

# 深層学習 Day1

秋葉洋哉

2024年6月23日

## 1 ニューラルネットワーク

### 1.1 識別モデルと生成モデル

識別モデルとは、データを目的のクラスに分類するための方法である。一方で、生成モデルとは、特定のクラスのデータを生成するための方法である。例えば、犬という画像を入力で与えた場合にそれが犬である確率を出力するのが識別モデルであり、犬というクラスを与えた場合に犬の画像を生成するのが生成モデルである。

識別モデルは、基本的に高次元 (=データ量の多いもの) から低次元 (=データ量の少ないもの) へと変換することに長けており、画像認識等で用いられる。一方で、生成モデルでは、低次元から高次元へと変換することに長けており、画像の生成等で用いられる。

識別モデルでは、決定木・ロジスティック回帰・SVM・ニューラルネットワークなどがある。生成モデルでは、隠れマルコフモデル・ベイジアンネットワーク・変分オートエンコーダー (VAE)・敵対的生成ネットワーク (GAN) などがある。

生成モデルは、ベイズの定理を用いて識別器を作成する方法である。入力が生まれる確率と、クラスが生まれる確率を与え、それを用いてクラスを推定する。つまり、

$$P(x|C_k)P(C_k) \quad (1)$$

によって各クラスの生起確率を求め、最大のクラスを選択するというものである。生成モデルではデータを人工的に生成できることが特徴である。ベイズの定理は、

$$P(C_k|x) = \frac{P(x|C_k)P(C_k)}{P(x)} \quad (2)$$

で表される。

### 1.2 万能近似定理

万能近似定理とは、任意の連続関数をニューラルネットワークで近似できるという定理である。この定理により、ニューラルネットワークは、非常に高い表現力を持つことがわかる。この定理により、ニューラルネットワークは、多くの分野で利用されている。

### 1.3 ニューラルネットワークの概要

ニューラルネットワークは、脳の神経細胞を模倣したモデルである。ニューラルネットワークは、入力層・中間層・出力層から構成される。入力層は、入力データを受け取る層であり、中間層は、入力データを変換する層であり、出力層は、データを出力する層である。多層ペーセプトロン (MLP)・畳み込みニューラルネットワーク (CNN)・再帰型ニューラルネットワーク (RNN) といった種類を有する。

確認テスト

Q: ディープラーニングは、結局なにをやろうとしているか 2 行以内で述べよ。また、最適化する値は何か。

A: ディープラーニングは、多層のニューラルネットワークを用いて、高度な特徴を抽出し、複雑な問題を解決することを目的としている。最適化する値は、誤差関数を最小化する重み  $w$  とバイアス  $b$  である。

Q: 入力層 2 ノード 1 層・中間層 3 ノード 2 層・出力層 1 ノード 1 層のネットワークを書け。

A: 図 1 を参照。

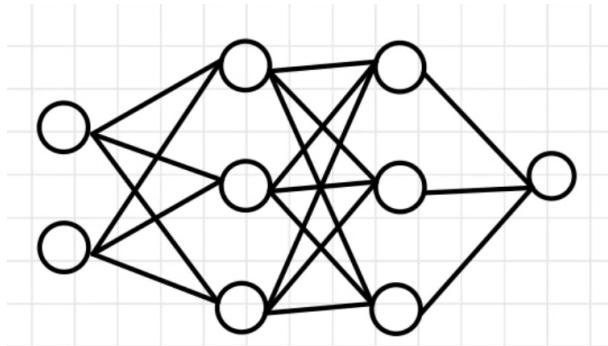


図 1: 入力層 2 ノード 1 層・中間層 3 ノード 2 層・出力層 1 ノード 1 層のネットワーク

ニューラルネットワークは、自動売買・チャットボット・画像認識・音声認識・自然言語処理・囲碁将棋 AI など、多くの分野で利用されている。

ニューラルネットワークの 1 つのノードに対して、入力  $x_i$ , 重み  $w_i$ , バイアス  $b$ , 総入力  $u$ , 出力  $z$ , 活性化関数  $f$  とすると、

$$u = \sum_{i=1}^n w_i x_i + b \quad (3)$$

$$z = f(u) \quad (4)$$

で表される。この出力は次のノードの入力として用いられることになる。

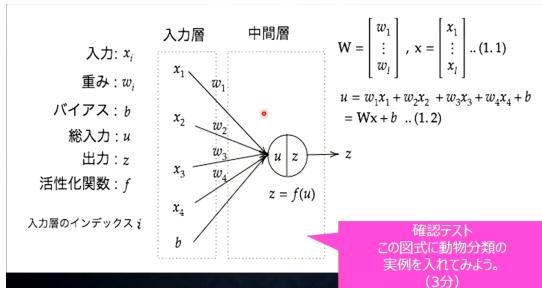
### 確認テスト

Q: 図 2a に動物分類の実例を入れろ。

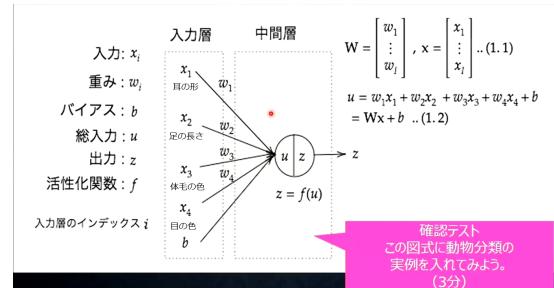
A: 図 2b を参照

Q: 図 3a の式を書け。

A: 図 3b を参照

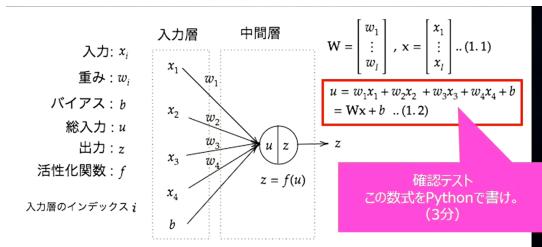


(a)



(b)

図 2



(a)

```

1 x = np.array([109, 455, 322, 504]) #入力値
2 W = np.array([[0.12, 0.20, 0.24, 0.55],
3               [0.34, 0.42, 0.14, 0.95],
4               [0.41, 0.03, 0.94, 0.22],
5               [0.40, 0.22, 0.41, 0.80]]) #重み
6 b = np.array(5.51) #バイアス
7 u = np.dot(x, W) + b #総入力
8 print(u)
9
[506.91 338.95 604.69 971.75]

```

(b)

図 3

中間層と出力層で用いられる活性化関数に違いがある。中間層では、ReLU 関数、シグモイド関数、ステップ関数といった関数が用いられることが多い。出力層では、ソフトマックス関数、シグモイド関数、恒等関数といった関数が用いられることが多い。出力層において用いる活性化関数と誤差関数は、扱う問題に応じて選択する必要がある。

1. 回帰問題の場合 → 恒等関数・二乗和誤差
2. 2 値分類問題の場合 → シグモイド関数・交差エントロピー誤差
3. 多クラス分類問題の場合 → ソフトマックス関数・交差エントロピー誤差

## 1.4 活性化関数

活性化関数とは、入力を出力に変換する関数である。活性化関数には、ステップ関数・シグモイド関数・ReLU関数・恒等関数・ソフトマックス関数などがある。

### 1.4.1 ステップ関数

閾値を超えたら発火する関数で、0か1を示す。Pythonのサンプルコードを示す。

```
def step_function(x):
    if x > 0:
        return 1
    else:
        return 0
```

### 1.4.2 シグモイド関数

勾配消失問題がある関数で、0から1の値を示す。Pythonのサンプルコードを示す。

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

### 1.4.3 ReLU関数

勾配消失問題を回避し、スペース化によって過学習を防ぐ関数である。Pythonのサンプルコードを示す。

```
def relu(x):
    return np.maximum(0, x)
```

### 1.4.4 恒等関数

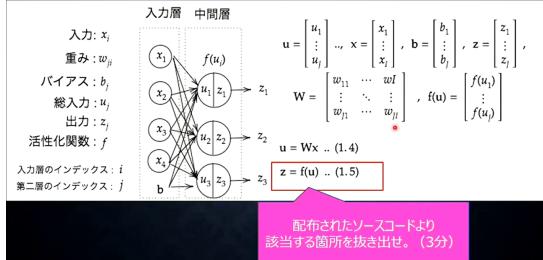
入力をそのまま出力する関数である。Pythonのサンプルコードを示す。

```
def identity_function(x):
    return x
```

### 確認テスト

Q: 図 4a の赤枠に該当する箇所をソースコードから抜き出せ。

A: 図 4b を参照



(a)

30 # 中間層出力  
31 z = functions.sigmoid(u)

(b)

図 4

### 1.4.5 ソフトマックス関数

出力の総和が 1 になる関数である。数式で表すと、以下の通りとなる。

$$f(i, u) = \frac{\exp(u_i)}{\sum_{k=1}^K \exp(u_k)} \quad (5)$$

Python のサンプルコードを示す。

```
def softmax(x):
    return np.exp(x) / np.sum(np.exp(x))
```

### 確認テスト

Q: 図 5a の丸 1 から丸 3 までの数式に該当するソースコードを示し、1 行ずつ処理の説明をせよ。

A: 図 5b を参照。それぞれの説明は以下の通り。

- ①. 2 層目の総出力  $u_2$  を入力として、ソフトマックス関数を適用する。
- ②. ソフトマックス関数の分子を計算する。
- ③. ソフトマックス関数の分母を計算する。

## 1.5 誤差関数

### 1.5.1 二乗和誤差

二乗和誤差とは、予測値と正解値の差の 2 乗和を 2 で割ったものである。数式で表すと、以下の通りとなる。

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^w (y_i - t_i)^2 \quad (6)$$

**softmax関数**

$$f(i, u) = \frac{e^{u_i}}{\sum_{k=1}^K e^{u_k}} \quad (1)$$

```
def softmax(x):
    if x.ndim == 2:
        x = x.T
    x = x - np.max(x, axis=0)
    y = np.exp(x) / np.sum(np.exp(x), axis=0)
    return y.T
x = x - np.max(x) # オーバーフロー対策
return np.exp(x) / np.sum(np.exp(x))
```

①～③の式に該当するソースコードを示し、一行づつ処理の説明をせよ。  
(5分)

(a)
(b)

図 5

### 確認テスト

Q: 二乗和誤差は、なぜ単なる引き算ではなく二乗するのか述べよ。

A: 二乗和誤差は、予測値と正解値の差を取ることで、正解値との誤差を表す。二乗することで、正解値との誤差が大きい場合に誤差が大きくなり、正解値との誤差が小さい場合に誤差が小さくなるようにするためにある。

Q: 二乗和誤差の  $\frac{1}{2}$  はどういう意味を持つか述べよ。

A:  $\sum_{i=1}^w (y_i - t_i)^2$  の微分値は以下の通りである。

$$\frac{\partial E}{\partial y_i} = \frac{\partial}{\partial y_i} \left( \sum_{i=1}^n (y_i - t_i)^2 \right) \quad (7)$$

$$= 2(y_i - t_i) \quad (8)$$

二乗和誤差の定義式で  $\frac{1}{2}$  を掛けるのは、前に出た 2 を消し、微分値を  $(y_i - t_i)$  として、計算を簡単にするためである。

### 平均二乗和誤差と二乗和誤差

機械学習でよく用いられる平均二乗誤差 (Mean Squared Error: MSE) は、二乗和誤差の分母 2 をデータ数に変えて計算したものである。数式で表すと、以下の通りとなる。

$$E(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^w (y_i - t_i)^2 \quad (9)$$

平均二乗和誤差は、データ数によって誤差のスケールが変わるため、データ数が異なる場合でも誤差を比較することができる。一方で、二乗和誤差は、データ数によって誤差のスケールが変わらないため、データ数が異なる場合は誤差を比較することができないが、計算が簡単という特徴がある。

### 1.5.2 交差エントロピー誤差

交差エントロピー誤差とは、2つの確率分布の違いを表す指標である。数式で表すと、以下の通りとなる。

$$E(\mathbf{w}) = - \sum_{i=1}^w t_i \log(y_i) \quad (10)$$

ただし、 $t_i$  は正解ラベル、 $y_i$  は予測値を表す。この式を例を示しながら説明する。写真に写っている動物が犬であるとする。分類器は、犬・猫・ネズミを見分けることができるとし、いずれかのクラスの確率を出力する。今、犬・猫・ネズミのそれぞれの確率を 0.8, 0.1, 0.1 であると予測した場合、交差エントロピー誤差は、

$$E = -1.0 * \log(0.8) - 0.0 * \log(0.1) - 0.0 * \log(0.1) \quad (11)$$

$$= 0.223 \quad (12)$$

となる。次に、犬・猫・ネズミのそれぞれの確率を 0.4, 0.3, 0.3 であると予測した場合、交差エントロピー誤差は、

$$E = -1.0 * \log(0.4) - 0.0 * \log(0.3) - 0.0 * \log(0.3) \quad (13)$$

$$= 0.52 \quad (14)$$

となる。このように、交差エントロピー誤差は、予測が正解に近いほど小さくなり、予測が不正解に近いほど大きくなる。これは直観的にも誤差関数として交差エントロピー誤差が適していることが分かる。

確認テスト

Q: 図 6a の丸 1 から丸 2 までの数式に該当するソースコードを示し、1行づつ処理の説明をせよ。

A: 図 6b を参照。それぞれの説明は以下の通り。

- ①. 正解値  $d$ 、予測値  $y$  を入力を与えて、交差エントロピー誤差を呼び出す。
- ②.  $\log y^d = d \log y$  として、numpy の log 関数を用いて計算する。1.0d-7 を加えるのは、 $\log(0)$  を防ぐためである。

①

$$E_n(\mathbf{w}) = - \sum_{i=1}^l d_i \log y_i \quad .. \text{交差エントロピー}$$

②

①～②の数式に該当するソースコードを示し、一行づつ処理の説明をせよ。  
(5分)

```
def cross_entropy_error(d, y):
    if y.ndim == 1:
        d = d.reshape(1, d.size)
        y = y.reshape(1, y.size)
    # 教師データがone-hot-vectorの場合、正解ラベルのインデックスに変換
    if d.size == y.size:
        d = d.argmax(axis=1)
        batch_size = y.shape[0]
    return -np.sum(np.log(y[np.arange(batch_size), d] + 1e-7)) / batch_size
```

(a)

functions.py

```
1_1_forward_propagation_after.ipynb
63 # 誤差
64 loss = functions.cross_entropy_error(d, y)①

38 # クロスエントロピー
39 def cross_entropy_error(d, y):
40     if y.ndim == 1:
41         d = d.reshape(1, d.size)
42         y = y.reshape(1, y.size)
43
44     # 教師データがone-hot-vectorの場合、正解ラベルのインデックスに変換
45     if d.size == y.size:
46         d = d.argmax(axis=1)
47
48     batch_size = y.shape[0] ②
49     return -np.sum(np.log(y[np.arange(batch_size), d] + 1e-7)) / batch_size
50
```

(b)

図 6

### 1.5.3 カテゴリカル交差エントロピー誤差

カテゴリカル交差エントロピー誤差とは、多クラス分類問題において用いられる誤差関数である。数式で表すと、以下の通りとなる。

$$E = - \sum_{j=1}^{L-1} \sum_{i=1}^{M-1} t_{ji} \log(y_{ji}) \quad (15)$$

ただし、 $t_i$  は正解ラベル、 $y_i$  は予測値、 $L$  はクラス数、 $M$  はクラス数を表す。交差エントロピー誤差をクラス間で拡張し、すべてを足し合わせている。

## 1.6 勾配降下法

勾配降下法とは、導関数を用いて数値的に方程式の解を求めるための方法である。深層学習では、学習を通して誤差関数を最小にするパラメータ  $\mathbf{w}$  を求めることが目的であり、勾配降下法はそのための手法の一つとして用いられる。勾配降下法は、以下の手順で行われる。

1. パラメータ  $\mathbf{w}$  を初期化する
2. 誤差関数  $E(\mathbf{w})$  を最小化するために、 $\mathbf{w}$  を更新する
3. 収束するまで 2 を繰り返す

数式で表すと、以下の通りとなる。

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon E(\mathbf{w}) \quad (16)$$

ここで、 $\epsilon$  は学習率を表す。学習率は、パラメータの更新量を調整するためのハイパーパラメータであり、大きすぎると発散し、小さすぎると収束が遅くなるため、適切な値を設定する必要がある。勾配降下法には様々な種類があり、Momentum・AdaGrad・Adadelta・Adam などがある。

確認テスト

Q: 式 (16) に該当するソースコードを示せ。

A: 図 7 を参照

```
1_3_stochastic_gradient_descent.ipynb
110 # 勾配降下の繰り返し
111 for dataset in random_datasets:
112     x, d = dataset['x'], dataset['d']
113     z1, y = forward(network, x)
114     grad = backward(x, d, z1, y)
115     # パラメータに勾配適用
116     for key in ('W1', 'W2', 'b1', 'b2'):
117         network[key] -= learning_rate * grad[key]
```

