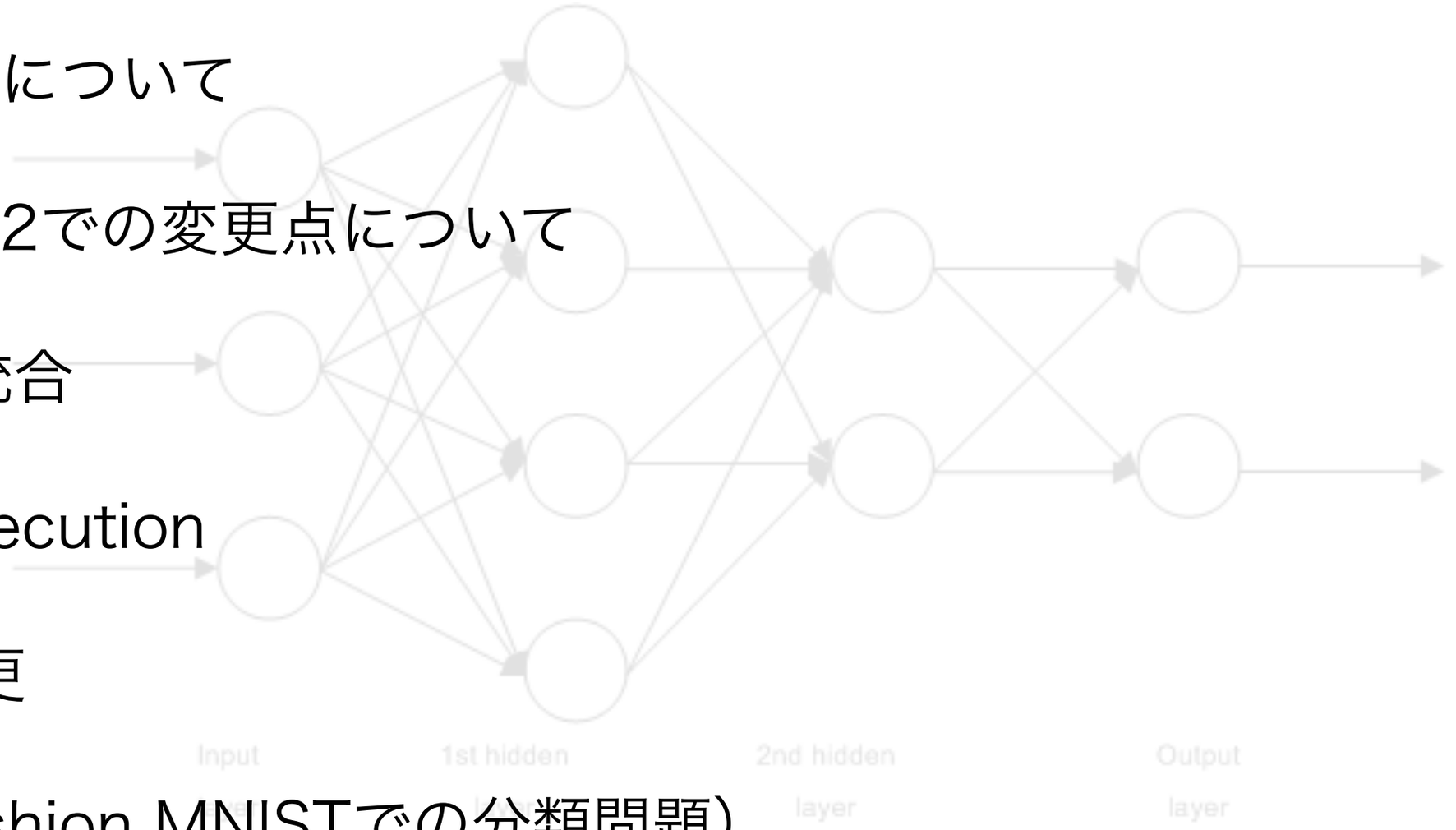
TensorFlow 2 入門

1からの変更部分とコードの書き方について

小西優祐（株式会社アカデメイア）

概要

- TensorFlowについて
- TensorFlow2での変更点について
  - Kerasの統合
  - Eager execution
  - その他変更
- 実装例（Fashion MNISTでの分類問題）
  - Fashion MNISTについて
  - コードサンプル



TensorFlowについて(1/3)

概要

- Googleが開発した機械学習、ディープラーニングのライブラリ
- Apache License 2.0で公開されているオープンソースソフトウェア
- 2015年11月の公開以来，様々なプロジェクトで用いられており，4100万以上のダウンロード，貢献者は1800人以上.
  (2019年3月時点)
- 用途
  - 顔認識，音声認識，画像検索，自動翻訳 etc.

