Stochastic Optimization

Variational Quantum Algorithms · Module 6 — Optimization Dynamics

Node 6.4

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Optimization Dynamics **Node:** 6.4

Every VQA optimizer is, by necessity, a **stochastic** optimizer. The cost function it sees is corrupted by shot noise (Section 3.7), the gradient it computes is corrupted by parameter-shift estimation noise, and on real hardware, the gates are corrupted by physical noise. The optimizer's job is not to optimize a deterministic function — it's to make good decisions under noise.

This node introduces the **stochastic optimization** framework that the rest of Module 6 lives inside. It covers the foundational theory (Robbins-Monro), the special class of zero-order methods (Kiefer-Wolfowitz, SPSA), and the practical implications for VQA optimizer design.

The thesis's "stochastic objective generator" abstraction (Module 1.8) is most explicitly cashed out here.

The Stochastic Optimization Setting

A stochastic optimization problem is:

$$\arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\xi}[f(\boldsymbol{\theta}, \xi)],$$

where ξ is a random variable representing noise, and the optimizer can only access noisy samples $f(\boldsymbol{\theta}, \xi)$ — not the true expectation.

For VQAs, $\boldsymbol{\theta}$ is the parameter vector, ξ is the randomness from quantum measurement (shot noise) plus hardware noise, and f is the cost-function-with-noise. The expected value $\mathbb{E}_{\xi}[f]$ is the noiseless cost $C(\boldsymbol{\theta})$ (assuming unbiased estimation).

The question is: **how do you find** $\arg\min C$ **when you can only see noisy estimates of** C **and its derivatives?**

This is the territory of stochastic optimization theory, which has been developed since the 1950s and is now a mature field.

Robbins-Monro: The Foundational Result

Robbins and Monro (1951) proved that a simple recursion can find roots of a function from noisy estimates of the function's value:

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

where a_k is a sequence of step sizes (the "learning rate schedule") satisfying:

1. $\sum_k a_k = \infty$ — the steps don't decrease too fast.
2. $\sum_k a_k^2 < \infty$ — the steps eventually decrease.

A canonical choice is $a_k = a/k$.

Theorem: Under these conditions, plus mild regularity on C (smoothness, bounded variance of the gradient estimator, unique minimum), the iterates $\boldsymbol{\theta}_k$ converge almost surely to the minimum of C.

Why this matters: Robbins-Monro gives the first formal guarantee that stochastic gradient descent works *despite the noise*, provided the learning rate is decreased appropriately. The decreasing learning rate is the key — it averages out noise asymptotically.

For VQA, the Robbins-Monro condition translates to: - Use a decreasing learning rate, e.g., $\eta_k = \eta_0/k$ or $\eta_0/\sqrt{k}$. - Allocate enough shots that the gradient estimate is unbiased and has finite variance. - Be patient — convergence is asymptotic, and the rate is typically $O(1/\sqrt{k})$ for noisy problems.

Convergence Rates In Stochastic Optimization

For a smooth, strongly convex C with bounded gradient variance σ^2, the convergence rate of stochastic GD is

$$\mathbb{E}[C(\boldsymbol{\theta}_k) - C^*] \leq \frac{\sigma^2}{\mu k} + O\left(\frac{1}{k^2}\right),$$

where μ is the strong-convexity parameter.

For VQAs (which are typically *not* convex), the right notion of convergence is to a critical point, not the global minimum:

$$\mathbb{E}[\|\nabla C(\boldsymbol{\theta}_k)\|^2] \leq \frac{C}{\sqrt{k}}.$$

The $1/\sqrt{k}$ rate is the canonical "stochastic optimization rate" — slower than the $1/k$ of deterministic GD on smooth-convex problems, because the noise contributes a floor on accuracy.

The implication: if you want to halve the gradient norm in stochastic optimization, you need *4x* more iterations, not just 2x. This is why VQA optimizers often need many hundreds or thousands of iterations to converge — the noise scaling is unfavorable.

Mini-Batch SGD And Its Quantum Analogue

In classical deep learning, **mini-batch SGD** estimates the gradient from a subset (mini-batch) of training data, which gives a noisy but unbiased gradient estimate. The mini-batch size trades off computation per step against gradient noise.

The VQA analogue is **shot allocation**: the number of shots per cost evaluation determines the gradient noise level. More shots = lower noise = larger possible step size = fewer iterations. Fewer shots = higher noise = smaller possible step size = more iterations.

The **total shot budget** is the right thing to optimize. Given a fixed total of S shots, you can spend them as: - Many cheap iterations: S/(cost per iter) iterations, each with high noise per iter. - Few expensive iterations: S/(cost per iter) iterations, each with low noise.

The optimal allocation depends on the noise structure. As a rule of thumb, **noisy gradients with many cheap iterations** wins for problems where the optimizer can recover from noise (e.g., convex problems or

near a minimum), while **clean gradients with few expensive iterations** wins for problems where noise causes instability (e.g., near a saddle point or in a barren plateau).

