지능형컴퓨팅과정 포트폴리오 경진대회

텐서플로우를 이용한
코로나 확진자수 예측 인공지능

딥러닝에 대해 학습하는 과정을 담았습니다

컴퓨터정보공학과 PE반
20192635 김한수

목차

📖시작하며,,,

동양미래대학교 인공지능응용프로그래밍 강의를 들으면서 공부한 내용을 토대로 인공지능응용프로그램이 경진대회 팀플에 활용할 코로나 확진자수 예측 인공지능을 만들어가는 과정을 담았습니다.

또한, 여기서 사용할 방식은?

코로나 확진자수는 **시간에 따라 변화**하므로 "**회귀분석**"을 이용하려 합니다. 시작하기 전에 사전지식을 학습하고 정리해보겠습니다.

🤔회귀분석이 뭐길래? 왜?

강환수 교수님의 인공지능응용프로그래밍 강의에서 회귀분석을 학습할 수 있었고, 다음과 같이 정리할 수 있습니다.

> ***회귀분석(Regression Analysis)***
>
> - 관찰된 **연속형 변수**들에 대해 두 변수 사이의 모형을 구한 뒤 적합도를 측정해내는 분석 방법
> - 주로 **시간에 따라 변화하는 데이터**나 어떤 영향, 가설적 실험, 인과 관계의 모델링 등의 통계적 예측에 이용

위의 내용을 살펴보면, "**시간에 따라 변화하는 연속형 변수 예측**"에 회귀분석을 활용하면 적절하다고 볼 수 있을 것입니다.

🤔개인적인 궁금증 : 왜 이름이 회귀분석일까?

> **여담(회귀의 어원)**
>
> **회귀, 回歸 : "한 바퀴 돌아서 본디의 자리나 상태로 돌아오는 것"**
>
> **회귀분석은** 통계학에서 **어떤 값이 극단치로 갈수록 평균적으로 되돌아가려는 경향이 있다는 "회귀분석"** 가설로부터 시작했다고 합니다.

💡조금 더 자세하게

회귀분석에도 여러가지 종류가 있습니다. 크게는 다음과 같이 나뉩니다.

하나 또는 여러개의 특징을 입력하는 것은 같으나 출력에서 차이점을 보입니다.

로지스틱 회귀는 "이진 분류"를 활용한다는 것을 제외하면 비슷한 방식입니다.

아래 강환수 교수님의 인공지능응용프로그래밍 강의자료에서 확인할 수 있습니다.

선형 회귀와 로지스틱 회귀

- **단순 선형 회귀 분석(Simple Linear Regression Analysis)**
  - 입력: 특징이 하나
  - 출력: 하나의 값 $H(x) = Wx + b$
    - **키로 몸무게 추정**
- **다중 선형 회귀 분석(Multiple Linear Regression Analysis)**
  - 입력: 특징이 여러 개, 출력: 하나의 값
    - **역세권, 아파트 평수, 주소로 아파트값을 추정**

$$y = W_1x_1 + W_2x_2 + \ldots W_nx_n + b$$

- **로지스틱 회귀(Logistic Regression)**
  - 이진 분류(Binary Classification)
  - 입력: 하나 또는 여러 개, 출력: 0 아니면 1
    - **타이타닉의 승객 정보로 죽음을 추정**

score(x)	result(y)
45	불합격
50	불합격
55	불합격
60	합격
65	합격
70	합격

강환수 교수님의 인공지능응용프로그래밍 강의자료

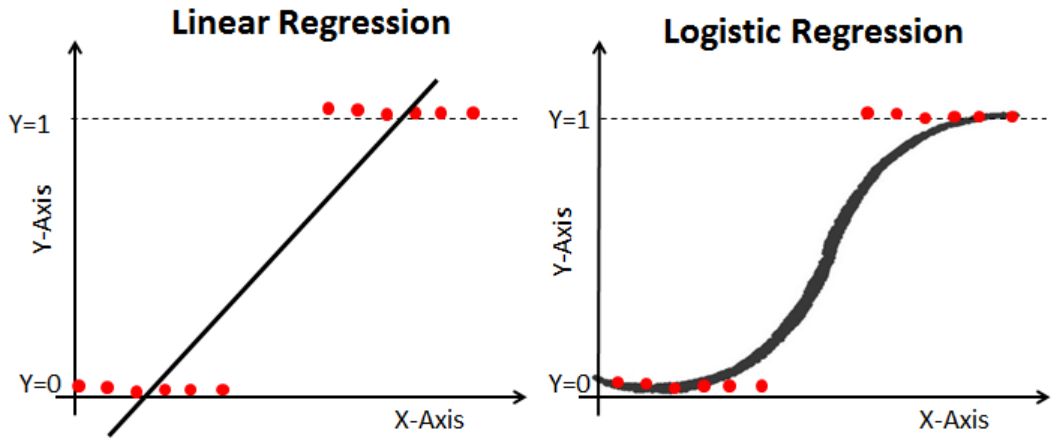
📚요약

이 내용을 다음과 같이 요약할 수 있습니다.

선형 회귀 분석, 로지스틱 회귀 분석 모두 **입력되는 특징들로 출력을 예측**하는 적합한 모델을 만들어가는 과정이며, 출력값에서 차이점을 보입니다.

- **선형 회귀 :** 이분법이 아닌 연속적인 값을 예측할 때 사용
- **로지스틱 회귀** : 이분법(Yes or No)로 분류할 때 사용

그리고 그래프를 그려보면 아래와 같은 모습을 볼 수 있습니다.



https://ebbnflow.tistory.com/129

더 자세한 내용은 프로그래밍을 진행하면서 담아내려고 합니다.

💨준비를 마쳤으니 출발.

따라서 위와 같은 근거로 회귀분석을 이용하여, 시간에 따라 변화하는 코로나 확진자수를 예측하러 출발해보겠습니다.

