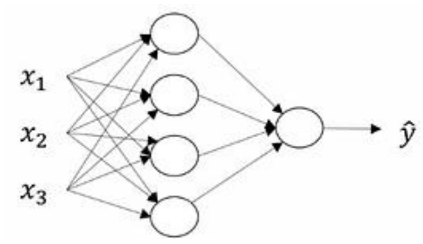
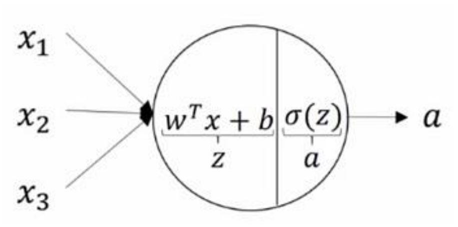
CHAPETR5

SHALLOW NEURAL NETWORKS

there is a type of neural network with a few numbers of hidden layers. Shallow neural networks consist of only 1 or 2 hidden layers. Understanding a shallow neural network gives us an insight into what exactly is going on inside a deep neural network.



The neuron is the atomic unit of a neural network. Given an input, it provides the output and passes that output as an input to the subsequent layer.



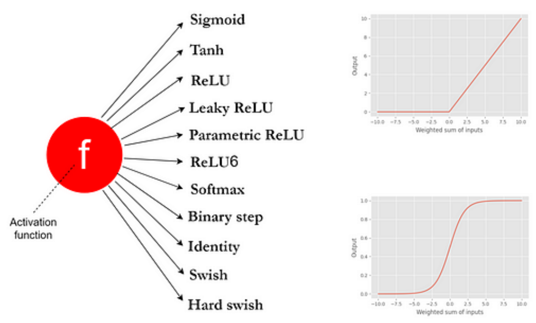
$$z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}, a_1^{[1]} = \sigma\left(z_1^{[1]}\right)$$
$$z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}, a_2^{[1]} = \sigma\left(z_2^{[1]}\right)$$
$$z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}, a_3^{[1]} = \sigma\left(z_3^{[1]}\right)$$
$$z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}, a_4^{[1]} = \sigma\left(z_4^{[1]}\right)$$

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

$$Z^{[1]} = X^{[1]T} X + b^{[1]}$$
$$A^{[1]} = \sigma\left(Z^{[1]}\right)$$

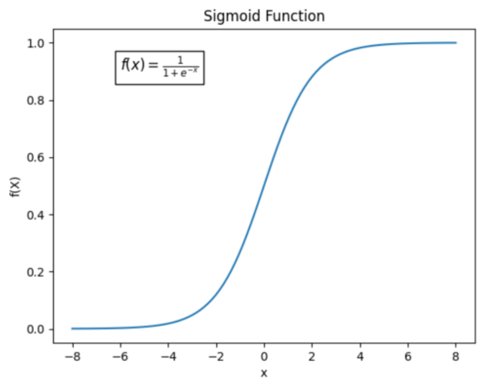
As you can see, the output will become a linear combination of a new Weight Matrix ***W***, Input ***X*** and a new Bias ***b***, which means that there remains no significance of the neurons present in the hidden layer and the weights present in the hidden layer. Therefore, to introduce non-linearity in the network, we use the activation functions.

There are many activation functions that can be used. These include ***Sigmoid***, ***Tanh***, ***ReLU***, ***Leaky ReLU*** and many others.



Sigmoid

Sigmoid functions transform data by mapping input values to a range between 0 and 1. The output of a sigmoid function can be interpreted as a value that represents the probability or intensity of an event. This makes sigmoid functions useful for representing relationships in data, as they can provide a measure of how likely or strong something is based on the input.

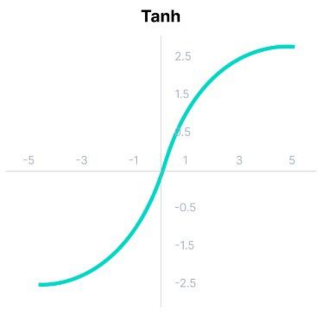


Tanh

Tanh is the **hyperbolic tangent function**, which is the hyperbolic analogue of the Tan circular function used throughout trigonometry. Tanh[α] is defined as the ratio of the corresponding hyperbolic sine and hyperbolic cosine functions via . Tanh may also be defined as , where is the base of the natural logarithm Log. has the same S-shape with the difference in output range of -1 to 1.

