Quantum Programming - Implementing a quantum based Principal Component Analysis

Raschack Selina

Quantum Programming SoSe 2021

Agenda

- Context
  - notations
  - classical PCA
  - quantum PCA
- Implementation
  - circuit initialization
  - simple quantum PCA with swap test
  - enhancements with quantum phase estimation
- Results
- Conclusion

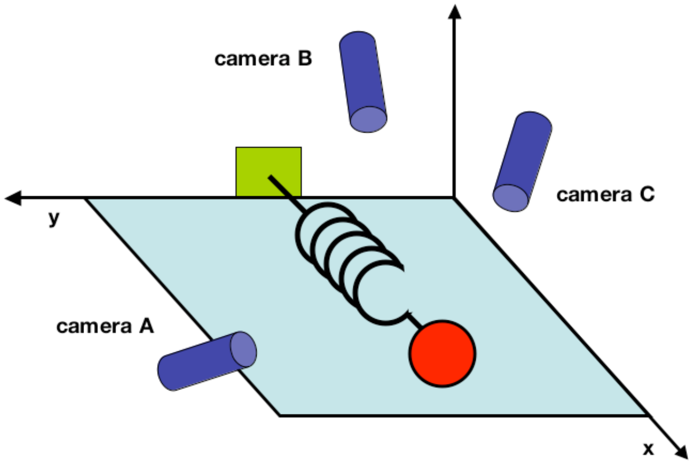
Context

Notations

- state vector: $|\psi\rangle$, arbitrary quantum state ψ
- state vector: $|0\rangle$, $|1\rangle$, classical states 0 and 1 respectively
- density matrix: $\rho = \Sigma_{i=1}^{n} p_i |\psi_i\rangle\langle\psi_i|$, probabilities p_i, $\Sigma_i p_i = 1$, dimensions n
  - example[a]: two qubits entangled state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = \frac{1}{\sqrt{2}}(1,0,0,1)^T$
  - example density matrix: $\rho = \frac{1}{\sqrt{2}}(1,0,0,1)^T \frac{1}{\sqrt{2}}(1,0,0,1) = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$
- covariance matrix: Σ
  - example[b]: two feature sets X and Y
  - example covariance matrix: $\Sigma = \begin{bmatrix} E(X \otimes X) & E(X \otimes Y) \\ E(Y \otimes X) & E(Y \otimes Y) \end{bmatrix}$, with $E(A)$ the expectation value of A

[a] [Melke2010]
[b] [Lokho2020]

Classical PCA

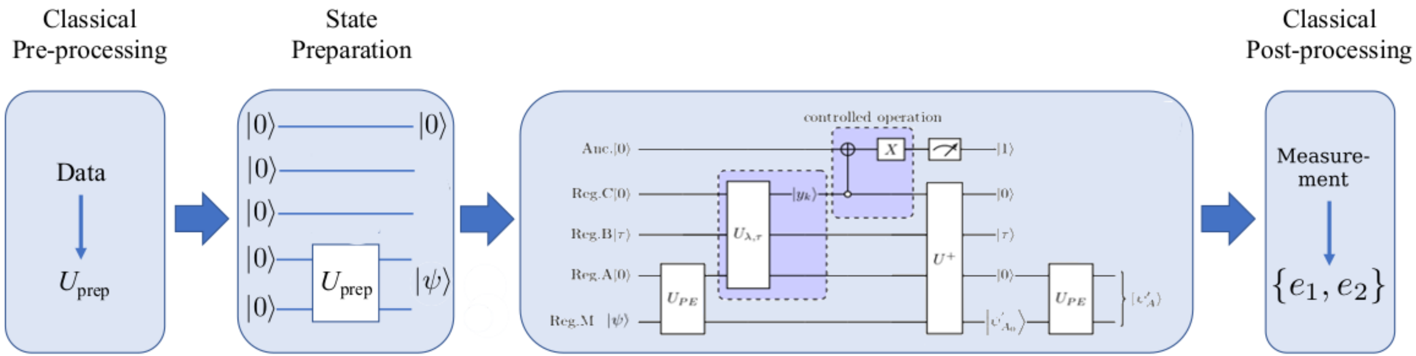


1

- problem: high dimensional data $\rightarrow$ which features are relevant?
- idea: reduce total amount of features
- techniques: eigenvalue decomposition, singular value tresholding

[1] [Shlen2014]

