AIoT AutoCar Prime 으로 배우는

온디바이스 AI 프로그래밍

7. 인공지능

7.5 Keras

Keras

- Keras
  - 딥러닝 구현에 특화된 고수준 신경망 API
  - 구글의 Tensorflow 프레임워크와 함께 사용 가능
  - CNN, RNN 등 복잡한 신경망을 쉬운 API로 제공
  - 직관적인 구조로 되어 있어, 다양한 신경망을 쉽게 설계 가능

모델과 레이어

- 모델
  - 입력층, 은닉층, 출력층 등 레이어가 연결되어 이루는 하나의 레이어 연결체
- 레이어
  - 모델을 구성하는 하나의 연산 계층
  - 가중치를 조절하여 모델 학습

모델과 레이어

- Keras를 이용하면 직관적으로 레이어를 연결해 학습 모델 생성 가능
- Keras의 Sequential 메소드로 순차 모델 생성 가능
- 순차 모델에 add 메소드로 레이어를 추가
  - 자동으로 가중치를 생성해 레이어 연결

모델과 레이어

- Keras 실습 예제
  - Tensorflow에서 라이브러리 import
  - 간단한 모델을 만들어 임의 값을 입력하고 출력 확인

```
01:     from tensorflow import keras
02:
03:     model = keras.models.Sequential()
04:
05:     model.add(keras.layers.Input(shape=(1,)))
06:     model.add(keras.layers.Dense(50))
07:     model.add(keras.layers.Dense(1))
08:
09:     value = model.predict([1])
10:     print(value)
```

모델과 레이어

- summary 메소드를 통해 모델의 구조 확인

```
11:             model.summary()
```

```
[출력]
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                100
_________________________________________________________________
dense_1 (Dense)              (None, 10)                510
=================================================================
Total params: 610
Trainable params: 610
Non-trainable params: 0
_________________________________________________________________
```

- 출력에는 입력층은 표시되지 않고 은닉층과 출력층만 표시
- 각 구간별 가중치의 수를 Parameter에 표시

컴파일

- 컴파일
  - Keras에서 어떤 방식과 손실 함수로 모델을 최적화할지 명시하는 과정
  - model 객체의 compile 메소드로 컴파일
  - 손실 함수, 최적화 함수 등을 선택해 모델의 학습 방향을 지정하는 역할
  - 손실 함수로 MSE, 최적화 함수로 SGD을 사용해 컴파일
    - MSE : 평균 제곱 오차, SGD : 경사하강법

```
model.compile(loss='MSE', optimizer='SGD')
```

선형 회귀

- Keras의 선형 회귀
  - 입력층과 출력층의 노드 : 1개
  - 평균 제곱 오차와 경사하강법으로 컴파일
  - model의 fit 메소드를 사용해 컴파일한대로 모델 학습
  - 기본적인 선형 회귀 모델을 목표로 X와 Y데이터를 입력
  - epochs에 학습 횟수를 입력하여 fit 메소드 호출

선형 회귀

```
01:     from tensorflow import keras
02:
03:     model = keras.models.Sequential()
04:
05:     model.add(keras.layers.Input(shape=(1,)))
06:     model.add(keras.layers.Dense(1))
07:
08:     model.compile(loss='MSE',
09:                   optimizer='SGD')
10:
11:     X=[[0],[1],[2],[3],[4],[5]]
12:     Y=[[0],[2],[4],[6],[8],[10]]
13:
14:     model.fit(X, Y, epochs=100)
```

선형 회귀

- 학습이 끝나면 model의 predict 메소드를 이용해 모델 출력 확인

```
15:     value = model.predict([[7], [-10], [358]])
16:     print(value)
```

선형 회귀

전체 코드

```
01:     from tensorflow import keras
01:
02:     model = keras.models.Sequential()
03:
04:     model.add(keras.layers.Input(shape=(1,)))
05:     model.add(keras.layers.Dense(1))
06:
07:     model.compile(loss='MSE',
08:                   optimizer='SGD')
09:
10:     X=[[0],[1],[2],[3],[4],[5]]
11:     Y=[[0],[2],[4],[6],[8],[10]]
12:
13:     model.fit(X, Y, epochs=100)
14:
15:     value = model.predict([[7], [-10], [358]])
16:     print(value)
```

로지스틱 회귀

- 로지스틱 회귀
  - 입력에 대해 0 ~ 1을 출력하는 회귀 모델
  - 기본적인 구성은 선형 회귀 모델과 같음
  - 활성화 함수, 손실 함수, 최적화 함수를 조절해 구현 가능

