

Tutorial **Solutions**: Machine learning ground states with RNN wave functions

Quantum and AI Career Trajectories Mini-Course: Computational Methods and their Applications

Mohamed Hibat-Allah
University of Waterloo | Perimeter Institute

May 24th, 2024

In this tutorial, you will learn how to use exact diagonalization (ED) and RNN wavefunctions to find the ground state of quantum many-body systems.

Recall from previous lectures in this school that the goal of Variational Monte Carlo (VMC) is to find the ground state of quantum many-body systems, where the main idea is to take an ansatz (called a variational wavefunction) “ $|\Psi_\lambda\rangle$ ” to find the ground state energy E_G by minimizing the variational energy $E_\lambda = \langle \Psi_\lambda | \hat{H} | \Psi_\lambda \rangle$ over a set of the variational parameters $\lambda = \{\lambda_1, \lambda_2, \dots, \lambda_{N_p}\}$. The latter can be done using a learning algorithm such as gradient descent. Here, we will use the learning algorithm known as the Adam optimizer [1].

In this tutorial, we use **positive RNN wavefunctions** to find the ground state of the Transverse-field Ferromagnetic Ising Model (TFIM) in 1D, whose Hamiltonian is given by

$$\hat{H}_{\text{TFIM}} = -J_z \sum_{i=1}^{N-1} \hat{\sigma}_i^z \hat{\sigma}_{i+1}^z - B_x \sum_{i=1}^N \hat{\sigma}_i^x,$$

where N is the number of spins, J_z , B_x are respectively the uniform ferromagnetic coupling between the spins and the uniform transverse magnetic field applied to all the spins. Here we assume open boundary conditions. For the sake of simplicity, we restrict the system size to $N = 10$ spins but you are encouraged to investigate larger system sizes in case you are interested. The parameter values are also restricted to $J_z = 1$ and $B_x = 1$, which corresponds to the critical point of this model. The jupyter notebook of this tutorial can be found on [Google colab](#).

Assuming a given spin configuration $\sigma = (\sigma_1, \sigma_2, \dots, \sigma_N)$ along the z-basis, where $\sigma_i = \pm 1$, we recall from the RNN lectures that the positive RNN wavefunction architecture (shown in Fig. 1) computes the amplitude $\Psi_{\text{RNN}}(\sigma)$ as follows:

- **Step 1:** the green box “RNN” in Fig. 1 takes two inputs $(\sigma_{n-1}, \mathbf{h}_{n-1})$ and outputs \mathbf{h}_n as

$$\mathbf{h}_n = f(U\sigma_{n-1} + V\mathbf{h}_{n-1} + \mathbf{b}),$$

where f is a non-linear activation function (equal to Tanh by default in the [jupyter notebook](#)). The dimension of the memory state \mathbf{h}_n is known as the

“number of memory units”, and you will be able to examine its effect in this tutorial.

- **Step 2:** h_n is fed into a Softmax fully-connected layer “S” (shown using magenta circles in Fig. 1) to compute the conditional probability P_n as

$$P_n \equiv P(\sigma_n | \sigma_{n-1}, \dots, \sigma_1) = \text{Softmax}(W\mathbf{h}_n + \mathbf{c}) \cdot \sigma_n,$$

where σ_n is a one-hot representation of the spin σ_n .

- **Step 3:** the amplitude of the spin configuration σ is computed as

$$\Psi_{\text{RNN}}(\sigma) = \sqrt{P_1 P_2 \dots P_N}.$$

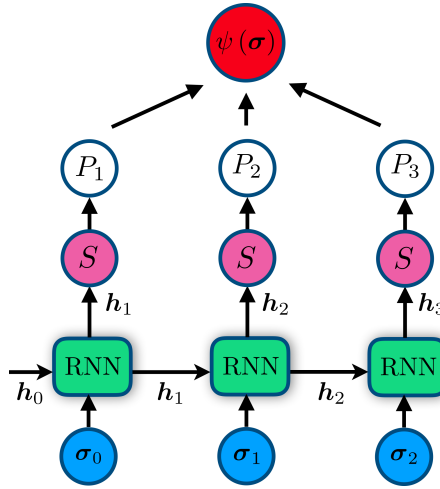


Figure 1: An illustration of positive RNN wavefunctions [2].

