**스키마**

* 이전시간 활용 개념 복습
  + 크로스엔트로피손실함수의 과정:
    - 소프트 맥스를 거쳐 도출한 클래스별 확률값에 로그를 취하고 타깃값들과 곱함
      * 타깃값들 중 확률값의 클래스에 해당하는 타깃값들은 1로, 그 외의 타깃값들은 0으로 구성되어야 함 = 원핫인코딩
  + 에포크(epochs) : 확률적 경사하강법에서 전체 데이터를 프로세스하는 횟수
* 딥러닝관련 개념 및 기법들
  + 딥러닝에서는 교차검증을 사용하지 않고 훈련세트와 테스트 세트로만 분리
    - 데이터의 양이 충분한 경우가 많으며 계산양이 많아 자원을 절약해야 함
  + 머신러닝과 딥러닝의 차이(=딥러닝을 사용하는 이유)
    - 랜덤포레스트기법은 트리들이 독립적으로 훈련
* 인공신경망 같은 경우엔 은닉층에서 한번에 훈련되며 층을 추가하여 단계별 학습가능
  + - 그레이디언트 부스팅이나 스태킹(앙상블 모델에 여러 단계를 추가하여 학습) 같은 경우엔 단계를 순차적으로 학습 but 층을 여러 개 추가하기 어려움
* 여러 층 추가가능(= deep)
* 표현학습 모델: 비정형 데이터의 특징을 뽑아 다음층으로 전달하는 방식을 사용(=특성공학을 수동적으로 뽑았던 것을 어느정도 자동으로 진행해줌)
  + **패션 MNIST**: 가장 대표적인 딥러닝 연습용 데이터셋
    - 0~9까지의 숫자로 지정된 10개의 의류 클래스와 28\*28의 픽셀값으로 구성된 6만개의 샘플을 포함 (60000, 28, 28)
    - 사용방법:
      * 불러와서 훈련세트와 테스트세트로 분리
        + From tensorflow import keras

(train\_input, train\_target), (test\_input, test\_target) = \keras.datasets.fashion.mnist.load\_data()

* + - * 입력과 타깃 샘플 살펴보기
        + Fig, axs = plt.subplot()와 for루프를 활용하여 10개의 이미지 출력해보기
        + Train\_target 도 10개정도 값 출력해 보기 \*이미지데이터에서 필수
        + 클래스별로 샘플이 몇 개씩 들어있는 지 확인해보기 \*필수

Np.unique(train\_target, return\_counts=True))

* + **인공신경망:** 딥러닝의 기반이 되는 알고리즘으로 로지스틱회귀모델과 완전히 동일한 체계를 지님
    - 로지스틱회귀를 사용한 분류 과정 및 원리:
      * 훈련 데이터 스케일링:
        + 이미지 파일 같은 경우엔 굳이 스케일링을 할 필요없이 편의를 위해 255로 나누어 0과 1 사이의 값으로 변화하는 정도로
      * 훈련 데이터에 reshape()을 사용하여 1차원배열의 묶음으로
      * SGDClassifier: 확률적경사하강법으로 진행되는 로지스틱손실함수를 이용한 로지스트 회귀 모델 적용
        + 과정과 관련한 개념:

로지스틱 회귀 적용

픽셀 하나하나에 가중치를 곱하고 마지막에 절편을 더하는 과정을 모든 클래스에 대해 반복하여 z값 도출

\*각 클래스의 각각의 픽셀에 대해 가중치가 서로 다르게 적용됨

Ex) 티셔츠 – 픽셀1\*w1, 바지 픽셀2\*w1’

현재 다중분류이므로 10개의 이진분류를 수행

클래스 하나를 양성 / 나머지를 음성으로 두는 과정을 모든 클래스에 한해 반복하여 클래스의 개수만큼 z값을 도출(OVR/OVA)

소프트맥스 함수를 적용하여 확률로 변환한다.

