Deep Learning for Image Analysis
Assignment 1

1MD120 61612 VT2024

Georgios Tsouderos

April 2024

Contents

1 Exercise 1. Partial derivatives

Given a fixed dataset $\{x_i, y_i\}_{i=1}^n$, based on the model, derive expressions for

$$\frac{\partial J}{\partial b} \text{ and } \frac{\partial J}{\partial w_j}$$

expressed in terms of $\frac{\partial J}{\partial z_i}$, $\frac{\partial z_i}{\partial b}$ and $\frac{\partial z_i}{\partial w_j}$ only.

1.1 Solution

Our model consists of the following equations:

$$z_i = \sum_{j=1}^{p} w_j x_{i_j} + b = \mathbf{w}^{\mathrm{T}}\mathbf{x_i} + b \tag{1}$$

where the MSE error for a single point is given by:

$$L_i = (y_i - z_i)^2 \tag{2}$$

and the cost J is the sum of L_i over all the training data:

$$J = \frac{1}{n}\sum_{i=1}^{n} L_i \tag{3}$$

To derive the J function over b, we have to use the chain rule. J is a function of L_i and L_i is a function of z_i and z_i is a function of b. So taking into account the above, we get:

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{n} \frac{\partial J}{\partial L_i}\frac{\partial L_i}{\partial z_i}\frac{\partial z_i}{\partial b} \tag{4}$$

To make the calculations simpler, we can substitute L_i to equation (3) and thus equation (4) becomes:

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{n} \frac{\partial J}{\partial z_i}\frac{\partial z_i}{\partial b} \tag{5}$$

Let's calculate each of the partial derivatives:

$$\frac{\partial J}{\partial L_i} = \frac{1}{n} \tag{4a}$$

$$\frac{\partial L_i}{\partial z_i} = -2(y_i - z_i) \tag{4b}$$

$$\frac{\partial z_i}{\partial b} = 1 \tag{4c}$$

So, multiplying the partial derivatives, we get :

$$\frac{\partial J}{\partial b} = -\frac{2}{n}(y_i - z_i) \tag{6}$$

Now let's derive the J function over w_j using the chain rule. J is a function of z_i and z_i is a function of w_j. As a result, we get:

$$\frac{\partial J}{\partial w_j} = \sum_{i=1}^{n} \frac{\partial J}{\partial z_i}\frac{\partial z_i}{\partial w_j} \tag{7}$$

2 Exercise 2. Derivations

Based on the model described in equations $(1), (2), (3)$ derive expressions for (the above used variables):

$$\frac{\partial J}{\partial z_i}, \frac{\partial z_i}{\partial b} \text{ and } \frac{\partial z_i}{\partial w_j}$$

2.1 Solution

To calculate $\frac{\partial J}{\partial z_i}$, we will again use the chain rule since J is a function of L_i and L_i is a function of z_i. Thus, we get:

$$\frac{\partial J}{\partial z_i} = \frac{\partial J}{\partial L_i}\frac{\partial L_i}{\partial z_i} \tag{8}$$

From equations $(4a), (4b)$ we get the result:

$$\frac{\partial J}{\partial z_i} = -\frac{2}{n}(y_i - z_i) \tag{9}$$

Again, from equation $(4c)$, we have that:

$$\frac{\partial z_i}{\partial b} = 1 \tag{10}$$

Now, let's calculate $\frac{\partial z_i}{\partial w_j}$:

$$\frac{\partial z_i}{\partial w_j} = \frac{\partial(\sum_{j=1}^{p} w_j x_{i_j} + b)}{\partial w_j} = \frac{\partial(w_1 x_{i_1} + .. + w_j x_{i_j} + ... + w_p x_{i_p} + b)}{\partial w_j} = x_{i_j} \tag{11}$$

3 Exercise 4. Evaluate your linear regression model

3.1 Loss of linear regression models

For the first classifier, taking only the horsepower as the input, with learning rate $\alpha = 0.01$ and with 2000 iterations, we achieve a total loss of $J = 23.9436$.

For the second classifier, with the same hyperparameters, we achieve a total loss of $J = 10.8810$.

