Postgraduate in Quantum Engineering - Final project
Universitat Politècnica de Catalunya

Quantum Shadow Tomography with neural networks

Avelino García i Pérez, Óscar Trujillo Acosta.

Tutor: Gael Sentís

Barcelona, May 13, 2021

Contents

I. Introduction and motivation

Any classical physical system is characterized at each moment in a distinctive and specific way, in a specific place and with particular characteristics. This is what we call "system state" and, in this classical sense, it depends on how we observe it (by measuring it). When we try to apply this rule to the quantum world we run into many problems because here this observation process is inherently probabilistic.

Quantum state tomography is the name given to a technique based on the realization of multiple quantum measurements that allows to completely reconstruct an unknown quantum state, with this information we can predict all the possible results of any future quantum measurement of this system.

As an example, suppose we are given an unknown state, ρ, of a single qubit. How can we determine what the state ρ is? If we have just a single copy then we cannot answer this question, the basic problem is that there is no quantum measurement which can distinguish non-orthogonal quantum states with certainty. However, we can estimate ρ if we have a large number of copies, for instance, if ρ is the quantum state produced by some experiment, then we simply repeat the experiment many times to produce many copies of the state ρ.

We can use this amount of copies to make many different measurements, many times each, to infer probabilities and these probabilities can be used to determine a density matrix which fits the best with the observations. The question is, how many copies do we need to estimate this state? This is one of the questions that we are going to address in this project.

II. Description of the problem and possible approaches

A full reconstruction of a state with n qubits making use of quantum tomography methods already needs the order of 4^n different measurement settings, this is because each qubit could be determined with the three Pauli matrices and also have the possibility of not been measure (identity matrix). Apart from this, each measure would need several repetitions, and considerable computing time to find a compatible density matrix.
Therefore, due to this exponential scaling of the number of required measurements, the brute-force approach is infeasible in computational as well as experimental resources.

A workaround is to make use of the named quantum "shadow tomography"[1,2], this technique is based on the idea that we can estimate the quantum state using only a polynomial number of state copies by making a restricted set of measurements, the same basis for example. This method does not allow you to have a complete representation of the quantum state for all bases but it does allow you to make a prediction for a specific one, which may be sufficient for certain use cases. Also, it is much easier to calculate thanks to the fact that you only need a polynomial number of measurements.

A tool that will help us to solve this problem is an artificial neural network[3]. In general, these machine learning methods perform well at finding features in a probability distribution and meanwhile, a shadow tomography with a fixed base also determines a probability distribution. Specifically we are interested in a Restricted Boltzmann Machine (RBM)[4], later we will go into detail about the characteristics of these RBMs but we can anticipate that the probability function they generate from a set of input data can be understood as the shadow of a wave function of a quantum state.

Another approach to solve the high computational demand problem that quantum tomography has is by making assumptions about the state, e.g., its rank. An instance of such approaches is compressed sensing[5]. We will not cover methods based on assumptions about the state here.