Quantum PCA

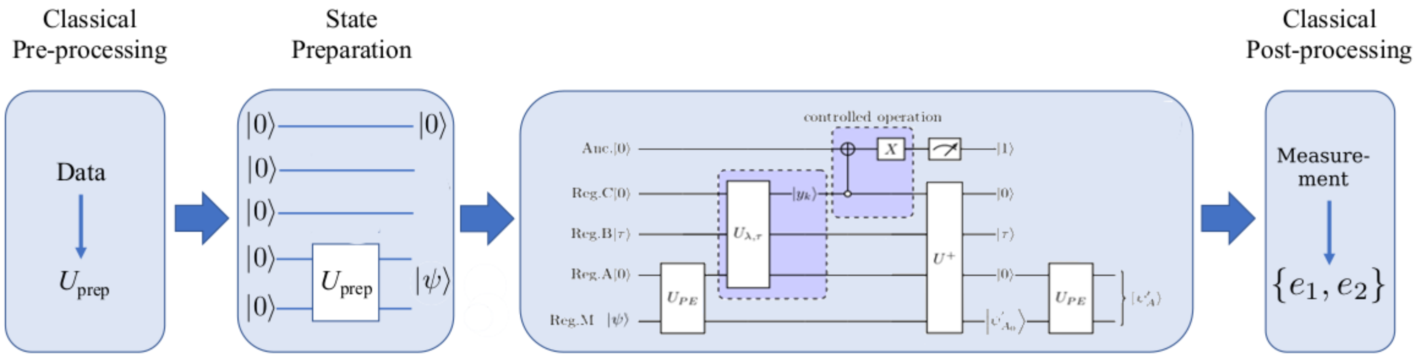


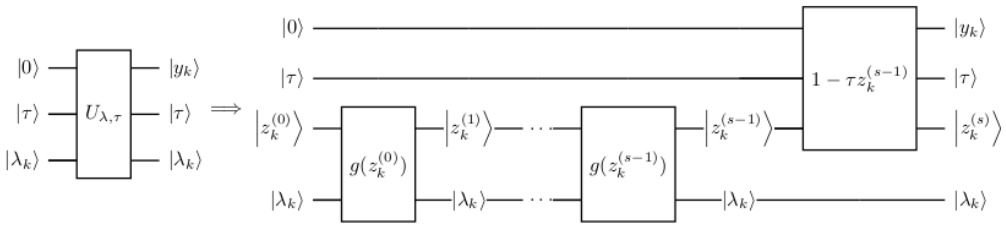
2

- operation U_{PE}: quantum phase estimation
- operation $U_{\lambda,r}$: singular value tresholding

[2] [Lokho2020], [He2021]

Quantum PCA





(a) The quantum circuit of the unitary operation $U_{\lambda,\tau}$.

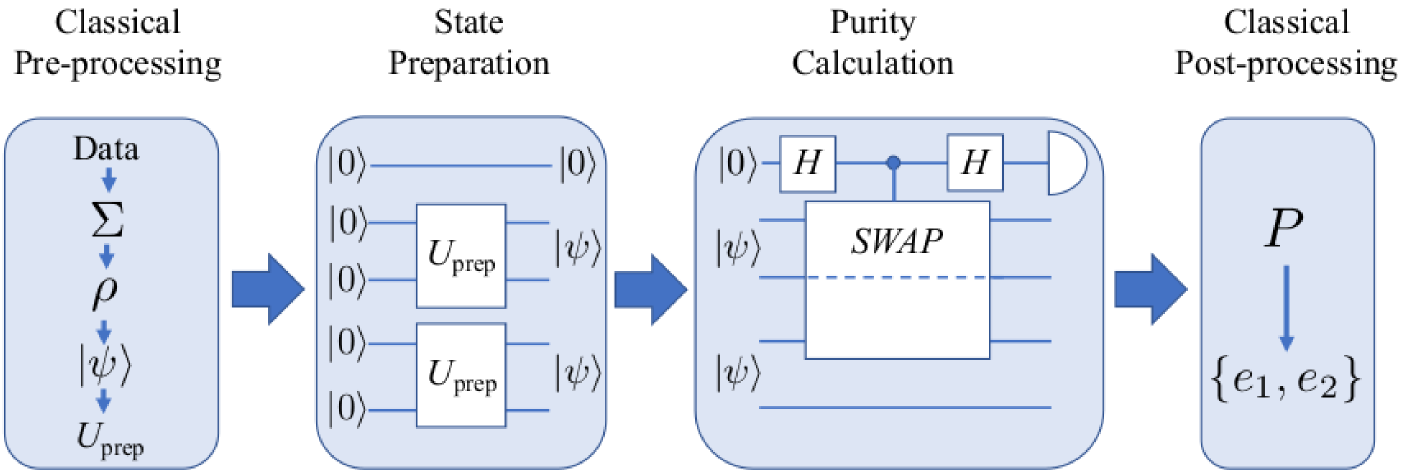
Newton Iteration

* 5 qFT multiplications
* 3 qFT additions

3

[3] [Lokho2020], [He2021]

