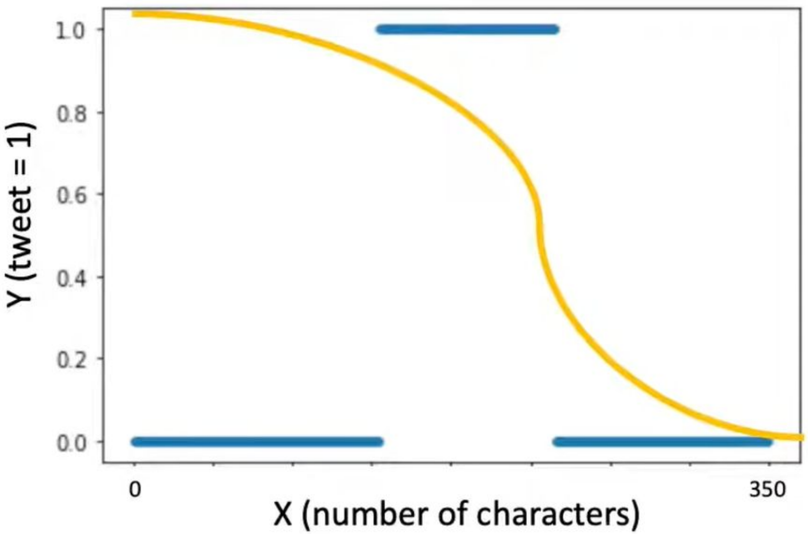
Neural Networks in NLP

Md. Mohsinul Kabir
SWE 4841

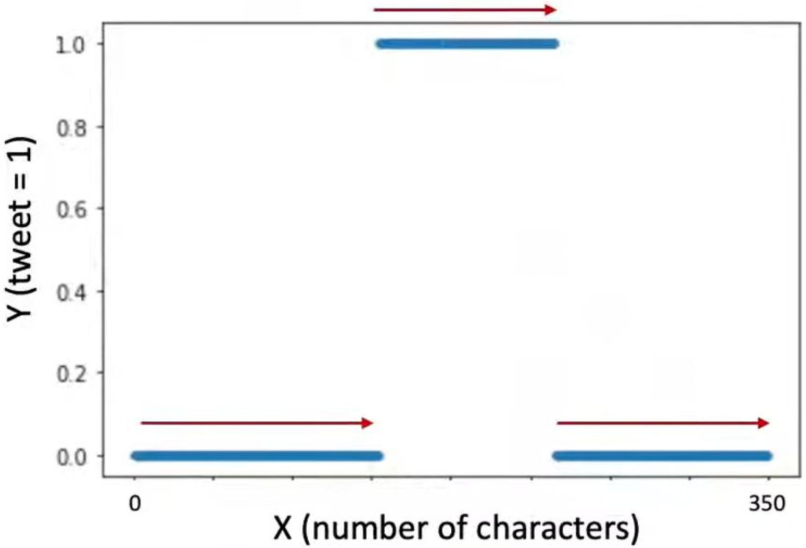
Neural Networks

- Many relationships in NLP are beyond the capabilities of a simple sigmoid or linear function to approximate.



Neural Networks

- One path we could take to solve the problem it to write out a brand new model; It will need sigmoid-like properties because there will need to be some areas where it increases, some other areas where it decreases and yet other areas where it is flat.

