

Deep Learning for Image Analysis 2023: Assignment 1

Adapted from Niklas Wahlström

March 20, 2023

Due: March 31, 2023

In this first assignment, you will implement a linear regression model by using gradient descent.¹ In the next assignment, we will extend this code to an implementation of a full neural network. Hence, you will benefit twice by keeping your code tidy, well structured, and suitably subdivided into functions, since the same code will provide a good start for the second hand-in assignment.

1 Linear regression with gradient descent

Consider a dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$. Each input is a vector $\mathbf{x}_i = [x_{i1}, \dots, x_{ip}]^T$ and each output $y_i \in \mathbb{R}$ is a scalar. We want to find a model for the output using linear regression. For one data point $i \in \{1, \dots, n\}$, the linear regression model can be expressed as

$$z_i = \sum_{j=1}^p w_j x_{ij} + b = \mathbf{w}^T \mathbf{x}_i + b, \quad (1a)$$

where the weight vector $\mathbf{w} = [w_1, \dots, w_p]^T$ and the offset b are the parameters of the model. The cost J is computed by summing the following loss L_i over all training data points

$$L_i = (y_i - z_i)^2, \quad (1b)$$

$$J = \frac{1}{n} \sum_{i=1}^n L_i. \quad (1c)$$

To train this model with gradient descent, we need access to the gradient of the cost function with respect to the model parameters, i.e. the partial derivatives $\frac{\partial J}{\partial w_1}, \dots, \frac{\partial J}{\partial w_p}$, and $\frac{\partial J}{\partial b}$, which you will derive below.

Exercise 1.1 Given a fixed dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$, based on the model in (1), derive expressions for

$$\frac{\partial J}{\partial b} \quad \text{and} \quad \frac{\partial J}{\partial w_j} \quad \text{expressed in terms of} \quad \frac{\partial J}{\partial z_i}, \quad \frac{\partial z_i}{\partial b}, \quad \text{and} \quad \frac{\partial z_i}{\partial w_j}. \quad (2)$$

Exercise 1.2 Based on the model in (1), derive expressions for

$$\frac{\partial J}{\partial z_i}, \quad \frac{\partial z_i}{\partial b}, \quad \text{and} \quad \frac{\partial z_i}{\partial w_j}. \quad (3)$$

Exercise 1.3 Implement in python, using `numpy`, a gradient descent algorithm (Lindholm et al. 2021, Chp 5.4) that minimizes the cost J with respect to the parameters \mathbf{w}, b , for the training data set $\{\mathbf{x}_i, y_i\}_{i=1}^n$. It should involve the following functionalities:

- **Initialize.** Write a function `initialize_parameters` that initializes the parameters \mathbf{w}, b . For linear regression it is sufficient to initialize all parameters with zeros.
- **Forward propagation.** Write a function `model_forward` that evaluates the model shown in Equation 1a.

¹A linear regression problem can be also solved analytically using the normal equations. However, for the purpose of it being an initial code that you can extend in following assignments, you should solve it here using gradient descent.

- **Compute cost.** Write a function `compute_cost` that computes the cost J according to the mathematical expression in Equation 1c.
- **Compute gradients.** Write a function `model_backward` that computes the gradient of the cost function with respect to all parameters $\frac{\partial J}{\partial b}$ and $\frac{\partial J}{\partial w}$. For this, you need the mathematical expression in Equations 2 and 3.
- **Take a step.** Write a function `update_parameters` that updates the parameters based on provided gradients. Scale the update step by the learning rate α , which will determine the size of the steps you take in each iteration.
- **Predict.** Write a function `predict` which, using the trained model, predicts z_i based on the corresponding input \mathbf{x}_i . This function will in this task more or less only be a wrapper for `model_forward`.

Your final function `train_linear_model` should minimize the loss on the provided data using gradient descent. This function will iteratively call the above functions that you have defined. It should have as inputs the training data x_i , their labels y_i , the number of iterations and the learning rate α . It should return the optimized parameters of the model. It is also recommended that you save and return the cost at each iteration in a vector.

Exercise 1.4 Evaluate the model on the provided `Auto.csv` dataset. You may use the supplied Python code `load_auto.py` for loading the data if you wish. Treat `mpg` (miles per gallon) as your output y_i and consider two different choices for your input \mathbf{x}_i : (i) only the input variable `horsepower`, and (ii) all input variables except `name`.