図 7

## 1.7 確率的勾配降下法 (SGD)

確率的勾配降下法とは、勾配降下法の一種であり、毎回違う学習データをランダムに選んで行う方法である。計算コストを下げたり、局所解に陥ることを防ぐために用いられる。勾配降下法を用いた深層学習モデルの学習データの選び方は、

1. バッチ学習
2. ミニバッチ学習
3. オンライン学習

の 3 つが挙げられる。

バッチ学習は、エポック毎に全てのデータを使って傾きを逐次求め、最適解を見つける方法である。学習結果が安定しやすいという特徴を持つ。新たな訓練データが追加された場合に、再度全データを用いて計算を行う必要があるため、計算コストが高い。

ミニバッチ学習は、すべてのデータの中から一部を取り出して学習する方法で、局所最適解を回避しやすいという特徴を持つ。バッチ学習とオンライン学習の中間の特徴を持つ。すべてのデータで学習し終えた時に 1 エポックが終了し、重みが更新される。

オンライン学習は、すべてのデータの中から 1 つを取り出して学習する方法で、ミニバッチ学習よりも局所最適解を回避しやすいという特徴を持つ。計算コストが低いが、学習結果が不安定であるという特徴を持つ。

深層学習のモデリングにおいては、ミニバッチ学習が最も一般的に用いられる。その理由として、確率的勾配法のメリットを損なわずに、計算資源を有効利用することができるからである。例えば、1000 万件の学習データがあった場合、それをバッチ学習で学習しようとすると、1000 万件のデータを一度に計算する必要がある。現代のコンピュータでは、1000 万件のデータの計算には莫大な時間を要してしまう。そのため 1000 件ずつのミニバッチを作成し、それを 10000 回繰り返す際に、他の資源でも同時平行で計算することで時間を短縮させる。これには、CPU のスレッド並列化や GPU の SIMD 並列化という技術が用いられる。

確認テスト

Q:  $\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \epsilon \nabla E_t$  の数式の意味を図に書いて説明せよ。

A: 図 8 を参照

## 1.8 誤差逆伝播法

勾配降下法を用いて、重みとバイアスを更新する際に用いられる方法が誤差逆伝播法である。誤差逆伝播法は、出力層から入力層に向かって、誤差を逆伝播させることで、各層の重みとバイアスを更新する方法である。誤差逆伝播法は、以下の手順で行われる。

1. 順伝播
2. 誤差逆伝播
3. 重みとバイアスの更新

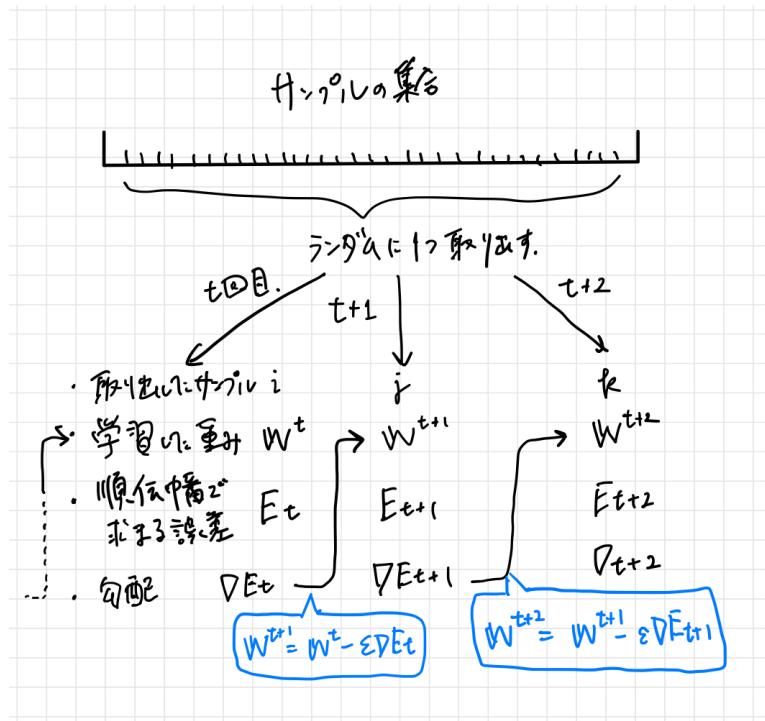


図 8

数式で表すと、以下の通りとなる。

$$w \leftarrow w - \epsilon \frac{\partial E}{\partial w} \quad (17)$$

$$b \leftarrow b - \epsilon \frac{\partial E}{\partial b} \quad (18)$$

ただし、 $w$  は重み、 $b$  はバイアス、 $\epsilon$  は学習率、 $E$  は誤差関数を表す。勾配を求めることができれば、上式を用いて重みとバイアスを更新することができる。

以下では、ノード  $i$  における前ノード  $j$  との間の重み  $w_{ij}$  の勾配を求める。まず、 $\frac{\partial E}{\partial w_{ij}}$  を求める。

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial u_i} \frac{\partial u_i}{\partial w_{ij}} \quad (19)$$

$u_i$  は、ノード  $i$  の総入力を表し、

$$u_i = \sum_{j=1}^m w_{ij} y_j + b_i \quad (20)$$

であるから、 $\frac{\partial u_i}{\partial w_{ij}}$  は、

$$\frac{\partial u_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left( \sum_{j=1}^m w_{ij} y_j + b_i \right) \quad (21)$$

$$= y_j \quad (22)$$

となる。また、 $\frac{\partial E}{\partial u_i}$  は、

$$\frac{\partial E}{\partial u_i} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial u_i} \quad (23)$$

で表される。ここで、 $E$  が二乗和誤差の場合、

$$\frac{\partial E}{\partial y_i} = \frac{\partial}{\partial y_i} \left( \frac{1}{2} \left( \sum_{i=1}^n (y_i - t_i)^2 \right) \right) \quad (24)$$

$$= y_i - t_i \quad (25)$$

となる。また、 $y_i$  が恒等関数の場合、

$$\frac{\partial y_i}{\partial u_i} = \frac{\partial}{\partial u_i} u_i \quad (26)$$

$$= 1 \quad (27)$$

となる。以上より、 $\frac{\partial E}{\partial u_i}$  は、

$$\frac{\partial E}{\partial u_i} = (y_i - t_i) \times 1 = y_i - t_i \quad (28)$$

となる。よって、 $\frac{\partial E}{\partial w_{ij}}$  は、

$$\frac{\partial E}{\partial w_{ij}} = (y_i - t_i) y_j \quad (29)$$

となる。同様に、バイアス  $b_i$  についても、

$$\frac{\partial E}{\partial b_i} = \frac{\partial E}{\partial u_i} \frac{\partial u_i}{\partial b_i} = y_i - t_i \quad (30)$$

となる。以上より、重みとバイアスの更新式は、

$$w_{ij} \leftarrow w_{ij} - \epsilon (y_i - t_i) y_j \quad (31)$$

$$b_i \leftarrow b_i - \epsilon (y_i - t_i) \quad (32)$$

となる。これを、ノード  $i$  におけるすべての前ノード  $j$  について計算するために、行列表現で表すと、

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon \mathbf{y}^T (y_i - t_i) \quad (33)$$

$$\mathbf{b} \leftarrow \mathbf{b} - \epsilon (y_i - t_i) \quad (34)$$

となる。ここで、 $\mathbf{y}$  を転置しているのは、 $\mathbf{w}$  と形状を合わせるためである。

図 9 に、誤差逆伝播の概要を示す。

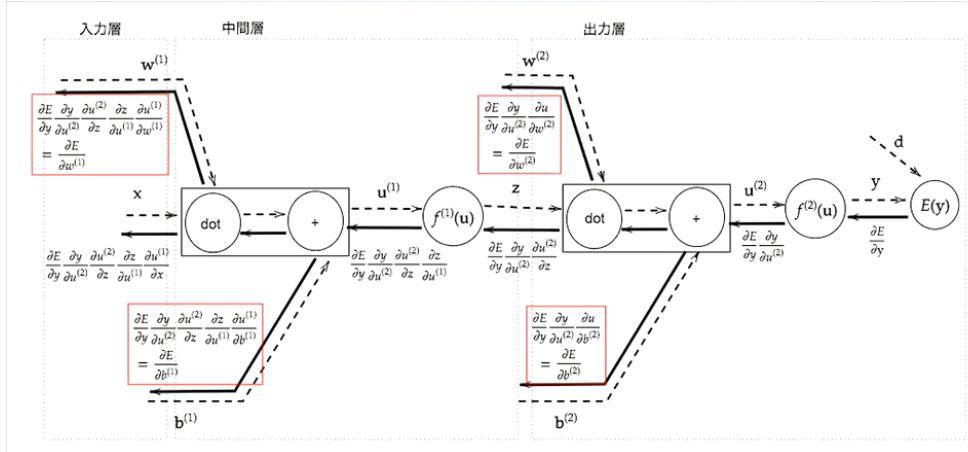


図 9: 誤差逆伝播法における勾配

### 確認テスト

Q: 誤差逆伝播法では不要な再帰的処理を避けることができる。すでに行った計算結果を保持しているソースコードを抽出せよ。

A: 図 10a を参照

Q:  $\frac{\partial E}{\partial y} \frac{\partial y}{\partial u}$  と、 $\frac{\partial E}{\partial y} \frac{\partial y}{\partial w_{ji}^{(2)}}$  に該当するソースコードを探せ。

A: 図 10b を参照

```

48 # 誤差逆伝播
49 def backward(x, d, z1, y):
50     # print("### 誤差逆伝播開始 #####")
51
52     grad = []
53
54     W1, W2 = network['W1'], network['W2']
55     b1, b2 = network['b1'], network['b2']
56
57     # 出力層でのデルタ
58     delta2 = functions.d_mean_squared_error(d, y)
59
60     # b2の勾配
61     grad['b2'] = np.sum(delta2, axis=0)
62
63     # W2の勾配
64     grad['W2'] = np.dot(z1.T, delta2)
65
66     # 中間層でのデルタ
67     #delta1 = np.dot(delta2, W2.T) * functions.d_relu(z1)
68
69     ## 試してみよう
70     delta1 = np.dot(delta2, W2.T) * functions.d_sigmoid(z1)
71
72     delta1 = delta1[:, np.newaxis]
73     # b1の勾配
74     grad['b1'] = np.sum(delta1, axis=0)
75     x = x[:, np.newaxis]
76     # W1の勾配
77     grad['W1'] = np.dot(x.T, delta1)

```

(a)

```

48 # 誤差逆伝播
49 def backward(x, d, z1, y):
50     # print("### 誤差逆伝播開始 #####")
51
52     grad = []
53
54     W1, W2 = network['W1'], network['W2']
55     b1, b2 = network['b1'], network['b2']
56
57     # 出力層でのデルタ
58     delta2 = functions.d_mean_squared_error(d, y)
59
60     # b2の勾配
61     grad['b2'] = np.sum(delta2, axis=0)
62
63     # W2の勾配
64     grad['W2'] = np.dot(z1.T, delta2)
65
66     # 中間層でのデルタ
67     #delta1 = np.dot(delta2, W2.T) * functions.d_relu(z1)
68
69     ## 試してみよう
70     delta1 = np.dot(delta2, W2.T) * functions.d_sigmoid(z1)
71
72     delta1 = delta1[:, np.newaxis]
73     # b1の勾配
74     grad['b1'] = np.sum(delta1, axis=0)
75     x = x[:, np.newaxis]
76     # W1の勾配
77     grad['W1'] = np.dot(x.T, delta1)

```

(b)

図 10

## 計算グラフ

計算グラフとは計算の過程をグラフで視覚化したものである。これを用いることで、複雑な偏微分の連なりを分解し、出力へ影響を与えるノードを絞りこむことができるようになる。計算グラフは複数の「ノード」と「エッジ」から構成される。例えば、 $a, b, c, d, e, f$  をノードとし、

$$e = ab \quad (35)$$

$$f = cd \quad (36)$$

$$g = e + f \quad (37)$$

を計算グラフで表すと、図 11 の青字ようになる。これは順伝播を表す。

ここで、最終的な出力である  $g$  が、それぞれのノードの影響をどの程度受けるか知りたいとする。偏微分について以下が成立することは明白である。

$$f(a, b) = ab \quad (38)$$

$$\frac{\partial f}{\partial a} = b \quad (39)$$

$$\frac{\partial f}{\partial b} = a \quad (40)$$

$$g(e, f) = e + f \quad (41)$$

$$\frac{\partial g}{\partial e} = 1 \quad (42)$$

$$\frac{\partial g}{\partial f} = 1 \quad (43)$$

$e, f$  は、値の大きさに依らずそれぞれ 1 の影響を  $g$  に与える。つまり、 $e$  の値が 1 変わったら  $g$  が 1 変わるという意味である。 $a, b, c, d$  は、 $e, f$  に対してそれぞれ 2, 3, 5, 4 の影響を与える。これらは同様に、 $a, b, c, d$  の値が 1 変わったら、 $e, f$  がそれぞれ 2, 3, 5, 4 変わるという意味である。さらに  $e, f$  は  $g$  に等倍の影響であるため、 $a, b, c, d$  が 1 変わったら、 $g$  が  $2 + 3 + 5 + 4 = 14$  変わることがわかる。これらの処理の流れは、図 11 の黄色で示されており、逆伝播を表す。

ここでは、計算グラフを用いることで、 $g$  に与える影響が最も大きいノードは  $c$  であることが分かった。

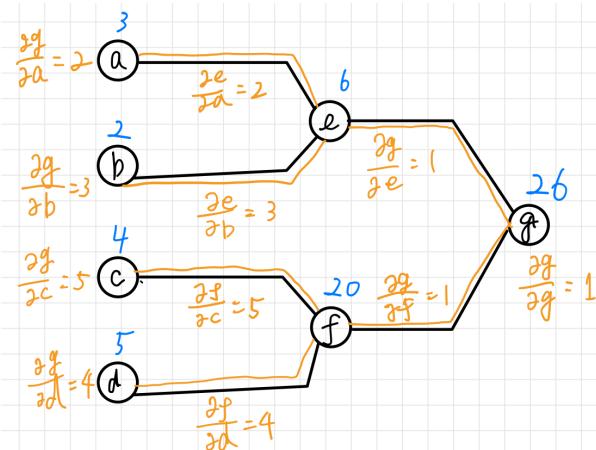


図 11: 計算グラフ

Affine 変換に対する計算グラフを表す。

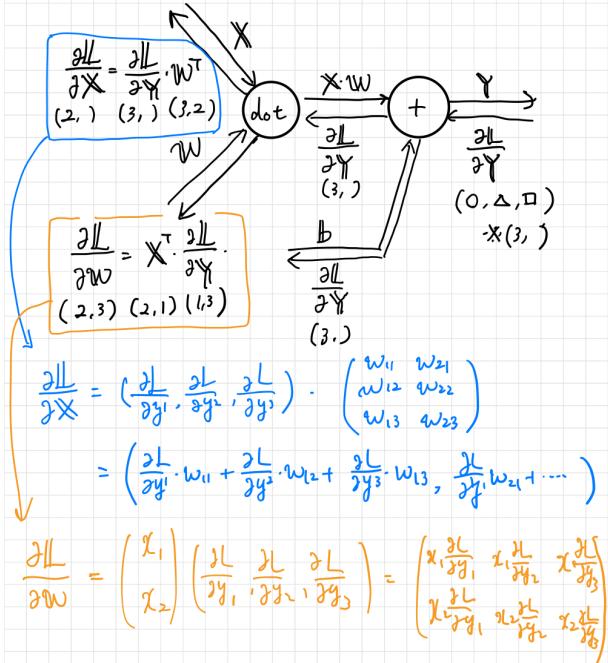


図 12: Affine 変換を計算グラフで表す

## ■参考文献

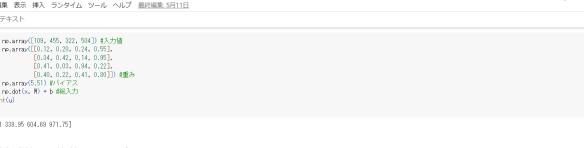
- 岡谷貴之/深層学習 改訂第2版 [機械学習プロフェッショナルシリーズ]/ 講談社サイエンティフィク/ 2022-01-17
- DeepLearning- 計算グラフについて理解する/@edo\_m18(Kazuya Hiruma) [https://qiita.com/edo\\_m18/items/7c95593ed5844b5a0c3b](https://qiita.com/edo_m18/items/7c95593ed5844b5a0c3b)
- 誤差逆伝播法を計算グラフを使って分かりやすく解説する [https://deeppage.net/deep\\_learning/2017/04/02/backpropagation.html](https://deeppage.net/deep_learning/2017/04/02/backpropagation.html)
- 交差エントロピー誤差をわかりやすく説明してみる/@kenta1984(Kenta Sasaki) <https://qiita.com/kenta1984/items/59a9ef1788e6934fd962>

