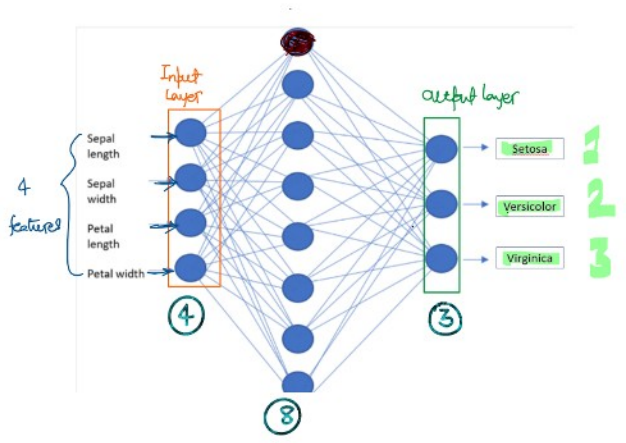
MLP Hands-on & Code Explanation

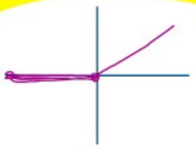
19 October 2025 11:06

IRIS data MLP model architecture



```
##### ADD SOME ACTIVATION FUNCTIONS

#1. ReLU Activation Function
def relu(self, z):
    return np.maximum(0,z)
```



is a NumPy element-wise comparison function
- it compares two arrays element by element and returns the larger value at each position

```
#### Derivative of ReLU for backpropagation
def relu_derivative(self, z):
    return np.where(z>0, 1, 0)
```

np.where (condition, x, y)

return x → where condition is true
return y → where condition is false.

```
### Softmax Activation Function
def softmax(self, z):
    exp_values = np.exp(z - np.max(z, axis=1, keepdims=True)) #subtract max for numerical stability -- numerical instability (read about it)
    return exp_values/np.sum(exp_values, axis=1, keepdims= True)
```

SOFTMAX ACTIVATION FUNCTION EXPLANATION

```
]: z = [[2.33, -1.46, 0.56]]   # shape (1,3)

]: softmax(z)

]: array([[0.83827314, 0.01894129, 0.14278557]])
```

Let us take an example:

For one sample row from IRIS dataset

$\hat{y}_i = \frac{e^{z_i}}{\sum_{k=1}^{C} e^{z_k}}$

Raw scores — raw score to probability

Class#1 setosa $z_1 = \boxed{2.33} \rightarrow P(\text{class}\#1) = \frac{e^{2.33}}{\underbrace{e^{2.33} + e^{-1.46} + e^{0.56}}_{D = 12.26}} = \frac{e^{2.33}}{12.26} = 0.8382 \rightarrow 84\%$ ← highest probability → Predicted class # Setosa

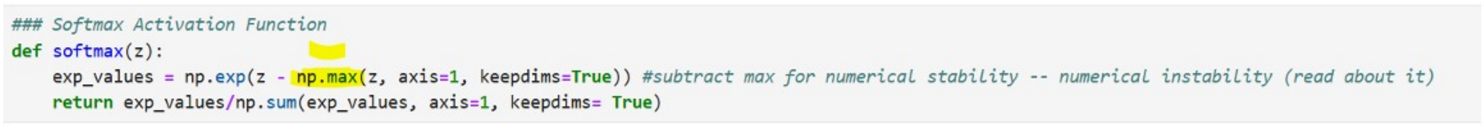
Class#2 versicolor $z_2 = \boxed{-1.46} \rightarrow P(\text{class}\#2) = \frac{e^{-1.46}}{D} = \frac{e^{-1.46}}{12.26} = 0.0189 \rightarrow 2\%$

class#3 virginica $z_3 = \boxed{0.56} \rightarrow P(\text{class}\#3) = \frac{e^{0.56}}{D} = \frac{e^{0.56}}{12.26} = 0.1427 \rightarrow 14\%$

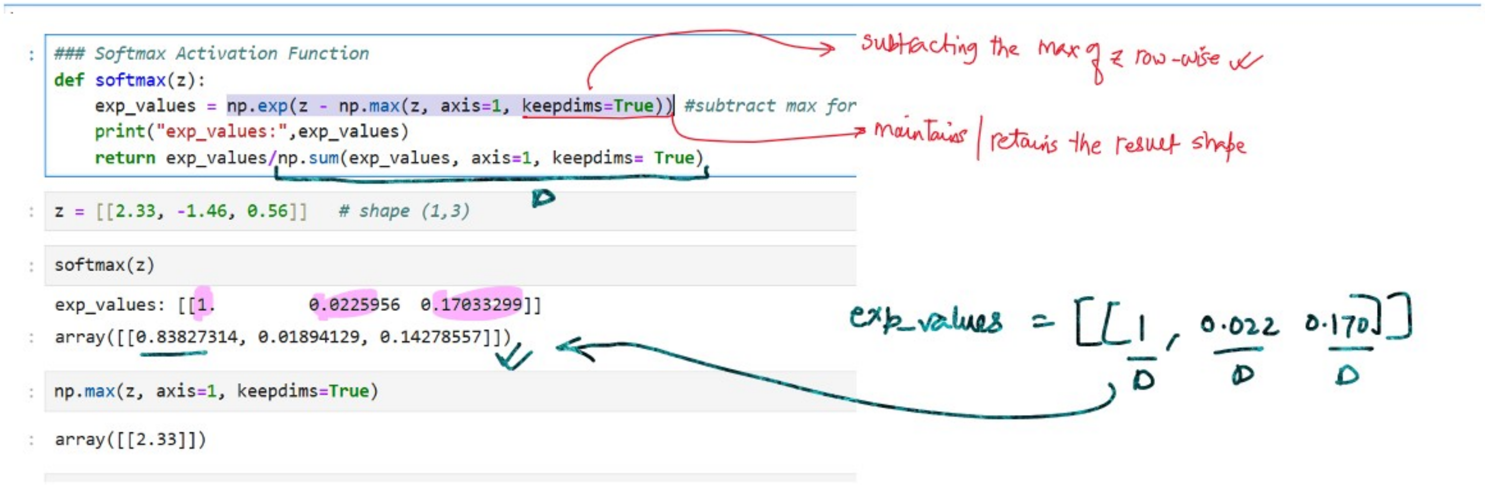
$\boxed{C=3}$

$K = 1$ to 3

Total = 100%



```
### Softmax Activation Function
def softmax(z):
    exp_values = np.exp(z - np.max(z, axis=1, keepdims=True)) #subtract max for numerical stability -- numerical instability (read about it)
    return exp_values/np.sum(exp_values, axis=1, keepdims= True)
```



```
### Softmax Activation Function
def softmax(z):
    exp_values = np.exp(z - np.max(z, axis=1, keepdims=True)) #subtract max for
    print("exp_values:",exp_values)
    return exp_values/np.sum(exp_values, axis=1, keepdims= True)
```

subtracting the max of z row-wise ✓

maintains | retains the result shape

D

```
z = [[2.33, -1.46, 0.56]]   # shape (1,3)
```

```
softmax(z)
```

```
exp_values: [[1.         0.0225956  0.17033299]]
array([[0.83827314, 0.01894129, 0.14278557]])
```

✓✓

$$exp\_values = \left[\left[\frac{1}{D}, \frac{0.022}{D} \; \frac{0.170}{D}\right]\right]$$

```
np.max(z, axis=1, keepdims=True)
```

```
array([[2.33]])
```