23 하계 학부연구생 프로그램

ch07. 인공 신경망

12223547 박혜민

인공신경망으로 패션아이템 분류하기

01 tensorflow 와 keras

02 optimizer

- SGD
- 적응적 합습률 옵티마이저

03 원-핫 인코딩

04 과대적합 방지

- optimizer
- drop out
- early stopping

Tensorflow vs Keras

구글에서 개발한 딥러닝 프레임워크

- Tensorflow에서 실행되는 high level API
- 현재 tensorflow 위에서만 작동
- tensorflow 와 같은 딥러닝 라이브러를 사용하여 딥러닝의 기본 연산 수행

keras

특징

- 딥러닝 모델 구축을 위한 모델 수준의 라이브러리이며, 연산수행은 벡엔드 엔진으로 텐서 라이브러리 사용함
- 거의 모든 종류의 딥러닝 모델을 간편하게 만들고 훈련시킬 수 있음

- 개발 흐름
  1. 입력 텐서와 타깃 텐서로 이루어진 훈련 데이터를 정의
  2. 입력과 타깃을 매핑하는 층으로 이루어진 네트워크(또는 모델)
  3. 손실 함수, 옵티마이저, 모니터링하기 위한 측정 지표를 선택하
  4. 훈련 데이터에 대해 모델의 fit() 메서드를 반복적으로 호출

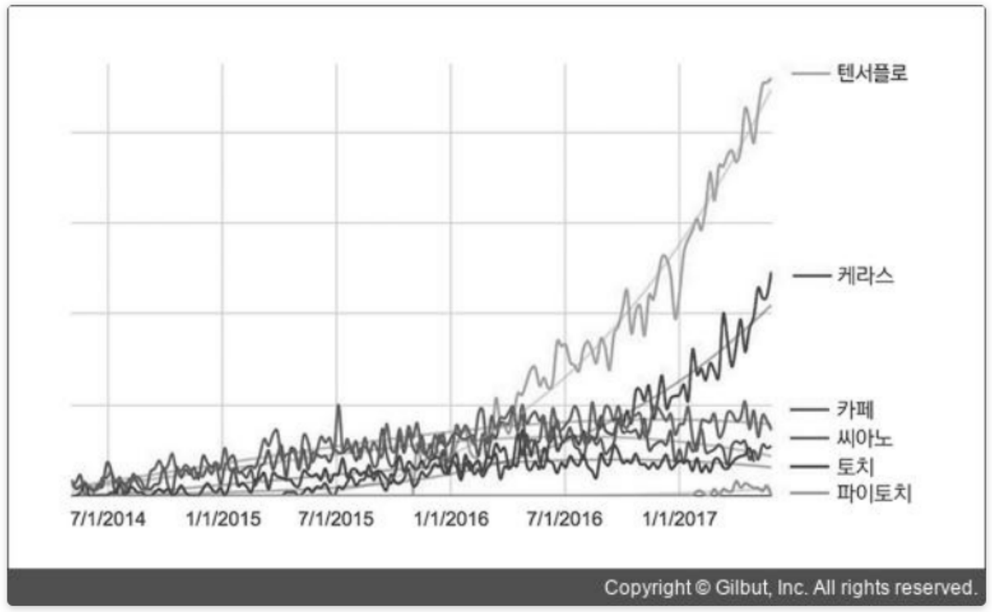


그림 3-2 딥러닝 프레임워크에 대한 구글 웹 검색 트렌드[9]

keras

모델을 정의하는 두 가지 방법

1. Sequential 클래스

- layer을 순서대로 쌓아 올린 네트워크

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(784,)))
model.add(layers.Dense(10, activation='softmax'))
```

첫 번째 층 * 입력 데이터 크기 전달
두번째 층

keras

모델을 정의하는 두 가지 방법

2. Functional API

- 각 층을 함수로서 정의 *입력 데이터의 크기를 인자로 입력 층 정의
- 이전층을 다음층 함수의 입력으로 사용, 변수로 할당
- Model() 함수에 입출력 정의

- Sequential 클래스와 달리 여러 입출력을 사용하거나 층을 공유하는 복잡한 모델 설계 가능