This is one of the genuine VQA-specific design problems: choosing the shot budget per evaluation, which has no clean classical analogue.

Zero-Order Stochastic Optimization

What if you don't have gradients at all — only noisy cost evaluations?

This is the **zero-order** setting, and it has its own theory.

Kiefer-Wolfowitz (1952): the original zero-order analogue of Robbins-Monro. Estimates each gradient component by finite differences:

$$\widehat{\partial_i C} = \frac{C(\boldsymbol{\theta} + h\mathbf{e}_i) - C(\boldsymbol{\theta} - h\mathbf{e}_i)}{2h}.$$

Each component requires 2 cost evaluations, so the full gradient costs $2P$ — same as parameter-shift, but with finite-difference error.

SPSA (Spall, 1992): Simultaneous Perturbation Stochastic Approximation. Estimates the *full gradient* from just **2 cost evaluations**, regardless of P. The trick: perturb all parameters simultaneously by a random sign vector $\boldsymbol{\Delta}$, and use the difference $C(\boldsymbol{\theta} + h\boldsymbol{\Delta}) - C(\boldsymbol{\theta} - h\boldsymbol{\Delta})$ to estimate the gradient projected onto the random direction.

The SPSA gradient estimate is

$$\widehat{\partial_i C} = \frac{C(\boldsymbol{\theta} + h\boldsymbol{\Delta}) - C(\boldsymbol{\theta} - h\boldsymbol{\Delta})}{2h\Delta_i}.$$

This is unbiased (in expectation over $\boldsymbol{\Delta}$) and gives a noisy but useful descent direction.

Convergence: SPSA converges at the same asymptotic rate as full-gradient stochastic descent, but with $P/2$ **times fewer cost evaluations per iteration**. For large P, this is a huge saving.

The cost is that the per-iteration gradient is much noisier, so SPSA needs more iterations to reach the same accuracy. The **total cost** can still be lower because the per-iteration cost drops from $O(P)$ to $O(1)$.

For VQAs with $P \sim 100$, SPSA can be 10-50x faster than vanilla parameter-shift gradient descent, especially in the early iterations where high gradient precision is unnecessary.

SPSA In VQA Practice

SPSA was popularized in the VQA community by **Kandala et al. (2017)** ("Hardware-efficient variational quantum eigensolver for small molecules"), which used it to optimize VQE on real superconducting hardware. Since then it has been a default choice for hardware experiments where shot budgets are tight.

Practical SPSA workflow:

1. Choose perturbation size h (typically $h = c/k^{0.101}$, the canonical Spall schedule).
2. Choose learning rate $a_k = a/(k + A)^{0.602}$, with A a small constant.
3. At each iteration, draw $\boldsymbol{\Delta}$ as a vector of ± 1 (Bernoulli rademacher).
4. Evaluate $C(\boldsymbol{\theta} + h\boldsymbol{\Delta})$ and $C(\boldsymbol{\theta} - h\boldsymbol{\Delta})$.
5. Compute the SPSA gradient estimate.
6. Update parameters with $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - a_k \widehat{\nabla C}$.

7. Repeat.

The schedules $h \sim 1/k^{0.101}$ and $a \sim 1/k^{0.602}$ come from Spall's analysis as the asymptotically optimal choices. In practice they are tuned per problem.

When To Use SPSA Vs Parameter-Shift

Use SPSA when:

- You have many parameters ($P > 50$) and the per-iteration cost of parameter-shift is prohibitive.
- The problem is well-conditioned enough that noisy gradients are tolerable.
- You're early in the optimization and want to make fast progress before fine-tuning.
- You're in a regime where parameter-shift's precision is wasted (e.g., early iterations where you just need a descent direction, not an accurate one).

Use parameter-shift when:

- You have few parameters ($P < 30$) and per-iteration cost is manageable.
- You're late in the optimization and need accurate gradients for fine-tuning.
- The problem is ill-conditioned and noisy gradients lead to instability.
- You need exact gradients for theoretical reasons (e.g., natural gradient with QFIM).

A common workflow: use SPSA for the first 100-1000 iterations to get into a good basin, then switch to parameter-shift for fine-tuning.

Other Zero-Order Methods

Coordinate descent: update one parameter at a time using its 1D gradient. Good for problems with structure.

Random search: sample random parameter perturbations, accept those that decrease the cost. Slow but very robust to noise.

Bayesian optimization: model the cost function as a Gaussian process, query points to maximize information gain. Expensive per iteration but data-efficient.

Trust-region methods: maintain a region within which the local approximation is trusted, only step within the region. Good for noisy problems.

These all have niches in VQA optimization, but parameter-shift gradient descent (with some adaptive variant) and SPSA are the dominant choices.

Structural Role

This node is the **theoretical foundation** for everything stochastic in Module 6. It justifies why noisy optimization is possible, gives the convergence rates that subsequent sections use, and introduces SPSA as the principal alternative to parameter-shift gradients.

