Exploration Of Subgradient Methods

Fatiha BARRADE, Fatima NOUTFI

Keywords. *Subgradient Polyak's Step Length , Projected Subgradient, Minimization,Constrained-Optimization*

1 Introduction

Subgradient methods play a crucial role in optimization, particularly when dealing with non-differentiable and convex functions. The concept of subgradients extends the idea of gradients for differentiable functions to a broader class of convex functions, allowing for efficient optimization in scenarios where traditional gradient-based methods may fall short.

In this document, we delve into various aspects of subgradient methods, exploring their definitions, properties, and applications in optimization. The fundamental idea revolves around the concept of subdifferential, denoted by f(x), which represents the set of all subgradients at a given point x. This set captures the potential directions of descent for non-differentiable convex functions.

2 Generalities

2.1 Motivation for Subgradient Descent

The motivation behind Subgradient Descent (SGD) stems from the recognition that traditional Gradient Descent (GD) algorithms rely on smoothness assumptions, particularly Lipschitz continuity of the gradient, which may not hold in many real-world scenarios. When faced with functions that are convex but lack smoothness, such as those with non-Lipschitz gradients, SGD offers a viable optimization approach.

Consider the example of minimizing the absolute value function, $|x|$. This function is convex but lacks a Lipschitz gradient property. Around the point where x equals 0, the gradient changes abruptly from -1 to 1 or vice versa, violating the Lipschitz condition. However, the function itself remains Lipschitz, adhering to the inequality $|f(x) - f(y)| \leq |x - y|$.

In such cases, traditional GD algorithms struggle, as they are designed to utilize smoothness assumptions to efficiently descend along the gradient towards the minimum. However, with functions like $|x|$, where the gradient may not provide consistent directional information, a different approach is needed.

This is where Subgradient Descent comes into play. Rather than relying on the gradient, which may not exist or be well-defined at certain points, SGD employs subgradients. A subgradient of a convex function at a point is a vector that provides a valid descent direction, though it may not necessarily point towards the steepest descent.

The fundamental idea behind SGD is to iteratively update the current solution by moving in the direction of a subgradient. While this approach may not guarantee convergence to the optimal solution in the same efficient manner as GD does for smooth functions, it still offers a principled way to optimize convex functions in the absence of smoothness assumptions.

Moreover, subgradients allow for the optimization of a broader class of functions beyond those with Lipschitz gradients. By considering the set of subgradients at a point, rather than a single gradient, SGD accommodates functions with nonsmooth regions, making it applicable to various real-world optimization problems.

2.2 Definition

A subgradient of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ at a point x is any vector $g \in \mathbb{R}^n$ such that for all y:

$$f(y) \geq f(x) + g^T(y - x)$$

This property generalizes the concept of a gradient for differentiable functions and holds even for non-differentiable convex functions. The set of all subgradients at a point x is denoted by $\partial f(x)$.

For convex and differentiable functions, if f is differentiable at x, then the subgradient g is uniquely determined and is equal to the gradient $\nabla f(x)$. However, for non-

differentiable points, there may be multiple subgradients.

2.3 Subdifferential

The set of all subgradients of a convex function f at a point x is formally defined as the subdifferential, denoted by $\partial f(x)$. It is represented as the collection of vectors g in $\mathbb{R}^n$ such that g is a subgradient of f at the point x. Mathematically, it is defined as follows:

$$\partial f(x) = \{g \in \mathbb{R}^n : g \text{ is a subgradient of } f \text{ at } x\}$$

Several important properties characterize the subdifferential:

- The subdifferential $\partial f(x)$ is a closed and convex set, preserving these properties even for nonconvex functions.
- The subdifferential is guaranteed to be nonempty, except for certain cases involving nonconvex functions where it may be empty.
- If the function f is differentiable at the point x, then the subdifferential $\partial f(x)$ reduces to the singleton set containing the gradient $\nabla f(x)$, i.e., $\partial f(x) = \{\nabla f(x)\}$.
- Conversely, if $\partial f(x) = \{g\}$, where g is a single subgradient, then f is differentiable at x, and the gradient $\nabla f(x)$ is equal to g. The uniqueness of the subgradient in this case implies differentiability.

