# No quantum speedup over gradient descent for non-smooth convex optimization

Ankit Garg
Microsoft Research

Robin Kothari
Microsoft Quantum

Praneeth Netrapalli
Microsoft Research

Suhail Sherif
Tata Institute of Fundamental Research

Gradient descent is a popular algorithm for minimizing functions in high-dimensional spaces. For some problems, such as convex function minimization, gradient descent provably converges to the function's global minimum. For other problems, such as finding good parameters of a deep neural network, gradient descent does not necessarily converge to a global minimum, and yet it has remarkable performance in practice.

Given the algorithm's popularity, it is interesting to ask if gradient descent can be sped up on a quantum computer. However, it's not obvious how to formalize this question since it's not clear what it means for a quantum algorithm to speed up a given classical algorithm. For example, does Shor's quantum algorithm for integer factorization speed up the best classical algorithm for the problem, or is it simply a different algorithm that solves the same problem?

One way to formalize the question *Can quantum computers speed up gradient descent?* is to consider a computational problem that is provably solved by gradient descent, and for which gradient descent is the optimal classical algorithm. We can then ask if quantum algorithms can solve this problem faster than gradient descent. The second condition is required since otherwise quantum computers would trivially outperform gradient descent by using the best classical algorithm. Fortunately, there is a canonical optimization task that is solved optimally by gradient descent: Black-box convex optimization of non-smooth functions.

**Black-box convex optimization.** In this problem we have to find an $\epsilon$-approximate minimum of a convex function $f : \mathbb{R}^n \to \mathbb{R}$ within a ball of radius $R$. In the black-box model, we do not assume any particular structure of the function $f$ (e.g., $f$ is a low-degree polynomial), and only assume that we have some efficient method of computing $f$ (i.e., we view $f$ as a black box).

Gradient descent and other common optimization algorithms additionally require black-box access to the gradient of $f$, or more precisely, since the gradient may not exist, some subgradient of $f$ (defined formally in the full version). We call this oracle that provides gradients (or subgradients) the *first-order oracle* and denote it by $\mathcal{FO}(f)$. We assume queries to $f$ and $\mathcal{FO}(f)$ cost the same because in many situations of interest, these functions are equally easy to compute due to the *cheap gradient principle*, explained in more detail in the full version. In short, in most models of computation, an algorithm for computing $f$ can be converted into one for $\mathcal{FO}(f)$ with similar cost.

In black-box convex optimization, it is also reasonable that the complexity of our algorithms depend on how quickly $f$ can change, since the value of $f$ at some point only constrains its values at nearby points if the function does not change too rapidly. The simplest assumption is that $f$ is Lipschitz with known Lipschitz constant $G$, but it may not be differentiable everywhere. (Note that any function that is convex everywhere must be Lipschitz in a ball of radius $R$.) This setting is also called the "non-smooth" setting, because another commonly studied assumption is that the function is smooth. We discuss the smooth case at the end of this abstract.

We are now ready to formally define the basic first-order convex minimization problem in the black-box setting. We use $B(x, R) \coloneqq \{y : \|x - y\| \leq R\}$ to denote an $\ell_2$-ball of radius $R$ around $x$.

**Problem 1** (First-order convex minimization)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ have Lipschitz constant at most $G$

on $B(\vec{0}, R)$. Then given $n$, $G$, $R$, and $\epsilon > 0$, the goal is to output a solution $x \in B(\vec{0}, R)$ such that

$$f(x) - \min_{y \in B(\vec{0},R)} f(y) \leq \epsilon, \tag{1}$$

while minimizing the number of queries to $f$ and $\mathcal{FO}(f)$.

Although the problem seems to involve 4 parameters, the parameters $G$, $R$, and $\epsilon$ are not independent since we can rescale the input and output spaces of $f$. Thus the complexity of this problem will be a function of $n$ and $GR/\epsilon$ only.

**Classical algorithms.** Gradient descent, or in this case *subgradient* descent, is a simple algorithm that starts from the starting point, say $\vec{0}$, and takes a small step in the opposite direction of the subgradient returned. Intuitively this brings us closer to the minimum since we are stepping in the direction where $f$ decreases the most. Since this is a constrained optimization problem, formally we use projected subgradient descent, which is subgradient descent with the added step of projecting the current vector back onto the ball $B(\vec{0}, R)$ after every step. Then we have the following [Bub15].

