Behavioral Data Science Homework 2
Kelly Modi, Dev Dwivedi
21 Feb 2025

Repository Link: https://github.com/kellymodi/homework-2.git

Problem 1: True/False Q (4)

1. *True*
2. *False* - Adam is actually faster because it uses its adaptive learning rate mechanism.
3. *False* - The shape is (m, n) instead of (n, m) so the Jacobian matrix of a function has m rows and n columns.
4. *False* - Mini-batch gradient descent is most commonly used because of its efficiency and stability.
5. *False* - TensorFlow's autodiff uses symbolic differentiation to compute gradients.
6. *False* - Only the first derivative is used in most cases.
7. *True*
8. *True*

Problem 2: PyTorch vs. TensorFlow (4)

PyTorch has a Pythonic syntax which makes it more intuitive and easier to use especially for people with a strong Python background. TensorFlow has a user-friendly high-level API called Keras but can take longer to learn and require more complex debugging. Another difference involves product readiness. PyTorch provides TorchScript and ONNX as deployment options, whereas TensorFlow has stronger support with TensorFlow Serving and TensorFlow Lite, which are more mature solutions. PyTorch is more favored in academia and research due to its flexibility from using a dynamic computation graph, which is a graph built as code is executed. TensorFlow is more commonly used in industry with better tooling for large-scale production. By default, TensorFlow uses a static computation graph, which means defining the entire graph before executing it, but TensorFlow 2.x offers a more dynamic approach.

Code Snippets:

```
import tensorflow as tf

# Specifying the model
model = tf.keras.Sequential([
    tf.keras.Input(shape=(20,)),
    tf.keras.layers.Dense(128, activation="tanh"),
    tf.keras.layers.Dense(128, activation="tanh"),
    tf.keras.layers.Dense(128, activation="tanh"),
```

```
    tf.keras.layers.Dense(1, activation="sigmoid")
])
```

```
import torch
import torch.nn as nn

# Define the model
class ASDClassifier(nn.Module):
    def __init__(self):
        super(ASDClassifier, self).__init__()
        self.fc1 = nn.Linear(20, 128)
        self.fc2 = nn.Linear(128, 128)
        self.fc3 = nn.Linear(128, 128)
        self.fc4 = nn.Linear(128, 1)
        self.tanh = nn.Tanh()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        x = self.tanh(self.fc1(x))
        x = self.tanh(self.fc2(x))
        x = self.tanh(self.fc3(x))
        x = self.sigmoid(self.fc4(x))
        return x

# Instantiate the model
model = ASDClassifier()
```

Problem 3: Gradient Descent (6)

Part 1:

The gradient of a multi-variable function is a vector that has all of the partial derivatives of the function and provides important information for optimization. The gradient shows the direction and rate of the steepest increase of the function. Moving in the direction of the gradient maximizes the function, while the opposite minimizes it. On that note, gradient descent is an optimization algorithm used to minimize functions by iteratively moving in the direction of the steepest descent. It is commonly used to optimize loss functions.

3-1) $C(\theta_1, \theta_2) = 2\theta_1^2 + 3\theta_2\theta_1$ gradient = $\nabla C(\theta_1, \theta_2)$

$\frac{\partial C}{\partial \theta_1} = 4\theta_1 + 3\theta_2$

$\frac{\partial C}{\partial \theta_2} = 3\theta_1$

$\nabla C(\theta_1, \theta_2) = (4\theta_1 + 3\theta_2, 3\theta_1)$

$\theta_t = \theta_{t-1} - \alpha \nabla C(\theta_t)$ $\alpha = 0.1$, $\theta_1 = 2$, $\theta_2 = 9$