TensorFlowについて(2/3)

対応環境，言語

- 対応環境：Windows, Mac, Linux
  Android, iOS（TensorFlow Lite）
  JavaScript（TensorFlow.js）

- コア部分はPython, C++, CUDAで記述されており，以下の言語のインターフェースを持つ.
  - Python
  - Java
  - C/C++
  - Go

TensorFlowについて(3/3)
導入方法

- pipで導入
  ```
  $ pip install tensorflow==2.1.0
  ```
  （デフォルトがpython2の人はpip3でインストール）

- インストール確認
  コンソールで”python”と打ち込んでインタプリタ環境に入り，以下のように入力
  ```
  >>> import tensorflow
  >>> tensorflow.__version__
  ```
  表示されるバージョンが2.0以降である事を確認
  ```
  '2.1.0'
  ```

TensorFlow2での変更点

Kerasの統合

- Kerasについて
  - KerasはPythonで書かれたニューラルネットワークのオープンソースライブラリで，TensorFlow, Theano等の上部で動作する．ニューラルネットワークのモデルが簡潔に書ける事に重点が置かれている．
- TensorFlow2において，ライブラリにKerasが統合され，ニューラルネットワークの実装が簡単になった．

TensorFlow2での変更点

Eager execution

- 従来のTensorFlowでは学習時に，モデルの入出力をplaceholderとして定義し，session.run()を実行する必要があった.
- TensorFlow2では,
  `outputs = f(input)`
  のようなよりシンプルな（Pythonの標準的なコードに近い）記述でモデルの学習を実行する事が出来る.

実装例（Fashion MNISTでの分類問題）

Fashion MNISTについて

- Fashion MNISTはMNISTを置き換える事を目的に作られた,
  60,000の訓練セットと10,000のテストセットからなる,
  Zalando（https://jobs.zalando.com/en/）が作成した服，靴
  などの画像のデータセット.

- MNIST（0〜9の手書き数字の分類）があまりに単純すぎるた
  め，その置き換えを目的に作成された.
  MNISTと同じ28 x 28の解像度で，10のクラスに分類される.

