Softmax Activation

```
import warnings
warnings.filterwarnings('ignore')
```

1) sigmoid( ) & softmax( ) 정의

- sigmoid( )

```
import numpy as np

def sigmoid(x):
    y = 1 / (1 + np.exp(-x))
    return y
```

- sigmoid( ) 시각화

```
import matplotlib.pyplot as plt

n = np.linspace(-10.0, 10.0, 2000)

plt.figure(figsize = (9, 6))
plt.plot(n, sigmoid(n))
plt.show()
```

- softmax( )

```
def softmax(x):
    m = np.max(x)
    sm = np.exp(x - m)/np.sum(np.exp(x - m))
    return(sm)
```

- softmax( ) 시각화

```
import matplotlib.pyplot as plt

n = np.linspace(-10.0, 10.0, 2000)

plt.figure(figsize = (9, 6))
plt.plot(n, softmax(n))
plt.show()
```

2) sigmoid( ) vs. softmax( ) 결과 비교

- 가상의 y_hat

```
y_hat = np.array([5, 0, -3])
```

- Sigmoid Activation 적용

```
np.set_printoptions(suppress = True, precision = 5)

print(sigmoid(y_hat))
print('%.5f' % np.sum(sigmoid(y_hat)))
```

```
[0.99331 0.5     0.04743]
1.54073
```

- Softmax Activation 적용

```
np.set_printoptions(suppress = True)

print(softmax(y_hat))
print('%.5f' % np.sum(softmax(y_hat)))
```

```
[0.99298 0.00669 0.00033]
1.00000
```

3) 추가 학습 진행 후 변화 비교

- 가상의 y_hat 업데이트

```
y_hat = np.array([10, -2, -9])
```

- Sigmoid Activation 재적용

```
print(sigmoid(y_hat))
print('%.5f' % np.sum(sigmoid(y_hat)))
```

```
[0.99995 0.1192  0.00012]
1.11928
```

- Softmax Activation 재적용

```
print(softmax(y_hat))
print('%.5f' % np.sum(softmax(y_hat)))
```

```
[0.99999 0.00001 0.     ]
1.00000
```

#

#

#

The End

#

#

#