2.4 Convergence

The convergence results for the subgradient method can be expressed as follows.
In the case of a constant step size and constant step length, the subgradient algorithm guarantees convergence to a solution within a certain range of the optimal value. Specifically, as the iteration index k approaches infinity, the difference between the current best function value $f(k)_{\text{best}}$ and the optimal value $f^\star$ diminishes:

$$\lim_{k\to\infty} (f(k)_{\text{best}} - f^\star) < \varepsilon,$$

where $f^\star$ denotes the optimal value of the problem ($f^\star = \inf_x f(x)$). This implies that the subgradient method finds an ε-suboptimal point within a finite number of steps. The parameter ε is a function of the step size parameter h and decreases with it.

For diminishing step size and step length rules, including the square summable but not summable step size rule, the algorithm ensures convergence to the optimal value:

$$\lim_{k\to\infty} f(x(k)) = f^\star.$$

It is noteworthy that such a simple algorithm can be employed to minimize any convex function for which subgradients can be computed at each point. Additionally, when the function f is differentiable, the subgradient method with a constant step size ensures convergence to the optimal value, provided the parameter α is chosen small enough. This convergence proof remains straightforward even in the case of differentiability.

2.5 Subgradient Calculus

In the realm of optimization, the pursuit of a nuanced understanding of subgradients involves distinguishing between the 'strong' and 'weak' calculus. The

2.5.1 Weak Subgradient

The primary aim in this method is to derive a singular subgradient, even when the possibility exists for multiple subgradients at a specific point. This approach proves practical, particularly in applications such as subgradient methods, localization techniques, and cutting-plane methods, where the requirement is satisfied by having just one subgradient at any given point.

2.5.2 Strong Subgradient

Strong calculus of subgradients endeavors to articulate a comprehensive set of subgradients $\partial f(x)$ as a function of x. This intricate task is particularly valuable in theoretical investigations, providing precise optimality conditions and facilitating a detailed exploration of the solution space.

2.5.3 Subgradient Algorithm

Algorithm 1 Generic Subgradient Method

1: **Initialization:** $x_0 \in \mathbb{R}^d$.
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: 1. Compute a subgradient $g_k \in \partial f(x_k)$.
4: 2. Compute a steplength $\alpha_k > 0$.
5: 3. Set $x_{k+1} = x_k - \alpha_k g_k$.
6: **end for**

3 Projected Subgradient Method

The projected subgradient method is an optimization technique used to minimize a convex, non differentiable function subject to constraints defined by a convex set. It operates by performing subgradient on the objective function while projecting parameter updates onto the constraint set after each update. This ensures that the solutions remain within the defined feasible set.

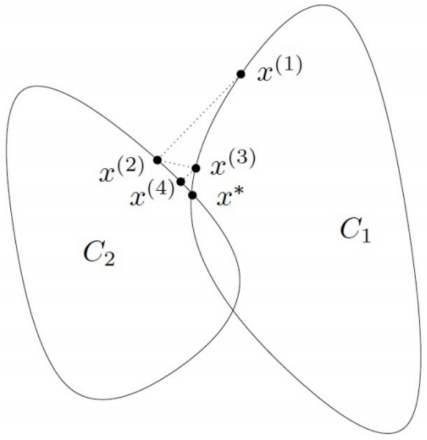

Figure 1: Projection

Projected Subgradient effectively navigates through such scenarios by ensuring that solutions remain within the defined feasible set.

For instance, consider the optimization problem of minimizing the non-differentiable function $f(x) = |x|$ subject to the nonconvex constraint $|x| \leq 1$.

Here, the non-differentiability of $f(x)$ poses a challenge for traditional subgradient-based methods. However, PSG handles this effectively by incorporating subgradients and projecting solutions onto the feasible set $|x| \leq 1$ at each iteration.

