Basic_of_ANN_(Artificial_Neural_Network)

April 10, 2025

1 Basic Understanding of ANN (Artificial Neural Network)

The solution to fitting more complex (*i.e.* non-linear) models with neural networks is to use a more complex network that consists of more than just a single perceptron. The take-home message from the perceptron is that all of the learning happens by adapting the synapse weights until prediction is satisfactory. Hence, a reasonable guess at how to make a perceptron more complex is to simply **add more weights**.

There are two ways to add complexity:

1. Add backward connections, so that output neurons feed back to input nodes, resulting in a **recurrent network**
2. Add neurons between the input nodes and the outputs, creating an additional ("hidden") layer to the network, resulting in a **multi-layer perceptron**

How to train a multilayer network is not intuitive. Propagating the inputs forward over two layers is straightforward, since the outputs from the hidden layer can be used as inputs for the output layer. However, the process for updating the weights based on the prediction error is less clear, since it is difficult to know whether to change the weights on the input layer or on the hidden layer in order to improve the prediction.

Updating a multi-layer perceptron (MLP) is a matter of:

1. moving forward through the network, calculating outputs given inputs and current weight estimates
2. moving backward updating weights according to the resulting error from forward propagation(using backpropagation method).

In this sense, it is similar to a single-layer perceptron, except it has to be done twice, once for each layer.

1.1 Activation Function of ANN:

In ANN we use sigmoid as an activation function in each layer instead of step function.Because ANN can solve non-linear problem so the output can be varied. Sigmoid outputs numbers 0 to 1. On the other hand step function outputs just 0 or 1.

- Formula of Sigmoid:
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$
- It can take number between $-\infty, +\infty$ and gives output 0 to 1.

1.1.1 Need of Activation Function:

- An activation function added into an artificial neural network in order to help the network learn complex patterns in the data.
- It also scales the data
- It filters out the important portion of data
- Without activation function Deep stack of network will behave like a single linear transformation.
  - **Example:** without activation function

$z_1 = x_1.w_1$,$z_2 = w_2.z_1$

$$z_2 = w_2.z_1$$

$$z_2 = w_2.x_1w_1$$

$$z_2 = Wx_1$$

  - So, you can see it has been a single neuron.Behave like a single linear transformation.
- Without activation function all the continuous function cannot be approximated.

1.2 Weight update of ANN (Backpropagation intiution):

- In the year 1986 a groundbreaking paper “Learning Internal Representation by Error Propagation” was published by -
  - David Rumelhart,
  - Geoffrey Hinton, &
  - Ronald Williams
- It depicted an efficient way to update weights and biases of the network based on the error/loss function by passing twice through the network i.e forward and backward pass.
  - forward pass: data is passed through the input layer to the hidden layer and it calculates ouput. Its nothing but making prediction.
  - error calculation: Based on loss function error is calculated to check how much deviation is there from the ground truth or actual value and predicted value.
  - error contribution from the each connection of the output layer is calculated.
  - Then algo goes a layer deep and calculates how much previous layer contributed into the error of present layer and this way it propagates till the input layer.
  - This reverse pass measures the error gradient accross all the connection.
  - At last by using these error gradients a gradient step is performed to update the weights.
- In MLP key changes were to introduce a sigmoid activation function

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

2 Now lets get some intuition of ANN.

At the first layer is Input or Buffer layer. Second layer called hidden layer & the 3rd layer called output layer.

In the classification outputs neuron can be multiple but in the case of Regression output neuron might be one.

2.1 Simple Example:

Lets take a simple neuron network , Here consider bias = 0

- So,

$$z_1 = (w_{11}.i_1) + (w_{12}.i_2)$$
$$\therefore \hat{y_1} = act(z_1)$$
$$z_2 = (w_{21}.i_1) + (w_{22}.i_2)$$
$$\therefore \hat{y_2} = act(z_2)$$

Let's define it as a metrix form,

$$\begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \end{bmatrix} * \begin{bmatrix} i_1 \\ i_2 \end{bmatrix} = \begin{bmatrix} (w_{11}.i_1) + (w_{12}.i_2) \\ (w_{21}.i_1) + (w_{22}.i_2) \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

$$>> \begin{bmatrix} act(z_1) \\ act(z_2) \end{bmatrix} = \begin{bmatrix} \hat{y_1} \\ \hat{y_2} \end{bmatrix}$$

$ W*X=Z $

$act(Z) = \hat{y}$

2.2 Error Calculation:

Now weight will be updated based on the proportional error!

In order to do that weight update rule (Backpropagration) can be used, that will be discuss further.

- weight update rule:

$$w = w - \eta \frac{\partial e}{\partial w}$$

3 Difference between Perceptron & Neural Network:

3.0.1 Perceptron:

- Perceptron is a single layer neural network, It can be also multi layer.
- Perceptron is a linear classifier (binary).
- It cann't solve non-linear problem.
- Perceptron use step function as an activation function.

3.0.2 Neural Network:

- A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.
- A neuron network is consisted by single layer or multiple layer
- It can be very depth
- It can solve non-linear problems.
- It can have many hidden layers.
- It use sigmoid, ReLu, softmax etc. as an activation function.

```
[ ]:
```