Variational Quantum Algorithms

Based on variational method of quantum mechanics
  • finding approximations to ground state - lowest energy eigenstate

General idea
  Prepare trial state based on parameter(s)
  Evaluate expected value of observable (Hamiltonian) for that state
    Minimum expected value = lowest eigenvalue
    (Our examples will actually try to maximize the expected value.)
  Adjust parameter(s) -- usually with classical optimizer
  Repeat to find minimum (or maximum)
Quantum

$|0\rangle \rightarrow$

Create state $|\psi(\beta, \gamma)\rangle$

Adjust parameters $\beta, \gamma$

Calculate $\langle H \rangle_\psi$

Run and measure a bunch of times, evaluate $H$ for each, compute average.
Quantum Approximate Optimization Algorithm (QAOA)

State = alternating circuits (operators).

Based on **cost** function and **mixing** function

\[
|\gamma, \beta\rangle = U(B, \beta_p) U(C, \gamma_p) \cdots U(B, \beta_1) U(C, \gamma_1) |s\rangle
\]

Depth (p): number of alternating operators

Variational Quantum Eigensolver (VQE)

Based on fixed “variational forms”


https://community.qiskit.org/textbook/ch-applications/vqe-molecules.html
QAOA
Objective Function

\[ C(z) = \sum_{\alpha=1}^{m} C_{\alpha}(z) \]

- \( z \) is an \( n \)-bit string.
- There are \( m \) clauses in the objective.
- Each clause typically involves only a few bits.
- \( C_{\alpha}(z) \) is 1 if \( z \) satisfies the clause.

**Satisfiability:** Is there a string \( (z) \) that satisfies all clauses?

**MaxSat:** Which string \( (z) \) satisfies the most clauses?

**Approximate Optimization:** Find a string \( (z) \) for which \( C(z) \) is close to the maximum.
Example: MaxCut

Divide a graph into two partitions, such that the number of edges that connect between partitions is maximized.

\[ C = \sum_{\langle jk \rangle} C_{\langle jk \rangle} \]

\[ C_{\langle jk \rangle} = \frac{1}{2} (-\sigma_j^z \sigma_k^z + 1) \]
Define a Unitary Operator

For a given state $z$, each clause that is satisfied applies a phase rotation of $\gamma$. So state(s) that maximize objective function will be rotated most.

$$U(C, \gamma) = e^{-i\gamma C} = \prod_{\alpha=1}^{m} e^{-i\gamma C_{\alpha}}$$

Range of $\gamma = 0$ to $2\pi$

$$e^{i\gamma A} = \cos(\gamma)I + i\sin(\gamma)A$$
Implementation

\[
\left| x_1 \right\rangle \quad U(C, \gamma) \quad \left| x_2 \right\rangle \quad \cdots \quad \left| x_n \right\rangle = \left| x_1 \right\rangle \quad \text{Control 1 with} \quad C_1 \quad \text{Control 2 with} \quad C_2 \quad \cdots \quad \text{Control} \quad m \quad \text{with} \quad C_m
\]

\[
\left| x_1 \right\rangle \quad \left| x_2 \right\rangle \quad \cdots \quad \left| x_n \right\rangle \quad \left| 0 \right\rangle \quad \text{Target 1} \quad R(\gamma) \quad \text{Target 2} \quad R(\gamma) \quad \cdots \quad \text{Target} \quad m \quad R(\gamma)
\]

Quirk

Implementation

Selective phase change

Quirk
So far...

What if we apply $U(C, \gamma)$ to input $|s\rangle$?

$$|s\rangle = \frac{1}{\sqrt{2^n}} \sum_{z} |z\rangle$$

Rotating phase does not change amplitude, so measuring will not give any advantage.

Each $z$ has a probability of $\frac{1}{\sqrt{2^n}}$ so it’s no better than guessing.

Need to convert phase into amplitude...
Define a Mixing Operator

\[
B = \sum_{j=1}^{n} \sigma_j^x
\]

\[
U(B, \beta) = e^{-i\beta B} = \prod_{j=1}^{n} e^{-i\beta \sigma_j^x}
\]
Choose 2p angles and...

$$|\gamma, \beta\rangle = U(B, \beta_p) U(C, \gamma_p) \cdots U(B, \beta_1) U(C, \gamma_1) |s\rangle$$

Get expected value:

$$F_p(\gamma, \beta) = \langle \gamma, \beta | C | \gamma, \beta \rangle$$

$$M_p = \max_{\gamma, \beta} F_p(\gamma, \beta)$$

Maximization at p-1 can be considered constrained maximization at p.

$$M_p \geq M_{p-1}$$

$$\lim_{p\to\infty} M_p = \max_z C(z)$$

Circuit depth is $mp + p$
Challenge: Finding Good Angles

Classical preprocessing (original paper)
   With $p = 1$, approximation ratio is 0.6924. (Choosing random partition = 2/3)

Create grid over $\gamma = [0,2\pi]$ and $\beta = [0,\pi]$ and explore.

Gradient descent -- can get stuck at local minima/maxima, expensive in terms of evaluations required. (not recommended)

For noisy:
   SPSA = Simultaneous Perturbation Stochastic Approximation

For simulator:
   SLSP = Sequential Least Squares Programming
   COBYLA = Constrained Optimization by Linear Approximation

From IBM textbook
Discussion

QAOA has not been demonstrated to perform better than classical.
Better approximation? Faster?

Can apply different mixing operators

Hadfield, et al., arXiv:1709.03489
Quantum Alternating Operator Ansatz (QAOA)

Dealing with more general
cost functions and constraints
Qiskit Aqua
Aqua Components

Pluggable algorithm classes
  Ising model operators
  QAOA, VQE, QGAN, ...

Circuits
  Boolean logic functions, QFT, phase estimation...

Optimizers
  COBYLA, SPSA, ...

Abstractions above circuits and gates
class QAOA(...)¶

Parameters

operator (BaseOperator) – Qubit operator
operator_mode (str) – operator mode, used for eval of operator
p (int) – the integer parameter p as specified in https://arxiv.org/abs/1411.4028
initial_state (InitialState) – the initial state to prepend the QAOA circuit with
mixer (BaseOperator) – the mixer Hamiltonian to evolve with.
optimizer (Optimizer) – the classical optimization algorithm.
num_nodes = weight_matrix.shape[0]
pauli_list = []
shift = 0
for i in range(num_nodes):
    for j in range(i):
        if weight_matrix[i, j] != 0:
            x_p = np.zeros(num_nodes, dtype=np.bool)
            z_p = np.zeros(num_nodes, dtype=np.bool)
            z_p[i] = True
            z_p[j] = True
            pauli_list.append([0.5 * weight_matrix[i, j],
                               Pauli(z_p, x_p)])
            shift -= 0.5 * weight_matrix[i, j]
return WeightedPauliOperator(paulis=pauli_list), shift

qiskit/optimization/ising/max_cut.py
circuit.u2(0, np.pi, q)
for idx in range(self._p):
    beta, gamma = angles[idx], angles[idx + self._p]
    circuit += self._cost_operator.evolve(
        evo_time=gamma, num_time_slices=1, quantum_registers=q
    )
    circuit += self._mixer_operator.evolve(
        evo_time=beta, num_time_slices=1, quantum_registers=q
    )
Useful Tutorials

qiskit-iqx-tutorials/qiskit/advanced/aqua/optimization/

max_cut_and_tsp.ipynb

vehicle_routing.ipynb