A QCNN for Quantum State Preparation

Carnegie Vacation Scholarship

David Amorim

Week 5
(29/07/2024 - 02/08/2024)

Table of Contents

Aims for the Week

The following aims were set at the last meeting (29/07/2024):

Improve Loss Function

Work on an improved version of WILL. Incorporate some phase extraction metrics (e.g. χ, ϵ) into the loss function.

Investigate Phase Extraction

Study the relationship between mismatch and the extracted phase, i.e. study the operator $\tilde{Q}^\dagger(\hat{I} \otimes \hat{R})\tilde{Q}$.

Mitigate Barren Plateaus

Work on strategies to mitigate barren plateaus, e.g. implement layer-by-layer training.

THIS WEEK INCLUDE EXAMPLES WITH HIGH m !!
KEEP WORKING ON CHIL!! MAYBE RENAME TO QRQ??v DOES NOT WORK !!! LLLL

Table of Contents

WILL Revisited

- As discussed at the meeting on 29/07, the definition of WILL (weighted $\mathrm{L_p}$ loss) was amended to:

$$\mathrm{WILL_{p,q}} = \left(\sum_k \left| x_k - y_k \right|^p + |x_k| \left| [k]_m - \Psi([k]_n) \right|^q \right)^{1/p}, \quad (1)$$

  where the changes to the previous definition are highlighted

- Testing this for different Ψ (with $L = 6$, $m = 3$ and 600 epochs) yielded the following optimal values for p, q:

$\Psi(f)$	p	q
$\sim f$	0.25	0.5
$\sim f^2$	1	1.5
Ψ_{H23}	0.75	2

Table 1: Optimal identified p, q values for WILL

Comparing SAM, WIM, and WILL

	SAM	WIM	WILL
μ	**3.4e-2**	6.0e-2	4.5e-1
σ	1.4e-1	1.1e-1	4.7e-1
ϵ	**1.9e-2**	9.2e-2	2.6e-1
χ	**3.2e-2**	5.1e-2	3.7e-1
Ω	**4.46**	3.19	0.76

Table 2: Comparing loss function metrics for $\Psi(f) \sim f$ ($L = 6$, $m = 3$, 600 epochs)

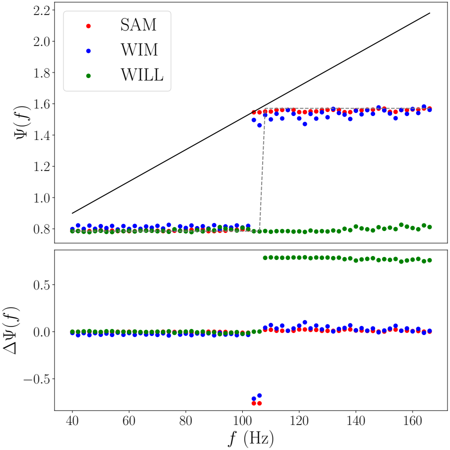

Figure 1: Comparing extracted phase functions for $\Psi(f) \sim f$ ($L = 6$, $m = 3$, 600 epochs)

Comparing SAM, WIM, and WILL

	SAM	WIM	WILL
μ	**1.9e-1**	2.3e-1	6.6e-1
σ	1.2e-1	**1.0e-1**	4.1e-1
ϵ	2.2e-1	4.2e-1	**2.8e-2**
χ	**1.9e-1**	2.0e-1	6.1e-1
Ω	**1.39**	1.05	0.57

Table 3: Comparing loss function metrics for $\Psi(f) \sim f^2$ ($L = 6$, $m = 3$, 600 epochs)

