 1: MAE on QM9. DimeNet sets the state of the art on 11 targets, outperforming the second-best model on average by 31 % (mean std. MAE).

Target	Unit	PPGN	SchNet	PhysNet	MEGNet-s	Cormorant	DimeNet
$\mu$	D	0.047	0.033	0.0529	0.05	0.13	<b>0.0286</b>
$\alpha$	$a_0^3$	0.131	0.235	0.0615	0.081	0.092	<b>0.0469</b>
$\epsilon_{\text{HOMO}}$	meV	40.3	41	32.9	43	36	<b>27.8</b>
$\epsilon_{\text{LUMO}}$	meV	32.7	34	24.7	44	36	<b>19.7</b>
$\Delta\epsilon$	meV	60.0	63	42.5	66	60	<b>34.8</b>
$\langle R^2 \rangle$	$a_0^2$	0.592	<b>0.073</b>	0.765	0.302	0.673	0.331
ZPVE	meV	3.12	1.7	1.39	1.43	1.98	<b>1.29</b>
$U_0$	meV	36.8	14	8.15	12	28	<b>8.02</b>
$U$	meV	36.8	19	8.34	13	-	<b>7.89</b>
$H$	meV	36.3	14	8.42	12	-	<b>8.11</b>
$G$	meV	36.4	14	9.40	12	-	<b>8.98</b>
$c_v$	$\frac{\text{cal}}{\text{mol K}}$	0.055	0.033	0.0280	0.029	0.031	<b>0.0249</b>
std. MAE	%	1.84	1.76	1.37	1.80	2.14	<b>1.05</b>
logMAE	-	-4.64	-5.17	-5.35	-5.17	-4.75	<b>-5.57</b>

# Readout: グラフに対する固定サイズ表現を得る



- `torch_geometric.nn.global_add_pool`  
`torch_geometric.nn.global_max_pool`  
`torch_geometric.nn.global_mean_pool`  
順番に依存しない集約操作 (sum, mean or max)

- `torch_geometric.nn.global_sort_pool`

最後の要素でsortしてトップkをconcat

- `torch_geometric.nn.GlobalAttention`

$$\mathbf{r}_i = \sum_{n=1}^{N_i} \text{softmax}(h_{\text{gate}}(\mathbf{x}_n)) \odot h_{\Theta}(\mathbf{x}_n)$$

- `torch_geometric.nn.Set2Set`

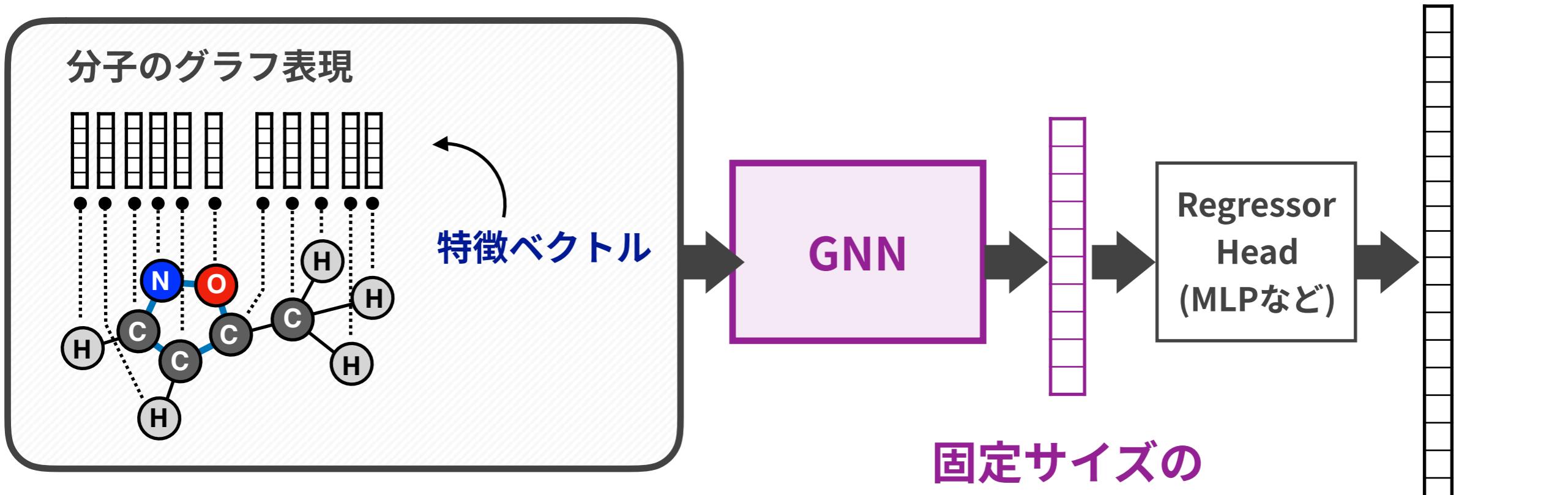
$$\mathbf{q}_t = \text{LSTM}(\mathbf{q}_{t-1}^*)$$

