케라스를 사용한 인공 신경만 소개 정리노트

4조-(이희구, 유제우)

※ 4조 의견

질문) 인공 신경망이 뭐야?

인공 신경망이란 뇌에 있는 생물학적 뉴런의 네트워크에서 영감을 받은 머신러닝 모델을 말한다. 딥러닝의 핵심이고, 강력하며 확정성이 좋다. 인공 신경망은 대규모 머신러닝 문제를 다루기에 적합하다. 그 예로, 수백만 개의 이미지를 분류하는 구글 이미지, 음성인식 서비스인 애플의 시리, 가장 좋은 비디오를 추천하는 유튜브, 스스로 기보를 익히면서 학습하는 딥마인드의 알파고 등이 있다.

질문) 인공 뉴런은 뭐야?

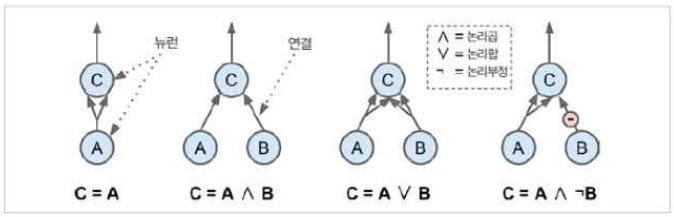


그림 10-3 간단한 논리 연산을 수행하는 인공 신경망

인공 뉴런은 생물학적 뉴런에 착안한 매우 단순한 신경망 모델이다. 하나 이상의 이진(on/off) 입력과 이진 출력 하나를 가지고, 입력이 일정 개수만큼 활성화되었을 때 출력을 내보낸다. 또한 인공 뉴런은 논리 연산을 수행해서 인공 뉴런의 네트워크를 만든다.

질문) 그럼 퍼셉트론은 뭐야?

가장 간단한 인공 신경망 구조로 1957년 프랑크 로젠블라트가 제안하였다. TLU 또는 LTU(linear threshold unit) 라 불리는 인공 뉴런 활용하고, 모든 입력은 가중치와 연결된다. 퍼셉트론에서 가장 널리 사용되는 계단 함수는 헤비사이드 계단 함수이다.

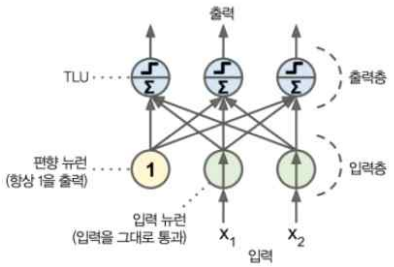
질문) TLU가 뭔데?

입력값과 가중치를 곱한 값들의 합에 계단함수(step function) 적용한 것이다.

질문) TLU와 선형 이진분류에 대해 설명해줘

하나의 TLU를 간단한 이진분류기로 활용 가능하다. 모든 입력값(특성)의 선형 조합을 계산한 후에 임곗값을 기준으로 양성/음성을 분류한다. 작동은 로지스틱 회귀 또는 선형 SVM 분류기와 비슷하다. TLU 모델 학습 = 최적의 가중치 $w\ i$ 를 찾기 이다.

질문) 퍼셉트론의 정의는?



[입력 두 개와 출력 세 개로 구성된 퍼셉트론]

하나의 층에 여러 개의 TLU로 구성된다. TLU 각각은 모든 입력과 연결된다.

질문) 퍼셉트론 학습 알고리즘은 어떻게 수행돼?

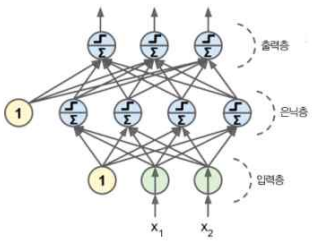
오차가 감소되도록 가중치를 조절하며 뉴런 사이의 관계를 강화시킨다. 하나의 샘플이 입력될 때 마다 예측한 후에 오차를 계산하여 오차가 줄어드는 방향으로 가중치를 조절한다.

질문) 퍼셉트론은 선형성이 있다는데?

