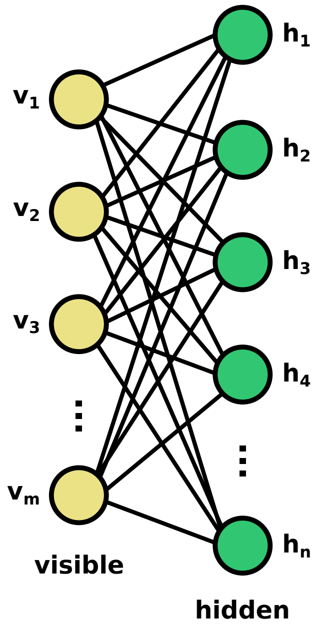
J. Carrasquilla, E. Inack, G. Torlai, R. Melko, L. Hayward Sierens

Tutorial 4: Learning classical thermodynamic observables with restricted Boltzmann machines

May 30, 2019

The objective of this tutorial is to train a restricted Boltzmann machine (RBM) on spin configurations and then sample new spin configurations from the trained RBM. You will compute thermodynamic observables from these samples and compare to known results (from Monte Carlo simulation). You will use and modify the Python programs `rbm.py`, `tutorial4_train_ising2d.py` and `tutorial4_sample_ising2d.py`

As seen in the lectures, the architecture of a RBM can be represented as



where $\mathbf{v} = (v_1, v_2, \dots v_m)^T$ consists of m visible units with $v_i \in \{0, 1\}$ and $\mathbf{h} = (h_1, h_2, \dots h_n)^T$ consists of n hidden units with $h_j \in \{0, 1\}$. The weights of the RBM are denoted as W_{ij}. The biases are denoted as b_i for the visible units and c_j for the hidden units. We use λ to denote all model parameters such that $\lambda = \{W, b, c\}$. Within the given python programs, the number of visible units m is stored in the variable `num_visible` and the number of hidden units n is stored in the variable `num_hidden`.

The probability distribution associated with the RBM is given by

$$p_\lambda(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_\lambda} e^{-E_\lambda(\mathbf{v},\mathbf{h})},$$

where

$$E_\lambda(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{m} b_i\, v_i - \sum_{j=1}^{n} c_j\, h_j - \sum_{ij} W_{ij}\, v_i\, h_j,$$

and

$$Z_\lambda = \sum_{\mathbf{v},\mathbf{h}} e^{-E_\lambda(\mathbf{v},\mathbf{h})}.$$

You will train an RBM to learn the distribution corresponding to spin configurations of the two-dimensional classical Ising model on an $L = 4$ lattice at a given temperature. The number of visible units will be equal to the number of spins N such that $m = N = L^2 = 16$. You have been given data corresponding to 11 different temperatures (1.0, 1.254, 1.508, 1.762, 2.016, 2.269, 2.524, 2.778, 3.032, 3.286 and 3.54) in the folder `MC_results`. We know that these configurations are generated (using Monte Carlo simulation) according to the Boltzmann distribution

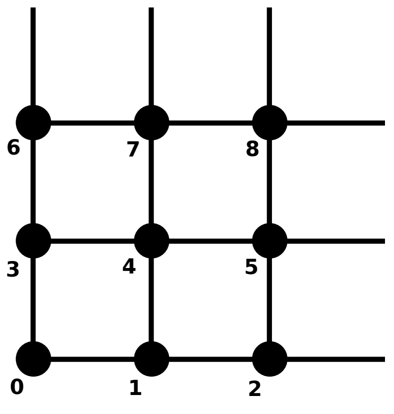
$$q(\mathbf{v}, T) = \frac{1}{Z} e^{-H(\mathbf{v})/T}, \tag{1}$$

where $Z = \sum_{\{\mathbf{v}\}} e^{-H(\mathbf{v})/T}$ is the partition function and $H(\mathbf{v}) = -J \sum_{\langle ij \rangle} v_i v_j$ is the Ising model Hamiltonian with critical temperature $T_\text{c} \approx 2.269J$. We wish to adjust the RBM parameters λ such that the RBM distribution $p_\lambda(\mathbf{v})$ is a good approximation of $q(\mathbf{v}, T)$. This training is done by minimizing the negative log-likelihood (NLL), which is equivalent to minimizing the Kullback-Liebler (KL) divergence. The RBM never has explicit knowledge of $q(\mathbf{v}, T)$ or $H(\mathbf{v})$.

