





## Classification Problem

	2	<del><math>\frac{2}{2+1+0}</math></del>		$\frac{e^2}{e^2 + e^1 + e^0}$	=	0.67	P(duck)
	1	<del><math>\frac{1}{2+1+0}</math></del>		$\frac{e^1}{e^2 + e^1 + e^0}$	=	0.24	P(beaver)
	0	<del><math>\frac{0}{2+1+0}</math></del>		$\frac{e^0}{e^2 + e^1 + e^0}$	=	0.09	P(walrus)

## Softmax Function

LINEAR FUNCTION

SCORES:

$z_1, \dots, z_n$

$$P(\text{class } i) = \frac{e^{z_i}}{e^{z_1} + \dots + e^{z_n}}$$

QUESTION

Is Softmax for  $n=2$  values the same as the sigmoid function?

```
In [1]: import numpy as np
import math

# Write a function that takes as input a list of numbers, and returns
# the list of values given by the softmax function.
def softmax(L):
    softOutput = []
    x = 0
    denom = sum([x + math.exp(i) for i in L])
    for i in L:
        softOutput.append(math.exp(i)/denom)
    return softOutput
```

```
In [2]: L = [2,1,0]
softmax(L)
```

Out[2]: [0.6652409557748219, 0.24472847105479764, 0.09003057317038046]

In [3]: **import** numpy **as** np

```
# Write a function that takes as input a list of numbers, and returns  
# the list of values given by the softmax function.
```

```
def softmax(L):  
    softOutput = []  
    x = 0  
    denom = sum([x + np.exp(i) for i in L])  
    for i in L:  
        softOutput.append(np.exp(i)/denom)  
    return softOutput
```

In [4]: L = [2,1,0]  
softmax(L)

Out[4]: [0.6652409557748219, 0.24472847105479764, 0.09003057317038046]

In [5]: np.exp(L)

Out[5]: array([7.3890561 , 2.71828183, 1. ])

In [6]: **import** numpy **as** np

```
def softmax(L):  
    expL = np.exp(L)  
    sumExpL = sum(expL)  
    result = []  
    for i in expL:  
        result.append(i*1.0/sumExpL)  
    return result
```

```
# Note: The function np.divide can also be used here, as follows:
```

```
def softmax1(L):  
    expL = np.exp(L)  
    return np.divide (expL, expL.sum())
```

In [7]: L = [2,1,0]  
softmax(L)

Out[7]: [0.6652409557748219, 0.24472847105479764, 0.09003057317038046]