■ 퍼셉트론 수렴 이론: 선형적으로 구분될 수 있는 모델은 언제나 학습이 가능하다.

각 출력 뉴런의 결정경계가 선형이다. 이로 인해, 복잡한 패턴 학습을 하지 못한다. 그러므로, 퍼셉트론은 매우 단순한 경우만 해결이 가능하다.

질문) 다층 퍼셉트론이 뭐야?



다층 퍼셉트론은 퍼셉트론을 여러 개 쌓아올린 인공신경망이다. 입력층 하나와 은닉층이라 불리는 하나 이상의 TLU층과 출력층으로 구성된다. 모든 층은 편향을 포함하며, 다음 층과 완전히 연결되어 있다.

질문) 심층 신경망은?
여러 개의 은닉층을 쌓아올린 인공신경망이다.

질문) 역전파 훈련 알고리즘이 뭐야?
다층 퍼셉트론은 층이 많을수록 훈련시키는 과정이 점점 더 어려워진다. 1986년에 소개된 역전파 훈련 알고리즘이 발표된 이후로 실용성을 갖출 수 있었다. 1단계 정방향으로 각 훈련 샘플에 대해 먼저 예측을 만든 후 오차를 측정한다. 2단계 역방향으로 각 층을 거치면서 각 출력연결이 오차에 기여한 정도를 측정한다. 3단계로 오차가 감소하도록 모든 가중치를 조정한다.

질문) MLP 특징에는 어떤 것들이 있을까?
랜덤하게 설정할 수 있고, 그렇지 않으면 층의 모든 뉴런이 동일하게 움직인다. 활성화 함수로 보통 계단함수 대신에 다른 함수를 사용한다. 그 예로, 로지스틱(시그모이드), 하이퍼볼릭 탄젠트 함수(쌍곡 탄젠트 함수), ReLU 함수 등이 있다.

질문) 활성화 함수 대체가 왜 필요해?
선형성을 벗어나기 위해서 필요하다. 선형 변환을 여러 개 연결해도 선형 변환에 머문다. 그럼 복잡한 문제 해결이 불가능하다. 비선형 활성화 함수를 충분히 많은 층에서 사용하면 매우 강력한 모델 학습이 가능하다.

질문) 회귀와 분류를 위한 다층 퍼셉트론에 대해 설명해줘
회귀를 위해 출력 뉴런 수를 예측해야 하는 값의 수에 따라 출력 뉴런을 설정한다. 출력값에 특별한 제한이 없다면 활성화 함수를 사용하지 않는다. 출력이 양수인 경우 ReLU 또는 softplus를 사용 가능하다. 손실함수로는 일반적으로 평균제곱오차 mse를 활용한다. 이상치가 많을 경우에는 평균절대값오차 mae를 사용할 수 있다.
분류를 위해서는 이진분류로 하나의 출력 뉴런을 사용할 수 있다. 활성화 함수로는 로지스틱 함수를 사용할 수 있다. 다중 레이블 이진분류로는 다층 퍼셉트론을 활용할 수 있다. 손실함수로는 크로스 엔트로피를 사용할 수 있다.

질문) 케라스?
모든 종류의 신경망을 손쉽게 만들어 주는 최상위 딥러닝 API 제공한다. 2015년 3월에 오픈소스로 공개되었다.

질문) 케라스 시퀀셜 API 활용한 이미지 분류 설명해줘

케라스를 사용하여 데이터셋 적재

```
fashion_mnist = keras.datasets.fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()
```

```
X_train_full.shape
```

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

```
X_valid, X_train = X_train_full[:5000] / 255., X_train_full[5000:] / 255.
y_valid, y_train = y_train_full[:5000], y_train_full[5000:]
X_test = X_test / 255.
```

시퀀셜 API를 사용하여 모델 만들기

- Sequential 클래스 내에 층을 쌓아 순차적 학습 지원
- 은닉층: 2개

```
model = keras.models.Sequential()
model.add(keras.layers.Flatten(input_shape=[28, 28]))
model.add(keras.layers.Dense(300, activation="relu"))
model.add(keras.layers.Dense(100, activation="relu"))
model.add(keras.layers.Dense(10, activation="softmax"))
```

