Subgradient Method

Indian Institute of Technology, Hyderabad
Chirag Mehta
ai20btech11006@iith.ac.in

April 29, 2022

Abstract

Non-differentiable functions are an important class of functions which often appear in optimization problems, a gradient descent method would fail to optimize such an objective because the gradient might not exist at all points that the method encounters. Methods such as interior point method work great, but it has its own limitations of being computationally inefficient. We will explore the subgradient method which is an iterative first-order method similar to gradient descent.

I. Introduction

Subgradient method is a simple algorithm used to minimize non-differentiable convex functions. This method is similar to vanilla gradient method which is used to optimize differentiable functions. [2]

i. Algorithm

Lets say we have a nondifferentiable function $f : \mathbb{R}^n \to \mathbb{R}$. The update rule of subgradient method says

$$\underline{x}^{(k+1)} = \underline{x}^{(k)} - \alpha_k \underline{g}^{(k)} \tag{1}$$

where $\underline{x}^{(k)}$ is the k^{th} iterate, α_k is the step step size at k^{th} iteration and $\underline{g}^{(k)}$ is any subgradient of f at $\underline{x}^{(k)}$. The subgradient $\underline{g}^{(k)}$ is any vector which satisfies

$$f(\underline{y}) \geq f(\underline{x}) + \underline{g}^T(\underline{y} - \underline{x}) \tag{2}$$

At a given pozint there can be more than one subgradients, we call the set of subgradients as subdifferential. This algorithm doesn't ensure that the loss always decreases, the loss can and often does increase, we store the best point encountered so far for that reason.

Theorem I.1. *If the function f is differentiable at $\underline{x}^{(k)}$ then $\underline{g}^{(k)}$ is equal to the gradient of f at $\underline{x}^{(k)}$*

Proof. Substitute $\underline{y} = \underline{x} + \lambda \underline{z},\ \lambda > 0$ in (2)

$$\frac{f(\underline{x} + \lambda \underline{z}) - f(\underline{x})}{\lambda} \geq \underline{g}^T \underline{z} \tag{3}$$

We can use the limit $\lambda \to 0$

$$\underline{\nabla} f(\underline{x})^T \underline{z} \geq \underline{g}^T \underline{z} \tag{4}$$

$$\underline{z}^T \left(\underline{\nabla} f(\underline{x}) - \underline{g} \right) \geq 0 \quad \forall \underline{z} \tag{5}$$

$$\therefore \underline{g} = \underline{\nabla} f(\underline{x}) \tag{6}$$

□

II. Example

Consider the following problem

$$min \quad max\{f_1, f_2, \ldots f_n\},\ f_i : \mathbb{R} \to \mathbb{R} \tag{7}$$

This problem is non differentiable but it can be solved using subgradient method. The first step is to calculate the subgradients are all the points that occur in our iterative algorithm. To obtain the subgradients we can directly use the right hand derivative as the subgradients would lie in the interval of left hand gradient and right hand gradient. A numerical example is discussed in (i)

III. SVM using subgradient method

A support vector machine is used for two class classification. The objective is to maximize the slab thickness while still satisfying few constraints. A hard-margin SVM can be formuated as

$$\min_{\underline{w},b} \ \underline{w}^T\underline{w} \tag{8}$$

$$\text{s.t: } y_i(\underline{w}^T\underline{x_i} + b) \geq 1 \tag{9}$$

We can transform this problem into

$$\min_{\underline{w},b} \ \underline{w}^T\underline{w} + \lambda \sum_i \max\left(0, 1 - y_i\left(\underline{w}^T\underline{x} + b\right)\right) \tag{10}$$

Where λ is the trade off factor, the higher is the value of this parameter, the higher is the penalty of voilating the given constraints. This problem is essentially a soft-margin SVM. We can now solve this problem by subgradient method.

The first step is to calculate the subgradients,

$$\underline{g_w} = 2 * \underline{w} + \lambda \sum_i y_i * \underline{x_i} \ \ for\, y_i\left(\underline{w}^T\underline{x} + b\right) < 1 \tag{11}$$

and

$$g_b = \lambda \sum_i y_i \ \ for\, y_i\left(\underline{w}^T\underline{x} + b\right) < 1 \tag{12}$$

[1]In practice, we tend to use mini-batch subgradient method because of its several advantages such as computational efficiency, stable convergence and faster learning.