$$\alpha_{i,t} = \text{softmax}(\mathbf{x}_i \cdot \mathbf{q}_t)$$

$$\begin{aligned}\mathbf{r}_t &= \sum_{i=1}^N \alpha_{i,t} \mathbf{x}_i \\ \mathbf{q}_t^* &= \mathbf{q}_t \parallel \mathbf{r}_t,\end{aligned}$$

# GNN = 下記の変換を何らかの合成関数でデザインしたもの

可変サイズの分子グラフから「固定サイズのベクトル表現」への変換を学習



**mol.x** 頂点の特徴ベクトル (11次元)

**mol.pos** 頂点の位置ベクトル (3次元)

**mol.edge\_attr** 辺の特徴ベクトル (4次元)

**mol.edge\_index** 辺と結合原子ペアの対応

**mol.y**  
DFT計算値  
(19次元)

# PyTorch Geometricの例 (Google Colab)

```
!pip install torch-scatter==latest+cu101 torch-sparse==latest+cu101 \
    torch-cluster==latest+cu101 -f https://pytorch-geometric.com/whl/torch-1.6.0.html
!pip install torch-geometric
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch_geometric as pyg

from tqdm import tqdm

dataset = pyg.datasets.QM9(root='/tmp/QM9')
data_size = len(dataset)
loader = pyg.data.DataLoader(dataset[:int(data_size * 0.8)], batch_size=64, shuffle=True)
test_loader = pyg.data.DataLoader(dataset[int(data_size * 0.8):], batch_size=64, shuffle=True)
```

```
def mlp(input_dim, hidden_dim):
    return nn.Sequential(
        nn.Linear(input_dim, hidden_dim),
        nn.ReLU(),
        nn.Linear(hidden_dim, hidden_dim))
```

# モデルとforward定義

```
class MyGNN(nn.Module):
    def __init__(self, input_dim, hidden_dim, output_dim):
        super(MyGNN, self).__init__()
        self.convs = nn.ModuleList()
        self.convs.append(pyg.nn.GINConv(mlp(input_dim, hidden_dim)))
        for l in range(2):
            self.convs.append(pyg.nn.GINConv(mlp(hidden_dim, hidden_dim)))
        self.head = nn.Sequential(
            nn.Linear(hidden_dim, hidden_dim), nn.Dropout(0.25),
            nn.Linear(hidden_dim, output_dim))
```

Regresso  
Head (MLP)

```
def forward(self, data): ←———— 出力を計算するforward
    x, edge_index, batch = data.x, data.edge_index, data.batch
    for conv in self.convs:
        x = conv(x, edge_index)
        x = F.relu(x)
        x = F.dropout(x, p=0.25, training=self.training)
    x = pyg.nn.global_mean_pool(x, batch) ←———— Readout
    x = self.head(x)
    return x
```

Message  
Passing (x 3層)

# 学習：モデルのパラメタを最適にする

```
model = MyGNN(dataset.num_node_features, 32, dataset.num_classes)
```

```
opt = torch.optim.Adam(model.parameters(), lr=0.01)
error = torch.nn.MSELoss(reduction='sum')
```

```
for epoch in range(10):           ← とりあえず10 epoch (数百必要)
    total_loss = 0
    model.train()
    for batch in tqdm(loader):
        opt.zero_grad()
        pred = model(batch)
        loss = error(pred, batch.y)      ← モデルに入れて予測を得る
        loss.backward()                 ← lossを計算しbackward
        opt.step()                     ← 勾配降下(パラメタ更新)
        total_loss += loss.item() * batch.num_graphs
    total_loss /= len(loader.dataset)
    print("loss", total_loss, epoch)
```