- Sequential 모델을 만들때 층의 리스트를 전달

```
model = keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28, 28]),
    keras.layers.Dense(300, activation="relu"),
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(10, activation="softmax")
    ])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
flatten (Flatten)            (None, 784)               0
_________________________________________________________________
dense (Dense)                (None, 300)               235500
_________________________________________________________________
dense_1 (Dense)              (None, 100)               30100
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1010
=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss="sparse_categorical_crossentropy",
              optimizer="sgd",
              metrics=["accuracy"])
```

모델 컴파일에서 모델을 생성하려면 선언된 모델을 컴파일 수행해야 한다. 손실함수, 옵티마이저, 평가기준 등을 지정한다.

```
history = model.fit(X_train, y_train, epochs=30,
                    validation_data=(X_valid, y_valid))
```

```
Epoch 1/30
1719/1719 [==============================] - 5s 2ms/step - loss: 0.7237 - accuracy: 0.7644 - val_loss: 0.5207 - val_accuracy: 0.8234
Epoch 2/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.4843 - accuracy: 0.8318 - val_loss: 0.4345 - val_accuracy: 0.8538
Epoch 3/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.4393 - accuracy: 0.8455 - val_loss: 0.5310 - val_accuracy: 0.7986
Epoch 4/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.4126 - accuracy: 0.8566 - val_loss: 0.3918 - val_accuracy: 0.8644
Epoch 5/30
1719/1719 [==============================] - 3s 2ms/step - loss: 0.3940 - accuracy: 0.8621 - val_loss: 0.3753 - val_accuracy: 0.8680
Epoch 6/30
```

이후, fit() 함수를 사용해 모델을 훈련시킨다.
evaluate() 메서드를 호출하여 손실과 정확도를 계산해준다. 마지막으로 predict() 메서드를 호출하여 모델을 사용하여 예측을 진행한다.

질문) 케라스 시퀀셜 API 활용한 회귀 설명해줘

모델 생성, 훈련 및 평가

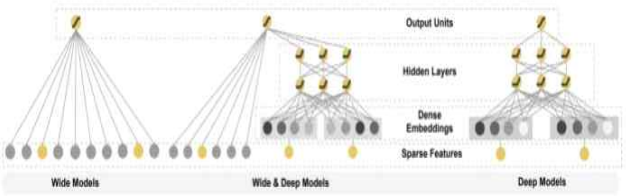
```
model = keras.models.Sequential([
    keras.layers.Dense(30, activation="relu", input_shape=X_train.shape[1:]),
    keras.layers.Dense(1)
])
model.compile(loss="mean_squared_error", optimizer=keras.optimizers.SGD(learning_rate=1e-3))
history = model.fit(X_train, y_train, epochs=20, validation_data=(X_valid, y_valid))
mse_test = model.evaluate(X_test, y_test)
X_new = X_test[:3]
y_pred = model.predict(X_new)
```

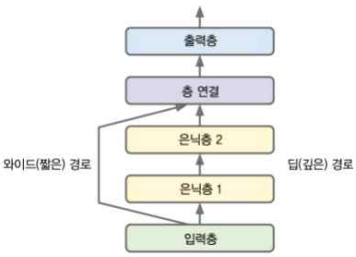
Sequential 클래스를 이용한 회귀용 MLP 구축은 분류용과 기본적으로 동일하다. 차이점으로는 하나의 값을 예측하기 때문에 출력층에 활성화 함수를 사용하지 않는 하나의 뉴런만 사용한다.

질문) 케라스 Sequential 클래스 활용의 장단점은 뭐가 있어?

사용하기 매우 쉬우며 성능이 우수하다. 입출력이 여러 개이거나 더 복잡한 네트워크를 구성하기 어렵다. Sequential 클래스 대신에 함수형 API, 하위클래스(subclassing) API 등을 사용하여 보다 복잡하며, 보다 강력한 딥러닝 모델 구축이 가능하다.