1. 공공데이터포털에서 데이터 수집

먼저 예측 모델을 만들기에 앞서 데이터를 구해야 했습니다.

그래서 저희는 공공데이터포털에 접속하여 적합한 데이터를 조사하였고,

아래와 같이 보건복지부에서 제공하는 코로나19 감영 현황 데이터 REST API를 사용하기로 했습니다.

기본정보

데이터명	보건복지부_코로나19 감염_현황 상세설명		
서비스유형	REST	심의여부	자동승인
신청유형	개발계정 | 활용신청	처리상태	승인
활용기간	2020-11-17 ~ 2022-11-17		

위 API는 아래와 같은 정보를 제공합니다.

출력결과(Response Element)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터	항목설명
결과코드	resultCode	2	필수	00	결과코드
결과메시지	resultMsg	50	필수	OK	결과메시지
한 페이지 결과 수	numOfRows	4	필수	10	한 페이지 결과 수
페이지 번호	pageNo	4	필수	1	페이지번호
전체 결과 수	totalCount	4	필수	3	전체 결과 수
게시글번호(감염현황 고유값)	SEQ	30	필수	74	게시글번호(감염현황 고유값)
기준일	STATE_DT	30	필수	20200315	기준일
기준시간	STATE_TIME	30	필수	00:00	기준시간
확진자 수	DECIDE_CNT	15	필수	8162	확진자 수
격리해제 수	CLEAR_CNT	15	필수	834	격리해제 수
검사진행 수	EXAM_CNT	15	필수	16272	검사진행 수
사망자 수	DEATH_CNT	15	필수	75	사망자 수
치료중 환자 수	CARE_CNT	15	필수	7253	치료중 환자 수
결과 음성 수	RESUTL_NEG_CNT	15	필수	243778	결과 음성 수
누적 검사 수	ACC_EXAM_CNT	15	필수	268212	누적 검사 수
누적 검사 완료 수	ACC_EXAM_COMP_CNT	15	필수	251940	누적 검사 완료 수
누적 환진률	ACC_DEF_RATE	30	필수	3.2396602365	누적 환진률
등록일시분초	CREATE_DT	30	필수	2020-03-15 10:01:22.000	등록일시분초
수정일시분초	UPDATE_DT	30	필수	null	수정일시분초

이 중에서 저희가 사용할 데이터는 확진자 수 기록을 한 기준일과 확진자 수 데이터 입니다.

출력결과(Response Element)

항목명(국문)	항목명(영문)	항목크기	항목구분	샘플데이터
기준일	STATE_DT	30	필수	20200315
확진자 수	DECIDE_CNT	15	필수	8162

자 그럼 이제 데이터를 한 번 확인해보러 가보겠습니다.

Google에서 제공하는 Colab 환경에서 진행했습니다.

먼저 데이터 수집을 진행했습니다.

공공데이터포털 링크 → 사용한 데이터

코로나 19 확진자 수 예측 프로그래밍

▾ 1. 데이터 수집

공공데이터포털에서 코로나 확진자 수 데이터 제공 API를 활용

사용한 데이터

```
[47]  1 # 공공데이터 사이트에서 data 가져오기
      2 # http 통신을 위한 requests 모듈 추가
      3 import requests
      4
      5 URL = "http://openapi.data.go.kr/openapi/service/rest/Covid19/getCovid19InfStateJson"
      6 serviceKey = "serviceKey=2LFIiMdTvuXZJ1HaadVE0OjcfxvNQ9S2dEckjNzBfIUeI1Az7bZjLpQUNKX68azfAVAOSTuIMA6t8RKPZauM%2Fg%3D%3D"
      7 pageNo = "&pageNo=1&numOfRows=10";
      8 dateRange = "&startCreateDt=20200110&endCreateDt=20201118";
      9 response1 = requests.get(URL + "?" + serviceKey +pageNo + dateRange)
```

2. 데이터 전처리

데이터를 수집한 후 모델링에 활용할 데이터를 확인하고 전처리하는 과정이 필요했습니다.

전처리의 이유는 다음과 같았습니다.

- 데이터를 확인해보니 적합하지 않은 값이 있었고, 이를 해결해야 했다.
- 많은 데이터 중 우리가 사용할 데이터는 "날짜", "누적 확진자 수"이며 불러온 데이터를 파싱할 필요가 있었다.

전처리를 진행한 과정과 코드는 다음과 같습니다.

▾ 2. 데이터 전처리

제공해주는 데이터 중에 날짜와 확진자 수만 가져오고 전처리

```
[12]  1 # 사용할 라이브러리 추가
      2 # 우리가 사용할 데이터는 xml형식의 데이터이므로 json으로 변경하여 사용.
      3 # json, xmltodict : xml형식을 dict형으로 변경하고 json으로 변경
      4 # pandas : 데이터를 좀 더 쉽게 다루기 위해 pandas 모듈 추가.
      5 # matplotlib : 시각화를 위해서 matplotlib 모듈 추가.
      6 # colab 외부 모듈 설치 : !pip install xmltodict
      7
      8 import json
      9 import xmltodict
     10 import pandas as pd
     11 import matplotlib.pyplot as plt
     12
```

```
[42]  1 # dict 형식으로 데이터를 저장.
      2 jsonString = json.loads(json.dumps(xmltodict.parse(response1.text)))
      3 print(jsonString)
      4
```

```
{'response': {'header': {'resultCode': '00', 'resultMsg': 'NORMAL SERVICE.'}, 'body': {'items': {'item': [{'accDefRate': '1.0499435105', 'accExamCnt': '2834362
```

위 과정을 통해 jsonString 중에 "item" 이라는 배열을 가져와야함을 확인했습니다.