III. Applied methods and results

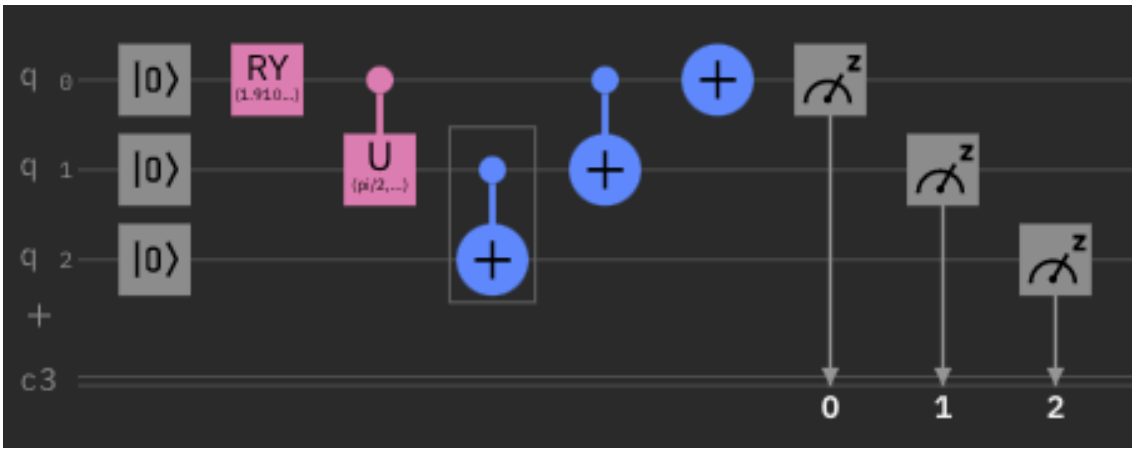
1. Quantum state

The first step to carry out this project is to perform measurements on a quantum state. To do this we have used Qiskit[6], an open-source SDK developed by IBM that allows us, among other things, to build, simulate and run quantum circuits. Using this tool we have the possibility to have the measurements directly on the quantum computers that IBM has open to the public. On the other hand, it is able to simulate the result of a perfect quantum circuit or simulate the errors we will have in a real quantum computer. That is why we have been able to develop a procedure that returns us a set of independent measurements of a simulated or real quantum circuit.

To make this project we have chosen to use the quantum state known as **W-state**, a paradigmatic N-qubit state showcasing genuine multipartite entanglement which appears in several quantum information applications, e.g. to encode the state 1 of a logical qubit in a robust way. In addition, thanks to its characteristics we can simplify the calculations, as we will see later. It is also a state that with Qiskit is relatively easy to program and scale for different numbers of qubits. It is a highly entangled quantum state which has real and positive coefficients, can be characterized by one basis and has the following shape:

$$|\Psi_W\rangle = \frac{1}{\sqrt{N}}\left(|100\ldots\rangle + |010\ldots\rangle + \cdots + |0\ldots01\rangle\right) \tag{1}$$

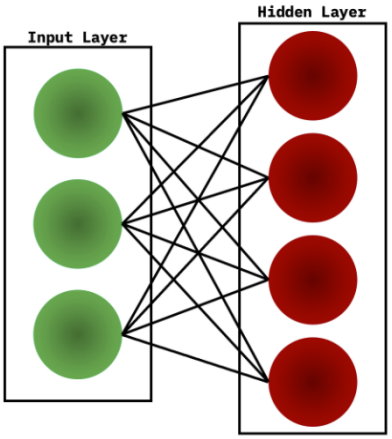
W-state is usually expressed with 3 qubits, in Qiskit can be generated by the next circuit:



As you can see we initialize the circuit with the qubits in 0 and we make the measurement in Z axis, the output of this circuit could be |001>, |010> and |100> with the same probability.

2. Restricted Boltzmann Machine (RBM)

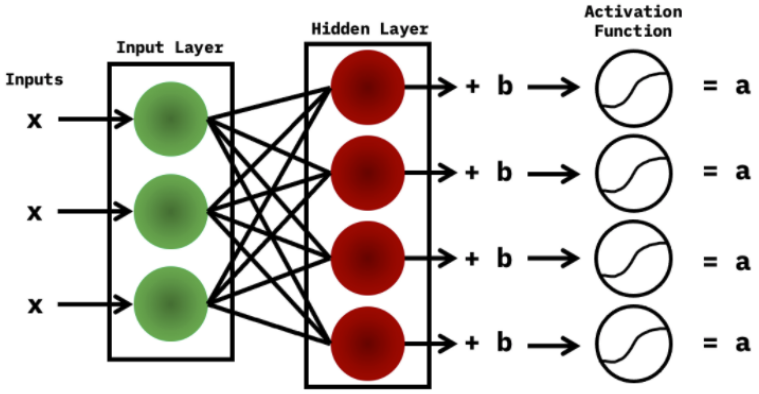
A Boltzmann Machine is a generative stochastic artificial neural network, in particular we are interested in the restricted version of it, the Restricted Boltzmann Machine (**RBM**), which is capable of learning a probability distribution given a set of inputs. This type of network consists of 2 layers, the first one is called the visible or input layer **σ** and the second one the hidden layer **h**. The expressive power of the model can be characterized by the ratio **α** = M/N between the number of hidden neurons M and visible neurons N. Each node in the visible layer is connected to every node in the hidden layer, it is considered restricted because no two nodes from the same layer share a connection. One network is characterized fundamentally by three different parameters, the **weights** or **W** which are a matrix, where the number of input nodes is the number of rows, and the number of hidden nodes is the number of columns, the input or visible layer bias array or **b** and the hidden layer bias array or **c**.