It also connects directly to the thesis's "stochastic objective generator" abstraction (Module 1.8), which is the conceptual frame for treating the VQA optimizer as a stochastic-decision-maker rather than as a smooth-function-minimizer.

Relations to Other Concepts

- **Robbins-Monro 1951** — the foundational stochastic approximation theorem.
- **Kiefer-Wolfowitz 1952** — the zero-order analogue.
- **Spall 1992** — SPSA, the efficient zero-order method.
- **Stochastic gradient descent (SGD)** — the classical deep learning workhorse.
- **Mini-batch optimization** — the classical analogue of shot allocation.

Worked Example — SPSA On A 4-Parameter VQE

Consider a 4-parameter ansatz with cost $C(\boldsymbol{\theta}) = \sum_{i=1}^{4}(\theta_i - i)^2$, so the minimum is at $\boldsymbol{\theta}^* = (1, 2, 3, 4)$ with $C^* = 0$.

Parameter-shift gradient: at any point, the gradient is exactly $2(\boldsymbol{\theta} - (1, 2, 3, 4))$, and computing it requires 8 cost evaluations.

SPSA gradient at $\boldsymbol{\theta}_0 = (0, 0, 0, 0)$:

Draw $\boldsymbol{\Delta} = (+1, +1, -1, +1)$. Evaluate

$$C(\boldsymbol{\theta}_0 + h\boldsymbol{\Delta}) - C(\boldsymbol{\theta}_0 - h\boldsymbol{\Delta}) =?$$

With $h = 0.1$: - $C(0.1, 0.1, -0.1, 0.1) = 0.81 + 3.61 + 9.61 + 15.21 = 29.24$. - $C(-0.1, -0.1, 0.1, -0.1) = 1.21 + 4.41 + 8.41 + 16.81 = 30.84$. - Difference: $29.24 - 30.84 = -1.60$.

SPSA gradient estimate:

$$\widehat{\partial_i C} = \frac{-1.60}{2 \cdot 0.1 \cdot \Delta_i} = -8/\Delta_i.$$

- For $i = 1$: $-8/(+1) = -8$. True gradient: $2(0 - 1) = -2$. SPSA estimate is biased high in magnitude but correct in sign. - For $i = 2$: $-8/(+1) = -8$. True: -4. - For $i = 3$: $-8/(-1) = +8$. True: -6. **Wrong sign!** - For $i = 4$: $-8/(+1) = -8$. True: -8. Correct by luck.

The single-iteration SPSA estimate is noisy and can be wrong on individual components. But **averaged over many iterations** (each with a fresh $\boldsymbol{\Delta}$), the estimate converges to the true gradient. In the long run, SPSA produces a valid descent direction with much less per-iteration cost than parameter-shift.

For this 4-parameter problem, SPSA converges in roughly the same number of iterations as parameter-shift, but each iteration costs 2 cost evaluations instead of 8 — a 4x saving. For a 100-parameter problem, the saving is 100x.

Visualization

Pending. Two trajectories on a 2D cost landscape: parameter-shift gradient descent (smooth, slow) and SPSA (jagged, fast). The SPSA trajectory is noisier per step but reaches the minimum in fewer total cost evaluations. Caption: "SPSA: noisy but cheap."

Exercises

1. Verify Robbins-Monro conditions for the schedule $a_k = 1/k$. Verify that they fail for $a_k = 1/k^2$ (sums to a finite value).
2. For the same 4-parameter cost function above, run SPSA for 100 iterations (analytically or in code) and observe how fast the parameters converge to the minimum.

3. (VQA) For a 50-parameter HEA with shot budget $N = 1000$ per evaluation, compare the total shot cost of (a) 1000 parameter-shift iterations and (b) 10000 SPSA iterations. Which uses fewer total shots? Which is likely to give a better final cost?

Research Extensions

- **Variance-reduced SPSA** — combining SPSA with control variates to reduce per-iteration noise.
- **Adaptive perturbation sizes** — choosing h_k based on the local landscape.
- **SPSA + Adam hybrids** — using SPSA gradients in an Adam-like update rule.
- **Quantum-aware SPSA** — exploiting the periodicity and sinusoidal structure of VQA cost functions to improve SPSA bias.

Cross-links to Other Courses

- **Stochastic Approximation Theory** — the broader field.
- **Reinforcement Learning** — uses similar zero-order techniques (REINFORCE, evolution strategies).
- **Module 3 (Shot Noise)** — the noise structure SPSA navigates.
- **Module 6 Section 6.1 (Gradient Descent in VQA)** — the parameter-shift baseline SPSA competes with.

Variational Quantum Algorithms · Module 6 — Optimization Dynamics · Node 6.4
Concepts: stochastic-optimization, random-variables, convergence, cost-function
Prerequisites: 6.1, 3.7
Enables: 6.5

hbar.university — own.your.cognition