그리고 그 데이터들을 아래와 같이 확인할 수 있었습니다.

```
1 datas = jsonString['response']['body']['items']['item']
2 datas
```

```
[{'accDefRate': '1.0499435105',
  'accExamCnt': '2834362',
  'accExamCompCnt': '2791674',
  'careCnt': '2842',
  'clearCnt': '25973',
  'createDt': '2020-11-18 09:34:00.934',
  'deathCnt': '496',
  'decideCnt': '29311',
  'examCnt': '42688',
  'resutlNegCnt': '2762363',
  'seq': '326',
  'stateDt': '20201118',
  'stateTime': '00:00',
  'updateDt': 'null'},
 {'accDefRate': '1.0451413254',
  'accExamCnt': '2815755',
  'accExamCompCnt': '2774553',
  'careCnt': '2644',
  'clearCnt': '25860',
  'createDt': '2020-11-17 09:38:28.585',
  'deathCnt': '494',
  'decideCnt': '28998',
  'examCnt': '41202',
  'resutlNegCnt': '2745555',
  'seq': '325',
  'stateDt': '20201117',
  'stateTime': '00:00',
  'updateDt': 'null'},
```

이후 pandas를 이용해 데이터를 보기 쉽게 정리했습니다.

그렇게 총 328개의 데이터셋을 확보할 수 있었습니다.

```
1 # pandas를 이용해 사용할 데이터(stateDt(기준일), decideCnt(확진자 수)) 정리
2 df = pd.DataFrame(datas)
3 df[["stateDt","decideCnt"]]
```

	stateDt	decideCnt
0	20201118	29311
1	20201117	28998
2	20201116	28769
3	20201115	28546
4	20201114	28338
...	...	...
323	20200205	18
324	20200204	0
325	20200203	15
326	20200202	2
327	20200101	0

328 rows × 2 columns

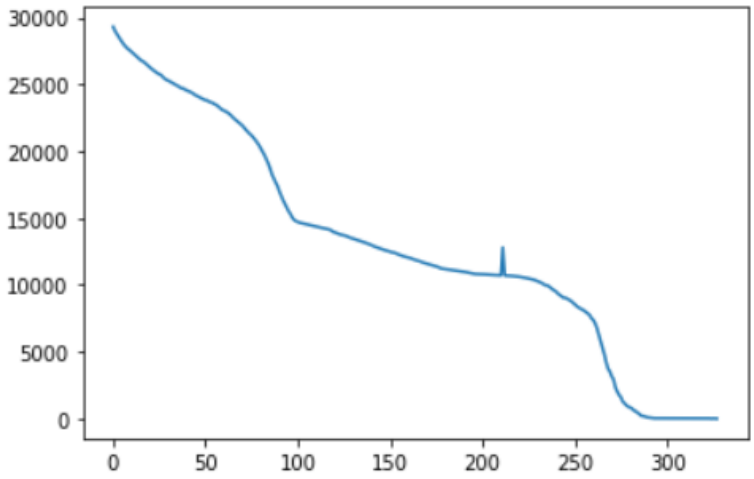
그 후 데이터를 확인해볼 필요가 있었습니다.

그 결과 데이터가 이상함을 감지할 수있었습니다.

▾ 사용할 데이터를 matplotlib를 사용해 확인해봤습니다.

```
[114]  1 df["decideCnt"].astype(int).plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1df9004e80>
```

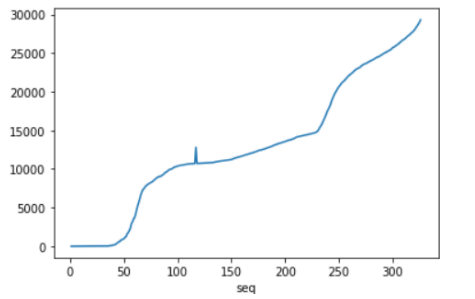


이상함을 감지하고 데이터를 우선 데이터를 정렬한 후에 데이터를 수정하기 시작했습니다.

▾ 😢벌써부터 데이터가 이상함을 감지할 수 있었고 우선 데이터가 정렬되어 있지 않음을 확인하고 먼저 데이터를 정렬했습니다.

```
[117]  1 # 기존 DataFrame을 cdf에 복사하고 진행했습니다.
       2 cdf = df
       3 # 먼저 object 형식으로 되어있는 "seq"와 "decideCnt"를 int형으로 변경해줍니다.
       4 cdf[["seq","decideCnt"]] = cdf[["seq","decideCnt"]].apply(pd.to_numeric)
       5
       6 # 그리고 "seq" 기준으로 정렬해준 뒤 "seq"를 인덱스로 사용하겠습니다.
       7 # pandas에 대한 사용법은 https://dandyrilla.github.io/2017-08-12/pandas-10min/ , https://c10106.tistory.com/3977 를 참고했습니다.
       8 cdf = cdf.sort_values(by="seq")
       9 cdf = cdf.set_index("seq")
      10
      11 # 그 결과
      12 cdf["decideCnt"].astype(int).plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1df91a1cc0>
```



▾ 🤔 결국 뭔가 데이터가 이상한 것을 확인할 수 있었습니다..

기존 확진자 수는 지속적으로 증가하는 값을 띄어야하는데 중간에 뾰족한 모습을 확인할 수 있습니다. 그래서 데이터를 조금 살펴보다가 또 이상한 점을 발견했습니다. 중간에 누적 확진자 수가 0으로 기록되어 있었습니다. 그래서 데이터를 전처리할 필요가 있었습니다,

```
[118] 1 zeroDatas = cdf["decideCnt"] == 0
      2 cdf[zeroDatas]
```

seq	accDefRate	accExamCnt	accExamCompCnt	careCnt	clearCnt	createDt	deathCnt	decideCnt	examCnt	resutlNegCnt	stateDt	stateTime	updateDt
1	None	NaN	NaN	NaN	1	2020-01-31 17:47:33.33	0	0	1	NaN	20200101	18:00	2020-02-03 12:21:56.56
4	None	NaN	NaN	NaN	0	2020-02-03 21:26:59.59	0	0	0	NaN	20200204	00:00	None

데이터를 수정하는 방식을 고민하다 결국 삭제하는 방법을 선택했습니다.