로지스틱 회귀

- 로지스틱 회귀 예제
  - 출력층에 활성화 함수를 Sigmoid로 지정
  - 컴파일 하고 학습
    - 손실 함수 : binary_crossentropy, 최적화 함수 : Adam
  - 모델은 음수와 양수를 구분하는 것을 목표로 X와 Y데이터를 입력
  - epochs에 학습 횟수를 입력하여 fit 메소드 호출

로지스틱 회귀

```
01:     from tensorflow import keras
02:
03:     model = keras.models.Sequential()
04:
05:     model.add(keras.layers.Input(shape=(1,)))
06:     model.add(keras.layers.Dense(1, activation='sigmoid'))
07:
08:     model.compile(loss='binary_crossentropy',
09:                   optimizer='Adam')
10:
11:     X=[[-3],[-2],[-1],[1],[2],[3]]
12:     Y=[[0],[0],[0],[1],[1],[1]]
13:
14:     model.fit(X, Y, epochs=100)
```

로지스틱 회귀

- 학습이 끝나면 model의 predict 메소드를 이용해 모델 출력 확인

```
15:     value = model.predict([[7], [-10], [358]])
16:     print(value)
```

로지스틱 회귀

전체 코드

```
01:     from tensorflow import keras
02:
03:     model = keras.models.Sequential()
04:
05:     model.add(keras.layers.Input(shape=(1,)))
06:     model.add(keras.layers.Dense(1, activation='sigmoid'))
07:
08:     model.compile(loss='binary_crossentropy',
09:             optimizer='Adam')
10:
11:     X=[[-3],[-2],[-1],[1],[2],[3]]
12:     Y=[[0],[0],[0],[1],[1],[1]]
13:
14:     model.fit(X, Y, epochs=100)
15:
16:     value = model.predict([[7], [-10], [358]])
17:     print(value)
```

심층신경망

- 심층신경망 실습
  - 심층신경망의 학습 능력을 실험해보기 위해 비교적 복잡한 데이터셋을 준비
  - 3개의 입력이 주어졌을 때 1번째 원소와 2번째 원소의 크기를 비교한 결과를 1번째 출력하고
  - 2번째 원소와 3번째 원소의 크기를 비교한 결과를 2번째 출력으로 하는 모델을 만드는 예제
  - 각 출력은 좌항이 크면 0, 우항이 크면 1을 출력

심층신경망

- 입력 3개, 출력 2개의 모델에 노드 100개씩 갖는 은닉층 레이어 4개 추가
- 활성화 함수는 Relu
- 컴파일
  - 출력층의 활성화 함수 : sigmoid, 손실 함수 : binary_crossentropy
- 심층신경망이 조금만 학습해도 충분한 학습이 이루어지는지 실험
  - 20회 학습

심층신경망

```
01:     from tensorflow import keras
02:
03:     model = keras.models.Sequential()
04:
05:     model.add(keras.layers.Input(shape=(3,)))
06:     model.add(keras.layers.Dense(100, activation='relu'))
07:     model.add(keras.layers.Dense(100, activation='relu'))
08:     model.add(keras.layers.Dense(100, activation='relu'))
09:     model.add(keras.layers.Dense(100, activation='relu'))
10:     model.add(keras.layers.Dense(2, activation='sigmoid'))
11:
12:     model.compile(loss='binary_crossentropy',
13:             optimizer='Adam')
14:
15:     X=[[1,-2,3],[2,-3,4],[3,5,4],[4,-5,6],[7,-5,3], [1,6,8], [3,8,1]]
16:     Y=[[0, 1],[0, 1],[1, 0],[0, 1],[1, 1],[0, 1],[1, 1],[1, 0]]
17:
18:     model.fit(X, Y, epochs=20)
```

심층신경망

- 학습이 끝나면 model의 predict 메소드를 이용해 모델 출력 확인

```
19:     value = model.predict([[10,-5,3], [-1,6,-8], [3,5,8]])
20:     print(value)
```

