Problem 1
[1pt] What is K-Means clustering? How is it different from Meanshift?

Ans: K-Means Clustering is an unsupervised algorithm which is used used to group data points into K groups or clusters. The idea behind the algorithm is as follows:-

1. Choose K distinct points as initial centroids.
2. Calculate euclidean distance of each point from these K centroids and assign it to that group/cluster to which it is nearest.
3. After assigning each point to a cluster, cluster centroids are recalculated as the mean of the points in their respective groups.
4. With the new centroids, the process is repeated from step 2 onwards until a point of convergence is reached.

Now how do we select this hyperparameter K? One way is to plot **total variance** v/s **K** and select that K for which the reduction in variance is greatest. Mean shift algorithm is a mode seeking algorithm. The main difference from K-Means is that the number of clusters K is not known beforehand. Number of clusters and its size only depend on the window size (bandwidth) in Meanshift algorithm.

Problem 2
[1pt] What is softmax? Under what cases does softmax computation become unstable?

Ans: Softmax is a function which is used to convert a vector x into a probability distribution. It is generally used in neural networks to get output probabilities for N classes. Equation of softmax is as follows:

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{1}$$

Softmax becomes numerically unstable when the range of the input vector is very large. This is due to the fact that exponentiation in this case leads to either underflow or overflow. One way to make this numerically stable is to shift x_i by $c = max(x)$ i.e

$$\sigma(x_i) = \frac{e^{x_i - c}}{\sum_j e^{x_j - c}} \tag{2}$$

Problem 3
[1pt] What is regularization in CNNs? Explain one common technique used for regularization.

Ans: Regularization is a technique used to reduce over-fitting by penalizing the parameters. It tries to reduce the variance of the estimator so that the expected error decreases. Few Regularization techniques are as follows:

1. L1
2. L2
3. Data Augmentation
4. Dropout

I will here explain both **L1** and **L2** regularization. Both **L1 and L2** add a regularization term to the cost function to decay the value of weights. These terms are as follows:

$$For L1 : Cost = Loss + \lambda * \sum_j ||w_j|| \tag{3}$$

$$For L2 : Cost = Loss + \lambda * \sum_j ||w_j||^2 \tag{4}$$

Here λ is the regularization hyper-parameter.

Problem 4
[1pt] What is a GAN? What does it mean for a GAN to 'mode collapse'?

Ans: **Generative Adversarial Networks (GANs)** are unsupervised models which are used to generate data from a target distribution. GANs consists of 2 sub-modules: a *generator* which is used to generate a realistic looking fake data and *discriminator* which is used to differentiate real and generated data. These two sub-modules compete against each other in the form of a minimax game.

Mode Collapse: It is one of the failure cases of GAN in which the generator produces outputs with little variety. This happens as generator learns that the examples of a particular mode are enough to fool the discriminator. When its discriminator's turn to learn, its determines that this mode is fake and examples of another mode are real. The generator then exploits this weakness and starts producing examples of the other mode.Thus, it is like a game of cat and mouse where the generator dosen't have the incentive to learn all the modes.

Problem 5
[1pt] What are some common initialization methods for Convolution layers and Fully Connected layers in a CNN?

Ans: Common weight initialization methods for CNNs and NNs are as follows:

- **Normal**
- **Uniform**
- **Xavier Normal**
- **Xavier Uniform**

Among these *Xavier* initialization is most popular and has shown to reduce the problem of vanishing and exploding gradients to some extent. The weights for a layer l in case of *Xavier Normal* are sampled from a normal distribution with variance as $\frac{1}{n_{l-1}}$ i.e

$$w_l \sim \mathcal{N}(\mu = 0, \sigma^2 = \frac{1}{n_{l-1}}) \tag{5}$$

Here n_{l-1} is the number of nodes in layer $l-1$.

Problem 6
[5pts] Colab Notebook: Link

Create a function that can find an approximate minimum in a given vector function fn that takes an n_d dimension vector as an input and gives a scalar-valued output. Treat the function fn as a black box; you can query it but you do not have any other information about it (More specifically, you cannot take the gradient of this function). Check the colab link above, edit it, and submit a link to your edited colab in the write-up.

Ans: Link for the solution: Colab Solution Notebook

Problem 7
[5pts] Colab Notebook: Link

One interesting application of machine learning is in the area of medical diagnoses. Implement Neural Network in python to classify the data into Benign or Malignant for the Wisconsin Diagnostic Breast Cancer (WDBC) dataset.

Create your model as a self-contained Google Colab notebook, and attach a link to your write-up. For your convenience, we have created an (incomplete) colab notebook here that you can copy and start working on.

Things to keep in mind:

- How will you choose the features?
- How will you train, validate, and test your model?
- How will you overcome overfitting?

Dataset: Link
Download Data from here: Link
Dataset Description:

#Attribute	Domain
1. Sample code number	id number
2. Clump Thickness	1 - 10
3. Uniformity of Cell Size	1 - 10
4. Uniformity of Cell Shape	1 - 10
5. Marginal Adhesion	1 - 10
6. Single Epithelial Cell Size	1 - 10
7. Bare Nuclei	1 - 10
8. Bland Chromatin	1 - 10
9. Normal Nucleoli	1 - 10
10. Mitoses	1 - 10
11. Class:	(2 for benign, 4 for malignant)

In your write-up, please attach training curves, final losses, and visualizations of errors made by the trained model, along with a description of the methods/tricks you tried to improve performance.

Ans: Link for the solution: Colab Solution Notebook