로지스틱 리그레션 설명

쉽게 설명하자면 선형 회귀를 시그모이드 함수에 대입해 이진분류에 활용한 것입니다.

(회귀식 하나 적기)

Y = w1x1+ w2x2+w3x3 + ... B

$$\frac{1}{1+\mathrm{e}^{-\mathrm{y}}} = \frac{1}{1+e^{(-\Sigma wx+b)}}$$

이제 시그모이드 함수에 한 가지 가정을 해보겠습니다. Y 값을 logit이라 가정하는 것입니다.

Logit은 ~~

이를 식에 대입해 보이면 $\frac{1}{1+\mathrm{e}^{-\log_{\mathrm{i}}\mathrm{t}}} = \frac{1}{1+\mathrm{e}^{\log\left(\frac{\mathrm{P}}{1-\mathrm{P}}\right)}}$ 결국 P가 나옴.

한 가지 의문점이 남을 수 있습니다. 방금 시그모이드 함수로 구한 값과 소프트 맥스 함수로 구한 값의 대소관계가 같다는 것입니다. 분류 문제라고 하면 이는 당연히 가장 큰 값을 가지므로 모두 X1이라고 결과를 예측할 것입니다.

그럼 소프트 맥스 함수나 시그모이드 함수 둘 중 하나만 쓰면 되는 것이라는 결론에 도달할 것입니다.Sigmoid가 가지지 못한 softmax의 특성이 있습니다.바로 입력값이 클수록 더 큰 확률값을 도출한다는 것입니다. 이는 학습에서 중요한 작용을 합니다. 언어 모델의 경우 각 단어에 대한 확률값을 업데이트하면서 학습을 하는데 이때 확률 값의 차이가 크다는 것은 확률의 업데이트 즉 학습이 더 크게 된다는 것을 의미합니다.

정리를 해보겠습니다....

로지스틱, 시그모이드, 소프트맥스 ....

3.4.3 구현 정리

[81]:
```
def sigmoid(x):
    return 1/(1+np.exp(-x))

def identity_function(x):
    return x

def init_network():   # 함수의 가중치와 편향을 초기화하고 이들을 딕셔너리 변수인 network에 저장
    network = {}      # network는 각 층에 필요한 매개변수(가중치 편향)를 저장
    network['W1'] = np.array([[0.1, 0.3, 0.5],[0.2, 0.4, 0.6]])
    network['b1'] = np.array([0.1, 0.2, 0.3])
    network['W2'] = np.array([[0.1, 0.4],[0.2, 0.5],[0.3, 0.6]])
    network['b2'] = np.array([0.1, 0.2])
    network['W3'] = np.array([[0.1, 0.3],[0.2,0.4]])
    network['b3'] = np.array([0.1, 0.2])
    return network

def forward(network, x):    # 입력 신호를 출력으로 변환하는 처리 과정을 모두 구현. 신호가 순방향으로 전달되므로 forward.
    W1, W2, W3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
    a1 = np.dot(x, W1)+b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2)+b2
    z2 = sigmoid(a2)
    a3 = np.dot(z2, W3)+b3
    y = identity_function(a3)   # 출력층으로 신호 전달할 땐 항등 함수를 활성화 함수로!
    return y

network = init_network()
x = np.array([1.0, 0.5])
y = forward(network, x)
print(y)
```

```
[0.31682708 0.69627909]
```

3.5 출력층 설계하기

- 기계학습 문제
  - 분류 : 데이터가 어느 클래스에 속하느냐의 문제
    - ex) 사진 속의 인물의 성별 분류
  - 회귀 : 입력 데이터에서 (연속적인) 수치를 예측하는 문제
    - ex) 사진 속 인물의 몸무게 예측

[1]:
```
import sys, os
sys.path.append(os.pardir)  # 부모 디렉터리의 파일을 가져올 수 있도록 설정
import numpy as np
from mnist import load_mnist

(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=False)

#각 데이터의 형상 출력
print(x_train.shape)
print(t_train.shape)
print(x_test.shape)
print(t_test.shape)
```

```
Downloading train-images-idx3-ubyte.gz ...
Done
Downloading train-labels-idx1-ubyte.gz ...
Done
Downloading t10k-images-idx3-ubyte.gz ...
Done
Downloading t10k-labels-idx1-ubyte.gz ...
Done
Converting train-images-idx3-ubyte.gz to NumPy Array ...
Done
Converting train-labels-idx1-ubyte.gz to NumPy Array ...
Done
Converting t10k-images-idx3-ubyte.gz to NumPy Array ...
Done
Converting t10k-labels-idx1-ubyte.gz to NumPy Array ...
Done
Creating pickle file ...
Done!
(60000, 784)
(60000,)
(10000, 784)
(10000,)
```

정리

- 신경망의 순전파는 각 층의 뉴런들이 다음 층의 뉴런으로 신호를 전달한다는 점에서 퍼셉트론과 같지만, 다음 뉴런으로 갈 때 신호를 변화시키는 활성화 함수에 큰 차이가 있다. 신경망에서는 매끄럽게 변화하는 시그모이드 함수를, 퍼셉트론에서는 갑자기 변화하는 계단 함수를 활성화 함수로 사용했다.

  - 신경망에서는 활성화 함수로 시그모이드 함수와 ReLU 함수 같은 매끄럽게 변화하는 함수를 이용한다.
  - 넘파이의 다차원 배열을 잘 사용하면 신경망을 효율적으로 구현할 수 있다.
  - 머신러닝 문제는 크게 회귀와 분류로 나눌 수 있다.
  - 출력층의 활성화 함수로는 회귀에서는 주로 항등 함수를, 분류에서는 소프트맥스 함수를 이용한다.
  - 분류에서는 출력층의 뉴런 수를 분류하려는 클래스 수와 같게 설정한다.
  - 입력 데이터를 묶은 것을 배치라 하며, 추론 처리를 배치 단위로 진행하면 훨씬 빠르게 결과를 얻을 수 있다.