질문) 함수형 API를 사용해 더 복잡한 모델을 만드는 법이 있을까?





모든 레이블을 순차적으로 처리하는 것 대신 다양한 신경망 구축을 위해 함수형 API를 활용한다. 순차적이지 않는 신경망의 예는 와이드&딥(Wide & Deep) 신경망이 있다. 와이드&딥 구조는 깊게 쌓은 층을 사용한 복잡한 패턴과 짧은 경로를 사용한 간단한 규칙이고 모두 학습 가능한 모델이다.

질문) 서브클래싱 API로 동적 모델은 어떻게 만들어?

Sequential 클래스와 함수형 API 는 모두 선언적 방식으로 정적임이다. 한 번 선언되면 변경할 수 없는 모델을 생성한다. 모델 저장, 복사, 공유가 용이하고, 모델 구조 출력 및 모델 분석이 용이하다. 반복문, 조건문 등을 활용하여 동적 모델을 생성하고자 할 경우 명령형 프로그래밍 방식이 요구된다. 서브클래스 API을 활용하여 동적 모델 생성이 가능하다.

질문) 동적 모델의 단점은?

모델 구조가 call() 메서드 안에 숨겨져 있어서, 케라스가 분석하기 어렵다. 모델 저장 및 복사가 불가능하다. summary() 메서드 활용이 제한된다. 층의 목록만 확인 가능하며, 층간의 연결정보를 알 수 없다. 케라스가 타입과 크기를 미리 확인할 수 없어 실수 발생 많아질 수 있다. 높은 유연성이 필요하지 않다면 추천하지 않는다.

질문) 신경망 하이퍼파라미터 튜닝은 어떻게 해?

케라스 모델 생성함수

```
def build_model(n_hidden=1, n_neurons=30, learning_rate=3e-3, input_shape=[8]):
    model = keras.models.Sequential()
    model.add(keras.layers.InputLayer(input_shape=input_shape))
    for layer in range(n_hidden):
        model.add(keras.layers.Dense(n_neurons, activation="relu"))
    model.add(keras.layers.Dense(1))
    optimizer = keras.optimizers.SGD(learning_rate=learning_rate)
    model.compile(loss="mse", optimizer=optimizer)
    return model
```

신경망은 조정해야할 하이퍼파라미터가 매우 많다. 그리드 탐색/랜덤 탐색을 활용해야 한다. 케라스 모델을 사이킷런의 추정기처럼 보여주어야 한다. KerasRegressor 클래스를 이용하여 케라스 모델을 사이킷런 모델처럼 작동하게 만들 수 있다. KerasRegressor 클래스의 객체를 생성할 때 아래 build_model() 함수와 같이 케라스 모델을 생성하는 함수를 입력해 주어야 한다.

KerasRegressor 클래스 활용 → build_model() 함수로 만들어진 케라스 모델을 감싸는 레퍼이다.

랜덤 탐색 활용 → 하이퍼파라미터가 많으므로 그리드 탐색보다 랜덤 탐색이 좋다. 랜덤 탐색은 하드웨어와 데이터셋의 크기, 모델의 복잡도, n_iter, cv 매개변수에 따라 시간이 걸린다.

질문) 더 좋은 방법은 없어?

랜덤탐색 기법은 매우 오래 걸린다. 하이퍼파라미터 최적화에 사용할 수 있는 다양한 라이브러리가 개발되었다. Hyperopt Hyperas, kopt, Talos Keras Tuner, Scikit-Optimize(skopt), Spearmint, Hyperband, Sklearn-Deap, 진화 알고리즘, AutoML

질문) 하이퍼파라미터를 조정

과대적합이 발생할 때까지 은닉층 수를 늘려가며 훈련시킬 수 있다. 과대적합되지 않도록 조기종료나 규제 기법 사용을 추천한다. 반복마다 일정한 값을 학습률에 곱하여 반복 수행, ▪ 고전적인 미니배치 경사하강법 보다 더 좋은 성능의 옵티마이저 사용 가능하다. 배치크기는 모델 성능과 훈련 시간에 큰 영향을 미칠 수 있다. 활성화 함수로는 은닉층은 ReLU 가 일반적이고, 출력층은 모델에 따라 다르다. 반복횟수(에포크)는 조기종료 기법을 사용을 추천한다. 최적의 학습률은 다른 하이퍼파라미터에 의존적아다. 다른 하이퍼파라이머터를 조정한 경우, 학습률도 조정해야 한다.

