Cost Function Design

Variational Quantum Algorithms · Module 3 — Cost Landscapes

Node 3.1

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Cost Landscapes **Node:** 3.1

This is the **opening node** of Module 3 and the place where the question shifts from *what is the ansatz?* to *what is the function we are minimizing on it?*

The cost function in a VQA is **not** uniquely determined by the problem. You start with a problem (find the ground state of H) and you have some freedom to choose what scalar you actually optimize against. This node walks through that freedom, the standard choices, and the consequences each choice has for the resulting landscape.

It is a node about **design decisions you have to make before the optimizer ever sees a number**.

The Default: Energy Expectation

The textbook VQA cost function is the energy expectation:

$$C(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H | \psi(\boldsymbol{\theta}) \rangle.$$

Why this is the default: - It is bounded below by the true ground state E_0 (variational principle). - It has a meaningful physical interpretation as the expected energy of the prepared state. - It is linear in the Pauli decomposition of H, so it can be computed by Pauli-by-Pauli measurement (node 2.7). - It is the cost the variational principle directly addresses.

For 95% of VQA work — VQE for chemistry, VQE for spin models, ground-state preparation — this is what you optimize.

When The Default Isn't Enough

There are several situations where the energy expectation is a *bad* cost function and you have to design something else.

Excited states

The variational principle gives you the *ground* state. To find an excited state, you need a cost function whose minimum is the excited state of interest. Three approaches:

Subspace-search VQE. Optimize against $\mathrm{Tr}(H\rho_S)$ over states in a fixed subspace, with constraints enforcing orthogonality to lower states.

Variational Quantum Deflation. Add penalty terms to the energy expectation that push the state away from already-found lower states:

$$C(\boldsymbol{\theta}) = \langle\psi(\boldsymbol{\theta})|H|\psi(\boldsymbol{\theta})\rangle + \beta \sum_{i\in\text{found}} |\langle\psi_i|\psi(\boldsymbol{\theta})\rangle|^2.$$

The penalty term raises the cost of any state with overlap with previously found eigenstates.

SSVQE. Train multiple states simultaneously with constraints on their orthogonality.

Each of these is a cost function design choice that changes the landscape topology.

Dynamics simulation

For preparing a state evolved by e^{-iHt} variationally, the cost is not energy but **fidelity to a target state**:

$$C(\boldsymbol{\theta}) = 1 - |\langle\psi_{\text{target}}|\psi(\boldsymbol{\theta})\rangle|^2.$$

Or, computed in another way, the **infidelity in time-evolved measurements**.

These cost functions have qualitatively different landscape structures from the energy expectation — they are bounded above as well as below, they have explicit fixed points at $\boldsymbol{\theta}_0 = \arg\min C$, and they typically have *more* local minima than the energy expectation.

Quantum machine learning

For QML problems (variational classifiers, generative models), the cost function is a problem-specific loss:

- **Cross-entropy** for classification.
- **MSE** for regression.
- **MMD or Wasserstein** for generative models.

The cost is a function of the *measured outputs* of the circuit, which themselves are expectation values, but the relationship between parameters and cost is now mediated by a classical loss function. This adds another layer of nonlinearity (and another source of design freedom).

Constrained optimization

When the problem has symmetry constraints (particle number, total spin), you can enforce them either:

- **Implicitly**, by restricting the ansatz to states obeying the symmetry, OR
- **Explicitly**, by adding penalty terms to the cost: $C(\boldsymbol{\theta}) = \langle H\rangle + \mu\langle(N - N_0)^2\rangle$.

The penalty approach changes the landscape — it creates new minima at the constrained points and lifts the energy of states outside the constraint surface.

Cost Function Design as an Optimization Move

The key insight: **the cost function is itself a hyperparameter.**

Two cost functions that nominally encode the same problem can produce dramatically different optimization behaviors: - different barren plateau profiles, - different numbers of local minima, - different gradient magnitudes, - different convergence rates.

Most VQA papers fix the cost function early and then spend the rest of their effort optimizing the optimizer. A better question is sometimes: **could we have chosen a different cost function whose landscape is friendlier?**

Examples of cost-function-side improvements:

- **Local cost functions** (Cerezo et al. 2021): replace a global cost like $\langle H \rangle$ with a sum of local 2-qubit observables. Provably avoids barren plateaus for shallow ansätze.
- **Variance penalization** (node 5.7 in this course): add a term proportional to the *variance* of the cost estimator, which keeps the optimizer in regions where the cost is more reliably measurable.
- **Cost function re-normalization**: shift and rescale the cost so its dynamic range matches the optimizer's expectations. Surprisingly effective for adaptive learning rate methods.

