Gradient Descent in VQA

Variational Quantum Algorithms · Module 6 — Optimization Dynamics

Node 6.1

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Optimization Dynamics **Node:** 6.1

This is the **opening node** of Module 6 — the module on optimization dynamics. With cost landscapes characterized (Module 3) and barren plateaus understood (Module 4), the question now becomes: **how do we actually move parameters to reduce the cost?**

The simplest answer, and the historical starting point of VQA optimization, is **gradient descent**: at each step, take a small move in the direction of steepest descent. This node walks through what gradient descent looks like in the VQA context, what makes it work, what makes it fail, and how it sets the baseline that all other optimizers are compared against.

The Update Rule

Vanilla gradient descent updates parameters according to

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \, \nabla C(\boldsymbol{\theta}_k),$$

where $\eta > 0$ is the **learning rate** (or step size).

For VQA, the gradient is estimated by parameter-shift (Section 3.8), so each step requires $2P$ cost evaluations for a P-parameter ansatz. Each cost evaluation is itself a noisy quantity (Section 3.7), so what the optimizer actually computes is

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta \, \widehat{\nabla C}(\boldsymbol{\theta}_k),$$

where $\widehat{\nabla C}$ is a stochastic estimator with mean equal to the true gradient and variance set by the shot budget.

This is **stochastic gradient descent (SGD)** in the strict sense — the gradient is unbiased but noisy. The convergence theory is the textbook stochastic-approximation theory (Robbins-Monro 1951), which Section 6.5 will treat in more detail.

The Three Hyperparameters

A naive gradient descent on a VQA has three knobs:

1. Learning rate η. Too small: convergence is slow. Too large: the optimizer overshoots minima and oscillates or diverges. Optimal η depends on the local Hessian (Section 3.5) — specifically, you need $\eta < 2/\lambda_{\max}$ where $\lambda_{\max}$ is the largest Hessian eigenvalue.

2. Shot budget per evaluation N. Too small: gradient noise overwhelms signal, optimizer does a random walk. Too large: each step costs more than necessary, total budget is wasted on precision that wasn't needed.

3. Stopping criterion. When to halt. Common choices: cost stops decreasing, gradient norm below a threshold, fixed budget exhausted.

These three are interrelated in non-obvious ways. For example, doubling N reduces noise by $\sqrt{2}$, which lets you take larger steps (effectively double η) with the same robustness. So "shots per step" and "step size" trade off against each other in a way that affects the overall budget allocation.

The thesis's "stochastic objective generator" framing (Module 1.8) makes this explicit: the optimizer is making decisions under noise, and N sets the noise level it sees. Tuning N is part of the optimizer's job, not just a fixed hyperparameter.

Why Gradient Descent Is The Default

Gradient descent is the default first-attempt optimizer for VQAs for several reasons:

1. It's local. You only need information at the current point — no global landscape knowledge required. This matches the VQA context, where global landscape information is unavailable.

2. It's cheap per step. $2P$ cost evaluations per step, no Hessian, no extra structure. This is the minimum information you can use and still be doing first-order optimization.

3. It has well-understood theory. For convex landscapes, convergence rate is $O(1/k)$. For L-smooth landscapes, GD with $\eta < 1/L$ is guaranteed to monotonically decrease cost (in expectation).

4. It's a clean baseline. Every fancier optimizer is "GD plus something." You can ask whether the "plus something" is worth the cost. This makes GD the reference point.

5. It works on small problems. For $n < 10$ qubits and shallow ansätze, gradient descent on a VQE problem typically converges in $O(100 - 1000)$ iterations to reasonable accuracy. It's not the fastest possible optimizer, but it's adequate.

Why Gradient Descent Often Fails On VQA

The list of GD's pathologies in VQA is long:

1. Barren plateaus (Module 4). At random initialization in a deep ansatz, the gradient is exponentially small in qubit count. GD makes essentially zero progress per step — the noise dominates the signal. No amount of patience helps; GD does a random walk.

2. Step size sensitivity. The optimal step size depends on the local Hessian, which varies wildly across parameter space. A single global η is rarely optimal everywhere. Too large: oscillations. Too small: glacial progress in the flat regions.

3. Curvature mismatch. GD uses the same step size in all directions. If the landscape has very different curvature in different directions (long, narrow valleys), GD zig-zags inefficiently. This is the "ill-conditioned Hessian" problem from classical optimization.

