CS 149 Project Proposal: How to Escape Saddle Points Efficiently

Devina Kumar and Barry Yang

Why this paper?

Finding local minima and maxima are essential to algorithms used in machine learning. One of the methods employed to find these minima and maxima is gradient descent. In a non-convex function, there may be numerous local minima and maxima, resulting in possible saddle points where an optimization algorithm may get "stuck." These saddle points can sometimes pose a problem because the solution that results from these saddle points may be suboptimal. The paper "shows that a perturbed form of gradient descent converges to a second-order stationary point in a number iterations which depends only poly-logarithmically on dimension" (Jin et al. 2017). Thus, the paper shows an interesting way in some cases of converging to a second-order stationary point without incurring much cost; this prevents that "stuck" behavior that is often associated with suboptimal solutions. Given the relevance of optimization algorithms in machine learning and deep learning, where much of the computer science world's focus is these days, this paper also explores methods of optimization techniques such as gradient descent in innovative ways to come up with solutions to well-known problems. Furthermore, the paper allows us to delve deeper into these complex, non-convex functions that are representative of many of the real-world data sets that a computer scientist may have to process and analyze.

Describe what you will do with this project.

One of the questions that this paper raises is what kind of limitations this algorithm has. Although the paper proposes new way to conceptualize the area around these saddle points, how applicable is this to saddle points in non-convex functions generally? Furthermore, under what conditions might this break down? One of the questions that the paper leaves open is how this algorithm could function with different kinds of gradient descent--accelerated gradient descent, for example. We aim to explore some of the possible limitations of this algorithm and examine how this algorithm can be transferred across optimization tools. In addition, we would like to explore some of the real-world applications of this algorithm. The paper gives the example of a symmetric low-rank matrix factorization problem. What other applications of this algorithm exist? And finally, we would like to examine the feasibility of implementing this algorithm. Is this algorithm manageable to implement, or is it simply unrealistic and better left in theory?