## 2 実装演習キャプチャ

```
1 # Google Colab の実行環境を構成する
2 import os
3 import sys
4
5 if 'BM_CODE' in os.environ:
6     BM_CODE = True
7 else:
8     BM_CODE = False
9
10 if BM_CODE:
11     # Google Colab のマウント
12     from google.colab import drive
13     drive.mount('/content/drive')
14     os.chdir('/content/drive/MyDrive/Colab Notebooks/配布ノートブック/DNN_code_colab_dv1/notebook')
15 else:
16     # ローカル環境のマウント
17     os.chdir('DNN_code_colab_dv1/notebook')
```

```
1 import os
2 os.path.append(os.getcwd()) # 現ディレクトリのファイルをインポートするための設定
3
4 import numpy as np
5 from collections import abc
6
7 def print_vec(vec):
8     print("vec = " + str(vec))
9     print(vec)
10    print("vec.shape = " + str(vec.shape))
11    print("vec.dtype = " + str(vec.dtype))
12
13    print_vec("vec")
```

```
1 # マトリクス乗算 (準備・ユニット)
2
3 # 亂数生成
4 x = np.array([0.1, 0.2])
5 y = np.array([3, 4])
6
7 # 乱数生成
8 z = np.random.rand(3) # 3x1のランダム配列
9 w = np.random.rand(10, 3) # 10x3のランダム配列
10 v = np.random.rand(5, size=2)
11
12 print_vec("x")
13 print_vec("y")
14
15 x @ y
16 x * y
17 print_vec("z")
18
19 # 亂数生成
20 u = np.random.rand(8) + b
21 print_vec("u")
22
23 # 亂数生成
24 r = np.random.rand(10, 5)
25 print_vec("r")
26
27 # 亂数生成
28 s = np.random.rand(10, 12)
29 print_vec("s")
30
31 # 亂数生成
32 t = np.random.rand(12, 1)
33 print_vec("t")
34
35 # 亂数生成
36 u @ v
37 print_vec("uv")
38
39 # 亂数生成
40 r @ s
41 print_vec("rs")
42
43 # 亂数生成
44 r @ t
45 print_vec("rt")
46
47 # 亂数生成
48 s @ t
49 print_vec("st")
50
51 # 亂数生成
52 u @ r
53 print_vec("ur")
54
55 # 亂数生成
56 v @ s
57 print_vec("vs")
58
59 # 亂数生成
60 w @ v
61 print_vec("wv")
62
63 # 亂数生成
64 z @ w
65 print_vec("zw")
66
67 # 亂数生成
68 x @ z
69 print_vec("xz")
70
71 # 亂数生成
72 y @ x
73 print_vec("yx")
74
75 # 亂数生成
76 r @ w
77 print_vec("rw")
78
79 # 亂数生成
80 s @ r
81 print_vec("sr")
82
83 # 亂数生成
84 t @ s
85 print_vec("ts")
86
87 # 亂数生成
88 v @ r
89 print_vec("vr")
90
91 # 亂数生成
92 u @ v
93 print_vec("uv")
94
95 # 亂数生成
96 w @ u
97 print_vec("wu")
98
99 # 亂数生成
100 s @ w
101 print_vec("sw")
102
103 # 亂数生成
104 t @ w
105 print_vec("tw")
106
107 # 亂数生成
108 v @ w
109 print_vec("vw")
110
111 # 亂数生成
112 u @ w
113 print_vec("uw")
114
115 # 亂数生成
116 v @ t
117 print_vec("vt")
118
119 # 亂数生成
120 w @ t
121 print_vec("wt")
122
123 # 亂数生成
124 s @ t
125 print_vec("st")
126
127 # 亂数生成
128 r @ t
129 print_vec("rt")
130
131 # 亂数生成
132 v @ t
133 print_vec("vt")
134
135 # 亂数生成
136 u @ t
137 print_vec("ut")
138
139 # 亂数生成
140 w @ v
141 print_vec("wv")
142
143 # 亂数生成
144 s @ v
145 print_vec("sv")
146
147 # 亂数生成
148 r @ v
149 print_vec("rv")
150
151 # 亂数生成
152 v @ r
153 print_vec("vr")
154
155 # 亂数生成
156 u @ r
157 print_vec("ur")
158
159 # 亂数生成
160 v @ w
161 print_vec("vw")
162
163 # 亂数生成
164 w @ v
165 print_vec("wv")
166
167 # 亂数生成
168 s @ w
169 print_vec("sw")
170
171 # 亂数生成
172 t @ w
173 print_vec("tw")
174
175 # 亂数生成
176 v @ w
177 print_vec("vw")
178
179 # 亂数生成
180 u @ w
181 print_vec("uw")
182
183 # 亂数生成
184 t @ w
185 print_vec("tw")
186
187 # 亂数生成
188 v @ w
189 print_vec("vw")
190
191 # 亂数生成
192 u @ w
193 print_vec("uw")
194
195 # 亂数生成
196 v @ t
197 print_vec("vt")
198
199 # 亂数生成
200 w @ t
201 print_vec("wt")
202
203 # 亂数生成
204 s @ t
205 print_vec("st")
206
207 # 亂数生成
208 r @ t
209 print_vec("rt")
210
211 # 亂数生成
212 v @ t
213 print_vec("vt")
214
215 # 亂数生成
216 u @ t
217 print_vec("ut")
218
219 # 亂数生成
220 w @ v
221 print_vec("wv")
222
223 # 亂数生成
224 s @ v
225 print_vec("sv")
226
227 # 亂数生成
228 r @ v
229 print_vec("rv")
230
231 # 亂数生成
232 v @ r
233 print_vec("vr")
234
235 # 亂数生成
236 u @ r
237 print_vec("ur")
238
239 # 亂数生成
240 v @ w
241 print_vec("vw")
242
243 # 亂数生成
244 w @ v
245 print_vec("wv")
246
247 # 亂数生成
248 s @ w
249 print_vec("sw")
250
251 # 亂数生成
252 t @ w
253 print_vec("tw")
254
255 # 亂数生成
256 v @ w
257 print_vec("vw")
258
259 # 亂数生成
260 u @ w
261 print_vec("uw")
262
263 # 亂数生成
264 t @ w
265 print_vec("tw")
266
267 # 亂数生成
268 v @ w
269 print_vec("vw")
270
271 # 亂数生成
272 u @ w
273 print_vec("uw")
274
275 # 亂数生成
276 v @ t
277 print_vec("vt")
278
279 # 亂数生成
280 w @ t
281 print_vec("wt")
282
283 # 亂数生成
284 s @ t
285 print_vec("st")
286
287 # 亂数生成
288 r @ t
289 print_vec("rt")
290
291 # 亂数生成
292 v @ t
293 print_vec("vt")
294
295 # 亂数生成
296 u @ t
297 print_vec("ut")
298
299 # 亂数生成
300 w @ v
301 print_vec("wv")
302
303 # 亂数生成
304 s @ v
305 print_vec("sv")
306
307 # 亂数生成
308 r @ v
309 print_vec("rv")
310
311 # 亂数生成
312 v @ r
313 print_vec("vr")
314
315 # 亂数生成
316 u @ r
317 print_vec("ur")
318
319 # 亂数生成
320 v @ t
321 print_vec("vt")
322
323 # 亂数生成
324 w @ t
325 print_vec("wt")
326
327 # 亂数生成
328 s @ t
329 print_vec("st")
330
331 # 亂数生成
332 r @ t
333 print_vec("rt")
334
335 # 亂数生成
336 v @ t
337 print_vec("vt")
338
339 # 亂数生成
340 u @ t
341 print_vec("ut")
342
343 # 亂数生成
344 w @ v
345 print_vec("wv")
346
347 # 亂数生成
348 s @ v
349 print_vec("sv")
350
351 # 亂数生成
352 r @ v
353 print_vec("rv")
354
355 # 亂数生成
356 v @ r
357 print_vec("vr")
358
359 # 亂数生成
360 u @ r
361 print_vec("ur")
362
363 # 亂数生成
364 t @ w
365 print_vec("tw")
366
367 # 亂数生成
368 v @ w
369 print_vec("vw")
370
371 # 亂数生成
372 w @ v
373 print_vec("wv")
374
375 # 亂数生成
376 s @ w
377 print_vec("sw")
378
379 # 亂数生成
380 t @ w
381 print_vec("tw")
382
383 # 亂数生成
384 v @ w
385 print_vec("vw")
386
387 # 亂数生成
388 u @ w
389 print_vec("uw")
390
391 # 亂数生成
392 t @ w
393 print_vec("tw")
394
395 # 亂数生成
396 v @ w
397 print_vec("vw")
398
399 # 亂数生成
400 u @ w
401 print_vec("uw")
402
403 # 亂数生成
404 v @ t
405 print_vec("vt")
406
407 # 亂数生成
408 w @ t
409 print_vec("wt")
410
409 # 亂数生成
410 s @ t
411 print_vec("st")
412
413 # 亂数生成
414 r @ t
415 print_vec("rt")
416
417 # 亂数生成
418 v @ t
419 print_vec("vt")
420
421 # 亂数生成
422 u @ t
423 print_vec("ut")
424
425 # 亂数生成
426 w @ v
427 print_vec("wv")
428
429 # 亂数生成
430 s @ v
431 print_vec("sv")
432
433 # 亂数生成
434 r @ v
435 print_vec("rv")
436
437 # 亂数生成
438 v @ r
439 print_vec("vr")
440
441 # 亂数生成
442 u @ r
443 print_vec("ur")
444
445 # 亂数生成
446 v @ t
447 print_vec("vt")
448
449 # 亂数生成
450 w @ t
451 print_vec("wt")
452
453 # 亂数生成
454 s @ t
455 print_vec("st")
456
457 # 亂数生成
458 r @ t
459 print_vec("rt")
460
461 # 亂数生成
462 v @ t
463 print_vec("vt")
464
465 # 亂数生成
466 u @ t
467 print_vec("ut")
468
469 # 亂数生成
470 w @ v
471 print_vec("wv")
472
473 # 亂数生成
474 s @ v
475 print_vec("sv")
476
477 # 亂数生成
478 r @ v
479 print_vec("rv")
480
481 # 亂数生成
482 v @ r
483 print_vec("vr")
484
485 # 亂数生成
486 u @ r
487 print_vec("ur")
488
489 # 亂数生成
490 t @ w
491 print_vec("tw")
492
493 # 亂数生成
494 v @ w
495 print_vec("vw")
496
497 # 亂数生成
498 w @ v
499 print_vec("wv")
500
501 # 亂数生成
502 s @ w
503 print_vec("sw")
504
505 # 亂数生成
506 t @ w
507 print_vec("tw")
508
509 # 亂数生成
510 v @ w
511 print_vec("vw")
512
513 # 亂数生成
514 u @ w
515 print_vec("uw")
516
517 # 亂数生成
518 t @ w
519 print_vec("tw")
520
521 # 亂数生成
522 v @ w
523 print_vec("vw")
524
525 # 亂数生成
526 u @ w
527 print_vec("uw")
528
529 # 亂数生成
530 v @ t
531 print_vec("vt")
532
533 # 亂数生成
534 w @ t
535 print_vec("wt")
536
537 # 亂数生成
538 s @ t
539 print_vec("st")
540
541 # 亂数生成
542 r @ t
543 print_vec("rt")
544
545 # 亂数生成
546 v @ t
547 print_vec("vt")
548
549 # 亂数生成
550 u @ t
551 print_vec("ut")
552
553 # 亂数生成
554 w @ v
555 print_vec("wv")
556
557 # 亂数生成
558 s @ v
559 print_vec("sv")
560
561 # 亂数生成
562 r @ v
563 print_vec("rv")
564
565 # 亂数生成
566 v @ r
567 print_vec("vr")
568
569 # 亂数生成
570 u @ r
571 print_vec("ur")
572
573 # 亂数生成
574 v @ t
575 print_vec("vt")
576
577 # 亂数生成
578 w @ t
579 print_vec("wt")
580
581 # 亂数生成
582 s @ t
583 print_vec("st")
584
585 # 亂数生成
586 r @ t
587 print_vec("rt")
588
589 # 亂数生成
590 v @ t
591 print_vec("vt")
592
593 # 亂数生成
594 u @ t
595 print_vec("ut")
596
597 # 亂数生成
598 w @ v
599 print_vec("wv")
600
601 # 亂数生成
602 s @ v
603 print_vec("sv")
604
605 # 亂数生成
606 r @ v
607 print_vec("rv")
608
609 # 亂数生成
610 v @ r
611 print_vec("vr")
612
613 # 亂数生成
614 u @ r
615 print_vec("ur")
616
617 # 亂数生成
618 t @ w
619 print_vec("tw")
620
621 # 亂数生成
622 v @ w
623 print_vec("vw")
624
625 # 亂数生成
626 w @ v
627 print_vec("wv")
628
629 # 亂数生成
630 s @ w
631 print_vec("sw")
632
633 # 亂数生成
634 t @ w
635 print_vec("tw")
636
637 # 亂数生成
638 v @ w
639 print_vec("vw")
640
641 # 亂数生成
642 u @ w
643 print_vec("uw")
644
645 # 亂数生成
646 t @ w
647 print_vec("tw")
648
649 # 亂数生成
650 v @ w
651 print_vec("vw")
652
653 # 亂数生成
654 u @ w
655 print_vec("uw")
656
657 # 亂数生成
658 v @ t
659 print_vec("vt")
660
661 # 亂数生成
662 w @ t
663 print_vec("wt")
664
665 # 亂数生成
666 s @ t
667 print_vec("st")
668
669 # 亂数生成
670 r @ t
671 print_vec("rt")
672
673 # 亂数生成
674 v @ t
675 print_vec("vt")
676
677 # 亂数生成
678 u @ t
679 print_vec("ut")
680
681 # 亂数生成
682 w @ v
683 print_vec("wv")
684
685 # 亂数生成
686 s @ v
687 print_vec("sv")
688
689 # 亂数生成
690 r @ v
691 print_vec("rv")
692
693 # 亂数生成
694 v @ r
695 print_vec("vr")
696
697 # 亂数生成
698 u @ r
699 print_vec("ur")
700
701 # 亂数生成
702 v @ t
703 print_vec("vt")
704
705 # 亂数生成
706 w @ t
707 print_vec("wt")
708
709 # 亂数生成
710 s @ t
711 print_vec("st")
712
713 # 亂数生成
714 r @ t
715 print_vec("rt")
716
717 # 亂数生成
718 v @ t
719 print_vec("vt")
720
721 # 亂数生成
722 u @ t
723 print_vec("ut")
724
725 # 亂数生成
726 w @ v
727 print_vec("wv")
728
729 # 亂数生成
730 s @ v
731 print_vec("sv")
732
733 # 亂数生成
734 r @ v
735 print_vec("rv")
736
737 # 亂数生成
738 v @ r
739 print_vec("vr")
740
741 # 亂数生成
742 u @ r
743 print_vec("ur")
744
745 # 亂数生成
746 v @ t
747 print_vec("vt")
748
749 # 亂数生成
750 w @ t
751 print_vec("wt")
752
753 # 亂数生成
754 s @ t
755 print_vec("st")
756
757 # 亂数生成
758 r @ t
759 print_vec("rt")
760
761 # 亂数生成
762 v @ t
763 print_vec("vt")
764
765 # 亂数生成
766 u @ t
767 print_vec("ut")
768
769 # 亂数生成
770 w @ v
771 print_vec("wv")
772
773 # 亂数生成
774 s @ v
775 print_vec("sv")
776
777 # 亂数生成
778 r @ v
779 print_vec("rv")
780
781 # 亂数生成
782 v @ r
783 print_vec("vr")
784
785 # 亂数生成
786 u @ r
787 print_vec("ur")
788
789 # 亂数生成
790 v @ t
791 print_vec("vt")
792
793 # 亂数生成
794 w @ t
795 print_vec("wt")
796
797 # 亂数生成
798 s @ t
799 print_vec("st")
800
801 # 亂数生成
802 r @ t
803 print_vec("rt")
804
805 # 亂数生成
806 v @ t
807 print_vec("vt")
808
809 # 亂数生成
810 u @ t
811 print_vec("ut")
812
813 # 亂数生成
814 w @ v
815 print_vec("wv")
816
817 # 亂数生成
818 s @ v
819 print_vec("sv")
820
821 # 亂数生成
822 r @ v
823 print_vec("rv")
824
825 # 亂数生成
826 v @ r
827 print_vec("vr")
828
829 # 亂数生成
830 u @ r
831 print_vec("ur")
832
833 # 亂数生成
834 v @ t
835 print_vec("vt")
836
837 # 亂数生成
838 w @ t
839 print_vec("wt")
840
841 # 亂数生成
842 s @ t
843 print_vec("st")
844
845 # 亂数生成
846 r @ t
847 print_vec("rt")
848
849 # 亂数生成
850 v @ t
851 print_vec("vt")
852
853 # 亂数生成
854 u @ t
855 print_vec("ut")
856
857 # 亂数生成
858 w @ v
859 print_vec("wv")
860
861 # 亂数生成
862 s @ v
863 print_vec("sv")
864
865 # 亂数生成
866 r @ v
867 print_vec("rv")
868
869 # 亂数生成
870 v @ r
871 print_vec("vr")
872
873 # 亂数生成
874 u @ r
875 print_vec("ur")
876
877 # 亂数生成
878 v @ t
879 print_vec("vt")
880
881 # 亂数生成
882 w @ t
883 print_vec("wt")
884
885 # 亂数生成
886 s @ t
887 print_vec("st")
888
889 # 亂数生成
890 r @ t
891 print_vec("rt")
892
893 # 亂数生成
894 v @ t
895 print_vec("vt")
896
897 # 亂数生成
898 u @ t
899 print_vec("ut")
900
901 # 亂数生成
902 w @ v
903 print_vec("wv")
904
905 # 亂数生成
906 s @ v
907 print_vec("sv")
908
909 # 亂数生成
910 r @ v
911 print_vec("rv")
912
913 # 亂数生成
914 v @ r
915 print_vec("vr")
916
917 # 亂数生成
918 u @ r
919 print_vec("ur")
920
921 # 亂数生成
922 v @ t
923 print_vec("vt")
924
925 # 亂数生成
926 w @ t
927 print_vec("wt")
928
929 # 亂数生成
930 s @ t
931 print_vec("st")
932
933 # 亂数生成
934 r @ t
935 print_vec("rt")
936
937 # 亂数生成
938 v @ t
939 print_vec("vt")
940
941 # 亂数生成
942 u @ t
943 print_vec("ut")
944
945 # 亂数生成
946 w @ v
947 print_vec("wv")
948
949 # 亂数生成
950 s @ v
951 print_vec("sv")
952
953 # 亂数生成
954 r @ v
955 print_vec("rv")
956
957 # 亂数生成
958 v @ r
959 print_vec("vr")
960
961 # 亂数生成
962 u @ r
963 print_vec("ur")
964
965 # 亂数生成
966 v @ t
967 print_vec("vt")
968
969 # 亂数生成
970 w @ t
971 print_vec("wt")
972
973 # 亂数生成
974 s @ t
975 print_vec("st")
976
977 # 亂数生成
978 r @ t
979 print_vec("rt")
980
981 # 亂数生成
982 v @ t
983 print_vec("vt")
984
985 # 亂数生成
986 u @ t
987 print_vec("ut")
988
989 # 亂数生成
990 w @ v
991 print_vec("wv")
992
993 # 亂数生成
994 s @ v
995 print_vec("sv")
996
997 # 亂数生成
998 r @ v
999 print_vec("rv")
1000
1001 # 亂数生成
1002 v @ r
1003 print_vec("vr")
1004
1005 # 亂数生成
1006 u @ r
1007 print_vec("ur")
1008
1009 # 亂数生成
1010 w @ v
1011 print_vec("wv")
1012
1013 # 亂数生成
1014 s @ v
1015 print_vec("sv")
1016
1017 # 亂数生成
1018 r @ v
1019 print_vec("rv")
1020
1021 # 亂数生成
1022 v @ r
1023 print_vec("vr")
1024
1025 # 亂数生成
1026 u @ r
1027 print_vec("ur")
1028
1029 # 亂数生成
1030 v @ t
1031 print_vec("vt")
1032
1033 # 亂数生成
1034 w @ t
1035 print_vec("wt")
1036
1037 # 亂数生成
1038 s @ t
1039 print_vec("st")
1040
1041 # 亂数生成
1042 r @ t
1043 print_vec("rt")
1044
1045 # 亂数生成
1046 v @ t
1047 print_vec("vt")
1048
1049 # 亂数生成
1050 u @ t
1051 print_vec("ut")
1052
1053 # 亂数生成
1054 w @ v
1055 print_vec("wv")
1056
1057 # 亂数生成
1058 s @ v
1059 print_vec("sv")
1060
1061 # 亂数生成
1062 r @ v
1063 print_vec("rv")
1064
1065 # 亂数生成
1066 v @ r
1067 print_vec("vr")
1068
1069 # 亂数生成
1070 u @ r
1071 print_vec("ur")
1072
1073 # 亂数生成
1074 v @ t
1075 print_vec("vt")
1076
1077 # 亂数生成
1078 w @ t
1079 print_vec("wt")
1080
1081 # 亂数生成
1082 s @ t
1083 print_vec("st")
1084
1085 # 亂数生成
1086 r @ t
1087 print_vec("rt")
1088
1089 # 亂数生成
1090 v @ t
1091 print_vec("vt")
1092
1093 # 亂数生成
1094 u @ t
1095 print_vec("ut")
1096
1097 # 亂数生成
1098 w @ v
1099 print_vec("wv")
1100
1101 # 亂数生成
1102 s @ v
1103 print_vec("sv")
1104
1105 # 亂数生成
1106 r @ v
1107 print_vec("rv")
1108
1109 # 亂数生成
1110 v @ r
1111 print_vec("vr")
1112
1113 # 亂数生成
1114 u @ r
1115 print_vec("ur")
1116
1117 # 亂数生成
1118 v @ t
1119 print_vec("vt")
1120
1121 # 亂数生成
1122 w @ t
1123 print_vec("wt")
1124
1125 # 亂数生成
1126 s @ t
1127 print_vec("st")
1128
1129 # 亂数生成
1130 r @ t
1131 print_vec("rt")
1132
1133 # 亂数生成
1134 v @ t
1135 print_vec("vt")
1136
1137 # 亂数生成
1138 u @ t
1139 print_vec("ut")
1140
1141 # 亂数生成
1142 w @ v
1143 print_vec("wv")
1144
1145 # 亂数生成
1146 s @ v
1147 print_vec("sv")
1148
1149 # 亂数生成
1150 r @ v
1151 print_vec("rv")
1152
1153 # 亂数生成
1154 v @ r
1155 print_vec("vr")
1156
1157 # 亂数生成
1158 u @ r
1159 print_vec("ur")
1160
1161 # 亂数生成
1162 v @ t
1163 print_vec("vt")
1164
1165 # 亂数生成
1166 w @ t
1167 print_vec("wt")
1168
1169 # 亂数生成
1170 s @ t
1171 print_vec("st")
1172
1173 # 亂数生成
1174 r @ t
1175 print_vec("rt")
1176
1177 # 亂数生成
1178 v @ t
1179 print_vec("vt")
1180
1181 # 亂数生成
1182 u @ t
1183 print_vec("ut")
1184
1185 # 亂数生成
1186 w @ v
1187 print_vec("wv")
1188
1189 # 亂数生成
1190 s @ v
1191 print_vec("sv")
1192
1193 # 亂数生成
1194 r @ v
1195 print_vec("rv")
1196
1197 # 亂数生成
1198 v @ r
1199 print_vec("vr")
1200
1201 # 亂数生成
1202 u @ r
1203 print_vec("ur")
1204
1205 # 亂数生成
1206 v @ t
1207 print_vec("vt")
1208
1209 # 亂数生成
1210 w @ t
1211 print_vec("wt")
1212
1213 # 亂数生成
1214 s @ t
1215 print_vec("st")
1216
1217 # 亂数生成
1218 r @ t
1219 print_vec("rt")
1220
1221 # 亂数生成
1222 v @ t
1223 print_vec("vt")
1224
1225 # 亂数生成
1226 u @ t
1227 print_vec("ut")
1228
1229 # 亂数生成
1230 w @ v
1231 print_vec("wv")
1232
1233 # 亂数生成
1234 s @ v
1235 print_vec("sv")
1236
1237 # 亂数生成
1238 r @ v
1239 print_vec("rv")
1240
1241 # 亂数生成
1242 v @ r
1243 print_vec("vr")
1244
1245 # 亂数生成
1246 u @ r
1247 print_vec("ur")
1248
1249 # 亂数生成
1250 v @ t
1251 print_vec("vt")
1252
1253 # 亂数生成
1254 w @ t
1255 print_vec("wt")
1256
1257 # 亂数生成
1258 s @ t
1259 print_vec("st")
1260
1261 # 亂数生成
1262 r @ t
1263 print_vec("rt")
1264
1265 # 亂数生成
1266 v @ t
1267 print_vec("vt")
1268
1269 # 亂数生成
1270 u @ t
1271 print_vec("ut")
1272
1273 # 亂数生成
1274 w @ v
1275 print_vec("wv")
1276
1277 # 亂数生成
1278 s @ v
1279 print_vec("sv")
1280
1281 # 亂数生成
1282 r @ v
1283 print_vec("rv")
1284
1285 # 亂数生成
1286 v @ r
1287 print_vec("vr")
1288
1289 # 亂数生成
1290 u @ r
1291 print_vec("ur")
1292
1293 # 亂数生成
1294 v @ t
1295 print_vec("vt")
1296
1297 # 亂数生成
1298 w @ t
1299 print_vec("wt")
1300
1301 # 亂数生成
1302 s @ t
1303 print_vec("st")
1304
1305 # 亂数生成
1306 r @ t
1307 print_vec("rt")
1308
1309 # 亂数生成
1310 v @ t
1311 print_vec("vt")
1312
1313 # 亂数生成
1314 u @ t
1315 print_vec("ut")
1316
1317 # 亂数生成
1318 w @ v
1319 print_vec("wv")
1320
1321 # 亂数生成
1322 s @ v
1323 print_vec("sv")
1324
1325 # 亂数生成
1326 r @ v
1327 print_vec("rv")
1328
1329 # 亂数生成
1330 v @ r
1331 print_vec("vr")
1332
1333 # 亂数生成
1334 u @ r
1335 print_vec("ur")
1336
1337 # 亂数生成
1338 v @ t
1339 print_vec("vt")
1340
1341 # 亂数生成
1342 w @ t
1343 print_vec("wt")
1344
1345 # 亂数生成
1346 s @ t
1347 print_vec("st")
1348
1349 # 亂数生成
1350 r @ t
1351 print_vec("rt")
1352
1353 # 亂数生成
1354 v @ t
1355 print_vec("vt")
1356
1357 # 亂数生成
1358 u @ t
1359 print_vec("ut")
1360
1361 # 亂数生成
1362 w @ v
1363 print_vec("wv")
1364
1365 # 亂数生成
1366 s @ v
1367 print_vec("sv")
1368
1369 # 亂数生成
1370 r @ v
1371 print_vec("rv")
1372
1373 # 亂数生成
1374 v @ r
1375 print_vec("vr")
1376
1377 # 亂数生成
1378 u @ r
1379 print_vec("ur")
1380
1381 # 亂数生成
1382 v @ t
1383 print_vec("vt")
1384
1385 # 亂数生成
1386 w @ t
1387 print_vec("wt")
1388
1389 # 亂数生成
1390 s @ t
1391 print_vec("st")
1392
1393 # 亂数生成
1394 r @ t
1395 print_vec("rt")
1396
1397 # 亂数生成
1398 v @ t
1399 print_vec("vt")
1400
1401 # 亂数生成
1402 u @ t
1403 print_vec("ut")
1404
1405 # 亂数生成
1406 w @ v
1407 print_vec("wv")
1408
1409 # 亂数生成
1410 s @ v
1411 print_vec("sv")
1412
1413 # 亂数生成
1414 r @ v
1415 print_vec("rv")
1416
1417 # 亂数生成
1418 v @ r
1419 print_vec("vr")
1420
1421 # 亂数生成
1422 u @ r
1423 print_vec("ur")
1424
1425 # 亂数生成
1426 v @ t
1427 print_vec("vt")
1428
1429 # 亂数生成
1430 w @ t
1431 print_vec("wt")
1432
1433 # 亂数生成
1434 s @ t
1435 print_vec("st")
1436
1437 # 亂数生成
1438 r @ t
1439 print_vec("rt")
1440
1441 # 亂数生成
1442 v @ t
1443 print_vec("vt")
1444
1445 # 亂数生成
1446 u @ t
1447 print_vec("ut")
1448
1449 # 亂数生成
1450 w @ v
1451 print_vec("wv")
1452
1453 # 亂数生成
1454 s @ v
1455 print_vec("sv")
1456
1457 # 亂数生成
1458 r @ v
1459 print_vec("rv")
146
```

