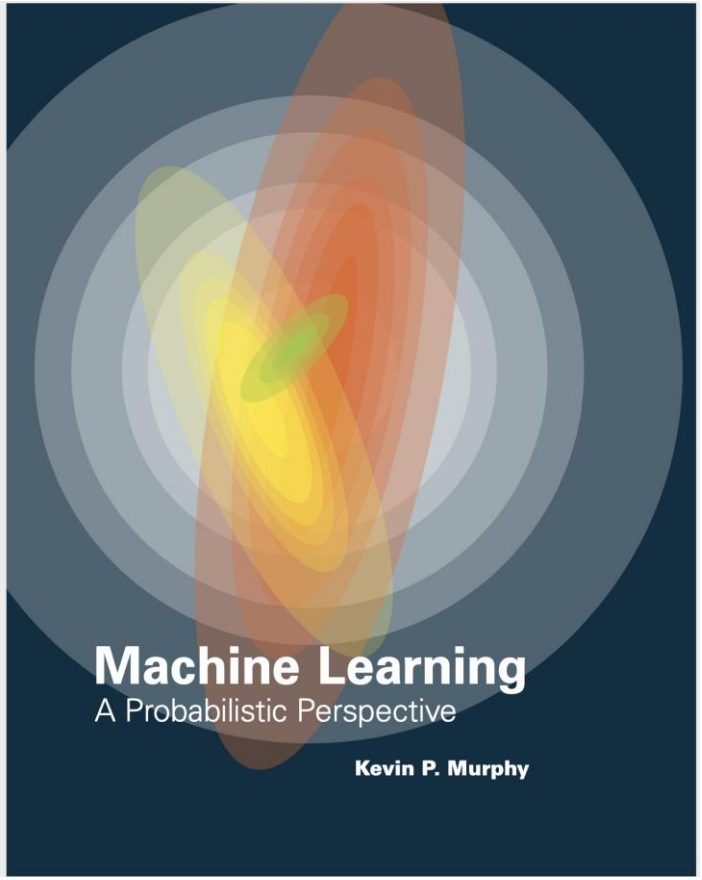
딥러닝 강의자료

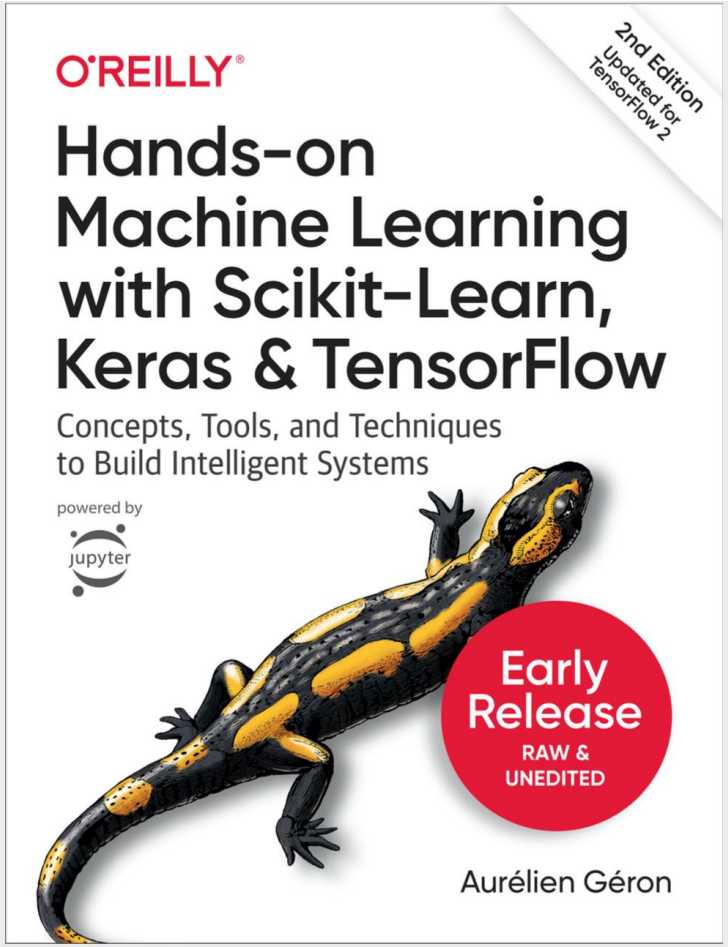
특강 – 인공 신경망으로 Fashion-MNIST

