CNN
(Convolutional Neural Network)
실습

임세진

https://youtu.be/T6zmaddDOno

한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

CryptoCraft LAB

01. Deep Learning Framework

01. Deep Learning Framework

- **딥러닝 프레임워크 (Deep Learning Framework)**

- 프레임워크 : 응용 프로그램을 개발하기 위한 라이브러리나 모듈 등을 효율적으로 사용할 수 있도록 하나로 묶어놓은 패키지
- 딥러닝 프레임워크 : 검증된 라이브러리와 사전 학습이 완료된 다양한 딥러닝 알고리즘을 제공해 주어 개발자가 손쉽게 사용할 수 있도록 해줌

01. Deep Learning Framework

- **딥러닝 기반 프레임워크 종류**

- Theano
  - 최초의 딥러닝 라이브러리
  - 파이썬 기반
  - CPU 및 GPU의 수치계산에 매우 유용
  - 비교적 확장성이 떨어지며 다중 GPU 지원이 부족

- TensorFlow
  - 가장 인기있는 딥러닝 라이브러리 (구글팀 개발, 오픈소스)
  - 파이썬 기반 (C/C++ 엔진 + 파이썬 API으로 제작 >> 빠른 실행 가능)
  - 여러 CPU, GPU, 플랫폼, 데스크톱, 모바일에서 사용가능
  - 비교적 속도가 느린 편임

01. Deep Learning Framework

- **딥러닝 기반 프레임워크 종류**

- Keras
  - 차세대 딥러닝 프레임워크
  - Theano와 TensorFlow 같은 저수준 라이브러리가 아니어서 직접 모델을 만들지 않아도 됨
  - Theano와 TensorFlow에서 작동 가능
  - 파이썬으로 제작
- Torch
  - Lua라는 스크립트 언어를 기반으로 제작 (파이썬보다 빠른 특징)
  - 페이스북, 구글에서도 이 프레임워크를 기반으로하여 자체 버전을 개발하여 사용할 정도로 효율적인 프레임워크임
  - 강화 학습에 필요한 사전 학습된 다양한 라이브러리 제공

01. Deep Learning Framework

- **딥러닝 기반 프레임워크 종류**

- DL4J (DeepLearningForJava)
  - 자바로 개발
  - 자바 외에도 Closure나 Scala와 같은 JVM 언어도 지원
  - 상업 / 산업 중심의 분산 딥러닝 플랫폼으로 널리 사용
  - 빅데이터 도구와 함께 사용할 수 있어 효율적인 딥러닝 가능

02. MNIST 기초

02. MNIST 기초

- **MNIST 기초**

- 머신러닝 기초 실습 ex) 프로그래밍 언어 실습에서의 “Hello World” 출력

- 사람 손으로 쓴 0~9까지의 숫자 이미지로 이루어진 컴퓨터 비전 데이터셋

- 55,000개의 학습 데이터(mnist.train)와 10,000개의 테스트 데이터(mnist.test)

