Project 2, The restricted Boltzmannn machine applied to the quantum many body problem. Deadline May 31

Computational Physics II FYS4411/FYS9411

Department of Physics, University of Oslo, Norway

Jan 11, 2019

Introduction

The idea of representing the wave function with a restricted Boltzmann machine (RBM) was presented recently by G. Carleo and M. Troyer, Science **355**, Issue 6325, pp. 602-606 (2017). They named such a wave function/network a *neural network quantum state* (NQS). In their article they apply it to the quantum mechanical spin lattice systems of the Ising model and Heisenberg model, with encouraging results. To further test the applicability of RBM's to quantum mechanics we will in this project apply it to a system of two interacting electrons (or bosons) confined to move in a harmonic oscillator trap.

Theoretical background and description of the physical system

We consider a system of electrons confined in a pure two-dimensional isotropic harmonic oscillator potential, with an idealized total Hamiltonian given by

$$\hat{H} = \sum_{i=1}^{N}\left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j}\frac{1}{r_{ij}}, \tag{1}$$

where natural units ($\hbar = c = e = m_e = 1$) are used and all energies are in so-called atomic units a.u. We will study systems of many electrons N as functions of the oscillator frequency ω using the above Hamiltonian. The Hamiltonian includes a standard harmonic oscillator part

$$\hat{H}_0 = \sum_{i=1}^{N}\left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right),$$

© 1999-2019, "Computational Physics II FYS4411/FYS9411":"http://www.uio.no/studier/emner/matnat/fys/FYS4411/index-eng.html". Released under CC Attribution-NonCommercial 4.0 license

and the repulsive interaction between two electrons given by

$$\hat{H}_1 = \sum_{i<j} \frac{1}{r_{ij}},$$

with the distance between electrons given by $r_{ij} = |\boldsymbol{r}_1 - \boldsymbol{r}_2|$. We define the modulus of the positions of the electrons (for a given electron i) as $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$.

In this project we will deal only with a system of two electrons in a quantum dot with a frequency of $\hbar\omega = 1$. The reason for this is that we have exact closed form expressions for the ground state energy from Taut's work for selected values of ω, see M. Taut, Phys. Rev. A **48**, 3561 (1993). The energy is given by 3 a.u. (atomic units) when the interaction between the electrons is included. We can however easily extend our system to say interacting bosons as in project 1.

If only the harmonic oscillator part of the Hamiltonian is included, the so-called unperturbed part,

$$\hat{H}_0 = \sum_{i=1}^{N} \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2 \right),$$

the energy is 2 a.u. The wave function for one electron in an oscillator potential in two dimensions is

$$\phi_{n_x,n_y}(x,y) = AH_{n_x}(\sqrt{\omega}x)H_{n_y}(\sqrt{\omega}y)\exp\left(-\omega(x^2+y^2)/2.\right.$$

The functions $H_{n_x}(\sqrt{\omega}x)$ are so-called Hermite polynomials while A is a normalization constant. For the lowest-lying state we have $n_x = n_y = 0$ and an energy $\epsilon_{n_x,n_y} = \omega(n_x + n_y + 1) = \omega$. Convince yourself that the lowest-lying energy for the two-electron system is simply 2ω.

The unperturbed wave function for the ground state of the two-electron system is given by

$$\Phi(\boldsymbol{r}_1, \boldsymbol{r}_2) = C\exp\left(-\omega(r_1^2 + r_2^2)/2\right),$$

with C being a normalization constant and $r_i = \sqrt{r_{i_x}^2 + r_{i_y}^2}$. Note that the vector $\boldsymbol{r}_i$ refers to the x and y position for a given particle. What is the total spin of this wave function? Find arguments for why the ground state should have this specific total spin.

Representing the wave function with a neural network. Our neural network of choice is the restricted Boltzmann machine. It is a two layer net where one is called the layer of visible nodes and the other the layer of hidden nodes. It is called restricted because there are no connections between nodes in the same layer. Meaning there's only a connection between two nodes if one is visible and the other hidden. These type of networks constitute the building blocks of the deep belief networks. The RBM is a generative network, meaning that the idea is for it to learn a *probability distribution*. Thus the network does

not produce an output directly, but a probability distribution from which we can generate an output. In our case this distribution corresponds to the wave function and the output we wish to generate are the positions taken by the particles in our system.

