|  |  |  |
| --- | --- | --- |
| BASIS | SIGMOID | SOFTMAX |
| 1. Range 2. Function/ Formula 3. Input Data 4. Dependency of classes 5. After application effect 6. Output Logic | σ: R → [0,1]  For any value of x, Sigmoid(x) lies between [0,1].  The input instance is not restricted to only one class, it can contain one or more or even of the classes at once as well.  Inclusion of a class does not depend on the occurrence of other class.  Probabilities/ Logits of the individual class predictions are not required to sum up to 1.  Define a threshold and accept **all** classes to be present beyond the selected threshold. | S: Rc → [0,1]c  The range is not independent for any x. It depends on the sum of the softmax values of the rest of the classes.    An input instance can belong to only one of the classes, not more.  Occurrence of a particular class in an instance means any other class cannot be included in the same instance.  The classes are considered mutually exclusive and exhaustive. Their probabilities sum to 1.  Take the maximum (**only one**) of the predicted probabilities. |

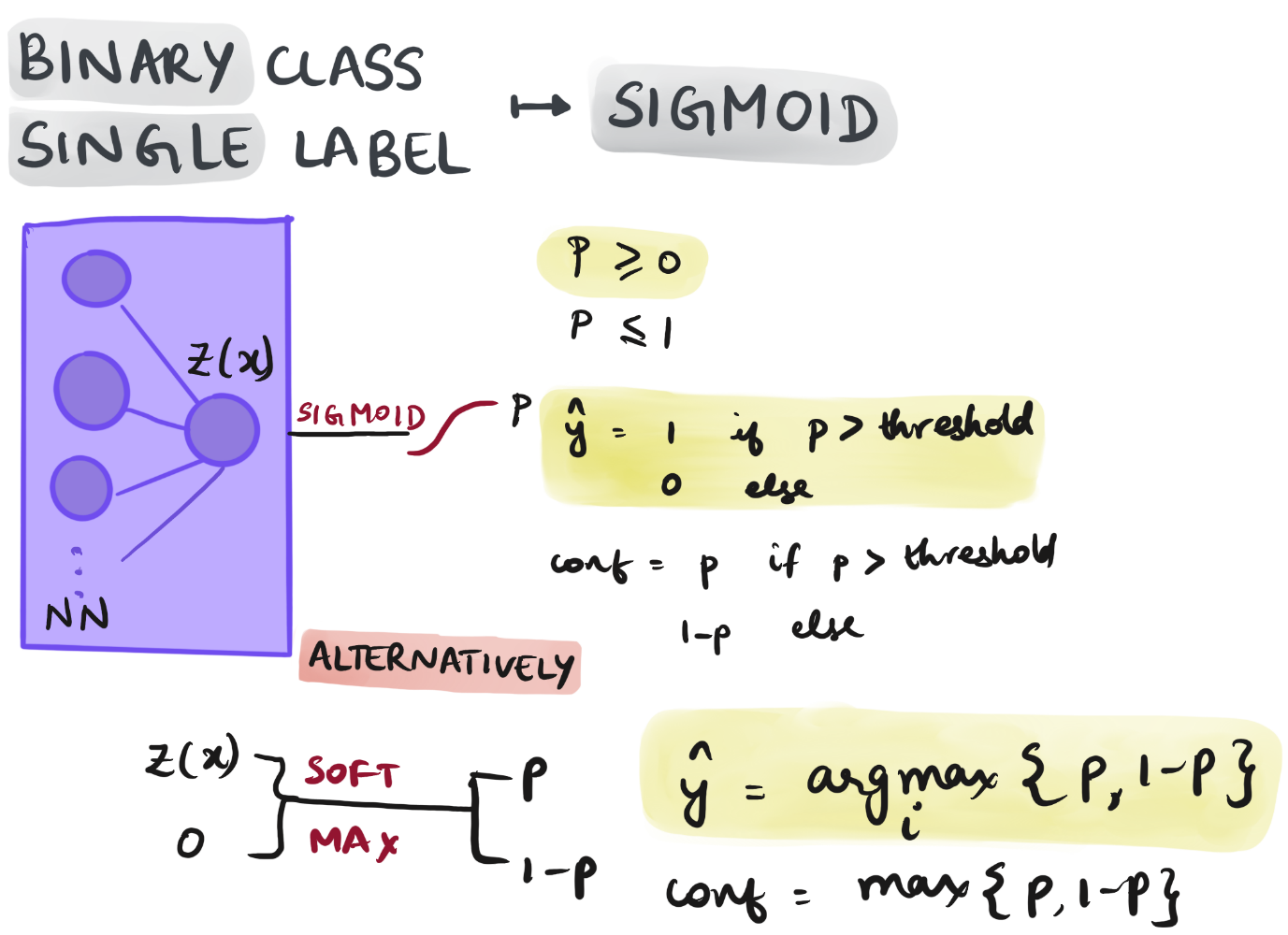
Binary Class

Multi Class

Single Label

Multi Label

|  |  |
| --- | --- |
| SIGMOID  SOFTMAX | SOFTMAX |
| Doesn’t make sense | SIGMOID |



You can always formulate the binary classification problem in such a way that both sigmoid and softmax will work. However, you should be careful to use the right formulation.

**Sigmoid** can be used when your last dense layer has a single neuron and outputs a single number which is a score. Sigmoid then maps that score to the range [0,1]. You can then assume that this is a probability distribution and say that the prediction is class 1 if the probability is larger than 0.5 and class 0 otherwise.

If you want to use **softmax**, you need to adjust your last dense layer such that it has two neurons. It must output two numbers which corresponds to the scores of each class, namely 0 and 1. Now, you can use softmax to convert those scores into a probability distribution. Finally, to get the predicted label, you still need to find the maximum in the probability distribution.

You cannot use softmax when you have a single neuron in the last layer. This will lead to some strange behaviour and performance will drop. Obviously, you can also not use sigmoid when you formulate the problem with two-dimensional last layer.

So it is either -

*model.add(Dense(2, activation='softmax'))*

or

*model.add(Dense(1, activation='sigmoid'))*