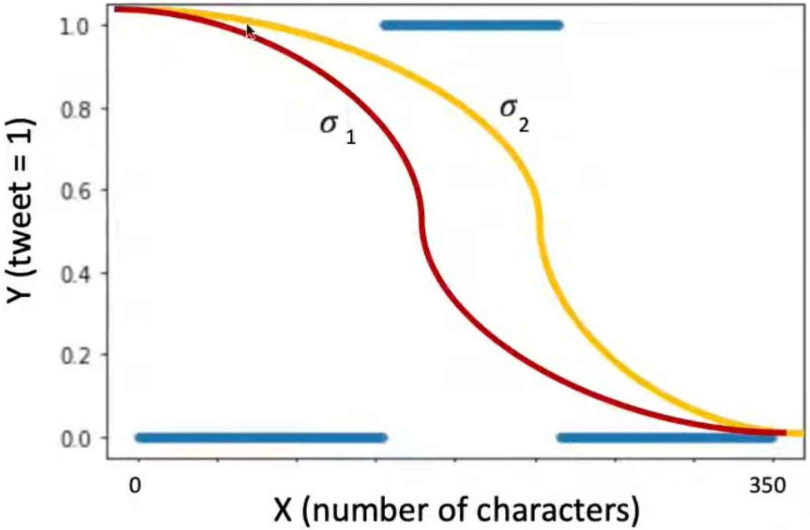


Neural Networks

- One sensible thing to try might be a linear combination of sigmoids

$$\hat{y} = w_0 + w_1\sigma_1(x) + w_2\sigma_2(x)$$

σ : a sigmoid with it's own parameters
w: weight of each sigmoid

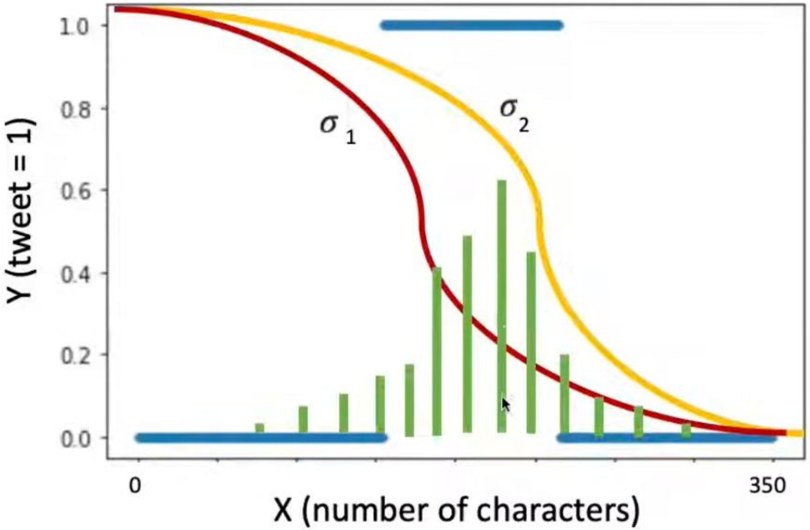


Neural Networks

- One sensible thing to try might be a linear combination of sigmoids

$$\hat{y} = w_0 + w_1\sigma_1(x) + w_2\sigma_2(x)$$

σ : a sigmoid with it's own parameters
w: weight of each sigmoid

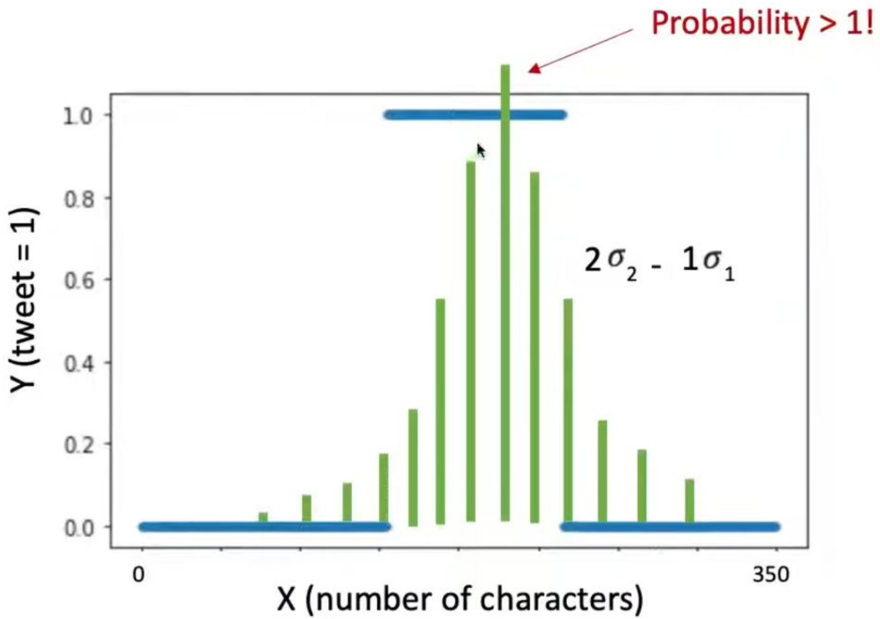


Neural Networks

- One sensible thing to try might be a linear combination of sigmoids

$$\hat{y} = w_0 + w_1\sigma_1(x) + w_2\sigma_2(x)$$

σ : a sigmoid with it's own parameters
w: weight of each sigmoid

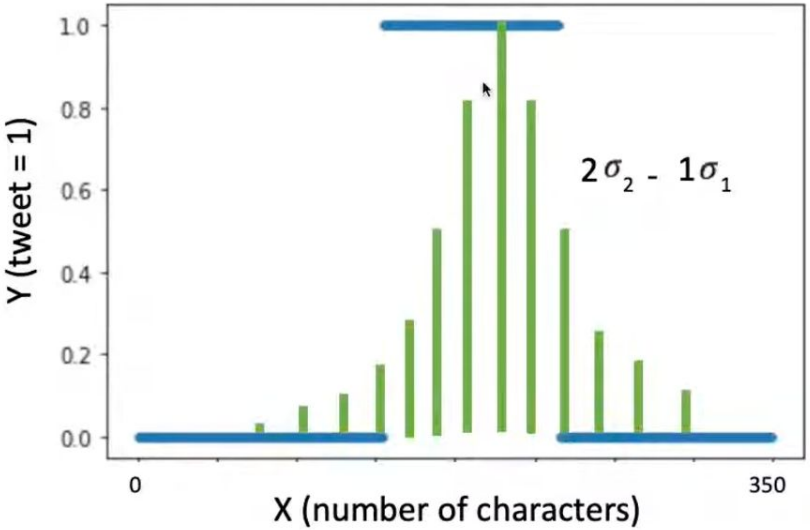


Neural Networks