Neural networks are referred to as falling under either supervised or unsupervised learning. Here we are not working with training data, thus it is not supervised. It's rather called reinforcement learning. From the variational principle we know that the NQS wave fucntion represents the ground state once the quantum mechanical energy is minimized. This information is used to optimize the weights and biases of the network.

For more information and practical guides to the RBM, check out the links in the literature section.

When working with the restricted Boltzmann machine we are given the joint probability distribution between the hidden and visible nodes.

Restricted Boltzmann Machine (RBM). The joint probability distribution is defined as

$$F_{rbm}(\mathbf{X}, \mathbf{H}) = \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{X}, \mathbf{H})} \tag{2}$$

where Z is the partition function/normalization constant

$$Z = \int \int \frac{1}{Z} e^{-\frac{1}{T_0} E(\mathbf{x}, \mathbf{h})} d\mathbf{x} d\mathbf{h} \tag{3}$$

It is common to ignore T_0 by setting it to one. Here E is known as the energy of a configuration of the nodes. Do not confuse this with the energy of the quantum mechanical system. Here it is a function which gives the specifics of the relation between the hidden and visible nodes. Different versions of RBMs will implement the energy function differently.

Gaussian-Binary RBM. The original and most common version of an RBM is called "binary-binary", meaning both visible and hidden nodes only take on binary values. In our case we wish to model continuous values (positions), thus the visible nodes should be continuous. We therefore choose an RBM called "Gaussian-binary".

$$E(\mathbf{X}, \mathbf{H}) = \sum_{i}^{M} \frac{(X_i - a_i)^2}{2\sigma_i^2} - \sum_{j}^{N} b_j H_j - \sum_{i,j}^{M,N} \frac{X_i w_{ij} H_j}{\sigma_i^2} \tag{4}$$

If $\sigma_i = \sigma$ then

$$E(\mathbf{X}, \mathbf{H}) = \frac{||\mathbf{X} - \mathbf{a}||^2}{2\sigma^2} - \mathbf{b}^T \mathbf{H} - \frac{\mathbf{X}^T \mathbf{W} \mathbf{H}}{\sigma^2} \tag{5}$$

Here $\mathbf{X}$ are the visible nodes (the position coordinates), $\mathbf{H}$ are the hidden nodes, $\mathbf{a}$ are the visible biases, $\mathbf{b}$ are the hidden biases and $\mathbf{W}$ is a matrix containing the weights characterizing the connection of each visible node to a hidden node.

The Wave Function. To find the marginal probability $F_{rbm}(X)$ we set:

$$F_{rbm}(\mathbf{X}) = \sum_{\mathbf{h}} F_{rbm}(\mathbf{X}, \mathbf{h}) \tag{6}$$

$$= \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{X},\mathbf{h})} \tag{7}$$

This is used to represent the wave function:

$$\Psi(\mathbf{X}) = F_{rbm}(\mathbf{X}) \tag{8}$$

$$= \frac{1}{Z} \sum_{\{h_j\}} e^{-E(\mathbf{X},\mathbf{h})} \tag{9}$$

$$= \frac{1}{Z} \sum_{\{h_j\}} e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2} + \sum_j^N b_j h_j + \sum_{i,j}^{M,N} \frac{X_i w_{ij} h_j}{\sigma^2}} \tag{10}$$

$$= \frac{1}{Z} e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}} \prod_j^N (1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}) \tag{11}$$

$$\tag{12}$$

The Monte Carlo procedure. In many aspects, the procedure of optimizing the NQS wave function will be very similar to the VMC method in project one. However, it requires a heavier emphasis on the minimization process. Whereas in project one you only had one or two parameters to optimize and could even determine them analytically, in this situation the biases and weights quickly add up to a high number of parameters to optimize, and it's hard, if possible at all, to determine them analytically. Thus minimizing the quantum mechanical energy and optimizing the parameters is important from the beginning. Still, the structure of the process is similar. You set up an initial guess of the NQS wave function by giving the weights and biases random, preferably small values. The process then follows the same structure as the VMC method.

