Activation functions

- Relu
- Sigmoid
- Softmax
- Tanh

Layers
Loss functions
Models

Activation functions

Activation functions are mathematical functions applied to the output of a neuron in neural network. They introduce non-linearity into the model so it can learn more complex patterns.

An Activation class is Layer subclass that can be attached to a layer (another Layer subclass) in the network.

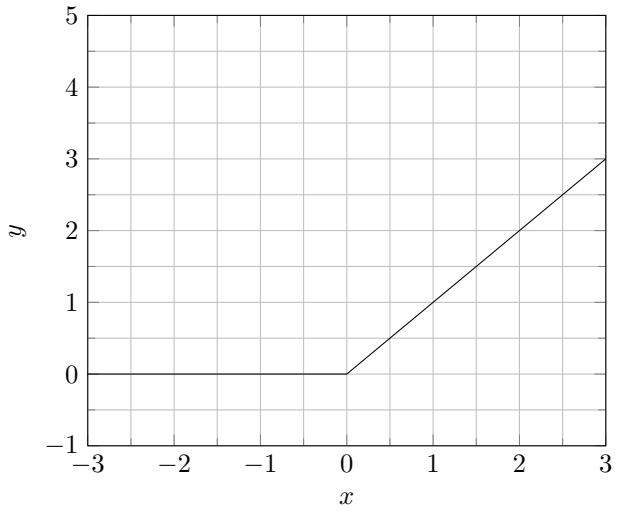
Activation class:

```
class Activation(Layer):
    def __init__(self, activation_function) -> None:
        activation = activation_functions.get_function(
                                            activation_function)
        self.name = activation_function
        self.activation = activation[0]
        self.activation_derivative = activation[1]

    def forward(self, input_data):
        self.input = input_data
        self.output = self.activation(self.input)
        return self.output

    def backward(self, output_error):
        return self.activation_derivative(self.input, output_error)
```

ReLU function

ReLU (Rectified Linear Unit) is an activation function that returns 0 for all negative input values and the input value itself for all positive values.

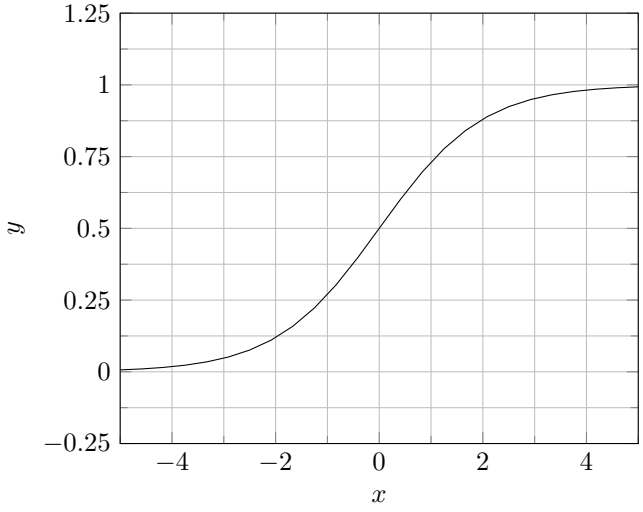
$$f(x) = max(0, x)$$



Sigmoid function

Sigmoid is an activation function that maps input values to a range between 0 and 1.

$$f(x) = \frac{1}{1 + e^{-x}}$$



Softmax function

Softmax is an activation function used for multi-class classification problems. It takes a vector of raw scores (logits) and converts them into a probability distribution over multiple classes meaning that each value will be in range $[0, 1]$ and the sum of those values is 1.

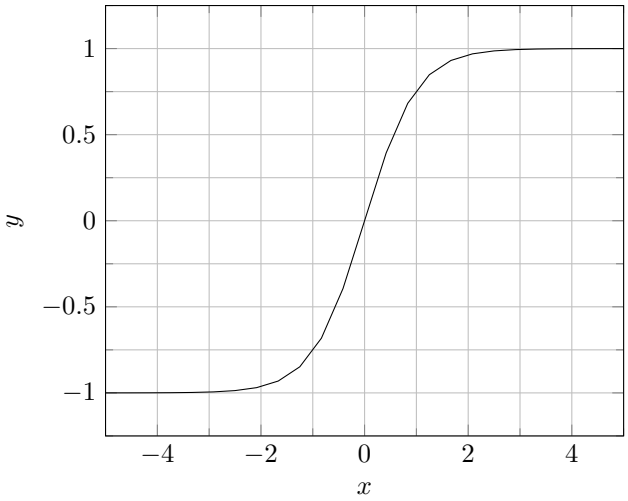
$$y(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}$$

NB! Because the derivative for Softmax function is calculated together with Categorical Cross Entropy loss, they can only be used in combination with each other. The last layer of the model should have Softmax as the activation function and the model should use Categorical Cross Entropy loss.

Tanh function

Tanh is an activation function that maps input values to a range between -1 and 1.

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



Layers

Layer is a functional unit that takes input data, applies a mathematical transformation to it and produces an output. Each layer consists of a set of neurons each of which computes a weighted sum of its inputs and applies an activation function to the result.

Base layer class:

```
class Layer:
    def __init__(self) -> None:
        self.input = None
        self.output = None

    def forward(self, input):
        raise NotImplementedError

    def backward(self, output_error, learning_rate):
        raise NotImplementedError

    def update(self, learning_rate):
        raise NotImplementedError
```

Each layer has a forward() function, backward() function and update() function. Forward function calculates the output given an input, backward function calculates the gradients that will be applied to the weights and update function updates the weights accordingly.

Loss functions

Models