Within this set-up, the variational parameters λ that we will be optimizing over are the weights U, V, W and the biases \mathbf{b}, \mathbf{c} . You can read more about this approach in Ref. [2].

To verify our results, we compare with exact diagonalization (ED), which can be thought of as a diagonalization of the 1D TFIM Hamiltonian. As an illustration, if $N = 2$, then $\hat{H}_{\text{TFIM}} |\sigma_1 \sigma_2\rangle = -J_z \sigma_1 \cdot \sigma_2 |\sigma_1 \sigma_2\rangle - B_x |\bar{\sigma}_1 \sigma_2\rangle - B_x |\sigma_1 \bar{\sigma}_2\rangle$, where $\bar{\sigma}_i$ denotes that the spin σ_i is flipped. As a consequence, the Hamiltonian can be put in matrix format with size $2^N \times 2^N = 2^2 \times 2^2$ as

$$\hat{H}_{\text{TFIM}} = \begin{bmatrix} -J_z & -B_x & -B_x & 0 \\ -B_x & J_z & 0 & -B_x \\ -B_x & 0 & J_z & -B_x \\ 0 & -B_x & -B_x & -J_z \end{bmatrix},$$

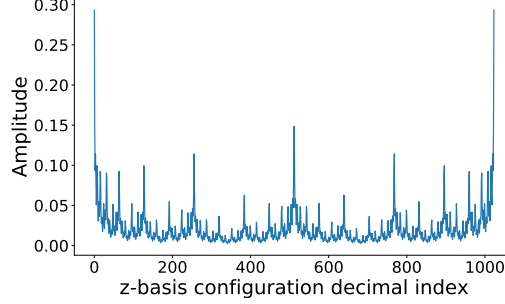
which one can diagonalize to find the exact ground state.

1. Check using exact diagonalization (ED), for $N = 10$, $J_z = 1 = B_x$, that the ground state of 1D TFIM has positive amplitudes.

Hint: You may find it helpful to plot the amplitudes of the ground state provided by the function “ED_1DTFIM” in Sec. 1 of the [jupyter notebook](#). You

are more than welcome to check how the diagonalization is performed in the code, but you are not required to understand all the details.

Solution: The results are shown in the following figure:



where we notice indeed that all the amplitudes of the ground state are positive.

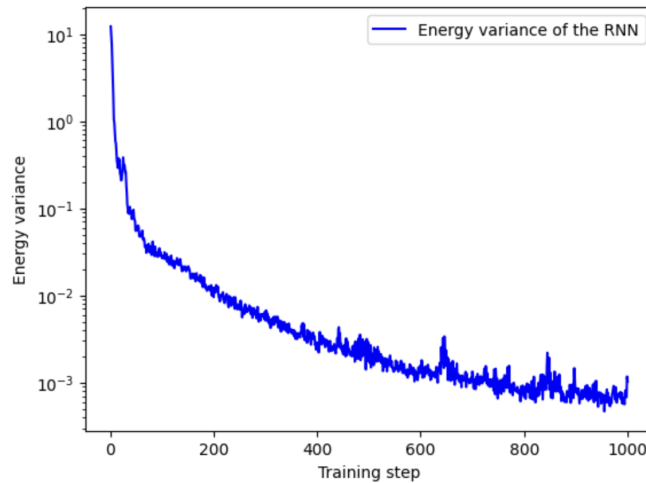
2. The property found in the previous question is a consequence of the negativity of the non-diagonal elements of the TFIM Hamiltonian (cf. Perron-Frobenius theorem). Argue, in this case, that it is enough to use a **positive RNN wavefunction**.

Solution: Since the amplitudes are all positive, then we can consider the ground state amplitudes as a square root of a probability distribution, which can be calculated by a positive RNN wavefunction. You can refer to Step 3 above.

3. Estimate the ground state energy of the 1D TFIM using a positive RNN wavefunction for $N = 10$, $J_z = 1 = B_x$ (see Sec. 2 of the [jupyter notebook](#)). Check the validity of the zero-variance principle (i.e., does the energy variance approach zero?). Compare with the ground state energy given by ED (see Sec. 3 of the [jupyter notebook](#)).