그리고 그 과정은 다음과 같습니다.

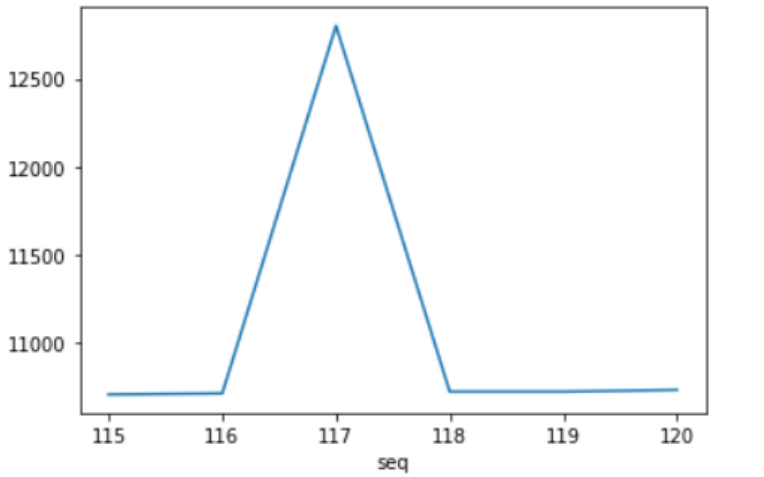
▾ 본격적인 데이터 전처리

전처리 방법을 고민해본 결과 다음과 같은 방식을 사용하기로 했습니다.

- 적합하지 않은 데이터는 삭제한다.(중간에 확진자 수 0, 그래프의 뾰족한 값)

```
[124] 1 # 뾰족한 부분은 쉽게 찾을 수 있었습니다. -> seq : 117
      2 cdf.loc[115:120]["decideCnt"].plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1df928b978>
```



```
[129] 1 # 이제 값이 00이었던 seq : 4, 값에 오류가 있는 seq : 117를 삭제해줍니다.
      2 cdf.drop([4,117])
      3 # seq를 깔끔하게 바꾸고 싶지만, seq는 단지 정렬용이었으니 그냥 진행하겠습니다.
```

seq	accDefRate	accExamCnt	accExamCompCnt	careCnt	clearCnt	createDt	deathCnt	decideCnt	examCnt	resutlNegCnt	stateDt	stateTime	updateDt
1	None	NaN	NaN	NaN	1	2020-01-31 17:47:33.33	0	0	1	NaN	20200101	18:00	2020-02-03 12:21:56.56
2	None	NaN	NaN	NaN	2	2020-02-03 12:22:49.49	2	2	2	NaN	20200202	09:00	None
3	None	NaN	NaN	NaN	0	2020-02-03 14:41:17.17	0	15	0	NaN	20200203	09:00	2020-02-04 14:19:46.46
5	None	NaN	NaN	NaN	0	2020-02-04 23:56:31.31	0	18	129	NaN	20200205	09:00	2020-02-05 9:43:16.16
6	None	NaN	NaN	NaN	1	2020-02-05 20:05:40.40	0	19	40	NaN	20200205	19:00	2020-02-06 10:05:36.36
...	...	...	...	...	...	...	...	...	...	...	...	...	...
322	1.0332542477	2777289	2742597	2210	25636	2020-11-14 09:31:38.207	492	28338	34692	2714259	20201114	00:00	null
323	1.0378476640	2786878	2750500	2362	25691	2020-11-15 09:40:16.172	493	28546	36378	2721954	20201115	00:00	null
324	1.0425904108	2797691	2759377	2516	25759	2020-11-16 09:34:22.212	494	28769	38314	2730608	20201116	00:00	null
325	1.0451413254	2815755	2774553	2644	25860	2020-11-17 09:38:28.585	494	28998	41202	2745555	20201117	00:00	null
326	1.0499435105	2834362	2791674	2842	25973	2020-11-18 09:34:00.934	496	29311	42688	2762363	20201118	00:00	null

326 rows × 13 columns

추가로 pandas에서 제공하는 함수들도 시도해봤습니다.

다행히 더이상 문제는 없어보였습니다.

```
[132] 1 # 추가로 공부하던 중 pandas에서 데이터 전처리를 도와주는 함수들이 있어서 시도해봤습니다.
      2 # null이거나 중복되는 값들을 찾아주는 함수를 사용했습니다.
      3 # 그 결과 중복 값은 존재하지 않았고 모델에 사용할 "decideCnt" 값도 null 값이 없음을 확인할 수 있었습니다.
      4 print(cdf.isnull().sum(), " 중복 값 : " , cdf.duplicated().sum())
```

```
accDefRate         60
accExamCnt         60
accExamCompCnt     60
careCnt            60
clearCnt            0
createDt            0
deathCnt            0
decideCnt           0
examCnt             0
resutlNegCnt       60
stateDt             0
stateTime           0
updateDt          100
dtype: int64  중복 값 :  0
```

3. 인공지능 모델링 문제 발생

🔥 드디어 전처리 끝, 모델링 시작을 하려고 했으나..