* + - 인공신경망과 비교
      * 인공신경망의 구조
        + 입력층과 출력층으로 구분되어 입력층을 대상으로 학습

입력층: 입력데이터(현 과정에선 픽셀) \*객체로 지정하여 입력시킬 떄도 있음

출력층: 출력데이터로 뉴런/유닛이라고 불림(현과정에선 클래스에 따른 z값들)

* + - * + 학습과정

클래스에 따라 입력층에 있는 픽셀들 하나하나에 각자 다른 가중치를 부여하는 로지스틱 회귀 과정을 거쳐 여러 z값 도출. \*그림으로 시각화 할 때 절편도 모델에 포함되지만 편의상 그림에 표기하지 않는 경우가 많음

* + - 주요 라이브러리(2021년 기준):
      * 텐서플로/케라스
    - **케라스 모델**: 가장 기본적인 딥러닝 모델
      * 생성 과정(패션 MNIST 기반)
        + Train\_test\_split()을 활용하여 훈련세트(80) 테스트 세트(20%) 분리

매개변수: test\_size = : 테스트세트의 비율 설정

* + - * + 출력층의 유닛 개수 지정 \*항상 클래스의 개수와 동일해야함

Kera.layers.Dense()

Dense(밀집층, 완전연결층): 케라스의 가장 기본이 되는 층

케라스의 층들은 모두 layers 안에 포함됨

매개변수

10 : 시작부분에 작성하여 유닛의 개수를 지정해줌

Activation = : 적용할 분류기를 지정하는 것으로 현 상황에선 다중분류이므로 소프트맥스함수 사용

‘softmamx’ : 소프트맥스

‘sigmoid’ : 시그모이드

Input\_shape=(,) : 입력층을 객체로 지정하여 요소를 추가하는 경우 그에 따른 입력층의 크기를 설정. 현 상황에선 input\_shape=(784, ) = 784개의 원소로 이루어진 1차원 배열 \*객체 지정 안해도 관례로 행함

* + - * + 모델 변수를 객체로 지정 🡪 훈련모델 생성하기

Model = keras.sequential(dense) : 인공신경망 모델에 객체를 지정하는 것으로 매개변수로 dense라는 출력층 입력

* + - * + 모델 설정

Model.compile() : 매개변수를 통해 모델에 설정을 추가한다.

매개변수:

Loss = : 손실값을 기록 및 출력하라는 매개변수

Loss = ‘categorical\_crossentropy’ : 크로스엔트로피손실함수(다중분류일 경우)

‘binary crossentropy’ : 로지스틱손실함수(이중분류일 경우)

‘sparse\_categorical\_crossentropy’ : 타깃값들이 원핫인코딩에 용이하게 0과 1의 형태가 아닌 그 외의 정수값들을 포함할 때 사용

Metrics=’accuracy’ : 추가적으로 정확도를 측정

* + - * + 모델 훈련

Model.fit(train\_scaled, train\_target, epochs=5)

* + - * + 모델을 테스트세트에 적용한 뒤 결과 출력

Model.evaluate(val\_scaled, val\_target)

* + - * 사이킷런과 케라스 API비교
        + 사이킷런은 클래스를 만들 때 가능한 많은 매개변수 사용
        + 케라스는 세가지 부분으로 분화

층을 만드는 부분(.layers)

모델객체를 지정하고 출력지표(compile())를 설정

훈련 및 평가를 하는 부분

* + **심층신경망:** 입력층을 제외하고 2개 이상의 층을 지니는 인공신경망 모델
    - 필수 개념:
      * 일반적으로 층을 셀 때 입력층은 제외하고 셈
      * **은닉층**: 입력층과 출력층에서 접근이 안되고 사이에 숨어 있다고 하여 은닉층이라고 불리며 출력층에 비선형식을 포함한 활성화함수를 만들어 사용
        + 은닉층이 출력하는 뉴런 개수는 정해져 있지 않지만 출력층보다 큰 은닉층을 사용해야 데이터 손실이 일어나지 않음
        + 자주 사용되는 함수:

시그모이드 함수

Relu 함수

Tanh 함수

* + - 생성 및 활용방법:
      * 은닉층 생성
        + Keras.layers.dense() : 출력층을 만들 때 와 동일한 방식으로 출력층 이전에 dense를 사용해 생성한 후 model = keras.sequential()에 리스트 형식으로 [은닉층, 출력층] 기입

층을 추가하는 다른 방법: 객체화 이후 add매써드를 사용해 층 추가

Model = keras.Sequential()

Model.add(keras.layers.Dense() …..