step 1: $\nabla C(2, 9) = (4 \cdot 2 + 3 \cdot 9, 3 \cdot 2) = (35, 6)$

$\theta_1 = 2 - 0.1(35) = -1.5$

$\theta_2 = 9 - 0.1(6) = 8.4$

step 2: $\nabla C(-1.5, 8.4) = [4(-1.5) + 3(8.4), 3(-1.5)] = (19.2, -4.5)$

$\theta_1 = -1.5 - 0.1(19.2) = -3.42$

$\theta_2 = 8.4 - 0.1(-4.5) = 8.85$

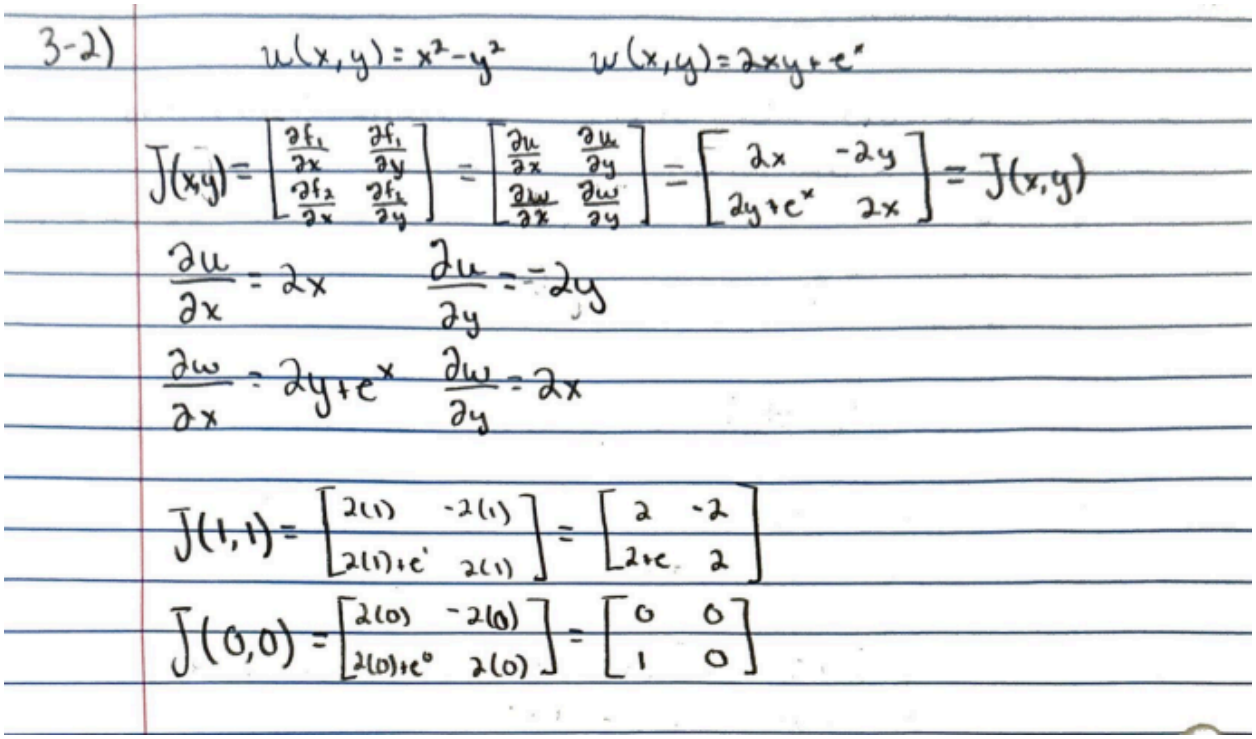
step 3: $\nabla C(-3.42, 8.85) = [4(-3.42) + 3(8.85), 3(-3.42)] = [12.87, -10.26]$

$\theta_1 = -3.42 - 0.1(12.87) = -4.707$

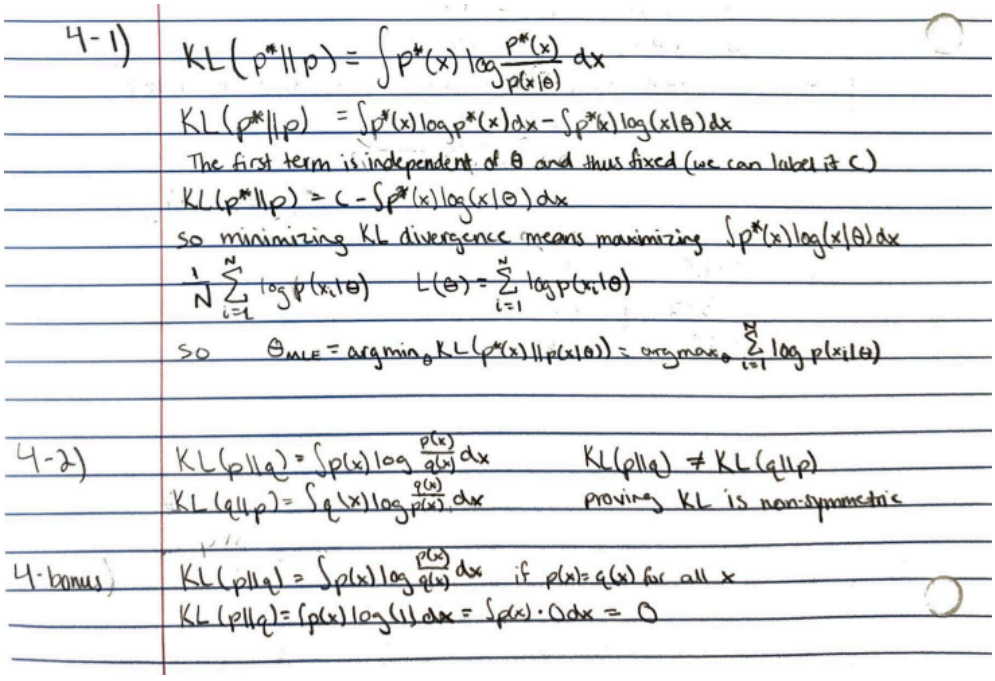
$\theta_2 = 8.85 - 0.1(-10.26) = 9.876$

step	θ_1	θ_2
initial	2	9
1	-1.5	8.4
2	-3.42	8.85
3	-4.707	9.876

Part 2:



Problem 4: Maximum Likelihood Estimation Continued (8)



Problem 5: Simple Mixture-of-Experts (MoE) Approach (10)

Included in GitHub Repository linked at the top.