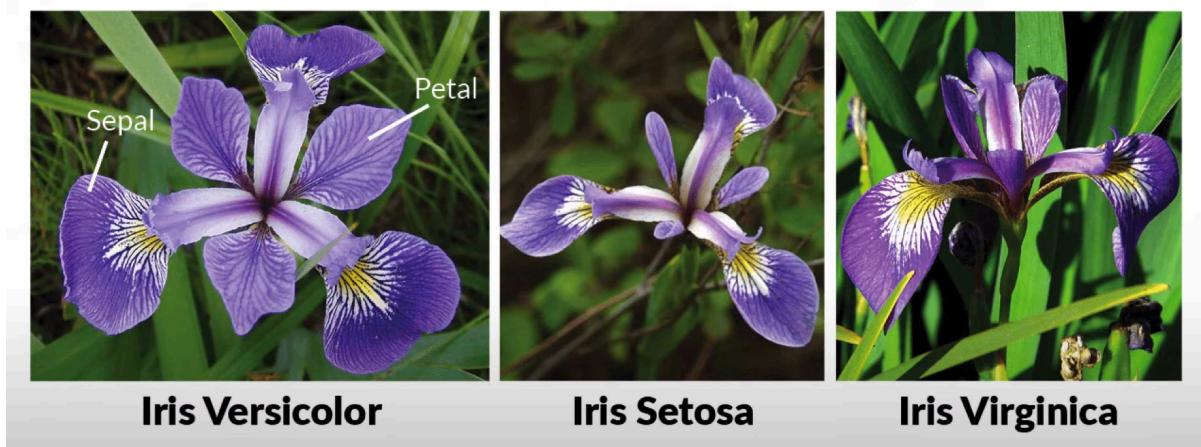


아이리스 품종 분류

지금부터 여러분은 식물학자입니다. 식물들의 생태계를 연구하고 있습니다. 특히 아이리스는 한국에서는 봇꽃이라고 불리는 꽃에 관심이 많은 데요. 꽃말은 좋은 소식, 사랑의 메시지라는 가지고 있습니다. 아이리스에 꽂혀 있는 당신은 그냥 아이리스가 좋습니다. 한가할 때면 봇꽃이 피는 아름다운 모습을 찾아 봅니다.



한번 구경해 볼까요. 봇꽃은 여러가지 종류가 있고, 종류에 따라서 꽃의 크기가 다릅니다. 영어로 꽂잎은 petal 꽂받침은 sepal이라고 하는데요 아이리스의 꽂잎과 꽂받침의 크기를 관찰했고, 관찰한 결과를 데이터로 정리했습니다.



그 데이터가 `iris.csv` 파일입니다. 데이터를 보면 꽂잎 폭과 길이 꽂받침의 폭과 길이, 이렇게 4개의 칼럼을 독립변수로 하고, 꽃의 품종을 종속변수로 하면 되겠구나 하는 느낌이 오시나요? 그럼 꽂잎과 꽃의 크기를 가지고, 아이리스의 품종을 구분하는 모델을 구현하면 되겠네요.