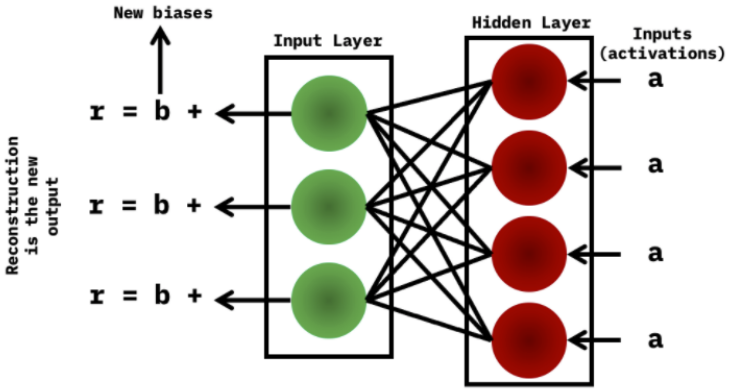
The number of nodes in the visible layer always matches the number of qubits of the dataset with which we want to train the network. The RBM will try to learn the patterns of those inputs.

An RBM is trained with two different steps, in the first step the inputs are taken into the input/visible layer, multiplied by the weights and added to the bias. After this, it goes through the activation function (sigmoid, eq 2), and the outputs decide whether the hidden state gets activated. The weights in the neural network are in a matrix, where the number of input nodes is the number of rows, and the number of hidden nodes is the number of columns. The primary hidden node obtains the vector multiplication of the inputs, and is multiplied by the first column of weights before the corresponding bias term is added to it.

$$\text{a} = \text{sigmoid}(x_1 w_1 + x_2 w_2 + x_3 w_3 + b) \tag{2}$$



The second step is the reconstruction, here we have the activations, which are the inputs at this point and are then passed to the hidden layer and then to the input later. After this, new biases are obtained, and the reconstruction is the new output.



The learning process works because the two steps happen subsequently, you first generate activations using the multiple inputs phase, then reconstruction takes place. When the reconstruction is taking place in an epoch, the main goal is to decrease the reconstruction error so that the weights are then adjusted per iteration accordingly by the algorithm to decrease the reconstruction error. To build this RBM we have used Tensorflow[7,8], a powerful open-source library for machine learning developed by Google.

This neural network is an energy-based model, sharing many properties of physical models in statistical mechanics. Through network training, an equilibrium state is reached in which the global states with the highest probability have the lowest energies. With the weights W and the nodes states of a trained RBM we can calculate a probability distribution given by the Boltzmann distribution

$$p_{\kappa}(\boldsymbol{\sigma}, \mathbf{h}) = \mathrm{e}^{\sum_{ij} W_{ij}^{\kappa} h_i \sigma_j + \sum_j b_j^{\kappa} \sigma_j + \sum_i c_i^{\kappa} h_i} \tag{3}$$

where we omitted the normalization and **κ** now consists of the weights **Wκ** connecting the two layers and the fields (biases) **bκ** and **cκ** coupled to each visible and hidden neuron, respectively. The distribution (of interest) over the visible layer is obtained by marginalization over the hidden degrees of freedom

$$p_{\kappa}(\boldsymbol{\sigma}) = \sum_{\mathbf{h}} p_{\kappa}(\boldsymbol{\sigma}, \mathbf{h}) = \mathrm{e}^{\sum_j b_j^{\kappa}\sigma_j + \sum_i \log\left(1+\mathrm{e}^{c_i^{\kappa}+\sum_j W_{ij}^{\kappa}\sigma_j}\right)} \tag{4}$$

In all our tests we will measure the qubits in the Z-base multiple times, these samples will make up the training set we will use to train the different RBMs we will create. Once trained, the objective is to receive as input a possible outcome of a configuration of measurements in each given qubit, and to generate an output that corresponds to the amplitude of the corresponding ket within the state that we want to determine.