## 注意

---

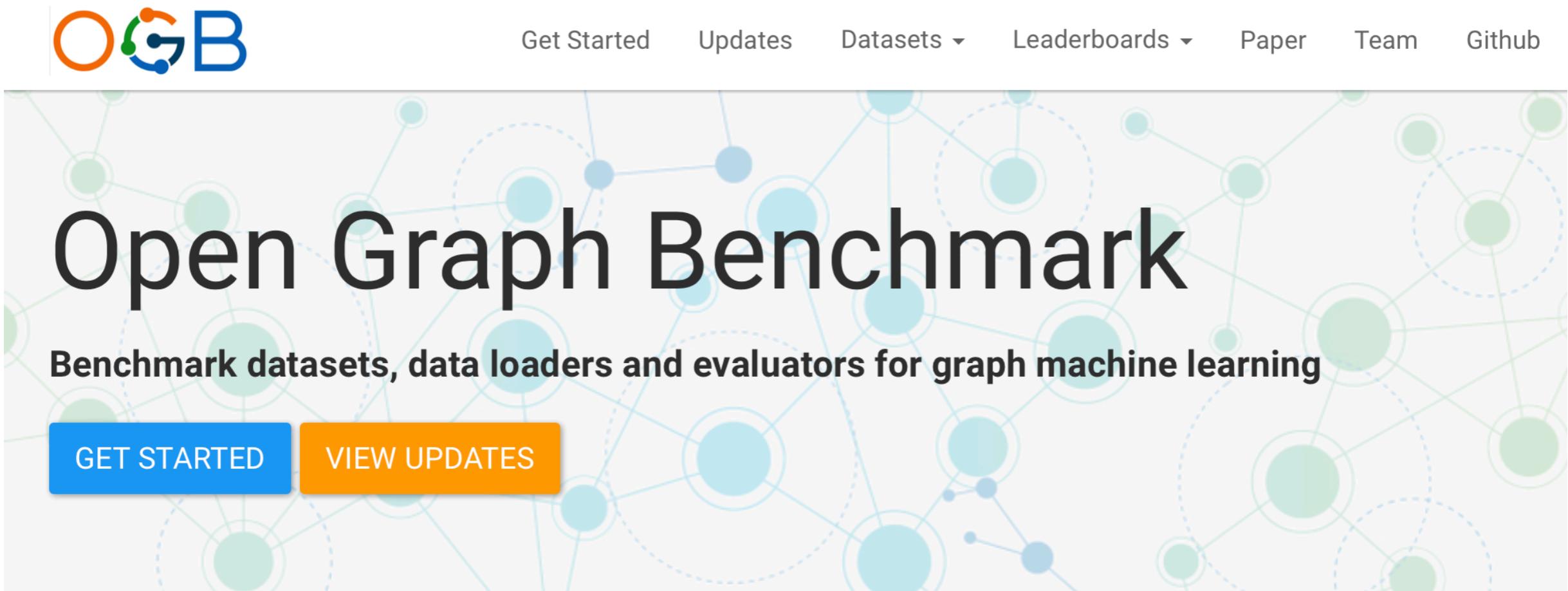
- QM9のタスクの場合はほぼxyz(x.pos)の情報が必要だが、単純なGNNはx.edge\_attrも使わないので適宜改変が必要
- 例えばx.edge\_attrに原子間距離を加えてConvで活用？

```
QM9_dist = []
for g in dataset:
    edge_attr = torch.empty(g.num_edges, g.num_edge_features + 1)
    for id, (s, t) in enumerate(zip(*g.edge_index)):
        d = torch.norm(g.pos[s]-g.pos[t])
        edge_attr[id] = torch.cat([g.edge_attr[id], d.reshape(1)])
    data = pyg.data.Data(x=g.x, z=g.z, pos=g.pos, edge_index=g.edge_index,
                          edge_attr=edge_attr, y=g.y, name=g.name, idx=id)
    QM9_dist.append(data)
```