IV. Results

i. Maximum of convex functions

Consider the following problem of the type maximum of linear functions

$$min \ \ max\{f_1, f_2, \ldots f_5\} \tag{13}$$

given

$$f_1(x) = -5x - 25$$
$$f_2(x) = -3x - 10$$
$$f_3(x) = -x + 1$$
$$f_4(x) = 2x + 4$$
$$f_5(x) = 5x + 20$$

We will use the algorithm described in (i) to arrive at the optimal solution. There is still uncertainty as to how to choose the step size α_k. There are a few different standard step size rules that we are comparing below.

1. Constant Step Size

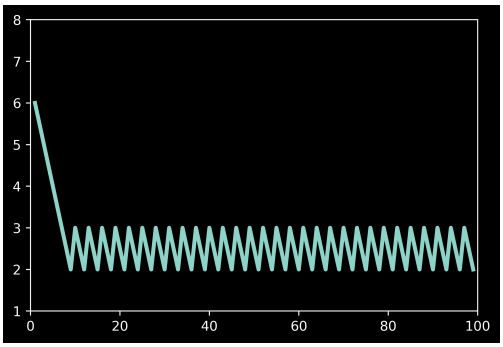

Figure 1: *Constant Step Size*

2. Constant Step Length

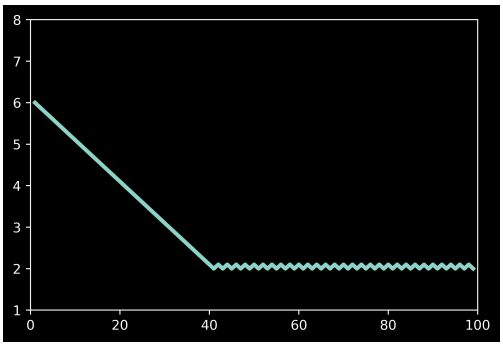

Figure 2: *Constant Step Length*

3. Square Summable But Not Summable

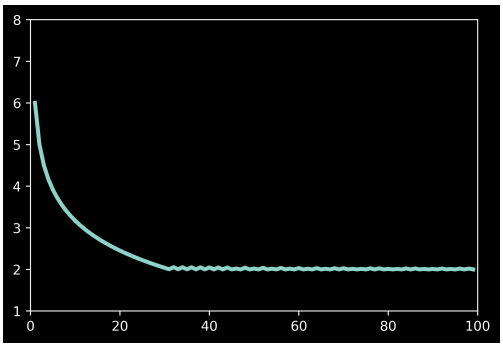

Figure 3: *Square Summable But Not Summable*

4. Non Summable Diminishing

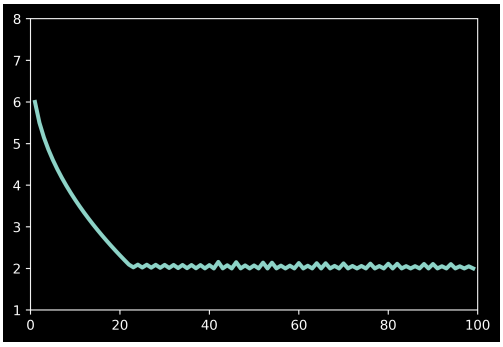

Figure 4: *Non summable diminishing*

We can see the difference in convergence across different step size rules.

The link to original code can be found here and the mp4 animation files can be found here.

ii. SVM

ii.1 Hard-margin and soft margin SVM

As discussed in section (III), a abnormally high tradeoff parameter would correspond to hard-margin SVM. A code demonstrating the same can be found here

ii.2 Spam Classification

Spam classification problem can be solved by soft margin SVM using linear kernel. The text in email can be tokenized and encoded in higher dimension. For our purposes, *sklearn.feature_extraction.text.CountVectorizer* [3] can be used. The SVM written from scratch using numpy is able to attain an accuracy of 98.56%. The code can be found here

References

[1] Stephen Boyd and Almir Mutapcic. Stochastic subgradient methods. *Lecture Notes for EE364b, Stanford University*, 2008.

[2] Stephen Boyd, Lin Xiao, and Almir Mutapcic. Subgradient methods. *lecture notes of EE392o, Stanford University, Autumn Quarter*, 2004:2004–2005, 2003.

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.