Expectation Value Estimation

Variational Quantum Algorithms · Module 1 — Hybrid Quantum Classical Systems

Node 1.3

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hybrid Quantum-Classical Systems **Node:** 1.3

This node answers the question that 1.1 raised and 1.2 deferred: **how do you actually get the number $\langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle$ out of a quantum computer?** The answer is "you don't — you get a noisy estimate of it, and the noise is intrinsic to how measurement works."

This is where "shot noise" enters the story. Everything about VQA optimization that distinguishes it from deterministic classical ML ultimately traces back to this node.

The Three-Step Recipe

Step 1: Decompose. Write the Hamiltonian as a weighted sum of Pauli strings:

$$H = \sum_{j=1}^{M} h_j P_j,$$

where each $P_j \in \{I, X, Y, Z\}^{\otimes n}$ is a tensor product of single-qubit Pauli operators and $h_j \in \mathbb{R}$. This decomposition exists for any Hermitian H — the Pauli strings form a basis for the space of Hermitian operators on n qubits.

Step 2: Measure each term. By linearity of expectation,

$$\langle\psi|H|\psi\rangle = \sum_{j=1}^{M} h_j \langle\psi|P_j|\psi\rangle.$$

So you can measure $\langle P_j\rangle$ separately for each j and sum the results. Each measurement requires rotating into the eigenbasis of P_j and then measuring in the computational basis. For a Pauli string like $X_1 Z_2$, you apply a Hadamard on qubit 1, do nothing on qubit 2, and measure both.

Step 3: Sample. Measurement in quantum mechanics is a random process. For N shots, you get N independent samples $s_j^{(1)}, \ldots, s_j^{(N)} \in \{-1, +1\}$ distributed according to the eigenvalue probabilities of P_j on $|\psi\rangle$. The unbiased estimator is the sample mean:

$$\widehat{\langle P_j\rangle} = \frac{1}{N} \sum_{i=1}^{N} s_j^{(i)}.$$

Combine across terms to get

$$\widehat{C}(\boldsymbol{\theta}) = \sum_{j=1}^{M} h_j \widehat{\langle P_j \rangle}.$$

That is the number the quantum layer returns to the classical optimizer.

Why This Is Noisy

Every step introduces uncertainty that does not exist in a classical function evaluation.

Shot noise. Even on a perfect device, $\widehat{\langle P_j \rangle}$ is a sample mean and therefore has variance. For a Pauli observable with true expectation value $p \in [-1, 1]$,

$$\mathrm{Var}\left[\widehat{\langle P_j \rangle}\right] = \frac{1 - p^2}{N} \leq \frac{1}{N}.$$

Combined across terms, the total variance of $\widehat{C}(\boldsymbol{\theta})$ scales as $O(\|h\|_1^2/N)$ in the worst case. Halving the uncertainty costs four times as many shots.

Hardware noise. On real devices, additional sources kick in: gate errors, decoherence during state preparation, readout errors, calibration drift. These are not sampling noise — they corrupt the prepared state before measurement — but from the optimizer's perspective they look similar: the number coming back is not the ideal $C(\boldsymbol{\theta})$.

Basis-change overhead. Measuring M Pauli terms naively requires M separate circuits (one per measurement basis). For molecular Hamiltonians, M can be in the thousands. This motivates **measurement grouping** — partitioning Pauli strings into commuting sets that can be measured simultaneously (Module 2).

What the Optimizer Actually Sees

From the classical layer's perspective, each call to the quantum layer returns a random variable:

$$\widehat{C}(\boldsymbol{\theta}) = C(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta}),$$

where $C(\boldsymbol{\theta})$ is the ideal expectation value and $\varepsilon(\boldsymbol{\theta})$ is a zero-mean noise term with variance $\sigma^2(\boldsymbol{\theta})$ that depends on the number of shots, the structure of H, and (on real hardware) the physical noise model.

This is a **stochastic oracle** in the sense of classical optimization theory. The optimizer never sees $C(\boldsymbol{\theta})$ directly, only samples from its distribution. Every classical optimization algorithm used in VQA must be robust to this — or at least behave reasonably given noisy inputs.

Module 5 is essentially about how to make optimizers behave reasonably given ε. Module 9 is essentially about how to reduce ε at the source.

