1.C) serially or parallel

2.A) Rosenblatt, 1958

3.C)

4.D) [8×3], [5×3]

5.A) A unit which does not respond completely to any of the training patterns

6.B) softmax

7.D) weights

8.A) output units are updated parallel

9.A) EarlyStopping & B) Dropout

10.A) It can be trained as a supervised learning problem

Ans 11.In that case we will use **Linear activation function**, Linear Activation Function, a linear activation function takes the form:A=cx However, a linear activation function has two major problems:

1. Not possible to use backpropagation (gradient descent) to train the model—the derivative of the function is a constant, and has no relation to the input, X. So it’s not possible to go back and understand which weights in the input neurons can provide a better prediction.

2. All layers of the neural network collapse into one—with linear activation functions, no matter how many layers in the neural network, the last layer will be a linear function of the first layer (because a linear combination of linear functions is still a linear function). So a linear activation function turns the neural network into just one layer.

A neural network with a linear activation function is simply a linear regression model. It has limited power and ability to handle complexity varying parameters of input data

Ans12. The learning rate hyperparameter controls the rate or speed at which the model learns. Specifically, it controls the amount of apportioned error that the weights of the model are updated with each time they are updated, such as at the end of each batch of training examples.

Given a perfectly configured learning rate, the model will learn to best approximate the function given available resources (the number of layers and the number of nodes per layer) in a given number of training epochs (passes through the training data).

Generally, a large learning rate allows the model to learn faster, at the cost of arriving on a sub-optimal final set of weights. A smaller learning rate may allow the model to learn a more optimal or even globally optimal set of weights but may take significantly longer to train.

At extremes, **a learning rate that is too large will result in weight updates that will be too large and the performance of the model** (such as its loss on the training dataset) will oscillate over training epochs. Oscillating performance is said to be caused by weights that diverge (are divergent). **A learning rate that is too small may never converge or may get stuck on a suboptimal solution**. Therefore, we should not use a learning rate that is too large or too small. we must configure the model in such a way that on average a “good enough” set of weights is found to approximate the mapping problem as represented by the respective training dataset.

Ans13. For three inputs the number of combinations of 0 and 1 is 8, and for four inputs the number of combinations is 16, for five inputs the number of combinations will be 32. So from the above it can be said that 8 = 2^3 , 16 = 2 ^4 and 32 = 2 ^5 (for three, four and five inputs). Thus**, the formula for the number of binary input patterns is: 2 n** , where n in the number of inputs,and it is denoted by 2n because it’s a binary classification(so 2 outputs in this case it’s 0 and 1 ).

Ans.14 **The vanishing gradient** are problems or an an issue that sometimes arises when training machine learning algorithms through gradient descent. This most often occurs in neural networks that have several neuronal layers such as in a deep learning system, but also occurs in recurrent neural networks.   The key point is that the calculated partial derivatives used to compute the gradient as one goes deeper into the network.  Since the gradients control how much the network learns during training, **if the gradients are very small or zero, then little to no training can take place, leading to poor predictive performance**.

**Exploding gradients** are a problem where large error gradients accumulate and result in very large updates to neural network model weights during training.

This has the effect of your model being unstable and unable to learn from your training data. An error gradient is the direction and magnitude calculated during the training of a neural network that is used to update the network weights in the right direction and by the right amount.

In deep networks or recurrent neural networks, error gradients can accumulate during an update and result in very large gradients. These in turn result in large updates to the network weights, and in turn, an unstable network. At an extreme, the values of weights can become so large as to overflow and result in NaN values.

**The explosion occurs through exponential growth by repeatedly multiplying gradients through the network layers that have values larger than 1.0**.

Ans15. **Epoch** – Represents one iteration over the entire dataset (everything put into the training model).

**Batch** – Refers to when we cannot pass the entire dataset into the neural network at once, so we divide the dataset into several batches.

**Iteration** – if we have 10,000 images as data and a batch size of 200. then an epoch should run 50 iterations (10,000 divided by 50).