[15]
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(units=1, input_shape=(1,), activation='linear'),
])
```

[16]
```
model.compile(
    optimizer='SGD',
    loss ='mse'
)

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(x_train, y_train,epochs=500,
                    batch_size=32
                    )

```

```
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 124/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 125/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 126/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 127/500
8/8 [==============================] - 0s 2ms/step - loss: nan
Epoch 128/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 129/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 130/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 131/500
8/8 [==============================] - 0s 2ms/step - loss: nan
Epoch 132/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 133/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 134/500
8/8 [==============================] - 0s 1ms/step - loss: nan
Epoch 135/500
```

> 단순하게 시간에 따라 증가하는 값을 x는 경과한 날짜로 y는 누적된 확진자 수로 하여 선형회귀를 이용하면 가능할 것이라 생각하고 진행해본 결과..
> 위와 같이 결과값이 **nan**으로 나오는 문제가 발생했습니다.
> **결국, 손실함수와 최적화 과정을 좀 더 학습하기로 했습니다.**

4. 손실함수에 대한 학습

먼저 딥러닝 혹은 머신러닝은 컴퓨터가 어떠한 값들을 예측하는데 사용할 함수를 알아내기 위해서 필요한 가중치와 편향을 찾아가는 과정이라는 것을 인지하고 이 과정에 유용한 손실함수에 대해서 학습해보았습니다.

손실함수란(Loss function)

> 목적함수(Objective function), 비용 함수(Cost function) 라고도 불리며,
> 딥러닝 과정에서 실제 값과 예측 값에 대한 **오차를 줄이기 위하여 최적화된 식**을 의미합니다.
>
> 즉. 딥러닝 모델의 최적화를 위해 손실함수에 대한 학습이 필요합니다.

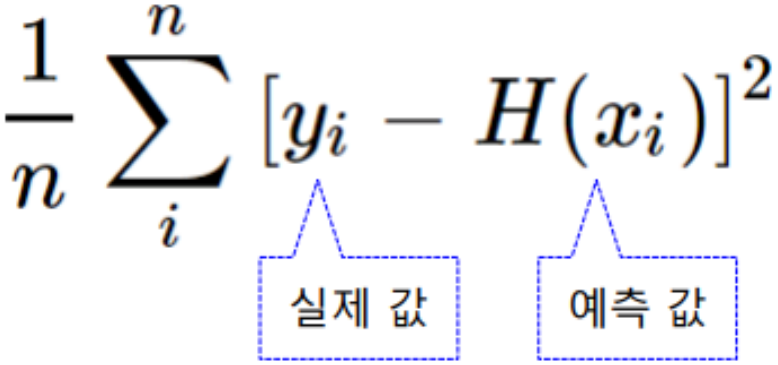
손실함수의 종류