Structural Role

This is the node that makes everything else in VQA feel different from classical ML. Without shot noise, VQA would be straightforward stochastic optimization of a deterministic black-box function. With shot noise, you are doing stochastic optimization of a **stochastic** function whose sampling cost you also have to budget.

The shot budget is the real currency of NISQ-era VQA. Iterations are free; shots are not. Module 6 picks this up when discussing how to compare optimizers fairly (shots, not iterations, is the x-axis).

Relations to Other Concepts

- **Monte Carlo estimation** — the sample-mean estimator is a textbook Monte Carlo, with convergence rate $1/\sqrt{N}$.
- **Importance sampling** — can reduce variance by reshuffling the measurement distribution; used in advanced shot-allocation schemes.
- **Pauli grouping / qubit-wise commuting groups** — partitioning $\{P_j\}$ into simultaneously measurable sets.
- **Classical shadows** — a newer technique (Huang, Kueng, Preskill 2020) that estimates many observables from a single randomized measurement protocol, often dramatically cheaper than naive Pauli measurement.

Worked Example — Estimating $\langle Z \rangle$ on a Single Qubit

Suppose $|\psi\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)|1\rangle$. Then $\langle Z \rangle = \cos\theta$.

Run N shots. Each shot returns $+1$ with probability $\cos^2(\theta/2)$ and -1 with probability $\sin^2(\theta/2)$. The sample mean is an unbiased estimator of $\cos\theta$ with variance

$$\frac{1 - \cos^2\theta}{N} = \frac{\sin^2\theta}{N}.$$

Notice: the variance is **smallest at** $\theta = 0$ (where $\langle Z \rangle = +1$ deterministically) and **largest at** $\theta = \pi/2$ (where the state is $|+\rangle$ and measurement outcomes are maximally random). So the hardness of estimating a cost value depends on **where you are in parameter space**, not just how many shots you spend. This is one of the reasons variance-aware optimization (node 5.7) is a real thing.

Visualization

Pending. Diagram: top panel shows a single-qubit Bloch sphere with $|\psi\rangle$ at some angle θ. Middle panel shows 100 measurement outcomes as +1/-1 ticks. Bottom panel shows the running sample mean converging to $\cos\theta$ with a shrinking confidence band of width $\sim 1/\sqrt{N}$.

Exercises

1. Derive $\mathrm{Var}[\widehat{\langle P_j \rangle}] = (1 - p^2)/N$ from the definition of sample variance. What is the maximum variance over all possible $p \in [-1, 1]$?
2. Suppose $H = 0.5\,X + 0.5\,Z$. Compute $\mathrm{Var}[\widehat{C}(\boldsymbol{\theta})]$ as a function of $\boldsymbol{\theta}$ assuming independent shot budgets $N_X = N_Z = N$.
3. (VQA) If you have a total shot budget of 10,000 and two Pauli terms with coefficients $h_1 = 1.0$ and $h_2 = 0.1$, how should you allocate shots between them to minimize the variance of $\widehat{C}$? (Hint: Lagrange multipliers.)

Research Extensions

- **Classical shadows (Huang-Kueng-Preskill)** — exponentially cheaper estimation of many observables.

- **Adaptive shot allocation** — using the variance of early measurements to decide where to spend later shots.
- **Measurement grouping** — abelian Pauli partitioning; graph-coloring formulation.
- **Quantum-enhanced estimation** — amplitude estimation gives $1/N$ scaling instead of $1/\sqrt{N}$, but requires deeper circuits than NISQ can support.

Cross-links to Other Courses

- **Statistics Module 2** — sample mean, variance, standard error; this is literally that chapter applied to quantum measurements.
- **Quantum Foundations** — measurement postulate, projective measurements, Born rule.
- **Hardware Noise Models (Module 9)** — the other source of $\varepsilon(\boldsymbol{\theta})$.

Variational Quantum Algorithms · Module 1 — Hybrid Quantum Classical Systems · Node 1.3
Concepts: expectation-value, born-rule, variance-and-covariance, tensor-product
Prerequisites: 1.1, 1.2
Enables: 1.4, 1.5, 1.8

hbar.university — own.your.cognition