Quantum PCA



4

- change 1: no singular value tresholding $\rightarrow$ post processing
- change 2: swap test instead of quantum phase estimation
- limitation: only 2×2 matrices Σ

[4] [Lokho2020]

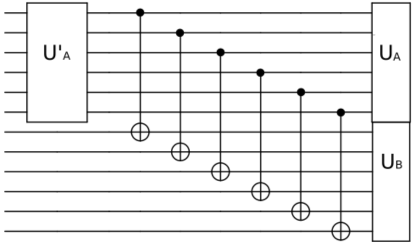
Implementation

Circuit Initialization - Classical Register (Bit-wise)

- idea: describe classical state as bit string of length n and use register of n qubits to represent as quantum state
- easy to implement, easy to measure
- limitations: large amount of qubits needed $\rightarrow$ hardware limitations, circuit complexity
- related works: [Corte2018] $\rightarrow$ reduce number of qubits, discard operations

Circuit Initialization - State Purification & Schmidt decomposition

- idea: describe a classical state as pure or mixed quantum state via density operation ρ
  - pure state: $\rho = \Sigma_{i=1}^{n} |\psi\rangle\langle\psi|$, trace $Tr(\rho) = 1$
  - mixed state: $\rho = \Sigma_{i=1}^{n} p_i |\psi_i\rangle\langle\psi_i|$, trace $Tr(\rho) < 1$
- reduced density operation: two systems A and B, then $\rho_A = Tr_B(\rho_{AB})$
  - describe entangled states
  - state purification: system A in a mixed state $\rightarrow$ a composed system AB can be found that complies with the reduced density operation ρ_A and AB in a pure state
- state preparation: use Schmidt decomposition to compute the unitary preparion operation:
  - $U_{prep} = (U_A \otimes U_B) CNOT_{AB} (U'_A \otimes I_B)$

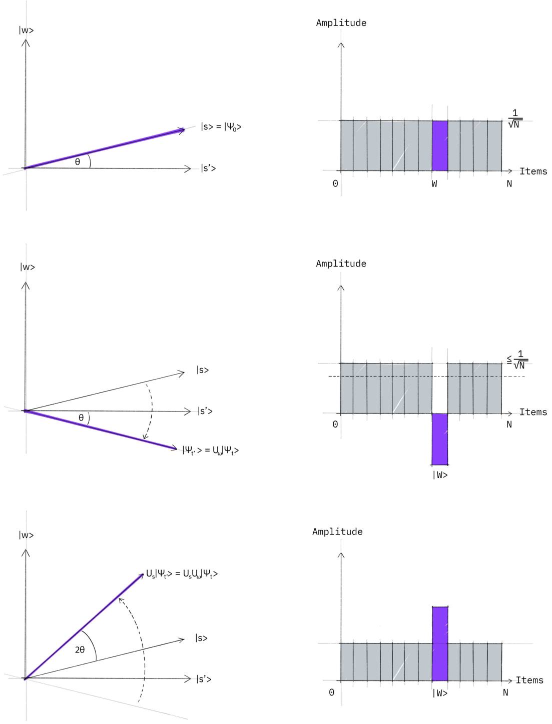


5

[5] [Lokho2020], [Niels2010]

Circuit Initialization - Amplitude Amplification

- idea: "search" the quantum state via grover algorithm
  - step 1: start in the uniform superposition $|s\rangle = H^{\otimes n}$
  - step 2: apply oracle operation U_f
  - step 3: apply reflection operation U_s, $U_s = 2|s\rangle\langle s| - 1$



6

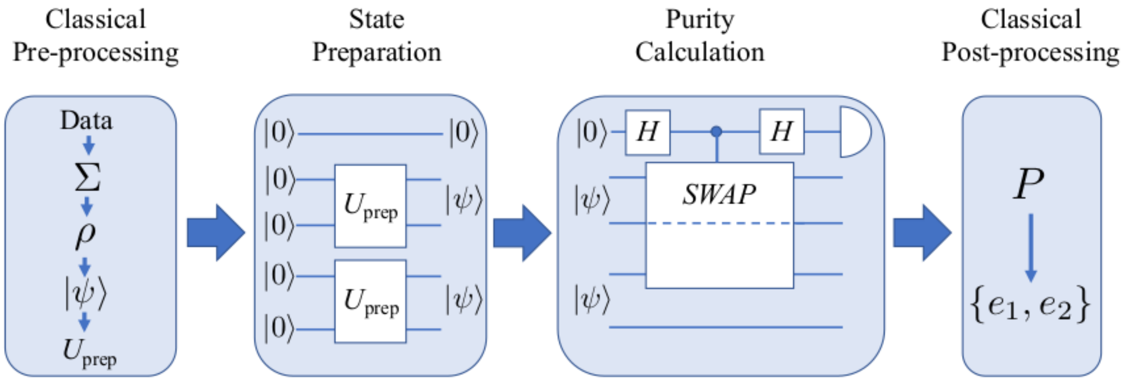
[6] [Daski2015], https://qiskit.org/textbook/ch-algorithms/grover.html