3.2 Mathematical expressions of the two models

The first classifier is trained with only the horsepower as input. Thus, the input variable has a shape of (392,1). As a result, the output is a scalar function of a single variable $z(x)$, and its mathematical expression is given by:

$$z(x) = -6.1 * x + 23.45$$

resulting in an output of shape (392,1), which has the same shape as $y\_\ train$

For the second classifier, we take the whole dataset X, except the feature name, resulting in the shape (392,7). Thus, $z(X)$ is a scalar function of a vector variable and its mathematical expression is given by:

$$z(\mathbf{X}) = \mathbf{X} * [-0.51, 1.3, -0.7, -5.1, 0.12, 2.7, 1.1]^{\mathrm{T}} + 23.45$$

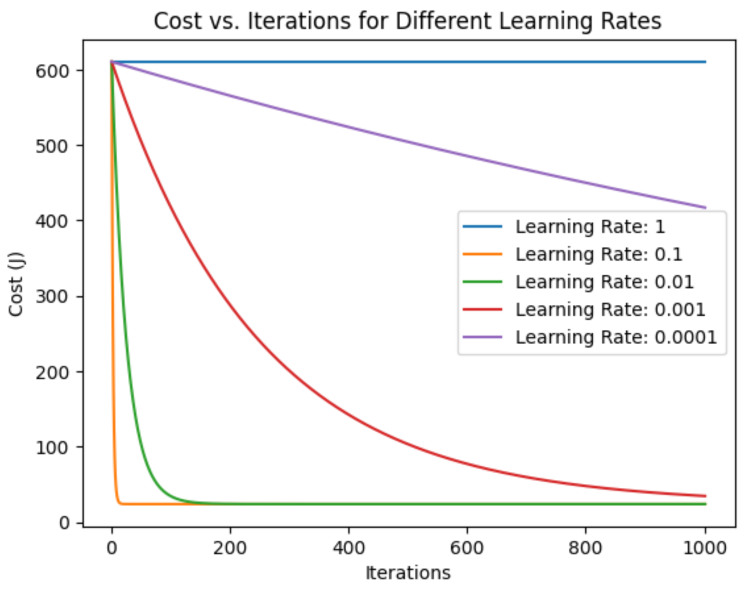
Since X is of shape (392,7) and $w^{\mathbf{T}}$ of shape (7,1), the resulting $z(X)$ is of shape (392,1) which is what we desire.

In the second classifier, the feature that carries the highest weight is the horsepower. Since we want to predict the miles per gallon a car is going to achieve, horsepower does indeed logically play a big part in that. Furthermore, the greater the horsepower of a vehicle is, the less miles per gallon it will achieve, justifying in that way the negative sign in the weight vector.

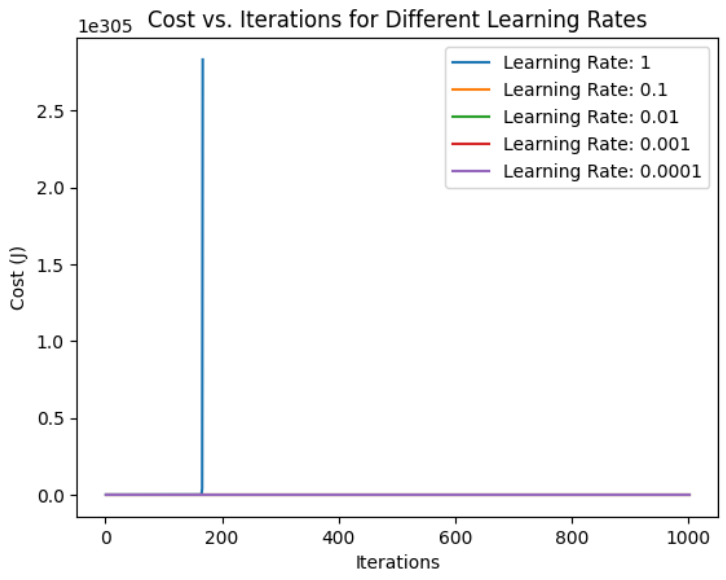
3.3 Plots of Cost vs Iterations for ranging learning rates

To study the relationship between the learning rate and the number of iterations we design plots portraying the cost J versus 1,000 iterations for learning rates [1, 1e-1,1e-2, 1e-3, 1e-4]

