GRADIENT DESCENT METHOD(S) AND LARGE SCALE APPLICATIONS

TUYEN TRUNG TRUONG

In mathematics, and also in the real life, one usually faces with the question of finding optimal values of a given function. For example, and this will be the main application to be aimed with in the course, at the core of its, Deep Learning can be formulated mathematically as follows: You have some given data called x_i (say a bunch of pictures, and here the x_i represents the values of the pixels in the computer for the image i) and you would like to correctly assign a label y_i (say whether there is a cat $y_i = 0$, there is a dog $y_i = 1$, there are both a cat and a dog $y_i = 2$, or there is neither cat nor dog $y_i = 3$). You will then design a neural network, which simply is a function $f(\alpha) = \sum_i f(x_i, y_i, \alpha)$ - called cost function - where i runs all over a part of the data set called training set, and α is a set of parameters, and want to find the values of α which minimise this function. [In practice, you need to modify this, by using the so-called mini-batches, but that's not important here.]

Another example, which is more pure mathematics, is usually when you want to solve a PDE, you can cook up some map $F(u)$ whose arguments are functions u satisfying certain conditions, and whose optima will solve the PDE you want.

In some cases, where the function is really simple, such as $f(\alpha_1, \alpha_2) = (2\alpha_1+\alpha_2-1)^2+2\alpha_1$, you can actually solve (either by hand or by computers) to find the correct optima. However, this is rare when you are doing real life questions and you have pressure on time. Thus approximation methods are needed. Gradient descent is such a method.

In its standard form, the gradient descent method is the following. Given $f : \mathbb{R}^m \to \mathbb{R}$ which is C^1. You choose a fixed number $\delta > 0$, and a given point $x_0 \in \mathbb{R}^m$. You construct the sequence $x_{n+1} = x_n - \delta \nabla f(x_n)$, and hope that it will converge to a minimum point. It does not always do so, but surprisingly, it usually do in practice. In this course, under the guidance from the lecturer, the student will study to know the reason for this and many more.

I can offer two directions:

Direction 1: Theory. The student will study theoretical properties of optimisation methods.

Direction 2: Applied. The student will run experiments to test the optimisation methods on Deep Neural Networks.

References: (More and customised references will be given later)

T. T. Truong and T. H. Nguyen, Backtracking gradient descent method for general C^1 functions, arXiv: 1808.05160. One part of the paper is published online in the journal Applied Mathematics and Optimization. Source code is available on GitHub.

Date: January 5, 2021.

2010 *Mathematics Subject Classification.* 32-xx,

Tuyen Trung Truong, Tat Dat To, Tuan Hang Nguyen, Thu Hang Nguyen, Hoang Phuong Nguyen and Maged Helmy, A modification of quasi-Newton's methods helping to avoid saddle points, arXiv:2006.01512. Source code is available on GitHub.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola, Dive into Deep Learning, Release 0.14.4, Online book.

Department of Mathematics, University of Oslo,
E-mail address: tuyentt@math.uio.no