**DEEP LEARNING – WORKSHEET 3 ANSWERSHEET**

**Q1 to Q8 are MCQs with only one correct answer. Choose the correct option.**

1. B) As number of hidden layers increase, model capacity increases
2. C) It normalizes (changes) all the input before sending it to the next layer
3. A) Network will not converge
4. D) All of these
5. B) (-3, 4, 4)
6. B) Simulate the network on a test dataset after every epoch of training. Stop training when the generalization error starts to increase
7. B) Stochastic Gradient Descent
8. A) Freeze all the layers except the last, re-train the last layer

**Q9 and Q10 are MCQs with one or more correct answers. Choose all the correct options.**

1. B) Training is too slow

C) Restrict activations to become too high or low

1. B) sigmoid

**Q11 to Q15 are subjective answer type question. Answer them briefly.**

1. If we do not apply an 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.
2. **Forward propagation** (or forward pass) refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the input layer to the output layer.

**Backpropagation** refers to the method of calculating the gradient of neural network parameters. In short, the method traverses the network in reverse order, from the output to the input layer, according to the chain rule from calculus.

1. Gradient Descent is an optimization algorithm used for minimizing the cost function in various machine learning algorithms. It is basically used for updating the parameters of the learning model.

**Types of gradient Descent:**

**Batch Gradient Descent:**This is a type of gradient descent which processes all the training examples for each iteration of gradient descent. But if the number of training examples is large, then batch gradient descent is computationally very expensive. Hence if the number of training examples is large, then batch gradient descent is not preferred. Instead, we prefer to use stochastic gradient descent or mini-batch gradient descent.

**Stochastic Gradient Descent:** This is a type of gradient descent which processes 1 training example per iteration. Hence, the parameters are being updated even after one iteration in which only a single example has been processed. Hence this is quite faster than batch gradient descent. But again, when the number of training examples is large, even then it processes only one example which can be additional overhead for the system as the number of iterations will be quite large.

**Mini Batch gradient descent:** This is a type of gradient descent which works faster than both batch gradient descent and stochastic gradient descent. Here *b* examples where*b<m* are processed per iteration. So even if the number of training examples is large, it is processed in batches of b training examples in one go. Thus, it works for larger training examples and that too with lesser number of iterations.

**Benefits:**

* High throughput: With mini-batch one can process a large number of input examples per second. The mini batching style of gradient descent is perhaps the only way to use the large number of cores at once in a GPU.
* (Sometimes) faster convergence: The high throughput may also translate to faster convergence depending on the variance in the dataset and the learning rate used.
* High quality gradient: Mini batching allows for a high-quality gradient and this will be really useful allowing one to use high learning rates.

**Cons:**

* Low final accuracy: Many times, mini batching may result in low final accuracy since the noise in the gradient is really helpful towards the end to extract that last 0.5%.

1. **Transfer learning** is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For example, knowledge gained while learning to recognize cars could apply when trying to recognize trucks.