M4L2 Solutions

January 2024

Exercise A

What is Gradient Descent? Briefly explain its concept.

Solution

Gradient Descent

Gradient descent is a generic optimization algorithm that measures the local gradient of the cost function with regards to θ. It is used to learn a model by minimizing the cost function $J(\theta)$ w.r.t the parameters θ. The cost function is defined as the error between the observed data y and the prediction $\hat{y} = h_\theta(x) = \theta^T x = \theta_0 + \theta_1 x_1$.

During the learning process, we iteratively calculate the gradient of the cost function and update the parameters of the model. The parameters are then updated in the opposite direction of the gradient of $J(\theta)$. If the gradient of $J(\theta)$ is positive, the parameters should be decreased and vice versa until having the smallest possible error of the cost function $J(\theta)$ by following an update rule

$$\theta := \theta - \eta \nabla J(\theta)$$

where η is the learning rate that defines how far to go in each step and ∇ is the gradient of the cost function $J(\theta)$.

Exercise B

For the cost function

$$J = \frac{1}{m} \sum_{i=1}^{m} (y_i - \theta_0 - \theta_i x_i)^2$$

derive the partial derivatives w.r.t to θ_0 and θ_1.

Solution

Partial Derivatives

Partial derivatve of the above cost function is

$$\frac{\partial J}{\partial \theta_0} = \frac{2}{m} \sum_{i=1}^{m} (y_i - \theta_0 - \theta_i x_i) \cdot (-1)$$

$$\frac{\partial J}{\partial \theta_1} = \frac{2}{m} \sum_{i=1}^{m} (y_i - \theta_0 - \theta_i x_i) \cdot (-x_i)$$

Exercise C

Retrieve the gas consumption dataset with tax, income, highway, drivers and gas as features columns. Fit a linear regression and plot the cost function

$$J = \frac{1}{m} \sum_{i=1}^{m} (y_i - \theta_0 - \theta_i x_i)^2$$

This is a univariate problem where your X is the percentage of population driving and y is the gas consumption in million gallons.

Solution

[1]:
```python
# load the required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
# more on this in Python Labs
from sklearn.linear_model import LinearRegression
```

[2]:
```python
df = pd.read_csv('https://raw.githubusercontent.com/kannansingaravelu/datasets/
    ↪main/gas_consumption.csv', names =['tax', 'income', 'highway', 'drivers',␣
    ↪'gas'])
df.head(3)
```

[2]:

	tax	income	highway	drivers	gas
1	9.0	3571	1976	0.525	541
2	9.0	4092	1250	0.572	524
3	9.0	3865	1586	0.580	561

[3]:
```python
# instantiate LR
lr = LinearRegression()
# fir the model
lr.fit(df[['drivers']], df[['gas']])
# retrieve the theta0 and theta1
lr.intercept_, lr.coef_
```

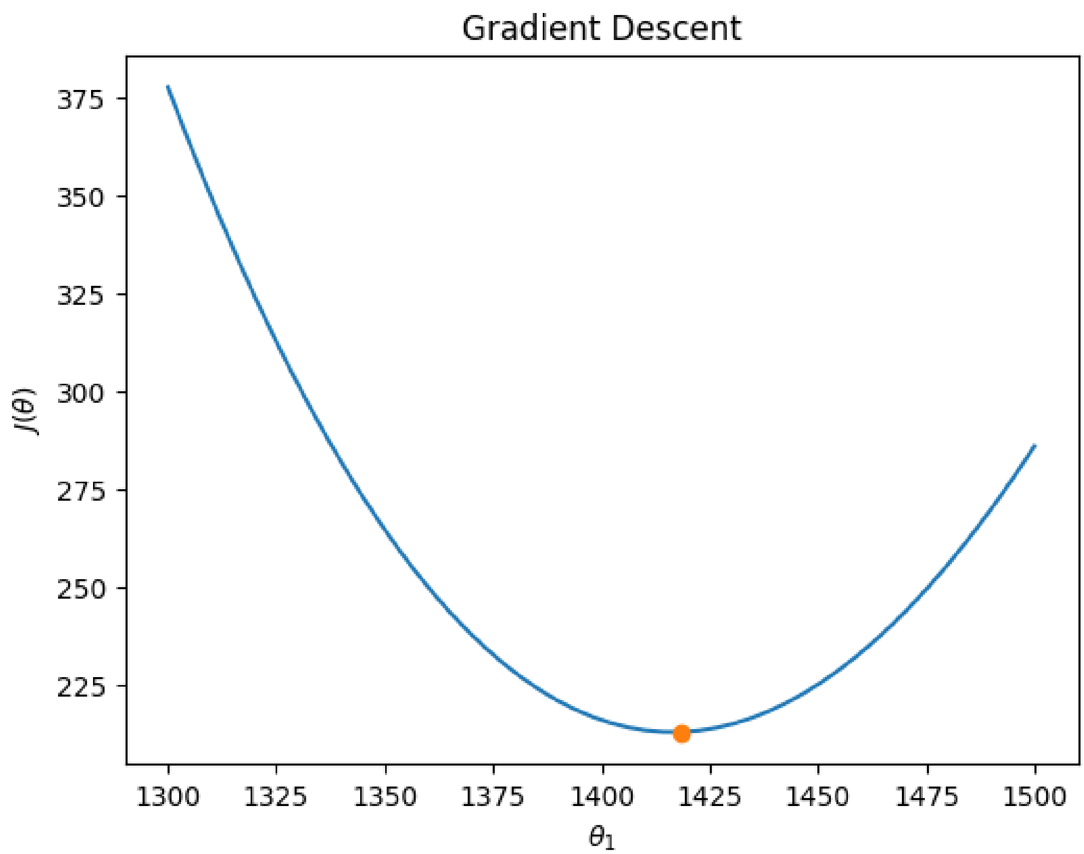
[3]:
```
(array([-227.30911749]), array([[1409.84211133]]))
```

Objective is to minimize the total square error where the residual depends on the model parameters θ_0 and θ_1. Let's now plot the cost function with respect to θ_1.

[4]:
```python
# intercept
theta0 = lr.intercept_
# parameters wrt to x
theta1 = np.linspace(1300, 1500)
# mean square error
```

```
mse = [1/m * ((df['gas'] - (theta0 + m * df['drivers']))**2).sum() for m in␣
 ↪theta1]

plt.plot(theta1, mse)
plt.xlabel(r"${\theta_1}$")
plt.ylabel(r"${J(\theta)}$")
plt.plot(1418.367347, 212.965817, marker="o")        # _cost = pd.
 ↪DataFrame(list(zip(theta1,mse)))
plt.title('Gradient Descent')
plt.show()
```



References

- Python Resources
- Scikit-learn Linear Models

* * *