딥러닝

딥러닝 모델의 학습 순서

1. 학습용 feature 데이터를 입력하여 예측값 구하기(순전파)
2. 예측값과 실제값 사이의 오차 구하기(LOSS 계산)
3. LOSS를 줄일 수 있는 가중치 업데이트 하기(역전파)
4. 1~3번 반복으로 LOSS를 최소로 하는 가중치 얻기

텐서플로우로 딥러닝 구현하기

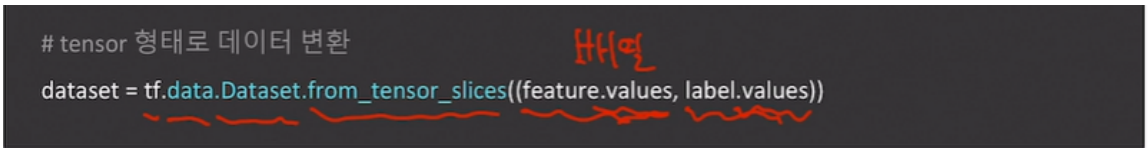
▼ 데이터 전 처리하기

Tensor 형태의 데이터로 입력을 받음

Tensor란 다차원 배열로 TensorFlow에서 사용하는 객체

1차원 벡터 2차원 매트릭스 3차원~ 텐서

배열로 만드는것이 중요 (df.value)



추가적인 전 처리 ⇒ Epoch, Batch

Epoch: 한번의 epoch는 전체 데이터 셋에 대해 한 번 학습을 완료한 상태

Batch: 나눠진 데이터 셋

Ex) 총 데이터 1000개, Batch size = 100

1 iteration = 100개 데이터에 대해서 학습

1 epoch = 1000 Batch size = 10 iteration

Iteration: epoch를 나누어서 실행하는 횟수

▼ 딥러닝 모델 구축하기

Keras

모델 클래스 객체 생성

```
tf.keras.models.Sequential()
```

모델의 각 Layer 구성

```
tf.keras.layers.Dense(units, activation)
```

- units : 레이어 안의 Node의 수
- activation : 적용할 activation 함수 설정

input Layer는 입력 형ㅌ에 대한 정보를 필요로함

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(10, input_dim=2, activation='sigmoid'), # 2개의 입력 변수, 10개의 노드
    tf.keras.layers.Dense(10, activation='sigmoid'), # 10개의 노드
    tf.keras.layers.Dense(1, activation='sigmoid'), # 1개의 노드
])
```

모델에 Layer 추가하기

```
[model].add(tf.keras.layers.Dense(units, activation))
```

- units : 레이어 안의 Node의 수
- activation : 적용할 activation 함수 설정

Ex)

```
model = tf.keras.models.Sequential()

model.add(tf.keras.layers.Dense(10, input_dim=2, activation='sigmoid'))
model.add(tf.keras.layers.Dense(10, activation='sigmoid'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

모델 학습 방식을 설정하기 위한 함수

```
[model].compile(optimizer, loss)
```

- optimizer : 모델 학습 최적화 방법
- loss : 손실 함수 설정

회귀 MSE

분류 Cross Entropy

모델을 학습시키기 위한 함수

```
[model].fit(x, y)
```

- x : 학습 데이터
- y : 학습 데이터의 label

▼ 모델 학습시키기

▼ 평가 및 예측하기

이미지 처리를 위한 데이터 전처리

합성곱 신경망 (Convolution Neural Network)

