Uppsala University

Deep learning

Pre-course assignment

Tong You

January 22, 2021

Contents

1 Exercise 1.1

First, we note that the cost J is a function of z_i. It is also clear that the z_i are functions of the weights w_j and the bias b. Using the chain rule for the given linear regression model we calculate and get:

$$\frac{\mathrm{d}J}{\mathrm{d}b} = \frac{\partial J}{\partial z_i}\frac{\mathrm{d}z_i}{\mathrm{d}b} \tag{1}$$

and:

$$\frac{\mathrm{d}J}{\mathrm{d}w_j} = \frac{\partial J}{\partial z_i}\frac{\mathrm{d}z_i}{\mathrm{d}w_j}. \tag{2}$$

2 Exercise 1.2

The expressions that were provided in 1 and 2 have to be expanded out in order to write out the update equations for the gradient descent algorithm in Exercise 1.3. First write out the given cost J in terms of y_i and z_i:

$$J = \frac{1}{n}\sum_{i=1}^{n} L_i = \frac{1}{n}\sum_{i=1}^{n}(y_i^2 - 2y_i z_i + z_i^2). \tag{3}$$

Then the partial derivative with respect to z_i is easy to determine:

$$\frac{\partial J}{\partial z_i} = \frac{1}{n}\sum_{i=1}^{n}(-2y_i + 2z_i) = \frac{2}{n}\sum_{i=1}^{n}(z_i - y_i) \tag{4}$$

Finally, given the equation for the z_i:

$$z_i = \sum_{j=1}^{p} w_j x_{ij} + b. \tag{5}$$

The equations for the derivatives of the z_i with respect to the w_j and b is:

$$\frac{\mathrm{d}z_i}{\mathrm{d}b} = 1 \tag{6}$$

$$\frac{\mathrm{d}z_i}{\mathrm{d}w_j} = x_{ij}. \tag{7}$$

3 Exercise 1.3

Armed with the above, write out equations 1 and 2 fully (using equations 6 and 7):

$$\frac{\mathrm{d}J}{\mathrm{d}b} = \frac{2}{n}\sum_{i=1}^{n}(z_i - y_i) \tag{8}$$

$$\frac{\mathrm{d}J}{\mathrm{d}w_j} = \frac{2}{n}\sum_{i=1}^{n}(z_i - y_i)x_{ij}. \tag{9}$$

All the previous equations are used to implement the gradient descent algorithm for the linear regression model, given below. The provided loading function was slightly altered, and is therefore attached in the Appendix as a reference.

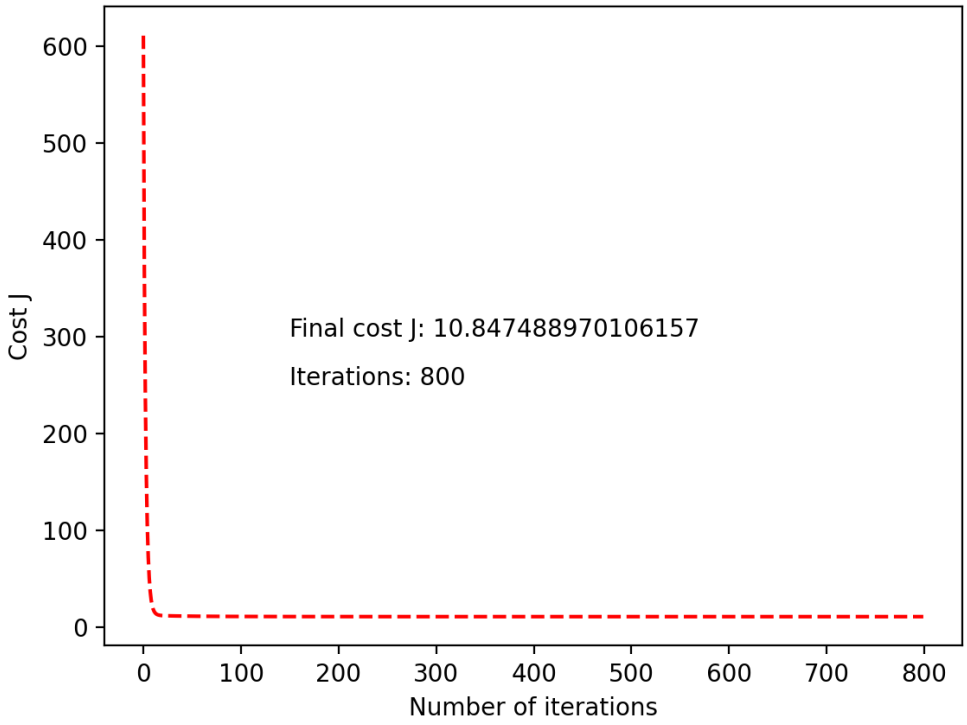
3.1 `linear_regression.py`

```
from load_auto import load_auto
from matplotlib import pyplot as plt
import numpy as np

xtrain_all, xtrain_horsepower, ytrain = load_auto()

# Preprocessing training data
training_all_mean = np.mean(xtrain_all, axis = 0)
training_hp_mean = np.mean(xtrain_horsepower, axis = 0)

training_all_std = np.std(xtrain_all, axis = 0)
training_hp_std = np.std(xtrain_horsepower, axis = 0)

xtn_all = (xtrain_all - training_all_mean) / training_all_std
xtn_hp = (xtrain_horsepower - training_hp_mean) / training_hp_std

def init_params(xt):
    n_feat = xt.shape[1]
    w = np.zeros((1, n_feat))
    b = np.zeros(1)
```