- MSE(Mean Square Error) : 평균제곱오차
- Categorical crossentropy
- Sparse Categorical crossentropy
- . . .

MSE(Mean Square Error) : 평균제곱오차

> **예측한 값과 실제 값 사이의 평균 제곱 오차를 정의.**
> 차가 커질수록 제곱 연산으로 인해서 값이 더욱 뚜렷해지는 특성.
> 제곱으로 인하여 오차가 양수이든 음수이든 누적 값을 증가시킴.
>
> ~~페트체크가 필요한 부분이지만 개인적으로 구글링을 해보니 일반적으로 사용하는 손실함수로 보입니다.~~

MSE에 대한 식은 다음과 같습니다.

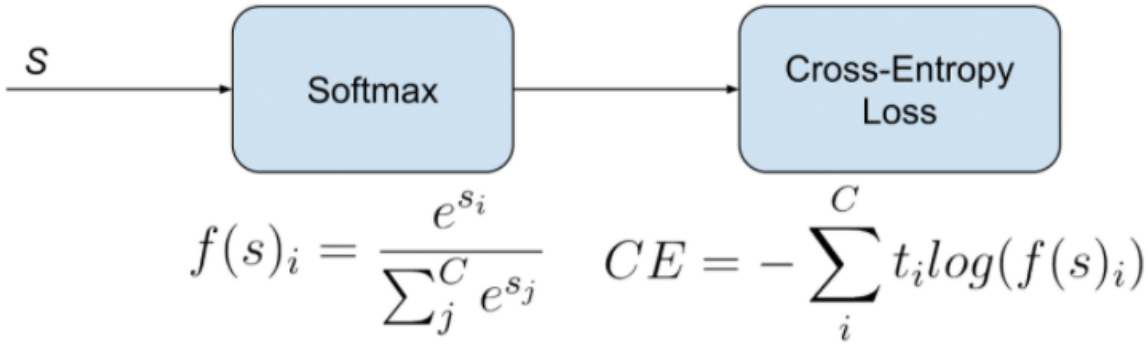


강환수 교수님의 인공지능응용프로그래밍 강의자료

Categorical Crossentropy

> **레이블 클래스가 2개 이상일 경우 사용**되고, 보통 softmax 다음에 연계되어 사용된다고 합니다. → softmax loss라고 부르기도 한다고..
>
> 뒤에 **Crossentropy**에 대해 조금 알아보니
> **엔트로피란 정보의 기대값**을 뜻하고, **정보량이란 어떤 사건에 가치를 매겨놓은 것**. 정보량을 측정하는데는 희소성 등의 기준이 있다는 등..
>
> 결국 **Crossentropy는 다른 사건의 확률을 곱해서 엔트로피를 계산한 것**이라고 합니다.
> 이런 정보들을 알아봤지만 개인적으로 **활성화 함수인 softmax와 결합해서 사용한다는 것**이 중요해보입니다.

softmax와 Crossentropy에 관한 흐름 및 식



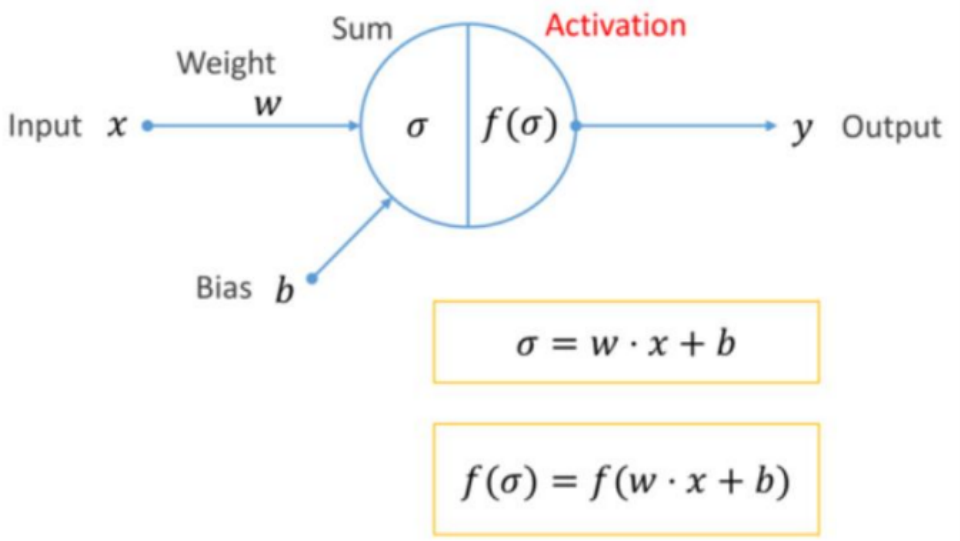
https://dsbook.tistory.com/64

5. 활성화 함수에 대한 학습

활성화 함수란?

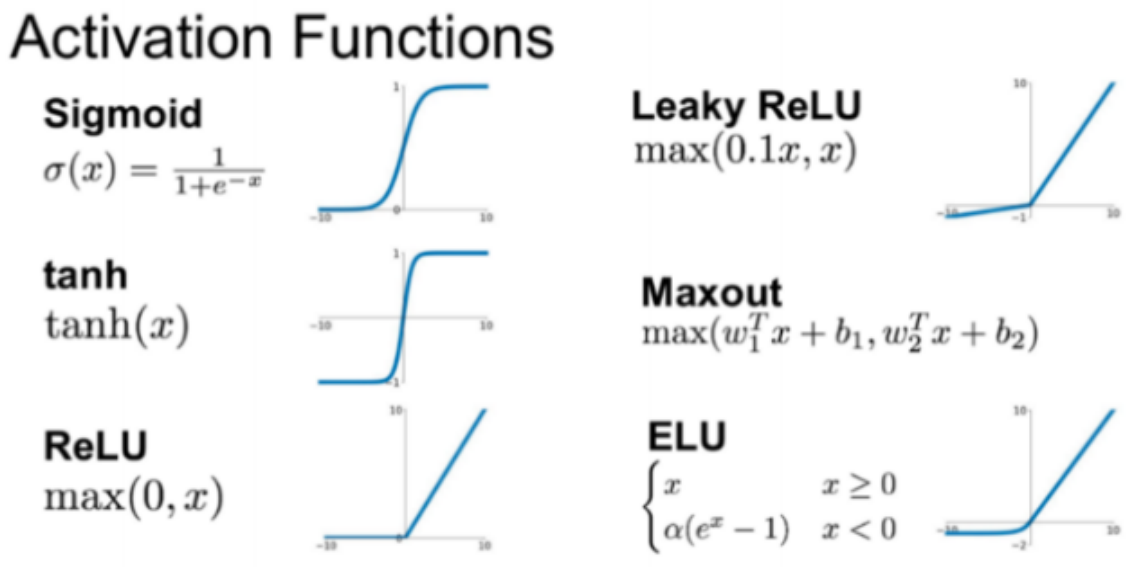
> **뉴런의 출력 값을 정하는 함수**
>
> 뉴런이 다음 뉴런으로 신호를 보낼 때 입력신호가 일정 기준 이상이면 보내고 기준에 도달하지 못하면 보내지 않을 수도 있다고 합니다.
>
> 이러한 **뉴런의 신호를 결정해주는 역할**을 활성화 함수로 만들었다고 이해하면 개인적으로 좋을 것 같습니다.

- **활성화 함수**
  - 뉴런의 출력 값을 정하는 함수



강환수 교수님의 인공지능응용프로그래밍 강의자료

활성화 함수의 종류



Different Activation Functions and their Graphs

강환수 교수님의 인공지능응용프로그래밍 강의자료

개인적으로 구글링을 해본 결과, 활성화 함수의 종류는 위 그림에 포함된 함수 외에도

- 선형함수 (Linear Function) : 말 그대로 직선적인 함수 (y=x)
- 계단함수 (Step Function) : 입력이 양수일때 1 음수일때 0 을 출력하는 이진적 함수

등등 추가적으로 존재한다고 합니다. 이 중에 몇 가지만 뽑아서 학습해보았습니다.

선형함수 (Linear Function)

~~제가 위에서 코로나 확진자 수 예측 모델링에 활용해봤던 함수입니다.~~
~~X는 코로나 시작일로 부터 지나온 날짜 수, Y는 코로나 누적 확진자 수를 사용하여 나름 괜찮은 결과가 나오지 않을까 기대했지만, 결과값으로 nan이 나오는 문제점이 발생했습니다. 인공지능 팀 프로젝트 발표 전에는 문제점을 해결해 완성했으면하는 바람입니다.~~