작은 필터를 순환시키는 방식

이미지의 패턴이 아닌 특징을 중점으로 인식

Ex) 귀 + 코 + 수염 + 꼬리 ⇒ 고양이

합성곱 신경망의 구조

입력 이미지의 특징을 추출, 분류하는 과정으로 동작

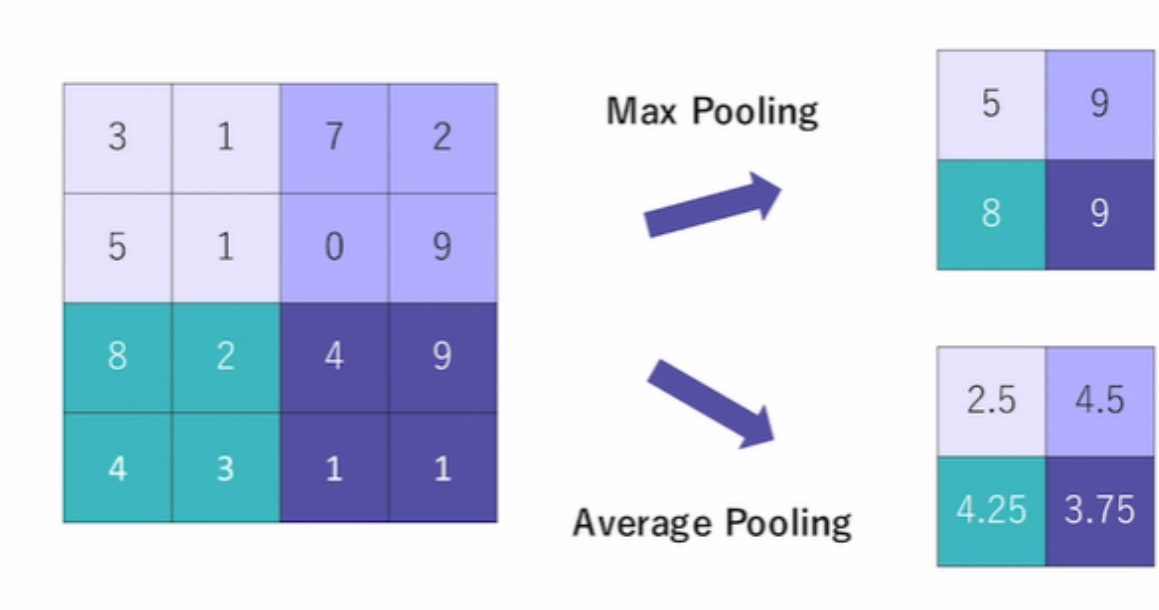
이미지에서 어떠한 특징이 있는지를 구하는 과정

필터(귀, 코, 수염등)가 이미지를 이동하며 새로운 이미지를 생성

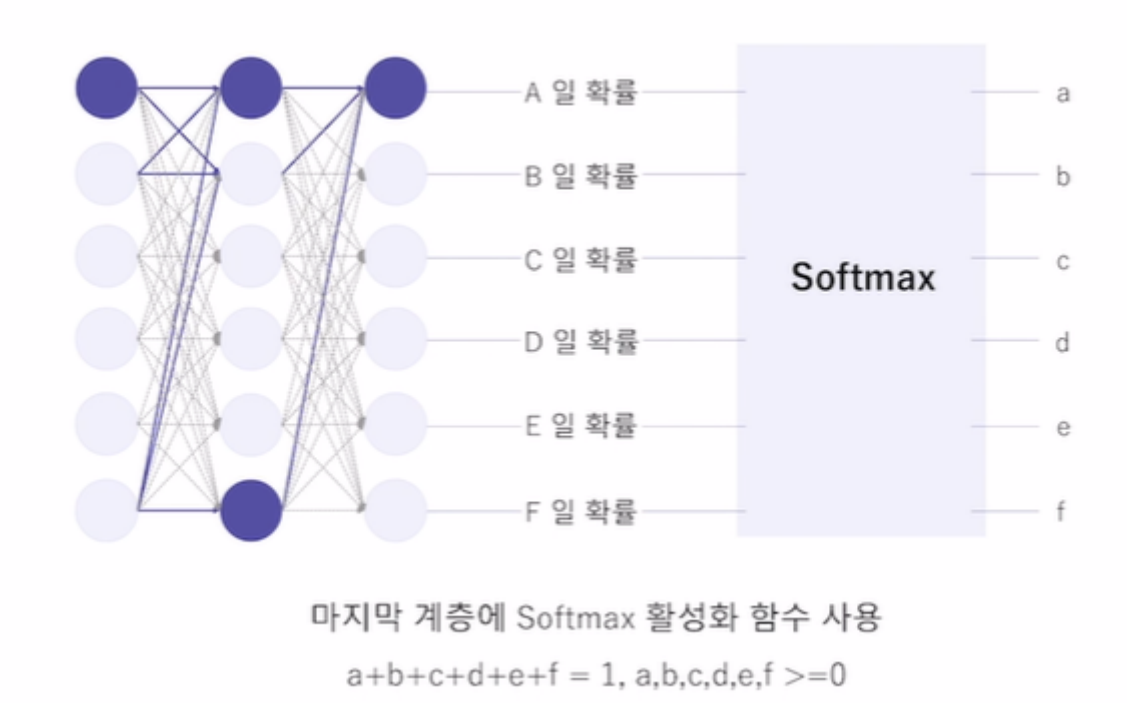
Padding: 원본 이미지의 상하좌우에 한 줄씩 추가

Striding 필터를 이동시키는 거리 설정

Pooling Layer: 이미지의 왜곡의 영향(노이즈)를 축소하는 과정 ↓



분류를 위한 Softmax 활성화 함수



마지막 계층에 Softmax 활성화 함수 사용

a+b+c+d+e+f = 1, a,b,c,d,e,f >=0

정리

합성곱(특징) → 풀링(사이즈, 노이즈) → 활성함수(분류)

Convolution Layer는 특징을 찾아내고, Pooling Layer 는 처리할 맵(이미지) 크기를 줄여준다. 이를 N번 반복한다.

반복할 때마다 줄어든 영역에서의 특징을 찾게 되고, 영역의 크기는 작아졌기 때문에 빠른 학습이 가능해진다.

Keras에서 CNN 모델을 만들기 위해 필요한 함수/메서드

CNN 레이어

tf.keras.layers.Conv2D(filters, kernel_size, activation, padding)

: 입력 이미지의 특징, 즉 처리할 특징 맵(map)을 추출하는 레이어입니다.

- filters : 필터(커널) 개수
- kernel_size : 필터(커널)의 크기
- activation : 활성화 함수