Executable File | 151 lines (151 loc) | 3.77 KB

Raw Blame ⌂ ⌐ ⌒

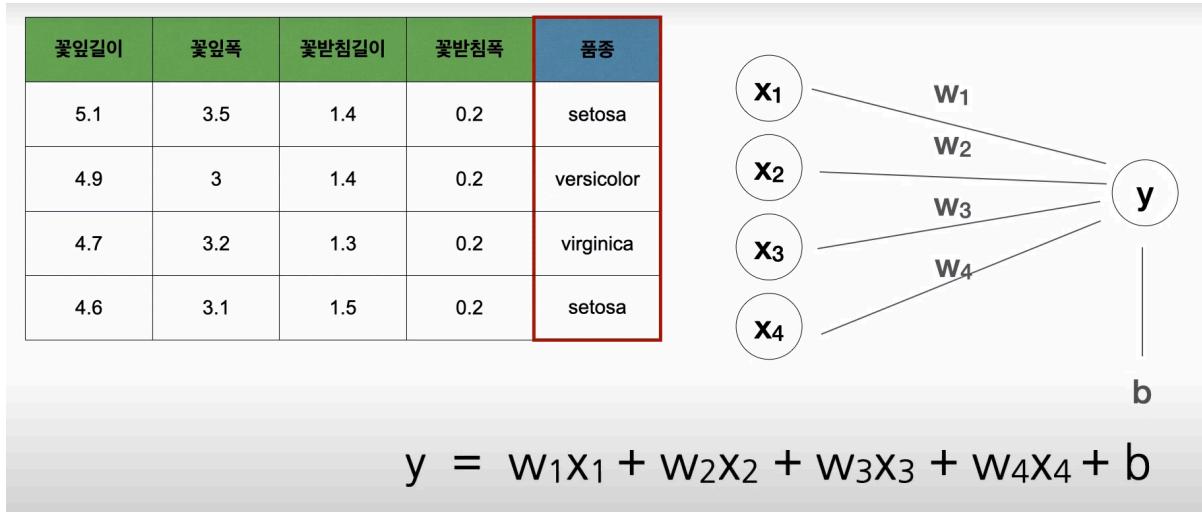
Search this file...

	꽃잎길이	꽃잎폭	꽃받침길이	꽃받침폭	품종
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

그런데 아이리스 데이터는 레모네이드나 보스턴 집값 데이터와는 차이가 있습니다. 우리가 이전에 사용한 데이터는 종속변수가 판매량, 집값 같은 수치형 변수였는데요. 아이리스 데이터의 종속변수는 품종으로 숫자가 아니네요. 범주형 데이터입니다. 종속변수가 범주형 데이터의 경우에는 분류 문제입니다.. 아이리스는 종속변수가 범주형이니 분류 문제입니다. 자 그럼 딥러닝을 분류 모델로 구성하는 코드를 만나봅시다.

원핫인코딩

먼저 아이리스 데이터 중 관측치 몇개를 가져와봤습니다. 모델의 모형인 퍼셉트론과식을 만들어 봅시다. 독립변수가 4개이므로 입력층에서는 4개의 입력을 받습니다. 종속변수는 1개이므로 예측값을 하나 만들 것이고 그림과 같은 모형은 아래 그림의 수식으로 표현할 수 있겠습니다.



그런데 원가 이상하게 느껴지시나요? 종속변수의 값이 범주형입니다. 숫자가 아닌 것이 수식의 결과로 나올 수는 없겠죠? 조금만 더 깊이 생각해보면, 수식에서는 입력이든 출력이든 모두 숫자만 들어갈 수 있습니다. 변수는 제외하고요. 이런 수식에 범주형 데이터를 사용하는 것은 가능해 보이지 않네요. 그럼 대체 어떻게 해야 할까요? 범주형의 종속변수만 가지고 먼저 생각해 봅시다.

원핫인코딩 onehot-encoding

품종	setosa	virginica	versicolor
setosa	1	0	0
virginica	0	1	0
versicolor	0	0	1
setosa	1	0	0
versicolor	0	0	1

이런 범주형 데이터는 수식에 사용할 수 있는 형태로 바꿔주는 과정을 거쳐야 합니다. 먼저 모든 범주들을 오른쪽 컬럼처럼 만들어 줍니다. 지금은 범주가 setosa, virginica, versicolor 이렇게 3가지가 있어서 3개의 컬럼이 생겼습니다. 먼저 모든 값을 0으로 한 다음 첫번째 데이터는 setosa이니, 컬럼을 1로 바꿔줍니다. 두번째 데이터는 virginica입니다. virginica컬럼을 1로 바꿔줍니다. 나머지 값에 대해서도 동일하게 해당 범주에 해당하는 컬럼의 값을 바꿔줍니다. 이렇게 바꾸고 나니 어떤가요? 데이터가 0 또는 1, 숫자 형태로 변화가 되었습니다. 이렇게 범주형 데이터를 1과 0의 데이터로 바꿔주는 과정을 원핫인코딩이라고 합니다.

