IIT BOMBAY

Indian Institute of Technology Bombay

CS 419 Midsem

Date: September 20, 2024 **Max Marks: 30** **Duration: 120 minutes**

Instructions:

- Answer all questions.
- Show all your work clearly, be as clear and precise as possible.

Useful Identities

1. Gradient of the dot product

$$\nabla_{\mathbf{u}}(\mathbf{a}^T\mathbf{u}) = \mathbf{a}$$

2. Gradient of a Quadratic Form

$$\nabla_{\mathbf{u}}(\mathbf{u}^T\mathbf{A}\mathbf{u}) = (\mathbf{A} + \mathbf{A}^T)\mathbf{u}$$

where:

- $\mathbf{a}$ is a vector of size $d \times 1$.
- $\mathbf{u}$ is a vector of size $d \times 1$.
- $\mathbf{A}$ is a matrix of size $d \times d$.

Question 1 : LINEAR REGRESSION

A dataset consists of n data points, each with a feature vector and a corresponding output:

$$\textit{Dataset}: \{(\mathbf{x}^{(i)}, y^{(i)}) \mid i = 1, 2, \ldots, n\}$$

where $\mathbf{x}^{(i)} \in \mathbb{R}^d$ is the input vector for the i-th data point and $y^{(i)} \in \mathbb{R}$ is the actual output label for the i-th data point. The linear regression model is defined as:

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \mathbf{x}^T\mathbf{w}$$

The linear regression loss is defined as:

$$J(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \mathbf{x}^{(i)T}\mathbf{w}\right)^2$$

However, this linear model may not be suitable for non-linear data. To address this limitation, we introduce a new model called locally weighted regression which weighs the loss of each data point as follows:

$$J_{\text{new}}(\mathbf{w} \mid \mathbf{x}) = \frac{1}{2n}\sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}^{(i)})\left(y^{(i)} - \mathbf{x}^{(i)T}\mathbf{w}\right)^2$$

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2}\right)$$

Note that the Loss is now conditionally dependent on point $\mathbf{x} \in \mathbb{R}^d$ in the input space. The training and inference procedure **are NOT exclusive anymore**. We cannot calculate the loss and closed form solution(of the optimal $\mathbf{w}$) for the given dataset once and discard the dataset. The dataset has to be used every time for every new query point $\mathbf{x}$.

Given a query point $\mathbf{x}$ for which prediction is to be performed, we will first calculate the conditional loss $J_{\text{new}}(\mathbf{w} \mid \mathbf{x})$ and then minimize it. The inference procedure for locally weighted regression involve slight modifications in the mathematical formulation to account for the weighted nature of the data. More formally the procedure is detailed below:

Given any general fixed point $\mathbf{x}$ in the input space, follow these steps to find $\hat{y}(\mathbf{x})$:

1. **Compute Weights:**
   Calculate the weight for each data point $\mathbf{x}^{(i)}$ in the dataset using the kernel function:
   $$K(\mathbf{x}, \mathbf{x}^{(i)}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}^{(i)}\|^2}{2}\right)$$

2. **Minimize the Weighted Loss Function:**
   Compute the minima of the locally weighted loss function $\mathbf{w}^*(\mathbf{x})$ by minimizing $J_{\text{new}}(\mathbf{w} \mid \mathbf{x})$ Note that the minimization is still performed w.r.t parameter $\mathbf{w}$

3. **Make Prediction:**
   Predict the output $\hat{y}(\mathbf{x})$ for the given point $\mathbf{x}$ using:
   $$\hat{y}(\mathbf{x}) = \mathbf{x}^T\mathbf{w}^*(\mathbf{x})$$

Lets begin the problem now

(a) **For this part only** assume $d = 1$. Analyse the loss $J_{\text{new}}(\mathbf{w} \mid \mathbf{x})$. What is the effect of weighing $\left(y^{(i)} - \mathbf{x}^{(i)T}\mathbf{w}\right)^2$ by the weight term $K(\mathbf{x}, \mathbf{x}^{(i)})$? How does this help the model to fit non linear data better? [2 marks]

As linear regression problems often have a closed form solution, it is better to formulate the problem in terms of matrices. We can then use matrix calculus to get the required results. In case of simple linear regression we can write model as :

$$\hat{\mathbf{Y}} = \mathbf{X}\mathbf{w}$$

and loss as

$$J(\mathbf{w}) = \frac{1}{2n}\sum_{i=1}^{n}\left(y^{(i)} - \mathbf{x}^{(i)T}\mathbf{w}\right)^2 = \frac{1}{2n}\|\mathbf{Y} - \hat{\mathbf{Y}}\|^2 = \frac{1}{2n}\|\mathbf{Y} - \mathbf{X}\mathbf{w}\|^2$$

where

$$\mathbf{X_{nxd}} = \begin{bmatrix} \text{........}\mathbf{x}^{(1)T}\text{........} \\ \text{........}\mathbf{x}^{(2)T}\text{........} \\ \vdots \\ \text{........}\mathbf{x}^{(n)T}\text{........} \end{bmatrix} = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_d^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{(n)} & x_2^{(n)} & \dots & x_d^{(n)} \end{bmatrix}$$

$$\mathbf{Y_{n\times1}} = \begin{bmatrix} y^{(1)} & y^{(2)} & \cdots & y^{(n)} \end{bmatrix}^T$$

