Nicholas Jaber
April 20th 2022
ECE 523
Dr. Iman Marvian

Quantum and Hybrid Stochastic Gradient Descent Algorithms

Gradient descent is the process of iteratively seeking the global minima of a high dimensional vector space. These iterations are known as steps and can be imagined as walking downhill to reach the lowest point of a valley. The purpose of this exercise is to minimize a function's weight. Minimizing function weight is a useful tool in many NP problems. These include, but are not limited to: machine learning, neural networks and least square regressions. Finding the global minima function weight for each of these use cases indicates the most similar state to an optimal solution. These techniques are referred to generally as optimization problems, because their efficacy relies heavily on the optimization of these computationally intensive processes that we will further describe later in the paper.

Direct computation of high order polynomials is often far less efficient than the highly iterative process of estimating global minima. Estimating the global minima of these functions has an NP-like runtime. This means that as problem size increases linearly, computational resources increase exponentially. For example, for each additional factor considered in a neural network weighting, the computational requirement to identify a similarly proximal state to the optimal solution doubles. Because of this rapid doubling of runtime, one must compromise their effectiveness by minimizing considered factors and branch connections. These minimizations lead to significantly less accurate estimation models.

Each of the previously mentioned use cases for gradient descent utilizes a variation of Newton's method. This is because Newton's method is the gold standard of efficient and generalizable gradient descent. This method uses the first and second spatial derivative to estimate the position of the nearest local minima. Newton's method is described in equation 1.

$$x^{(t+1)} = x^{(t)} - \eta H^{-1} \nabla f(x^{(t)}) \quad (1)$$

$\nabla f(x^{(t)})$ refers to the first order spatial derivative of the objective function, referred to as the gradient in future references. H refers to the Hessian matrix, a square matrix of each possible second order spatial derivative of the objective function, where $H_{ij} = d^2f / dx_i\, dx_j$. The relations of these variables are shown in equations 2-3.

$$\nabla f(x) = D(x)\, x \tag{2}$$

$$H\,(f(x)) = H_1 + D \tag{3}$$

η is the hyper parameter, also called the machine learning rate, which relates how aggressively one seeks the local minima as pictured by the descent step size. A highly aggressive approach is likely to overshoot the estimation in quickly changing terrain; a less aggressive approach will take a large number of iterations. As a result, η must be tailored to a specific problem's local terrain change rate. η must always be greater than 0, so that the direction of travel is in the opposite direction of the upward slope, described by $H^{-1} \nabla f(x^{(t)})$. By iteratively repeating and stepping towards the local minima and refining our estimations, one may estimate the state of the local minima. By repeating this iterative process recursively, one may identify many local minimas and compare the nearest local minima to many other local minima states. By doing so, one may acquire the estimated global minima. This recursive iteration is extremely inefficient for the high dimensional vector spaces needed for even relatively simple models. When scaling to large problem types, one needs an algorithm that rethinks the computational process to limit its run-time requirement.

Some simple versions of gradient descent such as quadratic polynomials can be easily computed with P-like computational resources, but the vast majority of useful applications of gradient descent depend on high order polynomials. As a result, we must rethink a generalized method to compute gradient descent. This algorithm must find a method to compute NP gradient descent problems with P-like computational time and hardware resources. Optimizing NP problems by using quantum speedups can enable P-like runtime scaling. This is because quantum computers offer exponential scaling, meaning that a linear expansion of computational resources can lead to an exponential expansion of computational complexity.

Rebentrost et al.'s paper Quantum gradient descent and Newton's method for constrained polynomial optimization focuses on non-quadratic convex or even

non-convex optimization problems. Rebentrost solves these problems by using phase estimation quantum matrix inversion techniques. The concept is to generate a large number of quantum state copies, of which a small portion is consumed in combination with the Hessian matrix and gradient vector from a local region of the objective function for each iterative step. Because of this constant overwriting rate, only low iteration optimization techniques would be practical on near term NISQ (Noisy Intermediate Scale Quantum) computers. As a result, this technique may most likely be implemented for local gradient descent searches, in combination with an unlocalized classical iteration technique. This can be done using quantum state exponentiation to estimate the gradient vector and Hessian matrix of the objective function. First, we will discuss what a quantum Newton's method looks like for high dimensional sparse polynomial objective functions. Then, we will discuss how this process can be expanded to include a small number of inhomogeneities.

To describe the process used to generate this quantum speedup, we must first define a few key variables and assumptions of the problem type. If the objective polynomial is of degree 2p and dimensionality N, then it contains N^{2p} coefficients. As previously mentioned, we can only optimize this process for sparse functions, wherein very few of these possible coefficients are nonzero. In the particular case when p = 1, then $H^{-1} \nabla f(x) = D(x)^{-1} D(x) x = x$.

