Smoothed Analysis: A Practical Approach for Practical Inputs

https://youtu.be/A6wp6LWuRbo

Sai Maruti Keerthana Ponnuru
MSCS, UFID: 3394-5082
University of Florida
Gainesville, Florida
ponnurus@ufl.edu

Shashank Reddy Boyapally
MSCS, UFID: 1923-9618
University of Florida
Gainesville, Florida
sboyapally@ufl.edu

***Abstract*— Smoothed analysis is a technique intended to study the behavior of algorithms under small perturbations of their input. It aims to provide insight into the practical performance of algorithms, which can sometimes deviate significantly from their worst-case or average-case behavior. We have attempted to provide an overview of the smoothed analysis approach, including its history, key results, and open questions. We describe several examples of smoothed analysis, including the simplex algorithm and k-means clustering. Overall, smoothed analysis is a catalyst in understanding the practical complexity of algorithms, and it holds great promise for advancing our understanding of the fundamental limits of computation.**

Keywords—Smoothed Analysis, K-means clustering, Simplex algorithms.

I. Introduction

The concept of typical instances refers to instances that are commonly encountered in practice. When an algorithm is run on a given instance, it can generate an output in a fast or slow manner, depending on the specific instance. Worst-case analysis aims to identify the instance that would take the longest time to process. However, most instances that occur in practice require considerably less time than these worst-case instances. Therefore, worst-case analysis may not provide a realistic estimate of the algorithm's performance in practice [1].

In contrast, average-case analysis considers the expected run time over a large set of random instances. However, this approach assumes that the instances are random and not structured. In practice, many instances may have some structure and may not conform to the average case. Smoothed analysis attempts to strike a balance between worst-case and average-case analysis by considering a family of instances that are close to the original instance but may have some random perturbations. This allows smoothed analysis to account for both the structured and random aspects of the instances encountered in practice, providing a more realistic estimate of the algorithm's running time.

II. Mathematical Introduction

The definition of smoothed analysis can be mathematically expressed as follows, Let I be an instance and T(I) be its running time. Let I_x denote an instance that is perturbed by x, where x represents a perturbation and $T(I_x)$ represents its running time. We define a probability space that consists of all possible perturbations. The magnitude of the perturbation is denoted by delta, which is a parameter that plays a crucial role in smoothed analysis[2].

Ω_δ - space of perturbations

In smoothed analysis, we consider a probability space of all possible perturbations for an instance and take the expectation over all of them to define the smoothed time for a specific instance.

$$T_\delta(I) = E_{x \in \Omega_\delta} T(I_x)$$

This process can be applied to all instances of a certain size, and for the worst instance, we take a perturbation of magnitude delta to account for some structure and randomness.

$$T(n,\delta) = \max_{I \in T_n} E_{x \in \Omega_\delta} T(I_x)$$

When delta equals zero, it represents the worst-case scenario, and when delta is huge, it corresponds to an average-case analysis where the instance is perturbed so much that it looks like an average instance. By choosing a value for delta between these two extremes, we can interpolate between worst and average case analysis, which we call smoothed analysis.

III. Literature Survey

A considerable amount of research has been conducted in this direction. One such pertinent work is "Smoothed Analysis: An Attempt to Explain the Behavior of Algorithms in Practice" that initiated the concept of smoothed analysis to address any limitations of worst-case analysis in algorithm design [3]. The authors proposed smoothed complexity as a measure of an algorithm's expected running time on slightly perturbed typical instances, which provides a more realistic estimate of its performance. The paper applied smoothed analysis to several algorithms, including the simplex algorithm for linear programming, and discussed its potential use in heuristics and machine learning. This work has had a significant impact on the analysis of algorithms and our understanding of their behavior in practice. Future work includes handling a larger dataset and using functions for phase detection. The paper gave a good understanding of where the research is headed.

Moving towards applications in the Machine Learning domain is "Smoothed Analysis of the k-Means Method" by David Arthur [4]. The paper presents a detailed analysis of the k-means clustering algorithm under smoothed

complexity, proving that it has polynomial smoothed run time on typical instances which measures its expected running time on instances that are slightly perturbed from the worst-case. The paper demonstrates that the complexity of the algorithm is polynomial, meaning that the algorithm is efficient on typical instances. He also provides experimental evidence to support his theoretical results, demonstrating that the algorithm performs well in practice on real-world datasets The paper highlights the usefulness of smoothed analysis for analyzing the behavior of algorithms in practice, and concludes with a discussion of future research directions. Overall, the paper provides valuable insights into the performance of the k-means algorithm on typical instances and its robustness to small perturbations in the input data.

III. Methodology

In the following section, we provide some instances of smoothed analysis, which are classified into two groups: Mathematical programming and Machine Learning.

A. Simplex Algorithm

A linear program with d variables and n constraints is represented as follows

$$\max zTx \quad \text{where} \quad Ax \leq y$$

Here, x is a vector in R_n representing the decision variables[9]. The linear program's data comprises the objective $z \in R_n$, constraint matrix A in $R_{m \times n}$, and vector y in R_m. A Simplex algorithm is used to procure the best solution for a linear programming problem[5].

