Labs
Machine Learning Course
Fall 2018

EPFL
School of Computer and Communication Sciences
Martin Jaggi & Rüdiger Urbanke
mlo.epfl.ch/page-157255-en-html/
epfmlcourse@gmail.com

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

Goals. The goal of this exercise is to

- introduce you to the PyTorch platform.
- discuss the vanishing gradient problem

Problem 1 (PyTorch Getting Started):

Tutorials. Installation instructions:

pytorch.org

We recommend using the following online tutorial:

pytorch.org/tutorials/beginner/pytorch_with_examples.html

Setup, data, and sample code. Obtain the folder `labs/ex12` of the course github repository

github.com/epfml/ML_course

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

Exercise 2: Basic Linear Regression

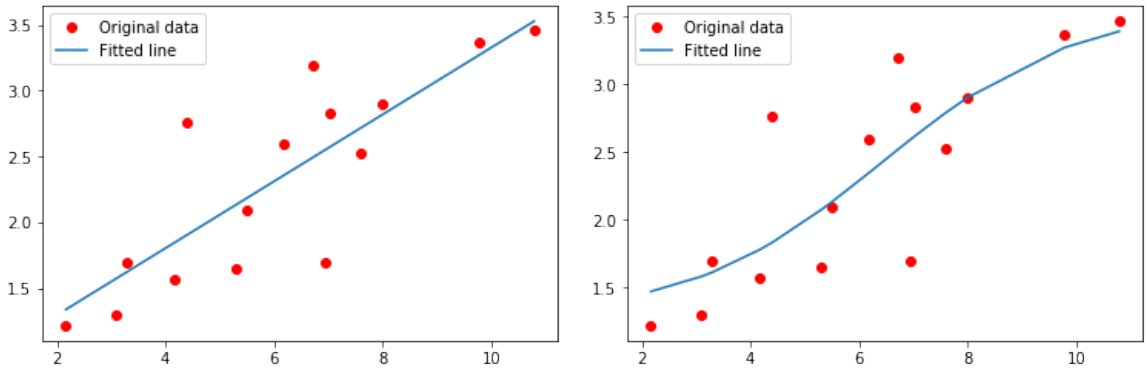
- Implement prediction and loss computation for linear regression in the `MyLinearRegression` class.
- Implement the gradient descent steps in the `train` function.
- **HINT:** don't forget to clear the gradients computed at previous steps.

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

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.



(a) Prediction with linear regression

(b) Prediction with MLP

Figure 1: Predictions made by various trained models.

Problem 2 (Vanishing Gradient Problem):

Over the past few years it has become more and more popular to use "deep" neural networks, i.e., networks with a large number of layers, sometimes counting in the hundreds. Emperically such networks perform very well but they pose new challenges, in particular in the training phase. One of the most challenging aspect is what is called the "vanishing gradient" problem. This refers to the fact that the gradient with respect to the parameters of the network tends to zero typically exponentially fast in the number of layers. This is a simple consequence of the chain rule which says that for a composite function $f(x) = g_n(g_{n-1}(\cdots g_1(x)\cdots))$ the derivative has the form

$$f'(x) = g_n'(\cdots)g_{n-1}'(\cdots)\cdots g_1'(x).$$

The aim of this exercise is to explore this problem.

Consider a neural net as introduced in the course with L layers, $K = 3$, and the sigmoid function as activation layer. Assume that all weights $w_{i,j}^{(l)}$ are bounded, lets say $|w_{i,j}^{(l)}| \leq 1$. Consider a regression task where the output layer has a single node representing a simple linear function with some bounded weights c_i, lets say $|c_i| \leq 1$. Hence the overall function, call it f is a scalar function on $\mathbb{R}^D$. Show that the derivative of this function with respect to the weight $w_{1,1}^{(1)}$ vanishes exponentially fast as a function of L at a rate of at least $(\frac{3}{4})^L$.