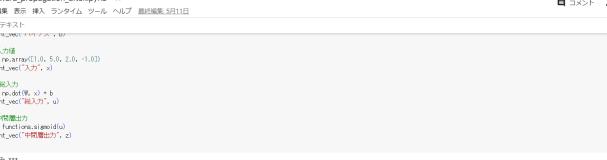


The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** "1\_1\_forward\_propagation\_after.ipynb" is displayed.
- Toolbar:** Includes icons for back, forward, search, and file operations.
- Header:** "co colab.research.google.com" and a "File" menu.
- Code Cell 1 (Top):**

```
1 x = np.array([100, 450, 322, 934]) #入力層  
2 W = np.array([[0.12, 0.25, 0.24, 0.95],  
3 [0.15, 0.2, 0.21, 0.75],  
4 [0.41, 0.01, 0.34, 0.21],  
5 [0.3, 0.22, 0.41, 0.93]]) #重み  
6 b = np.array([0.1, 0.2, 0.3, 0.7]) #バイアス  
7 u = np.dot(x, W) + b #前処理  
8 print(u)  
9  
# [506.31 331.05 614.48 911.75]
```
- Section Header:** "順伝播（単層・複数ユニット）"
- Code Cell 2 (Bottom):**

```
1 x = np.array([100, 450, 322, 934]) #入力層  
2 W = np.array([[0.12, 0.25, 0.24, 0.95],  
3 [0.15, 0.2, 0.21, 0.75],  
4 [0.3, 0.22, 0.41, 0.93],  
5 [0.35, 0.4, 0.35, 1],  
6 [0.3, 0.4, 0.35, 1],  
7 [0.3, 0.4, 0.35, 1],  
8 [0.3, 0.4, 0.35, 1],  
9 [0.3, 0.4, 0.35, 1],  
10 [0.3, 0.4, 0.35, 1],  
11 W1 = np.random((4,1))  
12 W2 = np.random((4,1))  
13 W3 = np.random((4,1))  
14 W4 = np.random((4,1))  
15 W5 = np.random((4,1))  
16 print(np.array([W1, W2, W3, W4, W5]))  
17  
18 # バイアス  
19 b = np.array([0.1, 0.2, 0.3])  
20 print(np.array([b1, b2, b3, b4, b5]))  
21  
22 # 入力層  
23 x = np.array([1, 0, 5, 0, 2, 0, -1, 0])
```