> **말 그대로 직선적인 함수(y=x)**
>
> 결론적으론 이 함수를 사용하면 뉴런 층을 쌓아가는 딥러닝 네트워크의 이점이 전혀 없다고 합니다.
>
> 2개의 뉴런 층을 쌓아도 X에 곱해지는 가중치는 또 다른 가중치로 치환이 가능하고 뒤에 더해지는 편향은 또 다른 편향으로 치환이 가능하기 때문에 하나의 선형함수를 사용한 것과 동일한 결과를 가지게 됩니다.

위 설명을 식으로 나타내면 다음과 같습니다.

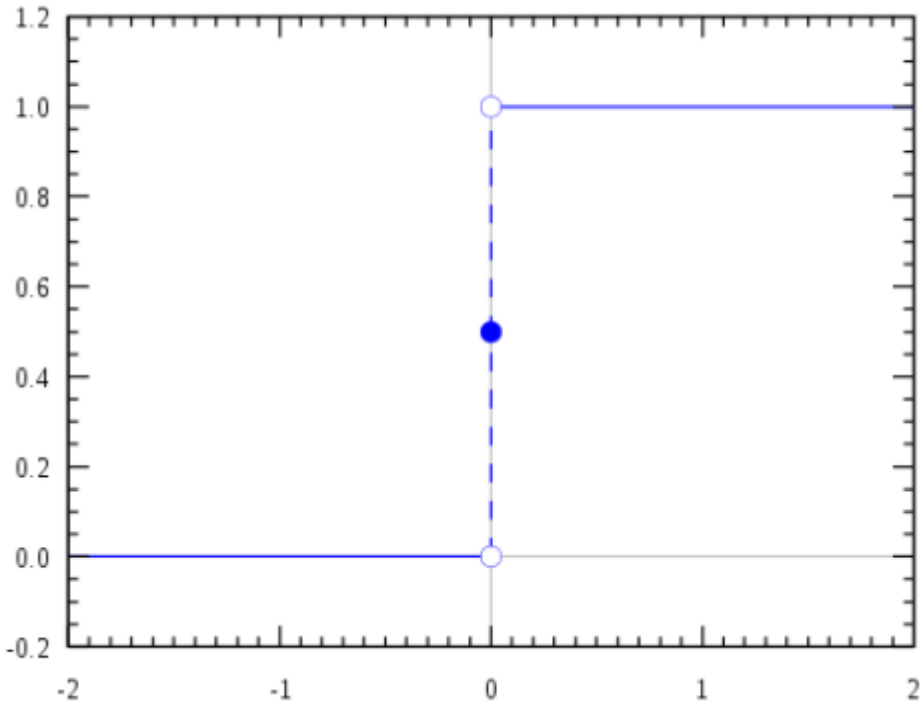
$$\sigma(Z) = W_\sigma Z + B_\sigma$$

$$\begin{aligned} A_2 &= W_\sigma\left(W_2{}^T\left(W_\sigma\left(W_1{}^T X + B_1\right) + B_\sigma\right) + B_2\right) + B_\sigma \\ &= \underbrace{W_\sigma W_2{}^T W_\sigma W_1{}^T}_{W} X + \underbrace{W_\sigma W_2{}^T W_\sigma B_1 + W_\sigma W_2{}^T B_\sigma + W_\sigma B_2 + B_\sigma}_{B} \\ &= WX + B \end{aligned}$$

https://bbangko.tistory.com/5

계단 함수 (Step Function)

> **입력이 양수일때 1 음수일때 0 을 출력하는 이진적 함수**
>
> **아래 그림은 계단함수의 그래프입니다.**
> 모델 Optimization 과정에서 미분을 해야하는데 그래프를 보면 미분 불가능한 것을 알 수 있죠.. 그래서 쓸 수가 없다고 합니다.



https://bbangko.tistory.com/5

Sigmoid Function

위에 활성함수 종류에 붙여둔 그래프 중에 첫번째로 우아한 곡선 모양을 띄는 함수입니다.

> **입력을 정규화(normalize) 하는 함수.**
>
> **즉 (0, 1) 사이의 값으로 변경시켜줍니다.**

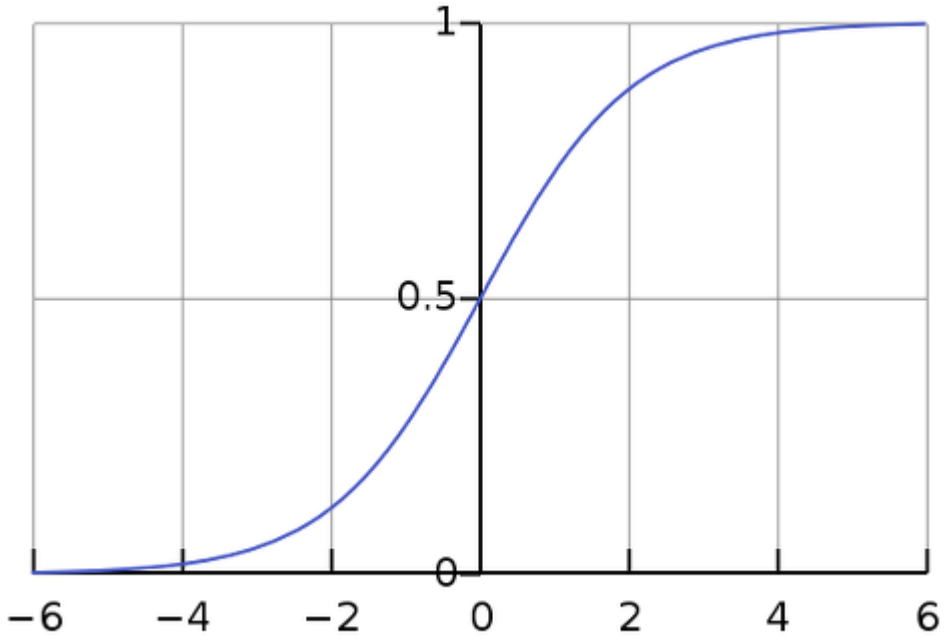
sigmoid 함수의 식

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

$$\frac{d}{\partial x} sigmoid(x) = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right)$$

https://bbangko.tistory.com/5

함수 그래프



Sigmoid Function

https://bbangko.tistory.com/5

Tanh Function

> **sigmoid function을 보완하고나 나온 함수라고 합니다.**
>
> **sigmoid와 달리 입력값을 (-1, 1) 사이의 값으로 정규화합니다.**

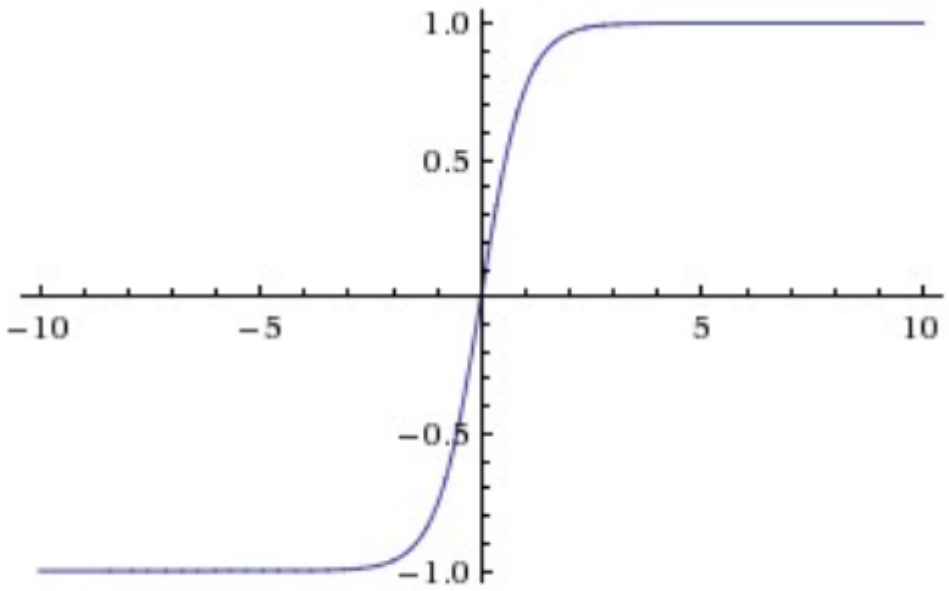
Tanh 함수 식

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\frac{d}{dx}tanh(x) = 1 - tanh(x)^2$$

