

이항 분류 및 다항 분류

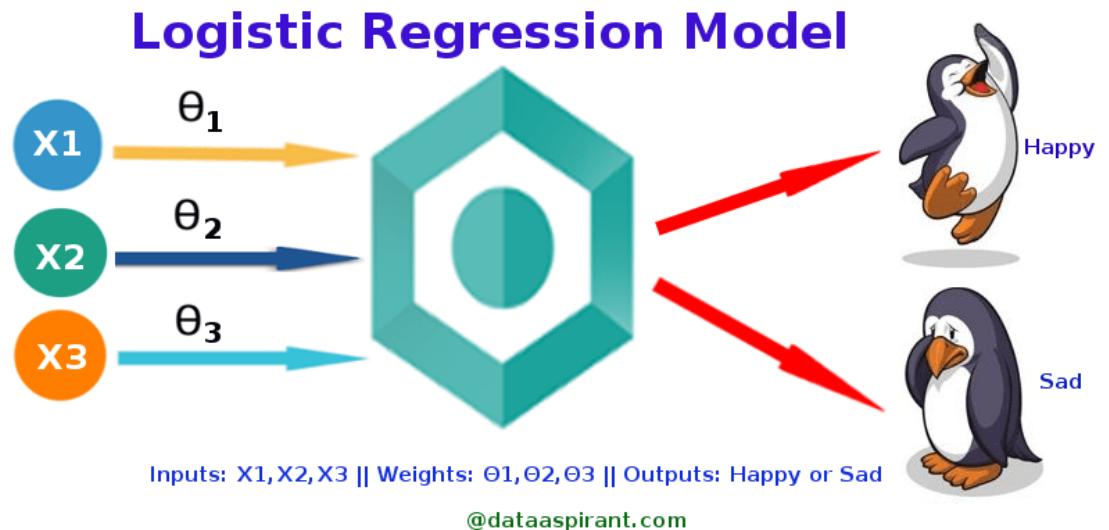
파일

- `classcification.ipynb`

이진(이항) 분류

• 두 가지로 분류하는 방법

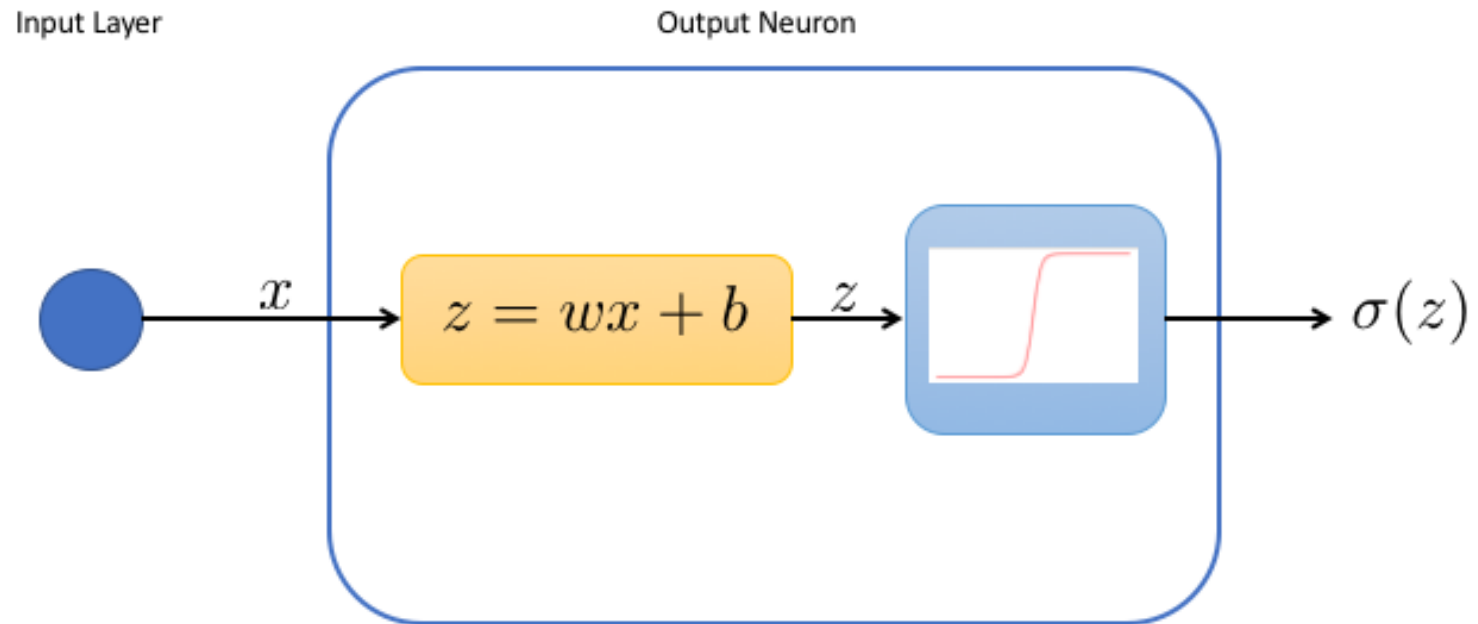
- PASS / FAIL
- SPAM / HAM
- 긍정positive과 부정negative
 - 리뷰 텍스트를 기반으로 영화 리뷰
- 로지스틱 회귀라고도 부름