- If we wrap this in another sigmoid, we can prevent the probabilities from falling out of range

$$\hat{y} = \sigma(w_0 + w_1\sigma_1(x) + w_2\sigma_2(x))$$

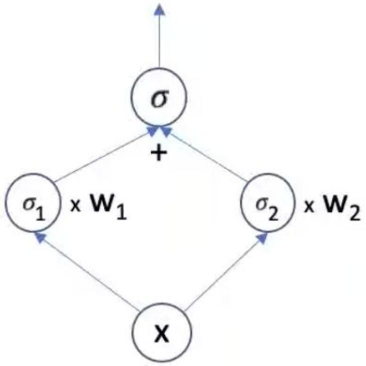
σ : a sigmoid with it's own parameters
w: weight of each sigmoid



Neural Networks

- This idea of stacking simple functions, to create more complex functions is the intuition behind **neural networks.**

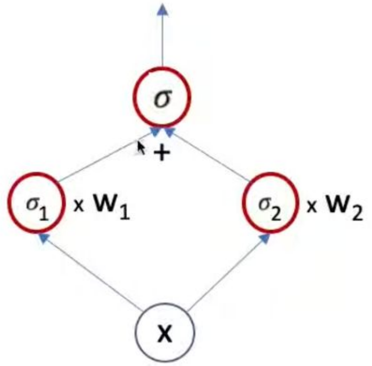
$$\hat{y} = \sigma(w_0 + w_1\sigma_1(x) + w_2\sigma_2(x))$$



Neural Networks

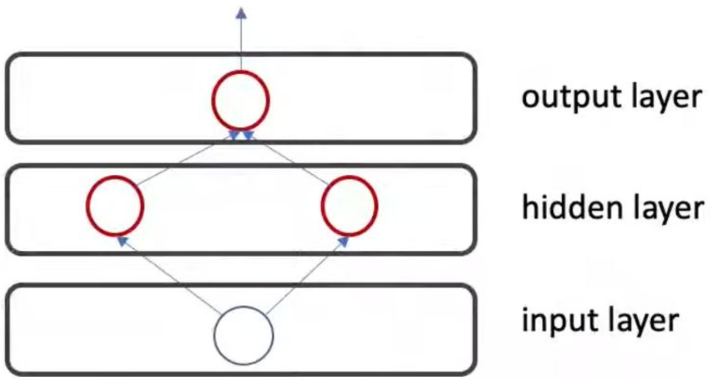
- Each instance where an activation function occurs is referred to as a **node**.

$$\hat{y} = \sigma(w_0 + w_1\sigma_1(x) + w_2\sigma_2(x))$$



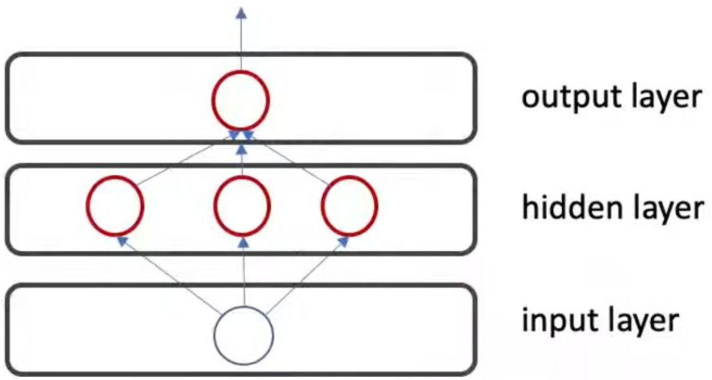
Neural Networks

- Sets of nodes that are combined together later, are referred to as **layers**.