The RBM wave-function can be defined as

$$\psi_{\lambda,\mu}(\mathbf{x}) = \sqrt{\frac{p_{\lambda}(\mathbf{x})}{Z_{\lambda}}}\, \mathrm{e}^{i\phi_{\mu}(\mathbf{x})/2} \tag{5}$$

where the networks pλ(x) and φμ(x) represent, respectively, the amplitude and phase of the state, and Z = ∑p(σ) is the normalization constant.

However, since we are using w state, each coefficient ΨW (σ) is real and positive, we only need to learn the amplitudes and we can adopt a simpler version of the RBM wave-function, that is

$$\psi_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) = \sqrt{\frac{p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}{Z_{\boldsymbol{\lambda}}}} \tag{6}$$

With this RBM we can make use of the Gibbs sampling algorithm in order to obtain a sequence of observations from a specified multivariate probability distribution. This procedure is very efficient since each neuron in one layer of the RBM is connected only to neurons of a different layer, thus enabling us to sample all units (in one layer) simultaneously.

3. Overlap

In order to train the network we need a way to quantify the performance of training, a good way to do this is to calculate the overlap **O** between the W state wave-function and the RBM wave-function

$$O = \langle \Psi_W | \psi_{\boldsymbol{\lambda}} \rangle = \sum_{\boldsymbol{\sigma}} \Psi_W(\boldsymbol{\sigma})\, \psi_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}) \tag{7}$$

where ΨW(σ) = δ(σ−2k)/√N for k ∈ (0,...,N −1). As we cannot perform the full sum in Eq.7 for large system sized N, and we don't know the normalization constant Zλ, we instead compute the square of the overlap as

$$\begin{aligned} O^2 &= \frac{\langle \Psi_W | \psi_{\boldsymbol{\lambda}} \rangle}{\langle \psi_{\boldsymbol{\lambda}} | \psi_{\boldsymbol{\lambda}} \rangle} \times \frac{\langle \Psi_W | \psi_{\boldsymbol{\lambda}} \rangle}{\langle \Psi_W | \Psi_W \rangle} \\ &= \frac{\sum_{\boldsymbol{\sigma}} |\psi_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})|^2 \, \frac{\Psi_W(\boldsymbol{\sigma})}{\psi_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}}{\sum_{\boldsymbol{\sigma}} |\psi_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})|^2} \times \frac{\sum_{\boldsymbol{\sigma}} |\Psi_W(\boldsymbol{\sigma})|^2 \, \frac{\psi_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}{\Psi_W(\boldsymbol{\sigma})}}{\sum_{\boldsymbol{\sigma}} |\Psi_W(\boldsymbol{\sigma})|^2} \\ &= \left\langle \frac{\Psi_W(\boldsymbol{\sigma})}{\sqrt{p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}} \right\rangle_{p_{\boldsymbol{\lambda}}} \times \left\langle \frac{\sqrt{p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma})}}{\Psi_W(\boldsymbol{\sigma})} \right\rangle_{|\Psi_W|^2} \\ &= \left(\frac{1}{n} \sum_{j=1}^{n} \frac{1}{\sqrt{p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma}_j)}} \sum_{k=0}^{N-1} \frac{\delta(\boldsymbol{\sigma}_j - 2^k)}{\sqrt{N}} \right) \times \left(\sum_{k=0}^{N-1} \sqrt{\frac{p_{\boldsymbol{\lambda}}(\boldsymbol{\sigma} = 2^k)}{N}} \right) \end{aligned} \tag{8}$$