- padding : 이미지가 필터를 거칠 때 그 크기가 줄어드는 것을 방지하기 위해서 가장자리에 0의 값을 가지는 픽셀을 넣을 것인지 말 것인지를 결정하는 변수. ‘SAME’ 또는 ‘VALID’

Maxpool 레이어

tf.keras.layers.MaxPool2D(padding)

: 처리할 특징 맵(map)의 크기를 줄여주는 레이어입니다.

- padding : ‘SAME’ 또는 ‘VALID’

Faltten 레이어

tf.keras.layers.Flatten()

: Convolution layer 또는 MaxPooling layer의 결과는 N차원의 텐서 형태입니다. 이를 1차원으로 평평하게 만들어줍니다.

Dense 레이어

tf.keras.layers.Dense(node, activation)

- node : 노드(뉴런) 개수
- activation : 활성화 함수

평가 방법

model.evaluate(X, Y)

evaluate() 메서드는 학습된 모델을 바탕으로 입력한 feature 데이터 X와 label Y의 loss 값과 metrics 값을 출력합니다.

예측 방법

model.predict_classes(X)

X 데이터의 예측 label 값을 출력합니다.

자연어 처리를 위한 데이터 전 처리

Ex) 기계 번역 모델, 음성 인식(시리, 빅스비)

자연어 처리 과정

1. **자연어 전 처리**
    a. Noise canceling(오류 교정)

       자연어 문장의 스펠링 체크 및 띄어쓰기 오류 교정

    b. Tokenizing(토큰화)

문장을 토큰으로 나눔, 토큰은 어절, 단어등으로 목적에 따라 다르게 정의

Ex) "딥러닝 기초 과목을 수강하고 있습니다"

