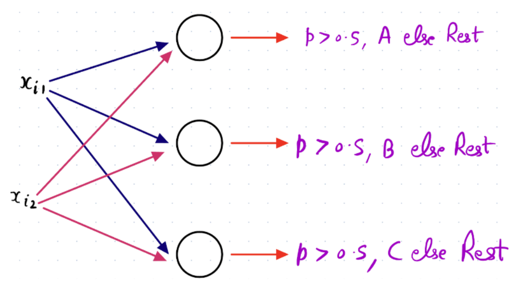
NN: Forward and Back Propagation

How can we adapt a NN to do multi-class classification?

Suppose we wish to do multi-class classification for 3 classes: A, B, and C.

Recall multi-class classification:

- We calculated the probability that a given data point belongs to class A, B, or C respectively.
- Then, we returned the class with the highest probability as the answer.
- This gives us the intuition that perhaps, our output layer should have 3 outputs. One for each class.



We cannot perform multi-class classification using a sigmoid because we might get $p > 0.5$ for more than one class.

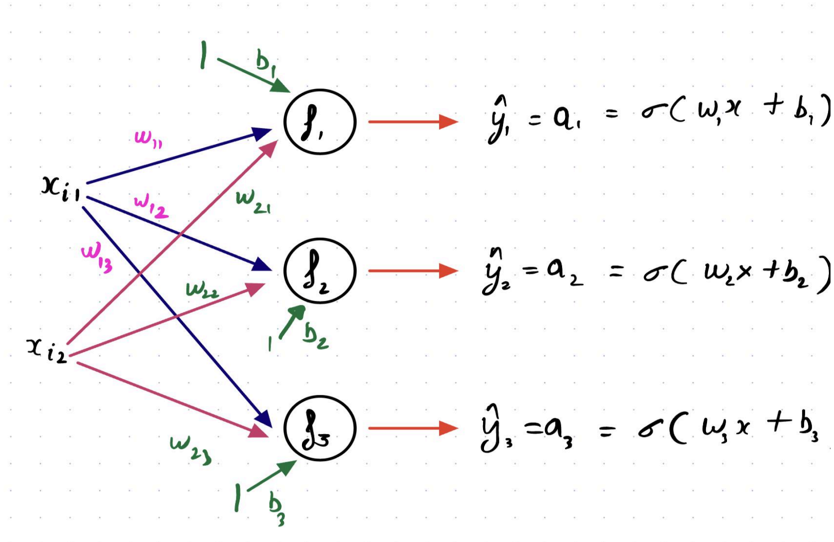
- The model will predict the presence of multiple classes in the output - [1, 1, 0], [1, 1, 1], [1, 0, 1]
- Conclusion: We want these probabilities values to sum to 1, as we had in Logistic Regression (p and 1-p).

It should be $p_A + p_B + p_C = 1$

To do this, we use the **softmax function** as activation in the neurons of the **output layer**.

$$p_i = \frac{e^{z_i}}{\sum_{i=0}^{k} e^{z_i}}$$

The model now looks like this:-



Note:

- There will be 2*3 = 6 weights, and it'll be stored as a 2x3 matrix: W_{2x3}
- There will be 3 biases, one for each neuron, and it'll be stored as a 1x3 matrix: b_{1x3}

How will we calculate the loss for this multi-class classification NN Model?
We use Categorical Cross Entropy.

Cross Entropy (CE_i) for i^{th} datapoint will be:

$$CE_i = -\sum_{j=1}^{k} y_{ij} log(P_{ij})$$

where,
k -> number of classes
y_{ij} -> one hot encoded label. Ex: [1,0,0], [0,1,0], or [0,0,1]
P_{ij} -> Calculated Probability of datapoint belonging to class j

This can be seen as log loss extended to the multiclass setting. For k=2, we will get log loss formulation.

How to train NN?
Let, m -> no of training examples
d -> no of features
n -> no of classes/neurons in the output layer

The process to train a NN is:-

- Randomly Initialise parameters: W and b matrices
- Do forward propagation
  - $Z = X_{m\,x\,d}.W_{d\,x\,n} + b_{1\,x\,n}$
  - $A = activation(Z)$
- Calculate the Loss
- Repeat until Loss converges
  - update $w_i = w_i - lr * (\partial Loss/\partial w_i)$
  - calculate the output using hypothesis and updated params
  - calculate the Loss