**Theorem 2.** *Projected subgradient descent algorithm solves Problem 1 using $(GR/\epsilon)^2$ queries.*

Observe that the query complexity of this algorithm, the number of queries made by the algorithm, is independent of $n$.[1] This is quite surprising at first and partly explains why gradient descent and its variants are popular in high-dimensional applications. More generally, we call such algorithms *dimension-independent algorithms*.

There also exist dimension-dependent algorithms for Problem 1 that work well when $n$ is small. For example, the center of gravity method solves this problem with $O(n \log(GR/\epsilon))$ queries [Bub15]. In this work we focus on dimension-independent algorithms and assume that $n$ is large (say, polynomially larger than the other parameters). When $n$ is large, it is known that we cannot improve over projected subgradient descent using any deterministic or randomized algorithm. We reprove the (well known) optimality of this algorithm among classical algorithms.

**Theorem 3.** *Any dimension-independent randomized algorithm for Problem 1 must make $\Omega((GR/\epsilon)^2)$ queries.*

This lower bound has been shown in prior works [NY83, WS17, BJL+19], but we believe our proof is simpler and the dimension $n$ in our proof seems to be smaller than that in prior works.

Our lower bound uses the following family of functions: For any $z \in \{-1, +1\}^n$, let $f_z(x_1, \ldots, x_n) = \max_{i \in [n]} z_i x_i$, where $n = O(1/\epsilon^2)$. We show that finding an $\epsilon$-approximate minimum requires $\Omega(n)$ queries to the oracles. We establish this by showing that with high probability, every query of a randomized algorithm only reveals $O(1)$ bits of information about $z$, but an $\epsilon$-approximate solution to this problem allows us to reconstruct $z$, which has $n$ bits of information.

**Quantum algorithms.** At first, it might seem that since gradient descent is a sequential, adaptive algorithm where each step depends on the previous one, there is little hope of quantum algorithms outperforming gradient descent. On the other hand, consider the hard family of functions described above. While this is a hard instance for classical algorithms, we show that there is a quantum algorithm that solves the problem on this family obtaining a quadratic speedup over any classical algorithm (and in particular, over gradient descent).

**Theorem 4.** *There is a quantum algorithm that solves Problem 1 on the class of functions that appear in the classical lower bound of Theorem 3 using $O(GR/\epsilon)$ queries to the oracle for $f$.*

[1] Of course, the time complexity of implementing this algorithm will be at least linear in $n$ since each query to either oracle requires us to manipulate a vector of length $n$.

Notably, unlike most quadratic speedups in quantum computing, the source of this quadratic speedup is not Grover's algorithm or amplitude amplification. Theorem 4 uses Belovs' quantum algorithm for learning symmetric juntas, which is constructed by exhibiting a feasible solution to the dual semidefinite program of the negative-weights adversary bound [Bel14].

Now that we have shown a quadratic quantum speedup on a family of instances known to be hard for classical algorithms, there is some hope that quantum algorithms may provide some speedup for Problem 1. Alas, our next result, which is our main result, shows that this is not the case.

**Theorem 5.** *Any dimension-independent quantum algorithm for Problem 1 must make* $\Omega((GR/\epsilon)^2)$ *queries.*

**Proof overview.** Our lower bound uses ideas from the recent lower bound against parallel randomized algorithms [BJL+19]. At a high level, the hard family of functions above does not work because although classical algorithms can only learn $O(1)$ bits of information per query, quantum algorithms can make queries in superposition and learn a little information about many bits simultaneously. We remedy this by choosing a new family of functions in which, with high probability, no matter what query the quantum algorithm makes, the oracle's response is the same. This allows us to control what the quantum algorithm learns per query. But now the instance is more complicated and the quantum algorithm learns $O(n)$ bits of information per query. Since the final output of the algorithm is a vector in $\mathbb{R}^n$, we cannot use the earlier argument that compared the information learned per query to the total information that needs to be learned. Instead, we use the venerable hybrid argument [BBBV97] to establish the lower bound.

**Related work.** Two recent works that appeared at QIP 2019 [vAGGdW20, CCLW20] are closely related to this work. They primarily study dimension-dependent quantum algorithms for this problem and related problems. Their techniques can be used to derive a lower bound in our setting as well, but it yields a suboptimal lower bound of $\Omega(GR/\epsilon)$.

**What about smooth functions?** Smooth functions are another natural class of functions that are well studied in optimization and it is natural to ask the same question for smooth functions. For our second result, we completely characterize the quantum query complexity of this problem as well.