This concept is effective for objective functions that are homogeneous polynomials of even degree high dimensional vector spaces and this even works with a small number of monomial inhomogeneity polynomials. Particularly, Rebentrost focuses on objective functions, which are multivariate homogeneous polynomials of even degrees under spherical constraints, and sufficiently low sparsity. Low sparsity allows for low computationally intensive matrix inversion.

Spherical constraint means that x values are represented as a quantum state, which is normalized to $x^T x = 1$. We assume that classical algorithms can acquire an x_0 estimate, which is reasonably proximal to the local minimum. We assume that the Hessian of the objective polynomial is positive semidefinite. This means that the Hessian is symmetric, all eigenvalues are real and greater than zero, and there is an orthonormal basis which contains an eigenvector of the Hessian. Additionally, objective

polynomials must be smooth and Lipschitz continuous. This ensures that the objective function has no saddle-points and is convergent to the local minimum for the monotonic terms.

To estimate the local minima, one must first analytically compute the local gradient vector and Hessian matrix. To do so, one may use the lie product formula to calculate $e^{iH\Delta t}$ as shown in equation 4.

$$e^{iH\Delta t} = e^{iH1\Delta t}\, e^{iD\Delta t} + O(p^4 \Lambda^2 \Delta t^2) \quad (4)$$

Here, Λ represents the sparsity of the polynomial. Next, one must simulate the time evolutions of the gradient and Hessian. Then, one must add the step vector to each copy of the state $|x^t>$. $|x^{t+1}>$ is proportional to $(|x^t> - \eta^t D^t |x^t>)$ with accuracy $\varepsilon^{t+1} = O(\eta^t \varepsilon^t_D + \varepsilon^t)$. Then, one must invert and apply Hessian and gradient as described in equation 1. Finally, one must measure each of the two ancillary qubits in the $|yes> = (1/\sqrt{2})(|0>_g + i|1>_d)$ and $|no> = (1/\sqrt{2})(i|0>_g + |1>_d)$ basis.

To complete this task, we make a few assumptions about the quality and size of our quantum computer. We assume that we are able to predictably and efficiently prepare quantum states. We assume that the amplitude information for x_0 can be stored as a normalized value in the z coordinate of a qubit efficiently. This computer should be able to operate on $2*p*\log_2 N + n_x$ qubits. Where n_x is calculated based on how many steps from the local minima x_0 is and how many states are consumed in each iteration. By utilizing 2 ancillary qubits, one may generate the state $(1/\sqrt{2})|x^{(t)}>|0>_g 1>_d$. The success of the ancillary |yes> and |no> basis measurements are shown using equations 2-3 and 5-6. Collectively, this gives us the probability of finding the state $|yes>|1>_d$ represented by $P_{yes,1}$.

$$1/4 < (C^{t+1}_D)^2 = 1 - 2\eta <x^t|D|x^t> + \eta^2 <x^t|D^2|x^t> \quad (5)$$

$$1/16 < P_{yes,1} = 1/4\, (C^{t+1}_D)^2 \quad (6)$$

Because of this, the maximum effective number of iterations is $O(1/P_{yes,1}) = O(1)$. In other words, this quantum Newton's method gradient descent algorithm is P-like and linearly related to problem size. This process is only bounded by the error accumulation. The next step error is $O(\varepsilon^{t+1}) = O(\varepsilon^t + \eta \varepsilon^t_D)$. Using phase estimation, ancilla rotation and measurement, one can obtain the total resource requirement τ as seen in equation 7.

$$\tau = O(1/\varepsilon^{t}_{D}) \quad (7)$$

For each step, we require 2p log N swap operations to be completed successfully. Additional errors occur from sampling accuracy of the Hessian. The errors in the system are due to imperfect state generation, time dependent decoherence, sampling errors and imperfect gate operation. Combined, these features are called the aggregate error rate shown in equation 8.

$$\varepsilon^{t} = \varepsilon^{t-1} + \eta^{t} \varepsilon^{t}_{D} \quad (8)$$

Rather than calculating the exact value of the Hessian matrix and gradient vector, stochasticity can be created by repeated measurements. One can over sample gradient vector and hessian matrix to lower sampling error rates. This has been shown to lead to much faster learning rates for neural networks. Additionally, stochasticity increases generalizability, because over sampling leads to more function agnostic error rates. This leads to more predictable gradient descent. Using classical gradient descent algorithms, it is prohibitively computationally intensive to compute exponentiation. Using established quantum exponentiation algorithms, one may acquire a quantum advantage. This advantage is only available to sparse Hessian matrices because a major computational burden in exponentiation is matrix inversion. Doing so in a sparse matrix can allow for efficient inversion of the Hessian. The probability of successfully navigating this Hessian matrix calculation is $P_{yes,1,1}^{nwt}$ as shown in equation 9-10. $P_{yes,1,1}^{nwt}$ represents the probability of acquiring the state $|yes\rangle|1\rangle_d|1\rangle_g$ using quantum Newton's method.

