Biomedical Data Science Lab
Fall 2019

Introduction to optimization with PyTorch

Benjamín Béjar Haro
Assistant Research Professor
Department of Biomedical Engineering
319B Clark Hall, Johns Hopkins University
E-mail: bbejar@jhu.edu

1 Pre-Lab Description

In this pre-lab we will learn how to implement gradient descent for finding local minima of a given cost function. This will provide us with a basic tool for many learning and classification problems since, at the end of the day, finding a classifier amounts to solving some optimization problem. In this pre-lab we will also learn how gradient descent can be implemented using PyTorch, a scientific library for developing machine (deep) learning methods. Towards that goal, we will be learning a linear classifier on the MNIST digit dataset. As a loss function, we will be using a simple quadratic function. You will first apply your calculus skills to the problem, and analytically solve it. Then you will learn how to solve the same problem by implementing the gradient descent method and applying it to the cost function. Finally, you will learn the basics of PyTorch by using the built-in functions to train the classifier. This pre-lab assignment needs to be solved in the companion IPython Notebook. All exercises are worth the same points.

Getting started. Go to the class website BMDS Lab and download the IPython notebook `.ipynb` for this pre-lab assignment. On your Google drive, create a folder (*e.g.,* `'My Drive/bmdslab/prelab-02/'`) and upload the downloaded notebook there. Open the IPython notebook in Google Colaboratory. Follow the instructions in the notebook and on this manual to complete the assignment.

Problem description. We are given a set of N feature-label pairs $\left\{(\boldsymbol{x}_i, c_i)\right\}_{i=0}^{N-1}$ where each $\boldsymbol{x}_i \in \mathbb{R}^p$ corresponds to a vectorized 28×28 grayscale image of a digit, and $c_i = \{0, 1, \ldots, 9\}$ is the digit's class. Since we are dealing with a multi-class classification problem we will encode each digit's class with a one-hot embedding as:

$$\boldsymbol{y}_i = [y_{i0}, \ldots, y_{in}], \quad y_{ij} = \begin{cases} 1 & c_i = j \\ 0 & \text{else} \end{cases}.$$

The goal is then to find a prediction function $f : \mathbb{R}^p \mapsto \{0, 1\}^n$ that maps features $\boldsymbol{x}_i$ (images) to labels $\boldsymbol{y}_i$. In order to do so, we will use a linear prediction function:

$$f(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x}, \quad \boldsymbol{W} \in \mathbb{R}^{n \times p},$$

where the jth row of $\boldsymbol{W}$ represents a predictor for the jth class. In order to decide upon the estimated class we take the strongest response of our set of predictors, that is:

$$\hat{c}_i = \arg\max_j \boldsymbol{W}\boldsymbol{x}_i.$$

Optimization problem. With all previous considerations in mind we can now define the optimization problem to estimate the parameters $\boldsymbol{W}$ of our linear predictor. In order to do that, we need to define some loss function on our predictions

that penalizes deviations from the true target. For this problem, we will be using a simple quadratic loss function $L\big(f(\boldsymbol{x}), \boldsymbol{y}\big) = \|\boldsymbol{y} - f(\boldsymbol{x})\|_2^2$. The goal is then to find the parameters $\boldsymbol{W}$ of our linear predictor function $f(\cdot)$ that minimize the average loss over the set of samples:

$$\min_{\boldsymbol{W}} \ \frac{1}{N} \sum_{i=0}^{N-1} \|\boldsymbol{y}_i - \boldsymbol{W}\boldsymbol{x}_i\|_2^2.$$

Note that the above optimization problem can be expressed in a compact form as:

$$\min_{\boldsymbol{W}} \ \frac{1}{N} \|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{X}\|_F^2, \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius (ℓ_2) norm of a matrix, and where the matrices $\boldsymbol{Y} = [\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{N-1}]$ and $\boldsymbol{X} = [\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{N-1}]$ consist of stacking the label and feature vector representations, respectively.

Exercise 1. Given the feature and label matrices $\boldsymbol{X} \in \mathbb{R}^{p \times N}$ and $\boldsymbol{Y} \in \mathbb{R}^{n \times N}$, find a closed-form solution $\boldsymbol{W}^\star$ for the optimization problem:

$$\min_{\boldsymbol{W}} \ \frac{1}{N} \|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{X}\|_F^2.$$

You can find the minimizer by setting the derivative of the cost function to zero.

Exercise 2. Using the data provided for training and the expression for the optimal predictor's weights derived in the previous exercise compute the optimal predictor over the training data. Apply also your predictor to the training set. Report classification accuracy over both training and testing sets.

Exercise 3. *(Optional)* Write down the equation for the gradient descent of the considered problem. Starting from an initial weight matrix of all zeros $\boldsymbol{W}^{(0)} = \boldsymbol{0}$ implement a gradient descent optimization algorithm to find the optimal solution to our classification problem. Run the method for a sufficiently large number of iterations or until you meet some convergence criterion (*e.g.,* relative change of the cost function smaller than some threshold). On two separate plots, display the evolution of the cost function over the iterations and the difference between your current estimate and the optimal estimate $\|\boldsymbol{W}^\star - \boldsymbol{W}^{(k)}\|_F^2$. Since the considered cost function has a unique and global minimizer your iterates should converge to the optimal solution obtained from the analytical expression.

Exercise 4. Repeat the exercise before but now using PyTorch instead of manually implementing the gradient descent step. You can use random initialization for the weights $\boldsymbol{W}^{(0)}$. Why initialization does not matter in this problem?