

## Problem Set 12, Dec 13, 2018 (PyTorch Introduction)

**Goals.** The goal of this exercise is to

- introduce you to the PyTorch platform.

### 1 Theory Questions

**Problem 1 (How to compute  $U$  and  $S$  efficiently):**

In class, we saw that solving the eigenvector/value problem for the matrix  $\mathbf{X}\mathbf{X}^\top$  gives us a way to compute  $U$  and  $S$ . But in some instances  $D \gg N$ . In those cases, is there a way to accomplish this computation more efficiently?

**Problem 2 (Positive semi-definite):**

Show that if  $\mathbf{X}$  is a  $N \times N$  symmetric matrix then the SVD has the form  $\mathbf{U}\mathbf{S}\mathbf{U}^\top$ , where  $\mathbf{U}$  is a  $N \times N$  unitary matrix and  $\mathbf{S}$  is a  $N \times N$  diagonal matrix with non-necessarily positive entries. Show that if  $\mathbf{X}$  is positive semi-definite, then all entries of  $\mathbf{S}$  are non-negative.

### 2 PyTorch Getting Started

**Tutorials.** Installation instructions:

[pytorch.org](http://pytorch.org)

We recommend using the following online tutorial:

[pytorch.org/tutorials/beginner/pytorch\\_with\\_examples.html](http://pytorch.org/tutorials/beginner/pytorch_with_examples.html)

**Setup, data, and sample code.** Obtain the folder `labs/ex11` of the course github repository

[github.com/epfml/ML\\_course](https://github.com/epfml/ML_course)

**Exercise 1: Torch** Familiarize yourself with the basics of pytorch through the tutorial.

**Exercise 2: Basic Linear Regression**

- Implement prediction and loss computation for linear regression in the `MyLinearRegression` class.
- Implement the gradient descent steps in the `train` function.
- **HINT:** don't forget to clear the gradients computed at previous steps.

Your output should be similar to that of Fig. 1, left.

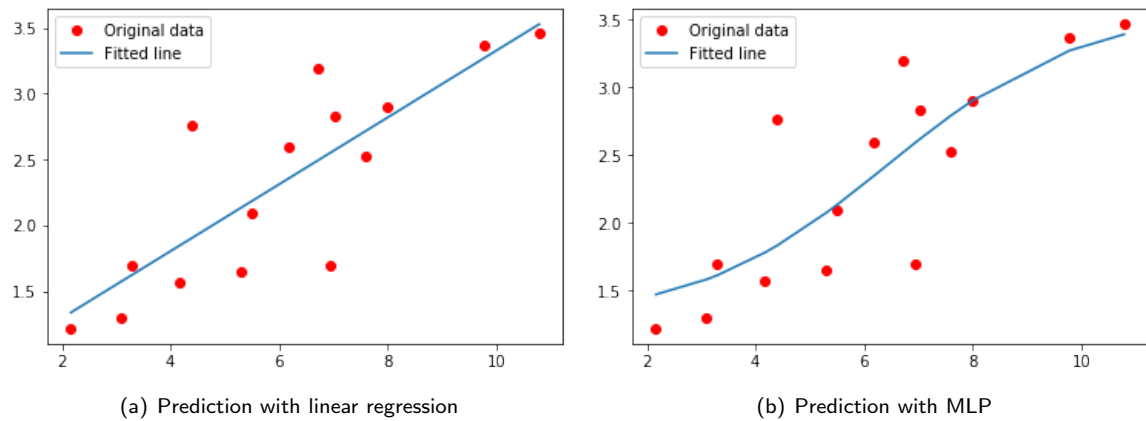


Figure 1: Predictions made by various trained models.

### Exercise 3: NN package

- Re-Implement Linear Regression using the routines from the `nn` package for defining parameters and loss in the `NNLinearRegression` class. Does the result that you obtain differ from the previous one? If so, why?
- Combine two linear layers and a non-linearity (sigmoid or ReLU) layer to build a Multi-Layer Perceptron (MLP) with one hidden layer, in the `MLP` class. Find the optimal hyper-parameters for training it.

Your prediction using the MLP should be non-linear, and for a hidden size of 2 might look like Fig. 1, right.