**Quantum Bayesian Inference Using Statevector Simulation for Intrusion Detection**

This documentation outlines how quantum statevector simulation was used to implement Bayesian inference for a quantum-enhanced intrusion detection scenario. It replaces the original QBayesian-based implementation, which could not be executed due to library incompatibilities and missing modules in the current Qiskit ecosystem.

The original implementation relied on the QBayesian class from qiskit\_machine\_learning.algorithms, which abstracts quantum Bayesian inference processes using a sampler backend. However, this approach encountered multiple errors during environment setup:

1. **ModuleNotFoundError**: The qiskit.primitives.Sampler and BaseSampler modules could not be imported.
2. **ImportError**: The QBayesian class could not be recognized due to incompatibility between the installed Qiskit version and the qiskit\_machine\_learning package.
3. **Installation Failure**: Attempting to install qiskit-primitives or required qiskit\_machine\_learning submodules returned errors such as "No matching distribution found."

As a result, the automated inference via QBayesian was not feasible in this environment.

**Manual Reconstruction via Statevector Simulation**

Instead of relying on QBayesian, the entire Bayesian inference process was reconstructed manually using Statevector from qiskit.quantum\_info. This method allows full control of quantum state analysis while remaining consistent with the principles of Bayesian reasoning.

**Step-by-Step Overview**

**1. Cybersecurity Scenario Context**

This quantum inference model simulates a simplified intrusion detection system:

* X: Represents whether an intrusion attempt has occurred (0 = no, 1 = yes)
* Y: Represents whether the alarm system has been triggered (0 = no, 1 = yes)

This scenario reflects a common real-world cybersecurity situation where an organization monitors its network for unauthorized access. When suspicious activity is detected (intrusion), an internal system may trigger an alarm. However, false positives or hardware noise could also cause alarms without actual intrusions. The Bayesian model quantifies these probabilistic relationships using quantum circuits.

For instance, if logs indicate an alert from a firewall or an IDS (Intrusion Detection System), the organization wants to calculate how likely it is that an actual intrusion occurred. Here, we are simulating this reasoning using quantum amplitude encoding.

The system's prior assumptions (likelihood of intrusion and alarm) and conditional dependence (alarm given intrusion) are modeled using a quantum circuit. This circuit forms a quantum Bayesian network. The objective is to infer the posterior probability of intrusion given that the alarm was triggered — computing the conditional P(X=1 | Y=1).

**2. Quantum Circuit Construction**

A 2-qubit quantum circuit is created. A Hadamard gate is applied to the X qubit to simulate an equal superposition (prior probability). A controlled rotation (CRY or CNOT) gate encodes the conditional dependency of the alarm Y on the intrusion X.

from qiskit import QuantumCircuit

qc = QuantumCircuit(2)

qc.h(0) # X in superposition

qc.cx(0, 1) # Alarm Y depends on Intrusion X

This simple quantum circuit mimics the dependency: if intrusion happens, it triggers the alarm. This encoding forms the probabilistic dependency required for inference.

**3. Simulation via Statevector**

The full quantum state is generated by applying the circuit to an initial state:

from qiskit.quantum\_info import Statevector

state = Statevector.from\_instruction(qc)

**4. Extract Joint Probability Distribution**

Using state.probabilities\_dict(qargs=[0,1]), the joint probability distribution over the qubits is extracted. This gives values such as:

{'00': 0.5, '11': 0.5}

This indicates that 50% of the time, there is no intrusion and no alarm; 50% of the time, there is an intrusion and an alarm. This output implies that the alarm is only raised in the presence of an intrusion, consistent with a deterministic detector.

**5. Evidence Conditioning (Bayesian Update)**

To perform inference, such as calculating P(X=1 | Y=1), we:

* Filter only outcomes where Y=1
* Normalize their probabilities

filtered = {k: v for k, v in probs.items() if k[1] == '1'}

total = sum(filtered.values())

posterior = {k: v/total for k, v in filtered.items()}

This yields a Bayesian posterior distribution based on observed evidence: the alarm being triggered.

**6. Manual Histogram Visualization**

Due to rendering issues with plot\_histogram, we switched to matplotlib for custom bar plots:

import matplotlib.pyplot as plt

plt.bar(posterior.keys(), posterior.values())

plt.xlabel("State")

plt.ylabel("Posterior Probability")

plt.title("P(X | Y=1)")

plt.grid(True)

plt.show()

This histogram visually presents how likely an intrusion was, given that an alarm was detected.

**Interpretation of Results**

The posterior distribution derived via statevector simulation replicates the behavior of quantum Bayesian inference. It shows that if an alarm is observed (Y=1), the system believes with high confidence that there was an intrusion (X=1). This aligns with classical Bayesian reasoning but is computed from a quantum circuit that encodes prior and conditional dependencies in the amplitudes of quantum states.

In the context of cybersecurity, this approach allows analysts to model and simulate probabilistic scenarios involving threat detection using quantum logic. By conditioning on the alarm being triggered, we deduce the likelihood of an actual intrusion occurring. This can form the basis of more advanced probabilistic quantum models for future intelligent security systems.

**Advantages of Manual Statevector Method**

* **Transparency**: Every step of the Bayesian update is visible and tunable.
* **Compatibility**: Bypasses broken dependencies in Qiskit’s high-level machine learning libraries.
* **Flexibility**: Works across different quantum backends and platforms.
* **Customizability**: Enables extensions to larger networks (e.g., 3+ qubits) and alternative conditional structures.

**Future Enhancements**

* Extend the model to 3 or more qubits (e.g., include Admin Response, Firewall Logs, etc.)
* Introduce quantum noise models and simulate with noisy backends
* Add real-world intrusion log datasets for learning conditional rotation parameters

**Conclusion**

Bayesian inference can still be robustly achieved in a quantum system using Statevector simulation, even without high-level abstractions like QBayesian. This approach maintains fidelity to quantum probabilistic logic and allows precise implementation of inference steps in cybersecurity threat detection scenarios.