Project 2 a): Analytical expressions. Once again you should start by analytically determining the local energy, given by

$$E_L = \frac{1}{\Psi} \hat{\mathbf{H}} \Psi \tag{13}$$

using the NQS Ψ and the Hamiltonian as defined earlier.

If your minimization method of choice is for example stochastic gradient descent (when using this method for neural network training, the step size is often referred to as the learning rate), you will also need the gradient of the local energy with respect to the RBM parameters α ($\mathbf{a}$, $\mathbf{b}$ and $\mathbf{W}$). It is given by

$$G_i = \frac{\partial \langle E_L \rangle}{\partial \alpha_i} = 2(\langle E_L \frac{1}{\Psi} \frac{\partial \Psi}{\partial \alpha_i} \rangle - \langle E_L \rangle \langle \frac{1}{\Psi} \frac{\partial \Psi}{\partial \alpha_i} \rangle) \tag{14}$$

where $\alpha_i = a_1, ..., a_M, b_1, ..., b_N, w_{11}, ..., w_{MN}$.

In addition to E_L then you will also need to find the expression for $\frac{\partial \Psi}{\partial \alpha_i}$.

You see here that the visible nodes (the position coordinates) and the corresponding visible biases are vectors of length M. The hidden nodes and the corresponding hidden biases are vectors of length N. The weight matrix is of size $M \times N$. While the number of hidden nodes (that is, N) is your own choice and should be experimented with, the number of visible nodes (M) should correspond to the number of particles (P) and the number of dimensions (D) in the system, that is $M = P \cdot D$.

Project 2 b): Initial code. Now implement the code. The structure of the code (how you organize your classes) can (probably) imitate what you did in project 1. Use standard Metropolis sampling and ignore interaction. This means excluding the repulsive interaction from your Hamiltonian and local energy calculations. In this case the analytically correct energy of the system is given by $E = \frac{1}{2}P \cdot D$. Optimize the NQS and compute the energy of 1 particle in 1D with 2 hidden units as accurately as you can. Experiment with the learning rate. What precision do you achieve? Eventually you may also experiment with changing the number of hidden units. Document your findings.

Project 2 c): Importance sampling. Add importance sampling to improve your method. Document the results and compare them to the brute force method.

Project 2 d): Statistical analysis. Include a proper statistical analysis by use of the blocking method for your results.

Project 2 e): From Metropolis to Gibbs sampling. For a system such as the one we currently study, where we know that the wave function is positive definite, we may use a sampling method called Gibbs sampling. In this case we represent the wave function as $\Psi(x) = \sqrt(F_{rbm})$ rather than $\Psi(x) = F_{rbm}(x)$.

In this method we sample from the joint probability of $\mathbf{x}$ and $\mathbf{h}$, in the form of a two step sampling process. The samples $\mathbf{x}$ by themselves then model the probability density $|\Psi(\mathbf{x})|^2$ as we wish. The updated samples are generated according to the conditional probabilities $P(X_i|\mathbf{h})$ and $P(H_j|\mathbf{x})$ respectively and accepted with the probability of 1.

$$P(X_i|\mathbf{h}) = \mathcal{N}(X_i; a_i + \mathbf{w}_{i*}\mathbf{h}, \sigma^2) \tag{15}$$

$$P(\mathbf{X}|\mathbf{h}) = \prod_i^M \mathcal{N}(X_i; a_i + \mathbf{w}_{i*}\mathbf{h}, \sigma^2) \tag{16}$$

$$= \mathcal{N}(\mathbf{X}; \mathbf{a} + \mathbf{W}\mathbf{h}, \sigma^2) \tag{17}$$

and

$$P(H_j|\mathbf{x}) = \frac{e^{(b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2})H_j}}{\sum_{h_j} e^{(b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2})h_j}} \tag{18}$$

$$= \frac{e^{(b_j+\frac{\mathbf{v}^T\mathbf{w}_{*j}}{\sigma^2})H_j}}{1+e^{b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2}}} \tag{19}$$

$$P(\mathbf{H}|\mathbf{x}) = \prod_j^N \frac{e^{(b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2})H_j}}{1+e^{b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2}}} \tag{20}$$

$$\tag{21}$$

Meaning

$$P(H_j=1|\mathbf{x}) = \frac{e^{b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2}}}{1+e^{b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2}}} \tag{22}$$

$$= \frac{1}{1+e^{-b_j-\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2}}} \tag{23}$$

$$P(H_j=0|\mathbf{x}) = \frac{1}{1+e^{b_j+\frac{\mathbf{x}^T\mathbf{w}_{*j}}{\sigma^2}}} \tag{24}$$