```
inputs = Input(shape=(10,))
hidden1 = Dense(64, activation='relu')(inputs)
hidden2 = Dense(64, activation='relu')(hidden1)
output = Dense(1, activation='sigmoid')(hidden2)
model = Model(inputs=inputs, outputs=output) # <-
```

```
# 두 개의 입력층을 정의
inputA = Input(shape=(64,))
inputB = Input(shape=(128,))

# 첫번째 입력층으로부터 분기되어 진행되는 인공 신경망을
x = Dense(16, activation="relu")(inputA)
x = Dense(8, activation="relu")(x)
x = Model(inputs=inputA, outputs=x)

# 두번째 입력층으로부터 분기되어 진행되는 인공 신경망을
y = Dense(64, activation="relu")(inputB)
y = Dense(32, activation="relu")(y)
y = Dense(8, activation="relu")(y)
y = Model(inputs=inputB, outputs=y)

# 두개의 인공 신경망의 출력을 연결(concatenate)
result = concatenate([x.output, y.output])

# 연결된 값을 입력으로 받는 밀집층을 추가(Dense layer)
z = Dense(2, activation="relu")(result)
# 선형 회귀를 위해 activation=linear를 설정
z = Dense(1, activation="linear")(z)

# 결과적으로 이 모델은 두 개의 입력층으로부터 분기되어
model = Model(inputs=[x.input, y.input], outputs=z)
```

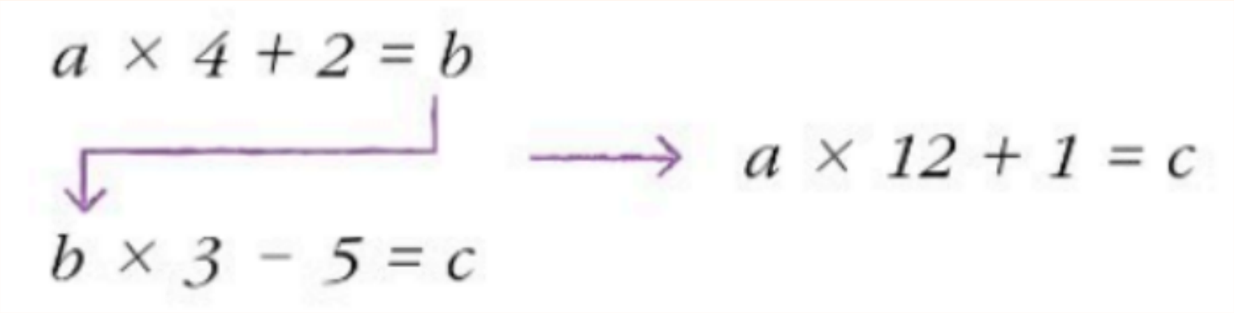
keras

활성함수

- 모든 은닉층에 활성함수 존재

은닉층에 활성함수가 있는 이유?

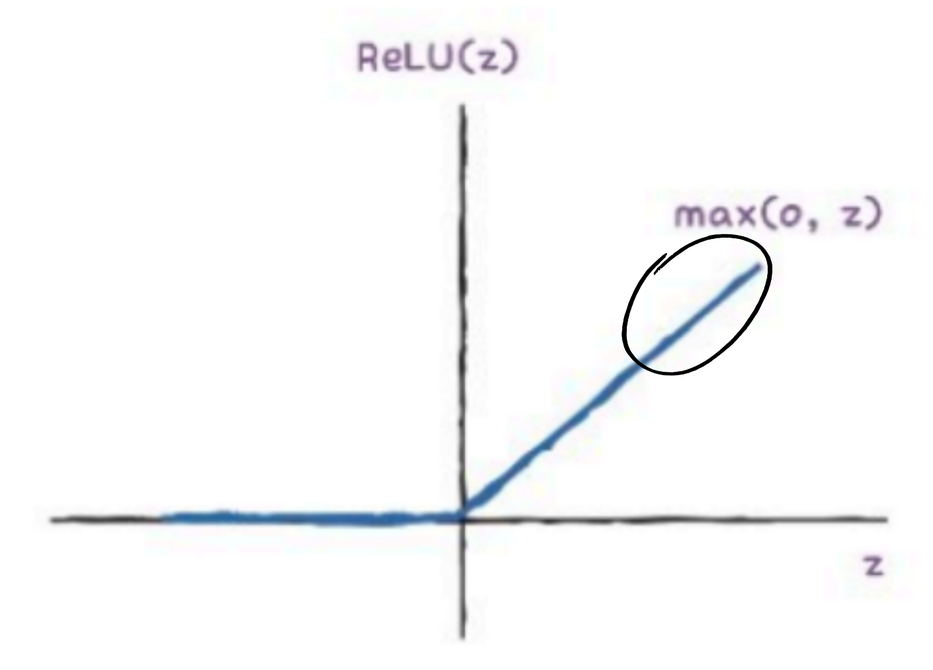
- 활성화 함수 없이 선형계산만 이뤄진다면 수행 역할의 의미가 없어지므로 비선형적으로 변형해주기 위해



keras

활성함수

렐루(ReLu)함수



$$ReLU(x) = max(0, x)$$

- 입력값이 양수면 입력값 출력, 아니면 0 출력
- 양수 입력값에 대해서 기울기가 항상 1

렐루(ReLu)함수를 은닉층에서 많이 사용하는 이유

1. 기울기 소실 문제 방지