가장 많이 사용되는 방식으로 if구문을 활용하는 등의 동적인 모델 구성 가능

* + - * Summary() 메써드로 모델 확인
        + 은닉층과 출력층에 관한 정리가 출력됨 – 결과값 이해:

은닉층과 출력층이 자동으로 dense와 dense\_1과 같이 이름이 부여됨

두 층의 출력값들의 shape에서 앞의 차원이 none이라고 되어 있는 이유는 모델을 fit할 때 임의하강법으로 훈련을 하게 되는데 이때 batchsize가 shape 앞의 차원으로 출력됨

디폴트인 32의 사이즈를 지닌 batch를 활요할 때는 none이 출력됨

Fit의 매개변수인 batchsize= 를 활용하여 임의로 지정가능

Param #는 가중치, 절편 등과 같은 모델 파라미터의 개수를 의미함

직접 모델 파라미터의 개수를 계산하여 summary()의 출력값과 비교하여 모델이 잘 생성되었는 지 확인 가능

파라미터의 개수 = (입력되는 데이터 개수)\*(가중치의 개수) + (절편의 개수)

모든 입력과 모든 뉴런이 연결되어 있는 완전연결로 층이 이루으므로 가중치로 인해 생겨나는 파라미터가 위와 같다

절편은 각 층의 뉴런(출력)을 이루는 식마다 하나 씩 존재하므로 (절편의 개수) = (뉴런의 개수)

* + - * + 인공신경망과 동일하게 compile하고 훈련
    - 은닉층에 대표적으로 사용되는 활성화 함수들의 사용방법:
      * **렐루 함수**: z값이 0보다 크면 z를 그대로 출력하고 z값이 0보다 작으면 0으로 출력한다.
        + 용도**:** 시그모이드 함수의 단점인 출력값이 너무 크거나 작아서 완만하게 이어지는 구간에 포함될 때 함수의 변화정도가 작아 비교하기 어려워지는 문제를 해결
        + 사용방법:

Model.add(keras.layers.Dense(100, activation=’relu’)) : 새로운 층에 활성화 함수로 렐루함수를 입력

* + - **Flatten 층:** 입력층 바로 이후에 코드를 작성하며 이미지 배열을 일차원 배열로 펼쳐주는 전처리 작업을 자동으로 시행해주는 유틸리티층
      * 사용방법:
        + Model.add(keras.layers.Flatten(input\_shape = (28, 28)))
    - **옵티마이저:** 하이퍼파라미터 중에서 하나로 경사하강법과 같은 효과적인 학습방식을 지정해줌
      * 사용방법:
        + Model.compile()이 optimizer= 매개변수에 문자열로 옵티마이저 옵션을 입력한다
        + 객체로 옵티마이저를 따로 지정한 후 compile 매개변수로 입력

*옵티마이저이름 = Keras.optimizers.옵티마이저이름()*

*Model.complie(optimizer = 옵티마이저이름)*

* + - * 대표적인 옵티마이저 \*SGD 이외의 옵티마이저는 강의에서 깊이있게 다루지 X
        + **기본 경사 하강법 옵티마이저**

Sgd

축약어를 풀어썻을 때 확률적 경사하강법을 의미하지만 실제 디폴트로 32개의 샘플씩 배치로 구성하여 측정하는 임의경사하강법을 사용함

모멘텀

네스테로프 모멘

* + - * + **적응적 학습율 옵티마이저:** 최적값(수렴점)에 가까워질수록 학습율을 줄이는 모델 🡺 최적값에 다가갈수록 신중한 학습을 진행

RMSprop

Adam

Adagrad