This approach not only enforces the constraint but also preserves feasibility, crucial for maintaining the integrity of the optimization process. Moreover, PSG's ability to handle nonconvex constraints ensures robustness in navigating complex solution spaces.

By focusing the optimization process on feasible regions, PSG potentially improves convergence compared to unconstrained methods, enhancing its effectiveness in solving challenging optimization problems.

Mathematical Formulation: Given a convex objective function $f(x)$ to minimize subject to the constraints $x \in C$, where C is a convex set, the projected subgradient method updates the parameter x iteratively as follows:

$$x_{k+1} = \Pi_C \left(x_k - \alpha_k \nabla f(x_k) \right)$$

where:

- x_k is the parameter vector at iteration k,
- $\nabla f(x_k)$ is the subgradient of the objective function $f(x)$ at x_k,
- α_k is the step size or learning rate at iteration k,
- $\Pi_C(\cdot)$ is the projection operator onto the convex set C.

The projection onto C, denoted as $\Pi_C(x)$, is defined as follows:

$$\Pi_C(x) = \arg\min_{y \in C} \|y - x\|_2$$

This projection operation ensures that the updated parameter x_{k+1} remains within the feasible set C after each update.

Algorithm 2 Projected Subgradient Method

Choose initial feasible solution $x_0 \in X$
Choose stepsize rule (constant, diminishing, Polyak's, Modified Polyak's)
Initialize iteration counter $k = 0$
repeat
    Compute subgradient s_k of the objective function f at x_k
    Choose stepsize α_k according to the chosen stepsize rule
    Update x_{k+1} using the projection operation:
        $x_{k+1} = \text{Projection}_X[x_k - \alpha_k \cdot s_k]$
    Increment iteration counter: $k = k + 1$
until convergence
Output final solution x_k

4 Projected Stochastic Subgradient Method

The projected stochastic subgradient method extends the projected subgradient method to cases where the objective function is not necessarily differentiable and/or can only be evaluated on a subset of the data at each iteration.

Mathematical Formulation: Given a convex objective function $f(x)$ to minimize subject to the constraints $x \in C$, where C is a convex set, the projected stochastic subgradient method updates the parameter x iteratively as follows:

$$x_{k+1} = \Pi_C \left(x_k - \alpha_k g_k \right)$$

where:

- g_k is a stochastic subgradient estimate of $\nabla f(x_k)$ computed on a subset of the data at iteration k.

The rest of the notation remains the same as in the projected subgradient method. The projection operation $\Pi_C(\cdot)$ ensures that the updated parameter x_{k+1} remains within the feasible set C.

Algorithm 3 Stochastic Projected Subgradient Method

Choose initial feasible solution $x_0 \in X$
Choose stepsize sequence $\{\alpha_k\}$ such that $\sum_{k=1}^{\infty} \alpha_k = \infty$ and $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$
Initialize iteration counter $k = 0$
repeat
    Sample a stochastic subgradient s_k of the objective function f at x_k
    Choose stepsize α_k according to the chosen stepsize sequence
    Update x_{k+1} using the projection operation:
        $x_{k+1} = \text{Projection}_X[x_k - \alpha_k \cdot s_k]$
    Increment iteration counter: $k = k + 1$
until convergence
Output final solution x_k

5 Proximal Subgradient

The proximal subgradient method is utilized for minimizing nonsmooth convex functions, employing a proximal operator and taking proximal steps after computing subgradients at each iteration. In scenarios where an infinite number of subgradients exist, the choice of a specific subgradient significantly affects optimization. To mitigate this, a regularization term, often implemented via the proximal operator, is introduced to constrain subgradient selection. This regularization term ensures alignment with desired constraints, guiding subgradient choice towards solutions meeting additional criteria beyond optimality. Consequently, the regularization term shapes the optimization process, influencing subgradient selection and yielding more favorable outcomes.

