DeepRL HW0:-

1. y = b0 + b1*x

   Where, b0 = 0 and b1 = 0.5

2. K-means is a clustering algorithm that groups data points into clusters based on their similairity to each other. The number of clusters K has to be specified. At each iteration, the centroid of the cluster is determined and we assign the datapoints to the closest cluster based on the distance of the point from the centroid of the clusters. The iteration is stopped when the assignments don't change in consecutive iterations.

K-means requires the number of clusters to be provided, whereas meanshift does not.

Meanshift requires O(kN^2) operations where N is the no of data points and k is the average number of iterations for each data point, whereas k-means requires only O(kN) operations.
Meanshift often does not perform well when their are outliers in the data, whereas k-means does not have this problem.

3. Softmax is a function that turns a vector of values into new values that sum to 1. Thus, these new values can be interpreted as probabilities.

$$\sigma(\mathbf{z})_i = \frac{e^{\beta z_i}}{\sum_{j=1}^{K} e^{\beta z_j}} \text{ or } \sigma(\mathbf{z})_i = \frac{e^{-\beta z_i}}{\sum_{j=1}^{K} e^{-\beta z_j}} \text{ for } i = 1, \ldots, K$$

Softmax instability :-

- When very large numbers are approximated as infinity (overflow). Results in the values being either 1 or 0
- When very small numbers are approximated as zero (underflow). Results in the values being almost the same.

4.

Regularization helps prevent overfitting by encouraging the model to generalize.

Dropout is one common technique that is used. Dropout layers will probabilistically drop nodes in the layer. As a result, no node will be assigned high parameters, and the output won't depend on a single node.

5.

Generative Adversarial Networks (GANs) consist of two sub-models : the generator model which is trained to generate new examples and a discriminator model which is trained to classify examples as either real or fake. Both the models are trained until the discriminator model is wrong about half the time. Thus, the generative model learns to generate plausible examples.

The generator might learn to produce only a certain small set of outputs that are especially plausible. The discriminator might get stuck in a local minimum and hence, might not be able to find the best strategy. As a result, the generator rotates through a small set of outputs. This is called a mode collapse.

6.

Steps involved in PCA:-

- Compute the mean for every one of the d features/dimensions in the dataset.
- Compute the covarience matrix for the dataset.
- Compute the eigenvectors and the corresponding eigen values.
- Select the k eigenvectors with largest eigenvalues to form a dxk dimensional matrix.
- Multiply the transponse of this dxk matrix with the transpose of the original data to get the final dataset.

7.

Xavier Initialization Method is a common initialization method.
In this method, the weights are drawn from the uniform distribution

W ~ [- sqrt(6)/ sqrt(N(j) + N(j+1)) , sqrt(6)/ sqrt(N(j) + N(j+1))], where N(j) = no of inputs to layer j.

Another method is Kaiming Initialization.

In this method, the mean of the activations is equal to 0 and the variance of the activation is kept constant across every layer. Therefore, the weights of layer L are picked from the distribution with mean = 0 and variance = sqrt(1/N(j)) . The biases are initialized to 0.

8.

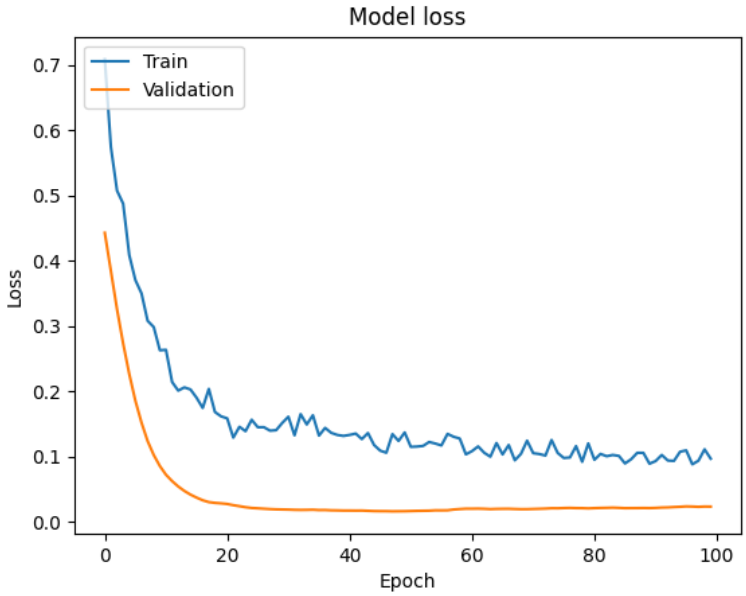
Methods used to prevent overfitting:

- Split data into train, validation and test set. Used the validation set to tune hyperparameters such as number of layers, number of dimensions in layers, etc. Final test on test set after tuning.
- Used dropout layers in between hidden layers, to probabilistically drop out nodes in the network and reduce overfitting and promote genralization.
- Used early stopping by training the model for a low number of epochs.

Final losses and accuracy:

```
Epoch 100/100
9/9 [==============================] - 0s 2ms/step - loss: 0.0974 - accuracy: 0.9689 - val_loss: 0.0239 - val_accuracy: 0.9853
Accuracy on validation set: 98.52941176470588
Accuracy on test set: 97.10144927536231
```

Training curve:



Visualization of errors on test set: