TC3020 Machine Learning
Teamwork Assignment 1: Linear Regression and Gradient Descent

Dr. Rafael Pérez Torres

August 31st, 2021

Abstract

In this assignment, you will implement the Gradient Descent and Linear Regression algorithms.

1 Description

Overall goal: Implement the Linear Regression (LR) and Gradient Descent algorithms studied during our previous sessions.

You must not employ specialized libraries (`sklearn` or similar) to produce your solution; it is ok to employ them to compare though. You have access to a partially implemented source code for the multivariate LR and Gradient Descent algorithms. Take your time to study the code and understand its structure. There are four main elements:

- A launcher (`launcher.py`) that creates some instances of the LR and runs them. It defines the hyperparameters learning rate and epochs.
- A LinearRegressor class (`LinearRegressor.py`) that implements the core of the LR algorithm. **You will be working in this file**.
- A prediction script (`prediction.py`) that helps to transform samples to be predicted by a LR that uses a mean-normalized dataset. **You will be working in this file**.
- A utils script (`utils.py`) that defines some functions to read the dataset, plot results, etc.

2 Implementation

- Locate the TODO comments in the `LinearRegressor.py` and (`prediction.py`) files and implement the required functionality.
- Pay attention to the dimensions of the parameters employed in the functions, so that you can be sure your implementation is returning arrays/values with the proper dimensions..
- Debug as much as you can to understand the code flow and identify issues.
- **Your implementation must be for multiple variables, vectorized**

See that in the `launcher.py` we are running two LinearRegressors in the dataset, one without and one with mean normalization over the dataset. Play around with the values of hyperparameters and see how the LinearRegressor struggles when mean normalization is not performed.

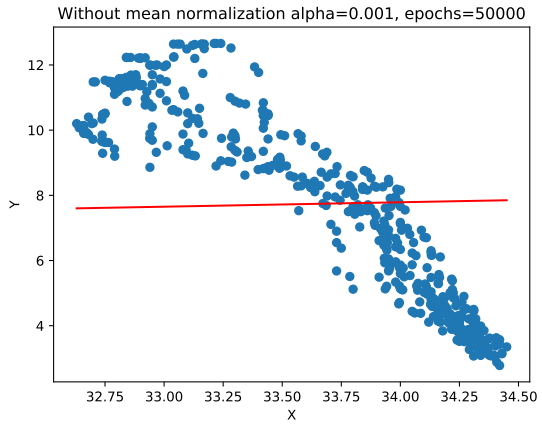
If you don't modify the seed used by `np.random` and the hyperparameter values (`epochs = 50000, alpha=0.001`) you shall see the following values as outputs:

- Without performing mean normalization:

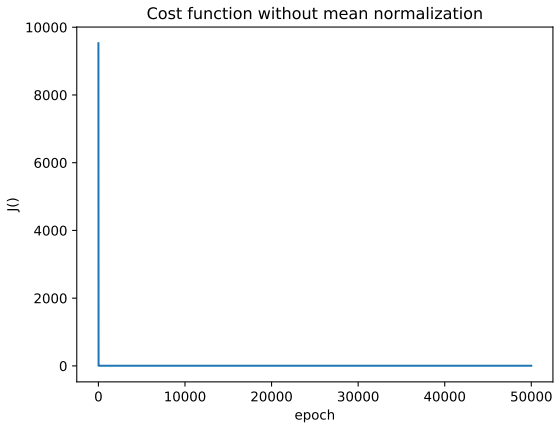
- $\theta = [[3.15247316, 0.13632909]]$
- $J(\theta) = 4.4255737381883975$.

- When performing mean normalization:
  - $\theta = [[6.33826530e-16, -8.67379234e-01]]$
  - $J(\theta) = 0.006513218372806775$.

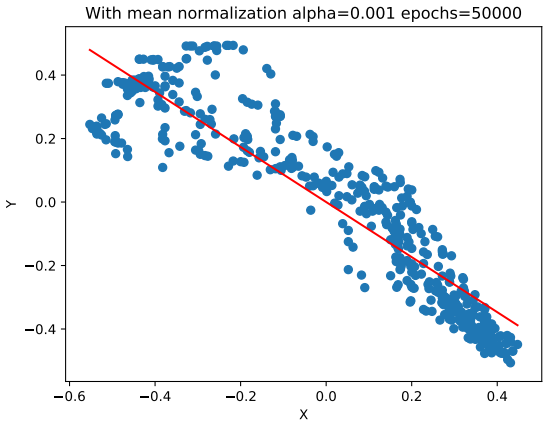
The predictions for the $[30, 31, 33.5]$ samples shall be $[24.90250228, 20.19387216, 8.42229684]$.
The companion plots would be as shown in Fig. 1.