$$(C^{t+1}_{H})^{2} = 1 - 2\eta \langle x^{t}|H^{-1} D|x^{t}\rangle + \eta^{2} \langle x^{t}|D\, H^{-2} D\,|x^{t}\rangle \quad (9)$$

$$1/16 < P_{yes,1,1}^{nwt} = 1/4\, (C^{t+1}_{H})^{2} \quad (10)$$

The phase estimation estimate the gradient with accuracy ε_{nwt} requires $O(p^3\Lambda^2/\varepsilon^{3}_{nwt})$ copies of $|x^t\rangle$. The phase estimation estimates the Hessian with accuracy ε_{nwt} requires $O(p^5\Lambda^2/\varepsilon^{3}_{nwt})$ copies of $|x^t\rangle$. For T iterations, one would need $O(p^{5T}\Lambda^{2T}/\varepsilon^{3T}_{nwt})$ copies of $|x^t\rangle$.

To expand this process from estimation of high order homogeneous polynomials to estimations of high order homogeneous polynomials with a small number of odd degree inhomogeneous terms; we must perform additional vector addition to account for the inhomogeneous terms. Additionally, when calculating the gradient and Hessian function, we must include additional inner and outer product terms, to account for time

evolution of the objective function. This process greatly slows down the computation, and limits the number of inhomogeneities. In all, current estimates can obtain highly accurate estimations of a global minimum in ~5-20 steps. While classical algorithms may fail in high dimensional vector spaces, because of the increased prevalence of saddle points. Quantum gradient descent can accommodate saddle points by taking the absolute value of each Hessian element.

The next technique we'll explore was presented by Sweke et al. in their paper, Stochastic gradient descent for hybrid quantum-classical optimization. This paper discusses the modern approaches to optimize the stochastic model and hyper-parameter using heuristics. These techniques are useful for the implementation of existing algorithms on NISQ computers. These techniques include: VQE (Variational Quantum Eigensolvers), QAOA (Quantum Approximate Optimization Algorithms) and quantum classifiers. Particularly, using classical computers to identify and refine the aforementioned variable quantities allows for more effective implementation of these algorithms on NISQ computers with very limited computational resources.

There are diminishing returns from shortening the step size as seen in the magnitude of the hyper-parameter, η. Similarly, There are diminishing returns from increasing the number of Hessian and gradient measurements. A large number of measurements may increase the accuracy of the subsequent step, however it may lead to an inordinate expansion of computational resources. Thus in aggregate slowing down computation. As shown in figure 1, optimizing the heuristics of the hyper parameter and utilizing information on previous gradient vectors and Hessian functions, we can get high shot-like performance with very few shots. Thus, these hybrid quantum-classical optimizations can massively increase computational efficiency. Additional efficiency benefits can be seen when sampling the objective function's value more infrequently relative to the frequency of shots as seen in figure 2. Finally, the use of more classically intensive processes, such as increasing stochasticity of gradient and Hessian measurements can lead to a vast decrease in computational cost, as seen in figure 3. In all, hybrid classical-quantum optimization algorithms will likely become the predominant method to compute global minimas, while quantum computers are not sufficiently powerful to solve these optimization problems on their own.

A major development in the modern evolution of quantum gradient descent was the novel algorithm by Harrow Hassidim Lloyd known as HHL. HHL can only apply a P-like speedup to machine learning when the objective function is well conditioned and has sparsity poly-logarithmic in the dimension. HHL's runtime is on the order of $O(\Lambda^2 \kappa^2 / \varepsilon)$. Here, κ represents the condition number of the function and ε represents the approximation error. Well conditioned and sparse functions are very atypical in real world conditions; as a result, HHL is not as widely applicable as we would like. Consequently, Kerenidis et al. in Quantum gradient descent for linear systems and least squares shows how to implement QRAM (Quantum Random Access Memory) data structures to broaden the applicability of HHL to real world conditions. The QRAM data structure is shown in equation 11.

$$O(\mu \kappa^2 / \varepsilon) \geq O(\Lambda \kappa^2 / \varepsilon) \quad (11)$$

Here, μ is dependent on the value of p, with higher p values leading to μ increasing relative to Λ. This includes not well conditioned, higher order and denser objective functions. This QRAM algorithm uses stochasticity in the same methodology as in Rebentrost's paper. Without stochastic gradient descent, estimates for needed QRAM range as high as 10^7 qubits. This is extremely far from the current state of the art and hence, infeasible.