- そのまま使う場合はSchNetやDimeNetが向いている
- 公式コードサンプルもチェック  
[https://github.com/rusty1s/pytorch\\_geometric/tree/master/examples](https://github.com/rusty1s/pytorch_geometric/tree/master/examples)

# Open Graph Benchmark <https://ogb.stanford.edu>

---



The Open Graph Benchmark (OGB) is a collection of realistic, large-scale, and diverse benchmark datasets for machine learning on graphs. OGB datasets are automatically downloaded, processed, and split using the [OGB Data Loader](#). The model performance can be evaluated using the [OGB Evaluator](#) in a unified manner.



# 本日の内容

---

GNN(Graph Neural Networks)でグラフ表現された分子を  
入力にして機械学習するやり方をのぞいてみませんか？

- 機械学習とは？
- 分子のグラフ表現
- 準備：深層学習のココロ (forwardとbackward)
- Graph Neural Networksとは？
- 自分でやってみるための情報
- おまけ：  
最近の話 (計算限界、分子生成、事前学習、...)

# Graph Neural Networksを知るための3つのソース

---

- CS224W: Machine Learning with Graphs (Stanford)

<http://web.stanford.edu/class/cs224w/>



 YouTube <https://youtu.be/-UjytpbqX4A>

- KDD2020 Tutorial  
**Deep Graph Learning: Foundations, Advances and Applications**

<https://ai.tencent.com/ailab/ml/KDD-Deep-Graph-Learning.html>

- Book "Deep Learning on Graphs"

[http://cse.msu.edu/~mayao4/dlg\\_book/index.html](http://cse.msu.edu/~mayao4/dlg_book/index.html)

# Pytorch Geometricを使ったハンズオン (CS224W)

<https://colab.research.google.com/drive/1DIQm9rOx2mT1bZETEeVUThxcrP1RKqAn>

The screenshot shows a Google Colab notebook interface. The title bar says 'Graph Neural Networks'. The menu bar includes 'ファイル' (File), '編集' (Edit), '表示' (View), '挿入' (Insert), 'ランタイム' (Runtime), 'ツール' (Tools), 'ヘルプ' (Help), and '変更は保存されません' (Changes not saved). The toolbar has buttons for '+ コード' (Code), '+ テキスト' (Text), and 'ドライブにコピー' (Copy to Drive). The right sidebar has icons for '共有' (Share), '接続' (Connect), '編集' (Edit), and a trash bin. The main content area has a title 'Graph Neural Networks' and a text block: 'In this tutorial, we will explore the implementation of graph neural networks and investigate what representations these networks learn. Along the way, we'll see how PyTorch Geometric and TensorBoardX can help us with constructing and training graph models.' Below this, a section titled '▼ Preliminaries: PyTorch' is expanded, containing text about PyTorch's features and imports. A code block shows the following imports:

```
[ ] Import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
import sklearn.metrics as metrics
```

# Kaggle CHAMPS (xyzが要るタスク)

<https://www.kaggle.com/c/champs-scalar-coupling/discussion/93972>

Featured Prediction Competition

## Predicting Molecular Properties

Can you measure the magnetic interactions between a pair of atoms?

 CHAMPS (CHeMistry And Mathematics in Phase Space) · 2,749 teams · a year ago

\$30,000 Prize Money

Overview Data Notebooks Discussion Leaderboard Rules New Topic

255th place

 Which graph CNN is the best (with starter kit at LB -1.469)?  219

Posted in [champs-scalar-coupling](#) a year ago

This post is for information and results using graph CNN. For a start:

useful Graph CNN lib:

- [https://github.com/rusty1s/pytorch\\_geometric](https://github.com/rusty1s/pytorch_geometric)
- <https://github.com/pfnet-research/chainer-chemistry>
- <https://github.com/jparkhill/TensorMol>

# ICLR2020: A Fair Comparisons of GNNs

---

<https://github.com/diningphil/gnn-comparison>

## A FAIR COMPARISON OF GRAPH NEURAL NETWORKS FOR GRAPH CLASSIFICATION

**Federico Errica\***

Department of Computer Science

University of Pisa

federico.errica@phd.unipi.it

**Marco Podda\***

Department of Computer Science

University of Pisa

marco.podda@di.unipi.it

**Davide Bacciu\***

Department of Computer Science

University of Pisa

bacciu@di.unipi.it

**Alessio Micheli\***

Department of Computer Science

University of Pisa

micheli@di.unipi.it

Table 3: Results on chemical datasets with mean accuracy and standard deviation are reported. Best performances are highlighted in bold.

