**Deep learning worksheet 2**

**Solution**

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

Q1. Operations in the neural networks can performed \_\_\_\_\_\_\_?

A) serially B) parallely C) serially or parallely D) None of the above

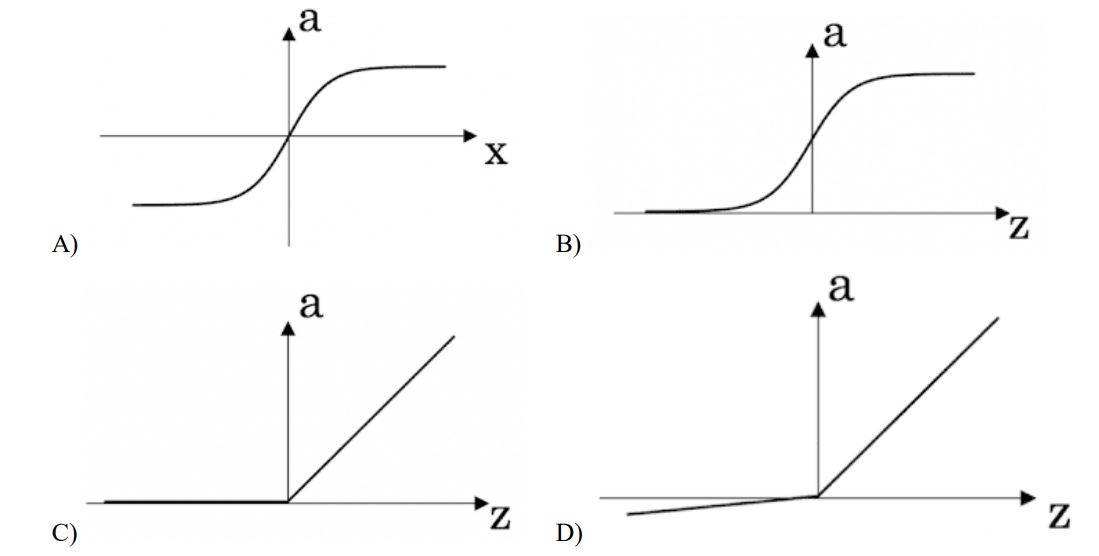
Ans. C

Q2. Who proposed the first perceptron model and when?

A) Rosenblatt, 1958 B) McCulloch-pitts, 1958 C) John Hopfield, 1982 D) McCulloch-pitts, 1982

Ans. A

Q3. Which one of these plots represents a ReLU activation function?



Ans. C

Q4. In a simple artificial neural network with 5 neurons in the input layer, 8 neurons in the hidden layer and 3 neurons in the output layer. What is the size of the weight matrices between hidden-output layers and inputhidden layers? A) [3×8], [5×8] B) [8×3], [5×8] C) [5×8], [8×5] D) [8×3], [5×3]

Q5. What is a dead unit in a neural network? A) A unit which does not respond completely to any of the training patterns B) The unit which produces the biggest sum-squared error C) A unit which doesn’t update during training by any of its neighbour D) None of these

Ans.

Q6. Which of the following functions can be used as an activation function if we wish to predict the probabilities of n classes such that sum of all n probabilities is equal to 1?

A) sigmoid B) softmax C) tanh D) ReLU

Ans. A

Q7. The amount of output of one unit received by another unit depends on what?

A) output unit B) input unit C) activation values D) weights

Ans. D

Q8. What is asynchronous update in neural networks?

A) output units are updated parallely B) output units are updated sequentially C) either sequentially or parallely D) None of the above

Ans. A

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

Q9. Which of the following techniques can be used to reduce overfitting in a neural network?

A) EarlyStopping B) Dropout C) checkpoints D) ReduceLROnPlateau

Ans. A, B

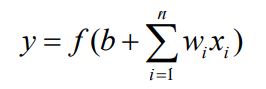
Q10. Why is an RNN used for machine translation, say translating English to Hindi?

A) It can be trained as a supervised learning problem. B) It is strictly more powerful than a Convolutional Neural Network C) It is applicable when the input/output is a sequence (e.g., a sequence of words) D) RNNs represent the recurrent process of Idea->Code->Experiment->Idea->

Ans. A, C

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

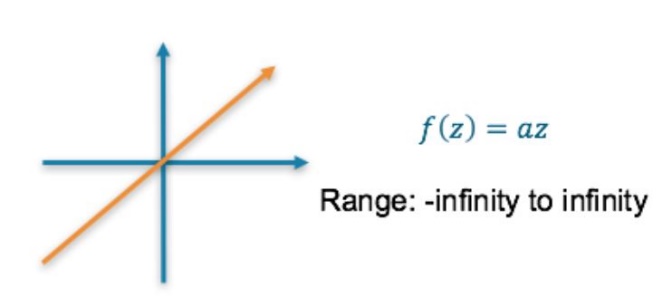
Q11. The output of a perceptron is calculated as follows:



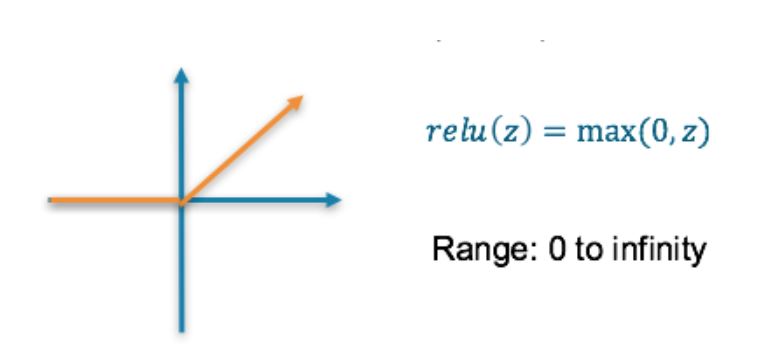
Where f x( ) is the activation function. If you want to build a perceptron which gives an output for linear regression, what will be the activation function you would use?

