인공지능을 위한 머신러닝 알고리즘

15. Keras를 통한 딥러닝 구현 및 실습

Tacademy

CONTENTS

학습 목표

- Keras 라이브러리의 특징과 기본적인 함수들을 사용할 수 있다.
- Keras로 컨볼루션 신경망을 구현할 수 있다.
- Keras를 사용하여 이미지를 분석할 수 있다.

1. Keras란?

Keras의 특징

- Keras의 기본 아이디어는 모델의 층들과 그것들의 입력과 출력에 있음
- Keras를 사용하여 모델 구축하기
  1. 입력/출력 데이터를 준비
  2. 첫 번째 층을 생성하고 입력 데이터에 맞게 설정해 줌
  3. 마지막 층을 생성하고 출력 데이터에 맞게 설정해 줌
  4. 입력 층과 출력 층 사이에 원하는 레이어를 생성해 줌

층 (layer)

- **Keras**는 미리 구현되어 있는 다양한 층들을 제공해줌:
  - 다층 퍼셉트론에서 사용되는 **Dense**층

Dense
keras.layers.core.Dense(output_dim, init='glorot_uniform', activation=**None**, weights=**None**, W_regularizer=**None**, b_regularizer=**None**, activity_regularizer=**None**, W_constraint=**None**, b_constraint=**None**, bias=**True**, input_dim=**None**)

  - 재현층, **LSTM, GRU, etc**

Reccurent
keras.layers.recurrent.Recurrent(weights=**None**, return_sequences=**False**, go_backwards=**False**, stateful=**False**, unroll=**False**, consume_less='cpu', input_dim=**None**, input_length=**None**)

층 (layer)

1D 컨볼루션 층

Convolution1D
keras.layers.convolutional.Convolution1D(nb_filter, filter_length, init='glorot_uniform', activation=**None**, weights=**None**, border_mode='valid', subsample_length=1, W_regularizer=**None**, b_regularizer=**None**, activity_regularizer=**None**, W_constraint=**None**, b_constraint=**None**, bias=**True**, input_dim=**None**, input_length=**None**)

2D 컨볼루션 층

Convolution2D
keras.layers.convolutional.Convolution2D(nb_filter, nb_row, nb_col, init='glorot_uniform', activation=**None**, weights=**None**, border_mode='valid', subsample=(1, 1), dim_ordering='default', W_regularizer=**None**, b_regularizer=**None**, activity_regularizer=**None**, W_constraint=**None**, b_constraint=**None**, bias=**True**)

활성함수 (activation function)

❖ 다양한 활성함수

- 간단한 활성함수: **Sigmoid, tanh, ReLu, softplus, hard_sigmoid, linear**
- 복잡한 활성함수: **LeakyReLu, PReLu, ELU, Parametric Softplus, Thresholded linear and Thresholded Relu**

❖ 목표 함수

- 에러 측정용 목표함수: **rmse, mse, mae, mape, msle**
- 최대 마진 목표함수: **squared_hinge, hinge**
- 분류용 목표함수: **binary_crossentropy, categorical_crossentropy**

저장/로드

❖ 모델 구조의 저장/로드 가능

```
from models import model_from_json

json_string = model.to_json()
model = model_from_json(json_string)
```

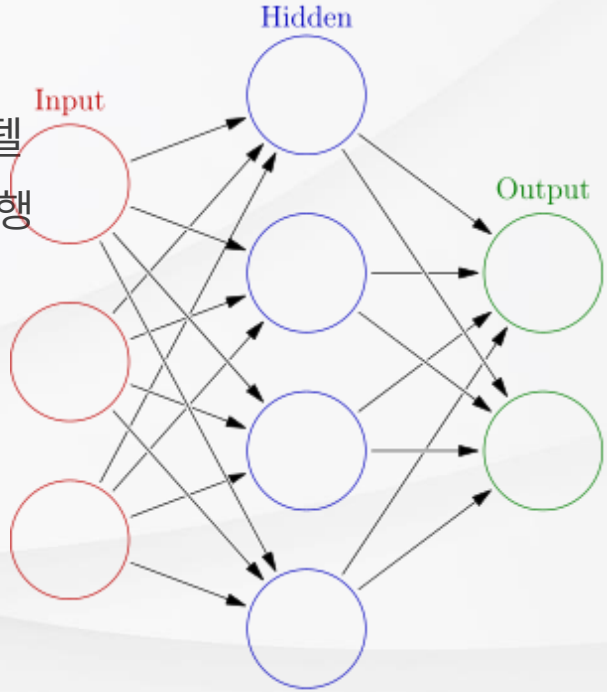
```
from models import model_from_yaml

yaml_string = model.to_yaml()
model = model_from_yaml(yaml_string)
```

