実践演習9-1

畳み込みニューラルネットワークでfasion MNISTデータの識別を行います。

準備

必要なライブラリ等を読み込みます。

In [0]:

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
```

データの読み込み

In [2]:

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

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-labels-idx1-ubyte.gz)
32768/29515 [================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/tr
ain-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/train-images-idx3-ubyte.gz)
26427392/26421880 [==============================] - 1s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-labels-idx1-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-labels-idx1-ubyte.gz)
8192/5148 [===============================================] - 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t1
0k-images-idx3-ubyte.gz (https://storage.googleapis.com/tensorflow/tf-keras-dataset
s/t10k-images-idx3-ubyte.gz)
4423680/4422102 [==============================] - 0s 0us/step
```

データサイズの確認

In [3]:

```
train_images.shape
```

Out[3]:

```
(60000, 28, 28)
```

In [4]:

```
test_images.shape
```

Out[4]:

```
(10000, 28, 28)
```

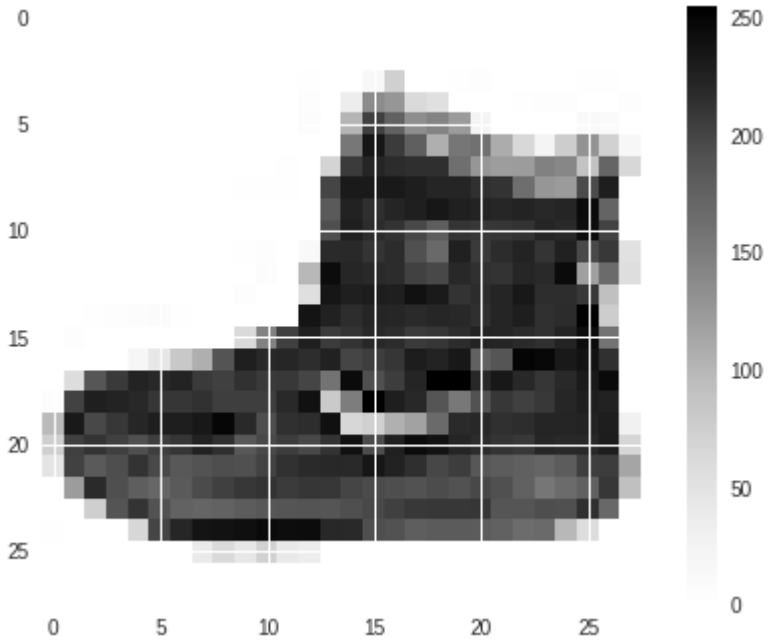
データの表示

In [5]:

```
plt.imshow(train_images[0])
plt.colorbar()
```

Out[5]:

```
<matplotlib.colorbar.Colorbar at 0x7f680a54ffd0>
```



ラベルとクラスとの対応

ラベル	クラス

ラベル	クラス
0	T-シャツ/トップ (T-shirt/top)
1	ズボン (Trouser)
2	プルオーバー (Pullover)
3	ドレス (Dress)
4	コート (Coat)
5	サンダル (Sandal)
6	シャツ (Shirt)
7	スニーカー (Sneaker)
8	バッグ (Bag)
9	アンクルブーツ (Ankle boot)

In [6]:

```
train_labels[0:20]
```

Out[6]:

```
array([9, 0, 0, 3, 0, 2, 7, 2, 5, 5, 0, 9, 5, 5, 7, 9, 1, 0, 6, 4],
      dtype=uint8)
```

元データでは濃淡が0から255までで表現されており、最大値が大きすぎるので、特徴の値の最大値を1とし、型をfloatにキャストします。

In [0]:

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

フィードフォワード型ネットワーク

まずベースラインとして3層のフィードフォワード型ネットワークで学習を行います。 Flattenは28×28の行列を784次元のベクトルに変換します。その後、中間層のユニット数を128、出力層のユニット数をクラス数(10)としてネットワークを構成します。