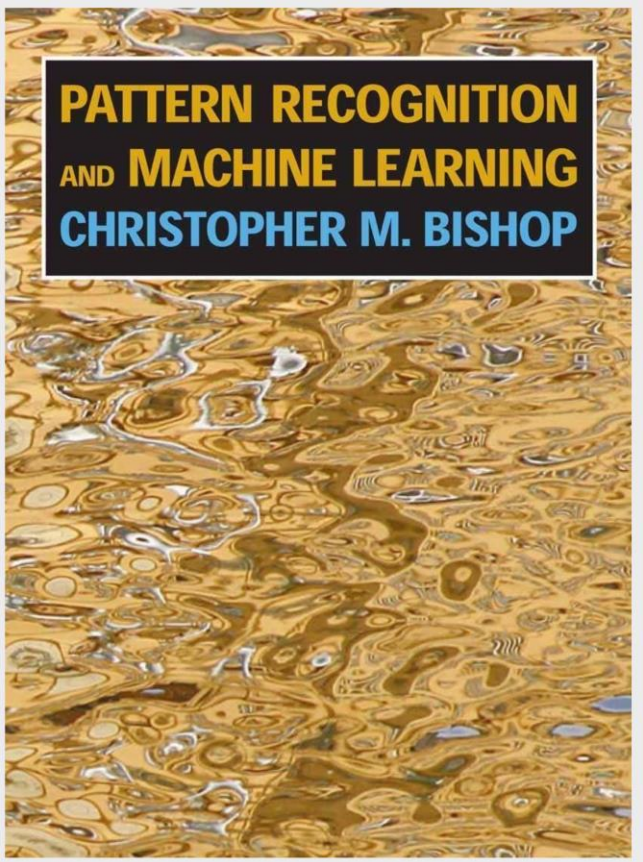
닥터윌컨설팅 딥러닝 R&D 책임연구원
고려대학교 인공지능대학원 박사과정

류회성(Hoe Sung Ryu)

