Activation Functions

 Romeo Kienzler

$$f(x) = sigmoid(x)$$
$$z_2 = XW_1$$
$$a_2 = f(z_2)$$
$$z_3 = a_2 W_2$$
$$\hat{y} = a_3 = f(z_3)$$

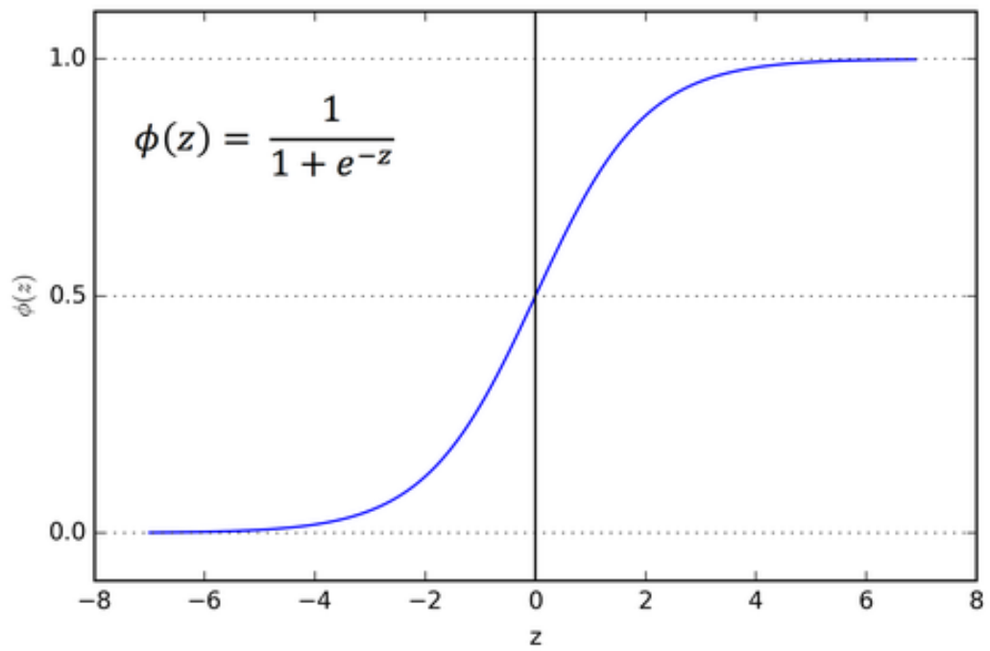


Image Credit:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

- add non-linearity to the model
- Universal Function approximator with single hidden layer

