**Student name: Thi Vu**

**Student number: 151394898**

**DATA.STAT.770 Dimensionality Reduction and Visualization, Spring 2025**

**Exercise set 1**

**Exercise 1:**

Code:

import numpy as np

def compute\_proportions(dimensions, num\_points=10\_000\_000):

results\_hypersphere = {}

results\_shell = {}

for d in dimensions:

# Generate points uniformly in [-1, 1]^d

points = np.random.uniform(-1, 1, size=(num\_points, d))

# Compute squared distances from the origin

squared\_distances = np.sqrt(np.sum(points\*\*2, axis=1))

# a) Proportion inside the hypersphere

inside\_hypersphere = squared\_distances <= 1

proportion\_hypersphere = np.mean(inside\_hypersphere)

results\_hypersphere[d] = proportion\_hypersphere

# b) Proportion inside the spherical shell

inside\_shell = (squared\_distances >= 0.95) & (squared\_distances <= 1)

proportion\_shell = np.sum(inside\_shell) / np.sum(inside\_hypersphere)

results\_shell[d] = proportion\_shell

return results\_hypersphere, results\_shell

# Dimensions to test

dimensions = [1, 2, 3, 4, 7, 11, 16]

# Run the computation

proportions\_hypersphere, proportions\_shell = compute\_proportions(dimensions)

# Display results

# a) Proportion of points inside the hypersphere

print("Proportion of points inside the hypersphere:")

for d, proportion in proportions\_hypersphere.items():

print(f"Dimension {d}: {proportion:.6f}")

# b) Proportion of points inside the spherical shell

print("\nProportion of points inside the spherical shell:")

for d, proportion in proportions\_shell.items():

print(f"Dimension {d}: {proportion:.6f}")

Result:

1. Proportion of points inside the hypersphere:

Dimension 1: 1.000000

Dimension 2: 0.785385

Dimension 3: 0.523193

Dimension 4: 0.308332

Dimension 7: 0.036971

Dimension 11: 0.000925

Dimension 16: 0.000004

1. Proportion of points inside the spherical shell:

Dimension 1: 0.050055

Dimension 2: 0.097479

Dimension 3: 0.142634

Dimension 4: 0.185454

Dimension 7: 0.301673

Dimension 11: 0.420154

Dimension 16: 0.534884

Exercise 2:

Code:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsRegressor

def generate\_data(d, num\_points=2000):

# Generate data points normally distributed around the origin

X = np.random.normal(0, 1, size=(num\_points, d))

# Compute the target variable y

y = X[:, 0] + np.sin(4 \* X[:, 0])

return X, y

def split\_data(X, y):

# Split into 1000 training and 1000 testing points

return X[:1000], y[:1000], X[1000:], y[1000:]

def train\_and\_predict\_knn(X\_train, y\_train, X\_test, n\_neighbors=5):

# Train a 5-Nearest Neighbor Regressor

knn = KNeighborsRegressor(n\_neighbors=n\_neighbors)

knn.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = knn.predict(X\_test)

return y\_pred

def compute\_mse(y\_true, y\_pred):

# Compute Mean Squared Error

return np.mean((y\_true - y\_pred)\*\*2)

def plot\_results(x1\_test, y\_test, y\_pred, d):

plt.figure(figsize=(8, 6))

plt.scatter(x1\_test, y\_test, color='blue', label='True Values', alpha=0.6, s=10)

plt.scatter(x1\_test, y\_pred, color='red', label='Predicted Values', alpha=0.6, s=10)

plt.title(f"Dimensionality: d = {d}")

plt.xlabel("x1")

plt.ylabel("y")

plt.legend()

plt.grid(True)

plt.show()

def main():

dimensions = [1, 2, 3, 4, 7, 11, 16]

mse\_results = {}

for d in dimensions:

# Generate data

X, y = generate\_data(d)

# Split into training and testing sets

X\_train, y\_train, X\_test, y\_test = split\_data(X, y)

# Train and predict using 5-NN

y\_pred = train\_and\_predict\_knn(X\_train, y\_train, X\_test)

# Compute mean squared error

mse = compute\_mse(y\_test, y\_pred)

mse\_results[d] = mse

# Plot results

plot\_results(X\_test[:, 0], y\_test, y\_pred, d)

# Print MSE results

print("Mean Squared Error for each dimensionality:")

for d, mse in mse\_results.items():

print(f"Dimension {d}: MSE = {mse:.6f}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

The plot:

A graph with red and blue lines

AI-generated content may be incorrect.

A graph with red and blue dots

AI-generated content may be incorrect.

A graph with red and blue dots

AI-generated content may be incorrect.

A graph with red and blue dots

AI-generated content may be incorrect.

A graph with red dots and blue lines

AI-generated content may be incorrect.

A graph with red dots and blue lines

AI-generated content may be incorrect.

A graph with red and blue dots

AI-generated content may be incorrect.

Mean Squared Error for each dimensionality:

Dimension 1: MSE = 0.001607

Dimension 2: MSE = 0.043631

Dimension 3: MSE = 0.143781

Dimension 4: MSE = 0.362643

Dimension 7: MSE = 0.654743

Dimension 11: MSE = 0.762133

Dimension 16: MSE = 0.979365