Google Colaboratory interface showing the execution of Python code. The code defines a function `f` that takes parameters `a`, `b`, and `c`. It calculates various mathematical operations including dot products, cross products, and matrix multiplication. The results are printed to the console.

```
def f(a, b, c):
    u = np.array([1, 2, 3])
    v = np.array([4, 5, 6])
    w = np.array([7, 8, 9])
    x = np.array([1, 0, -1])
    y = np.array([0, 1, 2])
    z = np.array([3, 4, 5])

    print("u + v =", u + v)
    print("u * v =", u * v)
    print("u * w =", u * w)
    print("u * x =", u * x)
    print("u * y =", u * y)
    print("u * z =", u * z)

    print("u * u =", u * u)
    print("v * v =", v * v)
    print("w * w =", w * w)
    print("x * x =", x * x)
    print("y * y =", y * y)
    print("z * z =", z * z)

    print("u * u * v =", u * u * v)
    print("v * v * w =", v * v * w)
    print("w * w * x =", w * w * x)
    print("x * x * y =", x * x * y)
    print("y * y * z =", y * y * z)
    print("z * z * u =", z * z * u)

    print("u * v * w =", u * v * w)
    print("v * w * x =", v * w * x)
    print("w * x * y =", w * x * y)
    print("x * y * z =", x * y * z)
    print("y * z * u =", y * z * u)
    print("z * u * v =", z * u * v)

    print("u * v * w * x =", u * v * w * x)
    print("v * w * x * y =", v * w * x * y)
    print("w * x * y * z =", w * x * y * z)
    print("x * y * z * u =", x * y * z * u)
    print("y * z * u * v =", y * z * u * v)
    print("z * u * v * w =", z * u * v * w)

    print("u * v * w * x * y =", u * v * w * x * y)
    print("v * w * x * y * z =", v * w * x * y * z)
    print("w * x * y * z * u =", w * x * y * z * u)
    print("x * y * z * u * v =", x * y * z * u * v)
    print("y * z * u * v * w =", y * z * u * v * w)

    print("u * v * w * x * y * z =", u * v * w * x * y * z)
    print("v * w * x * y * z * u =", v * w * x * y * z * u)
    print("w * x * y * z * u * v =", w * x * y * z * u * v)
    print("x * y * z * u * v * w =", x * y * z * u * v * w)

    print("u * v * w * x * y * z * u =", u * v * w * x * y * z * u)
    print("v * w * x * y * z * u * v =", v * w * x * y * z * u * v)
    print("w * x * y * z * u * v * w =", w * x * y * z * u * v * w)
    print("x * y * z * u * v * w * u =", x * y * z * u * v * w * u)
    print("y * z * u * v * w * u * v =", y * z * u * v * w * u * v)
    print("z * u * v * w * u * v * w =", z * u * v * w * u * v * w)

    print("u * v * w * x * y * z * u * v =", u * v * w * x * y * z * u * v)
    print("v * w * x * y * z * u * v * w =", v * w * x * y * z * u * v * w)
    print("w * x * y * z * u * v * w * u =", w * x * y * z * u * v * w * u)
    print("x * y * z * u * v * w * u * v =", x * y * z * u * v * w * u * v)
    print("y * z * u * v * w * u * v * w =", y * z * u * v * w * u * v * w)
    print("z * u * v * w * u * v * w * u =", z * u * v * w * u * v * w * u)

    print("u * v * w * x * y * z * u * v * w =", u * v * w * x * y * z * u * v * w)
    print("v * w * x * y * z * u * v * w * u =", v * w * x * y * z * u * v * w * u)
    print("w * x * y * z * u * v * w * u * v =", w * x * y * z * u * v * w * u * v)
    print("x * y * z * u * v * w * u * v * w =", x * y * z * u * v * w * u * v * w)
    print("y * z * u * v * w * u * v * w * u =", y * z * u * v * w * u * v * w * u)
    print("z * u * v * w * u * v * w * u * v =", z * u * v * w * u * v * w * u * v)

    print("u * v * w * x * y * z * u * v * w * u * v * w * u =", u * v * w * x * y * z * u * v * w * u * v * w * u)
    print("v * w * x * y * z * u * v * w * u * v * w * u * v =", v * w * x * y * z * u * v * w * u * v * w * u * v)
    print("w * x * y * z * u * v * w * u * v * w * u * v * w =", w * x * y * z * u * v * w * u * v * w * u * v * w)
    print("x * y * z * u * v * w * u * v * w * u * v * w * u =", x * y * z * u * v * w * u * v * w * u * v * w * u)
    print("y * z * u * v * w * u * v * w * u * v * w * u * v =", y * z * u * v * w * u * v * w * u * v * w * u * v)
    print("z * u * v * w * u * v * w * u * v * w * u * v * w =", z * u * v * w * u * v * w * u * v * w * u * v * w)
```

```
1.1_forward_propagation_after.ipynb ☆
ファイル フォルダ 組織 検索 ランタイム ツール ヘルプ 最終編集: 3月11日
コメント 共有
コード + テキスト
[x] 14 # プロセス実行
15 def forwardNetwork(x):
16     print("Input X0: ", x[0])
17     print("Input X1: ", x[1])
18     print("Input X2: ", x[2])
19     print("Input X3: ", x[3])
20
21     # 1層の入力
22     u1 = np.dot(x, W1) + b1
23     z1 = functions.relu(u1)
24
25     # 2層の入力
26     u2 = np.dot(z1, W2) + b2
27     z2 = functions.relu(u2)
28
29     # 3層の出力
30     u3 = np.dot(z2, W3) + b3
31     y = functions.softmax(u3)
32
33     print("Output Y: ", y)
34
35     return y, z1, z2
36
37 x = np.array([1, 2, 4, 3])
38 print("Input X: ", x)
39 print("Output Y: ", forwardNetwork(x))
40
```

```
1.1_forward_propagation_after.ipynb ☆
ファイル フォルダ 組織 検索 ランタイム ツール ヘルプ 最終編集: 3月11日
コメント 共有
コード + テキスト
[x] 71 x = np.random((1, 2, 4, 3)
72 print("x: ", x)
73
74 f = forwardNetwork(x)
75 v, z1, z2 = forwardNetwork(x)
76
77 v, z1, z2 = forwardNetwork(x)
78
```

\*\*\* h1 \*\*\*
[[0.45620901 0.89125593 0.87027453 0.8778844 1.109957]
 [0.16520901 0.89125593 0.45747059 0.47932323 0.80666963]
 [0.16520901 0.89125593 0.45747059 0.47932323 0.80666963]
 [0.55950401 0.89125593 0.8698757 0.4150111 0.4150111]
 [0.2177091 0.89256465 0.5272615 0.4054025 0.3890174]
 [0.57841301 0.7316023 0.84670865 0.76076956 0.755649]
 [0.21610908 0.89125593 0.8698757 0.4150111 0.4150111]
 [0.26109819 0.89125593 0.52688848 0.58079888 0.32799618]]
shape: (5, 4)

\*\*\* h2 \*\*\*
[[0.542177 0.2144609 0.2401513 0.6391972]
 [0.16520901 0.89125593 0.45747059 0.47932323 0.80666963]
 [0.16520901 0.89125593 0.45747059 0.47932323 0.80666963]
 [0.517903 0.89125593 0.7644717 0.2549249]
 [0.21610908 0.89125593 0.8698757 0.4150111 0.4150111]
 [0.57841301 0.7316023 0.84670865 0.76076956 0.755649]
 [0.21610908 0.89125593 0.8698757 0.4150111 0.4150111]
 [0.26109819 0.89125593 0.52688848 0.58079888 0.32799618]]
shape: (5, 4)

\*\*\* v \*\*\*
[[0.3120283 0.1525819 0.1942159 0.46454626 0.50154607]
 shape: (5, 4)

\*\*\* z1 \*\*\*
[[0.3120283 0.1525819 0.1942159 0.46454626 0.50154607]
 shape: (5, 4)

\*\*\* z2 \*\*\*
[[0.3120283 0.1525819 0.1942159 0.46454626 0.50154607]
 shape: (5, 4)]

```
1.1_forward_propagation_after.ipynb ☆
ファイル フォルダ 組織 検索 ランタイム ツール ヘルプ 最終編集: 3月11日
コメント 共有
コード + テキスト
[x] 100 nodes, radius: 3

```

▼ 多クラス分類 (2-3-4ネットワーク)

```
[1] 1 # モデルの構成
[2] 2 # 2-3-4ネットワーク
[3] 3 # 1以上で2-3-4-ノードの構成を3-5-6に変更してみよう
[4] 4 # モデルを2-3-4-ノードの構成で設定
[5] 5 # モデルを2-3-4-ノードの構成で設定
[6] 6 # モデルを2-3-4-ノードの構成で設定
```

```
1 # 多クラス分類 (2-3-ネットワーク)
2 # 2-2-4ネットワーク
3
4 # 1. 評価で使うノードの構成を 3-5-6 に変更してみよう
5
6 # モデルとデータフレーム
7 # ネットワーク構成
8 def forward(network):
9     print("forward ネットワークの初期化")
10
11    # ハイドレーション
12    # 各ノードの初期値を表示
13    # ネットワークの構成をサンプル生成
14
15    network = []
16
17    input_layer_size = 3
18    hidden_layer_size = 2
19    output_layer_size = 6
20
21    # ハイドレーション
22    # 各ノードの初期値を表示
23    # ネットワークの構成を表示
24    network[0] = np.random.rand(input_layer_size, hidden_layer_size)
25
26    network[1] = np.random.rand(hidden_layer_size, output_layer_size)
27    network[2] = np.random.rand(hidden_layer_size)
28
29    print("層[0]:", network[0])
30    print("層[1]:", network[1])
31    print("層[2]:", network[2])
32    print("層[3]:", network[3])
33
34
35    return network
```

```
1 # ネットワーク構成
2
3 def forward(network, x):
4
5     print("forward ネットワーク")
6     W1, b1 = network[0], network[1], network[2]
7
8     # 1. 前処理
9     # 2. データ入力
10    # 3. 関数実行
11    # 4. 活性化関数
12    # 5. 出力
13    y = functions.softmax(W1 @ x + b1)
14
15    # 出力結果
16    print("層[0]:", y)
17    print("層[1]:", y)
18    print("層[2]:", y)
19    print("層[3]:", y)
20
21    print("層[0]:", y)
22    print("層[1]:", y)
23    print("層[2]:", y)
24
25    return y
26
27 # 入力データ
28 x = np.arange(1, 2, 0.3)
29
30 # データ入力
31 d = np.zeros((0, 0, 1, 1, 0))
32
33 # ネットワークの初期化
34 network = init_network()
35
36 # ハイド
```

```
1 # ネットワーク構成
2
3 def forward(network, x):
4
5    loss = functions.cross_entropy(d, y)
6
7    if y.size == 1:
8        print("層[0]:", y)
9        print("層[1]:", y)
10       print("層[2]:", y)
11       print("層[3]:", y)
12
13       print("層[0]:", y)
14       print("層[1]:", y)
15       print("層[2]:", y)
16
17       print("層[0]:", y)
18       print("層[1]:", y)
19       print("層[2]:", y)
20
21       return loss
22
23 # ハイドレーション
24
25 # データ入力
26 x = np.array([0.493734, 0.8704020, 0.3796938, 0.5305694, 0.1184020])
27
28 # ネットワーク構成
29 network = init_network()
30
31 # ハイド
```

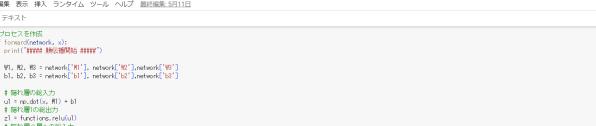
```
# 入力データ ***  
# shape: (10, 1)  
x = np.array([6.3401049, 69.5146079, 96.7456169, 91.0014957, 97.7957108],  
            [0.])  
*** 出力 ***  
y = np.array([1.111711e-03 1.4422815e-01 7.7251269e-04 9.3468071e-01  
            3.290244e-03 1.2002750e-04],  
            [0.])  
出力形状: (1, 1000000000000000)  
*** 総合評価 ***  
[1.111711e-03 1.4422815e-01 7.7251269e-04 9.3468071e-01  
            3.290244e-03 1.2002750e-04]  
shape: (1, 6)  
*** 誤差率 ***  
0.09355075586237  
shape: ()
```