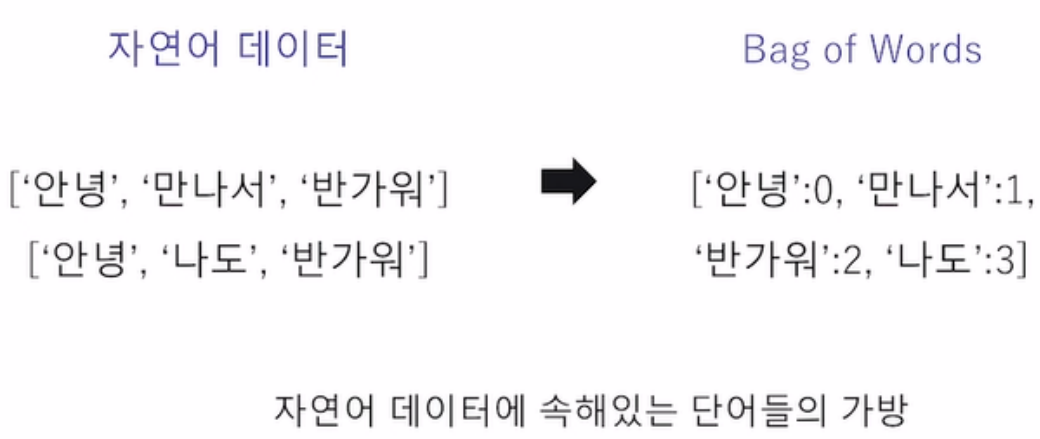
⇒ "딥" "러닝" "기초" "과목" "을" "수강" "하고" "있습니다"

c. StopWord removal(불용어 제거)

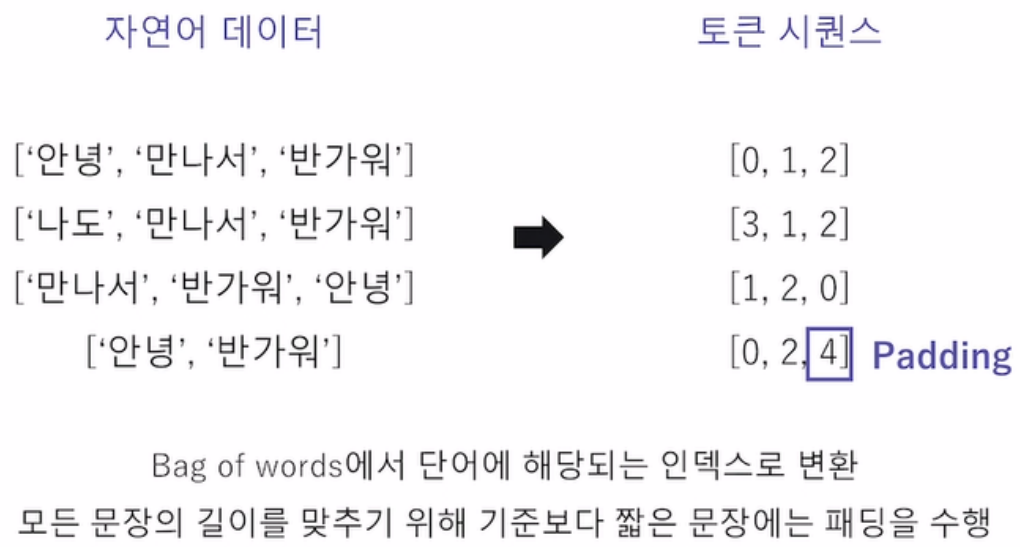
불필요한 단어를 의미하는 불용어 제거

Ex) 아, 휴, 아이쿠, 아이고, 그렇지 않으면, 그러나, 그런데, 하지만

d. Bag of Words



e. 토큰 시퀀스



⇒ Padding을 하는이유 = 통일 / 2개짜리를 Padding하여 3개로 만들어야함

f. Word Embedding(워드 임베딩)