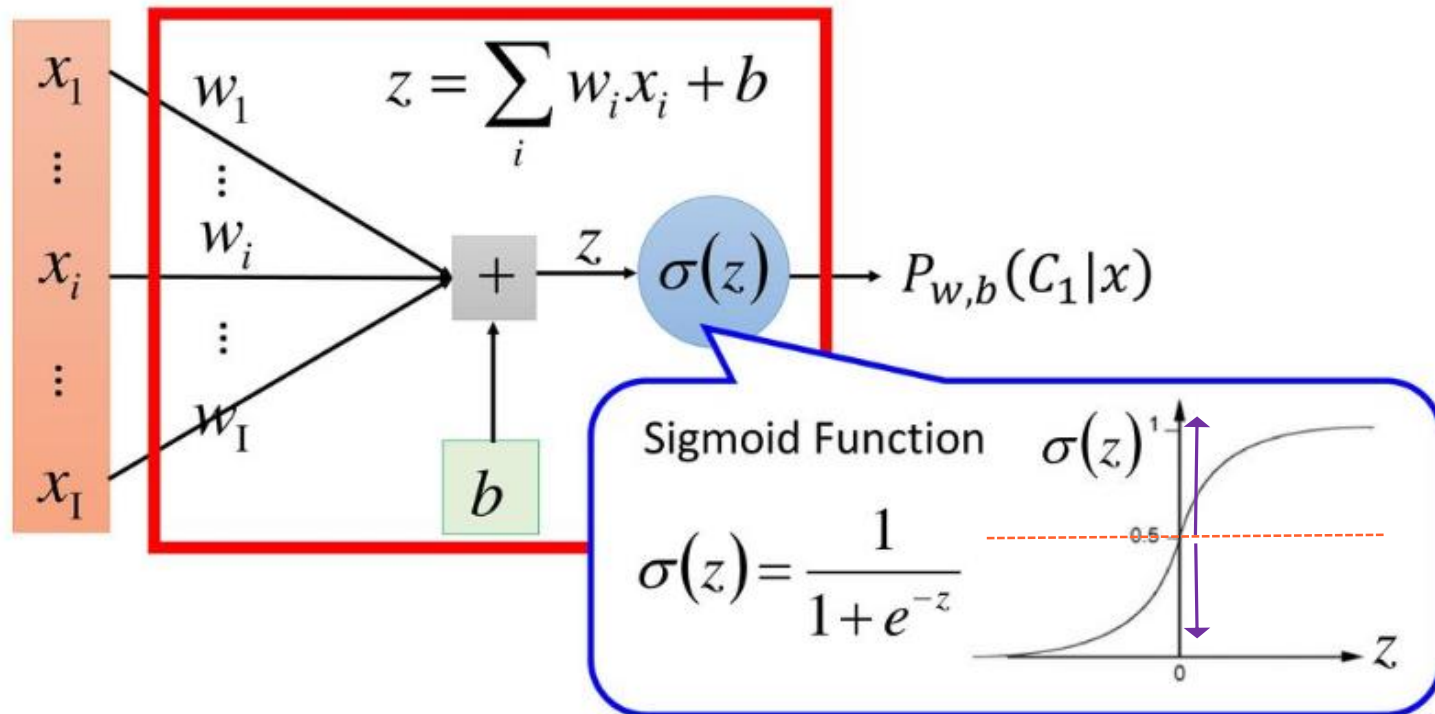
• 결과 기술 방식

- 4 개의 결과
 - 일반 레이블 방식
 - [0, 1, 0, 1]
 - One Hot Encoding 방식
 - [[1, 0], [0, 1], [1, 0], [0, 1]]

이진 분류 개념



이진 분류 활성화 함수



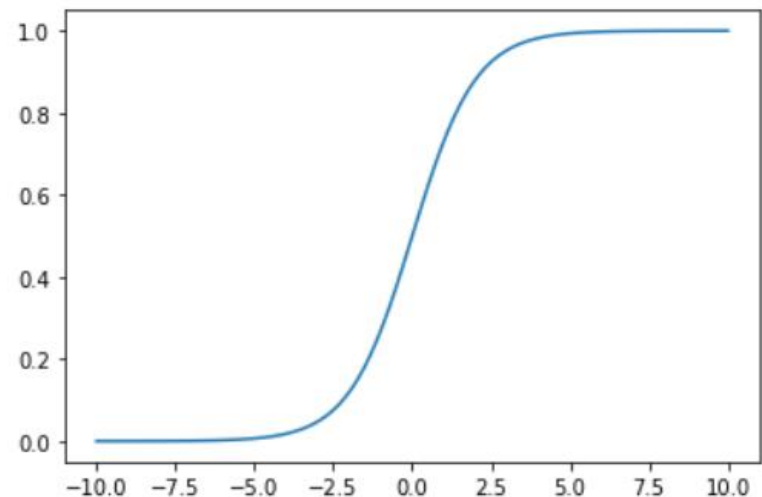
시그모이드 함수

- 이진분류 모델의 출력층에 주로 사용되는 활성화 함수
 - 0과 1사이의 값으로 출력
 - 출력 값이 특정 임계값(예를 들어 0.5) 이상이면 양성
 - 이하이면 음성이라고 판별

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

```
import numpy as np
import matplotlib.pyplot as plt

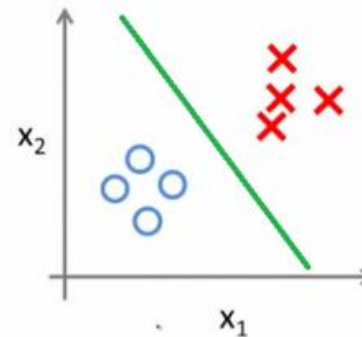
x = np.linspace(-10, 10, 100)
y = 1 / ( 1 + np.exp(-x) )
plt.plot(x, y)
plt.show()
```