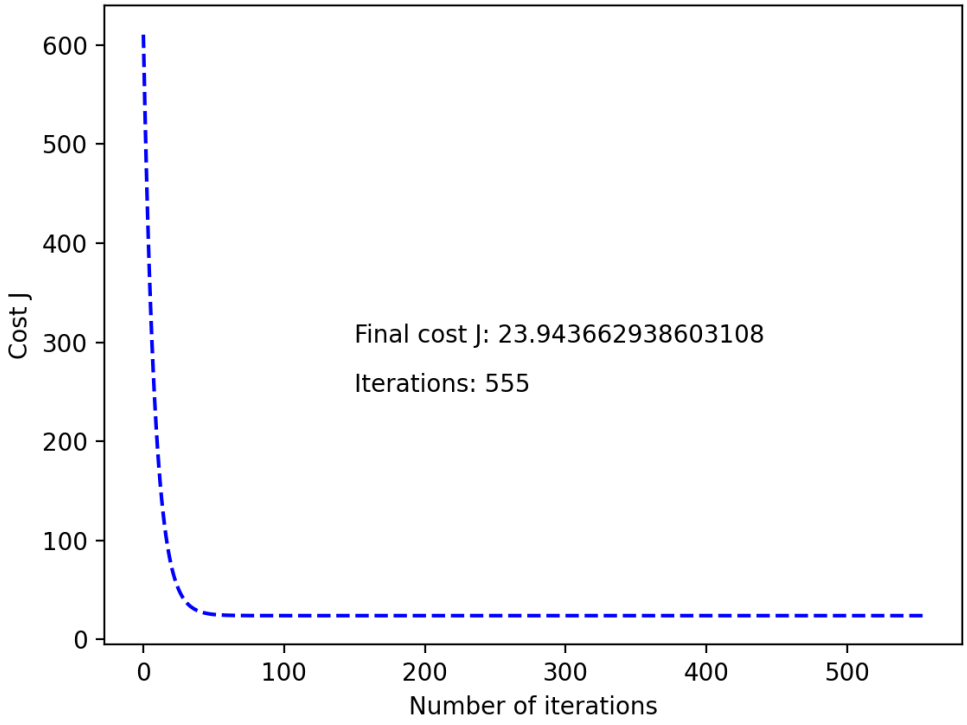
```
    return w, b

def gradient_descent(xtrain, ytrain, learning_rate, maxit):

    w, b = init_params(xtrain)
    wp = 0
    bp = 0
    zi = np.zeros(shape = (xtrain.shape[0], 1))
    J = []
    it = 0
    n = ytrain.shape[0]
    nw = 1
    nb = 1

    while (nw > 0 or nb > 0) and it != maxit:
        wp = w
        bp = b
        zi = np.sum(w * xtrain , axis = 1, keepdims = True) + b
        dJdb = (2/n) * np.sum(zi - ytrain, axis = 0, keepdims = True)
        dJdw = (2/n) * np.sum((zi - ytrain) * xtrain, axis = 0, keepdims = True)

        w = w - learning_rate * dJdw
        b = b - learning_rate * dJdb

        nw = np.sqrt(np.linalg.norm(w - wp))
        nb = np.sqrt(np.linalg.norm(b - bp))

        it += 1
        J.append((1/n) * np.sum((ytrain -  zi) ** 2))

    return J, it, w, b , zi
```