Solution: The ground state energy given by ED is equal to $E_{exact} = -12.3815$. You can check that the energy provided by the RNN wavefunction is very close to the exact value of the ground state energy.

If we also plot the energy variance, we find the following result:



where we can see that the energy variance is getting closer and closer to zero as we do more gradient descent steps. Indeed, this is coherent with what we expect from the zero-variance principle.

4. Study the effect of a certain hyperparameter:

- **Hyperparameter 1:** number of memory units (i.e., the size of the hidden state \mathbf{h}_n).
- **Hyperparameter 2:** activation function f .
- **Hyperparameter 3:** number of samples.
- **Hyperparameter 4:** learning rate.
- **Hyperparameter 5:** type of RNN cell, i.e., either a vanilla or a GRU cell.

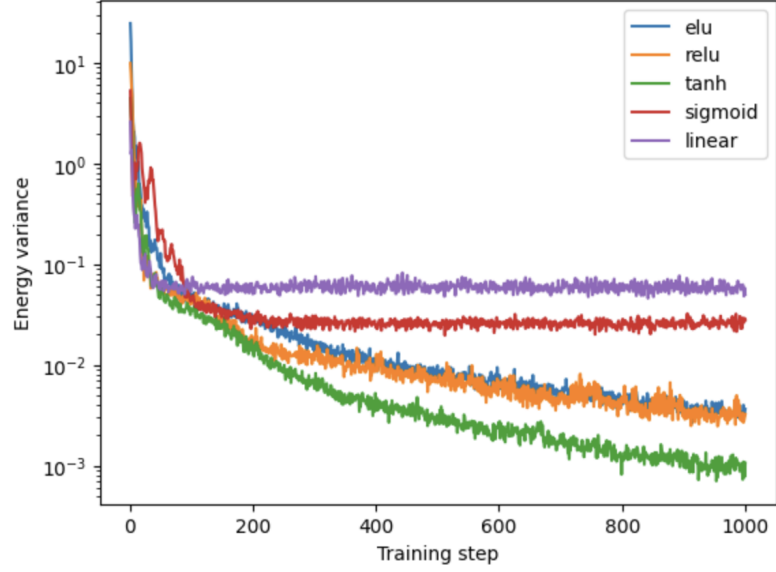
Play with your chosen hyperparameter and show how the accuracy changes. You can use a for loop to run multiple VMC optimizations and gather the energy variances in one plot to compare the convergence behavior. Feel free to add more code and make plots that visualize the trends.

For the study of activation functions, you can use Relu, Elu, Tanh, Sigmoid, or no activation function. For the study of the number of memory units and the number of samples, you can try changing the corresponding values in powers of 2. You may want to check Ref. [2] for inspiration.

Solution: You can check the results in Ref. [2], where Fig. 6.(a) shows that the energy variance is getting smaller as we increase the number of memory units. Indeed, this observation confirms that increasing the size of the memory state increases the expressivity of the RNN wavefunction.