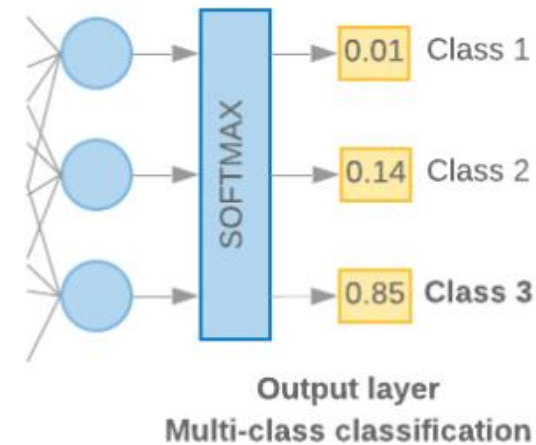
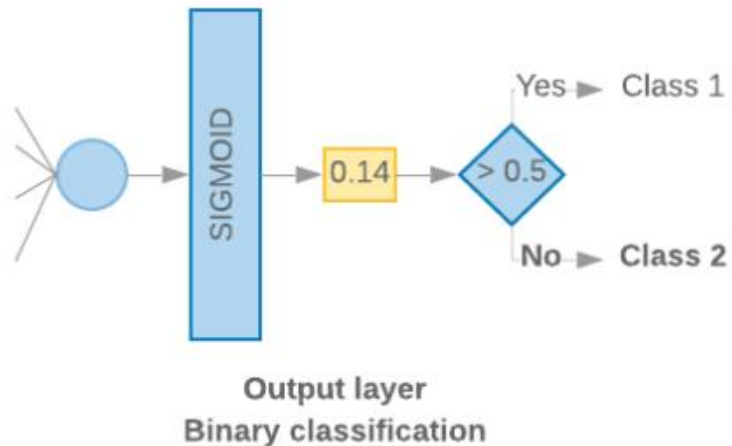
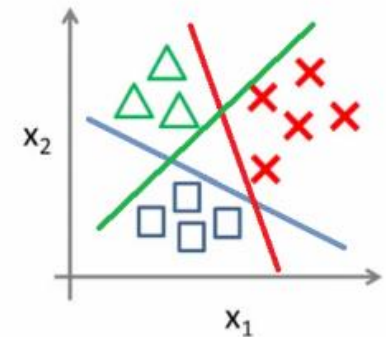
이진 분류와 다중 분류

- 시그모이드 함수와 소프트맥스 함수

Binary classification:

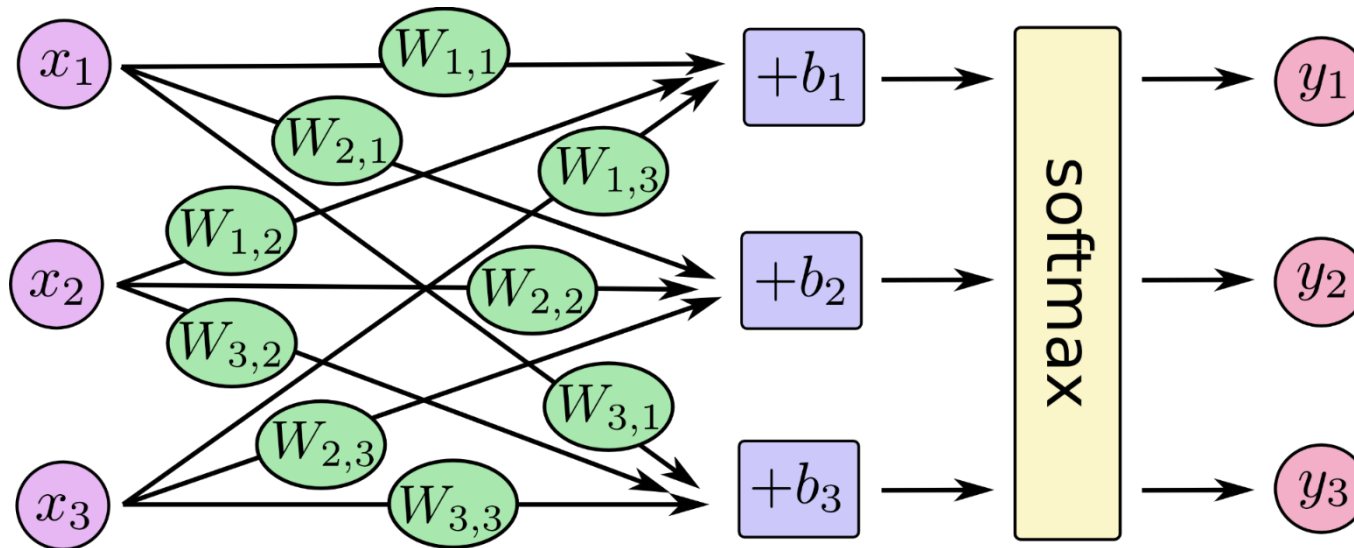


Multi-class classification:



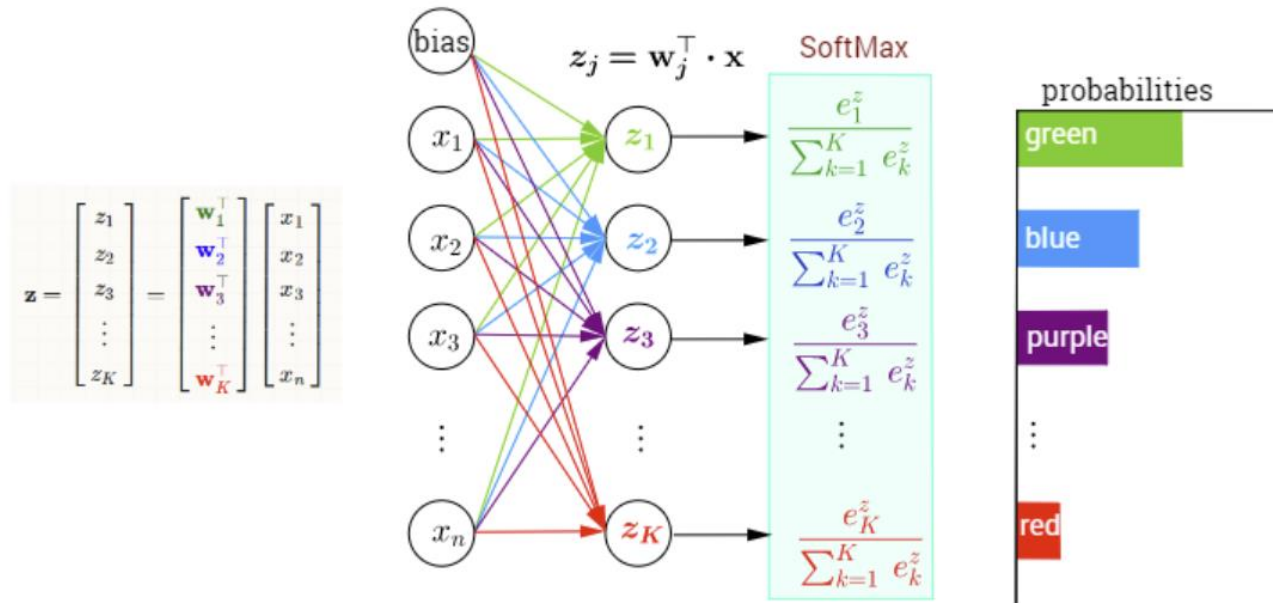
소프트맥스 함수

- 분류의 마지막 활성화 함수로 사용
 - 모든 y_i 의 합은 1
 - 각각의 y_i 는 그 분류의 확률



소프트맥스 함수

- 뉴런의 결과를 e의 지수승으로 하여 모든 합으로 나눈 결과
 - $\exp(x) / \text{tf.reduce_sum}(\exp(x))$



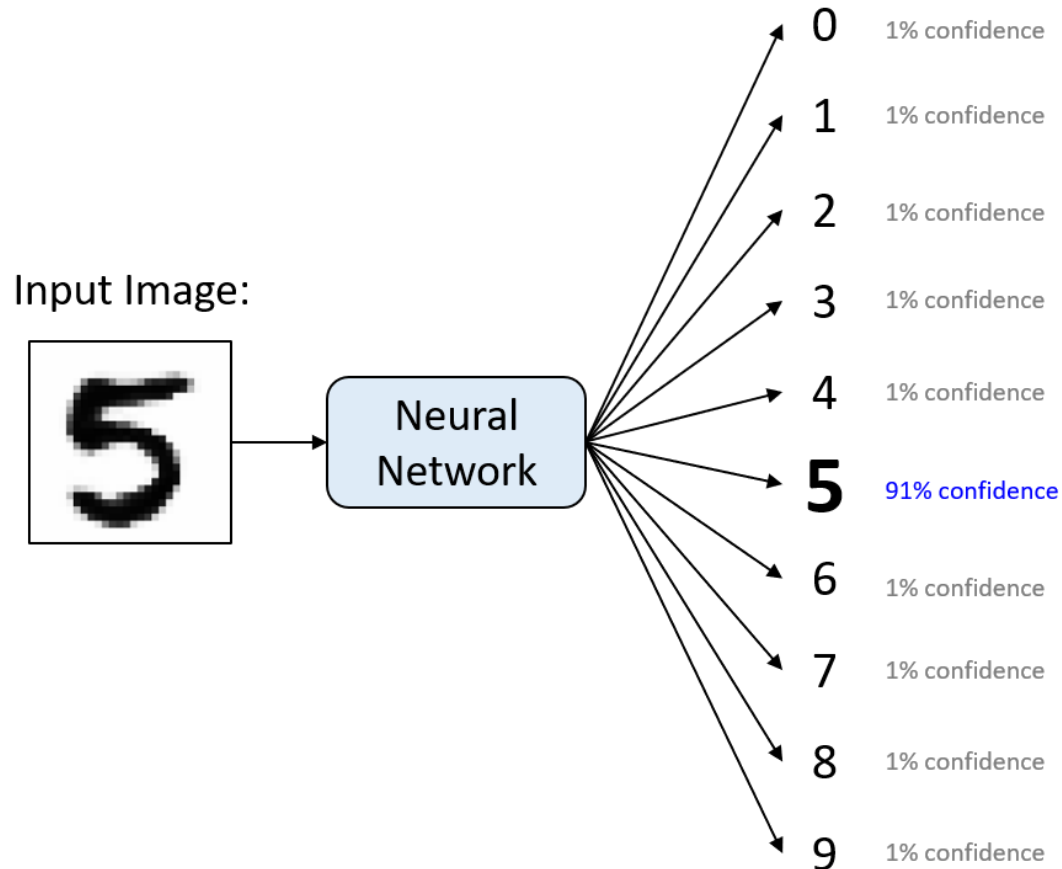
The softmax as

$$\sigma(j) = \frac{\exp(\mathbf{w}_j^T \mathbf{x})}{\sum_{k=1}^K \exp(\mathbf{w}_k^T \mathbf{x})} = \frac{\exp(z_j)}{\sum_{k=1}^K \exp(z_k)}$$

This will result in a normalization of the output adding up to 1, interpretable as a probability mass function.

대표적 다중 분류

- MNIST 손글씨



분류에서의 손실 함수

• 크로스 엔트로피

– 실제 데이터의 결과 값인 y

• $y=1$ 일 때

- 예측 값이 1에 가까워질수록 cost function의 값은 작아져야 함
- 반대로 0에 가까워질수록 cost function의 값이 무한대로 증가하게 되어 예측이 틀렸다는 것을 보여주어야 함