Each of these is a *cost-side* intervention — modify the function being optimized rather than the algorithm doing the optimizing. It is the dual to optimizer-side interventions and often the more impactful one when it works.

Cost Function Design and Module 5

This connects directly to Module 5's regularization thesis. The thesis distinguishes between two kinds of regularization:

- **Prior injection** — modify the cost function (objective-side intervention).
- **Dynamical modification** — modify the optimizer's update rule (dynamics-side intervention).

The cost function design choices in this node are all examples of prior injection. They change C before the optimizer sees it. Whether this is a good idea — and how it interacts with the optimizer's dynamics — is exactly what Module 5's thesis is about.

So this node is the **upstream** of Module 5's main argument: it is the place where the practitioner consciously chooses what objective they are optimizing, and that choice determines what kind of regularization makes sense.

Structural Role

This is the opening node of Module 3 and the bridge from the **physical** content of Module 2 (how do we get a number out of the device?) to the **landscape** content of Module 3 (what does the resulting function look like?).

Forwards to 3.2 (topology), 3.3 (minima), and to Module 5 (regularization, where cost-function-side modification appears in full).

Relations to Other Concepts

- **Loss function design in ML** — same problem, different domain. Cross-entropy vs MSE vs hinge vs focal loss is the classical analogue.
- **Penalty methods in classical optimization** — Lagrangian relaxation, augmented Lagrangians, barrier functions.
- **Variational principle (1.1)** — guarantees the energy expectation is a defensible default.
- **Surrogate cost functions** — replace the true cost with a cheaper-to-evaluate proxy.

Worked Example — Two Costs For The Same Problem

For H_2, consider two cost functions:

Cost A: Energy expectation.

$$C_A(\boldsymbol{\theta}) = \langle \psi(\boldsymbol{\theta}) | H_{H_2} | \psi(\boldsymbol{\theta}) \rangle.$$

Range: [-1.85, +0.5] hartree. Minimum at $\boldsymbol{\theta}^*$: -1.137 hartree (FCI). Barren plateau profile (HEA): yes, in the random-init regime.

Cost B: Energy expectation + variance penalty.

$$C_B(\boldsymbol{\theta}) = \langle H \rangle + \lambda \mathrm{Var}[\widehat{\langle H \rangle}].$$

For the same minimizer, the variance is small (eigenstates have zero variance for their corresponding eigenvalue), so the penalty pushes toward eigenstate-like configurations. Range: same as above, plus a positive correction. Minimum: still near the FCI ground state, but the basin is *narrower* and *deeper* — variance vanishes only at eigenstates.

Which is better? Cost A converges faster initially (gradient is larger). Cost B converges slower but gets less stuck in mid-energy plateaus. The right answer depends on the optimizer being used and the problem regime.

This is the kind of tradeoff that cost function design can navigate — and that the rest of Module 3 will give you the tools to reason about.

Visualization

Pending. Two side-by-side cost surfaces. Left: pure energy expectation, with mild gradient and a wide basin. Right: energy + variance penalty, with steeper gradient and narrower basin. Caption: "The cost function is a design choice, not a given."

Exercises

1. For a 1-parameter ansatz $|\psi(\theta)\rangle = R_Y(\theta)|0\rangle$ and Hamiltonian $H = Z$, compute both $\langle H \rangle(\theta)$ and $\mathrm{Var}[\langle H \rangle](\theta)$. Where are the minima of each?
2. The variational principle gives $\langle H \rangle \geq E_0$. Does the same kind of guarantee hold for the variance-penalized cost? Argue carefully.
3. (VQA) Design a cost function whose minimum is the *first excited state* of H, given knowledge of the ground state $|\psi_0\rangle$. What new local minima does your construction introduce?

Research Extensions

- **Adaptive cost reformulation** — reshape the cost during optimization based on observed difficulty.
- **Cost function bandits** — treat the choice of cost function as an arm to be explored.
- **Dual cost functions** — optimize a relaxation, then refine on the original.
- **Cost-shaping for QML** — designing classification losses tuned to circuit ansatz structure.

Cross-links to Other Courses

- **Machine Learning** — loss function design as a first-class concern.
- **Optimization Theory** — penalty methods, Lagrangian duality.
- **Module 5 (Regularization)** — the natural continuation of this node.
- **Module 8 (Expressivity vs Trainability)** — local vs global cost functions and barren plateaus.

Variational Quantum Algorithms · Module 3 — Cost Landscapes · Node 3.1
Concepts: cost-function, variational-principle, expectation-value
Prerequisites: 1.1, 2.5
Enables: 3.2, 3.3

hbar.university — own.your.cognition