実装例（Fashion MNISTでの分類問題）

Fashion MNISTの中身

28 x 28のグレースケールの画像

対応したラベル

　上記だと，9: Ankle boot

ラベル	記述
0	T-shirt//top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

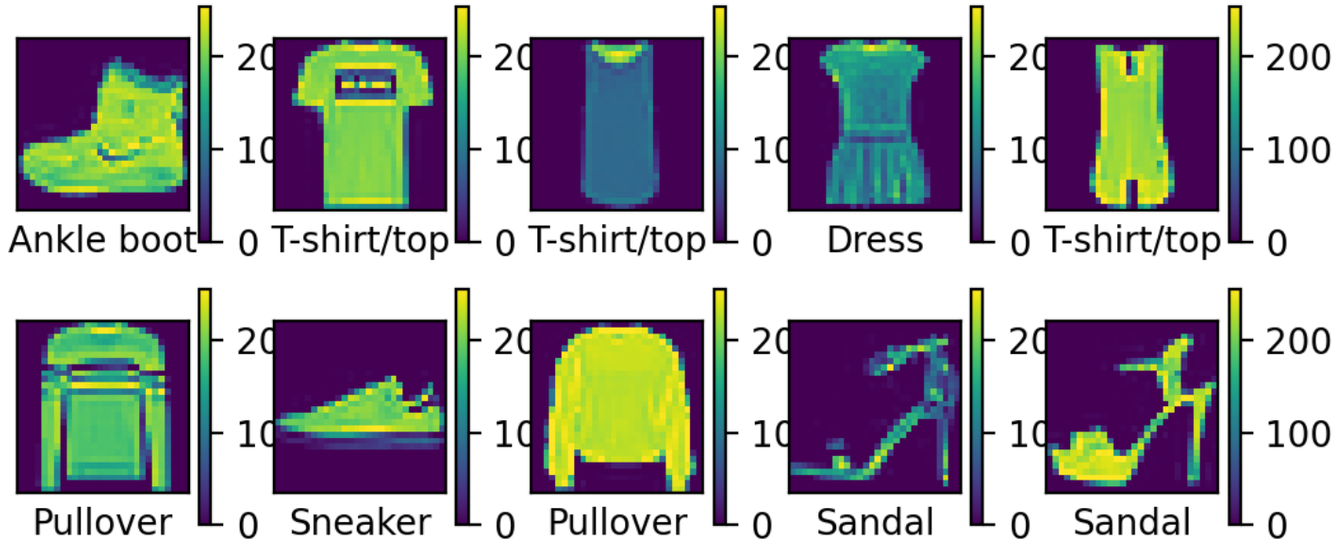
実装例（Fashion MNISTでの分類問題）

Fashion MNISTのデータロード

fmnist.pyを用いてFashion MNISTのデータをダウンロードし，サンプルの絵を表示する．以下の部分でFashion MNISTのデータを取得．（すでにダウンロードしている場合はそちらを用いる）

```
fashion_mnist = keras.datasets.fashion_mnist

(train_images, train_labels), (test_images,
test_labels) = fashion_mnist.load_data()
```



実装例（Fashion MNISTでの分類問題）

学習：前処理

学習は，train.pyで実行する.
まず，学習の前にデータに前処理を施す.
（学習時と判定時で同じ処理をする必要があるのに注意）

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

ここでは，0–255の整数値を取っているのを0.0–1.0の実数値に変換している.

実装例（Fashion MNISTでの分類問題）

学習：モデルの設定

モデルは，以下の部分で定義されている．

```
model = keras.Sequential([
        keras.layers.Flatten(input_shape=(28, 28)),
        # 入力を28x28=784次元のベクトルに変換

        keras.layers.Dense(128, activation='relu'),
        # 中間層，128次元，reluで活性化

        keras.layers.Dense(10, activation='softmax')
        # 出力層，10次元（=判定），softmaxで活性化

])
```

実装例（Fashion MNISTでの分類問題）

学習：中間層について

```
keras.layers.Dense(128, activation='relu')
```

上記では，入力が784次元，出力が128次元のベクトルとなる.
入力をx, 活性化前の出力をyとして以下のような式で表される.

$$y = W \cdot x + b$$

ここで，Wは128行784列の行列，bはバイアスと呼ばれる128次元のベクトル.
（ベクトルを列ベクトルとした場合）
学習時にはこのWとbを最適化する.

上記に活性化関数ReLUをかけたものが最終的な出力となる.

$$ReLU(x) = max(0, x)$$

実装例（Fashion MNISTでの分類問題）

学習：出力層について

```
keras.layers.Dense(10, activation='softmax')
```

上記では，入力が128次元，出力が10次元のベクトルとなる.
活性化前までは，中間層と同じ行列-ベクトル演算
活性化関数は，ここではsoftmaxを持ちている.
各分類を0-1の「確率」で表すため.

ソフトマックスは右の式で与えられる.

$$y_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

cf. Boltzmann分布
（統計力学）

$$p_i = \frac{e^{-E_i/T}}{\sum_{j=1}^{N} e^{-E_j/T}}$$

実装例（Fashion MNISTでの分類問題）

学習：モデルのコンパイル

モデルの学習に関する設定は以下の行で行われている.

```
model.compile(optimizer='adam',
                    # Adamオプティマイザーを設定
              loss='sparse_categorical_crossentropy',
              # Loss関数として交差エントロピーを用いる
              # これを最小化するように学習を実行する
              metrics=['accuracy'])
              # 評価関数としてaccuracyを用いる
              #（学習には使われない）
```

実装例（Fashion MNISTでの分類問題）

学習：Loss関数について

TensorFlowでは，Loss関数を最小化するようにモデルパラメータの更新を行う.

Loss関数として，ここでは多クラス交差エントロピーと呼ばれるものを用いている.

$$L = \sum_t \sum_c y_c^t \log y'^t_c$$

上記で，LがLoss関数
y_c^tがt番目サンプルの正答データのカテゴリーcの値
（y_c^tは，ラベルで与えられるカテゴリーだけ1で
　他0になるようなone-hot vector）
y'^t_cがt番目サンプルのモデル予想のカテゴリーcの値

実装例（Fashion MNISTでの分類問題）

学習：最適化手法について

ここでは最適化手法としてAdamが用いられている.
詳細は，Kingma and Ba (2014)，arXiv:1412.6980を参照.
https://arxiv.org/abs/1412.6980

Loss関数の勾配の1次モーメントと2次モーメントを用いて値を更新する事で，解が振動せず，早く収束する事を意図した手法.
機械学習で非常によく用いられる.