After training the RBM, we can then sample from it new spin configurations. Based on our theoretical knowledge of the Ising model, we can compute observables such as the energy $E(\mathbf{v})$, magnetization $M(\mathbf{v})$, specific heat $C_v(\mathbf{v})$ and susceptibility $\chi(\mathbf{v})$ for each sample. Assuming units where $J = k_\text{B} = 1$, we have

$$\begin{aligned}
E(\mathbf{v}) &= H(\mathbf{v}) = -\sum_{\langle ij \rangle} v_i v_j, \\
M(\mathbf{v}) &= \sum_i v_i, \\
C_v(\mathbf{v}) &= \frac{\langle E^2 \rangle - \langle E \rangle^2}{T^2}, \\
\chi(\mathbf{v}) &= \frac{\langle M^2 \rangle - \langle M \rangle^2}{T}.
\end{aligned}$$

When calculating these observables for a given spin configuration $\mathbf{v} = (v_1, v_2, \ldots, v_N)$, we note that the underlying lattice has periodic boundary conditions and uses a labelling for the sites such that, for example, on a 3×3 lattice:



T	$\langle E\rangle/N$	$\langle M\rangle/N$	C_v/N	χ/N
1.508	-1.9451(9)	0.9856(2)	0.25(2)	0.030(4)
2.269	-1.514(2)	0.849(1)	1.06(2)	0.316(9)
3.286	-0.744(2)	0.532(1)	0.574(6)	0.408(6)

Table 1: Thermodynamic observables measured on the Monte Carlo configurations used to train an RBM at various temperatures. Errors are indicated in parentheses.

We will compare the averages of our RBM-sampled observables to the known values we expect to find in Monte Carlo simulation. We will also explore how the number of hidden units n affects this comparison.

a) Start by examining Figure 4 of Reference [1]. How do the thermodynamic observables generated from the RBM depend on the number of hidden units $n = n_h$? How does the accuracy of each observables vary with the distance from the critical temperature T_c?

b) Let us train our first RBM at the critical temperature T_c with $n = 4$ hidden units. Run the code `tutorial4_train_ising2d.py` with `T = 2.269` and `num_hidden = 4`. This code will train the parameters λ of an RBM based on the given Monte Carlo samples such that $p_\lambda(\mathbf{v})$ is a good approximation of $q(\mathbf{v}, T)$. The resulting parameters will be saved to a file within the folder `RBM_parameters`. Increase the parameter `nsteps` until you are convinced that the NLL has roughly converged.

c) Examine the parameters `learning_rate_start`, `bsize`, `num_gibbs` and `num_samples` and explain how each are used to train the RBM. Experiment with adjusting each of these parameters to see how it affects the training behaviour.

d) Run `tutorial4_sample_ising2d.py` (again with with `T = 2.269` and `num_hidden = 4`) to generate new spin configuration samples. This program will also save the sample configurations to a file within the folder `RBM_samples`.

e) Write code that reads in the sampled configurations generated in part d) and calculates the corresponding observables $\langle E\rangle/N$, $\langle M\rangle/N$, C_v/N and χ/N. Compare with the results for the training samples, which are provided in Table 1. Are your discrepancies similar to those found in Figure 4 of Reference [1]?

   Hint: In order to check if your code is working, you can try calculating the observable quantities for the training configurations in the folder `MC_results` and verifying that you get the same results as in Table 1.

f) Repeat parts b), d) and e) for other values of the number of hidden units n. Once again consider how your results compare with Figure 4 of Reference [1].

g) Repeat parts b), d), e) and f) for temperatures above and below the critical temperature, such as the ones provided in Table 1. How does the difference between your sampled and trained observables depend on temperature? Once again examine how your results compare with Figure 4 of Reference [1].

For further information, check the modern machine learning software for quantum state tomography [2].

References

[1] G. Torlai and R. Melko, Phys. Rev. B **94**, 165134 (2016), `https://arxiv.org/abs/1606.02718`.

[2] QuCumber: wavefunction reconstruction with neural networks, `https://pypi.org/project/qucumber/`.