AI2101 - Convex Optimization Project

Subgradient Method

Chirag Mehta: ai20btech11006@iith.ac.in

Indian Institute Of Technology, Hyderabad

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]

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

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

Subdifferential

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.

Subdifferential Conti.

Theorem

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 $\lambda \to 0$

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

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

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

Step Size Rules

We saw the updation rule in (1), the term α_k is called step size. There are a few standard step size rules which are similar to what we use in gradient descent. We will discuss a few different selections for step size

Step Size Rules Conti.

1. Constant Step Size. $\alpha_k = \alpha$, $\alpha > 0$
2. Constant step length, this means that $\left\|\alpha_k * \underline{g}^{(k)}\right\| = \gamma$, where γ is a constant fixed ahead of time.
3. Square summable but not summable. The step satisfies

$$\alpha_k \geq 0, \sum_{k=1}^{\infty} \alpha_k^2 < \infty, \sum_{k=1}^{\infty} \alpha_k \to \infty \tag{7}$$

Step Size Rules Conti.

4. Nonsummable diminishing. The step sizes satisfy

$$\alpha_k \geq 0, \quad \lim_{k\to\infty} \alpha_k = 0, \quad \sum_{k=1}^{\infty} \alpha_k \to \infty \tag{8}$$

4. Nonsummable diminishing step lengths. The step sizes are chosen as $\alpha_k = \gamma_k / \left\| \underline{g}^{(k)} \right\|$ where

$$\gamma_k \geq 0, \quad \lim_{k\to\infty} \gamma_k = 0, \quad \sum_{k=1}^{\infty} \gamma_k \to \infty \tag{9}$$

Convergence

We can prove that using subgradient method the results would converge in a certain range in at most certain number of steps. There are few key assumptions here

1. $\left\|\underline{g_k}\right\| \leq G \, \forall k$
2. $\left\|\underline{x}^{(1)} - \underline{x}^*\right\| \leq R$ where $\underline{x}^*$ is the optimal solution.

Example

Consider the following problem

$$min \quad max\{f_1, f_2, \ldots f_n\}, \;\; f_i : \mathbb{R} \rightarrow \mathbb{R} \tag{10}$$

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.

Numerical Example

Consider the following problem of the type maximum of linear functions

$$min \quad max\{f_1, f_2, \ldots f_5\} \tag{11}$$

given

$$\begin{aligned} f_1(x) &= -5x - 25 \\ f_2(x) &= -3x - 10 \\ f_3(x) &= -x + 1 \\ f_4(x) &= 2x + 4 \\ f_5(x) &= 5x + 20 \end{aligned}$$

We will use the subgradient method algorithm to arrive at the optimal solution. There is still uncertainty as to how to choose the step size α_k. We will compare a few different step sizes