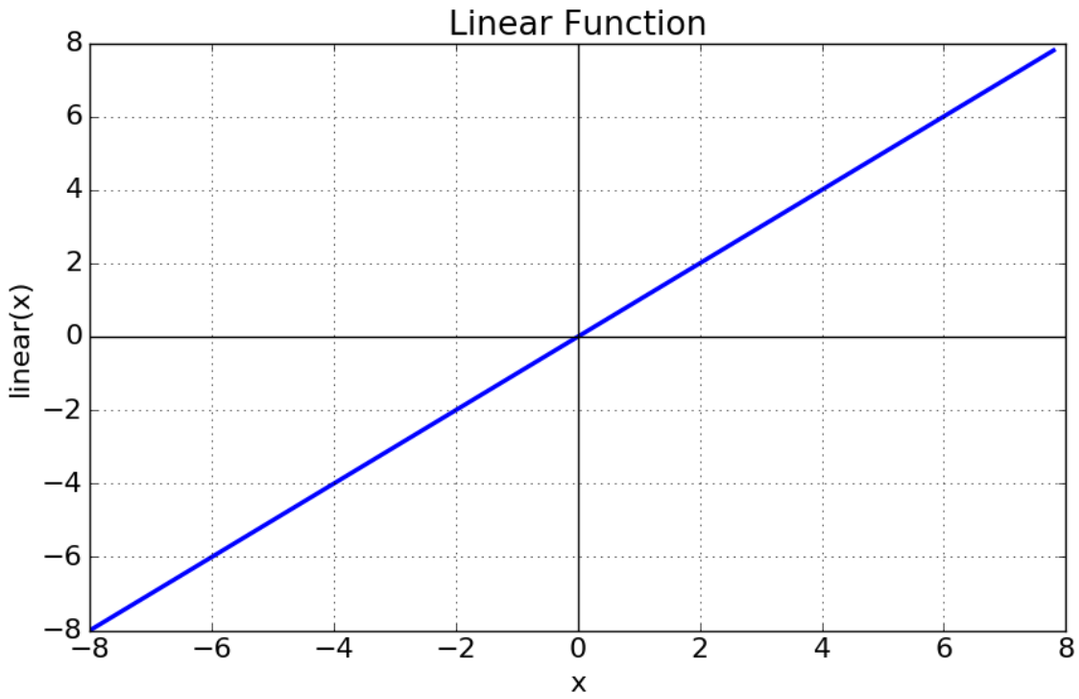


Image Credit:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

- but can add as many layers with linear activation, never get UFA property

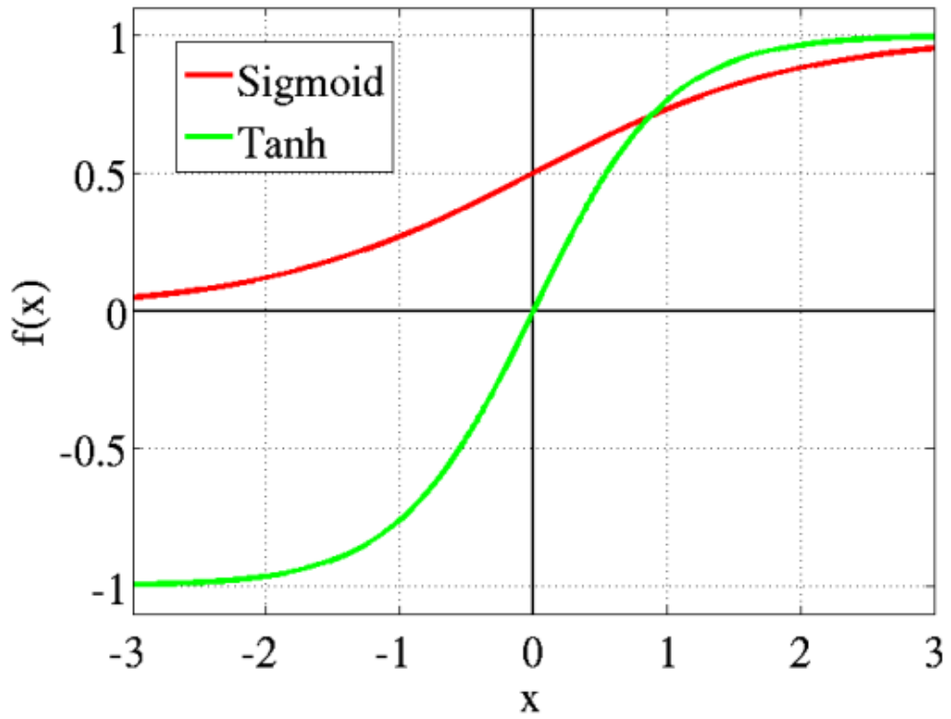


Image Credit:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

- better than logistic sigmoid since also covers negative range
- mostly used in output layer in binary classification

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

softmax

- mostly used in output layer in multiclass-classification

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

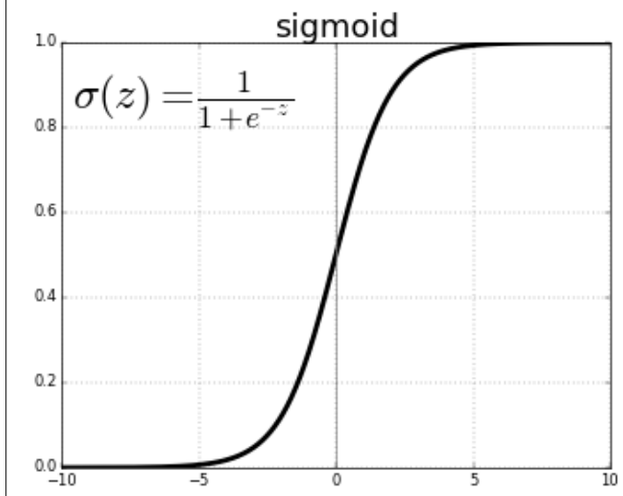
$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

$$\sigma : \mathbb{R}^K \to [0,1]^K$$

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$

so it has the property that the sum of all classes is one, and the most probable class approximates one whereas values for the other classes approximate zero. in case training was successful … with a one hot encoded vector .. a one hot encoded vector is a way to express class membership by just setting one element to one, the rest to zero