	<b>D&amp;D</b>	<b>NCI1</b>	<b>PROTEINS</b>	<b>ENZYMES</b>
Baseline	<b>78.4</b> $\pm$ 4.5	69.8 $\pm$ 2.2	<b>75.8</b> $\pm$ 3.7	<b>65.2</b> $\pm$ 6.4
DGCNN	76.6 $\pm$ 4.3	76.4 $\pm$ 1.7	72.9 $\pm$ 3.5	38.9 $\pm$ 5.7
DiffPool	75.0 $\pm$ 3.5	76.9 $\pm$ 1.9	73.7 $\pm$ 3.5	59.5 $\pm$ 5.6
ECC	72.6 $\pm$ 4.1	76.2 $\pm$ 1.4	72.3 $\pm$ 3.4	29.5 $\pm$ 8.2
GIN	75.3 $\pm$ 2.9	<b>80.0</b> $\pm$ 1.4	73.3 $\pm$ 4.0	59.6 $\pm$ 4.5
GraphSAGE	72.9 $\pm$ 2.0	76.0 $\pm$ 1.8	73.0 $\pm$ 4.5	58.2 $\pm$ 6.0

# 本日の内容

---

GNN(Graph Neural Networks)でグラフ表現された分子を  
入力にして機械学習するやり方をのぞいてみませんか？

- 機械学習とは？
- 分子のグラフ表現
- 準備：深層学習のココロ (forwardとbackward)
- Graph Neural Networksとは？
- 自分でやってみるための情報
- おまけ：  
最近の話 (計算限界、分子生成、事前学習、...)

# GNNは発展途上の技術で現在進行形で研究されている

---

- 学習できる表現に限界がある？

**Generalization and Representational Limits of Graph Neural Networks**

Garg, Jegelka, Jaakkola (ICML2020)

- Deepにできない？

**DeepGCNs: Can GCNs Go as Deep as CNNs?**

Li, Müller, Thabet, Ghanem (ICCV2019)

**PairNorm: Tackling Oversmoothing in GNNs**

Zhao, Akoglu (ICLR2020)

**Towards Deeper Graph Neural Networks with Differentiable Group Normalization**

Zhou, Huang, Li, Zha, Chen, Hu (NeurIPS2020)

- 事前学習ができない？

**Strategies for Pre-training Graph Neural Networks**

Hu, Liu, Gomes, Zitnik, Liang, Pande, Leskovec (ICLR2020)

**GROVER: Self-Supervised Message Passing Transformer on Large-scale Molecular Graphs** Rong, Bian, Xu, Xie, Wei, Huang, Huang (NeurIPS2020)

# 今日の話の応用・発展

---

## ● Graph Pooling

**Rethinking Pooling in Graph Neural Networks**

Mesquita, Souza, Kaski (NeurIPS2020)

**Path Integral Based Convolution and Pooling for Graph Neural Networks**

Ma, Xuan, Guang Wang, Li, Liò (NeurIPS2020)

## ● 分子の生成

**Hierarchical Generation of Molecular Graphs using Structural Motifs**

Jin, Barzilay, Jaakkola (ICML2020)

**GraphAF: a Flow-based Autoregressive Model for Molecular Graph Generation**

Shi, Xu, Zhu, Zhang, Zhang, Tang (ICLR2020)

## ● 化学反応の予測・逆合成

**RetroXpert: Decompose Retrosynthesis Prediction like A Chemist**

Yan, Ding, Zhao, Zheng, Yang, Yu, Huang (NeurIPS2020)

**A Graph to Graphs Framework for Retrosynthesis Prediction**

Shi, Xu, Guo, Zhang, Tang (ICML2020)