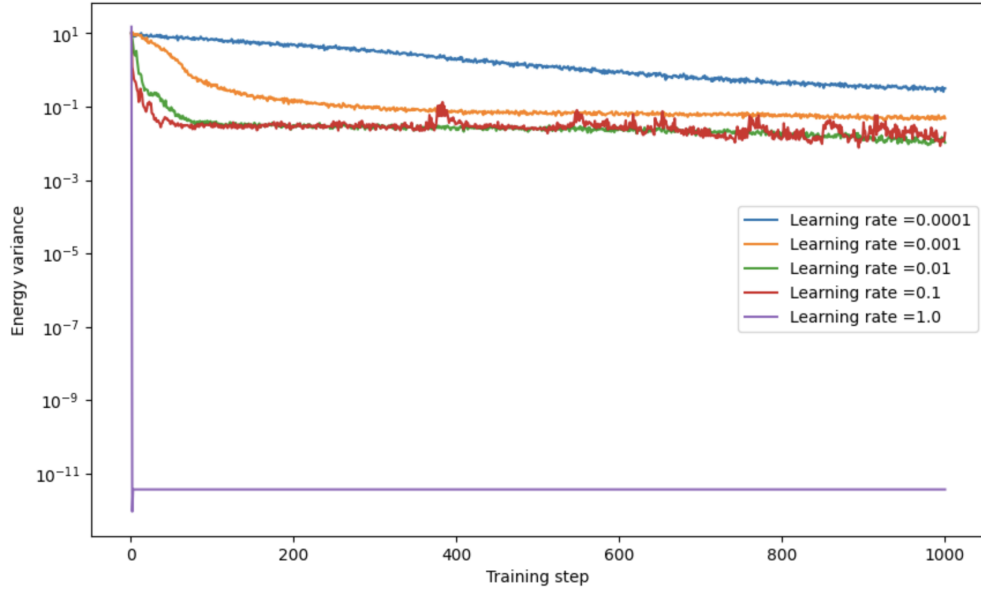
In the same reference, Fig. 9 shows that the energy variance weakly depends on the number of samples (for values between 100 and 1000). One possible explanation for this observation is that in principle only one sample σ is enough to achieve the ground state energy, as near convergence, we expect that $E_{loc}(\sigma) \approx E_G$. Hence, adding more samples after a certain threshold seems to be only helpful for reducing the statistical estimation of observables, and not a major factor for the training.

For the choice of the activation function, `tanh` is the winning function, but this is not guaranteed to be the case for other hyperparameters. The results can be shown in the following plot:



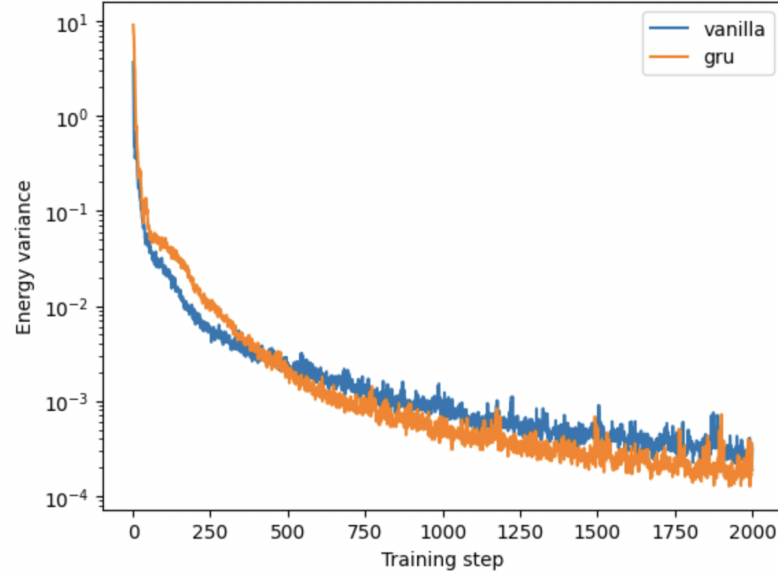
The linear activation function has the highest energy variance among the other activation functions.

The investigation of the learning rate was accomplished in powers of ten as given in the following illustration:



where it is clear that large learning rates and small learning rates are not optimal. The intermediate learning rate $lr = 0.01$ is best in terms of energy variance with a smooth convergence. For $lr = 1.0$, the optimization was stuck in a local minimum where the energy variance is very small (the corresponding energy was found to be close to -9 which is higher compared to the ground state energy). This is an example where the energy variance could be a misleading indicator for convergence accuracy. The variational energy in this case can often provide a complimentary picture.

After comparing a vanilla RNN with a tanh activation to a GRU, we obtain the following plot:



where it is clear that GRU outperforms the vanilla RNN.

5. Try switching the transverse magnetic field B_x from $+1$ to -1 . Do you expect the ground state energy to change? (You can check with ED if you are not sure about your answer). Is the ground state still positive? Can the pRNN wavefunction still find an approximate estimation of the ground state energy? Why? What can we do to fix the issue?

Solution: The ground state energy does not change, because by symmetry we can flip the x -axis to the opposite direction and hence change B_x back to $+1$. As a consequence, the ground state energy does not change as confirmed by exact diagonalization.