꽃잎길이	꽃잎폭	꽃받침길이	꽃받침폭	품종	# 학습할 데이터를 준비합니다. 아이리스 = pd.read_csv('iris.csv')
5.1	3.5	1.4	0.2	setosa	
4.9	3	1.4	0.2	versicolor	
4.7	3.2	1.3	0.2	virginica	
4.6	3.1	1.5	0.2	setosa	

꽃잎길이	꽃잎폭	꽃받침길이	꽃받침폭	품종.setosa	품종.virginica	품종.versicolor
5.1	3.5	1.4	0.2	1	0	0
4.9	3	1.4	0.2	0	1	0
4.7	3.2	1.3	0.2	0	0	1
4.6	3.1	1.5	0.2	1	0	0

데이터를 로딩하는 코드입니다.

```
아이리스 = pd.read_csv('iris.csv')
```

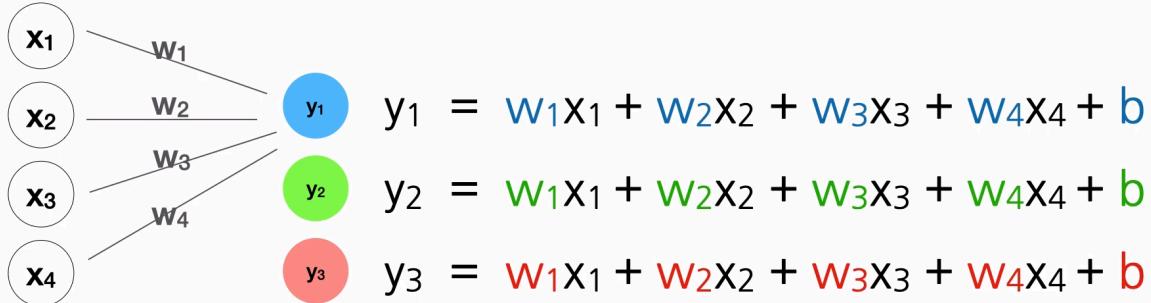
이렇게 데이터를 읽어 들이면, 모든 컬럼이 "아이리스"라는 변수에 들어 있겠죠. 범주형인 컬럼과 수치형인 컬럼이 섞여 있을 겁니다. 심지어 범주형 컬럼은 하나가 아니고, 여러개 될 수도 있어요. 원핫인코딩을 적용하려면 원가 막 복잡한 코드를 배워야 할 것만 같은데요. pandas에서 이 한줄의 코드를 사용하면,

```
인코딩 = pd.get_dummies(아이리스)
```

데이터 내의 범주형 변수들만 골라서 모조리 원핫인코딩 된 결과를 만들어 줍니다. 바로 그림의 라벨처럼요. 너무 감동적이지 않나요? 이제 독립변수와 종속변수만

분리해주면 데이터의 준비가 끝나는 것입니다. 바로 이변형된 데이터를 가지고 모델을 구성해 봅시다.

꽃잎길이	꽃잎폭	꽃받침길이	꽃받침폭	품종.setosa	품종.virginica	품종.versicolor
5.1	3.5	1.4	0.2	1	0	0
4.9	3	1.4	0.2	0	1	0
4.7	3.2	1.3	0.2	0	0	1
4.6	3.1	1.5	0.2	1	0	0