❖ 모델 파라미터의 저장/로드 가능

- **model.save_weights(filepath):** 모델의 가중치를 **HDF5 file**로 저장
- **model.load_weights(filepath, by_name=False):**
  - 모델의 가중치를 **HDF5** 파일로부터 불러옴(**save_weights**에 의해 만들어짐)
  - 모델의 구조가 다를 때 불러오고자 한다면, **by_name=True**로 설정
  - 층의 이름에 맞게 파라미터가 불러옴 ▪ 모델의 구조는 변하지 않음

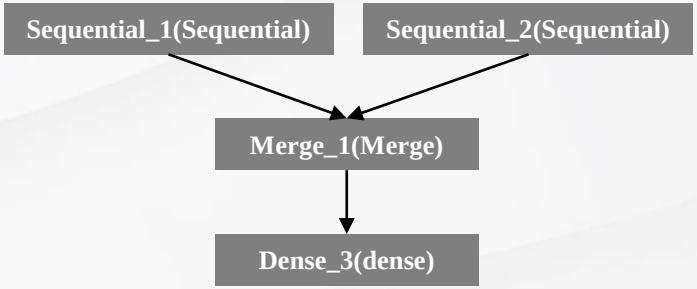
모델 종류: Sequential

- **Sequential** 모델은 층들의 스택(**stack**)으로 이뤄짐
- 이전 강의들에서 배운 컨볼루션/재현 신경망과 같은 모델
- 각 층은 그 다음 층에 입력을 제공해주는 객체의 역할 수행



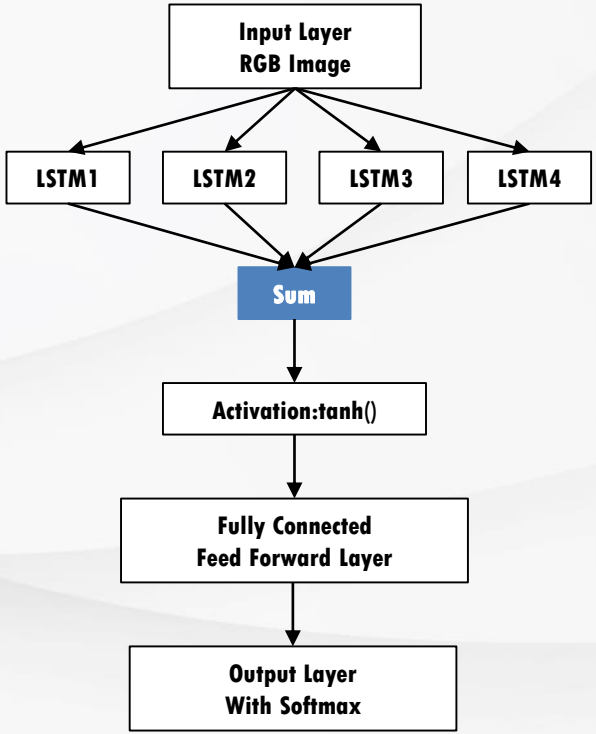
Sequential 모델의 예시

```
from keras.layers import Merge

left_branch = Sequential()
left_branch.add(Dense(32, input_dim=784))

right_branch = Sequential()
right_branch.add(Dense(32, input_dim=784))

merged = Merge([left_branch, right_branch], mode = ‘concat’)

final_model = Sequential()
final_model.add(merged)
final_model.add(Dense(10, activation=‘softmax’))
```



모델 종류: Graph

- **Graph** 모델에서 두 개 이상의 서로 다른 모델이 합쳐지거나 여러 모델로 나뉘어질 수 있음
- 다양한 개수의 입력과 출력 가능
- **Sequential** 모델과 다른 함수 제공