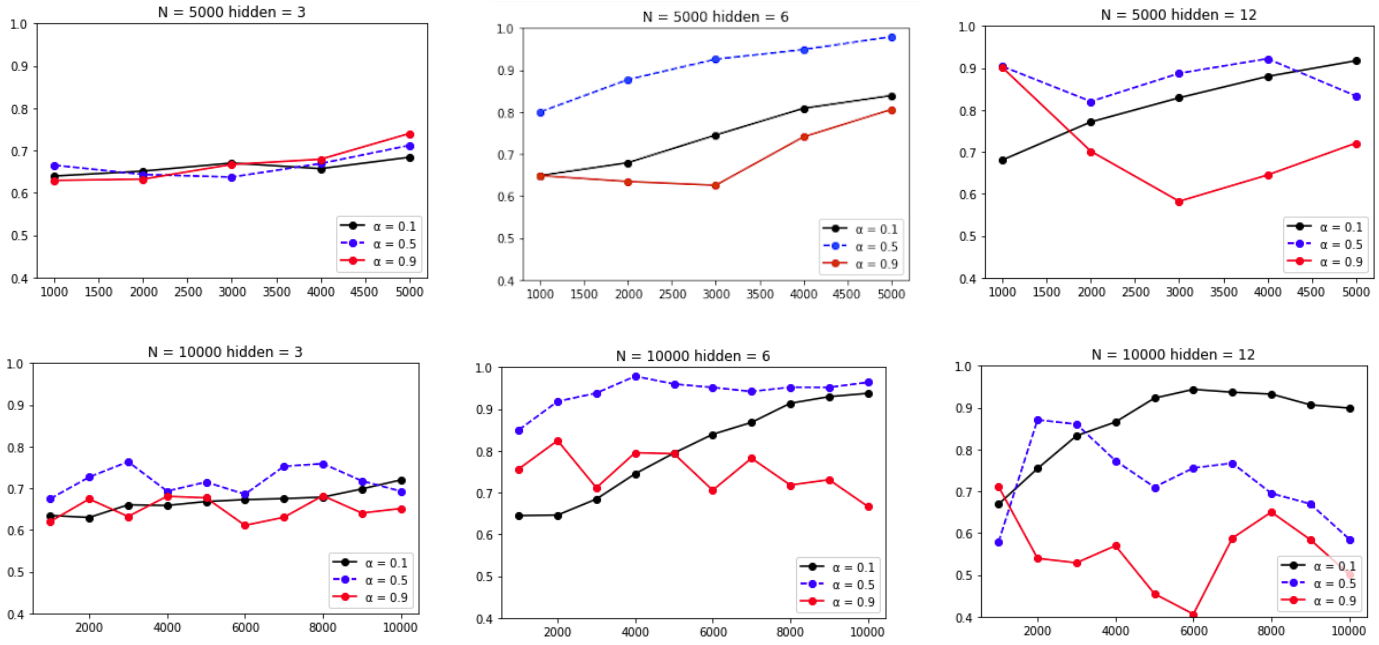
where the qubits configurations σ_j are drawn directly from the trained RBM distribution $p_\lambda(\sigma)$ by performing block Gibbs sampling from the two conditional distributions $p_\lambda(\sigma \mid h)$ and $p_\lambda(h \mid \sigma)$.

4. Results

In the following graphs we will analyze different parameters of the trained RBM, it is important to know what each variable means:

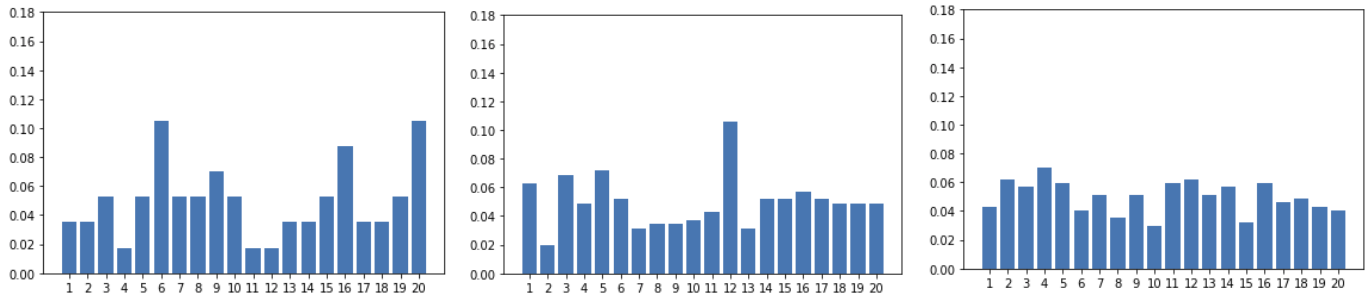
- α: it indicates the network learning rate, it is a configurable hyperparameter that has a value between 0 and 1 and controls how quickly the model is adapted to the problem. A learning rate that is too large can cause the model to converge too quickly to a suboptimal solution, whereas a learning rate that is too small can cause the process to get stuck.
- N: Is the number of measurements we use to train the RBM.
- Epoch: Is the number of times the learning algorithms will be executed, in each cycle (epoch) all the training data goes through the neural network so that it learns about them, if there are 10 cycles and 1000 data, each cycle the 1000 data will go through the neural network.