- **Pre-processing.** As pre-processing steps to construct your dataset $\{\mathbf{x}_i, y_i\}_{i=1}^n$ you need to
 1. Remove all rows with NA values².
 2. Normalize the input variables, for example by subtracting the mean and dividing with the standard deviation.

$$\tilde{x}_{ij} = \frac{x_{ij} - \bar{x}_j}{s_j}$$

- **Evaluate.** Now use the code developed in Exercise 1.3 (and `matplotlib` for visualization) to
 1. Fit two linear regression models for the two choices of inputs ((i) only `horsepower` and (ii) all except `name`). With a correct implementation, the final cost for the two models should be less than 25 and less than 12, respectively.
 2. Write out the mathematical expressions of your two models (with sensible number of decimals). For the second model, which of the input variables has the highest weight in the expression? Does this seem reasonable?
 3. To study the relationship between learning rate and number of iterations, produce a plot of the cost J versus 1,000 iterations for learning rates $[1, 1e-1, 1e-2, 1e-3, 1e-4]$ for the two models. Present the results as two plots, as shown in Figure 1. Provide a high level interpretation of what you observe in the plots.

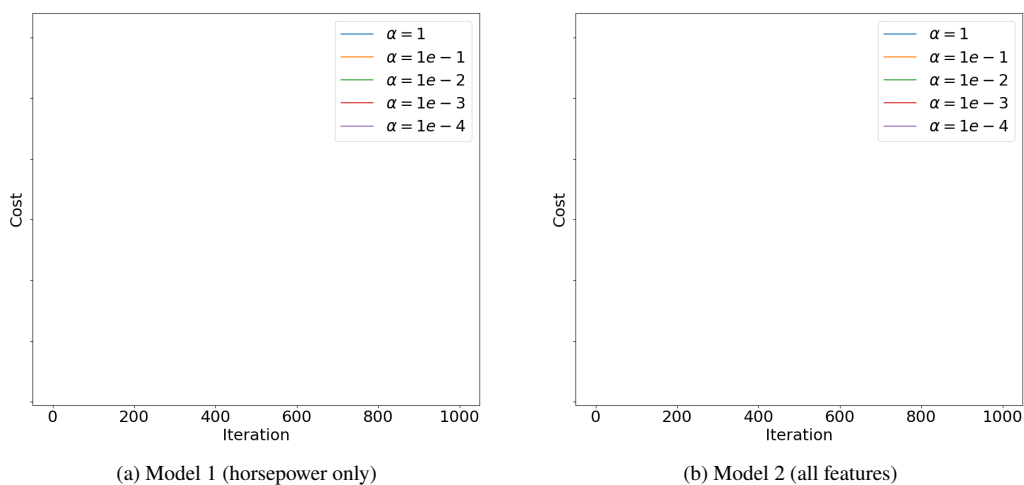


Figure 1: Cost vs iterations for different learning rates α . The actual lines and the y-axis values have been omitted.

²Already done if you use the provided loading code.

4. Try repeating any of the previous experiments without normalizing the input. What happens then?
5. For the **horsepower**-only model, produce a scatter plot with **horsepower** vs **mpg** together with a drawn line representing the linear regression model which you computed. Is it a good fit?

Use the 'Assignment template' (on the Course homepage) and submit a pdf with answers to Exercise 1.1, Exercise 1.2 and Exercise 1.4. For Exercise 1.3, submit a zip file with the code.

Exercise 1.5 *Voluntary bonus task: Add a softmax function to the output to handle this as a classification problem. Try to predict the origin of the car based on the other variables.*

A The auto.csv data

Origin: The dataset was used in the 1983 American Statistical Association Exposition.

Description: Gas mileage, horsepower, and other information for 392 vehicles.

Format: A data frame with 392 observations on the following 9 variables.

mpg: miles per gallon

cylinders: Number of cylinders between 4 and 8

displacement: Engine displacement (cu. inches)

horsepower: Engine horsepower

weight: Vehicle weight (lbs.)

acceleration: Time to accelerate from 0 to 60 mph (sec.)

year: Model year (modulo 100)

origin: Origin of car (1. American, 2. European, 3. Japanese)

name: Vehicle name

References

Lindholm, Andreas, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön (2021). *Supervised Machine Learning*. URL: <https://smlbook.org>.