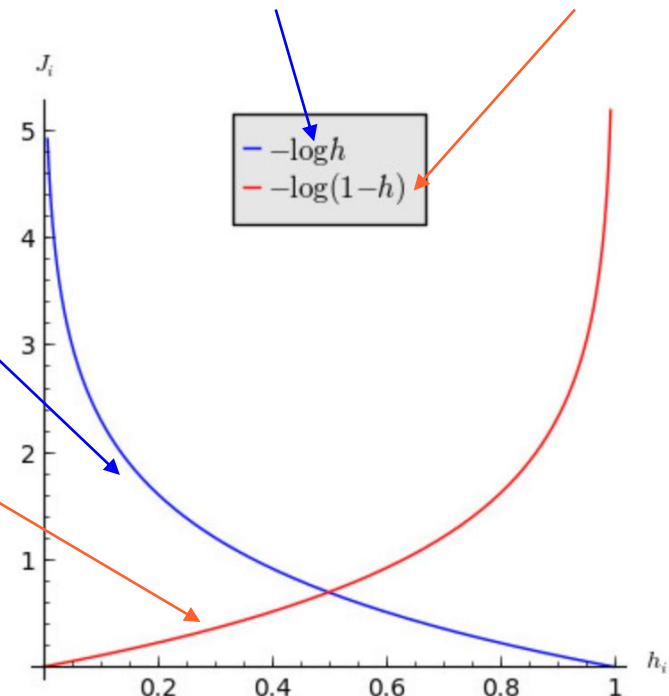
• $y=0$ 일 때

- 예측이 0으로 맞게 되면 cost function은 매우 작은 값을 가지고
- 반대로 예측이 1로 하게 되어 예측에 실패할 경우 cost 값이 무한대로 증가하여 틀렸다는 것을 알 수 있게 해야 함

$$\text{cost}(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$

$$C(H(x), y) = -y \log(H(x)) - (1 - y) \log(1 - H(x))$$



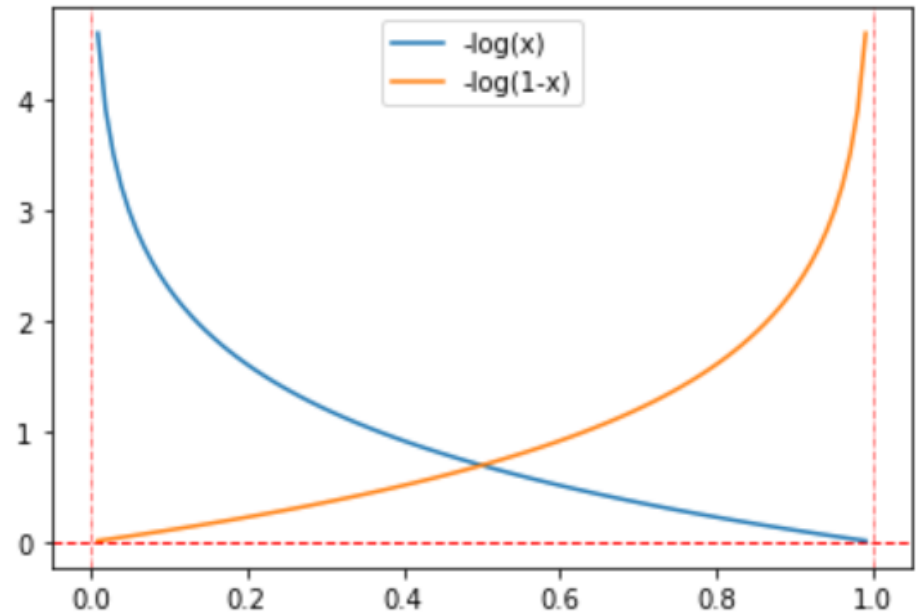
크로스 엔트로피 손실 함수 직접 그리기

- Cross entropy

```
import numpy as np
import matplotlib.pyplot as plt

alpha = 0.1e-1
x = np.linspace(0+alpha, 1-alpha, 100)
y1 = -np.log(x)
y2 = -np.log(1-x)
```

```
plt.axhline(y=0, color='r', linestyle='--', linewidth=1)
plt.axvline(x=1, color='r', linestyle='-.', linewidth=.5)
plt.axvline(x=0, color='r', linestyle='-.', linewidth=.5)
plt.plot(x, y1, label='-log(x)')
plt.plot(x, y2, label='-log(1-x)')
plt.legend(loc='best')
plt.show()
```