단순하게 Bag of Words의 인덱스로 정의된 토큰들에게 의미를 부여하는 방식

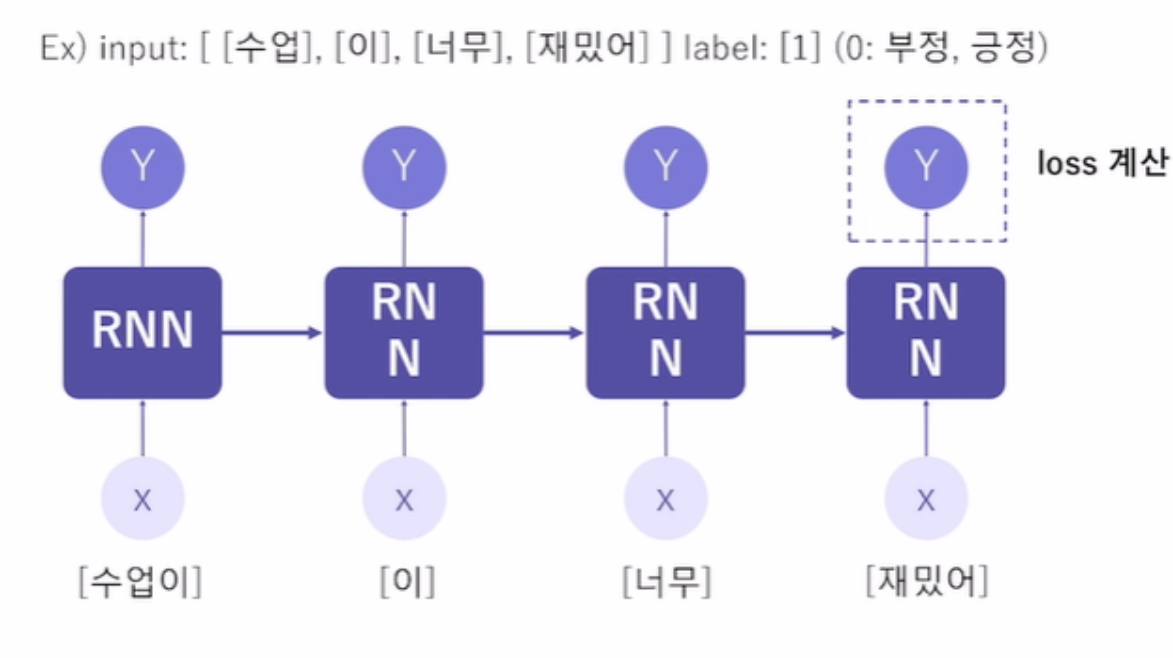
유사한 관계를 가지고 있는 벡터들끼리는 비슷한 벡터로 만들어짐

2. **단어표현**
3. **모델 적용하기**

자연어 분류를 위한 순환 신경망(Recurrent Neural Network) RNN

기존 퍼셉트론 계산과 비슷하게 X 입력 데이터를 받아 Y를 출력

출력 값을 두 갈래로 나뉘어 신경망에게 '기억'하는 기능을 부여



정리

임베딩 ⇒ RNN ⇒ 활성함수

임베딩은 토큰의 특징을 찾아내고, RNN 이 전 토큰의 영향을 받으며 학습한다.

Keras에서 RNN 모델을 만들기 위해 필요한 함수/라이브러리

일반적으로 RNN 모델은 입력층으로 Embedding 레이어를 먼저 쌓고, RNN 레이어를 몇 개 쌓은 다음, 이후 Dense 레이어를 더 쌓아 완성합니다.

임베딩 레이어

`tf.keras.layers.Embedding(input_dim, output_dim, input_length)`

들어온 문장을 단어 임베딩(embedding)하는 레이어

- `input_dim`: 들어올 단어의 개수
- `output_dim`: 결과로 나올 임베딩 벡터의 크기(차원)
- `input_length`: 들어오는 단어 벡터의 크기

RNN 레이어

`tf.keras.layers.SimpleRNN(units)`

: 단순 RNN 레이어

- `units`: 레이어의 노드 수

평가 방법

`model.evaluate(X, Y)`

evaluate() 메서드는 학습된 모델을 바탕으로 입력한 feature 데이터 X와 label Y의 loss 값과 metrics 값을 출력합니다.

예측 방법

`model.predict(X)`

X 데이터의 예측 label 값을 출력합니다.