

# 通信理論に特化した深層学習

## 第7回ゼミ資料

### TensorFlow 2 入門

豊橋技術科学大学  
電気・電子情報工学系  
准教授 竹内啓悟

## 簡単な学習例

### MNISTデータ

0～9の手書き数字の答え付画像データ(28 × 28ピクセル、ピクセル値0～255)

訓練用データ60000個、評価用データ10000個

### 学習目標

未知の画像データの正答率が最大になるように、2層全結合型順伝播ネットワークの重みとバイアスを学習したい。

### 学習方法

- 訓練用データ中の10000個を学習時の検証用データとして使用
- 損失関数に交差エントロピーを使用
- 中間層にReLU、出力層にSoft Max関数を使用
- Adamによる誤差逆伝播法
- エポック数20、ミニバッチサイズ100

### 参考URL

<https://www.tensorflow.org/tutorials/quickstart/beginner>

# Pythonコーディング規約

可読性を高めるために、Pythonではコーディング規約が定められている。

## 主な規約

- インデントは半角スペース4つ
- 最上位層の関数やクラス間の空行は、2行
- クラス内のメソッド間の空行は、1行
- 「,」の後に単語が続く場合は半角スペース(英語の標準記法)
- {=, +, -}の前後には半角スペースを入れる
- \*や/の前後にはスペースを入れない方がよい
- クラス名は、単語の頭文字を大文字にし、単語同士を直接つなぐ
- 関数(メソッド)名や変数は小文字の単語をアンダースコアでつなぐ
- 定数はすべて大文字の単語をアンダースコアでつなぐ

<https://pep8-ja.readthedocs.io/ja/latest/>

## ソースコード例

以降のページに示すコードを単一ファイル`***.py`に順に記述し、以下のコマンドを実行せよ。

```
python3 ***.py
```

## TensorFlowのインポート

Pythonでは以下のように記述する。

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

import tensorflow as tf
```

Cで言うインクルードと同じ。

最初のインポートは、警告メッセージを非表示にするための記述

## 学習結果を出力する関数plot\_weights

```
def plot_weights(weights):  
    T = int(len(weights)/2)  
    #層の数Tを計算  
    with open('weight.txt', mode = 'w') as f:  
        for t in range(T):  
            print('W' + str(t + 1), file = f)  
            print(weights[2*t], file = f)  
            print('b' + str(t + 1), file = f)  
            print(weights[2*t + 1], file = f)
```

**入力** 後述するmodel.get\_weightsメソッドの出力

**機能** T層ニューラルネットワークに対して、入力層から出力層に向かって重みとバイアスの学習結果(W1, b1), ..., (WT, bT)をファイルweight.txtに書き出す。

Pythonでは、Cにおける{}ではなく、インデント(半角スペース4つ)で各命令の区切り位置を指定する。

## 学習曲線を出力する関数plot\_history

```
def plot_history(training_history):  
    history_dict = training_history.history  
    loss_his = history_dict['loss']  
    val_loss_his = history_dict['val_loss']  
    acc_his = history_dict['accuracy']  
    val_acc_his = history_dict['val_accuracy']
```

```
    with open('data.txt', mode = 'w') as f:  
        for i in range(len(loss_his)):  
            print(i + 1, loss_his[i], val_loss_his[i], acc_his[i], val_acc_his[i], file = f)
```

Keras 3では大文字が小文字に変換されるので注意

評価関数BERを使用した場合、辞書のkeyはberに変換される

**入力** 後述するmodel.fitメソッドが出力するクラスtraining\_history

**機能** 外部ソフトを使ってグラフを作成するためのデータをdata.txtファイルに書き込む。各列は以下の通り。

エポック数, 損失(訓練用), 損失(検証用), 正答率(訓練用), 正答率(検証用)

## MNISTデータの取り込みとデータの前処理

```
mnist = tf.keras.datasets.mnist
#WebからMNISTデータをダウンロード

(x_train, y_train), (x_test, y_test) = mnist.load_data()
#(訓練用画像, 答え), (評価用画像, 答え)
x_train, x_test = x_train/255.0, x_test/255.0
#ピクセル値を0~255から、区間[0, 1]の値に変換

x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
#訓練用データ60000個の最初の10000個を検証用を利用
```

[m:n]はm番目からn-1番目までの要素を指定する。

## ネットワークの構築と学習方法の設定

```
model = tf.keras.models.Sequential([  
    #入力 $28 \times 28 (=784)$ の2次元データ  
    tf.keras.Input(shape = (28, 28)),  
    #2次元データを1次元データに変換  
    tf.keras.layers.Flatten(),  
    #128個のReLUを持つ全結合型の中間層  
    tf.keras.layers.Dense(128, activation = 'relu'),  
    #0~9に対応する10個のノード(Soft Max関数を使用)を持つ出力層  
    tf.keras.layers.Dense(10, activation = 'softmax')])  
  
model.compile(  
    optimizer = 'adam',  
    loss = 'sparse_categorical_crossentropy',  
    metrics = ['accuracy']  
)  
#Adamによる誤差逆伝播、交差エントロピー誤差関数、  
#性能評価は正答率
```

上記のように、()内では改行してコードを見やすくして良い。



## 学習の実行と学習結果の表示