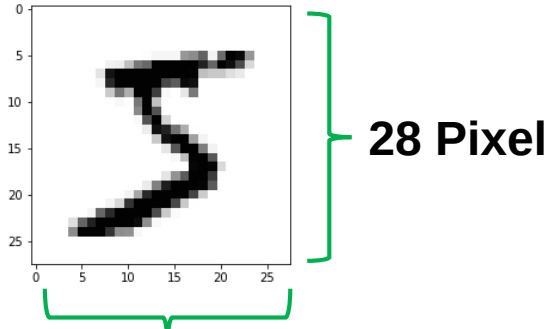
- MNIST의 각 이미지는 28x28 픽셀로 이루어짐

  - **mnist.train.images[55000, 784]**

MNIST의 학습 데이터

28x28

각 이미지가 가진 픽셀 데이터 수



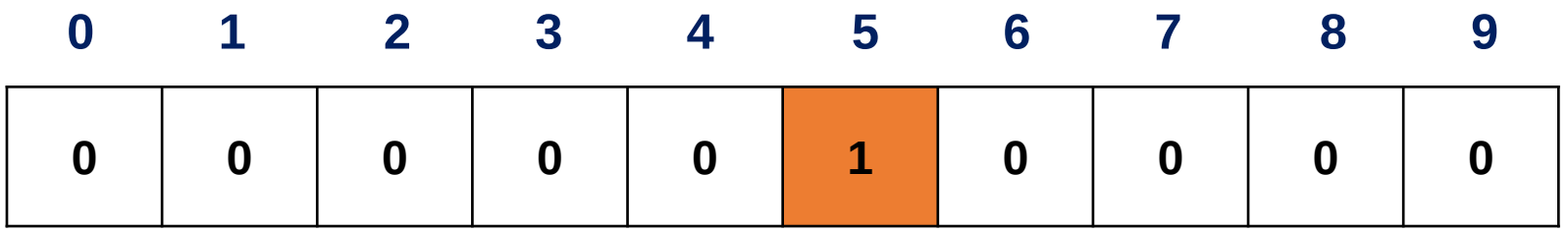
02. MNIST 기초

- MINIST 이미지 라벨

- MNIST의 각 이미지가 의미하는 숫자를 나타냄

- One-hot Vector 방식으로 표현

각 숫자를 배열의 인덱스로 사용하고,
해당 숫자의 인덱스에 해당하는 배열의 값을 1로 설정하는 방식



이미지가 숫자 5일 때의 라벨

- mnist.train.label[55000, 10]

학습데이터의 0~9까지를 표현한 라벨

02. Keras로 MNIST 학습 및 예측 실습

03. Keras로 MNIST 학습 및 예측 실습

```
class CNN(models.Sequential):
  def __init__(self, input_shape, num_classes):
    super().__init__()

    self.add(layers.Conv2D(32, kernel_size=(3, 3), input_shape=input_shape))
    self.add(layers.BatchNormalization(axis=1))
    self.add(layers.Activation('relu'))
    self.add(layers.MaxPooling2D(pool_size=(2, 2)))
    self.add(layers.Conv2D(64, kernel_size=(3, 3)))
    self.add(layers.BatchNormalization(axis=1))
    self.add(layers.Activation('relu'))
    self.add(layers.MaxPooling2D(pool_size=(2, 2)))
    self.add(layers.Dropout(0.25))
    self.add(layers.Flatten())
    self.add(layers.Dense(128))
    self.add(layers.BatchNormalization(axis=1))
    self.add(layers.Activation('relu'))
    self.add(layers.Dropout(0.25))
    self.add(layers.Dense(num_classes, activation='softmax'))

    self.compile(loss=keras.losses.categorical_crossentropy,
                 optimizer='adam', metrics=['accuracy'])
```

순차적으로 레이어 층을 더해주는 순차모델 사용

Convolution layer

활성화 함수

Pooling layer

Dropout layer. 정규화. 임의의 뉴런을 무작위로 선택 > 선택된 뉴런을 제외하고 학습

Flatten layer

Dense layer. 입력과 출력을 모두 연결해주는 레이어

모델 컴파일 : 평가지표 설정, 최적화 함수, 손실 함수 설정 + 가중치 초기화

03. Keras로 MNIST 학습 및 예측 실습

```
class MnistData():
  def __init__(self):
    (x_train, y_train), (x_test, y_test) = datasets.mnist.load_data()

    y_train = np_utils.to_categorical(y_train)
    y_test = np_utils.to_categorical(y_test)

    img_rows, img_cols = x_train.shape[1:]

    if backend.image_data_format() == 'channels_first':
      x_train = x_train.reshape(x_train.shape[0], 1, img_rows, img_cols)
      x_test = x_test.reshape(x_test.shape[0], 1, img_rows, img_cols)
      input_shape = (1, img_rows, img_cols)
    else:
      x_train = x_train.reshape(x_train.shape[0], img_rows, img_cols, 1)
      x_test = x_test.reshape(x_test.shape[0], img_rows, img_cols, 1)
      input_shape = (img_rows, img_cols, 1)

    x_train = x_train.astype('float32')
    x_test = x_test.astype('float32')
    x_train /= 255.0
    x_test /= 255.0

    self.input_shape = input_shape
    self.num_classes = 10
    self.x_train, self.y_train = x_train, y_train
    self.x_test, self.y_test = x_test, y_test
```

MNIST 데이터셋을 학습용, 테스트용으로 나누기

데이터 전처리 (정규화)