4. Coordinate-system dependence. GD's behavior depends on how you parameterize the ansatz. If you reparameterize (e.g., $\theta \to 2\theta$), the trajectory changes in a non-trivial way. This is "lack of reparameterization invariance" and it is the central problem the **natural gradient** (Section 6.2) addresses.

5. No escape from local minima. GD converges to whatever critical point is closest to the initialization. For non-convex VQA landscapes (which are most of them), this means GD is at the mercy of the starting point.

6. Stagnation in noise. Once the gradient drops below the shot noise floor, GD effectively stops. Even if there is a meaningful descent direction further away, GD can't get there because it has lost the signal.

These failure modes motivate every other optimizer in Module 6. Quantum natural gradient (6.2) addresses #4. Adam (6.3) addresses #2 and #3. Stochastic optimization theory (6.4, 6.5) addresses #6. Population methods (6.7-6.8) address #5.

A Cautionary Worked Example

Consider the simplest possible VQA: a single qubit, $|\psi(\theta)\rangle = R_Y(\theta)|0\rangle$, $H = Z$, $C(\theta) = \cos(\theta)$, with global minimum at $\theta = \pi$.

Setup. Start at $\theta_0 = 0.1$. Learning rate $\eta = 0.5$. $N = 100$ shots per evaluation.

Run. - $k = 0$: $C(0.1) \approx 0.995$. Gradient $\approx -\sin(0.1) = -0.0998$. Update: $\theta_1 = 0.1 + 0.5 \cdot 0.0998 = 0.150$. - $k = 1$: $C(0.15) \approx 0.989$. Gradient ≈ -0.149. Update: $\theta_2 = 0.15 + 0.075 = 0.225$. - $k = 2$: $C(0.225) \approx 0.975$. Gradient ≈ -0.223. Update: $\theta_3 = 0.225 + 0.111 = 0.336$.

The optimizer is making progress, but slowly. The gradient grows as it moves away from the starting point (because $C''(\theta) = -\cos(\theta)$ is positive in this region, i.e., the landscape is convex), so successive steps get larger.

Eventually it crosses $\theta = \pi/2$ where the gradient peaks at magnitude 1 (the inflection point), and then enters the descending side of the cosine. At $\theta \approx \pi$, the gradient is again ~ 0 and the optimizer slows down — this is approaching the minimum. With $\eta = 0.5$, the optimizer converges to within 10^{-3} of π in about 30 iterations.

Now repeat with shot noise. $N = 100$ gives gradient noise of order $\sin(\theta)/\sqrt{100} = 0.1\sin(\theta)$. When the gradient is ~ 0.1 (early or late in the trajectory), the noise is comparable to the signal and the optimizer's progress becomes erratic. At the minimum, the gradient and noise are both ~ 0.001, so the optimizer wanders around the minimum forever rather than converging exactly.

Now scale to 10 qubits with random init in a 5-layer HEA. Initial gradient is $\sim 10^{-2}$. Shot noise at $N = 100$ is ~ 0.1. Signal is below noise. **GD does a random walk and never converges.**

This progression — works in 1D, breaks at 10 qubits — is the canonical reason VQA optimizers need to do something more sophisticated than vanilla gradient descent.

What Gradient Descent Looks Like To The Stochastic Objective Generator

In the thesis's framing (Module 1.8), the optimizer interacts with a stochastic objective generator:

- The generator takes a parameter vector $\boldsymbol{\theta}$ as input.
- It returns a noisy scalar $\widehat{C}(\boldsymbol{\theta})$.
- The generator's properties (variance structure, gradient signal, plateau regions) are determined upstream by the ansatz, the Hamiltonian, the encoding, the noise model, and the shot budget.

Gradient descent's job, in this framing, is to **make decisions about where to query next** based on a sequence of past noisy evaluations. Vanilla GD is a particularly simple decision rule: query at $\boldsymbol{\theta}_k - \eta\widehat{\nabla C}$.

This framing makes clear that GD is **agnostic to the structure of the noise** and to the specific properties of the VQA. It treats the cost function as a black box. Optimizers that exploit VQA-specific structure (parameter-shift, periodicity, sinusoidal cost dependence, QFIM) can do better — and that is what the rest of Module 6 develops.

Structural Role

This is the **baseline** node of Module 6. It defines the simplest first-order optimizer, identifies its failure modes, and sets up the rest of the module as a sequence of refinements that address those failures.

It also forwards directly to Module 5 (regularization), where the same gradient descent dynamics are augmented with regularization terms that encode prior knowledge or shape the optimizer's path.