2. 계산이 단순하여 학습속도가 빠름

Optimizer

최적화 알고리즘

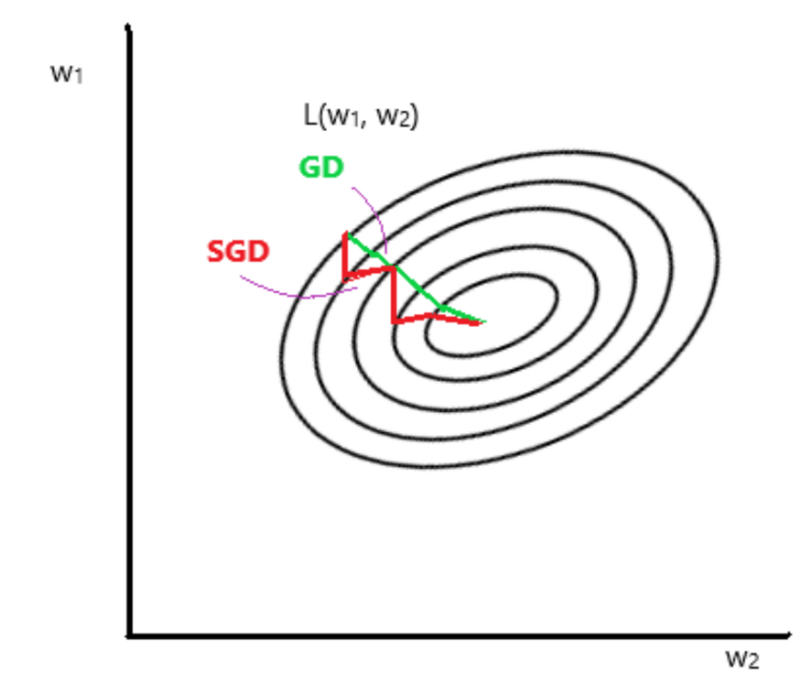
최적화란?

- loss function의 최소값을 찾아가는 것

Gradient descent 종류

- Batch Gradient Descent(GD)
- Stochastic Gradient Descent(SGD)
- Mini-batch Gradient Descent

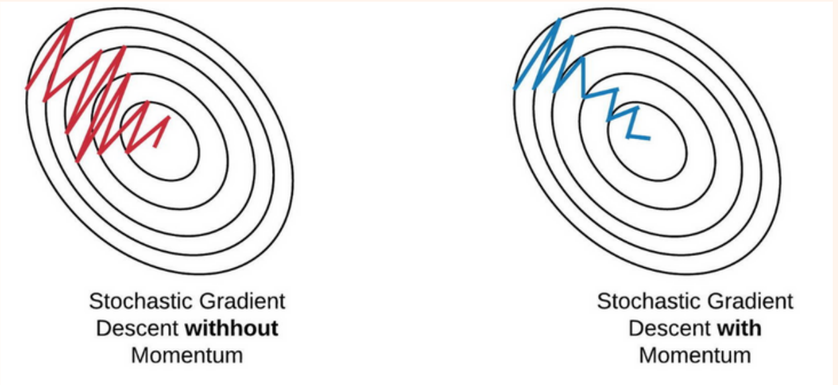
$$\theta_{t+1} = \theta_t - \eta \nabla_\theta J(\theta)$$



Optimizer

최적화 알고리즘

Momentum



- 현재 step 에서 과거의 이동한 gradient 를 더해줌
- 관성이 붙어 Local optimum 위험 감소
- 궤적이 크게 변하지 않아 안정적으로 감소

Optimizer

최적화 알고리즘

Nesterov Accelerated Gradient(Nag)

- 현재 위치에서 Momentum만 계산하여 이동하고, 그 지점을 기준으로 Gradient 계산하여 이동
- 모멘텀과 같이 과도한 업데이트가 일어나지 않아 최적값을 지나치는 위험 감소

Optimizer

최적화 알고리즘

Adagrad

