Homework 1

Overview

Due: 21 Jan 2026

- **Unless otherwise directed, please derive and show your work.**
- **Do not try to search directly for the answers.**
- **You may speak to other students about the assignment at a high-level (e.g. sharing related references/slides).**
- **However, sharing complete or partial answers is strictly prohibited.**
- **Please see the *UNC Student* _*Code of Conduct*_ for additional details on maintaining academic integrity**

Part 1: Documentation of ML Implementation

- For **this problem** you are welcome to **choose documentation** from **any** programming **language** (*R, Python, Julia, MATLAB*, etc.) in which an **implementation** of a **machine learning** (ML) method is discussed
- During the **first units** of the course, you have been **exploring** a variety of **concepts** from **calculus**, **probability**, **statistics**, and **linear algebra** that support **multiple methods** in **machine learning** methods and **algorithms**
- In this problem you are asked to **compare documentation** from **two** different **implementations** of any **ML method** or **algorithm** related to any of the **fundamental concepts** discussed in ***Units 1*** *and* ***2***

A. (10 pts)	• **Include** a direct **link** to the **documentation** for the **two implementations** you selected • These two **implementations** may come from **different** programming **languages**, or **different** library/**packages** in the **same** programming **language** • Please **summarize** in **2 – 3 sentences** some high-level **differences** that you notice between the **two implementations**

B. (6 pts)	• Frem the **method** you selected in **part 'A'**, list a research **article** from **any domain**) in which the **ML method** of your choice has **been used** • Include **title**, **authors**, **abstract**, and a direct **link** to the specific **article**

Part 2: Eigendecomposition

- For this problem you can **create** a **sample matrix** (following the stated properties) of your choice to **solve** problems **numerically**, **OR write** down the **mathematical expression**(s) for **solutions/proofs** in each case
- Suppose that $A \in R^{nxn}$ can be **written** down as $A = QDQ^T$ where $D \in R^{nxn}$ is a **diagonal matrix** and $Q^{-1} = Q^T$
- If you **want** to **solve** this problem **computationally**, you could use **this** sample **matrix**:

$$A = \begin{bmatrix} 13 & -4 & 2 \\ -4 & 11 & -2 \\ 2 & -2 & 8 \end{bmatrix}$$

- Confirm that the **matrix decomposition** below returns A (that is, verify that the product of QDQ^T:

$$A = \begin{bmatrix} \frac{\sqrt{5}}{3} & -\frac{2}{3} & 0 \\ -\frac{4}{3\sqrt{5}} & -\frac{2}{3} & \frac{1}{\sqrt{5}} \\ \frac{2}{3\sqrt{5}} & \frac{1}{3} & \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} 17 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{5}}{3} & -\frac{4}{3\sqrt{5}} & \frac{2}{3\sqrt{5}} \\ -\frac{2}{3} & -\frac{2}{3} & \frac{1}{3} \\ 0 & \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix}$$

A. **(8 pts)**	• show that A is **symmetric**
B. **(8 pts)**	• show that **QDQ^T** is the **Eigendecomposition** of A • That is, **show** that the **columns** of **Q** are **eigenvectors** of A (and **specify** corresponding **eigenvalues**)
C. **(8 pts)**	• write out the **Eigendecomposition** of $A + \lambda I$ for some $\lambda > 0$
D. **(8 pts)**	• **read** the **data** stored in the *Advertising.csv* in a matrix B

	• **compute** B^TB and **report** any two **properties** of the resulting **matrix** (e.g. **eigenvalues,** trace**, determinant,** singular value **decomposition,** largest eigenvalue**, etc.)**`
Part 3: Probability of Statistics Basics	
A. (12pts)	• **assume** that you are **tasked** with **building** a simple **binary classifier** that will eventually **predict** whether a user will **click** on an **ad** (***yes/no***) • right now, you do **not** have a **model** but only **historical click** rates. • suppose that the **historical click**-through **rate** (***CTR***) is ***0.3*** • in **this problem** you will **simulate** user **behavior** and **observe** how the **estimated *CTR* varies** across **samples**
i.	• **simulate *1000* users** using the ***binomial distribution*** • each **user** has a ***0.3* probability** of **clicking**
ii.	• now, **simulate *1000* experiments**, each with ***100* users** • **plot** the **distribution** of **estimated *CTRs*** to see the **variability**
iii.	• **comment** on your **results**

B. (12 pts)	• **suppose** that a medical **test** for a **disease** has the **following characteristics**:
	  ○ **sensitivity** (true **positive** rate): ***P(test+|disease+) = 0.95***
	  ○ **specificity** (true **negative** rate): ***P(test-|disease-) = 0.90***
	  ○ **prevalence**: ***P(disease+) = 0.02***

i.	• **calculate** the **probability** that a person **has** the **disease** given that they **test positive**: ***P(disease+|test-)***
ii.	• **calculate** the **probability** that a **person** does not have the **disease** given that they test **negative *P(disease-|test-)***
iii.	• **simulate** a **population** of ***100,000*** people and **empirically estimate** these probabilities

	• **compare** your **results** to the theoretical **calculations** • *the **theoretical** and **empirical results** should **closely match***
C. (8 pts)	• let $N(x\,\|\,\mu, \Sigma) = (2\pi)^{-k/2} \det(\Sigma)^{-1/2} \exp(-1/2\,(x-\mu)^T \Sigma^{-1} (x-\mu))$ be the **multivariate normal** for $x^2 \in R^d$ with **mean** μ and **covariance** Σ • **suppose** further that Σ is a **diagonal** • show, for **diagonal covariance**, that $N(x / \mu, \Sigma)$ is an **independent distribution** over the **dimensions** of x
if solving the problem numerically	• Define the **multivariate normal distribution** with a **diagonal covariance** matrix using `` ` ``***np.diag()***`` ` `` • Show that the **distribution** is **independent** over the **dimensions** by **demonstrating** that the **joint PDF** can be **written** as the **product** of the **individual PDFs** for each **dimension** • **Create** a **function** to **calculate** the **multivariate normal PDF** and use a **loop** to **evaluate** it for different **values** of the **variable**
Part 4: Gradients	
A. (8 pts)	• suppose there is a function $f(\theta)$ with gradient $\nabla_\theta f$ • based on some current **value** of the **input**, θ_0, *what is* an update that will **yield** a θ_1 such that $f(\theta_0) \geq f(\theta_1)$?
B. (4 pts)	• using ‘***A’***, **derive** an iterative **algorithm** to **minimize** f *with* respect *to* θ when starting *with* θ_0
C. (12 pts	• consider the code-block below • use the **algorithm** in ‘***B’*** to minimize $f(\theta) = \|\|\theta\|\|_2^2 - 2v^T\theta$, where $\theta \in R^3$ and $v = [0.2, 0.1, 0.3]^T$

```python
import numpy as np
import matplotlib.pyplot as plt

def loss(theta):
    v = np.array(object=[0.2, 0.1, 0.3])
    return np.linalg.norm(x=theta)**2 - 2*v.dot(theta)
```

```
theta = np.array(object=[1.0, 0.0, 0.0])
losses = []

for i in range(stop/100):
    theta = None  # TODO implement update  <<<<<<<<<<<
    losses.append(object/loss(theta=theta))
```

```
#plt.plot(losses)

#print(theta)
```