들어가기 앞서

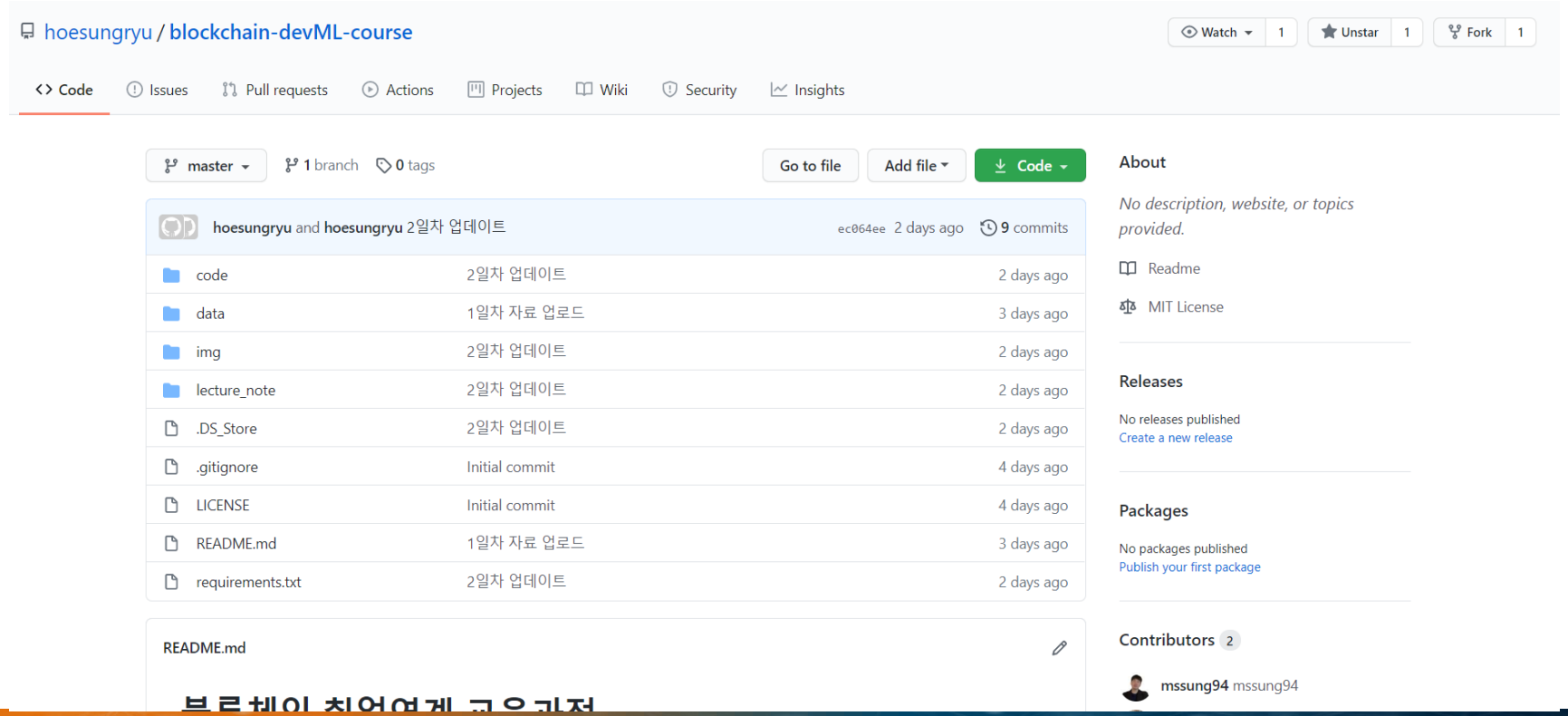






들어가기 앞서

- GitHub
  - https://github.com/hoesungryu/blockchain-devML-course



강의내용

- Fashion-MNIST 데이터셋을 이용해서 딥러닝
  - EDA
  - 데이터 전처리
  - 모델링
  - 결과 확인

[EDA]

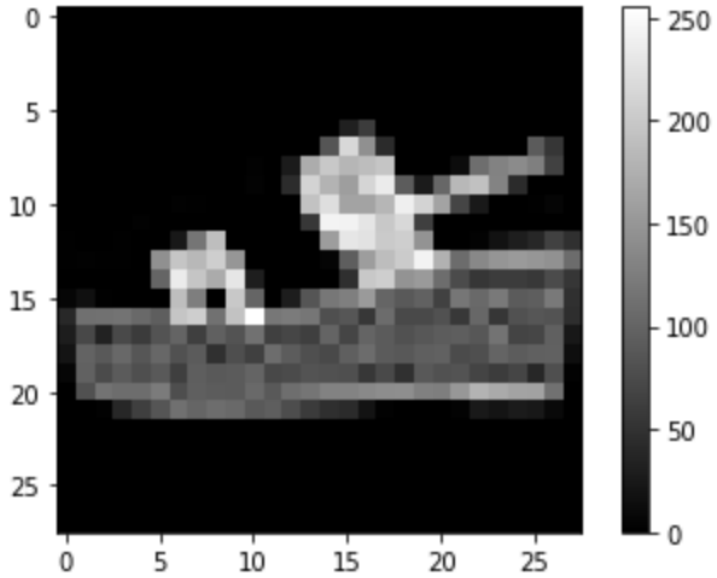
EDA

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

EDA

In [6]:
```
1  # 데이터 샘플링
2  plt.imshow(train_images[12], cmap='gray') # image 보여주기
3  plt.colorbar() # 오른쪽 컬러바
4  plt.show()
```

executed in 193ms, finished 00:45:53 2020-08-12



EDA

In [8]:
```
1  plt.figure(figsize=(10,10))
2
3  for i in range(16):
4      plt.subplot(4,4,i+1)
5      plt.xticks([]) # x 축에 아무것도 표시하지 않기
6      plt.yticks([])
7      plt.grid(False)
8      plt.imshow(train_images[i], cmap='gray')
9      plt.xlabel(class_names[train_labels[i]])
10 plt.show()
```

executed in 440ms, finished 00:45:54 2020-08-12

[데이터 전처리]

데이터 전처리

In [7]:
```
1 train_images = train_images / 255.0
2 test_images = test_images / 255.0
```
executed in 208ms, finished 00:45:53 2020-08-12

[모델링]