▼ 回帰 (2-3-2ネットワーク)

```
1 # 定義  
2 # 2-3-2ネットワーク  
3  
4 # 1.試してみようノードの構成を 3-5-4 に変更してみよう  
5  
6 # ネットワークを定義  
7 # ネットワークを実装  
8 def init_network():  
9     print("【NET】ネットワークの初期化 【NET】")  
10    input_layer_size = 2  
11    hidden1_layer_size = 3  
12    hidden2_layer_size = 5  
13    output_layer_size = 2  
14  
15    print("【NET】")  
16    # 亂数生成器の生成  
17    # 乱数生成器の生成  
18    network["W1"] = np.random.rand(input_layer_size, hidden1_layer_size)  
19    network["b1"] = np.zeros(hidden1_layer_size) * np.random.uniform()  
20  
21    network["W2"] = np.random.rand(hidden1_layer_size, output_layer_size)  
22    network["b2"] = np.zeros(output_layer_size) * np.random.uniform()  
23  
24    print("【NET】", network["W1"])  
25    print("【NET】", network["b1"])  
26    print("【NET】", network["W2"])  
27    print("【NET】", network["b2"])  
28  
29    return network
```

```
10 # ディープ学習  
11 def forward(network, x):  
12     print("【NET】前処理 【NET】")  
13  
14     W1, b1, W2, b2 = network["W1"], network["b1"],  
15     W3, b3, W4, b4 = network["W2"], network["b2"],  
16     W5, b5, W6, b6 = network["W3"], network["b3"],  
17     W7, b7, W8, b8 = network["W4"], network["b4"]  
18  
19     # 前処理(入力)  
20     x1 = np.dot(x, W1) + b1  
21     # 非線形関数(f)  
22     z1 = f(x1) * np.exp(-z1)  
23     # 重み乗算(2)  
24     z2 = np.dot(z1, W2) + b2  
25     # 非線形関数(f)  
26     y1 = f(z2) * np.exp(-y1)  
27  
28     # 前処理(2-3-2)  
29     y2 = np.dot(y1, W3) + b3  
30     # 非線形関数(f)  
31     z3 = f(y2) * np.exp(-z3)  
32     # 重み乗算(3)  
33     z4 = np.dot(z3, W4) + b4  
34     # 非線形関数(f)  
35     y3 = f(z4) * np.exp(-y3)  
36  
37     # 前処理(3-2-1)  
38     y4 = np.dot(y3, W5) + b5  
39     # 非線形関数(f)  
40     z5 = f(y4) * np.exp(-z5)  
41     # 重み乗算(4)  
42     z6 = np.dot(z5, W6) + b6  
43     # 非線形関数(f)  
44     y5 = f(z6) * np.exp(-y5)  
45  
46     # 前処理(4-1)  
47     y6 = np.dot(y5, W7) + b7  
48     # 非線形関数(f)  
49     z7 = f(y6) * np.exp(-z7)  
50     # 重み乗算(5)  
51     z8 = np.dot(z7, W8) + b8  
52     # 非線形関数(f)  
53     y7 = f(z8) * np.exp(-y7)  
54  
55     # 入力値  
56     re_error1 = 0.1, 0.2, 0.3  
57     re_error2 = 0.1, 0.2, 0.3  
58     y, z1, z2, forwardNetwork, x = re_error1, re_error2, 0.1, 0.2, 0.3  
59     d = re_error2[0], 0.4, 0.5  
60     d1 = function_mean_maxed_error(d, y)  
61  
62     # 各点  
63     print("【NET】結果表示 【NET】")  
64     print("【NET】(x,y) = (%f,%f)" % (x, y))  
65     print("【NET】(x,z1) = (%f,%f)" % (x, z1))  
66     print("【NET】(x,z2) = (%f,%f)" % (x, z2))  
67     print("【NET】(x,y1) = (%f,%f)" % (x, y1))  
68     print("【NET】(x,y2) = (%f,%f)" % (x, y2))  
69     print("【NET】(x,y3) = (%f,%f)" % (x, y3))  
70     print("【NET】(x,y4) = (%f,%f)" % (x, y4))  
71     print("【NET】(x,y5) = (%f,%f)" % (x, y5))  
72     print("【NET】(x,y6) = (%f,%f)" % (x, y6))  
73     print("【NET】(x,y7) = (%f,%f)" % (x, y7))  
74  
75     # 各点  
76     print("【NET】(x,z3) = (%f,%f)" % (x, z3))  
77     print("【NET】(x,z4) = (%f,%f)" % (x, z4))  
78     print("【NET】(x,z5) = (%f,%f)" % (x, z5))  
79     print("【NET】(x,z6) = (%f,%f)" % (x, z6))  
80     print("【NET】(x,z7) = (%f,%f)" % (x, z7))  
81     print("【NET】(x,z8) = (%f,%f)" % (x, z8))  
82  
83     # 各点  
84     print("【NET】(x,y8) = (%f,%f)" % (x, y8))  
85     print("【NET】(x,y9) = (%f,%f)" % (x, y9))  
86     print("【NET】(x,y10) = (%f,%f)" % (x, y10))  
87  
88     # 各点  
89     print("【NET】(x,y11) = (%f,%f)" % (x, y11))  
90     print("【NET】(x,y12) = (%f,%f)" % (x, y12))  
91     print("【NET】(x,y13) = (%f,%f)" % (x, y13))  
92     print("【NET】(x,y14) = (%f,%f)" % (x, y14))  
93     print("【NET】(x,y15) = (%f,%f)" % (x, y15))  
94  
95     # 各点  
96     print("【NET】(x,y16) = (%f,%f)" % (x, y16))  
97     print("【NET】(x,y17) = (%f,%f)" % (x, y17))  
98     print("【NET】(x,y18) = (%f,%f)" % (x, y18))  
99     print("【NET】(x,y19) = (%f,%f)" % (x, y19))  
100    print("【NET】(x,y20) = (%f,%f)" % (x, y20))  
101  
102    # 各点  
103    print("【NET】(x,y21) = (%f,%f)" % (x, y21))  
104    print("【NET】(x,y22) = (%f,%f)" % (x, y22))  
105    print("【NET】(x,y23) = (%f,%f)" % (x, y23))  
106    print("【NET】(x,y24) = (%f,%f)" % (x, y24))  
107    print("【NET】(x,y25) = (%f,%f)" % (x, y25))  
108  
109    # 各点  
110    print("【NET】(x,y26) = (%f,%f)" % (x, y26))  
111    print("【NET】(x,y27) = (%f,%f)" % (x, y27))  
112    print("【NET】(x,y28) = (%f,%f)" % (x, y28))  
113    print("【NET】(x,y29) = (%f,%f)" % (x, y29))  
114    print("【NET】(x,y30) = (%f,%f)" % (x, y30))  
115  
116    # 各点  
117    print("【NET】(x,y31) = (%f,%f)" % (x, y31))  
118    print("【NET】(x,y32) = (%f,%f)" % (x, y32))  
119    print("【NET】(x,y33) = (%f,%f)" % (x, y33))  
120    print("【NET】(x,y34) = (%f,%f)" % (x, y34))  
121    print("【NET】(x,y35) = (%f,%f)" % (x, y35))  
122  
123    # 各点  
124    print("【NET】(x,y36) = (%f,%f)" % (x, y36))  
125    print("【NET】(x,y37) = (%f,%f)" % (x, y37))  
126    print("【NET】(x,y38) = (%f,%f)" % (x, y38))  
127    print("【NET】(x,y39) = (%f,%f)" % (x, y39))  
128    print("【NET】(x,y40) = (%f,%f)" % (x, y40))  
129  
130    # 各点  
131    print("【NET】(x,y41) = (%f,%f)" % (x, y41))  
132    print("【NET】(x,y42) = (%f,%f)" % (x, y42))  
133    print("【NET】(x,y43) = (%f,%f)" % (x, y43))  
134    print("【NET】(x,y44) = (%f,%f)" % (x, y44))  
135    print("【NET】(x,y45) = (%f,%f)" % (x, y45))  
136  
137    # 各点  
138    print("【NET】(x,y46) = (%f,%f)" % (x, y46))  
139    print("【NET】(x,y47) = (%f,%f)" % (x, y47))  
140    print("【NET】(x,y48) = (%f,%f)" % (x, y48))  
141    print("【NET】(x,y49) = (%f,%f)" % (x, y49))  
142    print("【NET】(x,y50) = (%f,%f)" % (x, y50))  
143  
144    # 各点  
145    print("【NET】(x,y51) = (%f,%f)" % (x, y51))  
146    print("【NET】(x,y52) = (%f,%f)" % (x, y52))  
147    print("【NET】(x,y53) = (%f,%f)" % (x, y53))  
148    print("【NET】(x,y54) = (%f,%f)" % (x, y54))  
149    print("【NET】(x,y55) = (%f,%f)" % (x, y55))  
150  
151    # 各点  
152    print("【NET】(x,y56) = (%f,%f)" % (x, y56))  
153    print("【NET】(x,y57) = (%f,%f)" % (x, y57))  
154    print("【NET】(x,y58) = (%f,%f)" % (x, y58))  
155    print("【NET】(x,y59) = (%f,%f)" % (x, y59))  
156    print("【NET】(x,y60) = (%f,%f)" % (x, y60))  
157  
158    # 各点  
159    print("【NET】(x,y61) = (%f,%f)" % (x, y61))  
160    print("【NET】(x,y62) = (%f,%f)" % (x, y62))  
161    print("【NET】(x,y63) = (%f,%f)" % (x, y63))  
162    print("【NET】(x,y64) = (%f,%f)" % (x, y64))  
163    print("【NET】(x,y65) = (%f,%f)" % (x, y65))  
164  
165    # 各点  
166    print("【NET】(x,y66) = (%f,%f)" % (x, y66))  
167    print("【NET】(x,y67) = (%f,%f)" % (x, y67))  
168    print("【NET】(x,y68) = (%f,%f)" % (x, y68))  
169    print("【NET】(x,y69) = (%f,%f)" % (x, y69))  
170    print("【NET】(x,y70) = (%f,%f)" % (x, y70))  
171  
172    # 各点  
173    print("【NET】(x,y71) = (%f,%f)" % (x, y71))  
174    print("【NET】(x,y72) = (%f,%f)" % (x, y72))  
175    print("【NET】(x,y73) = (%f,%f)" % (x, y73))  
176    print("【NET】(x,y74) = (%f,%f)" % (x, y74))  
177    print("【NET】(x,y75) = (%f,%f)" % (x, y75))  
178  
179    # 各点  
180    print("【NET】(x,y76) = (%f,%f)" % (x, y76))  
181    print("【NET】(x,y77) = (%f,%f)" % (x, y77))  
182    print("【NET】(x,y78) = (%f,%f)" % (x, y78))  
183    print("【NET】(x,y79) = (%f,%f)" % (x, y79))  
184    print("【NET】(x,y80) = (%f,%f)" % (x, y80))  
185  
186    # 各点  
187    print("【NET】(x,y81) = (%f,%f)" % (x, y81))  
188    print("【NET】(x,y82) = (%f,%f)" % (x, y82))  
189    print("【NET】(x,y83) = (%f,%f)" % (x, y83))  
190    print("【NET】(x,y84) = (%f,%f)" % (x, y84))  
191    print("【NET】(x,y85) = (%f,%f)" % (x, y85))  
192  
193    # 各点  
194    print("【NET】(x,y86) = (%f,%f)" % (x, y86))  
195    print("【NET】(x,y87) = (%f,%f)" % (x, y87))  
196    print("【NET】(x,y88) = (%f,%f)" % (x, y88))  
197    print("【NET】(x,y89) = (%f,%f)" % (x, y89))  
198    print("【NET】(x,y90) = (%f,%f)" % (x, y90))  
199  
200    # 各点  
201    print("【NET】(x,y91) = (%f,%f)" % (x, y91))  
202    print("【NET】(x,y92) = (%f,%f)" % (x, y92))  
203    print("【NET】(x,y93) = (%f,%f)" % (x, y93))  
204    print("【NET】(x,y94) = (%f,%f)" % (x, y94))  
205    print("【NET】(x,y95) = (%f,%f)" % (x, y95))  
206  
207    # 各点  
208    print("【NET】(x,y96) = (%f,%f)" % (x, y96))  
209    print("【NET】(x,y97) = (%f,%f)" % (x, y97))  
210    print("【NET】(x,y98) = (%f,%f)" % (x, y98))  
211    print("【NET】(x,y99) = (%f,%f)" % (x, y99))  
212    print("【NET】(x,y100) = (%f,%f)" % (x, y100))  
213  
214    # 各点  
215    print("【NET】(x,y101) = (%f,%f)" % (x, y101))  
216    print("【NET】(x,y102) = (%f,%f)" % (x, y102))  
217    print("【NET】(x,y103) = (%f,%f)" % (x, y103))  
218    print("【NET】(x,y104) = (%f,%f)" % (x, y104))  
219    print("【NET】(x,y105) = (%f,%f)" % (x, y105))  
220  
221    # 各点  
222    print("【NET】(x,y106) = (%f,%f)" % (x, y106))  
223    print("【NET】(x,y107) = (%f,%f)" % (x, y107))  
224    print("【NET】(x,y108) = (%f,%f)" % (x, y108))  
225    print("【NET】(x,y109) = (%f,%f)" % (x, y109))  
226    print("【NET】(x,y110) = (%f,%f)" % (x, y110))  
227  
228    # 各点  
229    print("【NET】(x,y111) = (%f,%f)" % (x, y111))  
230    print("【NET】(x,y112) = (%f,%f)" % (x, y112))  
231    print("【NET】(x,y113) = (%f,%f)" % (x, y113))  
232    print("【NET】(x,y114) = (%f,%f)" % (x, y114))  
233    print("【NET】(x,y115) = (%f,%f)" % (x, y115))  
234  
235    # 各点  
236    print("【NET】(x,y116) = (%f,%f)" % (x, y116))  
237    print("【NET】(x,y117) = (%f,%f)" % (x, y117))  
238    print("【NET】(x,y118) = (%f,%f)" % (x, y118))  
239    print("【NET】(x,y119) = (%f,%f)" % (x, y119))  
240    print("【NET】(x,y120) = (%f,%f)" % (x, y120))  
241  
242    # 各点  
243    print("【NET】(x,y121) = (%f,%f)" % (x, y121))  
244    print("【NET】(x,y122) = (%f,%f)" % (x, y122))  
245    print("【NET】(x,y123) = (%f,%f)" % (x, y123))  
246    print("【NET】(x,y124) = (%f,%f)" % (x, y124))  
247    print("【NET】(x,y125) = (%f,%f)" % (x, y125))  
248  
249    # 各点  
250    print("【NET】(x,y126) = (%f,%f)" % (x, y126))  
251    print("【NET】(x,y127) = (%f,%f)" % (x, y127))  
252    print("【NET】(x,y128) = (%f,%f)" % (x, y128))  
253    print("【NET】(x,y129) = (%f,%f)" % (x, y129))  
254    print("【NET】(x,y130) = (%f,%f)" % (x, y130))  
255  
256    # 各点  
257    print("【NET】(x,y131) = (%f,%f)" % (x, y131))  
258    print("【NET】(x,y132) = (%f,%f)" % (x, y132))  
259    print("【NET】(x,y133) = (%f,%f)" % (x, y133))  
260    print("【NET】(x,y134) = (%f,%f)" % (x, y134))  
261    print("【NET】(x,y135) = (%f,%f)" % (x, y135))  
262  
263    # 各点  
264    print("【NET】(x,y136) = (%f,%f)" % (x, y136))  
265    print("【NET】(x,y137) = (%f,%f)" % (x, y137))  
266    print("【NET】(x,y138) = (%f,%f)" % (x, y138))  
267    print("【NET】(x,y139) = (%f,%f)" % (x, y139))  
268    print("【NET】(x,y140) = (%f,%f)" % (x, y140))  
269  
270    # 各点  
271    print("【NET】(x,y141) = (%f,%f)" % (x, y141))  
272    print("【NET】(x,y142) = (%f,%f)" % (x, y142))  
273    print("【NET】(x,y143) = (%f,%f)" % (x, y143))  
274    print("【NET】(x,y144) = (%f,%f)" % (x, y144))  
275    print("【NET】(x,y145) = (%f,%f)" % (x, y145))  
276  
277    # 各点  
278    print("【NET】(x,y146) = (%f,%f)" % (x, y146))  
279    print("【NET】(x,y147) = (%f,%f)" % (x, y147))  
280    print("【NET】(x,y148) = (%f,%f)" % (x, y148))  
281    print("【NET】(x,y149) = (%f,%f)" % (x, y149))  
282    print("【NET】(x,y150) = (%f,%f)" % (x, y150))  
283  
284    # 各点  
285    print("【NET】(x,y151) = (%f,%f)" % (x, y151))  
286    print("【NET】(x,y152) = (%f,%f)" % (x, y152))  
287    print("【NET】(x,y153) = (%f,%f)" % (x, y153))  
288    print("【NET】(x,y154) = (%f,%f)" % (x, y154))  
289    print("【NET】(x,y155) = (%f,%f)" % (x, y155))  
290  
291    # 各点  
292    print("【NET】(x,y156) = (%f,%f)" % (x, y156))  
293    print("【NET】(x,y157) = (%f,%f)" % (x, y157))  
294    print("【NET】(x,y158) = (%f,%f)" % (x, y158))  
295    print("【NET】(x,y159) = (%f,%f)" % (x, y159))  
296    print("【NET】(x,y160) = (%f,%f)" % (x, y160))  
297  
298    # 各点  
299    print("【NET】(x,y161) = (%f,%f)" % (x, y161))  
300    print("【NET】(x,y162) = (%f,%f)" % (x, y162))  
301    print("【NET】(x,y163) = (%f,%f)" % (x, y163))  
302    print("【NET】(x,y164) = (%f,%f)" % (x, y164))  
303    print("【NET】(x,y165) = (%f,%f)" % (x, y165))  
304  
305    # 各点  
306    print("【NET】(x,y166) = (%f,%f)" % (x, y166))  
307    print("【NET】(x,y167) = (%f,%f)" % (x, y167))  
308    print("【NET】(x,y168) = (%f,%f)" % (x, y168))  
309    print("【NET】(x,y169) = (%f,%f)" % (x, y169))  
310    print("【NET】(x,y170) = (%f,%f)" % (x, y170))  
311  
312    # 各点  
313    print("【NET】(x,y171) = (%f,%f)" % (x, y171))  
314    print("【NET】(x,y172) = (%f,%f)" % (x, y172))  
315    print("【NET】(x,y173) = (%f,%f)" % (x, y173))  
316    print("【NET】(x,y174) = (%f,%f)" % (x, y174))  
317    print("【NET】(x,y175) = (%f,%f)" % (x, y175))  
318  
319    # 各点  
320    print("【NET】(x,y176) = (%f,%f)" % (x, y176))  
321    print("【NET】(x,y177) = (%f,%f)" % (x, y177))  
322    print("【NET】(x,y178) = (%f,%f)" % (x, y178))  
323    print("【NET】(x,y179) = (%f,%f)" % (x, y179))  
324    print("【NET】(x,y180) = (%f,%f)" % (x, y180))  
325  
326    # 各点  
327    print("【NET】(x,y181) = (%f,%f)" % (x, y181))  
328    print("【NET】(x,y182) = (%f,%f)" % (x, y182))  
329    print("【NET】(x,y183) = (%f,%f)" % (x, y183))  
330    print("【NET】(x,y184) = (%f,%f)" % (x, y184))  
331    print("【NET】(x,y185) = (%f,%f)" % (x, y185))  
332  
333    # 各点  
334    print("【NET】(x,y186) = (%f,%f)" % (x, y186))  
335    print("【NET】(x,y187) = (%f,%f)" % (x, y187))  
336    print("【NET】(x,y188) = (%f,%f)" % (x, y188))  
337    print("【NET】(x,y189) = (%f,%f)" % (x, y189))  
338    print("【NET】(x,y190) = (%f,%f)" % (x, y190))  
339  
340    # 各点  
341    print("【NET】(x,y191) = (%f,%f)" % (x, y191))  
342    print("【NET】(x,y192) = (%f,%f)" % (x, y192))  
343    print("【NET】(x,y193) = (%f,%f)" % (x, y193))  
344    print("【NET】(x,y194) = (%f,%f)" % (x, y194))  
345    print("【NET】(x,y195) = (%f,%f)" % (x, y195))  
346  
347    # 各点  
348    print("【NET】(x,y196) = (%f,%f)" % (x, y196))  
349    print("【NET】(x,y197) = (%f,%f)" % (x, y197))  
350    print("【NET】(x,y198) = (%f,%f)" % (x, y198))  
351    print("【NET】(x,y199) = (%f,%f)" % (x, y199))  
352    print("【NET】(x,y200) = (%f,%f)" % (x, y200))  
353  
354    # 各点  
355    print("【NET】(x,y201) = (%f,%f)" % (x, y201))  
356    print("【NET】(x,y202) = (%f,%f)" % (x, y202))  
357    print("【NET】(x,y203) = (%f,%f)" % (x, y203))  
358    print("【NET】(x,y204) = (%f,%f)" % (x, y204))  
359    print("【NET】(x,y205) = (%f,%f)" % (x, y205))  
360  
361    # 各点  
362    print("【NET】(x,y206) = (%f,%f)" % (x, y206))  
363    print("【NET】(x,y207) = (%f,%f)" % (x, y207))  
364    print("【NET】(x,y208) = (%f,%f)" % (x, y208))  
365    print("【NET】(x,y209) = (%f,%f)" % (x, y209))  
366    print("【NET】(x,y210) = (%f,%f)" % (x, y210))  
367  
368    # 各点  
369    print("【NET】(x,y211) = (%f,%f)" % (x, y211))  
370    print("【NET】(x,y212) = (%f,%f)" % (x, y212))  
371    print("【NET】(x,y213) = (%f,%f)" % (x, y213))  
372    print("【NET】(x,y214) = (%f,%f)" % (x, y214))  
373    print("【NET】(x,y215) = (%f,%f)" % (x, y215))  
374  
375    # 各点  
376    print("【NET】(x,y216) = (%f,%f)" % (x, y216))  
377    print("【NET】(x,y217) = (%f,%f)" % (x, y217))  
378    print("【NET】(x,y218) = (%f,%f)" % (x, y218))  
379    print("【NET】(x,y219) = (%f,%f)" % (x, y219))  
380    print("【NET】(x,y220) = (%f,%f)" % (x, y220))  
381  
382    # 各点  
383    print("【NET】(x,y221) = (%f,%f)" % (x, y221))  
384    print("【NET】(x,y222) = (%f,%f)" % (x, y222))  
385    print("【NET】(x,y223) = (%f,%f)" % (x, y223))  
386    print("【NET】(x,y224) = (%f,%f)" % (x, y224))  
387    print("【NET】(x,y225) = (%f,%f)" % (x, y225))  
388  
389    # 各点  
390    print("【NET】(x,y226) = (%f,%f)" % (x, y226))  
391    print("【NET】(x,y227) = (%f,%f)" % (x, y227))  
392    print("【NET】(x,y228) = (%f,%f)" % (x, y228))  
393    print("【NET】(x,y229) = (%f,%f)" % (x, y229))  
394    print("【NET】(x,y230) = (%f,%f)" % (x, y230))  
395  
396    # 各点  
397    print("【NET】(x,y231) = (%f,%f)" % (x, y231))  
398    print("【NET】(x,y232) = (%f,%f)" % (x, y232))  
399    print("【NET】(x,y233) = (%f,%f)" % (x, y233))  
400    print("【NET】(x,y234) = (%f,%f)" % (x, y234))  
401    print("【NET】(x,y235) = (%f,%f)" % (x, y235))  
402  
403    # 各点  
404    print("【NET】(x,y236) = (%f,%f)" % (x, y236))  
405    print("【NET】(x,y237) = (%f,%f)" % (x, y237))  
406    print("【NET】(x,y238) = (%f,%f)" % (x, y238))  
407    print("【NET】(x,y239) = (%f,%f)" % (x, y239))  
408    print("【NET】(x,y240) = (%f,%f)" % (x, y240))  
409  
410    # 各点  
411    print("【NET】(x,y241) = (%f,%f)" % (x, y241))  
412    print("【NET】(x,y242) = (%f,%f)" % (x, y242))  
413    print("【NET】(x,y243) = (%f,%f)" % (x, y243))  
414    print("【NET】(x,y244) = (%f,%f)" % (x, y244))  
415    print("【NET】(x,y245) = (%f,%f)" % (x, y245))  
416  
417    # 各点  
418    print("【NET】(x,y246) = (%f,%f)" % (x, y246))  
419    print("【NET】(x,y247) = (%f,%f)" % (x, y247))  
420    print("【NET】(x,y248) = (%f,%f)" % (x, y248))  
421    print("【NET】(x,y249) = (%f,%f)" % (x, y249))  
422    print("【NET】(x,y250) = (%f,%f)" % (x, y250))  
423  
424    # 各点  
425    print("【NET】(x,y251) = (%f,%f)" % (x, y251))  
426    print("【NET】(x,y252) = (%f,%f)" % (x, y252))  
427    print("【NET】(x,y253) = (%f,%f)" % (x, y253))  
428    print("【NET】(x,y254) = (%f,%f)" % (x, y254))  
429    print("【NET】
```

```
① forward_propagation_after.pyrbn 実行  
ファイル フォルダ 表示 帰入 ランタイム ツール ヘルプ 最近の履歴 3月1日  
コメント 共有 セーブ 編集  
☰  
検索 2種分類 (2-3-1ネットワーク)  
[x]  
② 1 ネットワーク  
2 2-3-1ネットワーク  
③ 4 亂数生成器の実装を 5-10-20-1 に変更してみよう  
5  
6 # ランダムな初期値を生成する  
7 フィードフォワードネットワーク  
8 def init(*shape):  
9     print("初期化 フィードフォワードネットワーク")  
10    network = []  
11    network.append([1])  
12    network[0] + np.random((1, shape[0]))  
13    (0.1, 0.3, 0.5, 0.7), (0.2, 0.4, 0.6, 0.8), (0.3, 0.5, 0.7, 0.9),  
14    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
15    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
16    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
17    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
18    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
19    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
20    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
21    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
22    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
23    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
24    (0.1, 0.3, 0.5, 0.7), (0.3, 0.5, 0.7, 0.9), (0.1, 0.3, 0.5, 0.7),  
25  
26    return network  
27  
28  
29 # フィードフォワード  
30 def forward(network, x):  
31     for layer in range(1, len(network) - 1):  
32         print("層 " + str(layer) + " の実装")  
33  
34     W1, b1, Z1 = network[0], network[1], network[0].dotnetwork[1]  
35     b2, Z2 = network[1], network[2].dotnetwork[1]  
36  
37     # 順序確認
```



```
# 1.1_forward_propagation_after_pyrbn

# ファイル 設定 表示挿入 ランタイム ソール ヘルプ 最終編集: 5月1日

# コード + テキスト 検索

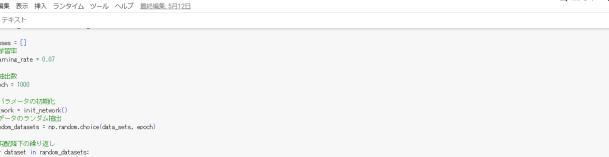