Numerical Example Conti

Constant Step Size

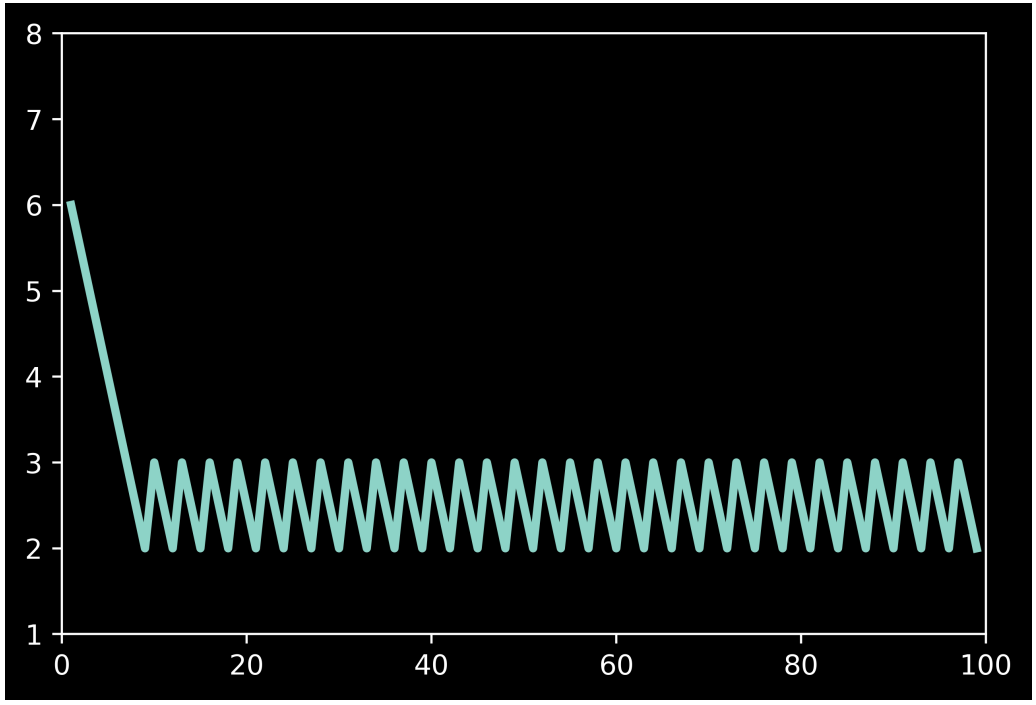

Figure: Constant Step Size

Numerical Example Conti

Constant Step Length

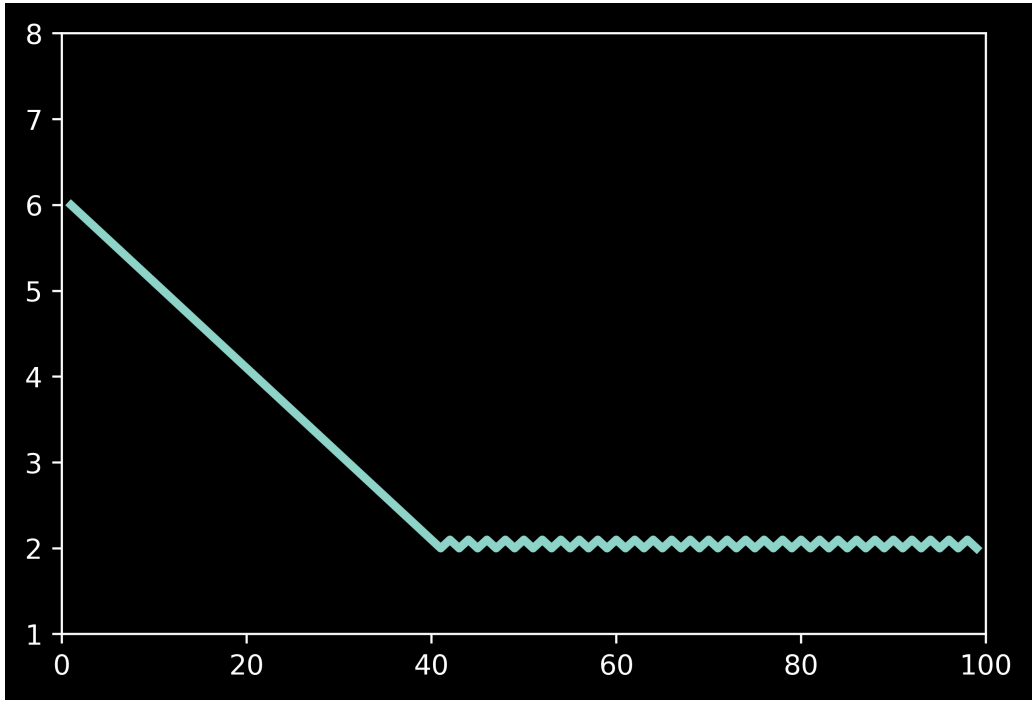

Figure: Constant Step Length

Numerical Example Conti

Square Summable But Not Summable

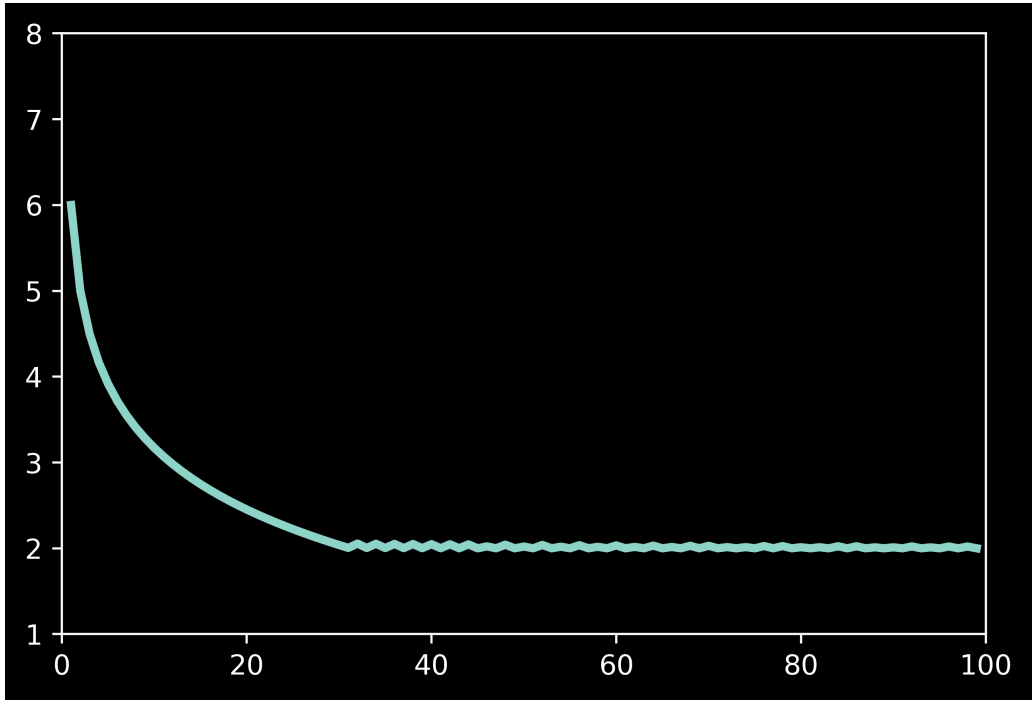

Figure: Square Summable But Not Summable

Numerical Example Conti

Non Summable Diminishing

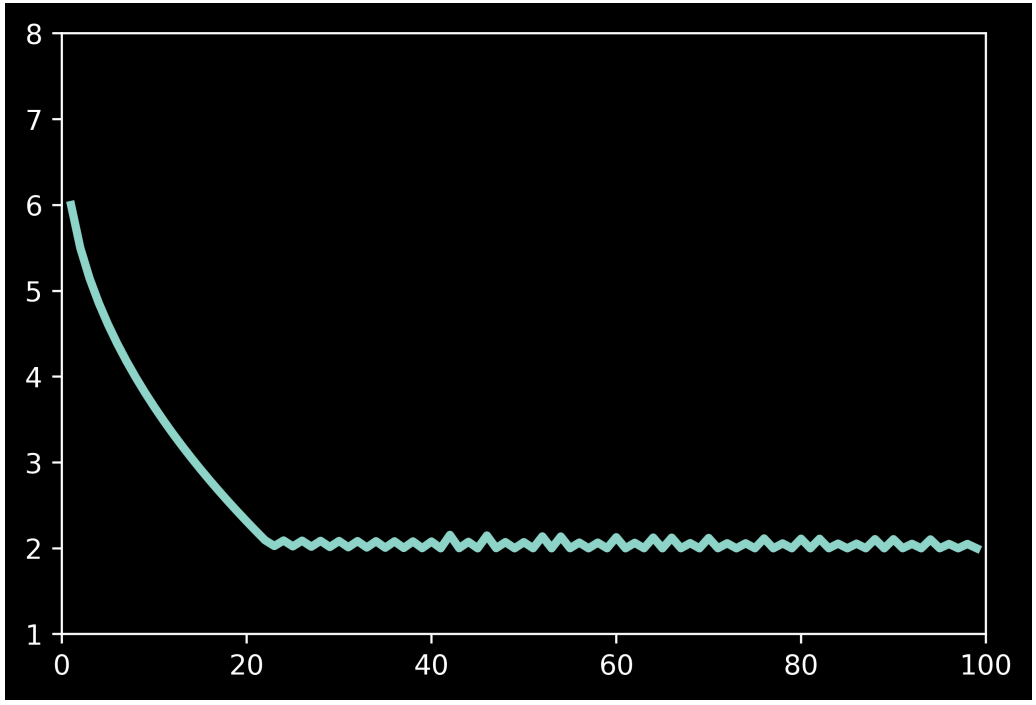

Figure: Non summable diminishing

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

$$\min_{\underline{w},b} \ \underline{w}^T \underline{w} \tag{12}$$

$$\text{s.t: } y_i(\underline{w}^T \underline{x_i} + b) \geq 1 \tag{13}$$

[4]We can transform this problem into

$$\min_{\underline{w},b} \ \underline{w}^T \underline{w} + \lambda \sum_i \max\left(0, 1 - y_i\left(\underline{w}^T \underline{x} + b\right)\right) \tag{14}$$

SVM using subgradient method Conti.

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.

SVM using subgradient method Conti.

The first step is to calculate the subgradients,

$$\underline{g}_{\underline{w}} = 2\underline{w} + \frac{1}{n} \times \lambda \sum_i y_i \left(\underline{x_i}\right) \quad for\ y_i \left(\underline{w}^T \underline{x} + b\right) < 1 \tag{15}$$

and

$$g_b = \frac{1}{n} \times \lambda \sum_i^n y_i \quad for\ y_i \left(\underline{w}^T \underline{x} + b\right) < 1 \tag{16}$$

[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.

SVM results

Now, we can use these update rules to train our SVM. Given below are few illustrations of our SVM.

