Homework Sheet 5

Group members:
Dhvaniben Jasoliya
Leutrim Uka
Nicola Horst
Taqueer Rumaney
Yuvraj Dhepe

1. **6 Points**
Calculate by hand two iterations of steepest gradient descent with line search for $\frac{1}{2}||X\beta - y||^2$ for $X = (2,1;1,0)$ and $y = (1;1)$ with initial iterate $\beta_0 = (1 \quad 1)^T$.

Rewrite X, y, β_0 using matrix notation:

$$X = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \beta_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

To solve the task, we use the algorithm as defined in the lecture

---

Algorithm 1 Steepest Descent for Least-Squares

---

for $j = 1, \ldots$ **do**
  Compute residual $\mathbf{r}_j = \mathbf{y} - \mathbf{X}\beta_j$
  Determine the SD direction $\mathbf{d}_j = \mathbf{X}^\top \mathbf{r}_j$
  Compute step size $\alpha_j = \frac{\mathbf{r}_j^\top \mathbf{X}\mathbf{d}_j}{\|\mathbf{X}\mathbf{d}_j\|^2}$
  Take the step $\beta_{j+1} = \beta_j + \alpha_j \mathbf{d}_j$
end for

Step 1 $(j=0)$:

$$\rightarrow r_0 = y - X\beta_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -2 \\ 0 \end{bmatrix}$$

$$\rightarrow d_0 = X^T r_0 = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -2 \\ 0 \end{bmatrix} = \begin{bmatrix} -4 \\ -2 \end{bmatrix}$$

$$\rightarrow \alpha_0 = \frac{r_0^T \cdot X \cdot d_0}{||Xd_0||^2} = \frac{20}{116} = 0{,}17$$

$$r_0^T X d_0 = \begin{bmatrix} -2 & 0 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -4 \\ -2 \end{bmatrix} = \begin{bmatrix} -4 & -2 \end{bmatrix} \cdot \begin{bmatrix} -4 \\ -2 \end{bmatrix} = 16 + 4 = 20$$

$$||Xd_0||^2 = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -4 \\ -2 \end{bmatrix} = \begin{bmatrix} -10 \\ -4 \end{bmatrix} = (-10)^2 + (-4)^2 = 100 + 16 = 116$$

$$\rightarrow \beta_1 = \beta_0 + \alpha_0 d_0 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 0{,}17 \cdot \begin{bmatrix} -4 \\ -2 \end{bmatrix} = \begin{bmatrix} 0{,}31 \\ 0{,}66 \end{bmatrix}$$

Step 2 $(j=1)$:

$$\rightarrow r_1 = y - X\beta_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0{,}31 \\ 0{,}66 \end{bmatrix} = \begin{bmatrix} -0{,}28 \\ 0{,}69 \end{bmatrix}$$

$$\rightarrow d_1 = X^T r_1 = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} -0{,}27 \\ 0{,}68 \end{bmatrix} = \begin{bmatrix} 0{,}14 \\ 0{,}28 \end{bmatrix}$$

$$\rightarrow \alpha_1 = \frac{r_1^T X d_1}{||Xd_1||^2} = \frac{0{,}095}{0{,}02} = 4{,}99$$

$$r_1^T X d_1 = \begin{bmatrix} -0{,}28 & 0{,}69 \end{bmatrix} \cdot \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0{,}14 \\ 0{,}28 \end{bmatrix} = \begin{bmatrix} 0{,}14 & -0{,}28 \end{bmatrix} \cdot \begin{bmatrix} 0{,}14 \\ 0{,}28 \end{bmatrix} = 0{,}095$$

$$||Xd_1||^2 = 0{,}02$$

$$\rightarrow \beta_2 = \beta_1 + \alpha_1 d_1 = \begin{bmatrix} 0{,}31 \\ 0{,}66 \end{bmatrix} + 4{,}99 \begin{bmatrix} 0{,}14 \\ 0{,}28 \end{bmatrix} = \begin{bmatrix} 1 \\ -0{,}72 \end{bmatrix}$$

2. **6 points**
Show the solution to the so-called ridge regression is given by

$$\hat{\beta}_{\text{Ridge}} = \arg\min_{\beta} \|y - X\beta\|_2^2 + \lambda \|\beta\|_2^2 = (X^T X + \lambda I_p)^{-1} X^T y \tag{1}$$