Recalculate the values needed for the local energy and its gradient with the form of the wave function given by $\Psi(x) = \sqrt{(}F_{rbm})$.

Project 2 f): Gibbs sampling Implement a Gibbs sampling method as an alternative to the Metropolis sampling. Reproduce the energy of the same system as in b), document your results. Again experiment with the learning rate and the number of hidden values. Experiment with changing the σ of the neural netwrok. Comment. Eventually you may increase the number of particles and dimensions. Do the produced energy compare well with the analytical values?

Project 2 g): Interaction. Include the interaction. Remember that for the interacting case we have an analytical answer when we look at two particles in two dimensions (the energy shoud be 3 a.u.). Also, in the interacting case it does not make sense to look at more than two particles since we are currently looking at fermions and have not accounted for the Pauli exclusion principle. As before, experiment with the learning rate and number of hidden values and document how well the network reproduces the analytical value.

Literature.

1. M. Taut, Phys. Rev. A **48**, 3561 - 3566 (1993)

2. G. Carleo and M. Troyer, Science **355**, Issue 6325, pp. 602-606 (2017)
3. A Beginner's Tutorial for Restricted Boltzmann Machines
4. A Practical Guide to Training Restricted Boltzmann Machines

Introduction to numerical projects

Here follows a brief recipe and recommendation on how to write a report for each project.

- Give a short description of the nature of the problem and the eventual numerical methods you have used.
- Describe the algorithm you have used and/or developed. Here you may find it convenient to use pseudocoding. In many cases you can describe the algorithm in the program itself.
- Include the source code of your program. Comment your program properly.
- If possible, try to find analytic solutions, or known limits in order to test your program when developing the code.
- Include your results either in figure form or in a table. Remember to label your results. All tables and figures should have relevant captions and labels on the axes.
- Try to evaluate the reliabilty and numerical stability/precision of your results. If possible, include a qualitative and/or quantitative discussion of the numerical stability, eventual loss of precision etc.
- Try to give an interpretation of you results in your answers to the problems.
- Critique: if possible include your comments and reflections about the exercise, whether you felt you learnt something, ideas for improvements and other thoughts you've made when solving the exercise. We wish to keep this course at the interactive level and your comments can help us improve it.
- Try to establish a practice where you log your work at the computerlab. You may find such a logbook very handy at later stages in your work, especially when you don't properly remember what a previous test version of your program did. Here you could also record the time spent on solving the exercise, various algorithms you may have tested or other topics which you feel worthy of mentioning.

Format for electronic delivery of report and programs

The preferred format for the report is a PDF file. You can also use DOC or postscript formats or as an ipython notebook file. As programming language we prefer that you choose between C/C++, Fortran2008 or Python. The following prescription should be followed when preparing the report:

- Use Devilry to hand in your projects, log in at `http://devilry.ifi.uio.no` with your normal UiO username and password.
- Upload **only** the report file! For the source code file(s) you have developed please provide us with your link to your github domain. The report file should include all of your discussions and a list of the codes you have developed. The full version of the codes should be in your github repository.
- In your github repository, please include a folder which contains selected results. These can be in the form of output from your code for a selected set of runs and input parameters.
- Still in your github make a folder where you place your codes.
- In this and all later projects, you should include tests (for example unit tests) of your code(s).
- Comments from us on your projects, approval or not, corrections to be made etc can be found under your Devilry domain and are only visible to you and the teachers of the course.

Finally, we encourage you to work two and two together. Optimal working groups consist of 2-3 students. You can then hand in a common report.