# Purely Data-Drivenの限界：vs Chematica

nature 2020.10

## Computational planning of the synthesis of complex natural products

Barbara Mikulak-Klucznik, Patrycja Gołębiowska, Alison A. Bayly, Oskar Popik, Tomasz Klucznik, Sara Szymkuć, Ewa P. Gajewska, Piotr Dittwald, Olga Staszewska-Krajewska, Wiktor Beker, Tomasz Badowski, Karl A. Scheidt, Karol Molga , Jacek Mlynarski , Milan Mrksich  & Bartosz A. Grzybowski 

Nature (2020) | Cite this article

Science 2020.09

### RESEARCH ARTICLE

## Synthetic connectivity, emergence, and self-regeneration in the network of prebiotic chemistry

 Agnieszka Wołos<sup>1,2,\*</sup>,  Rafał Roszak<sup>1,2,\*</sup>,  Anna Żądło-Dobrowolska<sup>1,\*</sup>,  Wiktor Beker<sup>1,2</sup>,  Barbara Mikulak Klucznik<sup>1,2</sup>,  Grzegorz Spólnik<sup>1</sup>,  Mirosław Dygas<sup>1</sup>,  Sara Szymkuć<sup>1,2,†</sup>,  Bartosz A. Grzybowski<sup>1,2,3,4,†</sup>

<sup>1</sup>Institute of Organic Chemistry, Polish Academy of Sciences, Warsaw, Poland.

<sup>2</sup>Allchemy, Inc., Highland, IN, USA.

<sup>3</sup>Center for Soft and Living Matter of Korea's Institute for Basic Science (IBS), Ulsan, South Korea.

<sup>4</sup>Department of Chemistry, Ulsan National Institute of Science and Technology, Ulsan, South Korea.

✉ Corresponding author. Email: [saraszymkuc@gmail.com](mailto:saraszymkuc@gmail.com) (S.S.); [nanogrzybowski@gmail.com](mailto:nanogrzybowski@gmail.com) (B.A.G.)

\* These authors contributed equally to this work.

- Hide authors and affiliations

**Limitations of data-driven AI in advanced synthesis planning.** There are multiple reasons why purely data-oriented AI methods trained on even the largest reaction repositories (Reaxys, SciFinder, etc.) but without mechanistic insight are not adequate for natural product syntheses: (i) The reaction rules automatically derived from such repositories do not capture all relevant stereochemical information (not to mention that a sizeable fraction of reactions suffer from manual-entry errors; for comprehensive discussion of these and other aspects, see<sup>26</sup>); (ii) Since failed and side-reactions are usually *not* published, the machine cannot deduce the scope of potentially conflicting groups – this is especially problematic when a given reaction type has only few literature precedents for complex scaffolds and it is not warranted to assume that not-reported reactions are impossible (see also next paragraph); (iii) These repositories are largely dominated by simple, non-stereoselective reactions and so scoring-functions derived from them to select most promising reaction “moves” are geared towards these simpler and more popular chemistries (for detailed discussion see<sup>21</sup>). Because chess-like, total syntheses of natural products are a very small fraction of these repositories, the machine is not incentivized to learn and purposefully apply sequences of position-building but not structure-simplifying steps, does not learn for which rare scaffolds it is worth to perform FGIs or structure-complexifying steps, when to perform two or more different reactions on the same molecule, which reactions should be performed in tandem, or how to navigate around intermittent reactivity conflicts.

**Reaction rules, reactivity conflicts, and selectivity issues.** The main set of Chematica’s >100,000 expert-coded transforms are mechanism-based – these rules generalize and are broader than the underlying literature precedents. From the beginning of our effort on Chematica, the rules have been added to the software following an

# まとめ

---

GNN(Graph Neural Networks)でグラフ表現された分子を  
入力にして機械学習するやり方をのぞいてみませんか？

- 機械学習とは？
- 分子のグラフ表現
- 準備：深層学習のココロ (forwardとbackward)
- Graph Neural Networksとは？
- 自分でやってみるための情報
- おまけ：  
最近の話 (計算限界、分子生成、事前学習、...)