Circuit Initialization Challenges

- textbook examples often quantum states that are easy to implement, e. g. classical states
- paper examples often very brief description how to initialize quantum state
- method specifics:
  - bit-wise: properly apply discard operations
  - state purification: find composed state in respect to the density matrix of given data set
  - amplitude amplification: find oracle operation in respect to the given data set

Simple Quantum PCA with Swap Test

- used libraries: python, unittest, numpy, cirq
- cirq testing: package cirq.testing
  - assert_same_circuits → tests if two circuits are *equivalent* to each other
  - assert_has_diagram → tests the text representation



Simple Quantum PCA with Swap Test - State Preparation

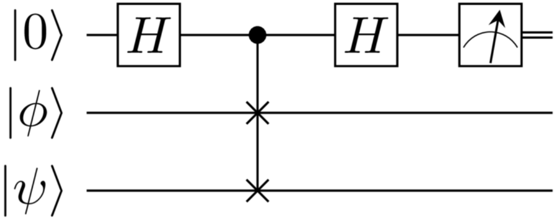
- cirq: gate vs. operation vs. circuit
  - implementation circuit initialization: internal method of "SimpleSwapTestQPCA" cirq circuit
  - unitary preparation operation to initialize $|\psi\rangle$ can be computed independently
  - circuit makes sure enough copies of $|\psi\rangle$ are initialized

```
class SimpleSwapTestQPCA(cirq.Circuit):
    def __init__(self, *contents: 'cirq.OP_TREE', prep_operation=None):
        super().__init__(*contents)
        self._anchilla = cirq.LineQubit(0)
        self._register_a = cirq.LineQubit.range(1, 3)
        self._register_b = cirq.LineQubit.range(3, 5)
        self._initialize_states(prep_operation)
        self.append(swap_test(self._anchilla,
                              self._register_a[0], self._register_b[0]))

    def _initialize_states(self, prep_operation=None):
        if prep_operation is not None:
            self.append(prep_operation.on(*self._register_a))
            self.append(prep_operation.on(*self._register_b))
```

Simple Quantum PCA with Swap Test - Purify Calculation

- swap test: operation to check level of difference of two quantum states
  - input: two quantum states, each represented with n qubits
  - output: level of equality of the two states
  - if both states are orthogonal to each other: probability to measure 0 is $\frac{1}{2}$
  - if both states are equal to each other: probability to measure 0 is 1

7



[7] https://en.wikipedia.org/wiki/Swap_test

Simple Quantum PCA with Swap Test - Purify Calculation

- cirq: gate vs. operation vs. circuit
  - implementation swap test: cirq gate "SwapTestGate" and "swap test" function which returns a cirq operation
  - drawback: gates only know qubits but not register of qubits → swap test composition makes some assumptions how qubits are entered into the gate
  - better: directly implement as cirq operation
- cirq: equality vs. value equality
- end-to-end testing: simulate circuit → find acceptable delta between expected and actual result