Relations to Other Concepts

- **Robbins-Monro 1951** — the foundational paper on stochastic approximation, the parent theory.
- **Cauchy 1847** — the original gradient descent paper.
- **Convex optimization** — the regime where GD has clean convergence guarantees.
- **L-smoothness** — the standard assumption needed for GD convergence proofs.
- **Module 5 regularization** — the dynamics-side framework that builds on top of GD.

Worked Example — A Two-Parameter VQE

Consider a 2-qubit VQE with ansatz $R_Y(\theta_1) \otimes R_Y(\theta_2) \cdot \mathrm{CNOT}_{0,1} \cdot |00\rangle$, Hamiltonian $H = Z_0 Z_1 + 0.5 X_0 + 0.5 X_1$.

Cost function (after a tedious but standard expansion):

$$C(\theta_1, \theta_2) = \cos(\theta_1)\cos(\theta_2) + 0.5\sin(\theta_1) + 0.5\sin(\theta_2).$$

Gradients:

$$\frac{\partial C}{\partial \theta_1} = -\sin(\theta_1)\cos(\theta_2) + 0.5\cos(\theta_1).$$
$$\frac{\partial C}{\partial \theta_2} = -\cos(\theta_1)\sin(\theta_2) + 0.5\cos(\theta_2).$$

Setting both gradients to zero: critical points at the solutions of $\tan(\theta_1)\cos(\theta_2) = 0.5$ and $\cos(\theta_1)\tan(\theta_2) = 0.5$.

Numerically, one minimum is at approximately $(\theta_1, \theta_2) \approx (-0.46, -0.46)$ with cost ≈ -1.12.

Run GD from $\boldsymbol{\theta}_0 = (0.5, 0.5)$ with $\eta = 0.1$, no noise: - Step 0: gradient $\approx (0.0, 0.0)$ — wait, let me reconsider. At $(0.5, 0.5)$, $-\sin(0.5)\cos(0.5) + 0.5\cos(0.5) \approx -0.421 + 0.439 = 0.018$. So gradient is small but nonzero. - The optimizer slowly drifts toward $(-0.46, -0.46)$ over ~ 50 steps.

Now with shot noise at $N = 1000$ per cost evaluation: gradient noise is ~ 0.03, comparable to the gradient near the start. The optimizer makes erratic progress for the first ~ 100 steps, then settles into a slow descent once the gradient is large enough.

This 2-parameter example shows the basic GD behavior in miniature: convergence works but is sensitive to noise, particularly when the gradient is small compared to the noise level.

Visualization

Pending. A 2D contour plot of $C(\theta_1, \theta_2)$ with a GD trajectory overlaid. Two trajectories shown: one noiseless (smooth curve to the minimum) and one noisy (erratic walk that eventually finds the minimum). Caption: "GD: simple, slow, noise-limited."

Exercises

1. For a 1D cost $C(\theta) = \theta^2$, show that GD with $\eta < 1$ converges, with $\eta = 1$ reaches the minimum in one step, and with $\eta > 2$ diverges. What is the equivalent statement for a 2D quadratic?
2. For shot-noisy GD on $C(\theta) = \cos(\theta)$ starting at $\theta_0 = 0.1$, estimate the shot budget needed to reach $\theta = \pi$ within 10^{-2} precision. How does this scale with N?
3. (VQA) For a 5-qubit HEA with 3 layers and a global $H = \sum_i Z_i$, estimate the typical gradient magnitude and the shot budget needed for GD to make meaningful progress. At what qubit count does GD become infeasible?

Research Extensions

- **Adaptive shot scheduling for GD** — protocols that allocate more shots when the gradient is small and fewer when it is large.
- **Line-search GD for VQA** — using a backtracking line search to choose η per step.
- **Restart strategies** — when to restart GD with a different initialization to avoid local minima.
- **GD with momentum vs vanilla GD** — quantitative comparison on standard VQA benchmarks.

Cross-links to Other Courses

- **Numerical Optimization** — gradient descent, line search, convergence theory.
- **Stochastic Approximation** — Robbins-Monro, Kiefer-Wolfowitz.
- **Module 3 (Cost Landscapes)** — the geometry GD navigates.
- **Module 4 (Barren Plateaus)** — the failure mode that breaks GD on large problems.
- **Module 5 (Regularization)** — the dynamics-side framework GD lives inside.

Variational Quantum Algorithms · Module 6 — Optimization Dynamics · Node 6.1
Concepts: gradient-descent, stochastic-optimization, convergence, parameterized-circuits
Prerequisites: 3.5, 3.7, 3.8
Enables: 6.2, 6.3, 6.4

hbar.university — own.your.cognition