Graph 모델의 예시

```
def ranking_loss(y_true, y_pred):
    pos = y_pred[:,0]
    neg = y_pred[:,1]
    loss = -K.sigmoid(pos-neg) # use loss = K.maximum(1.0 + neg - pos, 0.0) if you want to use margin ranking loss
    return K.mean(loss) + 0 * y_true


def get_graph(num_users, num_items, latent_dim):

    model = Graph()
    model.add_input(name='user_input', input_shape=(num_users,))
    model.add_input(name='positive_item_input', input_shape=(num_items,))
    model.add_input(name='negative_item_input', input_shape=(num_items,))

    model.add_node(layer=Dense(latent_dim, input_shape = (num_users,)),
                   name='user_latent',
                   input='user_input')
    model.add_shared_node(layer=Dense(latent_dim, input_shape = (num_items,)),
                          name='item_latent',
                          inputs=['positive_item_input', 'negative_item_input'],
                          merge_mode=None,
                          outputs=['positive_item_latent', 'negative_item_latent'])

    model.add_node(layer=Activation('linear'), name='user_pos', inputs=['user_latent', 'positive_item_latent'], merge
    model.add_node(layer=Activation('linear'), name='user_neg', inputs=['user_latent', 'negative_item_latent'], merge

    model.add_output(name='triplet_loss_out', inputs=['user_pos', 'user_neg'])
    model.compile(loss={'triplet_loss_out': ranking_loss}, optimizer=Adam())#Adagrad(lr=0.1, epsilon=1e-06))

    return model
```

Keras의 장단점

❖ 장점

- 쉬운 모델 구현
- 다양한 함수 제공
- 손쉽게 커스터마이징 가능
- GPU를 활용한 빠른 계산
- 직관적인 함수 제공
- 활발한 커뮤니티

❖ 단점

- 생성 모델의 부족
- 고수준 라이브러리
- Theano 사용에 따른 에러 분석의 어려움

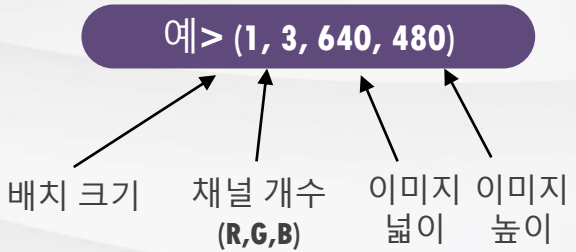
2. Keras로 컨볼루션 신경망 구현하기

VGG-16 모델 정의

- 모델 종류: **Sequential**
- 역할: 층들의 스택을 위한 빈 모델 생성

```
# build the VGG16 network
model = Sequential()
```

- **Sequential** 모델의 첫 층에서는 입력 데이터의 모양이 어떻게 생겼는지에 관한 정보를 알고 있어야 함
- 인수
  - **batch_input_shape:** 데이터 튜플의 모양



VGG-16 모델 정의

- 층 이름: **ZeroPadding2D**
- 층 종류: 컨볼루션 층
- 역할: **2D** 입력(예> 이미지)를 위한 제로 패딩 층
- 인수
  - 패딩: **int** 형식의 길이 **2** 또는 길이 **4** 튜플
  - **batch_input_shape**

```
model.add(ZeroPadding2D((1, 1),
                        batch_input_shape=(1, 3,
                        img_width, img_height)))
```

VGG-16 모델 정의

- 층 이름: **Convolution2D**
- 층 종류: 컨볼루션 층
- 역할: **2D** 입력에 대해 컨볼루션 계산
- 인수
  - **nb_filter:** 사용할 컨볼루션 필터 개수
  - **nb_row:** 컨볼루션 커널의 행 개수
  - **nb_col:** 컨볼루션 커널의 열 개수
  - 활성함수: 사용할 활성함수 이름, 입력하지 않을 시 선형 활성함수 사용 **(a(x) = x)**

```
model.add(Convolution2D(64, 3, 3,
                        activation='relu',name='conv1_1'))
```

VGG-16 모델 정의

- 층 이름: **MaxPooling2D**
- 층 종류: 컨볼루션 층
- 역할: 시계열 데이터에 대해서 최대 풀링 계산
- 인수
  - **pooling_length:** 최대 풀링을 적용할 지역 크기 설정
  - **stride:** 다운 스케일할 **int** 값 (예> **2**는 입력의 크기를 절반으로 줄임)

```
model.add(MaxPooling2D((2, 2),
                       strides=(2, 2)))
```

학습정리

지금까지 [Keras를 통한 딥러닝 구현 및 실습]에 대해서 살펴보았습니다.

Keras란?

Keras는 고수준의 다양한 모델 층, 활성함수, 목표함수 등을 제공하여 손쉽게 딥러닝 모델을 구현하게 해주는 라이브러리
사용할 입력/출력 데이터를 모델의 처음과 마지막 층에 설정하고
가운데 층을 목적에 맞게 선택가능

Keras로 컨볼루션 신경망 구현하기

Sequential 모델의 ZeroPadding2D / Convolution2D / MaxPooling2D 함수를 사용하여 VGG-16 모델 구현

컨볼루션 신경망이 보는 세상의 모습

컨볼루션 신경망의 컨볼루션층들은
직선/색 등의 저차원부터 도형 등의 고차원 특징 추출