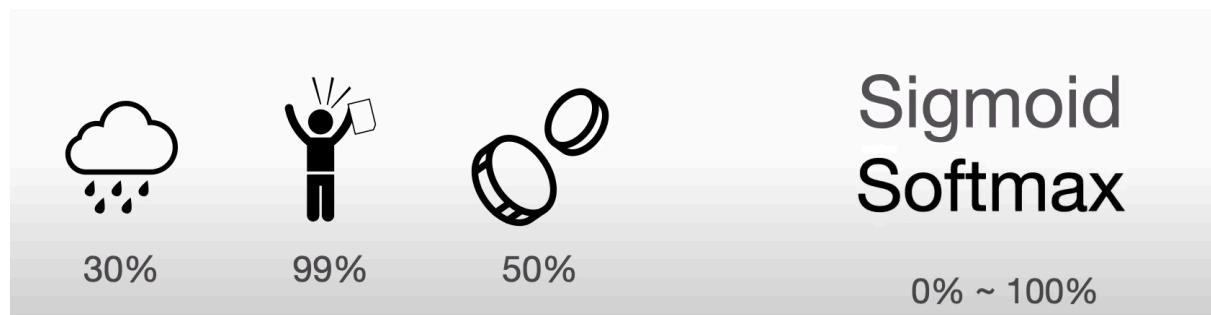
독립변수가 4개네요. 입력층은 4개의 입력을 받아야 합니다. 위 그림 x_1, x_2, x_3, x_4 처럼요. 원핫인코딩이 되면서 종속변수의 개수가 3개가 되었습니다. `setosa` 컬럼을 위해 출력을 하나 만들어 주어야 합니다. `setosa`인지 아닌지 판단하는 수식이 하나 필요하겠네요. 그게 첫번째 식입니다. 두번째 식은 `versicolor`인지 아닌지 판단하는 출력과 수식입니다. 세번째 식은 `virginica` 컬럼에 대한 출력과 수식입니다. 컴퓨터는 세가지 수식의 가중치를 모두 찾아야 하는 건데요. 그래서 모델을 만드는 코드에서

```
X = tf.keras.layers.Input(shape=[4]) # 여기가 독립변수 4개
Y = tf.keras.layers.Dense(3, activation='softmax')(X) # 여기가 종속변수 3개
model = tf.keras.models.Model(X, Y)
model.compile(loss='categorical_crossentropy',
metrics='accuracy')
```

입력 부분과 출력 부분의 숫자가 다음과 같이 4와 3이 되는 것입니다.

분류 예측

우리가 만드는 분류 모델이 하는 일은 분류를 추측하는 일입니다. 분류를 추측하는 건 어떻게 표현할까요. 내일 비가 오거나 오지 않거나 하는 분류가 있습니다. 우리는 내일 비가 올 확률은 30%야 라고 표현 합니다.



합격을 하거나 합격을 하지 않거나 하는 분류가 있습니다. 합격할 확률은 99%야 라고 표현을 하죠. 또 동전의 앞면이 나올 확률은 50%야 라고 표현을 합니다. 우리는 이렇게 0% ~100% 사이의 확률값으로 분류를 표현합니다. 분류 모델이 분류를 추측하는 것도 사람이 표현하듯 확률로 표현해 주면 좋겠죠. 그렇게 하도록 만들어 주는 도구가 바로 활성화 함수입니다.

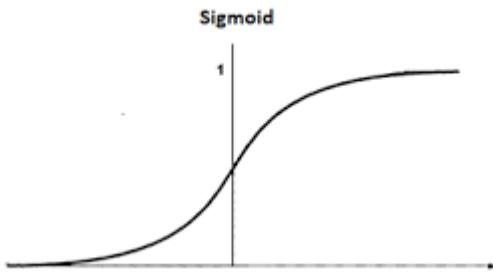
활성화 함수

딥러닝을 하면서 사용하는 활성화 함수는 크게 3가지 있습니다.
sigmoid,softmax,ReLU 가 그 3가지에요!

우선, 시그모이드 함수부터 알아봅시다!

시그모이드

시그모이드 함수의 생김새는 다음과 같습니다.



