Assignment 1: CSCI433/933
Exploring Linear Regression through Deep Learning & Statistical Learning

Instructor: Professor Philip O. Ogunbona
School of Computing and IT
Room 3.113

Total marks: 100
Submission deadline: Week 7

1 Objective

This assignment aims to bridge classical statistical learning with deep learning by investigating linear regression through the lenses of:

1. Statistical learning (ridge, lasso, elastic net)
2. Deep learning (linear neural networks, weight decay, dropout)
3. Regularization techniques (L1, L2, dropout, data augmentation)

Students will compare performance across techniques using a real-world dataset and critically analyze the trade-offs.

2 Dataset

We will use a modified version of the "Bike Sharing Dataset" from UCI Machine Learning Repository. The dataset consists of bike rental counts as the target variable and features such as temperature, humidity, and season.

2.1 Download & Preprocessing

Dataset Source: [UCI Bike Sharing Dataset](https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset)

To simplify the task, the dataset has been preprocessed to include the features shown in Table 1:

Students should split the dataset into 80% training and 20% testing. Before conducting any experiment with a dataset, it is important to visualize the data and understand its statistics. These observations may be useful in explaining some of the results. For this dataset, use knowledge from Chapter 2 of the book by Géron (2019, Chapter 2). It is highly recommended that you self-study this chapter before proceeding with any of the tasks in this assignment. The dataset provided with this assignment contains ordinal features.

Table 1: Table showing the description of features

Feature	Description
temp	Normalized temperature in Celsius
humidity	Normalized humidity
windspeed	Normalized wind speed
season	Encoded as 1 (winter), 2 (spring), 3 (summer), 4 (autumn)
holiday	Binary: 1 if holiday, 0 otherwise
workingday	Binary: 1 if working day, 0 otherwise
hour	Hour of the day (0-23)
bike_rented	Target variable (number of bikes rented)

3 Tasks

3.1 Statistical Learning Approach (40 Marks)

1. Standard Linear Regression (5 Marks) - Implement Ordinary Least Squares (OLS) linear regression and evaluate performance.

2. Ridge Regression (10 Marks) - Apply Ridge regression (L2 regularization). - Tune hyperparameter λ (lambda) using cross-validation.

3. Lasso Regression (10 Marks) - Apply Lasso regression (L1 regularization). - Compare sparsity effects against Ridge.

4. Elastic Net (5 Marks) - Implement Elastic Net, combining L1 and L2 penalties. - Discuss its advantages over Ridge and Lasso.

5. Analysis (10 Marks) - Compare models using RMSE and R^2. - Discuss how regularization affects feature importance.

3.2 Deep Learning Approach (40 Marks)

1. Linear Neural Network (5 Marks) - Implement a single-layer neural network without activation (pure linear regression).

2. Weight Decay (10 Marks) - Add L2 regularization in the neural network (equivalent to Ridge). - Experiment with different weight decay values.

3. Dropout Regularization (10 Marks) - Introduce Dropout (even though it's rare for regression). - Analyze its impact.

4. Feature Engineering & Data Augmentation (5 Marks) - Transform existing features (e.g., polynomial features, interaction terms). - Investigate effects of engineered features.

5. Analysis (10 Marks) - Compare deep learning vs. statistical approaches. - Discuss the pros and cons of neural networks for regression tasks.

3.3 Discussion & Report (20 Marks)

- Provide a critical comparison of the statistical vs. deep learning approaches. You must include at least 4 references and appropriately cited in your report.
- Discuss the role of regularization in both frameworks.
- Reflect on computational efficiency, interpretability, and overfitting risks.

4 Submission Requirements

- Code (Python code only)
- Report (PDF, 4-6 pages, Latex preferred)
- A template has been provided with this specification.
- Plots & Visualizations to compare models

5 Starter code

```
import pandas as pd
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
df = pd.read_csv("bike_rental_data.csv")

# Features and target
X = df.drop(columns=["bikes_rented"])
y = df["bikes_rented"]

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
 random_state=42)

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# --- Statistical Learning ---
# Linear Regression
lin_reg = LinearRegression()
```

```
lin_reg.fit(X_train_scaled, y_train)

# Ridge Regression
ridge = Ridge(alpha=1.0)
ridge.fit(X_train_scaled, y_train)

# Lasso Regression
lasso = Lasso(alpha=0.1)
lasso.fit(X_train_scaled, y_train)

# Elastic Net
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic_net.fit(X_train_scaled, y_train)

# Evaluate models
def evaluate(model, X, y, name):
y_pred = model.predict(X)
print(f"{name}: RMSE={mean_squared_error(y, y_pred, squared=False):.2f},
 R^2={r2_score(y, y_pred):.2f}")

evaluate(lin_reg, X_test_scaled, y_test, "Linear Regression")
evaluate(ridge, X_test_scaled, y_test, "Ridge Regression")
evaluate(lasso, X_test_scaled, y_test, "Lasso Regression")
evaluate(elastic_net, X_test_scaled, y_test, "Elastic Net")

# --- Deep Learning Approach ---
class LinearNN(nn.Module):
def __init__(self, input_dim):
super(LinearNN, self).__init__()
self.linear = nn.Linear(input_dim, 1)  # Single-layer linear model

def forward(self, x):
return self.linear(x)

# Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train_scaled, dtype=torch.float32)
y_train_tensor = torch.tensor(y_train.values, dtype=torch.float32).view(-1, 1)
X_test_tensor = torch.tensor(X_test_scaled, dtype=torch.float32)
y_test_tensor = torch.tensor(y_test.values, dtype=torch.float32).view(-1, 1)

# Initialize model, loss, and optimizer
model = LinearNN(X_train.shape[1])
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01, weight_decay=0.01)
 # L2 regularization (weight decay)

# Train model
epochs = 100
for epoch in range(epochs):
```

```
model.train()
optimizer.zero_grad()
y_pred = model(X_train_tensor)
loss = criterion(y_pred, y_train_tensor)
loss.backward()
optimizer.step()

if epoch % 10 == 0:
print(f"Epoch {epoch}: Loss = {loss.item():.4f}")

# Evaluate on test data
model.eval()
y_pred_tensor = model(X_test_tensor).detach().numpy()
test_rmse = mean_squared_error(y_test, y_pred_tensor, squared=False)
test_r2 = r2_score(y_test, y_pred_tensor)
print(f"Deep Learning Model: RMSE={test_rmse:.2f}, R^2={test_r2:.2f}")
```

References

Géron, A. (2019). *Hands-on machine learning with scikit-learn, keras, and tensorflow* (Second ed.). O'Reilly.