To prove the equation above, we have to minimize the following expression:

$$\begin{aligned} L(\beta) &= (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta \\ &= (y^T - \beta^T X^T)(y - X\beta) + \lambda \cdot \beta^T \beta \\ &= y^T y - y^T X\beta - \beta^T X^T y + \beta^T X^T X \beta + \lambda \cdot \beta^T \beta \\ &= \beta^T (X^T X + \lambda I)\beta - 2\beta^T X^T y + y^T y \end{aligned}$$

To find the minimum, we have to derive the expression w.r.t. β:

$$\begin{aligned} \frac{\partial}{\partial \beta} L(\beta) &= \frac{\partial}{\partial \beta}\left[\beta^T (X^T X + \lambda I)\beta - 2\beta^T X^T y + y^T y\right] \\ &= \frac{\partial}{\partial \beta}\left[\beta^T (X^T X + \lambda I)\beta\right] - 2 \cdot \frac{\partial}{\partial \beta} \beta^T X^T y + 0 \\ &= 2(X^T X + \lambda I)\beta - 2X^T y \end{aligned}$$

Set $\frac{\partial}{\partial \beta} L(\beta) = 0$:

$$2(X^T X + \lambda I)\beta - 2X^T y = 0$$

$$\cancel{2}(X^T X + \lambda I)\beta = \cancel{2}X^T y$$

$$(X^T X + \lambda I)\beta = X^T y$$

$$\beta = \frac{X^T y}{X^T X + \lambda I}$$

$$\boxed{\beta = (X^T X + \lambda I)^{-1} X^T y}$$

Task_3

December 20, 2023

0.1 Task 3.1

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
# Group Members: Dhvaniben Jasoliya, Leutrim Uka, Nicola Horst, Tauqeer␣
↪Rumaney, Yuvraj Dhepe
# Data
sq_footage = np.array([1500, 1800, 2200, 1200, 1600])
num_bedrooms = np.array([3, 4, 5, 2, 3])
distance_from_city = np.array([2, 1, 0, 3, 2])
price = np.array([200000, 220000, 250000, 180000, 210000])

# Create the design matrix
X = np.column_stack((np.ones_like(sq_footage), sq_footage, num_bedrooms,␣
↪distance_from_city))
y = price

# Hyperparameter
max_iterations = 3000
epsilon = 1e-6  # Convergence criterion

# Initialize coefficients
np.random.seed(42)  # For reproducibility
beta = np.random.rand(4)

# Perform steepest descent iterations
for iteration in range(max_iterations):
    # Compute residual
    r = y - X @ beta

    # Determine the steepest descent direction
    d = X.T @ r

    # Compute step size
    alpha = np.dot(r.T, X @ d) / np.dot(X @ d, X @ d)

    # Take the step
    beta = beta + alpha * d
```

```
    # Check the convergence criterion
    if np.linalg.norm(d) < epsilon:
        break

# Print the final coefficients
print("Task 1")
print("Intercept (beta0):", beta[0])
print("Coefficient for sq_footage (beta1):", beta[1])
print("Coefficient for num_bedrooms (beta2):", beta[2])
print("Coefficient for distance_from_city (beta3):", beta[3])
```

```
Task 1
Intercept (beta0): 2913.0568348244433
Coefficient for sq_footage (beta1): 111.43911265967715
Coefficient for num_bedrooms (beta2): 429.8411027810578
Coefficient for distance_from_city (beta3): 14134.901023172968
```

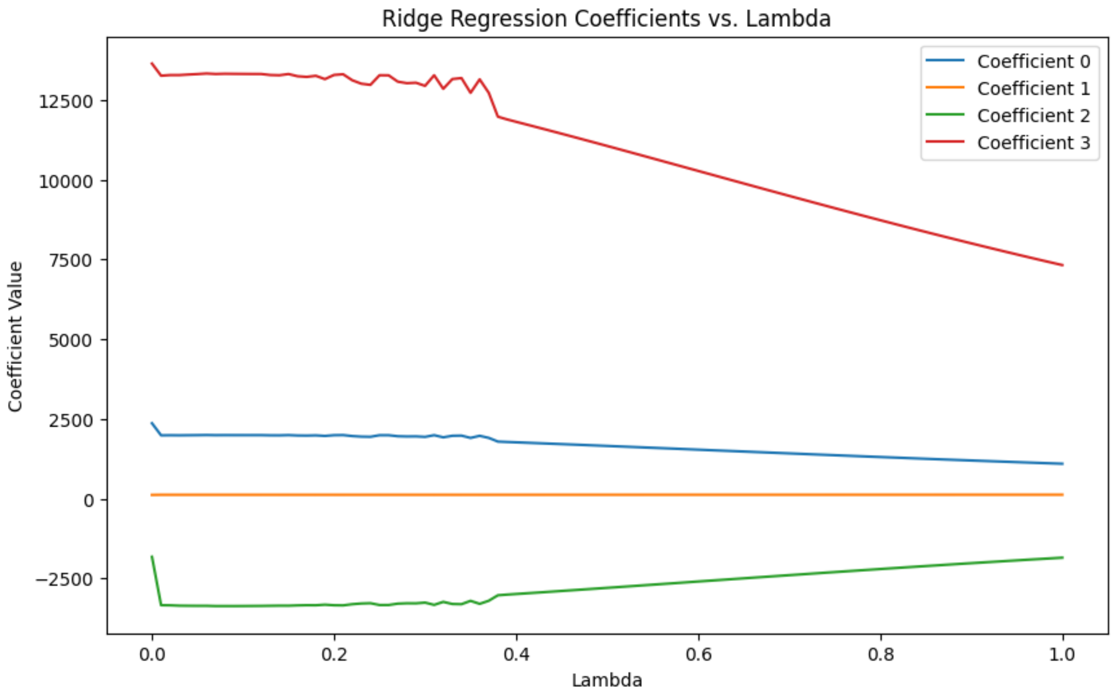
[2]:
```
# Making Prediction on one of the available data point
new_sq_footage = np.array([1500])
new_num_bedrooms = np.array([3])
new_distance_from_city = np.array([2])