4 Exercise 1.4

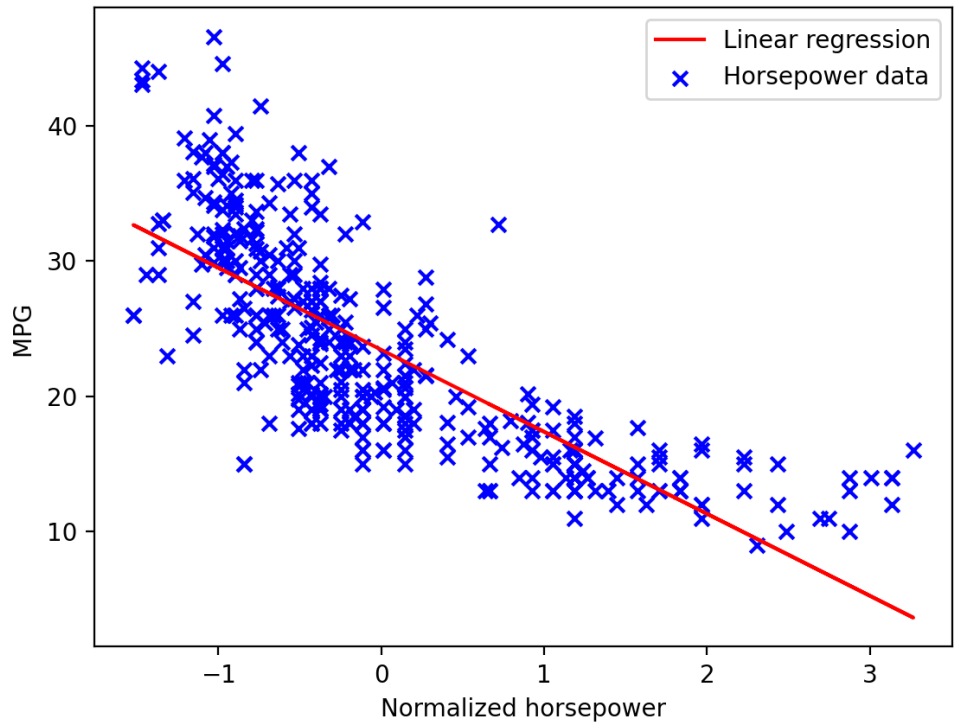
4.1 All normalized input data



4.2 Only normalized horsepower



4.3 Scatter plot of horsepower-only with linear regression



The code used to generate the plots of the cost J versus the number of iterations and the scatter plot is given below.

```
J_all , it_all, w_all, b_all, z_all = gradient_descent(xtn_all, ytrain, 0.1,
maxit = 800)
plt.figure(1)
plt.plot(np.arange(0,it_all), J_all, 'r--')
plt.annotate("Final cost J: "+str(J_all[len(J_all) - 1]), xy = (150,300))
plt.annotate("Iterations: "+str(it_all), xy = (150,250))
plt.xlabel('Number of iterations')
plt.ylabel('Cost J')

J_hp , it_hp, w_hp, b_hp, z_hp = gradient_descent(xtn_hp, ytrain, 0.03,
maxit = 800)
```

```
plt.figure(2)
plt.plot(np.arange(0,it_hp), J_hp, 'b--')
plt.annotate("Final cost J: "+str(J_hp[len(J_hp) - 1]), xy = (150,300))
plt.annotate("Iterations: "+str(it_hp), xy = (150,250))
plt.xlabel('Number of iterations')
plt.ylabel('Cost J')

plt.figure(3)
plt.scatter(xtn_hp, ytrain, marker = "x", c = "blue")
plt.plot(xtn_hp, z_hp, 'r')
plt.legend(["Linear regression", "Horsepower data"])
plt.xlabel('Normalized horsepower')
plt.ylabel('MPG ')
plt.show()
```

5 Appendix

5.1 load_auto.py

```
import numpy as np
import pandas as pd

def load_auto():

    # import data
        Auto = pd.read_csv('/Users/tongyou/Desktop/Deep learning course/Auto.csv'
        ,na_values='?', dtype={'ID': str}).dropna().reset_index()

        # Extract relevant data features
        X_train = Auto[['cylinders','displacement','horsepower','weight',
        'acceleration','year','origin']].values
        X_train_horsepower = Auto[['horsepower']].values
        Y_train = Auto[['mpg']].values

        return X_train, X_train_horsepower, Y_train
```