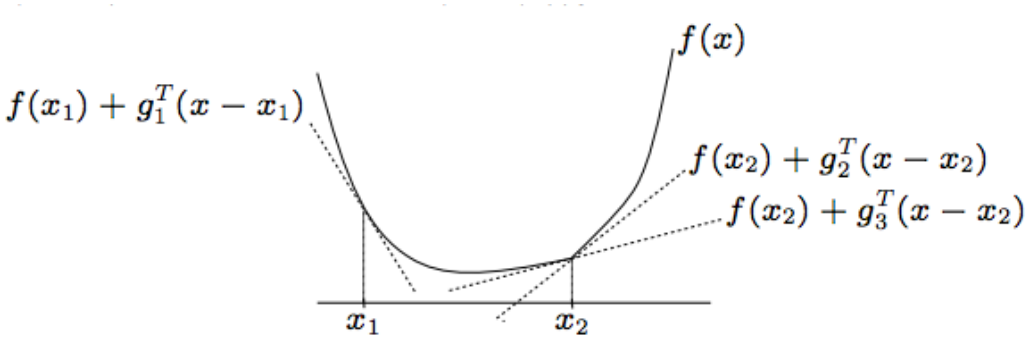

Figure 2: Multiple Subgradient

Mathematical Formulation: Given a nonsmooth convex objective function $f(x)$ and a smooth convex regularization term $g(x)$, the proximal subgradient method updates the parameter x iteratively as follows:

$$x_{k+1} = \text{prox}_{\alpha_k g}(x_k - \alpha_k \nabla f(x_k))$$

where:

- $\nabla f(x_k)$ is the subgradient of the objective function $f(x)$ at x_k,
- $\text{prox}_{\alpha_k g}(\cdot)$ is the proximal operator associated with the function $g(x)$ with a step size α_k.

The proximal operator $\text{prox}_{\alpha_k g}(y)$ is defined as follows:

$$\text{prox}_{\alpha_k g}(y) = \arg\min_x \left\{ g(x) + \frac{1}{2\alpha_k} \|x - y\|_2^2 \right\}$$

Algorithm 4 Proximal Subgradient Method with Explicit Proximal Operator

1: **Input:** Initial point x_0, step size sequence $\{\alpha_k\}$, parameter λ, maximum number of iterations K
2: **for** $k = 0, 1, 2, \ldots, K-1$ **do**
3: Compute subgradient s_k of f at x_k
4: Compute proximal update:
5: $x_{k+1} = \text{prox}_{\lambda \cdot g}(x_k - \alpha_k \cdot s_k) = \arg\min_x \left\{ g(x) + \frac{1}{2\lambda} \|x - (x_k - \alpha_k \cdot s_k)\|^2 \right\}$ ▷ Proximal step
6: **end for**
7: **Output:** Final iterate x_K

6 Regularized Proximal Subgradient

To incorporate regularization into the proximal subgradient method, we augment the original subproblem with an additional regularization term. Specifically, the proximal subproblem becomes:

$$\min_{x\in\mathbb{R}^n} f(x_k) + \nabla f(x_k)^\top (x - x_k) + \frac{1}{2\alpha_k}\|x - x_k\|_2^2 + \lambda_k \|x\|_1.$$

This problem is amenable to obtaining a closed-form solution. The iterative update involves computing the usual gradient step $x_k - \alpha_k \nabla f(x_k)$ and subsequently applying the soft-thresholding function $s_{\alpha_k\lambda_k}(\cdot)$ to each component. The soft-thresholding function $s_{\alpha_k\lambda_k}(\cdot)$, defined for $\mu > 0$ and $t \in \mathbb{R}$, is expressed as:

$$s_\mu(t) = \begin{cases} t + \mu & \text{if } t < -\mu, \\ t - \mu & \text{if } t > \mu, \\ 0 & \text{otherwise.} \end{cases}$$

This function acts component-wise on the result of the gradient step, providing a mechanism for introducing sparsity and promoting certain coefficients to zero. The resulting update scheme lies at the core of the proximal algorithm, commonly known as the Iterative Soft-Thresholding Algorithm (ISTA). This method is widely employed in signal and image processing due to its efficacy in solving problems with ℓ_1-regularization.