# 29 F フォワードプロパゲーション
# 30 def forward_propagation(x, y, z1, z2):
# 31     print("forward propagation!!!")
# 32
# 33     W1, b1, Z1 = initialize(1, "relu", network["W1"], network["b1"])
# 34     b2, Z2 = initialize("tanh", "relu", network["b2"], network["Z2"])
# 35
# 36     # 前の層の出力
# 37     a1 = np.dot(x, W1) + b1
# 38     z1 = np.tanh(a1)
# 39     z1 = functions.relu(z1)
# 40
# 41     a2 = np.dot(z1, W2) + b2
# 42     z2 = np.tanh(a2)
# 43     z2 = functions.relu(z2)
# 44
# 45     y_hat = np.dot(z2, W3) + b3
# 46     z3 = functions.sigmoid(z3)
# 47
# 48     print("前回の重み", w)
# 49     print("前回のバイアス", b)
# 50
# 51     print("出力", y_hat)
# 52     print("目標値", y)
# 53
# 54     return y_hat, z1
# 55
# 56 # 入力データ
# 57 x = np.array([1., 2., 3., 4., 5.])
# 58
# 59 # ベクトル化
# 60 d = np.reshape(x, (1, 5))
# 61
# 62 # ネットワーク構成
# 63 network = initialize(1, "relu", network["W1"], network["b1"])
# 64
# 65 # リセット
# 66 loss = float('inf')
# 67 print("初期loss", loss)
# 68 print("初期重み", network["W1"])
# 69 print("初期バイアス", network["b1"])
# 70
# 71 while loss > 0.0001:
```

```
# Google Colab での動作を調べる
# ドライブマウント
# フォルダとファイルと同じフォルダへの移動
# Googleドライブのマイドライブを基にDNN_code/DNN_code_colab_day1 フォルダを置くことを仮定しています。必要に応じて、パスを変更してください。
# ! # Google Colab での動作を調べる
# ! import sys
# ! import os
# ! if 'BM_CODE' in os.environ:
# !     BM_CODE = True
# ! else:
# !     BM_CODE = False
# ! # Google Drive のマウント
# ! if BM_CODE:
# !     from google.colab import drive
# !     drive.mount('/content/drive')
# !     print("Google Drive mounted at /content/drive")
# !     os.chdir("/content/drive")
# ! else:
# !     print("Not mounted at /content/drive")
# ! # sys.path の設定
# ! # ! import sys
# ! # ! sys.path.append(os.getcwd()) # 複数ディレクトリのファイルをインポートするための設定
```

```
# importと関数定義
# ! import numpy as np
# ! import tensorflow as tf
# ! import tensorflow.contrib.slim as slim
# ! def print_vector(tensor, n=10):
# !     print("[" + " " * 4 + tensor[0] + " " * 4)
# !     for i in range(1, n):
# !         print(" " * 4 + tensor[i] + " " * 4)
# !     print(" " * 4 + tensor[-1] + " " * 4)
# !     print("]")
# ! # TensorFlow
# ! def init_network():
# !     # 亂数生成器の初期化
# !     np.random.seed(42)
# !     # ネットワーク構造の初期化
# !     network = []
# !     # 入力層
# !     network.append(np.random.rand(2, nodesIn))
# !     # 隠れ層
# !     network.append(np.random.rand(2, nodesH))
# !     # 出力層
# !     network.append(np.random.rand(2, nodesO))
# !     # 各層のノード数を記録
# !     nodesIn = len(network[0])
# !     nodesH = len(network[1])
# !     nodesO = len(network[2])
# !     # 各層のノード数を記録
# !     print("Input layer: " + str(nodesIn))
# !     print("Hidden layer: " + str(nodesH))
# !     print("Output layer: " + str(nodesO))
```

```
# 前処理
# ! def forwardNetwork(x2):
# !     z1 = f(x2, W1, b1, nodesIn, nodesH)
# !     v1, k1 = network["W1"], network["b1"]
# !     v2, k2 = network["W2"], network["b2"]
# !     u1 = np.dot(x2, k1) + b1
# !     z1 = f(u1, w1)
# !     # 活性化関数
# !     z1 = f(z1, w2)
# !     v3, k3 = network["W3"], network["b3"]
# !     u2 = np.dot(z1, k3) + b3
# !     z2 = f(u2, w3)
# !     # 活性化関数
# !     z2 = f(z2, w4)
# !     v4, k4 = network["W4"], network["b4"]
# !     u3 = np.dot(z2, k4) + b4
# !     z3 = f(u3, w5)
# !     # 活性化関数
# !     z3 = f(z3, w6)
# !     v5, k5 = network["W5"], network["b5"]
# !     u4 = np.dot(z3, k5) + b5
# !     z4 = f(u4, w7)
# !     # 活性化関数
# !     z4 = f(z4, w8)
# !     v6, k6 = network["W6"], network["b6"]
# !     u5 = np.dot(z4, k6) + b6
# !     z5 = f(u5, w9)
# !     # 活性化関数
# !     z5 = f(z5, w10)
# !     v7, k7 = network["W7"], network["b7"]
# !     u6 = np.dot(z5, k7) + b7
# !     z6 = f(u6, w11)
# !     # 活性化関数
# !     z6 = f(z6, w12)
# !     v8, k8 = network["W8"], network["b8"]
# !     u7 = np.dot(z6, k8) + b8
# !     z7 = f(u7, w13)
# !     # 活性化関数
# !     z7 = f(z7, w14)
# !     v9, k9 = network["W9"], network["b9"]
# !     u8 = np.dot(z7, k9) + b9
# !     z8 = f(u8, w15)
# !     # 活性化関数
# !     z8 = f(z8, w16)
# !     v10, k10 = network["W10"], network["b10"]
# !     u9 = np.dot(z8, k10) + b10
# !     z9 = f(u9, w17)
# !     # 活性化関数
# !     z9 = f(z9, w18)
# !     v11, k11 = network["W11"], network["b11"]
# !     u10 = np.dot(z9, k11) + b11
# !     z10 = f(u10, w19)
# !     # 活性化関数
# !     z10 = f(z10, w20)
# !     v12, k12 = network["W12"], network["b12"]
# !     u11 = np.dot(z10, k12) + b12
# !     z11 = f(u11, w21)
# !     # 活性化関数
# !     z11 = f(z11, w22)
# !     v13, k13 = network["W13"], network["b13"]
# !     u12 = np.dot(z11, k13) + b13
# !     z12 = f(u12, w23)
# !     # 活性化関数
# !     z12 = f(z12, w24)
# !     v14, k14 = network["W14"], network["b14"]
# !     u13 = np.dot(z12, k14) + b14
# !     z13 = f(u13, w25)
# !     # 活性化関数
# !     z13 = f(z13, w26)
# !     v15, k15 = network["W15"], network["b15"]
# !     u14 = np.dot(z13, k15) + b15
# !     z14 = f(u14, w27)
# !     # 活性化関数
# !     z14 = f(z14, w28)
# !     v16, k16 = network["W16"], network["b16"]
# !     u15 = np.dot(z14, k16) + b16
# !     z15 = f(u15, w29)
# !     # 活性化関数
# !     z15 = f(z15, w30)
# !     v17, k17 = network["W17"], network["b17"]
# !     u16 = np.dot(z15, k17) + b17
# !     z16 = f(u16, w31)
# !     # 活性化関数
# !     z16 = f(z16, w32)
# !     v18, k18 = network["W18"], network["b18"]
# !     u17 = np.dot(z16, k18) + b18
# !     z17 = f(u17, w33)
# !     # 活性化関数
# !     z17 = f(z17, w34)
# !     v19, k19 = network["W19"], network["b19"]
# !     u18 = np.dot(z17, k19) + b19
# !     z18 = f(u18, w35)
# !     # 活性化関数
# !     z18 = f(z18, w36)
# !     v20, k20 = network["W20"], network["b20"]
# !     u19 = np.dot(z18, k20) + b20
# !     z19 = f(u19, w37)
# !     # 活性化関数
# !     z19 = f(z19, w38)
# !     v21, k21 = network["W21"], network["b21"]
# !     u20 = np.dot(z19, k21) + b21
# !     z20 = f(u20, w39)
# !     # 活性化関数
# !     z20 = f(z20, w40)
# !     v22, k22 = network["W22"], network["b22"]
# !     u21 = np.dot(z20, k22) + b22
# !     z21 = f(u21, w41)
# !     # 活性化関数
# !     z21 = f(z21, w42)
# !     v23, k23 = network["W23"], network["b23"]
# !     u22 = np.dot(z21, k23) + b23
# !     z22 = f(u22, w43)
# !     # 活性化関数
# !     z22 = f(z22, w44)
# !     v24, k24 = network["W24"], network["b24"]
# !     u23 = np.dot(z22, k24) + b24
# !     z23 = f(u23, w45)
# !     # 活性化関数
# !     z23 = f(z23, w46)
# !     v25, k25 = network["W25"], network["b25"]
# !     u24 = np.dot(z23, k25) + b25
# !     z24 = f(u24, w47)
# !     # 活性化関数
# !     z24 = f(z24, w48)
# !     v26, k26 = network["W26"], network["b26"]
# !     u25 = np.dot(z24, k26) + b26
# !     z25 = f(u25, w49)
# !     # 活性化関数
# !     z25 = f(z25, w50)
# !     v27, k27 = network["W27"], network["b27"]
# !     u26 = np.dot(z25, k27) + b27
# !     z26 = f(u26, w51)
# !     # 活性化関数
# !     z26 = f(z26, w52)
# !     v28, k28 = network["W28"], network["b28"]
# !     u27 = np.dot(z26, k28) + b28
# !     z27 = f(u27, w53)
# !     # 活性化関数
# !     z27 = f(z27, w54)
# !     v29, k29 = network["W29"], network["b29"]
# !     u28 = np.dot(z27, k29) + b29
# !     z28 = f(u28, w55)
# !     # 活性化関数
# !     z28 = f(z28, w56)
# !     v30, k30 = network["W30"], network["b30"]
# !     u29 = np.dot(z28, k30) + b30
# !     z29 = f(u29, w57)
# !     # 活性化関数
# !     z29 = f(z29, w58)
# !     v31, k31 = network["W31"], network["b31"]
# !     u30 = np.dot(z29, k31) + b31
# !     z30 = f(u30, w59)
# !     # 活性化関数
# !     z30 = f(z30, w60)
# !     v32, k32 = network["W32"], network["b32"]
# !     u31 = np.dot(z30, k32) + b32
# !     z31 = f(u31, w61)
# !     # 活性化関数
# !     z31 = f(z31, w62)
# !     v33, k33 = network["W33"], network["b33"]
# !     u32 = np.dot(z31, k33) + b33
# !     z32 = f(u32, w63)
# !     # 活性化関数
# !     z32 = f(z32, w64)
# !     v34, k34 = network["W34"], network["b34"]
# !     u33 = np.dot(z32, k34) + b34
# !     z33 = f(u33, w65)
# !     # 活性化関数
# !     z33 = f(z33, w66)
# !     v35, k35 = network["W35"], network["b35"]
# !     u34 = np.dot(z33, k35) + b35
# !     z34 = f(u34, w67)
# !     # 活性化関数
# !     z34 = f(z34, w68)
# !     v36, k36 = network["W36"], network["b36"]
# !     u35 = np.dot(z34, k36) + b36
# !     z35 = f(u35, w69)
# !     # 活性化関数
# !     z35 = f(z35, w70)
# !     v37, k37 = network["W37"], network["b37"]
# !     u36 = np.dot(z35, k37) + b37
# !     z36 = f(u36, w71)
# !     # 活性化関数
# !     z36 = f(z36, w72)
# !     v38, k38 = network["W38"], network["b38"]
# !     u37 = np.dot(z36, k38) + b38
# !     z37 = f(u37, w73)
# !     # 活性化関数
# !     z37 = f(z37, w74)
# !     v39, k39 = network["W39"], network["b39"]
# !     u38 = np.dot(z37, k39) + b39
# !     z38 = f(u38, w75)
# !     # 活性化関数
# !     z38 = f(z38, w76)
# !     v40, k40 = network["W40"], network["b40"]
# !     u39 = np.dot(z38, k40) + b40
# !     z39 = f(u39, w77)
# !     # 活性化関数
# !     z39 = f(z39, w78)
# !     v41, k41 = network["W41"], network["b41"]
# !     u40 = np.dot(z39, k41) + b41
# !     z40 = f(u40, w79)
# !     # 活性化関数
# !     z40 = f(z40, w80)
# !     v42, k42 = network["W42"], network["b42"]
# !     u41 = np.dot(z40, k42) + b42
# !     z41 = f(u41, w81)
# !     # 活性化関数
# !     z41 = f(z41, w82)
# !     v43, k43 = network["W43"], network["b43"]
# !     u42 = np.dot(z41, k43) + b43
# !     z42 = f(u42, w83)
# !     # 活性化関数
# !     z42 = f(z42, w84)
# !     v44, k44 = network["W44"], network["b44"]
# !     u43 = np.dot(z42, k44) + b44
# !     z43 = f(u43, w85)
# !     # 活性化関数
# !     z43 = f(z43, w86)
# !     v45, k45 = network["W45"], network["b45"]
# !     u44 = np.dot(z43, k45) + b45
# !     z44 = f(u44, w87)
# !     # 活性化関数
# !     z44 = f(z44, w88)
# !     v46, k46 = network["W46"], network["b46"]
# !     u45 = np.dot(z44, k46) + b46
# !     z45 = f(u45, w89)
# !     # 活性化関数
# !     z45 = f(z45, w90)
# !     v47, k47 = network["W47"], network["b47"]
# !     u46 = np.dot(z45, k47) + b47
# !     z46 = f(u46, w91)
# !     # 活性化関数
# !     z46 = f(z46, w92)
# !     v48, k48 = network["W48"], network["b48"]
# !     u47 = np.dot(z46, k48) + b48
# !     z47 = f(u47, w93)
# !     # 活性化関数
# !     z47 = f(z47, w94)
# !     v49, k49 = network["W49"], network["b49"]
# !     u48 = np.dot(z47, k49) + b49
# !     z48 = f(u48, w95)
# !     # 活性化関数
# !     z48 = f(z48, w96)
# !     v50, k50 = network["W50"], network["b50"]
# !     u49 = np.dot(z48, k50) + b50
# !     z49 = f(u49, w97)
# !     # 活性化関数
# !     z49 = f(z49, w98)
# !     v51, k51 = network["W51"], network["b51"]
# !     u50 = np.dot(z49, k51) + b51
# !     z50 = f(u50, w99)
# !     # 活性化関数
# !     z50 = f(z50, w100)
# !     v52, k52 = network["W52"], network["b52"]
# !     u51 = np.dot(z50, k52) + b52
# !     z51 = f(u51, w101)
# !     # 活性化関数
# !     z51 = f(z51, w102)
# !     v53, k53 = network["W53"], network["b53"]
# !     u52 = np.dot(z51, k53) + b53
# !     z52 = f(u52, w103)
# !     # 活性化関数
# !     z52 = f(z52, w104)
# !     v54, k54 = network["W54"], network["b54"]
# !     u53 = np.dot(z52, k54) + b54
# !     z53 = f(u53, w105)
# !     # 活性化関数
# !     z53 = f(z53, w106)
# !     v55, k55 = network["W55"], network["b55"]
# !     u54 = np.dot(z53, k55) + b55
# !     z54 = f(u54, w107)
# !     # 活性化関数
# !     z54 = f(z54, w108)
# !     v56, k56 = network["W56"], network["b56"]
# !     u55 = np.dot(z54, k56) + b56
# !     z55 = f(u55, w109)
# !     # 活性化関数
# !     z55 = f(z55, w110)
# !     v57, k57 = network["W57"], network["b57"]
# !     u56 = np.dot(z55, k57) + b57
# !     z56 = f(u56, w111)
# !     # 活性化関数
# !     z56 = f(z56, w112)
# !     v58, k58 = network["W58"], network["b58"]
# !     u57 = np.dot(z56, k58) + b58
# !     z57 = f(u57, w113)
# !     # 活性化関数
# !     z57 = f(z57, w114)
# !     v59, k59 = network["W59"], network["b59"]
# !     u58 = np.dot(z57, k59) + b58
# !     z58 = f(u58, w115)
# !     # 活性化関数
# !     z58 = f(z58, w116)
# !     v60, k60 = network["W60"], network["b60"]
# !     u59 = np.dot(z58, k60) + b59
# !     z59 = f(u59, w117)
# !     # 活性化関数
# !     z59 = f(z59, w118)
# !     v61, k61 = network["W61"], network["b61"]
# !     u60 = np.dot(z59, k61) + b60
# !     z60 = f(u60, w119)
# !     # 活性化関数
# !     z60 = f(z60, w120)
# !     v62, k62 = network["W62"], network["b62"]
# !     u61 = np.dot(z60, k62) + b61
# !     z61 = f(u61, w121)
# !     # 活性化関数
# !     z61 = f(z61, w122)
# !     v63, k63 = network["W63"], network["b63"]
# !     u62 = np.dot(z61, k63) + b62
# !     z62 = f(u62, w123)
# !     # 活性化関数
# !     z62 = f(z62, w124)
# !     v64, k64 = network["W64"], network["b64"]
# !     u63 = np.dot(z62, k64) + b63
# !     z63 = f(u63, w125)
# !     # 活性化関数
# !     z63 = f(z63, w126)
# !     v65, k65 = network["W65"], network["b65"]
# !     u64 = np.dot(z63, k65) + b64
# !     z64 = f(u64, w127)
# !     # 活性化関数
# !     z64 = f(z64, w128)
# !     v66, k66 = network["W66"], network["b66"]
# !     u65 = np.dot(z64, k66) + b65
# !     z65 = f(u65, w129)
# !     # 活性化関数
# !     z65 = f(z65, w130)
# !     v67, k67 = network["W67"], network["b67"]
# !     u66 = np.dot(z65, k67) + b66
# !     z66 = f(u66, w131)
# !     # 活性化関数
# !     z66 = f(z66, w132)
# !     v68, k68 = network["W68"], network["b68"]
# !     u67 = np.dot(z66, k68) + b67
# !     z67 = f(u67, w133)
# !     # 活性化関数
# !     z67 = f(z67, w134)
# !     v69, k69 = network["W69"], network["b69"]
# !     u68 = np.dot(z67, k69) + b68
# !     z68 = f(u68, w135)
# !     # 活性化関数
# !     z68 = f(z68, w136)
# !     v70, k70 = network["W70"], network["b70"]
# !     u69 = np.dot(z68, k70) + b69
# !     z69 = f(u69, w137)
# !     # 活性化関数
# !     z69 = f(z69, w138)
# !     v71, k71 = network["W71"], network["b71"]
# !     u70 = np.dot(z69, k71) + b70
# !     z70 = f(u70, w139)
# !     # 活性化関数
# !     z70 = f(z70, w140)
# !     v72, k72 = network["W72"], network["b72"]
# !     u71 = np.dot(z70, k72) + b71
# !     z71 = f(u71, w141)
# !     # 活性化関数
# !     z71 = f(z71, w142)
# !     v73, k73 = network["W73"], network["b73"]
# !     u72 = np.dot(z71, k73) + b72
# !     z72 = f(u72, w143)
# !     # 活性化関数
# !     z72 = f(z72, w144)
# !     v74, k74 = network["W74"], network["b74"]
# !     u73 = np.dot(z72, k74) + b73
# !     z73 = f(u73, w145)
# !     # 活性化関数
# !     z73 = f(z73, w146)
# !     v75, k75 = network["W75"], network["b75"]
# !     u74 = np.dot(z73, k75) + b74
# !     z74 = f(u74, w147)
# !     # 活性化関数
# !     z74 = f(z74, w148)
# !     v76, k76 = network["W76"], network["b76"]
# !     u75 = np.dot(z74, k76) + b75
# !     z75 = f(u75, w149)
# !     # 活性化関数
# !     z75 = f(z75, w150)
# !     v77, k77 = network["W77"], network["b77"]
# !     u76 = np.dot(z75, k77) + b76
# !     z76 = f(u76, w151)
# !     # 活性化関数
# !     z76 = f(z76, w152)
# !     v78, k78 = network["W78"], network["b78"]
# !     u77 = np.dot(z76, k78) + b77
# !     z77 = f(u77, w153)
# !     # 活性化関数
# !     z77 = f(z77, w154)
# !     v79, k79 = network["W79"], network["b79"]
# !     u78 = np.dot(z77, k79) + b78
# !     z78 = f(u78, w155)
# !     # 活性化関数
# !     z78 = f(z78, w156)
# !     v80, k80 = network["W80"], network["b80"]
# !     u79 = np.dot(z78, k80) + b79
# !     z79 = f(u79, w157)
# !     # 活性化関数
# !     z79 = f(z79, w158)
# !     v81, k81 = network["W81"], network["b81"]
# !     u80 = np.dot(z79, k81) + b80
# !     z80 = f(u80, w159)
# !     # 活性化関数
# !     z80 = f(z80, w160)
# !     v82, k82 = network["W82"], network["b82"]
# !     u81 = np.dot(z80, k82) + b81
# !     z81 = f(u81, w161)
# !     # 活性化関数
# !     z81 = f(z81, w162)
# !     v83, k83 = network["W83"], network["b83"]
# !     u82 = np.dot(z81, k83) + b82
# !     z82 = f(u82, w163)
# !     # 活性化関数
# !     z82 = f(z82, w164)
# !     v84, k84 = network["W84"], network["b84"]
# !     u83 = np.dot(z82, k84) + b83
# !     z83 = f(u83, w165)
# !     # 活性化関数
# !     z83 = f(z83, w166)
# !     v85, k85 = network["W85"], network["b85"]
# !     u84 = np.dot(z83, k85) + b84
# !     z84 = f(u84, w167)
# !     # 活性化関数
# !     z84 = f(z84, w168)
# !     v86, k86 = network["W86"], network["b86"]
# !     u85 = np.dot(z84, k86) + b85
# !     z85 = f(u85, w169)
# !     # 活性化関数
# !     z85 = f(z85, w170)
# !     v87, k87 = network["W87"], network["b87"]
# !     u86 = np.dot(z85, k87) + b86
# !     z86 = f(u86, w171)
# !     # 活性化関数
# !     z86 = f(z86, w172)
# !     v88, k88 = network["W88"], network["b88"]
# !     u87 = np.dot(z86, k88) + b87
# !     z87 = f(u87, w173)
# !     # 活性化関数
# !     z87 = f(z87, w174)
# !     v89, k89 = network["W89"], network["b89"]
# !     u88 = np.dot(z87, k89) + b88
# !     z88 = f(u88, w175)
# !     # 活性化関数
# !     z88 = f(z88, w176)
# !     v90, k90 = network["W90"], network["b90"]
# !     u89 = np.dot(z88, k90) + b89
# !     z89 = f(u89, w177)
# !     # 活性化関数
# !     z89 = f(z89, w178)
# !     v91, k91 = network["W91"], network["b91"]
# !     u90 = np.dot(z89, k91) + b90
# !     z90 = f(u90, w179)
# !     # 活性化関数
# !     z90 = f(z90, w180)
# !     v92, k92 = network["W92"], network["b92"]
# !     u91 = np.dot(z90, k92) + b91
# !     z91 = f(u91, w181)
# !     # 活性化関数
# !     z91 = f(z91, w182)
# !     v93, k93 = network["W93"], network["b93"]
# !     u92 = np.dot(z91, k93) + b92
# !     z92 = f(u92, w183)
# !     # 活性化関数
# !     z92 = f(z92, w184)
# !     v94, k94 = network["W94"], network["b94"]
# !     u93 = np.dot(z92, k94) + b93
# !     z93 = f(u93, w185)
# !     # 活性化関数
# !     z93 = f(z93, w186)
# !     v95, k95 = network["W95"], network["b95"]
# !     u94 = np.dot(z93, k95) + b94
# !     z94 = f(u94, w187)
# !     # 活性化関数
# !     z94 = f(z94, w188)
# !     v96, k96 = network["W96"], network["b96"]
# !     u95 = np.dot(z94, k96) + b95
# !     z95 = f(u95, w189)
# !     # 活性化関数
# !     z95 = f(z95, w190)
# !     v97, k97 = network["W97"], network["b97"]
# !     u96 = np.dot(z95, k97) + b96
# !     z96 = f(u96, w191)
# !     # 活性化関数
# !     z96 = f(z96, w192)
# !     v98, k98 = network["W98"], network["b98"]
# !     u97 = np.dot(z96, k98) + b97
# !     z97 = f(u97, w193)
# !     # 活性化関数
# !     z97 = f(z97, w194)
# !     v99, k99 = network["W99"], network["b99"]
# !     u98 = np.dot(z97, k99) + b98
# !     z98 = f(u98, w195)
# !     # 活性化関数
# !     z98 = f(z98, w196)
# !     v100, k100 = network["W100"], network["b100"]
# !     u99 = np.dot(z98, k100) + b99
# !     z99 = f(u99, w197)
# !     # 活性化関数
# !     z99 = f(z99, w198)
# !     v101, k101 = network["W101"], network["b101"]
# !     u100 = np.dot(z99, k101) + b100
# !     z100 = f(u100, w199)
# !     # 活性化関数
# !     z100 = f(z100, w200)
```

```
○ 1_stochastic_gradient_descent.py  
□ colab.research.google.com □  
コメント 共有 ハンズオン 印刷範囲: 5/12/21  
コード ナビゲート  
1 # 本問題のデータ  
2 #delta1 = np.random(delta2, M, T) = functions.d_relu(z1)  
3  
4 #print("delta1: " + str(delta1))  
5  
6 #print("delta2: " + str(delta2))  
7  
8 #print("delta3: " + str(delta3))  
9  
10 #print("delta4: " + str(delta4))  
11  
12 #print("delta5: " + str(delta5))  
13  
14 #print("delta6: " + str(delta6))  
15  
16 #print("delta7: " + str(delta7))  
17  
18 #print("delta8: " + str(delta8))  
19  
20 #print("delta9: " + str(delta9))  
21  
22 #print("delta10: " + str(delta10))  
23  
24  
25 #print("delta11: " + str(delta11))  
26  
27  
28 #print("delta12: " + str(delta12))  
29  
30  
31 #print("delta13: " + str(delta13))  
32  
33  
34 #print("delta14: " + str(delta14))  
35  
36  
37 #print("delta15: " + str(delta15))  
38  
39  
40 #print("delta16: " + str(delta16))  
41  
42  
43 #print("delta17: " + str(delta17))  
44  
45  
46 #print("delta18: " + str(delta18))  
47  
48  
49 #print("delta19: " + str(delta19))  
50  
51  
52 #print("delta20: " + str(delta20))  
53  
54  
55 #print("delta21: " + str(delta21))  
56  
57  
58 #print("delta22: " + str(delta22))  
59  
60  
61 #print("delta23: " + str(delta23))  
62  
63  
64 #print("delta24: " + str(delta24))  
65  
66  
67 #print("delta25: " + str(delta25))  
68  
69  
70 #print("delta26: " + str(delta26))  
71  
72  
73 #print("delta27: " + str(delta27))  
74  
75 #print("delta28: " + str(delta28))  
76  
77  
78 #print("delta29: " + str(delta29))  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023
```