- 현시점까지 계산된 모든 Gradient 의 제곱합에 반비례하는 학습률을 가짐
- 학습이 진행될 수록 학습률 감소
- 최저점에 도달하기도 전에 이동량이 줄어들 수 있다는 단점 존재

RMSProp

- 과거 Gradient 값들과 현재 Gradient 값들의 비율을 하이퍼 파라미터로 조절
- 새로운 gradient 값을 크게 반영하여 Adagrad의 단점 보완

Optimizer

최적화 알고리즘

Adam

- RMSProp + Momentum 으로 적응적 학습률과 Adaptive Learning rate이 합쳐짐
- 각각 관련 하이퍼 파라미터 가짐

one-hot encoding

- 타깃값을 해당 클래스만 1이고 나머지는 0인 배열로 만드는 것

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

one-hot encoding을 사용하는 이유

- 모든 클래스 간의 관계를 균등하게 분배할 수 있음

label encoding에서의 MSE) Apple과 Chicken의 오차 < Apple과 Broccoli의 오차

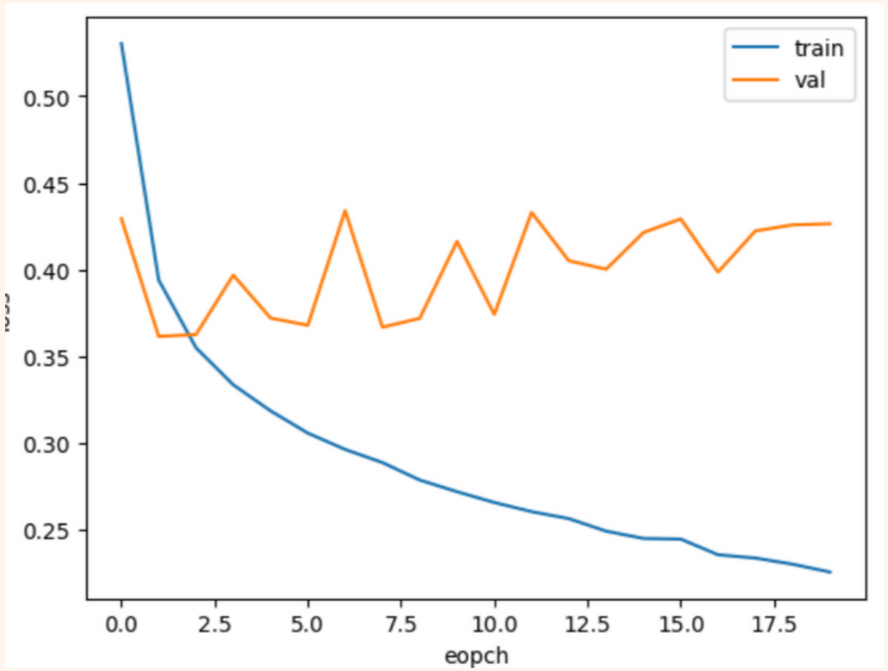
One-Hot Encoding에서의 MSE) 어떤 클래스끼리 비교해도 동일

- 자연어처리에서 문자 데이터를 숫자로 변환해야 하기 때문