Theorem 1: For All A, simplex method takes expected time polynomial (d,n,1/σ). Proof discussed in [6].

The shadow of a polytope refers to its projection onto the plane formed by two axes: 'z' and 't'. By using the shadow vertex pivot rule, we can estimate the expected sizes of these shadows. We can then count the facets by dividing them into N directions, which allows us to count the number of pairs in different facets [7]. This count is denoted by Pr[Different Facets], which is less than c/N. The smoothed analysis of the simplex method is based on the idea that, after applying some perturbation, most corners of the polytope will have an angle that is far away from being flat.

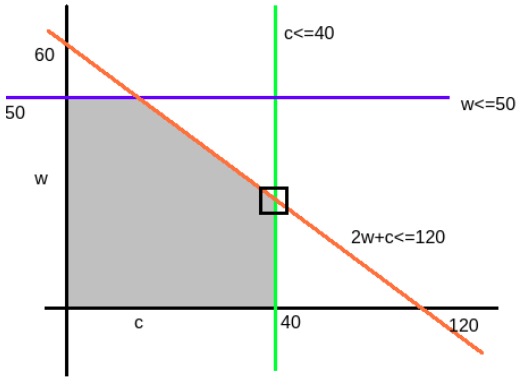

Figure 1. Simplex algorithm

The Figure 1 shows the fundamental of linear programming where the maximum of the variable 100w+150c is maximized following the constraints. The square shows a particular corner.

To use practical input data in an algorithm, it is necessary to perform two phases. Phase one is using arbitrary data with no randomness. In phase two, a little randomness is added to the data obtained in the previous phase. The modified input data can then be used as real-world typical data in the algorithm. Smoothed analysis is a method of analyzing algorithms where the input data is perturbed slightly by introducing random noise to the input. This helps in determining the robustness of an algorithm to small changes in the input data.

In the worst case scenario, the algorithm's complexity is represented as the maximum time taken to process an input set denoted as given in traditional analysis:

$$\text{WorstCase}_G(n) = \max_{y \in \Omega_n} T_G[y]$$

For the average case, the complexity is calculated individually for each case over the domain of all real values and is represented as:

$$\text{Average}^S{}_G(n) = \mathbf{E}_{y \in_S \Omega_n} [T_G(y)]$$

In smoothed analysis of simplex algorithms, the input data is slightly altered by adding a small amount of Gaussian noise, which is denoted by the variable σ. The algorithm's maximum complexity is then calculated for each perturbed input, where g is a random vector taken from the Gaussian distribution. The degree of perturbation to the input is determined by the parameter sigma (σ).

Definition 1: Assuming that algorithm A has an input domain of R_n, the smoothed complexity of A with σ-Gaussian perturbations can be expressed as [3]:

$$\text{Smooth}^{\sigma}{}_G(n) = \max_{x' \in [-1,1]^n} E_g [T_G(x' + g)]$$

The algorithm is provided with real-time input data with added randomness for each instance given by σ. The maximum is computed using input data associated with noise as y+g. This method helps in understanding the algorithm's complexity between the average and worst case scenarios, identifying the smoothed complexity of the algorithm.

B. K-means

This algorithm is a well-known clustering method known for its fast performance in practice. In a clustering problem, a group of entities is to be divided into subsets based on their similarity, typically shown as coordinates in a high-dimensional space R^d. During a single step, two scenarios may cause a substantial decrease in potential: When a cluster center undergoes a significant change in position or a data point is moved from one to the other & the point is considerably far from the bisector of the clusters, a potential drop in the system occurs. The bisector is the hyperplane that divides the 2 cluster centers. This potential drop can be quantified by considering the distance between the data point and the bisector, as well as the distance between the two cluster centers.

However, it has been found to possess worst-case running time, which is a problem for theoretical analysis. Smoothed analysis has been used to address this issue, but the current bounds are not satisfactory, as they are super-polynomial in the figure of data points. The run time of the k-means method after smoothed analysis is polynomial in n and $1/\sigma$. This means that the k-means method can be expected to run in polynomial time on an input data set that has been randomly perturbed.

The analysis of k-means algorithm in the smoothed setting is done using the potential function Ψ. Once the first iteration is completed, Ψ will always be limited by a polynomial in n and $1/\sigma$. Hence, if can estimate the decrease in Ψ in every iteration, then the k-means algorithm will terminate quickly [4].

To perform smoothed analysis on the k-means algorithm, a small amount of Gaussian noise is introduced to the input data points represented as high-dimensional vectors. The level of noise added is determined by a parameter σ. The algorithm is then executed on the perturbed input, and the objective function value is computed. This process is repeated several times for different perturbed inputs, and the average objective function value is calculated to determine the smoothed complexity of the algorithm.

The smoothed analysis of k-means has indicated that the algorithm can withstand small perturbations in input data. The smoothed complexity of the algorithm has been established as O(kn log n), k gives the no. of clusters and n the figure of data points. This is a notable improvement compared to the known NP-hard worst-case complexity of the algorithm. The smoothed analysis of k-means has offered valuable insights into its practical behavior. In several instances, the algorithm rapidly converges to an acceptable solution, despite its high worst-case complexity. Overall, smoothed analysis has contributed to a better understanding of the performance of k-means and has provided theoretical assurances about its behavior when input data is perturbed.