```
BATCH_SIZE = 100
```

```
#デフォルトは32だが、計算時間の都合で100に設定
```

```
#訓練用、検証用、評価用データサイズのすべてを割り切る値が良い
```

```
training_history = model.fit(  
    partial_x_train, partial_y_train,  
    epochs = 20, batch_size = BATCH_SIZE,  
    validation_data = (x_val, y_val))
```

```
#エポック数を20にして学習を実行
```

```
plot_history(training_history)
```

```
#学習曲線データを出力
```

```
weights = model.get_weights()
```

```
plot_weights(weights)
```

```
#学習後の重みを出力
```

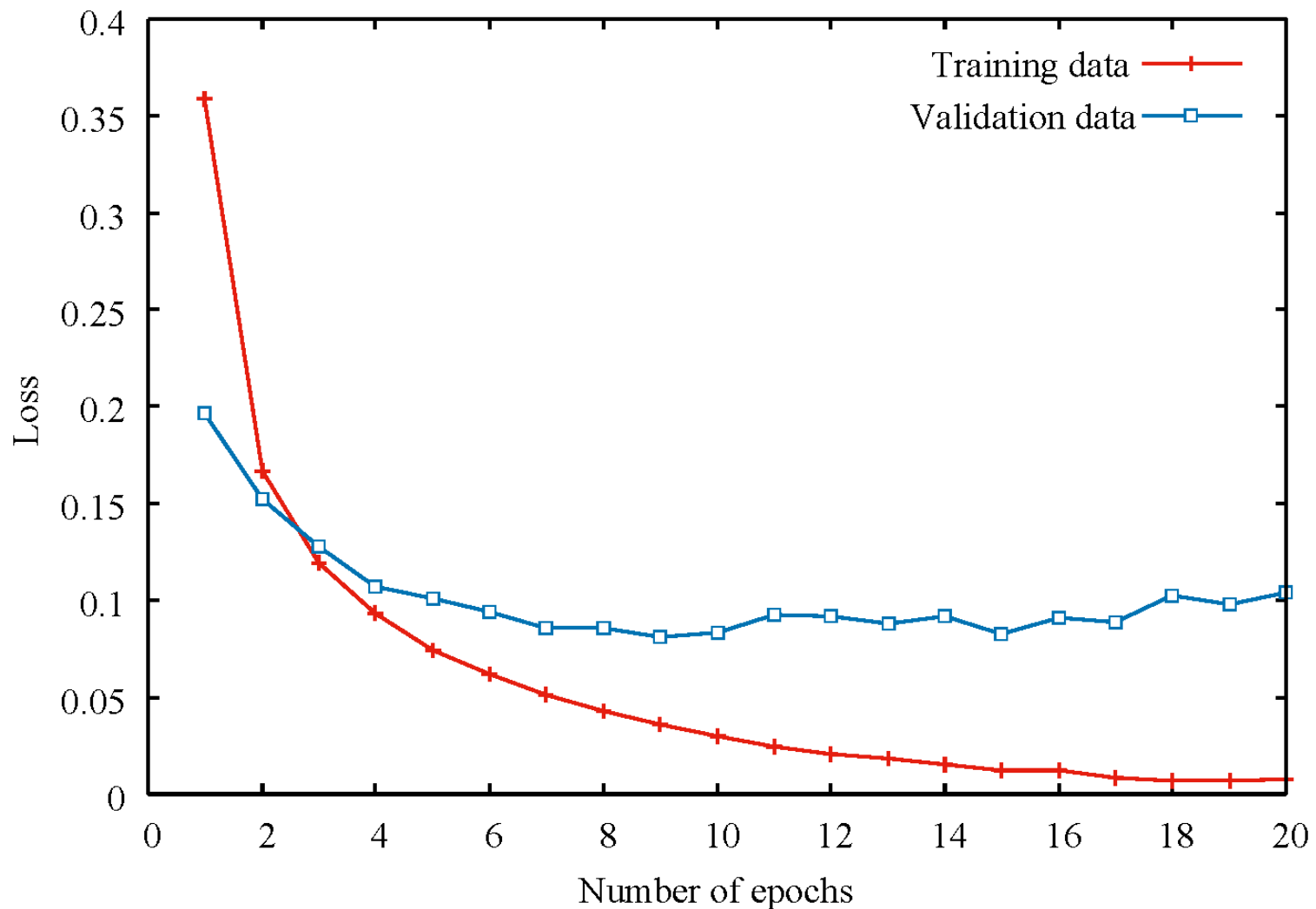
## 評価結果の出力

```
test_loss, test_acc = model.evaluate(  
    x_test, y_test,  
    batch_size = BATCH_SIZE)  
  
print(test_loss, test_acc)  
#評価用データによる損失と正答率を出力
```

並列計算に関わる問題で、本来必要ないはずの評価でも、  
`batch_size`を指定できる。

一度に処理する評価用データの数を表し、  
学習時に使用した`batch_size`とは無関係

## 学習曲線



検証用データ(評価用ではない)にとって、エポック7程度で学習を打ち切った方が良さそう。

## 正答率の結果

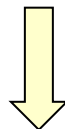
エポック数	訓練用データ	検証用データ	評価用データ
2	95.2%	95.5%	95.9%
7	98.5%	97.3%	97.4%
20	99.9%	97.6%	97.5%

学習結果は施行ごとに若干異なることに注意

### 考察

検証用と評価用データは同じ正答率のふるまいをしている。

エポック数を増やしても、評価用データの正答率は改善しない。



検証用データの正答率を使って、**評価用データを一切使うことなく**、適切なエポック数を設定できる。