The advantage of QRAM is the ability to reliably store states over time. Doing so prevents decoherence and allows for fewer qubits to do the same amount of work. As mentioned in the previous paper, many qubits are needed and consumed for each iteration of the quantum Newton's method. However, the QRAM model allows for the same qubits to be utilized repeatedly, meaning that one only needs the number of qubits suitable to complete the most computationally intensive iteration.

Similar to quantum Newton's method, QRAM can solve semidefinite positive symmetric polynomials. One method implemented with QRAM is SVE (singular value estimation). SVE allows for the generalization of QRAM data structures to solve dense linear systems. SVE uses phase estimation and eigenvectors from the matrix representation of the objective polynomial to compute the next step.

Using SVE, we can improve the runtime to solve linear systems with runtime $O(\alpha \mu \log(\alpha^2 / \varepsilon))$. Here, α represents the number of steps needed to reach a local

minimum. With an optimal structure, one can have runtumes as low as $O(\mu / \varepsilon)$. Currently QRAM devices are not sufficiently reliable for the storage of even a small number of qubits, let alone to serve as a data structure for at least 10^6 qubits. Consequently, these designs are focused on their application to far future quantum storage devices.

In total, it seems that the most feasible evolution of quantum gradient descent algorithms can be divided into three major eras, barring any additional major developments. The first era would be the development of hybrid classical-quantum gradient descent algorithms, designed for sparse homogenous high order polynomials. These computations would be carried out on a small scale quantum computer in conjunction with assistance from classical computer optimization. The second era would be the development of purely quantum Newton's method, designed for sparse polynomials with high order homogeneity and a small number of low order inhomogeneities. These computations would be carried out exclusively on NISQ computers. The final era would be to implement QRAM data structure based algorithms. These algorithms could only be implemented in the far future development of quantum devices, but would allow for the computation of dense polynomials. The future of quantum gradient descent is a rapidly advancing field, which will become increasingly impactful as we evolve quantum hardware from experiment to industrial application.