https://bbangko.tistory.com/5

함수 그래프



Tanh Function

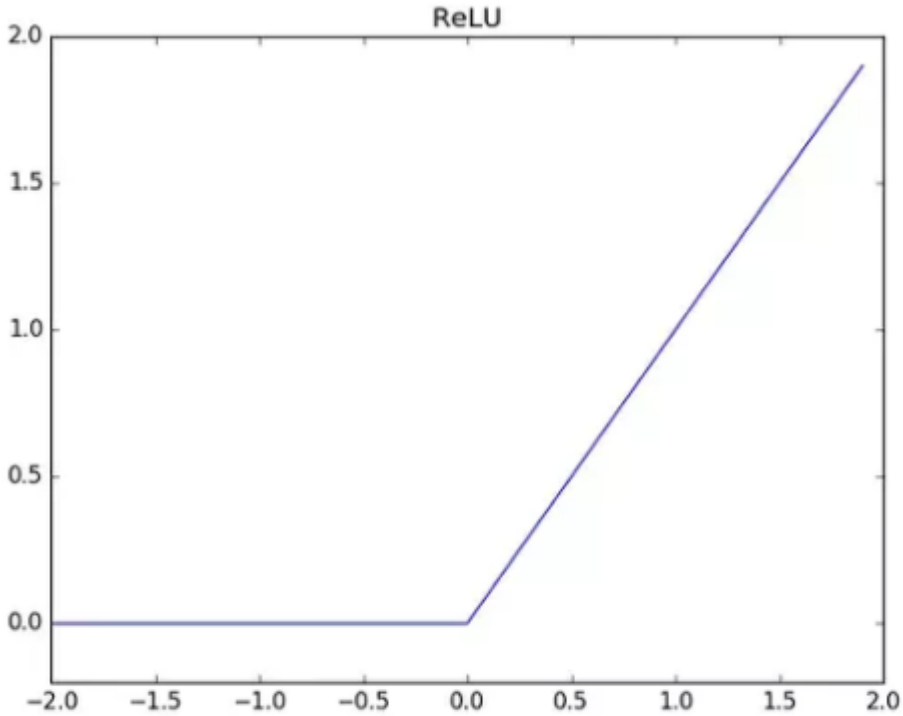
https://bbangko.tistory.com/5

ReLU Function (Rectified Linear Unit)

> **양수에서는 Linear Function과 같으며 음수는 0으로 버려지는 함수**

> Linear Function에서 뉴런 층을 많이 쌓을 수 없었던 점을 보완 하면서 exp()함수를 실행하지 않아 sigmoid나 tanh보다 빠르게 학습이 가능하다고 합니다.

ReLU 함수의 그래프



Relu Function

https://bbangko.tistory.com/5

여기까지 학습하고 남은 내용은 인공지능 팀 프로젝트 경진대회에서 풀어가려 합니다.

6. 후기

처음 시작할 때는
뭐 인공지능 조건들 여러개 입력값으로 넣고 결과값 넣고
입력과 결과에 대한 관계를 아주 잘 나타낸 함수를 찾는다.
간단하지 않나? 라고 생각했지만 케이스에 적합한 방식을 선택해야 했고 생각보다 긴 시간이 소모됐습니다.

긴 시간 자발적으로 공부하며 얻어가는 것들이 많았지만, 아쉬운 것들도 많이 남습니다. 학습 자료들을 보면 대부분 방법을 알려주고 이것들이 어떤 특성을 가지고 어떤 케이스에 사용하는 것이 적합하고 왜 생겨난 것인지 원초적인 것을 담아내지 않는 것이 정말 아쉬웠습니다.

그리고 공공데이터 포털에서 가져온 데이터를 보며 공공기관에서 제공하는 것이지만 오류가 있을 수 있고 데이터 전처리 과정이 정말 중요하다는 것을 알 수 있는 경험이었던 것 같습니다.

긴 글 읽어주신 것에 감사드리며, 이만 마치겠습니다.
감사합니다.

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Dense(units=1, input_shape=(1,), activation='relu')
])
```

```
model.compile(
    optimizer='rmsprop',
    loss ='mse'
)

model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_8 (Dense)              (None, 1)                 2
=================================================================
Total params: 2
Trainable params: 2
Non-trainable params: 0
_________________________________________________________________
```

```
history = model.fit(x_train, y_train,epochs=1000,
                    batch_size=2
                    )

```

```
line_x = np.arange(min(x_train), max(x_train), 0.01)
line_y = model.predict(line_x)

plt.plot(line_x,line_y,'r-')
plt.plot(x_train,y_train,'bo')
plt.show()
```