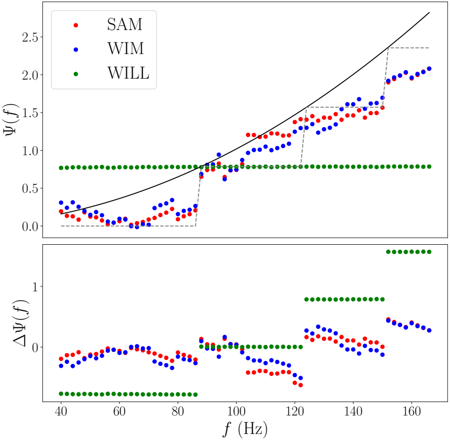

Figure 2: Comparing extracted phase functions for $\Psi(f) \sim f^2$ ($L = 6$, $m = 3$, 600 epochs)

Comparing SAM, WIM, and WILL

	SAM	WIM	WILL
μ	**6.8e-2**	8.4e-2	7.6e-2
σ	1.8e-1	**1.2e-1**	2.6e-1
ϵ	4.5e-2	1.8e-1	**7.3e-3**
χ	7.4e-2	1.0e-1	**6.2e-2**
Ω	**2.75**	2.07	2.48

Table 4: Comparing loss function metrics for Ψ_{H23} ($L = 6$, $m = 3$, 600 epochs)

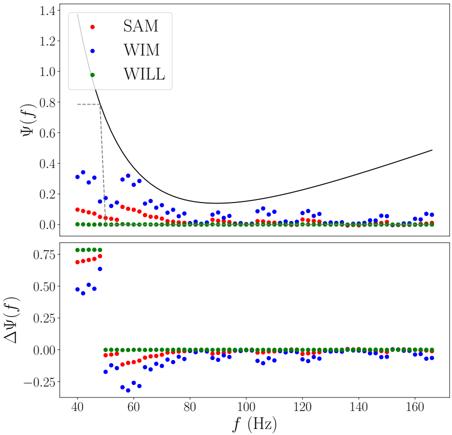

Figure 3: Comparing extracted phase functions for Ψ_{H23} ($L = 6$, $m = 3$, 600 epochs)

Other Approaches

- Attempts to define a loss function based directly on $\hat{Q}^\dagger \hat{R} \hat{Q}$, e.g. minimising χ, were unsuccessful
- This is due to the *qiskit machine learning* environment being build around sampler primitives which return quasi-probabilities instead of probability amplitudes
- Thus, phases cannot be directly taken into account for gradient calculation
- A possible work-around could be to switch to a QCNN based on an estimator primitive, which calculates the expectation value of an observable w.r.t to the state prepared by the network
- This would require the construction of an appropriate operator (note: qiskit supports non-Hermitian observables)
- Beyond this, no further *ansatze* for loss functions come to mind

An Estimator-based PQC

- Let $|\tilde{\phi}\rangle$ be the state produced by the PQC:

$$|\tilde{\phi}\rangle = \sum_k \tilde{A}(k) e^{i\tilde{\Psi}(k)} |k\rangle \tag{2}$$

- The desired output state is

$$|\phi\rangle = \sum_k A(k) e^{i\Psi(k)} |k\rangle \tag{3}$$

- An estimator-based optimiser calculates the loss and gradients for each epoch based on the expectation value

$$\mathbb{E}(\tilde{\phi}) = \langle\tilde{\phi}|\hat{O}|\tilde{\phi}\rangle = \sum_{k,k'} \tilde{A}(k')\tilde{A}(k) \exp\left(i\left[\tilde{\Psi}(k) - \tilde{\Psi}(k')\right]\right) \langle k'|\hat{O}|k\rangle\,, \tag{4}$$

  for some operator $\hat{O}$

An Estimator-based PQC

- Now construct $\hat{O}$ such that

$$\langle k'|\hat{O}|k\rangle \equiv \frac{1}{A(k')A(k)} \exp\left(-i\left[\Psi(k) - \Psi(k')\right]\right) \tag{5}$$

  for $A(k), A(k') \neq 0$
- Then

$$\mathbb{E}(\phi) = 1 \tag{6}$$

  so that we can train the network to generate $|\phi\rangle$ by minimising $|1 - \mathbb{E}(\tilde{\phi})|$
- This is highly speculative but could be worth trying?

Table of Contents

Next Steps

- ...