Formally, the problem considered is the same as Problem 1, but instead of assuming that $f$ is $G$-Lipschitz in $B(\vec{0}, R)$, we assume that $f$ is $L$-smooth in $B(\vec{0}, R)$. This means $f$ is differentiable in $B(\vec{0}, R)$ and for all $x, y \in B(\vec{0}, R)$, we have $\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$.

For smooth functions, gradient descent is not the optimal classical algorithm. Indeed, gradient descent solves the problem with $O(LR^2/\epsilon)$ queries, whereas the best classical algorithm is accelerated gradient descent, which makes $O(\sqrt{LR^2/\epsilon})$ queries [NY83, Nes83]. As before, among dimension-independent algorithms, it is known that this is the best complexity achievable [NY83].

We then ask the same question: Can quantum computers do better? Our result is negative again. We prove that even quantum algorithms must make $\Omega(\sqrt{LR^2/\epsilon})$ queries in the worst case.

Our lower bound for the smooth case builds on the construction in Theorem 5. The functions that appear in that lower bound are not smooth (indeed, they are not even differentiable everywhere), so we cannot simply use the same functions. But we use a method called "Moreau smoothing" [Mor65, Yos88] that transforms a Lipschitz function into a smooth function, which is an idea used in prior works as well [GN15, DG19]. But we cannot smooth the function too much, otherwise the minimum will become too easy to find, and so there's a balance to be struck to construct a proper lower bound instance. We show that with the appropriate amount of smoothing, we get a family of functions for which the proof strategy used in the non-smooth case works with some additional work. Thus there is no quantum speedup over accelerated gradient descent for minimizing smooth convex functions.

## References


[vAGGdW20] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Convex optimization using quantum oracles. *Quantum*, 4:220, January 2020. doi:10.22331/q-2020-01-13-220. [p. 3]

[BBBV97] Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997. doi:10.1137/S0097539796300933. [p. 3]

[Bel14] Aleksandrs Belovs. Quantum algorithms for learning symmetric juntas via adversary bound. In *Proceedings of the 2014 IEEE 29th Conference on Computational Complexity (CCC 2014)*, CCC '14, page 22–31, 2014. doi:10.1109/CCC.2014.11. [p. 3]

[BJL+19] Sébastien Bubeck, Qijia Jiang, Yin Tat Lee, Yuanzhi Li, and Aaron Sidford. Complexity of highly parallel non-smooth convex optimization. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*, pages 13900–13909, 2019. URL: http://papers.nips.cc/paper/9541-complexity-of-highly-parallel-non-smooth-convex-optimization. [pp. 2, 3]

[Bub15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Found. Trends Mach. Learn.*, 8(3–4):231–357, November 2015. doi:10.1561/2200000050. [p. 2]

[CCLW20] Shouvanik Chakrabarti, Andrew M. Childs, Tongyang Li, and Xiaodi Wu. Quantum algorithms and lower bounds for convex optimization. *Quantum*, 4:221, January 2020. doi:10.22331/q-2020-01-13-221. [p. 3]

[DG19] Jelena Diakonikolas and Cristóbal Guzmán. Lower bounds for parallel and randomized convex optimization. In *Proceedings of the Thirty-Second Conference on Learning Theory*, volume 99, pages 1132–1157. PMLR, 2019. URL: http://proceedings.mlr.press/v99/diakonikolas19c.html. [p. 3]

[GN15] Cristóbal Guzmán and Arkadi Nemirovski. On lower complexity bounds for large-scale smooth convex optimization. *Journal of Complexity*, 31(1):1 – 14, 2015. URL: http://www.sciencedirect.com/science/article/pii/S0885064X14000831, doi:10.1016/j.jco.2014.08.003. [p. 3]

[Mor65] Jean-Jacques Moreau. Proximité et dualité dans un espace hilbertien. *Bulletin de la Société mathématique de France*, 93:273–299, 1965. [p. 3]

[Nes83] Yu. E. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Sov. Math., Dokl.*, 27:372–376, 1983. [p. 3]

[NY83] Arkadiĭ Nemirovsky and David Borisovich Yudin. *Problem complexity and method efficiency in optimization*. Wiley, 1983. [pp. 2, 3]

[WS17] Blake Woodworth and Nathan Srebro. Lower bound for randomized first order convex optimization. *arXiv preprint arXiv:1709.03594*, 2017. arXiv:1709.03594. [p. 2]

[Yos88] Kosaku Yosida. *Functional analysis*, volume 123. springer, 1988. [p. 3]