# Create the design matrix for new data
X_new = np.column_stack((np.ones_like(new_sq_footage), new_sq_footage,
 ↪new_num_bedrooms, new_distance_from_city))

# Use the final coefficients to make predictions on new data
predictions_new = X_new @ beta

# Print the predictions for new data"
print("Predictions for new data:", predictions_new)
```

```
Predictions for new data: [199631.05117903]
```

0.1.1 Task 3.2

[5]:
```
sq_footage = np.array([1500, 1800, 2200, 1200, 1600])
num_bedrooms = np.array([3, 4, 5, 2, 3])
distance_from_city = np.array([2, 1, 0, 3, 2])
price = np.array([200000, 220000, 250000, 180000, 210000])

# Create the design matrix
X = np.column_stack((np.ones_like(sq_footage), sq_footage, num_bedrooms,
 ↪distance_from_city))
y = price
```

```
# Hyperparameters
max_iterations = 1000
epsilon = 1e-6  # Convergence criterion
learning_rate = 0.01  # Initial learning rate

# Regularization parameter values
lambda_values = np.arange(0, 1.01, 0.01)

# Initialize coefficients
beta_history = np.zeros((len(lambda_values), X.shape[1]))  # Store coefficients␣
 ↪for each lambda

# Ridge regression iterations for different lambda values
for idx, lambda_val in enumerate(lambda_values):
    np.random.seed(42)  # For reproducibility
    beta = np.random.rand(X.shape[1]) # Initialize coefficients

    for iteration in range(max_iterations):
        # Compute residual
        r = y - X @ beta

        # Determine the steepest descent direction with regularization term
        d = X.T @ r + lambda_val * beta

        # Compute step size
        alpha = np.dot(r.T, X @ d) / np.dot(X @ d, X @ d)

        # Take the step
        beta = beta + alpha * d

        # Check the convergence criterion
        if np.linalg.norm(d) < epsilon:
            break

    # Save coefficients for the current lambda
    beta_history[idx, :] = beta

# Plot the coefficients as a function of lambda
plt.figure(figsize=(10, 6))
for i in range(X.shape[1]):
    plt.plot(lambda_values, beta_history[:, i], label=f'Coefficient {i}')

plt.xlabel('Lambda')
plt.ylabel('Coefficient Value')
plt.title('Ridge Regression Coefficients vs. Lambda')
plt.legend()
plt.savefig("Task_3.2_Plot.png", dpi = 150, bbox_inches = 'tight')
```



[4]:
```
# Making Prediction on a data point
new_sq_footage = np.array([1500])
new_num_bedrooms = np.array([3])
new_distance_from_city = np.array([2])
# Create the design matrix for new data
X_new = np.column_stack((np.ones_like(new_sq_footage), new_sq_footage,
 ↪new_num_bedrooms, new_distance_from_city))

# Use the final coefficients to make predictions on new data
predictions_ridge = X_new @ beta

# Print the predictions for new data
print("Predictions for new data using Ridge Regression:", predictions_ridge)
```

```
Predictions for new data using Ridge Regression: [194347.35468185]
```