The screenshot shows a Google Colab interface with the following details:

- Title:** 3\_stochastic\_gradient\_descent.pyrb ☆
- File Path:** ファイル > 設定 > 鍵入、ランタイム > ハードウェア
- Execution Date:** 最終編集: 5月12日
- Code Content:**

```
100 # 定数
101 losses = []
102 # ランダム初期化
103 learning_rate = 0.07
104 x = 10
105 y = 5 * x
106 epoch = 1000
107
108 # リストにデータを追加
109 data_set = []
110 # データのランダム化
111 random.shuffle(data_set, random.choice(data_sets, epoch))
112
113 # ランダム化されたデータを表示
114 print("ランダム化されたデータ:")
115 for d in data_set:
116     print(d)
117 # データを学習用と検証用に分離
118 train_size = int(len(data_set) * 0.8)
119 train, test = data_set[:train_size], data_set[train_size:]
120
121 # ニューラルネットワークの初期化
122 w = np.random.randn(1, 1)
123 b = np.random.randn(1)
124 loss_fn = mean_squared_error
125
126 print("=====初期状態=====")
127 batch_size = max(1, epoch)
128
129 for i in range(epoch):
130     # ランダムな順序でデータを取得
131     for d, y in random.sample(test, batch_size):
132         # ニューラルネットワークの出力
133         y_hat = w * d + b
134         # バックプロパゲーション
135         grad_w, grad_b = backprop(d, y, y_hat)
136         # パラメータの更新
137         w -= learning_rate * grad_w
138         b -= learning_rate * grad_b
139         # ネットワークの損失を計算
140         losses.append(loss_fn(y_hat, y))
141
142 # 損失関数の可視化
143 plt.plot(range(len(losses)), losses, '.')
144 plt.title('学習過程')
145 plt.xlabel('エポック')
146 plt.ylabel('損失関数')
147 plt.show()
```
- Output:** A line plot titled "学習過程" (Learning Process) showing the loss function over 1000 epochs. The x-axis is labeled "エポック" (Epoch) and the y-axis is labeled "損失関数" (Loss Function). The plot shows a downward trend from approximately 2.5 to 0.5.