In [0]:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
```

損失関数と最適化器を指定します。metricsは学習の進行に従って表示されるものです。

In [0]:

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

繰り返し回数を指定して学習を行います。

In [10]:

```
model.fit(train_images, train_labels, epochs=5, batch_size=200)
```

```
Epoch 1/5
60000/60000 [==============================] - 3s 55us/step - loss: 0.5930 - acc: 0.
7965
Epoch 2/5
60000/60000 [==============================] - 1s 21us/step - loss: 0.4209 - acc: 0.
8522
Epoch 3/5
60000/60000 [==============================] - 1s 20us/step - loss: 0.3800 - acc: 0.
8663
Epoch 4/5
60000/60000 [==============================] - 1s 20us/step - loss: 0.3590 - acc: 0.
8718
Epoch 5/5
60000/60000 [==============================] - 1s 20us/step - loss: 0.3370 - acc: 0.
8795
```

Out[10]:

```
<tensorflow.python.keras.callbacks.History at 0x7f6848a25f98>
```

テストデータで評価します。

In [11]:

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

```
10000/10000 [==============================] - 1s 72us/step
Test accuracy: 0.8691
```

層数を増やしてみます。

In [12]:

```
model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=200)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

```
Epoch 1/5
60000/60000 [==============================] - 2s 28us/step - loss: 0.5621 - acc: 0.
8057
Epoch 2/5
60000/60000 [==============================] - 1s 24us/step - loss: 0.3866 - acc: 0.
8604
Epoch 3/5
60000/60000 [==============================] - 1s 23us/step - loss: 0.3432 - acc: 0.
8754
Epoch 4/5
60000/60000 [==============================] - 1s 22us/step - loss: 0.3175 - acc: 0.
8834
Epoch 5/5
60000/60000 [==============================] - 1s 23us/step - loss: 0.3019 - acc: 0.
8893
10000/10000 [==============================] - 1s 75us/step
Test accuracy: 0.8759
```

畳み込みニューラルネットワーク

データの変換（チャネル数=1を加えて4-dテンソルに）

In [0]:

```
train_images = train_images.reshape(train_images.shape[0], 28, 28, 1)
test_images = test_images.reshape(test_images.shape[0], 28, 28, 1)
```

ネットワークの構成の確認

In [14]:

```
model = keras.Sequential([
    keras.layers.Conv2D(16, kernel_size=(3, 3),
                activation=tf.nn.relu,
                input_shape=(28, 28, 1)),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Conv2D(32, (3, 3), activation=tf.nn.relu),
    keras.layers.MaxPooling2D(pool_size=(2, 2)),
    keras.layers.Flatten(),
    keras.layers.Dense(128, activation=tf.nn.relu),
    keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 32)        4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 32)          0
_________________________________________________________________
flatten_2 (Flatten)          (None, 800)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               102528
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 108,618
Trainable params: 108,618
Non-trainable params: 0
_________________________________________________________________
```

In [15]:

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(train_images, train_labels, epochs=5, batch_size=200)
test_loss, test_acc = model.evaluate(test_images, test_labels)
print('Test accuracy:', test_acc)
```

```
Epoch 1/5
60000/60000 [==============================] - 4s 72us/step - loss: 0.6378 - acc: 0.
7751
Epoch 2/5
60000/60000 [==============================] - 2s 39us/step - loss: 0.4045 - acc: 0.
8549
Epoch 3/5
60000/60000 [==============================] - 2s 38us/step - loss: 0.3557 - acc: 0.
8730
Epoch 4/5
60000/60000 [==============================] - 2s 38us/step - loss: 0.3284 - acc: 0.
8819
Epoch 5/5
60000/60000 [==============================] - 2s 38us/step - loss: 0.3081 - acc: 0.
8884
10000/10000 [==============================] - 1s 89us/step
Test accuracy: 0.8828
```