- 직관적으로 파악 가능

과대적합 방지

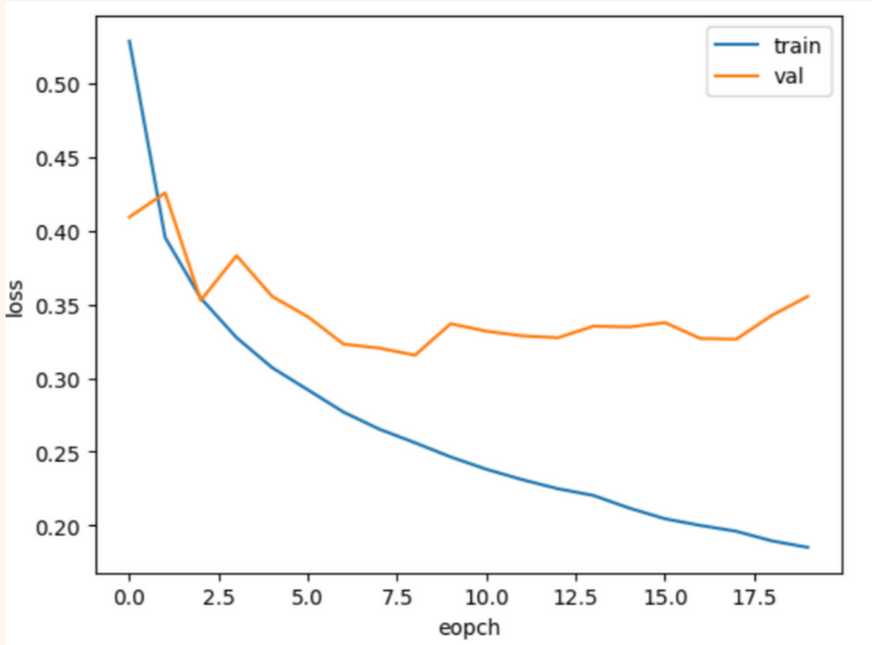
Optimizer 조정

<RMSprop>



- epoch 증가할수록 검증 셋 손실 증가, 훈련 셋 손실 감소하는 과대적합 모델

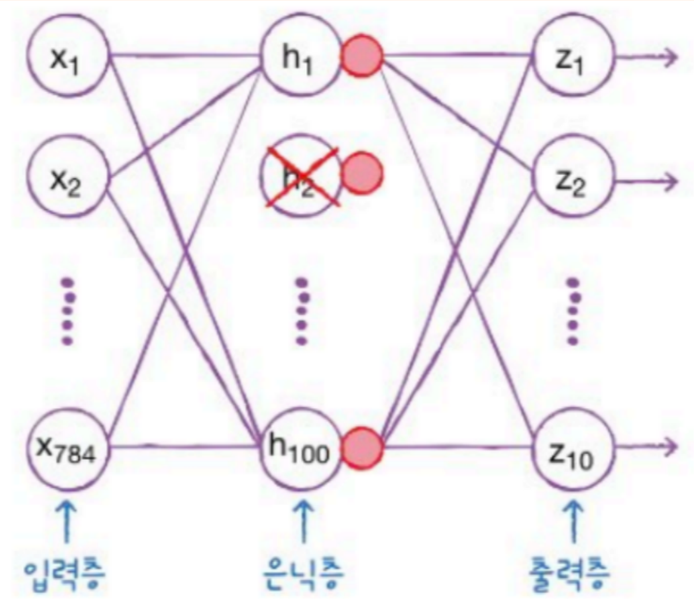
<Adam>



- 적응적 학습률 사용하여 과대적합 완화

Drop out

훈련 과정에서 층에 있는 일부 뉴런을 랜덤하게 꺼서(출력을 0으로 만들어)과대적합을 막음

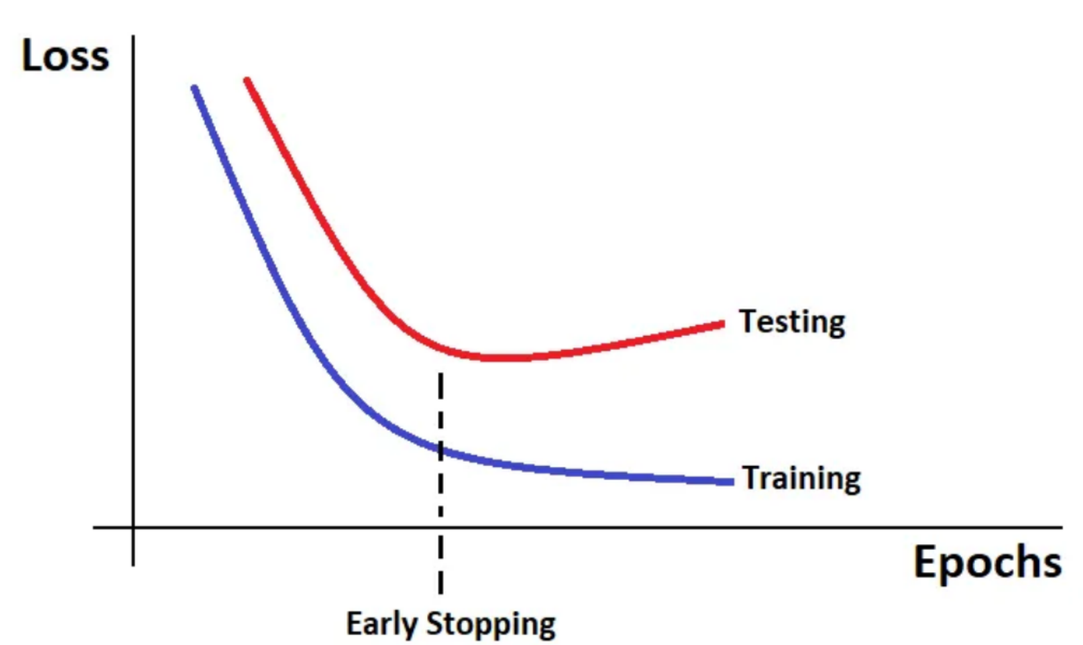


드롭아웃이 과대적합을 막을 수 있는 이유

- 너무 많은 특성이 존재할 경우 이 중 일부만 사용해도 충분한 결과 도출 가능
- 랜덤하게 노드를 끄는 것을 반복하면 모든 입력에 가중치를 잘 분산시키는 일반화된 모델 도출
- 검증 셋, 테스트셋에선 드롭아웃 적용 X
- 제거 비율을 하이퍼 파라미터로 조정
- 앙상블 학습과 유사한 효과

Early stopping

검증세트가 가장 낮은 지점에서 훈련을 종료



call back 객체

- 훈련 과정 중에 어떤 작업을 수행할 수 있게 하는 객체
- patience 매개변수로 검증 점수가 향상되지 않더라도 참을 에포크 횟수 지정