크로스 엔트로피 손실 함수

- **tf.keras.losses.categorical_crossentropy**
 - 정답
 - **y_true = [[0, 1, 0], [0, 0, 1]]**
 - 예측
 - **y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]**
 - 함수 적용
 - **Loss = tf.keras.losses.categorical_crossentropy(y_true, y_pred)**
 - 결과
 - **loss.numpy()**

```
1 import tensorflow as tf
2
3 y_true = [[0, 1, 0], [0, 0, 1]]
4 y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
5 loss = tf.keras.losses.categorical_crossentropy(y_true, y_pred)
6 loss.numpy()
```

정답이 2인데, 정답을 1이 0.8로 예측

```
array([0.05129331, 2.3025851 ], dtype=float32)
```

정답이 2인데, 정답을 1이 0.8로
예측해 손실 값이 매우 큼

일반 값 사용 크로스 엔트로피 손실 함수

- **tf.keras.losses.categorical_crossentropy**
 - 정답: 원 핫 인코딩 유형
 - `y_true = [[0, 1, 0], [0, 0, 1]]`
- **tf.keras.losses.sparse_categorical_crossentropy**
 - 정답: 일반 유형
 - `y_true = [[1], [2]]`

```
import tensorflow as tf

# y_true = [[0, 1, 0], [0, 0, 1]]
y_true = [[1], [2]]
y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
loss = tf.keras.losses.sparse_categorical_crossentropy(y_true, y_pred)
loss.numpy()
```

일반 값을 원핫으로 변환

- 일반 값을 원핫으로 변환해 크로스 엔트로피 손실 함수

```
import tensorflow as tf

y_true = [[1], [2]]
y_true = tf.one_hot(y_true, depth=3)
print(y_true)
y_true = tf.reshape(y_true, [-1, 3])
print(y_true)
y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]

loss = tf.keras.losses.categorical_crossentropy(y_true, y_pred)
loss.numpy()
```

```
↳ tf.Tensor(
  [[[0. 1. 0.]

   [0. 0. 1.]]], shape=(2, 1, 3), dtype=float32)
tf.Tensor(
  [[0. 1. 0.]

   [0. 0. 1.]], shape=(2, 3), dtype=float32)
array([0.05129331, 2.3025851 ], dtype=float32)
```

크로스 엔트로피 손실 값 직접 계산

$$\text{cost}(W) = \frac{1}{m} \sum c(H(x), y)$$

$$c(H(x), y) = \begin{cases} -\log(H(x)) & : y = 1 \\ -\log(1 - H(x)) & : y = 0 \end{cases}$$

```
import tensorflow as tf
import numpy as np
```

```
y_true = tf.reshape(tf.one_hot([[1], [2]], depth=3), [-1, 3])
y_pred = [[0.05, 0.95, 0], [0.1, 0.8, 0.1]]
```

```
loss = tf.keras.losses.categorical_crossentropy(y_true, y_pred)
print(loss.numpy())
```

```
print(-np.log(0.95), -np.log(0.1))
```

```
↳ [0.05129331 2.3025851 ]
   0.05129329438755058 2.3025850929940455
```


Softmax 함수

- 확률 값
 - 결과를 모두 더하면 1

```
import tensorflow as tf
import numpy as np

a = np.array([[0.3, 2.9, 4.0]])
sm = tf.keras.activations.softmax(tf.convert_to_tensor(a))
print(sm.numpy())
```

```
↳ [[0.01821127 0.24519181 0.73659691]]
```