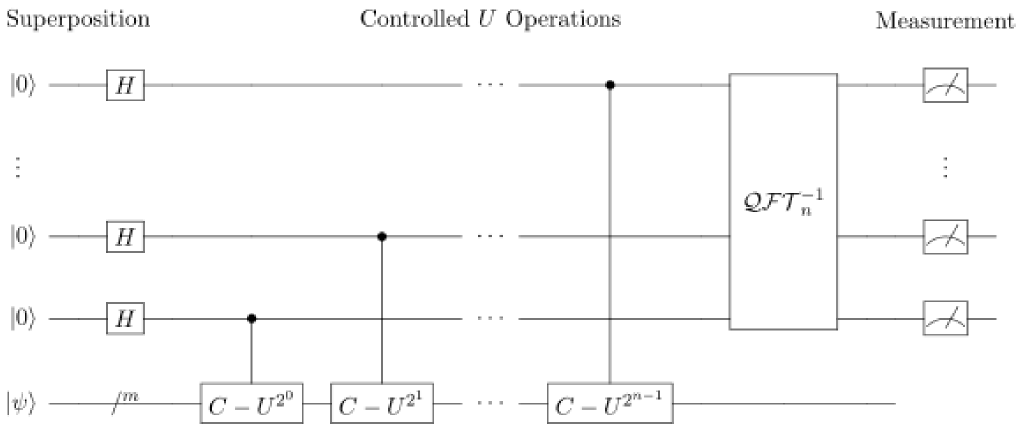
```python
@value.value_equality
class SwapTestGate(raw_types.Gate):
    def __init__(self, num_qubits=3):
        self._num_qubits = max(0, num_qubits)

    def num_qubits(self) -> int:
        return self._num_qubits

    def _value_equality_values_(self):
        return self._num_qubits

    def _decompose_(self, qubits):
        if len(qubits) == 0:
            return

        if len(qubits) < 3 or (len(qubits) - 1) % 2 != 0:
            raise NotImplementedError

        start_index_first_register = 1
        start_index_second_register = (len(qubits) + 1) // 2

        yield cirq.H(qubits[0])
        for i in range(start_index_first_register, start_index_second_register):
            yield cirq.SWAP(qubits[i], qubits[start_index_second_register - 1 + i])\
                .controlled_by(qubits[0])
        yield cirq.H(qubits[0])
```

Simple Quantum PCA with Swap Test - Post-processing

- formula to calculate the two eigenvalues[a] with purity $P = Tr(\rho^2)$:
  - $e_1 = Tr(\Sigma) * (1 + \sqrt{1 - 2(1 - P)})/2$
  - $e_2 = Tr(\Sigma) * (1 - \sqrt{1 - 2(1 - P)})/2$
- this formula is specific for 2×2 matrices and maybe even mixed states
- example from [He2021]: formula does not calculate eigenvalues correctly

[a] [Lokho2020]

Enhancements with Quantum Phase Estimation



[8]

- more generalized circuit for quantum based eigenvalue calculation
- algorithm can calculate n eigenvalues not only two
- changes in the preprocessing: need to compute unitary operation for the QPE
- changes in the post-processing: calculate eigenvalues from amplitudes

[8] https://en.wikipedia.org/wiki/Quantum_phase_estimation_algorithm

Results

Results

- possible code snippets to look into:
  - implementation swap test gate & swap test operation
  - implementation qpe operation
  - cirq unit tests
  - end-to-end test cases
  - test datasets

Conclusion

Conclusion

- different algorithms for qPCA have been proposed by research community $\rightarrow$ often additional constraints for practical implementing
  - types of limitations discussed: number of qubits, qubit layout, circuit depth, circuit noise
- integration: quantum state initialization is a non-trivial task when implementing and testing quantum algorithms
  - three techniques discussed: classical register, density matrix, amplitude amplification
- design: support through the different high-level libraries
  - python cirq library discussed: circuit, operation, gate, value equality, testing

Conclusion

- implementation: support through the different high-level libraries
  - python unit-testing and end-to-end component testing discussed: classical assertions, circuit simulation
  - not discussed: other types of testing for quantum algorithms, e. g. projection based runtime assertions
  - not discussed: error code correction to address noise or mixed states
- maintenance: enhancements on the simple qPCA algorithm applied
  - types of effects / changes discussed: preprocessing, post-processing, testing

References

- paper:

[Shlen2014] J. Shlens (2014): "A Tutorial on Principal Component Analysis" (arXiv: 1404.1100v1)

[Melke2010] D. v. Melkebeek (2010): *CS 880: Quantum Information Processing*, "Lecture 20: Density Operator Formalism"

[He2021] He et. al. (2021): "A Low Complexity Quantum Principal Component Analysis Algorithm" (arXiv: 2010.00831v2)

[Lokho2020] Lokhov et. al. (2020): "Quantum Algorithm Implementations for Beginners" (arXiv: 1804.03719v2)

[Corte2018] J. A. Cortese, T. M. Braje (2018): "Loading Classical Data into a Quantum Computer" (arXiv: 803.01958v1)

[Niels2010] M. A. Nielsen, I. L. Chuang (2010): "Quantum Computation and Quantum Information - 10th Anniversary Edition"

Daski2015 A. Daskin (2015): "Quantum "Principal Component Analysis" (arXiv: 1512.02109v1)

References

- internet ressources:
  - https://en.wikipedia.org/wiki/Quantum_phase_estimation_algorithm
  - https://en.wikipedia.org/wiki/Swap_test
  - https://qiskit.org/textbook/ch-algorithms/grover.html
  - https://qiskit.org/textbook/ch-quantum-hardware/density-matrix.html