Figures

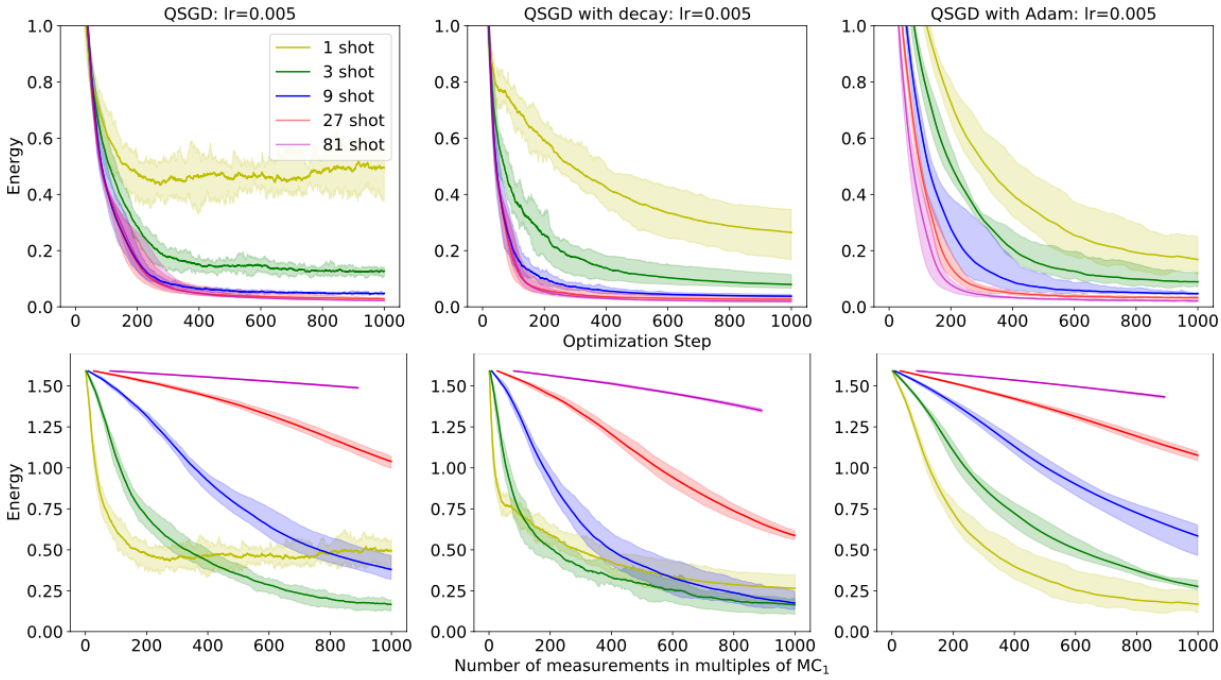

Figure 1 The top row of graphs indicates the energy as an analog to computer runtime with respect to the number of steps to complete a VQE problem for the various number of shots: 1, 3, 9, 27, 81. Each experiment is run 8*shots times. The bottom row of graphs indicates the energy with respect to the number of sampling measurements used to estimate the local gradient and Hessian value. Column 1 represents the default optimization scheme. Column 2 represents a learning rate decay, which decreases η by a factor of 2 for the last 20 steps to more precisely refine the estimation. Column 3 represents the Adam optimizer scheme, which uses knowledge of the previous gradient and Hessian value to estimate the update values with fewer stochastic sampling shots.

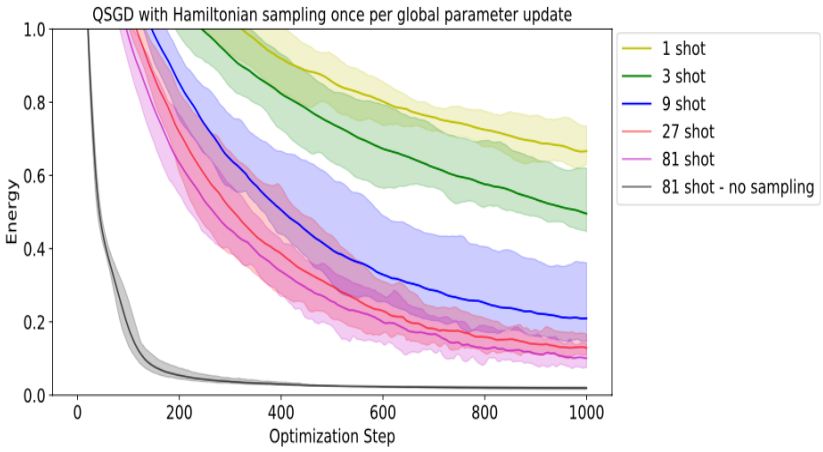

Figure 2 The computational cost to complete a VQE problem with single hamiltonian sampling per step. This experiment was carried out 8 times for each of the shot values: 1, 3, 9, 27, 81, and exact. This graph indicates that regardless of shots, VQE solutions always are more computationally intensive when there are fewer optimization steps.

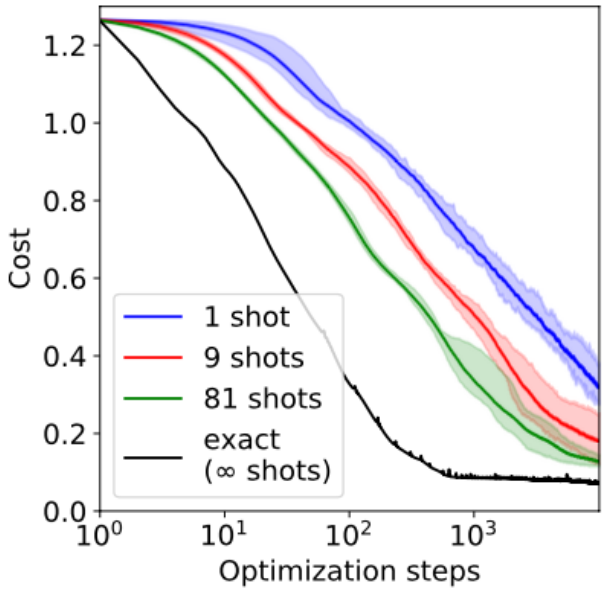

Figure 3 Shows the computational cost to complete a VQE problem. This experiment was carried out 8 times for each of the shot values: 1, 9, 81, and exact. This graph indicates that regardless of shots, VQE solutions always are more computationally intensive when there are fewer optimization steps.

Bibliography

1. Rebentrost, P., Schuld, M., Wossnig, L., Petruccione, F. & Lloyd, S. Quantum gradient descent and Newton's method for constrained polynomial optimization. *New J. Phys.* **21**, (2019).

2. Kerenidis, I. & Prakash, A. Quantum gradient descent for linear systems and least squares. *Phys. Rev. A* **101**, 022316 (2020).

3. Sweke, R. *et al.* Stochastic gradient descent for hybrid quantum-classical optimization. *Quantum* **4**, (2020).