데이터 전처리

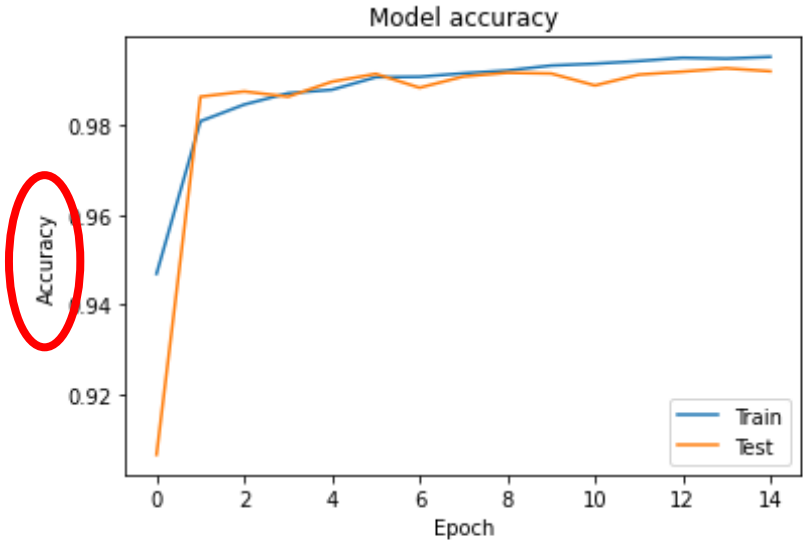
03. Keras로 MNIST 학습 및 예측 실습

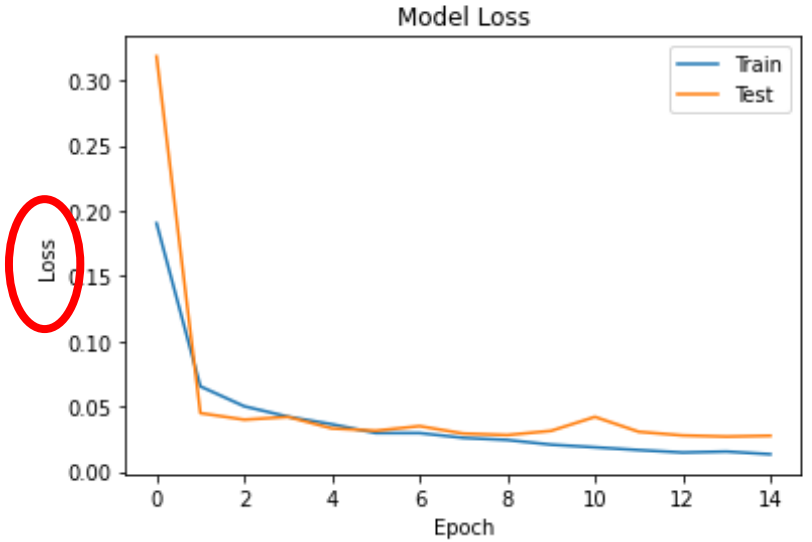
```
def train():
  batch_size = 100
  epochs = 15 전체 데이터에 대한 학습 횟수

  data = MnistData() 데이터를 불러와 CNN input 규격에 맞게 변형하는 class
  model = CNN(data.input_shape, data.num_classes)

  history = model.fit(data.x_train, data.y_train, epochs=epochs, 학습 실행
                      batch_size=batch_size, validation_split=0.2, verbose=2)

  performance_test = model.evaluate(data.x_test, data.y_test, batch_size=100,
                                    verbose=0)
  print('\nTest Result ->', performance_test)

  model.save_weights('cnn_model_mnist.h5')

  plot_acc(history)
  plt.show() Training 정확도를 보여줌
  plot_loss(history)
  plt.show() Training 의 오차를 수치화하여 보여줌
```