심층신경망

전체 코드

```
01: from tensorflow import keras
02:
03: model = keras.models.Sequential()
04:
05: model.add(keras.layers.Input(shape=(3,)))
06: model.add(keras.layers.Dense(100, activation='relu'))
07: model.add(keras.layers.Dense(100, activation='relu'))
08: model.add(keras.layers.Dense(100, activation='relu'))
09: model.add(keras.layers.Dense(100, activation='relu'))
10: model.add(keras.layers.Dense(2, activation='sigmoid'))
```

```
11:
12: model.compile(loss='binary_crossentropy', optimizer='Adam')
13:
14: X=[[1,-2,3],[2,-3,4],[3,5,4],[4,-5,6],[7,-5,3], [1,6,8], [3,8,1]]
15: Y=[[0, 1],[0, 1],[1, 0],[0, 1],[1, 1],[0, 1],[1, 1],[1, 0]]
16:
17: model.fit(X, Y, epochs=20)
18:
19: value = model.predict([[10,-5,3], [-1,6,-8], [3,5,8]])
20: print(value)
```

내용 정리

- 케라스(Keras)
  - 딥러닝에 특화된 고수준 신경망 API 라이브러리
- 모델:입력층, 은닉층, 출력층이 연결된 하나의 레이어 연결체
- 레이어 : 모델을 구성하는 하나의 연산 계층
- 컴파일 : 모델이 어떻게 학습할지 명시하는 과정.
  - 손실 함수, 최적화 함수 등을 설정.

연습문제

- 문제 51. 다음은 코드 조각들입니다. Keras로 3개의 입력에 대해 4개의 출력을 하는 모델을 작성하고자 할 때 코드를 순서에 맞게 나열하세요.

A	`01: model = keras.models.Sequential()`
B	`01: model.add(keras.layers.Input(shape=(3,)))`
C	`01: from tensorflow import keras`
D	`01: model.add(keras.layers.Dense(4))`

연습문제

- 문제 52. 어떤 모델 A를 평균 제곱 오차로 손실율을 계산하고, 경사하강법으로 최적화하는 모델로 컴파일하려고 할 때 빈 칸에 들어갈 내용을 작성하세요.

```
01:          A.compile(                    )
```

연습문제

- 문제 53. 다음 코드는 Keras로 로지스틱 회귀 모델을 구현한 코드입니다. 질문을 읽고 답하세요.

```
01:     from tensorflow import keras
02:
03:     model = keras.models.Sequential()
04:
05:     model.add(keras.layers.Input(shape=(1,)))
06:     model.add(keras.layers.Dense(1, activation='sigmoid'))
07:
08:     model.compile(loss='binary_crossentropy',
09:                   optimizer='Adam')
10:
11:     X=[ X ]
12:     Y=[ Y ]
13:
14:     model.fit(X, Y, epochs=100)
```

연습문제

- A. 양수면 1, 음수면 0을 출력하도록 회귀하고자 할 때 빈 칸 X, Y에 들어갈 학습 데이터셋을 작성하세요.
- B. 로지스틱 회귀 모델에 −1, 1001, −10000를 입력했을 때의 출력을 작성해보세요.
- C. 0.1을 입력했을 때 오차가 ± 0.01 이하인 모델을 만들어보고 모델의 손실율을 작성해보세요.

연습문제

- 문제 54. 다음 코드들은 각각 Keras를 이용한 선형 회귀 모델을 구현한 코드와 Cds센서를 이용하여 밝기 값을 출력하는 코드입니다. 조도계를 사용하거나, 스마트폰에서 '조도계' 애플리케이션을 다운 받아 다음 문제를 해결하세요. (단, 조도계의 단위는 Lux로 합니다.)

```
01:     from tensorflow import keras
02:
03:     model = keras.models.Sequential()
04:
05:     model.add(keras.layers.Input(shape=(1,)))
06:     model.add(keras.layers.Dense(1))
07:
08:     model.compile(loss='MAE', optimizer='Adam')
```

```
01:     from pop import Cds
02:
03:     cds = Cds(7)
04:
05:     value = cds.readAverage()
```

연습문제

- A. 빈 배열 2개를 생성하고, Cds 값과 조도계 값을 동시에 측정하여 Cds 값 배열과 조도계 값 배열을 만들어보세요.
- B. Keras를 이용하여 A에서 만든 두 배열을 선형 회귀하는 코드를 작성하세요.
- C. 회귀 모델의 출력과 실제 조도계의 값을 비교해보고 데이터셋 추가 수집, 추가 학습 등 방법으로 ±30 lux 미만의 오차 범위를 갖는 회귀 모델을 만들어 보세요.

연습문제

- 문제 55. 다음 신경망 그림을 Keras로 구현해보세요.