HW5 :

Question 1:

a) Compare the advantages and disadvantages of Decision Trees and Neural Networks. Additionally, explain which type of dataset each of these models is more suitable for.

Advantages and Disadvantages:

1. **Interpretability and Explainability:**
    - *Decision Trees* provide high interpretability as they follow a hierarchical structure, making it easy to trace the decision-making process. This feature is crucial in domains like healthcare and finance where decision transparency is important.
    - *Neural Networks*, on the other hand, are often referred to as a "black box," as it is difficult to explain how they arrive at a decision. This makes them less interpretable compared to Decision Trees.
2. **Training Speed and Computational Cost:**
    - *Decision Trees* are fast to train, especially on small to medium-sized datasets. They require minimal computational resources and can be trained relatively quickly on a standard CPU.
    - *Neural Networks*, especially deep networks, require significant computational resources and time to train. Deep learning models often need specialized hardware like GPUs or TPUs, making them more expensive to train compared to Decision Trees.
3. **Ability to Model Complex, Non-linear Relationships:**
    - *Decision Trees* perform well when the decision boundaries are simple and linear. However, they struggle to capture complex non-linear relationships.

- *Neural Networks*, with their multi-layer architecture, excel in capturing complex, non-linear patterns. They are widely used in tasks like image recognition, speech processing, and natural language processing, where relationships between features are intricate.

4. **Risk of Overfitting and Generalization:**
    - Both models are prone to overfitting but tackle it in different ways. *Decision Trees* can overfit if they become too complex, retaining too much detail from the training data. Techniques like pruning or limiting tree depth can reduce overfitting.
    - *Neural Networks* can also overfit, particularly deep models. Regularization methods like dropout, batch normalization, and L2 regularization are used to mitigate overfitting.
5. **Need for Data and Feature Engineering:**
    - *Decision Trees* work well with small datasets and can also handle missing values naturally. However, they often require manual feature selection to achieve optimal performance.
    - *Neural Networks* generally require large datasets to perform well and can automatically extract relevant features from raw data, making them highly effective in tasks like image processing where feature extraction is complex.
6. **Scalability and Performance on Large Datasets:**
    - *Decision Trees* are efficient on small to medium datasets but may struggle with large datasets due to increased complexity in time and memory.
    - *Neural Networks*, particularly convolutional neural networks (CNNs), are designed to scale efficiently with large datasets and can process vast amounts of data to identify meaningful patterns.
7. **Robustness to Noise and Handling Missing Data:**
    - *Decision Trees* are sensitive to small changes in the data and can produce drastically different models with slight variations in the dataset. They are less robust to noisy data.

- *Neural Networks* are generally more robust to noise due to their distributed learning structure, although they still require proper regularization to avoid learning invalid patterns.

Suitable Datasets for Each Model:

- **Decision Trees:**
  - Structured and tabular data with clear hierarchical relationships between features.
  - Small to medium-sized datasets.
  - Datasets with missing values.
  - Simple to moderately complex patterns.
  - Imbalanced datasets.
- **Neural Networks:**
  - Unstructured data like images, audio, and text.
  - Large datasets.
  - Non-linear and complex relationships.
  - Time-series data.

b) Why is using the activation function f(x) = x in the hidden layers of an MLP not suitable?

The main purpose of activation functions in hidden layers is to introduce non-linearity, allowing the network to model complex patterns. If we use f(x) = x (i.e., a linear function) in the hidden layers, the network behaves like a single-layer perceptron (SLP), unable to model non-linear relationships. This would essentially render additional layers redundant because a single-layer network can already achieve the same result.

Additionally, in deep networks, adding layers helps extract hierarchical features. With a linear activation, adding layers has no effect on the network's capability to model complex relationships.

In backpropagation, activation functions' derivatives are used to update weights. For a linear activation, the derivative is constant (1), meaning the gradient would not change as it propagates through the layers, making backpropagation ineffective.

c) Why is the RELU activation function not differentiable?

The RELU function is not differentiable at zero because its left-hand and right-hand derivatives are not the same:

- $\lim_{h \to 0^-} \frac{\text{RELU}(0+h) - \text{RELU}(0)}{h} = 0$
- $\lim_{h \to 0^+} \frac{\text{RELU}(0+h) - \text{RELU}(0)}{h} = 1$

Despite this, RELU is still used in neural network training. Since it is only non-differentiable at a single point (x = 0), frameworks like PyTorch assume a gradient of 0 at this point, which avoids issues during backpropagation and allows the model to continue learning.

d) In the gradient ascent learning process, which points in the loss function landscape can cause premature stopping or slow learning? Which of these points presents a bigger challenge for learning, and why? Also, suggest two methods to avoid getting stuck in these points during training.

In the gradient ascent process, two types of points in the loss function landscape can hinder learning:

1. **Saddle Points:** These points have zero gradients but differ from minima because the function increases in some directions and decreases in others. Gradient ascent can slow down or halt near saddle points because the gradient becomes very small, leading to a stagnant learning process.
2. **Local Maxima:** Local maxima are points where the function's value is higher than its neighboring points, but not necessarily the global maximum. If the model gets stuck at a local maximum, it might believe learning has ended, even though better solutions exist.

Local maxima present a bigger challenge because the model might incorrectly assume that further improvements are impossible. In contrast, saddle points can often be overcome by small random changes in the gradient.

Two methods to avoid getting stuck:

1. **Momentum or Adam Optimizer:** Momentum helps keep the gradients moving in previous directions, increasing the chances of escaping saddle points. Adam optimizer combines momentum with adaptive learning rates, adjusting the learning rate when gradients are small, preventing learning from halting.
2. **Noise or Simulated Annealing:** Introducing small random noise into the gradients can help the model escape saddle points or local maxima. Simulated annealing gradually reduces the noise during the later stages of training, allowing the model to settle into a stable solution.