Ans.

## Linear Activation Function: This results in a numerical value which we require



**ReLU Activation Function** — This results in a numerical value greater than 0 (Zero)



## Loss Function used for linear problem is - Mean squared error (MSE) — This finds the average squared difference between the predicted value and the true value

**Q12. What will happen if we use very large or very small learning rates?**

Ans.

**Learning Rate :** The amount that the weights are updated during training is referred to as the step size or the “*learning rate*.”

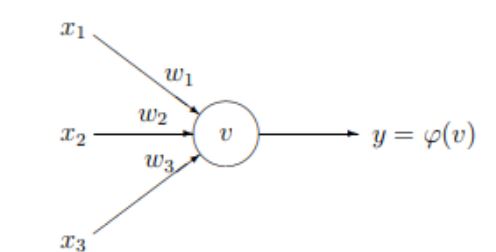
Specifically, the learning rate is a configurable hyperparameter used in the training of neural networks that has a small positive value, often in the range between 0.0 and 1.0.

The learning rate controls how quickly the model is adapted to the problem. Smaller learning rates require more [training epochs](https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/) given the smaller changes made to the weights each update, whereas larger learning rates result in rapid changes and require fewer training epochs.

**If we use very large or very small LR -**

A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.  
The challenge of training deep learning neural networks involves carefully selecting the learning rate. It may be the most important hyperparameter for the model.

**Q13. Below is a diagram if a single artificial neuron:**



The node has three inputs x = (x1, x2, x3) that receive only binary signals (either 0 or 1). How many different input patterns this node can receive? What if the node had four, five inputs? Can you give a formula that computes the number of binary input patterns for a given number of inputs?

Ans.

The formula for the number of binary input patterns is: 2n , where n in the number of inputs.

So for three, four and five inputs -

23 = 8 , 24 =16 and 25 =32

**Q14. What Are Vanishing and Exploding Gradients?**

Ans.

**Vanishing Gradients:**

A problem with training networks with many layers (e.g. deep neural networks) is that the gradient diminishes dramatically as it is propagated backward through the network. The error may be so small by the time it reaches layers close to the input of the model that it may have very little effect. As such, this problem is referred to as the “*vanishing gradients*” problem. Vanishing gradients make it difficult to know which direction the parameters should move to improve the cost function. The term vanishing gradient refers to the fact that in a feedforward network (FFN) the backpropagated error signal typically decreases (or increases) exponentially as a function of the distance from the final layer.

There are many techniques that can be used to reduce the impact of the vanishing gradients problem for feed-forward neural networks, most notably alternate weight initialization schemes and use of alternate activation functions.

**Exploding Gradients:**

In deep multilayer Perceptron networks, exploding gradients can result in an unstable network that at best cannot learn from the training data and at worst results in NaN weight values that can no longer be updated.

There are some subtle signs that you may be suffering from exploding gradients during the training of your network, such as:

* The model weights quickly become very large during training.
* The model weights go to NaN values during training.
* The error gradient values are consistently above 1.0 for each node and layer during training.

To Fix Exploding Gradients we can use some techniques like -

### Re-Design the Network Model

### Use Long Short-Term Memory Networks

### Use Gradient Clipping

### Use Weight Regularization

**Q15. What Is the Difference between Epoch, Batch, and Iteration in Deep Learning?**

Ans.

We need terminologies like epochs, batch size, iterations only when the data is too big which happens all the time in machine learning and we can’t pass all the data to the computer at once. So, to overcome this problem we need to divide the data into smaller sizes and give it to our computer one by one and update the weights of the neural networks at the end of every step to fit it to the data given -   
  
**Epochs:**

One Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE. Since one epoch is too big to feed to the computer at once we divide it in several smaller batches. One epoch leads to underfitting. As the number of epochs increases, more number of times the weight are changed in the neural network and the curve goes from **underfitting** to **optimal** to **overfitting** curve.  
  
**Batch Size:**   
Total number of training examples present in a single batch. you can’t pass the entire dataset into the neural net at once. So, you **divide dataset into Number of Batches or sets or parts.** Just like you divide a big article into multiple sets/batches/parts like Introduction, Gradient descent, Epoch, Batch size and Iterations which makes it easy to read the entire article for the reader and understand it.

**Mini-batch size**:   
Is the number of examples the learning algorithm processes in a single pass (forward and backward). A **Mini-batch** is a small part of the dataset of given **mini-batch size**.

# 

# **Iterations:** To get the iterations you just need to know multiplication tables or have a calculator. Iterations is the number of batches needed to complete one epoch. The number of batches is equal to number of iterations for one epoch. We can divide the dataset of 2000 examples into batches of 500 then it will take 4 iterations to complete 1 epoch. Where Batch Size is 500 and Iterations is 4, for 1 complete epoch.

