**DEEP LEARNING WORKSHEET-3 SOLUTION**

1. B
2. C
3. A
4. D
5. C
6. B
7. B
8. A
9. B and C
10. B and C
11. Activation functions are really important for a Artificial Neural Network to learn and make sense of something really complicated and Non-linear complex functional mappings between the inputs and response variable.They introduce non-linear properties. Their main purpose is to convert a input signal of a node in a A-NN to an output signal.

If we do not apply a Activation function then the output signal would simply be a simple linear function.A linear function is just a polynomial of one degree. a linear equation is easy to solve but they are limited in their complexity and have less power to learn complex functional mappings from data. A Neural Network without Activation function would simply be a Linear regression Model, which has limited power and does not perform good most of the times.

1. **Forward propagation** is where you would give a certain input to your neural network, say an image or text. The network will calculate the output by *propagating* the input signal through its layers. In other words, the output form one layer becomes the input to the next one, where the output from the last one is the “answer”.

In order to do that accurately, the network needs to be trained. This is done through **backpropagation**. Basically, during training all the parameters of the network’s layers need to be updated (“optimized”). For this reason, the network needs to know in what “direction” the update should go thus it needs to calculate the so-called gradient with respect to a function known as the *loss function,* which is a way of saying how “wrong” or “incorrect” the network still is, so hopefully it becomes better next time. Since finding the gradients of all the parameters with respect to this loss function is a complex equation involving the so-called “chain rule”, in computation it is being *propagated,* where every layer adds its own contribution to that gradient, starting from the last one (hence the name).

1. **Stochastic Gradient Descent**

Stochastic gradient descent (SGD) checks the error for each training example within the dataset, meaning it updates the parameters for each training example **one by one**. Depending on the problem, this can make SGD faster than batch gradient descent. One advantage is the **frequent updates** allow us to have a pretty detailed rate of improvement.

The frequent updates, however, are more computationally expensive than the batch gradient descent approach. Additionally, the frequency of those updates can result in noisy gradients, which may cause the error rate to jump around instead of slowly and smooth decreasing. Basically a Zig-Zag is observed. **SGD with momentum** is basically used to reduce this zig zag by using moving Average.

**Batch Gradient Descent**

Batch gradient descent, also called vanilla gradient descent, calculates the error for each example within the training dataset, but only after all training examples have been evaluated does the model get updated. This whole process is like a cycle and it's called a training epoch. It can be stuck in Local Minima very easily. It also requires the entire training dataset be in memory and available to the algorithm.

**Mini-batch gradient descent**

Mini-batch gradient descent is the go-to method since it’s a combination of the concepts of SGD and batch gradient descent. It simply splits the training dataset into small batches and performs an update for each of those batches. This creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. Common mini-batch sizes range between 50 and 256, but like any other machine learning technique, there is no clear rule because it varies for different applications. This is the go-to algorithm when training a neural network and it is the most common type of gradient descent within deep learning.

1. **Benefits of Mini Batch Gradient descent**

This is a mixture of both stochastic and batch gradient descent. The training set is divided into multiple groups called batches. Each batch has a number of training samples in it. At a time a single batch is passed through the network which computes the loss of every sample in the batch and uses their average to update the parameters of the neural network. For example, say the training set has 100 training examples which is divided into 5 batches with each batch containing 20 training examples. This means that the equation in figure2 will be iterated over 5 times (number of batches).This ensures the following advantages of both stochastic and batch gradient descent are used due to which Mini Batch Gradient Descent

1. Transfer learning is the reuse of a pre-trained model on a new problem. It's currently very popular in deep learning because it can train deep neuralnetworks with comparatively little data.