Hard Margin SVM: $\lambda = 1e9$

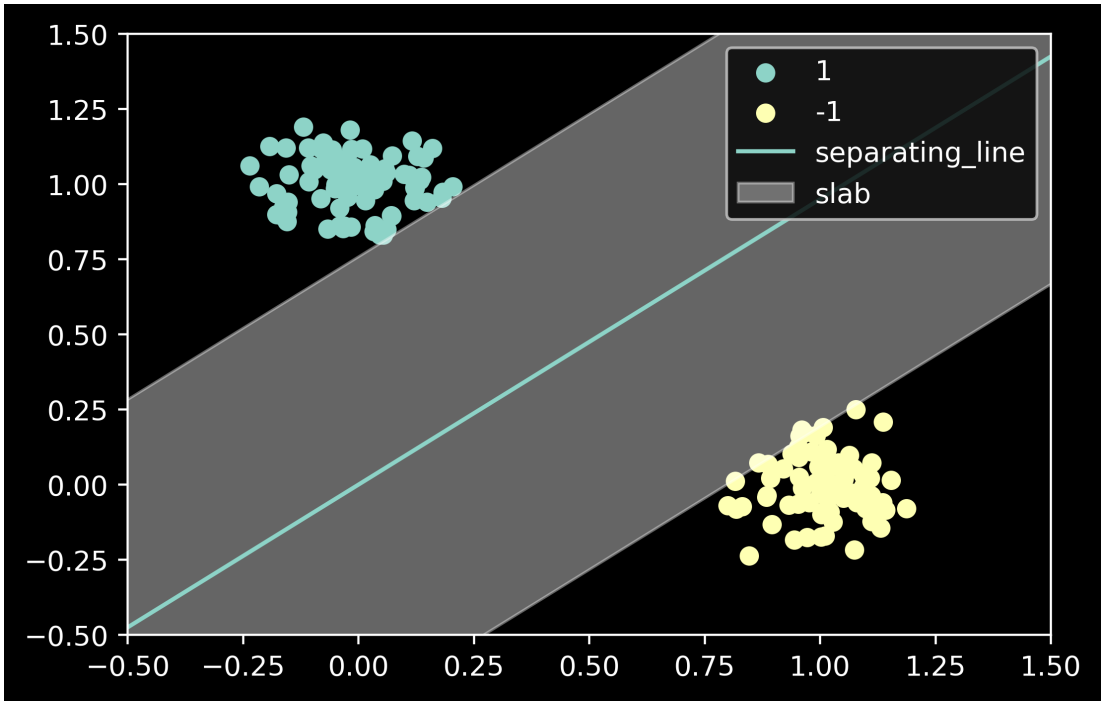

Figure: Hard Margin SVM

SVM results

Soft Margin SVM: $\lambda = 100$

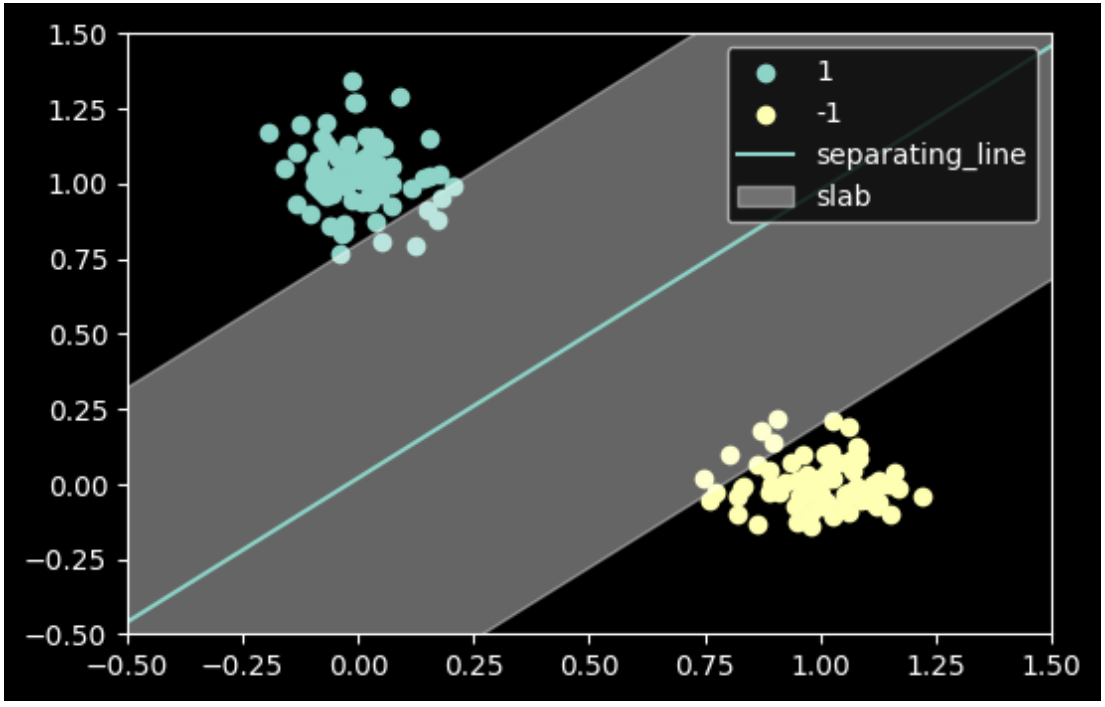

Figure: Soft Margin SVM

Spam Classification using SVM

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

Stephen Boyd and Almir Mutapcic.

Stochastic subgradient methods.

Lecture Notes for EE364b, Stanford University, 2008.

Stephen Boyd, Lin Xiao, and Almir Mutapcic.

Subgradient methods.

lecture notes of EE392o, Stanford University, Autumn Quarter, 2004:2004–2005, 2003.

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.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro.

Pegasos: Primal estimated sub-gradient solver for svm.

In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 807–814, New York, NY, USA, 2007. Association for Computing Machinery.