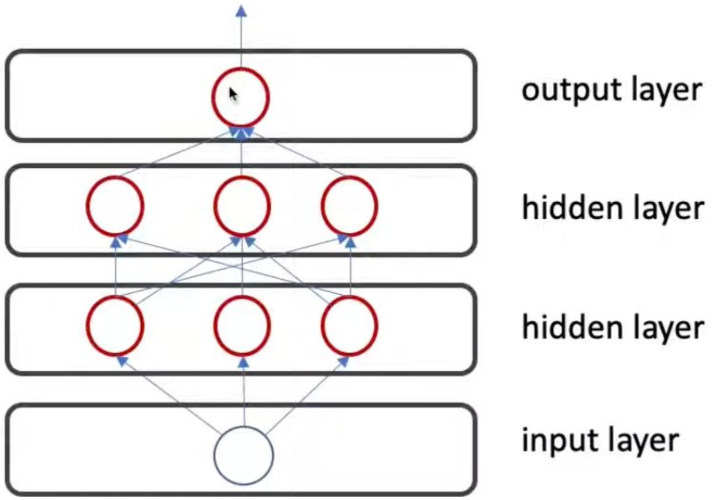
Neural Networks

- We can increase the flexibility of the model to perform approximations in two ways:
  - 1. Increasing the number of nodes



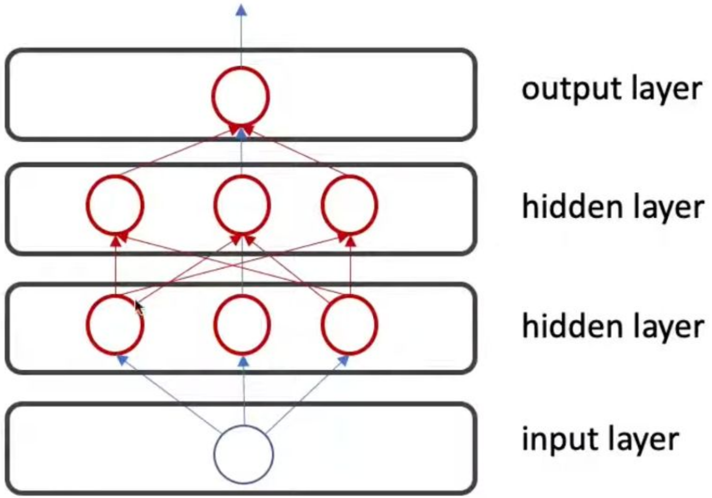
Neural Networks

- We can increase the flexibility of the model to perform approximations in two ways:
  - 1. Increasing the number of nodes
  - 2. Increasing the number of layers



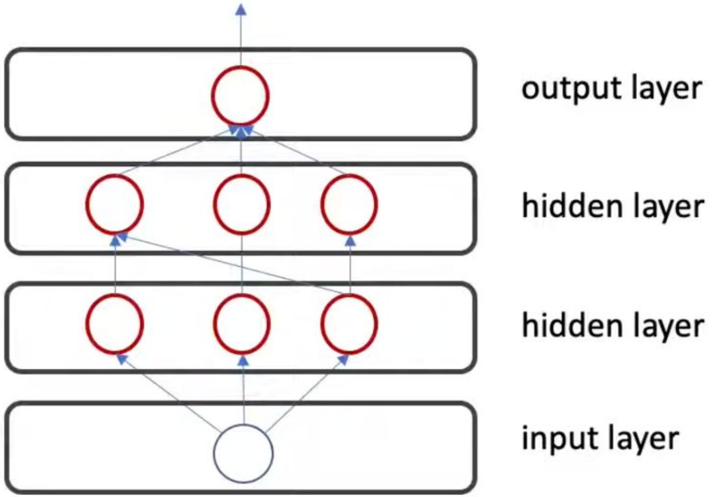
Neural Networks

- This particular configuration is referred to as a fully-connected, feed forward neural network.