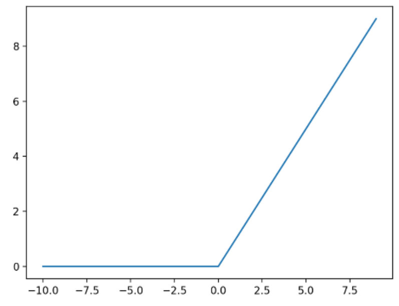
Tanh

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



ReLU

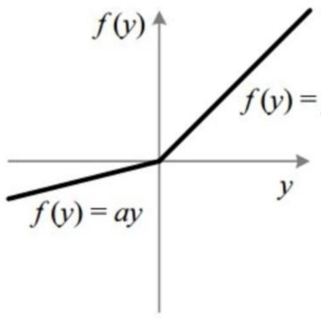
Rectified Linear Units, or **ReLUs**, are a type of activation function that are linear in the positive dimension, but zero in the negative dimension. The kink in the function is the source of the non-linearity. Linearity in the positive dimension has the attractive property that it prevents non-saturation of gradients (contrast with sigmoid activations), although for half of the real line its gradient is zero.



Leaky ReLU

The Leaky ReLU is a popular activation function that is used to address the limitations of the standard ReLU function in deep neural networks by introducing a small negative slope for negative function inputs, which helps neural networks to maintain better information flow both during its training and after.

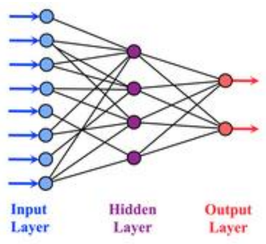
LeakyReLU(x)=max(alpha∗x,x)



Tasks

What is the structure of a shallow neural network?

Shallow neural networks are composed of **an input layer, one or two hidden layers, and an output layer**. The input layer receives the data, which is then processed through the hidden layers to produce the final output.



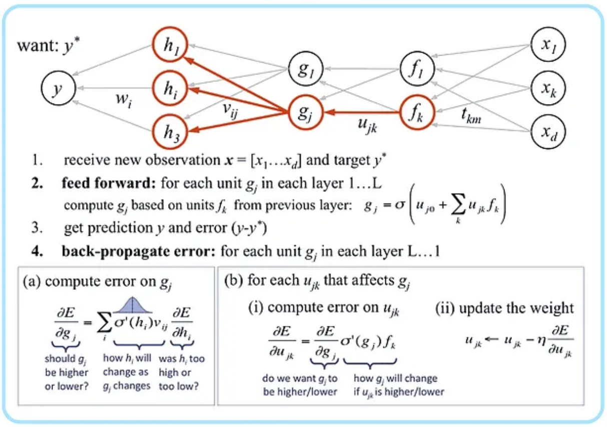
Explain the purpose and role of activation functions.

The activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it. The purpose of the activation function is **to introduce non-linearity into the output of a neuron**.

How is a shallow neural network trained using backpropagation?

By implementing backpropagation with code, we can train artificial neural networks to learn various tasks. The algorithm works by **propagating the error from the output layer of the network to the hidden layers of the network, and then adjusting the weights accordingly.**

Backpropagation



1. receive new observation $\mathbf{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

(a) compute error on g_j

$$\underbrace{\frac{\partial E}{\partial g_j}}_{\text{should } g_j \text{ be higher or lower?}} = \sum_i \underbrace{\sigma'(h_i) v_{ij}}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

(b) for each u_{jk} that affects g_j

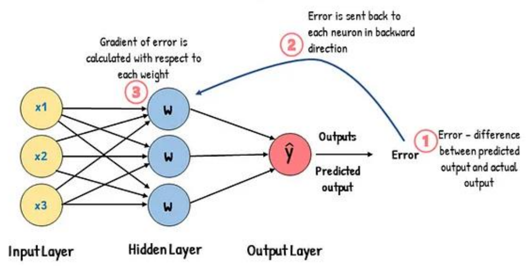
(i) compute error on u_{jk}

$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

(ii) update the weight

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

Backpropagation


Gradient of error is calculated with respect to each weight
3
Error is sent back to each neuron in backward direction
2
x1
x2
x3
W
W
W
ŷ
Outputs
Predicted output
Error
1
Error – difference between predicted output and actual output
Input Layer
Hidden Layer
Output Layer

What are some typical applications of shallow neural networks?