학습 실습

03. Keras로 MNIST 학습 및 예측 실습

```
Epoch 1/15
480/480 - 3s - loss: 0.1904 - accuracy: 0.9469 - val_loss: 0.3182 - val_accuracy: 0.9066
Epoch 2/15
480/480 - 2s - loss: 0.0654 - accuracy: 0.9809 - val_loss: 0.0449 - val_accuracy: 0.9863
Epoch 3/15
480/480 - 2s - loss: 0.0502 - accuracy: 0.9846 - val_loss: 0.0399 - val_accuracy: 0.9875
Epoch 4/15
480/480 - 2s - loss: 0.0422 - accuracy: 0.9871 - val_loss: 0.0420 - val_accuracy: 0.9863
Epoch 5/15
480/480 - 2s - loss: 0.0364 - accuracy: 0.9879 - val_loss: 0.0332 - val_accuracy: 0.9897
Epoch 6/15
480/480 - 2s - loss: 0.0297 - accuracy: 0.9907 - val_loss: 0.0315 - val_accuracy: 0.9914
Epoch 7/15
480/480 - 2s - loss: 0.0297 - accuracy: 0.9908 - val_loss: 0.0350 - val_accuracy: 0.9883
Epoch 8/15
480/480 - 2s - loss: 0.0260 - accuracy: 0.9915 - val_loss: 0.0293 - val_accuracy: 0.9908
Epoch 9/15
480/480 - 2s - loss: 0.0244 - accuracy: 0.9922 - val_loss: 0.0281 - val_accuracy: 0.9917
Epoch 10/15
480/480 - 2s - loss: 0.0208 - accuracy: 0.9933 - val_loss: 0.0314 - val_accuracy: 0.9915
Epoch 11/15
480/480 - 2s - loss: 0.0188 - accuracy: 0.9937 - val_loss: 0.0421 - val_accuracy: 0.9888
Epoch 12/15
480/480 - 2s - loss: 0.0166 - accuracy: 0.9943 - val_loss: 0.0307 - val_accuracy: 0.9912
Epoch 13/15
480/480 - 2s - loss: 0.0148 - accuracy: 0.9950 - val_loss: 0.0278 - val_accuracy: 0.9919
Epoch 14/15
480/480 - 2s - loss: 0.0155 - accuracy: 0.9948 - val_loss: 0.0270 - val_accuracy: 0.9927
Epoch 15/15
480/480 - 2s - loss: 0.0135 - accuracy: 0.9952 - val_loss: 0.0276 - val_accuracy: 0.9920

Test Result -> [0.019687438383698463, 0.9933000206947327]
```





03. Keras로 MNIST 학습 및 예측 실습

```
def predict():
  img = Image.open("input.png")
  img_data = np.array(img)

  plt.imshow(img_data)
  plt.show()

  if backend.image_data_format() == 'channels_first':
    input_shape = (1, 28, 28)
    img_data = img_data.transpose(2, 0, 1)[1].reshape(1, 1, 28, 28)
  else:
    input_shape = (28, 28, 1)
    img_data = img_data[:, :, 1].reshape(1, 28, 28, 1)

  img_data = img_data.astype('float32') / 255.0

  model = CNN(input_shape, 10)
  model.load_weights('cnn_model_mnist.h5')

  output = model.predict(img_data)
  print("Answer :", np.argmax(output))
```

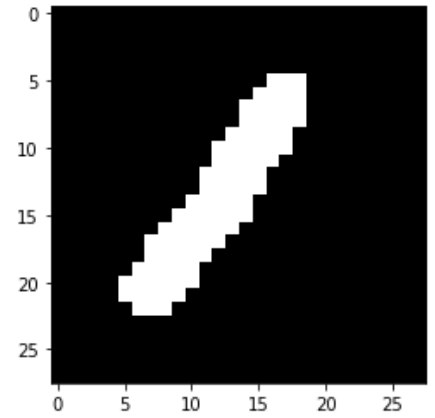
Test Data 가져오기

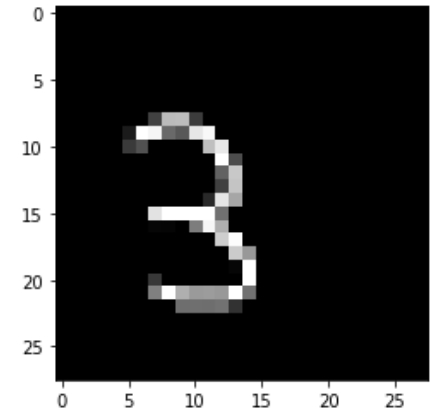
(행, 열, 채널 수). 흑백이므로 채널 수 = 1

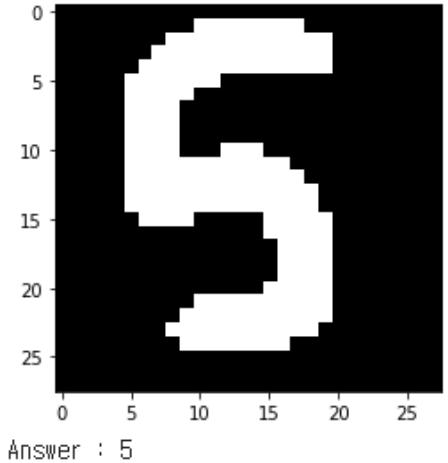
Training data

Predict

예측 실습







감사합니다