他の手法
SGD, RMSprop, AdaGrad etc.

実装例（Fashion MNISTでの分類問題）

学習：評価関数について

学習時に計算され，モデルの性能評価に用いられる
評価関数を指定.
これは，学習そのものには用いられない.

ここではaccuracyが用いられる.

実装例（Fashion MNISTでの分類問題）

学習：学習

以下の箇所で学習を実行している.

model.fit(train_images, train_labels, epochs=5)

上記で，１つ目の引数に学習用データ，2つ目の引数に学習用の正答ラベル．epochsとして学習のエポック数を与えている.

エポック数：全データ1通り学習するだけのステップ数を1エポックと定義する.

実装例（Fashion MNISTでの分類問題）

学習：学習済モデルの保存

学習済みのモデルは，HDF5形式のファイルとして保存可能.

```
model.save('clf_fmnist.h5')
```

上記部分で，学習済みモデルをclf_fmnist.h5ファイルに保存.
このファイルを読み込む事で，学習済みモデルを用いた判定が可能となる.

実装例（Fashion MNISTでの分類問題）

学習：学習済モデルの保存

学習済みのモデルは，HDF5形式のファイルとして保存可能.

```
model.save('clf_fmnist.h5')
```

上記部分で，学習済みモデルをclf_fmnist.h5ファイルに保存.
このファイルを読み込む事で，学習済みモデルを用いた判定が可能となる.

実装例（Fashion MNISTでの分類問題）

学習：train.pyの実行

train.pyを実行すると，コンソールに学習時のlossやaccuracyが出力され，
学習後に学習済みファイルclf_fmnist.h5が生成される．
最後にテストデータの0番目の画像と判定結果が表示される．

…

Train on 60000 samples

Epoch 1/5

60000/60000 [==============================] - 5s 77us/sample - loss: 0.5000 - accuracy: 0.8247

…

10000/10000 - 1s - loss: 0.3622 - accuracy: 0.8707

Test accuracy: 0.8707

…

実装例（Fashion MNISTでの分類問題）

判定：前処理，モデルのロード

判定データについても学習時と同様の前処理を行う必要がある.

```
train_images = train_images / 255.0
test_images = test_images / 255.0
```

以下の部分で学習時に生成したclf_fmnist.h5を読み込む.

```
model = keras.models.load_model('clf_fmnist.h5')
```

実装例（Fashion MNISTでの分類問題）

判定：判定

判定は以下の部分で実行している.

```
predictions = model.predict(test_images)
```

上記predictionsに各モデルの判定結果が格納される.
test_images[i]の判定結果がpredictions[i]となっている.

各predictions[i]は以下のようなベクトルになっている.

```
[1.7727913e-05 1.7757287e-07 8.1428175e-07 1.4998831e-06
1.1081520e-06 2.0753149e-02 2.8975912e-05 9.3254028e-03
5.1301431e-05 9.6981984e-01]
```

上記で，j番目の要素がカテゴリーjの確率に対応している.

実装例（Fashion MNISTでの分類問題）

判定：1サンプルに対しての実行

model.predictでは，複数サンプルについて判定を実行する．
1サンプルだけについて判定したい場合（predict_single.pyの例）は，以下のようにして次元を拡張する必要がある．

```
img = (np.expand_dims(img,0))
```

参考サイト(1/2)

- tensorflow.org
  TensorFlowの公式サイト．特に以下を参照している.

  - はじめてのニューラルネットワーク：分類問題の初歩
    https://www.tensorflow.org/tutorials/keras/classification

  - モデルの保存と復元
    https://www.tensorflow.org/tutorials/keras/save_and_load

  - Effective TensorFlow 2
    https://www.tensorflow.org/guide/effective_tf2

  - Migrate your TensorFlow 1 code to TensorFlow 2
    https://www.tensorflow.org/guide/migrate

参考サイト(2/2)

- GitHubのTensorFlowリポジトリ
  変更点等についてリリースノートを参照.
  https://github.com/tensorflow/tensorflow/blob/master/RELEASE.md