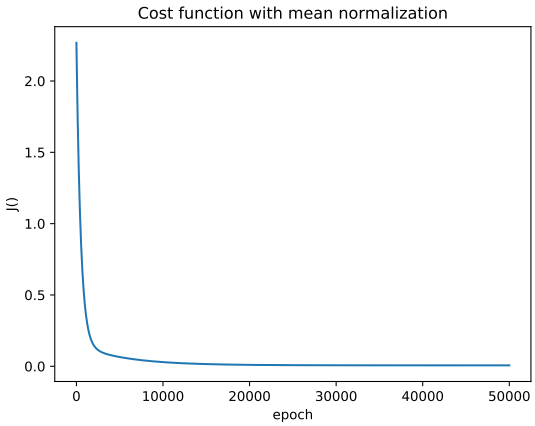
(a) LR without mean normalization



(b) LR cost function without mean normalization



(c) LR with mean normalization



(d) LR cost function with mean normalization

Figure 1: Expected results for the LR implementation

3 Further work

Prepare a short report (3 text pages top, could be more with cover, figures), where you present your findings when playing around with values for the hyperparameters.

- You must elaborate on what happens when you set the alpha value (learning rate) large vs small.
- Cases where the code fails (if any).

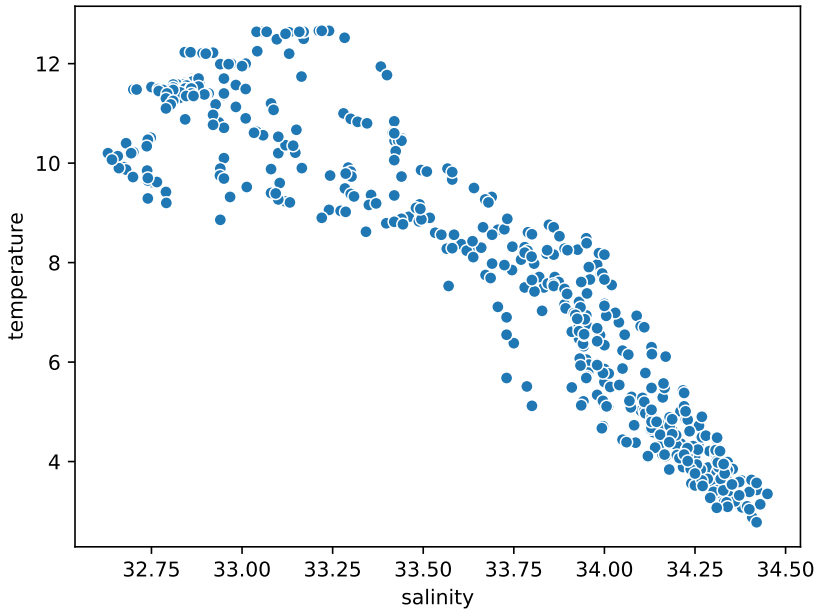


Figure 2: Salinity and temperature features in the *bottle* dataset by CalCOFI.

- The impact of changing the number of iterations.
- How GD is influenced when mean normalization over the input dataset features is/is not performed.

4 Your dataset

A deliberately selected portion of the *bottle* dataset from the CalCOFI organisation (California Cooperative Oceanic Fisheries Investigations) is provided to you[1]. *Since 1949, hydrographic and biological data of the California Current System have been collected on CalCOFI cruises. The 70+ year hydrographic time-series includes temperature, salinity, oxygen and phosphate observations*, among others.

Originally, the dataset included 864862 rows and 74 columns, but it has been reduced to 500 rows and 2 columns. The reduced dataset is already sanitised so you can start using it right away.

We are focusing on the temperature and salinity properties, both given as numbers. We can describe this use case as follows: can we predict the temperature of the water if we are given the salinity level? Have a look at Figure 2 to get an idea of what your answer could be.

In your implementation, when you load the dataset, think of salinity as the input set X and consider temperature as the property y you want to predict.

5 Notes

- Your code should be ready to be run by just executing the `launcher.py` script.
- Include everything (even datasets) in your submitted code so that it can be executed straightforwardly.
- Do employ relative paths in your code (don't make use of absolute paths otherwise the datasets might not be found).
- **It will be penalized if your code can't be run directly.**

[1] `https://calcofi.org/ccdata/database.html`

6 Deliverables

- Your implementation as a zip file with the source code.
  - Name your file as `A1_A2_A3_A4-HW-01.zip` where A1, A2, A3, A4 are the reg number (matrícula) of students.
  - Include the names of team members and student numbers as comments.
- Your report as a PDF. Include a cover for your report file.

A single submission per team is enough, there is no need for submitting more than once. Feedback (if any) will be given over Canvas to the student that uploads the assignment on behalf of the team.