함수식은

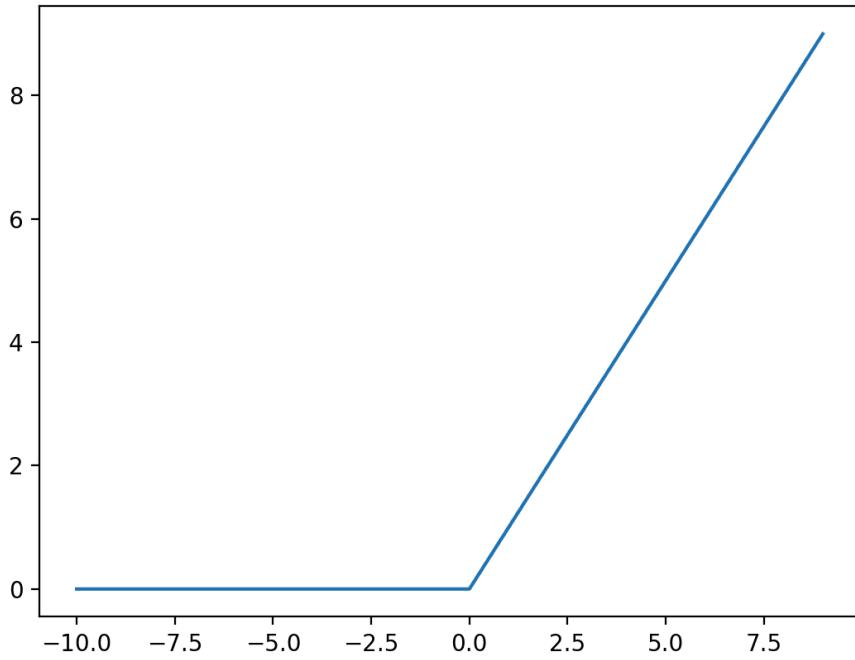
$$\frac{1}{1 + e^{-x}}$$

다음과 같습니다.

- 시그모이드 함수는 입력값이 커지면 커질수록 1에 수렴하고, 작아지면 작아질수록 0에 수렴합니다.
- 시그모이드 함수를 미분하면, 양 쪽으로 향할수록 변화값이 거의 없습니다.
- 따라서, 오류역전파를 할 때, Vanishing Gradient 현상이 발견될 수 있습니다.
- 0 또는 1을 반환하기 때문에, 이진 분류 모델의 마지막 활성화 함수로 사용됩니다!

ReLU

- ReLU 함수는 은닉층의 활성화 함수로 사용됩니다.
- 입력값이 0보다 작거나 같을 때는 항상 0을 출력하고, 0보다 크면 입력값과 동일한 출력값을 출력합니다.



소프트맥스

- softmax 함수는 시그모이드와 비슷하게, 0~1사이로 변환하여 출력하지만, 출력값들의 합이 1이 되도록 하는 함수입니다.
- 따라서 다중 분류의 최종 활성화 함수로 사용됩니다!!

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

- 소프트 맥스의 식은 위와 같다.

간단 정리 !

시그모이드 - 이진 분류 모델의 마지막 활성화 함수 !

소프트맥스 - 다중 분류 모델의 마지막 활성화 함수 !

ReLU - 기본적으로 은닉층에 사용하는 활성화 함수 !

분류 문제는 회귀 문제의 비해서 조금 어렵습니다. 확률의 개념이 등장하기 때문인데요. 이런 어려운 개념이 등장할 때, 빨리 넘어서야 하는 장애물로 여기지는 말아주세요. 우리도 사람들을 사귈 때에도 매우 좋은 사람이지만 조금 까다로워서 천천히 조심해서 사귀어야 하는 친구들이 있듯이 확률이라는 지식과도 천천히 사귀는 것을 권장드립니다. 범주형 데이터를 확률로 바라보는 경험과 연습을 자주하며 마주치다 보면 확률의 개념에 자연스럽게 다가설 수 있을 겁니다.

```
1 # -*- coding: utf-8 -*-
2 import tensorflow as tf
3 import pandas as pd
4
5 파일경로 = './data/iris.csv'
6 아이리스 = pd.read_csv(파일경로)
7 아이리스.head()
8
9 인코딩 = pd.get_dummies(아이리스)
10 인코딩.head()
11
12 print(인코딩.columns)
13
14 독립 = 인코딩[['꽃잎길이', '꽃잎폭', '꽃받침길이', '꽃받침폭']]
15 종속 = 인코딩[['품종_setosa', '품종_versicolor', '품종_virginica']]
16 print(독립.shape, 종속.shape)
17
18 X = tf.keras.layers.Input(shape=[4])
19 Y = tf.keras.layers.Dense(3, activation='softmax')(X)
20 model = tf.keras.models.Model(X, Y)
21 model.compile(loss='categorical_crossentropy',
22 | | | | metrics=['accuracy'])
23
24 model.fit(독립, 종속, epochs=100)
25
26 print(model.predict(독립[:5]))
27 print(종속[:5])
28
29 print(model.predict(독립[-5:]))
30 print(종속[-5:])
31
32 print(model.get_weights())
```