Definition 2(Bregman divergence):

Bregman divergence, also known as Bregman distance, is a way to quantify the dissimilarity between two points, which is defined using a function that is strictly convex.

$$D_f(a,b)=f(a)-f(b)-\langle \nabla f(b), a-b \rangle$$

Bregman distance(D_f), defines distance between two points a and b in a convex closed set C with non-empty interior C^o that belongs to the zone C.

Research has established upper bounds for k-means run time after smooth analysis along nearly random Bregman distances, which include poly(n√k, $1/\sigma$) and k^kd * poly(n, $1/\sigma$). However, can the polynomial bound be extended to cover Bregman distances in general is still uncertain [8].

IV Challenges

Although the findings from the conducted research indicate that the simplex algorithm generally operates within a polynomial time frame. Nevertheless, our comprehension of the simplex algorithm's performance is not done. Here we address issues with analyzing the shortcomings and the open challenges.

Although we have shown that the shadow-vertex algorithm has a polynomial smoothed complexity, resulting polynomial is quite large[9]. However, our analysis provides some understanding of why the algorithm should be efficient. The proofs rely on worst-case scenarios that are not likely to hold at once, and not effort put in optimizing the polynomial coefficients or exponents[10]. They chose to no more optimize the polynomial for two reasons: it would make the paper longer and more difficult to comprehend, and we believe that simplifying the analysis could lead to improved bounds[3]. Furthermore, our intuition about the algorithm's performance mostly comes from the shadow size bound, which is not as bad as the bound for the two-phase algorithm.

Some of the open challenges for future work include,

1. Finding the smoothed complexity of algorithms that do not have a known subexponential worst-case running time. It is still unclear whether such algorithms have a polynomial smoothed complexity or not.

2. To develop smoothed analysis techniques for more general models of computation. While many smoothed analysis results are available for algorithms that work on vectors and matrices, there are many other models of computation, such as graphs, strings, and trees, for which smoothed analysis could be applied, and developing techniques for these models is a challenge.

V Results & Conclusion

We see that though Worst-case complexity and and Average-case complexities are well known and good comparators for comparing the performance of algorithms, but they might not portray the actual performance of algorithms in the real-world. In order to solve this issue, we have taken a step towards smoothed analysis which majorly uses arbitrary inputs and then adds some noise to it, by which the input simulates a real-world data point. A similar case can be seen while working with the K-means algorithm. Though the theoretical approach shows a exponential time in the worst case analysis as a NP-hard problem, but in real world when we took an arbitrary input and applied perturbations we see we get a complexity of O(knlogn). Similar observations were seen when the simplex algorithm from Linear Programming was smooth analyses.

This eventually shows us that not all algorithms which are exponential in theoretical aspect are not usable in real world. For example- quicksort works worst when the input is sorted, but in real world one never sorts an already sorted list. Smoothed Analysis is a practical approach to measure the performance of algorithms in real world.

References

[1] Spielman, Daniel A., and Shang-Hua Teng. "Smoothed analysis: Motivation and discrete models." Algorithms and Data Structures: 8th International Workshop, WADS 2003, Ottawa, Ontario, Canada, July 30-August 1, 2003. Proceedings 8. Springer Berlin Heidelberg, 2003.

[2] Fourer, Robert. "A simplex algorithm for piecewise-linear programming I: Derivation and proof." Mathematical programming 33.2 (1985): 204-233.

[3] Spielman, Daniel A., and Shang-Hua Teng. "Smoothed analysis: an attempt to explain the behavior of algorithms in practice." Communications of the ACM 52.10 (2009): 76-84.

[4] Arthur, David, Bodo Manthey, and Heiko Röglin. "Smoothed analysis of the k-means method." Journal of the ACM (JACM) 58.5 (2011): 1-31.

[5] Spielman, Daniel A., and Shang-Hua Teng. "Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time." Journal of the ACM (JACM) 51.3 (2004): 385-463.

[6] Dadush, Daniel, and Sophie Huiberts. "A friendly smoothed analysis of the simplex method." Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing. 2018.

[7] Deshpande, Amit, and Daniel A. Spielman. "Improved smoothed analysis of the shadow vertex simplex method." 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05). IEEE, 2005.

[8] Manthey, Bodo, and Heiko Röglin. "Worst-case and smoothed analysis of k-means clustering with Bregman divergences." Algorithms and Computation: 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings 20. Springer Berlin Heidelberg, 2009.

[9] Spielman, Daniel A., and Shang-Hua Teng. "Smoothed analysis of termination of linear programming algorithms." Mathematical Programming 97 (2003): 375-404.

[10] Teng, Shang-Hua. "Smoothed analysis of algorithms and heuristics." Computing and Combinatorics: 11th Annual International Conference, COCOON 2005 Kunming, China, August 16–19, 2005 Proceedings 11. Springer Berlin Heidelberg, 2005.