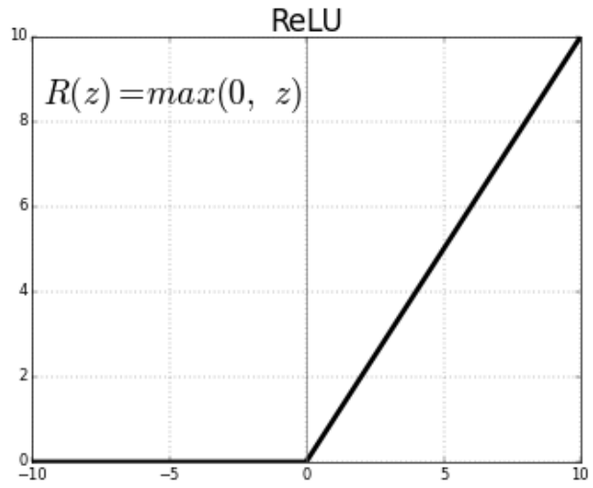
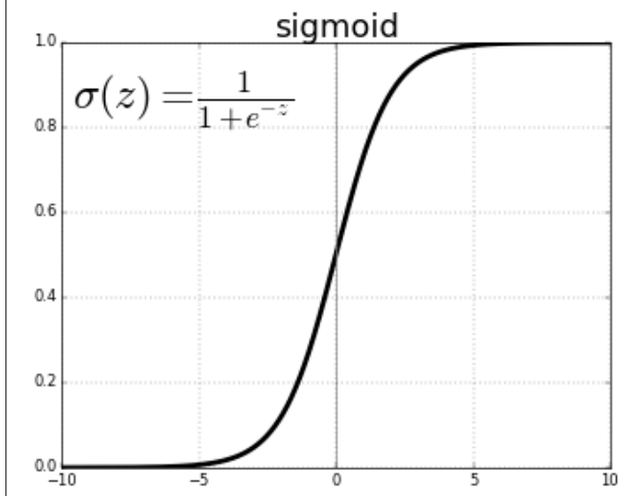


Image Credit:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

- so doesn’t this function look linear?
- not, not a straight line
- with multiple relu functions summed up we can build arbitrary functions
- mostly used activation function today …



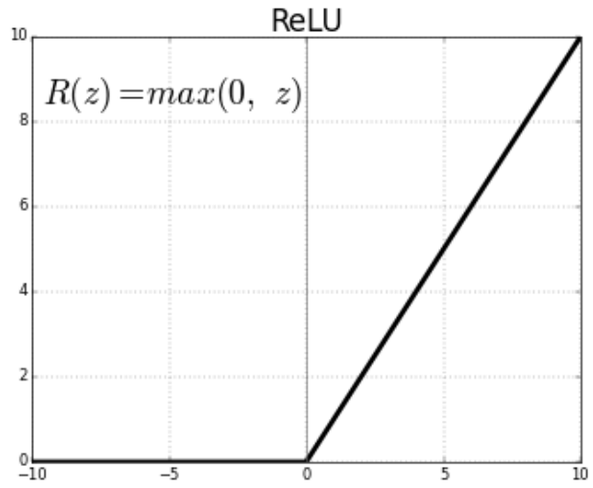


Image Credit:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

- so doesn’t this function look linear?
- not, not a straight line
- with multiple relu functions summed up we can build arbitrary functions
- mostly used activation function today …
- NN with relu converge faster
- compu complexity is less than logistic sigmoid or tanh

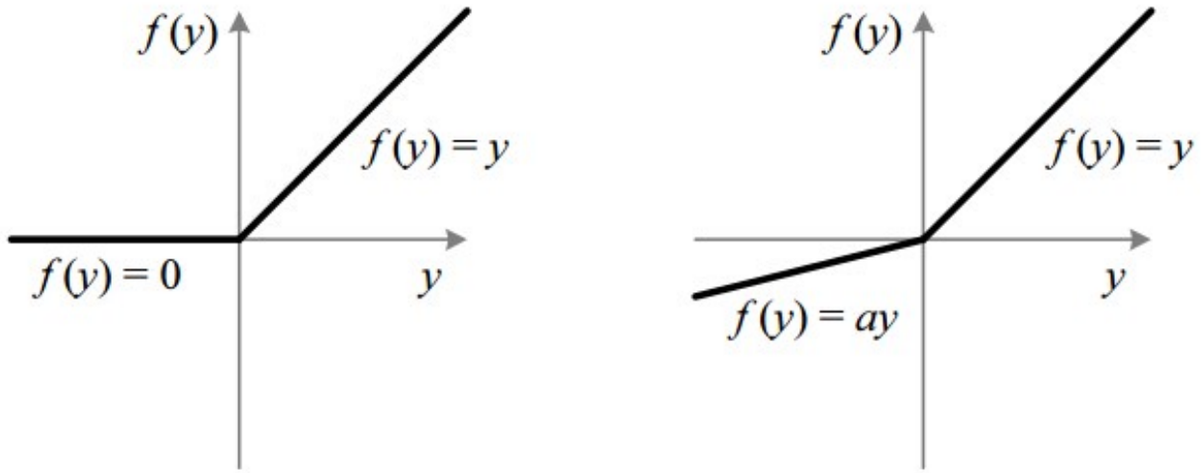


Image Credit:
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

- increases value range of function
- Relu can cause dead neurons
- if NN training not performing, just try it out

Summary

start with RELU whenever possible for input and hidden layers
output layer activation depends on problem and the cost function , but rule of thumb

- regression => linear
- classification => sigmoid / softmax

nobody stops you to try your own combinations

Bias-Variance Tradeoff

 Romeo Kienzler