6.1 Iterative Soft-Thresholding Algorithm ISTA

This algorithm is derived from the proximal gradient method, a fixed-point iteration. The objective function being minimized comprises a quadratic term, representing the fidelity to the original signal, and a relaxed sparsity term, incorporating an 1 norm penalty for sparsity promotion. The proximal gradient update step in ISTA involves applying the soft-thresholding operator with a parameter determined by the trade-off between sparsity and fidelity. The algorithm iteratively refines an estimate of the sparse code by combining the soft-thresholding operator and the gradient of the quadratic term. The ISTA update is often presented with a fixed maximum step-size, ensuring convergence by controlling the Lipschitz constant. In summary, ISTA is a fixed-point iteration leveraging soft-thresholding and quadratic difference gradient operators, offering a versatile approach to sparse signal reconstruction with adjustable sparsity-fidelity

trade-offs. ¡

Algorithm:

Initialization: $x_0 \in \mathbb{R}^d$.

For $k = 0, 1, \ldots$ **do**

1. Compute the gradient of the smooth part $\nabla f(x_k)$.
2. Compute a steplength $\alpha_k > 0$.
3. Compute x_{k+1} component-wise through the following rule:

$$[x_{k+1}]_i = \begin{cases} [x_k - \alpha_k \nabla f(x_k)]_i + \alpha_k \lambda & \text{if } [x_k - \alpha_k \nabla f(x_k)]_i < -\alpha_k \lambda \\ [x_k - \alpha_k \nabla f(x_k)]_i - \alpha_k \lambda & \text{if } [x_k - \alpha_k \nabla f(x_k)]_i > \alpha_k \lambda \\ 0 & \text{if } [x_k - \alpha_k \nabla f(x_k)]_i \in [-\alpha_k \lambda, \alpha_k \lambda]. \end{cases}$$

end

Conclusion

In practical deep learning scenarios, it's not uncommon to encounter non-differentiable functions, which can pose a significant challenge for optimization. In such cases, resorting to subgradients becomes necessary to navigate through these complexities. However, the existence of an infinite number of possible subgradients introduces a crucial consideration: the selection of an appropriate method becomes paramount. The effectiveness of any chosen method is intricately linked to the specific problem being addressed.

Each problem may demand a nuanced approach, considering factors such as the structure of the function, the desired optimization goals, and computational constraints. Consequently, identifying the most suitable optimization technique becomes an essential task, ensuring that the chosen method aligns closely with the problem's characteristics. This process often involves a careful balance between computational efficiency and optimization efficacy.

Moreover, the selection of an optimization method can significantly impact the overall performance and convergence properties of the deep learning model. Thus, it's essential to explore and evaluate various optimization strategies, ranging from classic approaches to more advanced techniques tailored specifically for non-differentiable functions.

References

[1] Santambrogio, F. (2017). Euclidean, metric, and Wasserstein gradient flows: an overview. *Lectures in Mathematics, ETH Zürich.* Retrieved from Springerlink.com.
[2] Ambrosio, L., Gigli, N., & Savaré, G. (2008). *Gradient Flows in Metric Spaces and in the Space of Probability Measures*. Birkhäuser Verlag, Basel – Boston – Berlin. ISBN 3-7643-2428-7.
[3] S. Boyd, *Convex optimization, EE364B lecture notes*, Stanford.
[4] D. Bertsekas, *Convex optimization and algorithms*, 2015.
[5] A. Beck, *First-order methods in optimization*, Vol. 25, SIAM, 2017.
[6] S. Bubeck, *Convex optimization: algorithms and complexity*, Foundations and trends in machine learning, 2015.
[7] L. Vandenberghe, *Optimization methods for large-scale systems, EE236C lecture notes*, UCLA.
[8] B. Polyak, *Introduction to optimization*, Optimization Software, 1987.
[9] A. Nemirovski, A. Juditsky, G. Lan, A. Shapiro, *Robust stochastic approximation approach to stochastic programming*, SIAM Journal on optimization, 2009.