Shallow neural networks, typically those with one or two hidden layers, are simpler compared to deep neural networks but are still quite powerful and suitable for various applications. Some common applications of shallow neural networks include:

1. **Classification Tasks**:
   - **Spam Detection**: Classifying emails as spam or non-spam.
   - **Handwritten Digit Recognition**: Recognizing digits in handwritten samples, such as the MNIST dataset.

- **Customer Segmentation**: Grouping customers based on their behavior and preferences.

2. **Regression Tasks**:
    - **House Price Prediction**: Estimating the price of a house based on features such as size, location, and number of rooms.
    - **Stock Market Prediction**: Predicting the future prices of stocks based on historical data.
3. **Function Approximation**:
    - Approximating complex mathematical functions where the relationship between input and output is non-linear.
4. **Pattern Recognition**:
    - **Face Recognition**: Identifying or verifying a person from a digital image or a video frame.
    - **Speech Recognition**: Converting spoken language into text.
5. **Time Series Prediction**:
    - **Weather Forecasting**: Predicting future weather conditions based on historical data.
    - **Sales Forecasting**: Predicting future sales based on past sales data.
6. **Control Systems**:
    - **Robot Control**: Controlling the movement of robots or drones.
    - **Autonomous Vehicle Navigation**: Assisting in the navigation and control of autonomous vehicles.
7. **Medical Diagnosis**:
    - **Disease Classification**: Classifying medical images to detect diseases such as cancer or pneumonia.
8. **Signal Processing**:
    - **Noise Reduction**: Filtering out noise from audio signals.
    - **Image Denoising**: Removing noise from images to enhance their quality.
9. **Game Playing**:
    - Simple board games and puzzles where the strategy can be learned from data.

What are the limitations of shallow neural networks?

1. **Computational Complexity:**
    a. **Explanation:** A wide and shallow network with a vast number of input neurons can result in high computational complexity. Training and making predictions with such networks may be computationally expensive, especially as the number of parameters increases.
2. **Overfitting:**
    a. **Explanation:** Shallow networks may struggle with capturing complex patterns in the data. The model might end up fitting the training data too closely, leading to overfitting. Overfitting occurs when the model performs well on training data but poorly on unseen data.
3. **Limited Representational Power:**
    a. **Explanation:** Shallow networks might not have enough depth to learn hierarchical representations of features in the data. Deep networks, with multiple hidden layers, are often better suited for capturing intricate relationships and representations.
4. **Feature Extraction Challenges:**
    a. **Explanation:** Wide and shallow networks may face challenges in automatic feature extraction. Deep networks, by design, learn hierarchical features from data, allowing them to automatically extract and represent meaningful features.

Code a Shallow NN from scratch.

Weights between input and hidden layer:

-12.26290141   5.46686902   -1.03353223   5.46338203 -12.27438694   -1.03396692

Weights between hidden and output layer:

12.91121054   12.91395278   -33.48404954

Write the math of it.

1. Input layer: The input data is represented as a vector `$x = [x_1, x_2, ..., x_n]$

2. Hidden layer: $h = [h_1, h_2, ..., h_m]$

   $h_j = \sigma(\sum_i w_{ji}^{(1)} x_i + b_j^{(1)})$

3. Output layer: $y = [y_1, y_2, ..., y_k]$

$y_l = \sigma(\sum_j w_{lj}^{(2)} h_j + b_l^{(2)})$

1. Output layer error gradients: $\delta_l^{(2)} = (y_l - t_l)\, \sigma'(\sum_j w_{lj}^{(2)} h_j + b_l^{(2)})$

2. Hidden layer error gradients: $\delta_j^{(1)} = \sigma'(\sum_i w_{ji}^{(1)} x_i + b_j^{(1)}) \sum_l \delta_l^{(2)} w_{lj}^{(2)}$

3. Weight updates:

   The weights are updated using gradient descent:

   $w_{lj}^{(2)} \leftarrow w_{lj}^{(2)} - \eta\, \delta_l^{(2)} h_j$

   $w_{ji}^{(1)} \leftarrow w_{ji}^{(1)} - \eta\, \delta_j^{(1)} x_i$

4. Bias updates:

   The biases are updated similarly:

   $b_l^{(2)} \leftarrow b_l^{(2)} - \eta\, \delta_l^{(2)}$

   $b_j^{(1)} \leftarrow b_j^{(1)} - \eta\, \delta_j^{(1)}$