

Codelet 2: Linear Regression using PyTorch

The due date for this codelet is **Wednesday, Feb 5 at 11:59PM**.

Introduction

The aim of this codelet is to build your PyTorch skills via the concrete implementation of parts of linear regression. You should draw on Chapter 3 of the textbook and the PyTorch documentation.

Important: Only use PyTorch and in-built Python. The use of any other libraries, including the books d2l library results in an automatic unsatisfactory grade for this assignment. A core goal of this class is to build your competencies as a machine learning engineer. I want to minimize abstractions from other libraries so that you build these skills.

Outline

- ☐ Grading
- ☐ Linear Regression
 - ☐ Model
 - ☐ Training
 - ☐ Reflection

Your assignment

Your task is to:

1. Download `codelet2.zip` from the course website and open it. You will find these instructions and `codelet2.py` which has scaffolding for you.
2. Complete each of the three broad tasks below in `codelet1.py` and include a file called `codelet2.pdf` with your answers to the written questions.

Grading

When assessing your work, satisfactory achievement is demonstrated, in part, by:

- Simple and clear code
- Code passes relevant tests
- Explanation extends beyond repeating the textbook/material online
- Response are concrete and clearly demonstrate an understanding of the concepts

Linear Regression

You will complete an implementation of linear regression trained using gradient descent. You should build on `codelet2.py`. You have two tasks:

1. Complete the model class
2. Implement mini-batch gradient descent

Linear Regression Model

The basic structure of your model is provided in the class `LinearRegression`. Your task is to complete this function with any **relevant** methods. Do not add methods that serve no substantive purpose in your code or are not used (that is draw from the book with care). For example, you should not add a plotting method. Note, the loss function is correctly implemented and should not be modified. In your code, you must use PyTorch's Linear layer to handle your model's weights (and compute your model's output). Please note the introduction of this codelet. You may only use PyTorch in completing your implementation.

In addition to completing your model, you must write code that returns the error in your model's estimation of **w** and **b**. See below for the formatting (it draws heavily on the example provided in the book).

To evidence completion of this task, and to check your own progress, please train your model using the **train** method. The **train** method should work without any tweaks. Do not modify or otherwise change that function. You should add to the pdf accompanying your code a screenshot of the output of your code. Mine, for example, looks as follows:

```
0 39.86579513549805
10 12.93497085571289
20 2.6193177700042725
error in estimating w: tensor([-0.5257,  0.3273,  0.6399])
error in estimating b: tensor([0.4122])
```

Training

Your second task is to adapt the **train** function to implement mini-batch gradient descent in the function **batched_train**. In order to help think through this process, you must copy the code from **train** to your pdf document and annotate the function. Note what each line does. You should make sure to note which line calculates the gradient and which line updates the weights of your model.

In implementing mini-batch gradient descent, here are your decision specifications:

- Your training function should iterate over your entire dataset 3 times (that is, it should train for 3 epochs)
- You should output the average loss over the epoch
- Your code should work with any reasonable **batch_size** (batch sizes less than the total size of your dataset, for example). You can assume the user only uses reasonable **batch_size** values.

To evidence completion of this task, and to check your own progress, please train your model using your **batched_train** function and print the error of each of your parameters, as before. You should add to the pdf accompanying your code a screenshot of the output of your code. Mine, for example, looks as follows:

```
0 12.478017888963223
1 6.058776119351387
2 2.4219589471817016
error in estimating w: tensor([ 0.5886, -0.3861,  0.0994])
error in estimating b: tensor([0.0387])
```

Wrap-up

Finally, you should reflect on the performance of both the **train** method provided and your implementation of **batched_train**. You should concretely answer the following questions in the pdf accompanying your code:

1. Is **train** an implementation of stochastic, mini-batch, or batch gradient descent? Why?
2. How many times does your model see each sample in **train** vs **batched_train**?
3. Compare the final performance of the two approaches to training (i.e., how good are the final models). Which approach to training do you prefer and why? Be concrete in discussing the performance and your choice.