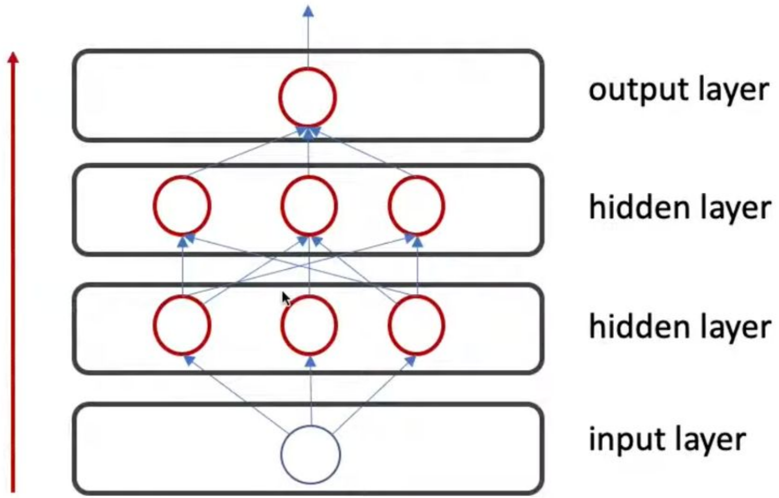
Neural Networks

- This particular configuration is referred to as a ~~fully-connected~~, feed forward neural network.



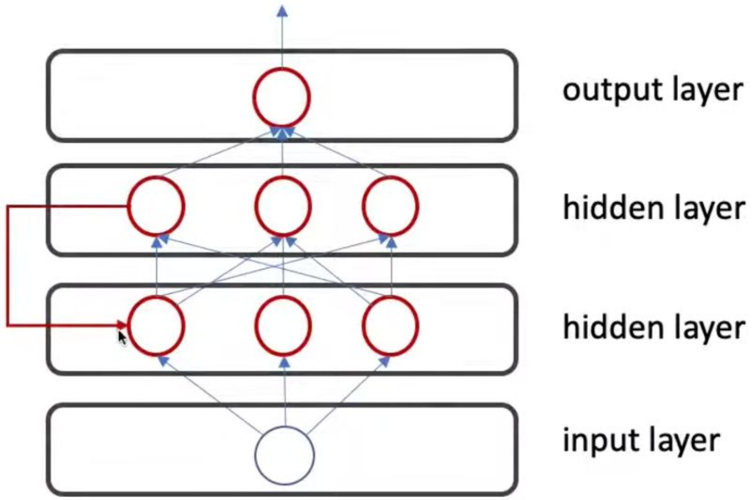
Neural Networks

- This particular configuration is referred to as a fully-connected, feed forward neural network.



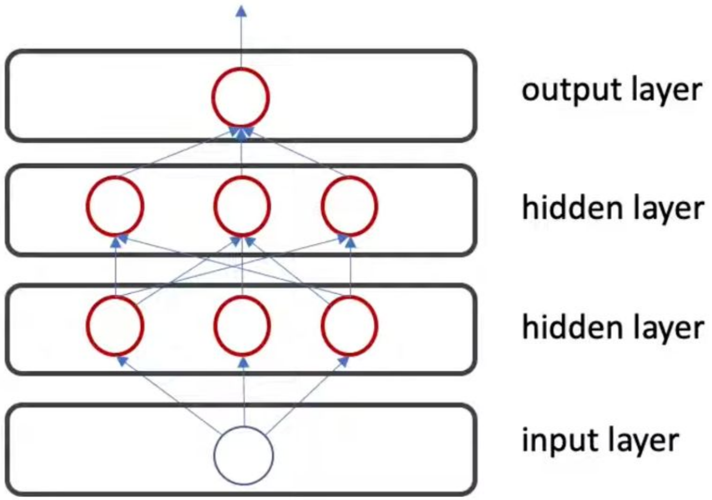
Neural Networks

- This particular configuration is referred to as a fully-connected, ~~feed forward~~ neural network.



Neural Networks

- Notice that the number of model parameters will increase very quickly as we add more nodes to the model; this is exactly why we need **gradient descent** for optimization of the loss function.



Neural Networks

- However, the gradient in the loss with respect to the parameters is no longer smooth like it was with the simple logistic regression; this means that small differences in the initial conditions of the parameters can have important consequences for what the networks learn.

Gradient Descent

f (x) = nonlinear function of x