모델링

In [9]:

```
# model define
model = keras.Sequential([
    # layer1
    keras.layers.Flatten(input_shape=(28, 28)),  # 이미지 사이즈를 넣어주기 == input 크기 ,
    # layer 2
    keras.layers.Dense(     activation='elu'),  # layer 에 들어갈 노드개수 == 점의 개수
    keras.layers.Dense(     activation='elu'),
    keras.layers.Dense(     activation='elu'),
    # layer 3
    keras.layers.Dense(10, activation='softmax')  # 클래스가 10개인 분류이므로 softmax
])

model.summary()
```

executed in 87ms, finished 00:45:54 2020-08-12

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None,     )
_________________________________________________________________
dense_1 (Dense)              (None,     )
_________________________________________________________________
dense_2 (Dense)              (None,     )
_________________________________________________________________
dense_3 (Dense)              (None, 10)                330
=================================================================
Total params: 29,642
Trainable params: 29,642
Non-trainable params: 0
_________________________________________________________________
```

모델링

- 모델을 만들면 항상 아래 과정을 반복
  - model.compile()
  - model.fit()
  - model.evaluate()

모델링

- model.compile()

In [13]:
```
# 모델을 compile  해주기
model.compile(optimizer='adam',  # 학습률을 어떻게 업데이트 할 인가?
              loss='sparse_categorical_crossentropy',  # 멀티분류
              metrics=['accuracy', f1score])  # 평가지표를 무엇으로 할것인가?
# 분류 metrics = accuracy, f1, recall, precision
```

executed in 136ms, finished 00:45:56 2020-08-12

[모델 학습하기]

모델 학습하기

- model.fit()

In [16]:

```
# 모델 fitting
# fit(X, y) 학습
# epoch == 전체 데이터를 몇번 사용해서 학습할 것인가
# verbose 진행률을 프린트
model.fit(train_images, train_labels, epochs=10, verbose=1)
```

executed in 52.3s, finished 02:00:10 2020-08-12

```
Epoch 1/10
60000/60000 [==============================] - 5s 87us/sample - loss: 0.5025 - acc: 0.8180 - f1score: 0.8871
Epoch 2/10
60000/60000 [==============================] - 5s 83us/sample - loss: 0.3799 - acc: 0.8618 - f1score: 0.92291s - los
s: 0.383
Epoch 3/10
60000/60000 [==============================] - 5s 84us/sample - loss: 0.3444 - acc: 0.8731 - f1score: 0.9261
Epoch 4/10
60000/60000 [==============================] - 6s 95us/sample - loss: 0.3242 - acc: 0.8793 - f1score: 0.9281
Epoch 5/10
60000/60000 [==============================] - 5s 84us/sample - loss: 0.3075 - acc: 0.8850 - f1score: 0.9299
Epoch 6/10
60000/60000 [==============================] - 5s 79us/sample - loss: 0.2955 - acc: 0.8907 - f1score: 0.9301
Epoch 7/10
60000/60000 [==============================] - 5s 89us/sample - loss: 0.2851 - acc: 0.8928 - f1score: 0.9320
Epoch 8/10
60000/60000 [==============================] - 6s 92us/sample - loss: 0.2772 - acc: 0.8972 - f1score: 0.9327
Epoch 9/10
60000/60000 [==============================] - 5s 75us/sample - loss: 0.2672 - acc: 0.8989 - f1score: 0.9329
Epoch 10/10
60000/60000 [==============================] - 6s 94us/sample - loss: 0.2611 - acc: 0.9011 - f1score: 0.9340
```

모델 학습하기

- model.evaluate()

```
# Test 평가
# (tensorflow == evaluate)  === (sklearn== predict )
# model.predict(X_test, y_test)
test_loss, test_acc, test_f1_score = model.evaluate(test_images, test_labels, verbose=1)
```

executed in 682ms, finished 02:00:52 2020-08-12

```
10000/10000 [==============================] - 1s 66us/sample - loss: 0.3545 - acc: 0.8712 - f1score: 0.9305
```

```
print(f'우리모델의 정확도: {round(test_acc*100,2)} %')
print('우리모델의 f1_score: {:.3f}'.format(test_f1_score))
```

executed in 4ms, finished 02:01:41 2020-08-12

```
우리모델의 정확도: 87.12 %
우리모델의 f1_score: 0.931
```

[추론하기]

추론하기

- model.predict()