In order to determine which were the best parameters to train an RBM with for example 20 qubits we start to make tests with a smaller state as this reduces complexity and allows us to do overlap calculation and estimation more quickly. In the next graphics we see the evolution of the overlap and the estimated overlap obtained with an RBM trained with 5.000 different measurements of a 6 qubits w-state and we can compare the results with different learning rates. In these cases the number of hidden neurons is the same as the visible ones, 6.



We can see how a 0.5 learning rate has in general better results than the others 2 (0.1 and 0.9). On the other hand if we see the behaviour of the hidden neurons, we conclude the best option is to use the same number of hidden neurons as visible.

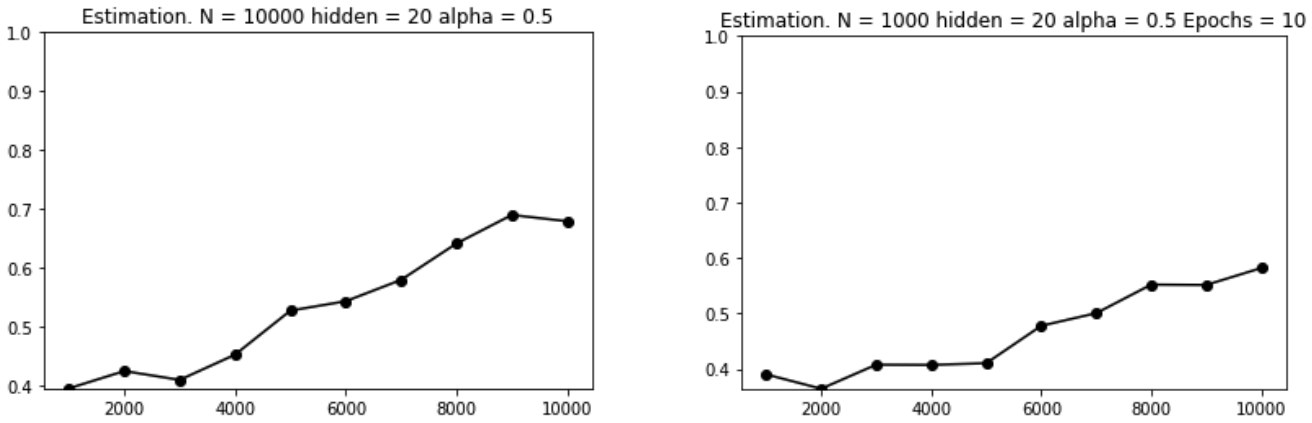
Thanks to the previous data we have selected a learning rate α = 0.5 to make the rest of the test with a larger number of qubits. In the next graphics we compare the occurrence of each of the superposed states in the W state for 20 qubits. Note that the total dimension of a state of 20 qubits is already well out of reach for standard full tomography methods. We plot three histograms obtained by sampling a RBM trained on a data set containing 50, 1,000 and 20,000 independent samples. In the three cases the RBM have been trained with a learning rate α = 0.5 and the same number of hidden neurons as visible ones, 20.