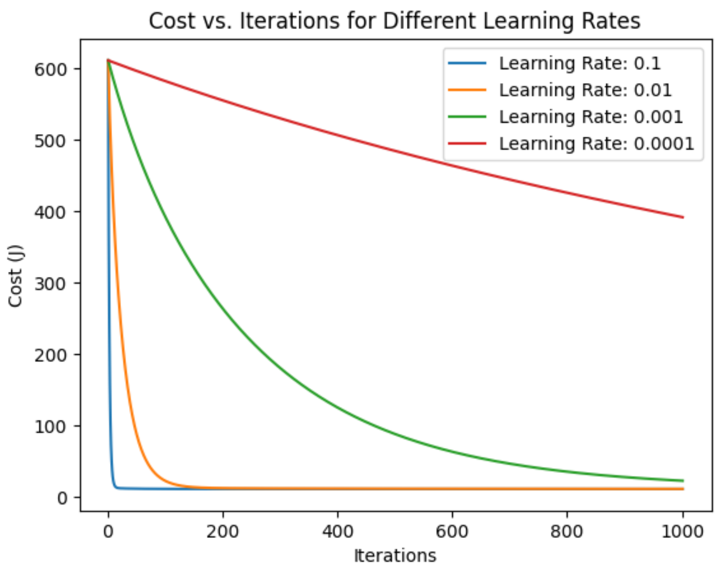
For the first classifier (only horsepower as input) we get:



Accordingly, the plot for the second classifier is the following:



We can observe that for the learning rate $\alpha = 1$, the cost goes to infinity. Thus, we remove it to get a better view of the rest:



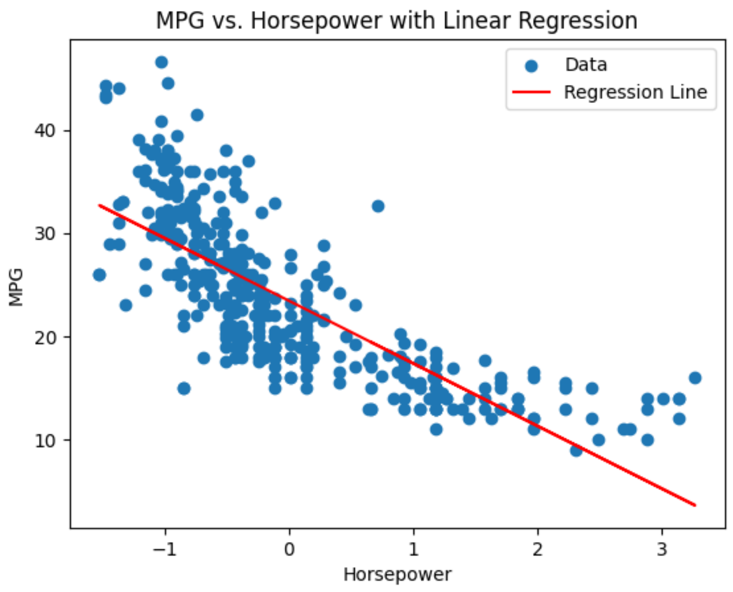
The main concept of linear regression is to minimize the model's loss function. In order to do that, we calculate the partial derivative of the loss function with regard to the model parameters. In doing so, we calculate the gradient of the loss function and by moving towards the negative side of the gradient, we get closer to the minimum. The amount of "distance" we cover when taking the step can be adjusted by a hyper-parameter, called learning rate. The learning rate is a scalar value typically in $[1, -\infty)$ which is multiplied by the step value and acts as an adjustable parameter. As we can see from the graphs, with a big learning rate (ex. $\alpha = 1$), we run the risk of oscillating around the minimum or suffer from exploding losses. With a very small learning rate (ex. $\alpha = 1e - 4$), the model is learning but the rate with which it does is very slow and as a result more iterations may be needed to find the minimum. Another risk is getting stuck on a local minima, if multiple exist.

3.4 Experiments without normalizing the input

When trying to fit the linear regression model without normalizing the input data, the loss function goes to ∞. When dealing with unnormalized data, due to the MSE error, which squares the difference between the true and the predicted value, exploding or vanishing gradients can appear, like in our case. Generally, if there is a huge scale difference between the features, convergence to the minimum value may also need more iterations. In addition, there is a bigger sensitivity to outliers since, due to the difference in scale of our data, the regression line can be skewed to better fit features which don't play an important role.

3.5 Scatter plot of regression model

We can visualize how well our model performs from the following scatter plot:



We can clearly observe that less horsepower is indeed indicative of more miles per gallon, though it should be emphasized that it is not a causality relationship even though they seem correlated. Furthermore, we can see that our model has done a good job fitting the regression line. Probably however, a non linear classifier would have done a better job since there is a distinct curve in the data.