※ 저희 조는 퍼셉트론과 케라스를 통한 분류와 회귀에 대해 조사하고, 이를 바탕으로 어떤 상황에서 어떤 모델을 선택해야 하는지에 대해 고민해 보았습니다.

[퍼셉트론을 이용한 분류와 회귀]

- 퍼셉트론은 간단하면서도 효과적인 모델로, 분류와 회귀 문제에 활용할 수 있습니다.

< 1.1 분류 >

퍼셉트론을 이용한 분류는 주로 이진 분류에 적합합니다. 간단한 결정 경계를 통해 두 개의 클래스로 데이터를 분류할 수 있습니다.

적용 예시: 스팸 메일 여부를 판별하는 문제에서, 퍼셉트론은 이메일의 특징을 기반으로 스팸 여부를 식별할 수 있습니다.

< 1.2 회귀 >

퍼셉트론은 선형 회귀 모델로 활용할 수 있습니다. 출력이 실수값이므로 회귀 문제에 적합합니다.

적용 예시: 단순한 선형 회귀 문제인 주택 가격 예측에서, 퍼셉트론은 주택의 특징을 기반으로 가격을 예측할 수 있습니다.

[케라스를 활용한 분류와 회귀]

- 케라스는 다양한 신경망 모델을 쉽게 구축할 수 있는 강력한 도구입니다.

< 2.1 분류 >

케라스의 Sequential API를 사용하면 간단한 분류 모델을 구성할 수 있습니다. 이미지 분류와 같은 문제에 적합합니다.

적용 예시: Sequential API를 활용하여 간단한 신경망을 구현하여 스팸 이미지를 분류할 수 있습니다.

< 2.2 회귀 >

회귀 모델도 Sequential API로 쉽게 구현 가능합니다. 주가 예측과 같은 문제에 활용할 수 있습니다.

적용 예시: 주가 예측을 위해 Sequential API를 사용하여 간단한 회귀 모델을 만들 수 있습니다.

[모델 선택과 결론]

모델 선택은 주어진 문제의 특성과 복잡성을 고려하여 신중히 결정되어야 하는 중요한 과정입니다. 각 모델은 특별한 장점을 가지고 있어, 퍼셉트론과 케라스 각각의 특성을 고려하여 최적의 선택이 필요합니다.

퍼셉트론은 간단하면서도 직관적인 모델로, 주로 선형 문제에 적합합니다. 이 모델은 단순하고 이해하기 쉬우며, 특히 데이터가 선형적으로 구분될 때 효과적으로 활용될 수 있습니다.

반면에 케라스는 다양한 신경망 구조를 쉽게 구현할 수 있는 강력한 라이브러리입니다. Sequential API를 통해 간단한 모델부터, 함수형 API 및 서브클래싱 API를 이용하면 더 복잡하고 다양한 아키텍처를 구성할 수 있습니다. 특히 복잡한 패턴 및 특징을 학습하는 데 우수한 성능을 발휘합니다.

문제에 따라 퍼셉트론 또는 케라스를 선택함으로써 최적의 모델을 구축할 수 있습니다. 단순한 선형 문제에는 퍼셉트론이 효과적이지만, 데이터의 복잡성이나 다양한 특징을 고려해야 하는 경우에는 케라스를 활용하는 것이 더 적절할 것입니다. 이러한 모델 선택은 항상 주어진 문제 도메인과 데이터 특성을 고려하여 신중히 결정되어야 하며, 실험을 통해 최적의 모델을 찾는 것이 중요합니다.