50 1.000 20.000

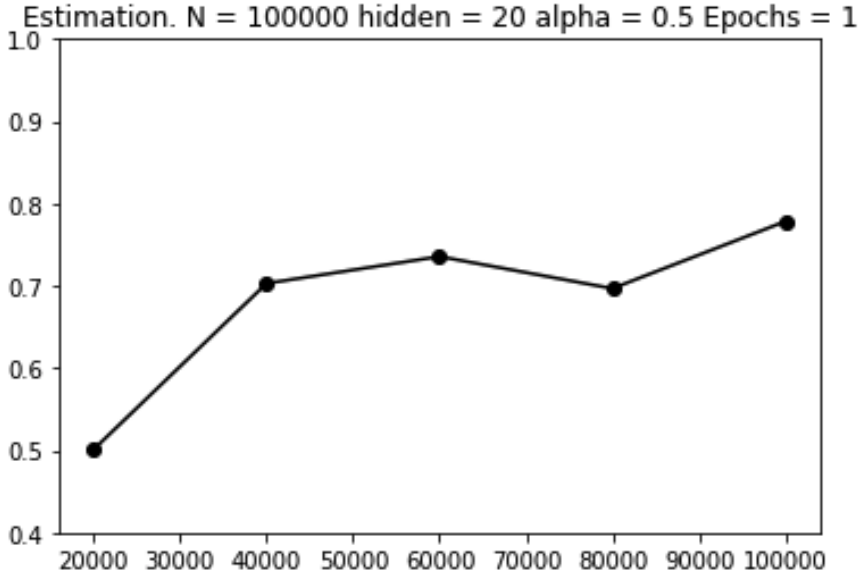
We can see how the distribution of occurrences improves when the RBM is trained with a larger dataset which means that the RBM has a better representation of the state when the dataset is larger.

In the next graphics we plot the evolution of the estimated overlap using a different number of epochs and dataset but the same number of steps.



We can see it is better to use a greater dataset rather than a smaller dataset with several learning loops (epochs).

In the last test we try to show how the overlap estimation increases by using a greater number of dataset. In this case we use a total of 100.000 independent measurements and we get a max overlap of 0.78.



IV. Conclusions and possible improvements

In conclusion we can say that making use of RBM and, in general, machine learning to solve the problem of quantum tomography raised at the beginning of this project is a very good approach due to their power, flexibility and ease of use. Specifically, our results suggest that the RBM approach performs well with a highly entangled quantum state like W-state whereas we can't say the same for structureless or random states. The study of these other cases could be the next step to take to complete this project.

The main reference for the development to this work has been the paper *Shadow Tomography of Quantum States*[1] from which we have been able to extract most of the formulas and methods, however we have found different unspecified variables when training the RBM, such as the learning rate, the epochs and the number of bounces in the Gibbs sampling.

This research can be improved by adding more comparison between different configurations or using different approaches to solve the problem and comparing the results. We can make the same tests with different quantum states, like GHZ state for example, we can study how the noise in the quantum measurements affects the RBM training or use another neural network structure instead of RBM, test it by covering methods based on assumptions like compressed sensing as we mentioned in the possible approaches... In summary, we can say that this type of study opens up many interesting research areas in which the comparison of results is important to obtain an optimal configuration that allows us to achieve our objectives.

References

1. Aaronson, Scott. *Shadow Tomography of Quantum States*. arXiv:1711.01053. 2017.
2. Huang, Hsin-Yuan., Kueng, Richard., Preskill, John. *Predicting Many Properties of a Quantum System from Very Few Measurements*. arXiv:2002.08953. 2020.
3. Torlai, G., Mazzola, G., Carrasquilla, J. et al. *Neural-network quantum state tomography*. Nature Phys 14, 447–450. 2018. https://doi.org/10.1038/s41567-018-0048-5
4. Hinton, Geoffrey. *A Practical Guide to TrainingRestricted Boltzmann Machines*. 2010
5. Gross, D., Liu, Y.-K., Flammia, S. T., Becker, S. & Eisert, J. *Quantum state tomography via compressed sensing*. Phys. Rev. Lett. 105, 150401. 2010.
6. *Qiskit: An Open-source Framework for Quantum Computing.* 2019.
7. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems.* 2015.
8. https://developer.ibm.com/technologies/deep-learning/tutorials/build-a-recommendation-engine-with-a-restricted-boltzmann-machine-using-tensorflow/
9. G.I. Struchalin, Ya.A. Zagorovskii, E.V. Kovlakov, S.S.Straupe, S.P. Kulik. *Experimental Estimation of Quantum State Properties from Classical Shadows*. arXiv:2008.05234. 2020
10. Carleo, Giuseppe., Troyer, Matthias. *Solving the quantum many-body problem with artificial neural networks*. arXiv:1606.02318. 2017.
11. Link to code https://github.com/QuantumBCN2021/RBM