In this case, the non-diagonal matrix elements of the TFIM Hamiltonian are no longer negative, which means that the ground state amplitudes are no longer positive (Perron-Frobenius theorem does not hold). As a result, we cannot write the amplitudes as a square root of a probability distribution. Hence, we cannot use a positive RNN wavefunction anymore.

To go around this issue (known as the sign problem), we use complex RNN wavefunctions which you can read more about in Ref. [2].

PS: If you have finished the tutorial early, you can take the best hyperparameters you found and play with larger system sizes or different values of the transverse magnetic field B_x , to see how the positive RNN wavefunction performs (see Sec. 4 of the [jupyter notebook](#)).

To compare with the exact results, you can make use of the [exact expression of the ground state energy](#) at the critical point ($J_z = 1, B_x = 1$) for open boundary conditions:

$$E_G = 1 - \frac{1}{\sin\left(\frac{\pi}{2(2N+1)}\right)}.$$

If you are interested to see a tutorial about the application of complex RNN wavefunctions, you can check [this jupyter notebook](#) with more details in Ref. [2].

For more details about the applications of RNN wavefunctions you can check Refs. [2–10].

Finally, if you are interested in playing with Transformer wave functions, you can check the code posted on this repository https://github.com/APRIQuOt/VMC_with_LPTFs.

Acknowledgments

I would like to acknowledge Lauren Hayward for her valuable help with the writing and editing of this tutorial. I also thank Sylvain Capponi for the helpful feedback.

References

1. Kingma, D. P. & Ba, J. *Adam: A Method for Stochastic Optimization* 2014. <https://arxiv.org/abs/1412.6980>.
2. Hibat-Allah, M., Ganahl, M., Hayward, L. E., Melko, R. G. & Carrasquilla, J. Recurrent neural network wave functions. *Phys. Rev. Research* **2**, 023358. <https://link.aps.org/doi/10.1103/PhysRevResearch.2.023358> (2 June 2020).
3. Roth, C. *Iterative Retraining of Quantum Spin Models Using Recurrent Neural Networks* 2020. <https://arxiv.org/abs/2003.06228>.
4. Carrasquilla, J. & Torlai, G. How To Use Neural Networks To Investigate Quantum Many-Body Physics. *PRX Quantum* **2**, 040201. <https://link.aps.org/doi/10.1103/PRXQuantum.2.040201> (4 Nov. 2021).
5. Casert, C., Vieijra, T., Whitelam, S. & Tamblyn, I. Dynamical Large Deviations of Two-Dimensional Kinetically Constrained Models Using a Neural-Network State Ansatz. *Phys. Rev. Lett.* **127**, 120602. <https://link.aps.org/doi/10.1103/PhysRevLett.127.120602> (12 Sept. 2021).
6. Hibat-Allah, M., Inack, E. M., Wiersema, R., Melko, R. G. & Carrasquilla, J. Variational neural annealing. *Nature Machine Intelligence* **3**, 952–961. ISSN: 2522-5839. <https://doi.org/10.1038/s42256-021-00401-3> (Nov. 2021).
7. Luo, D. *et al.* *Gauge Invariant Autoregressive Neural Networks for Quantum Lattice Models* 2021. <https://arxiv.org/abs/2101.07243>.
8. Morawetz, S., Vlugt, I. J. S. D., Carrasquilla, J. & Melko, R. G. U(1)-symmetric recurrent neural networks for quantum state reconstruction. *Physical Review A* **104**. <https://doi.org/10.1103/PhysRevA.104.012401> (July 2021).
9. Czischek, S., Moss, M. S., Radzihovsky, M., Merali, E. & Melko, R. G. *Data-Enhanced Variational Monte Carlo for Rydberg Atom Arrays* 2022. <https://arxiv.org/abs/2203.04988>.
10. Hibat-Allah, M., Melko, R. G. & Carrasquilla, J. Supplementing Recurrent Neural Network Wave Functions with Symmetry and Annealing to Improve Accuracy. *Machine Learning and the Physical Sciences (NeurIPS 2021)*. https://ml4physicalsciences.github.io/2021/files/NeurIPS_ML4PS_2021_92.pdf.