A similar formulation can be made for locally weighted regression. We just carefully need to design a matrix $\mathbf{K}$ and use it to formulate :

$$J_{\text{new}}(\mathbf{w} \mid \mathbf{x}) = \frac{1}{2n}\sum_{i=1}^{n} K(\mathbf{x}, \mathbf{x}^{(i)})\left(y^{(i)} - \mathbf{x}^{(i)T}\mathbf{w}\right)^2$$

(b) What is the shape of matrix $\mathbf{K}$ [1 mark]

(c) How does any general element $K_{i,j}$ look for matrix $\mathbf{K}$ [1 mark]

(d) How is the loss $J_{\text{new}}(\mathbf{w} \mid \mathbf{x})$ formulated in terms of $\mathbf{X}, \mathbf{Y}, \mathbf{K}, \mathbf{w}$ [2 marks]

(e) Once the loss is formulated for above part **DERIVE** a closed form solution for $\mathbf{w}^*(\mathbf{x})$. [4 marks]

Question 2 : LOSS FUNCTIONS

P1. Suppose you have a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1, 2, 3\}$. y_i is an *ordered categorical* variable, meaning that the labels are discrete and ordered $(0 < 1 < 2 < 3)$. The labels can be considered as scores. You are also given a linear predictor model $m_\theta(\mathbf{x}_i) = \theta^T \mathbf{x}_i$.

(I) Provide a method to convert the predictions of m_θ to ordered categorical and then calculate the loss on that. Justify your choice in a few sentences. [2 marks]

P2. Now consider a setting where you have a real human h that can provide predictions for each example $\mathbf{x}_i$. The predictions generated by human are denoted as $h(\mathbf{x}_i)$. h is a fixed function and is known in advance, meaning h has no trainable parameters. You can still evaluate the performance of h on the dataset $\mathcal{D}$ through the loss function $\mathcal{L}(h(\mathbf{x}_i), y_i)$. For the next question, assume that the loss function $\mathcal{L}$ can be used for both human h and model m_θ.

(II) Your task is to complete the optimization problem below that quantifies the loss when both m_θ and h are used as predictors, in terms of the loss $\mathcal{L}$ mentioned above. h makes predictions on a fixed set of examples $S \subset \mathcal{D}$ and m_θ makes predictions on the rest. Explain your answer with a concrete example of S and $\mathcal{D}$. [2 marks]

$$\min_\theta \left[\qquad\qquad\qquad\qquad\qquad\qquad \right]$$

P3. We will now introduce the *pairwise ranking loss*. Consider a pair of samples in $\mathcal{D}$ for which y_i and $y_{i'}$ are the true labels and $\hat{y}_i$ and $\hat{y}_{i'}$ are the predicted labels on these pair of points. Then the ranking loss is defined as:

$$\ell(\hat{y}_i, \hat{y}_{i'}, r_{ii'}) = [\Delta - sign(r_{ii'})(\hat{y}_i - \hat{y}_{i'})]_+$$

where, $\hat{y}_i, \hat{y}_{i'} \in \mathbb{R}$, $\Delta \geq 0$ is the loss margin, $r_{ii'} = (y_i - y_{i'})$ and $[expr\ ]_+$ evaluates to 0 if $expr\ < 0$ else $expr$.

(III)

$$\hat{\mathbf{y}} = \begin{bmatrix} 3.0 \\ 1.5 \\ 0.3 \\ 2.4 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} 0 \\ 3 \\ 1 \\ 2 \end{bmatrix}$$

Evaluate the pairwise ranking loss for above data using $\Delta = 1$:

$$\sum_{1 \leq i < j \leq 4} \ell(\hat{y}_i, \hat{y}_j, r_{ij})$$

[4 marks]

(IV) Describe, in your own words, what the ranking loss measures. Also explain what happens when the margin Δ is changed? [2 marks]

Question 3 : PYTORCH

```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, TensorDataset

class SimpleNN(nn.Module):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.fc1 = nn.Linear(10, 50, bias=False)
        self.fc2 = nn.Linear(30, 50, bias=False)
        self.fc3 = nn.Linear(30, 10,  bias=False)
    def forward(self, x):
        out = x
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        x = self.fc3(x)
        return x + out              # Output layer

num_samples = 1000
num_features = 10
num_classes = 4
X = torch.randn(num_samples, num_features)
y = torch.randint(0, num_classes, (num_samples,))

# Create a DataLoader
dataset = TensorDataset(X, y)
dataloader = DataLoader(dataset, batch_size=32)

# Initialize the network, loss function, and optimizer
model = SimpleNN()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Training loop
num_epochs = 5
for epoch in range(num_epochs):
    train_loss = 0
    for batch_X, batch_y in dataloader:
        # Forward pass
        outputs = model(batch_X)
        loss = criterion(outputs, batch_y)

        # Backward pass and optimization
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

print("Training_Completed")
```

The above figure shows a PyTorch code snippet for training a basic neural network. The code contains **two** errors with respect to dimensional mismatch. Identify them and clearly report what the errors are. Explain in a few sentences why the code will break because of that error. [10 marks]