University of Oslo
FYS4411
Computational physics II: Quantum mechanical systems

Project 2: The Restricted Boltzmann Machine Applied to a Quantum Many Body Problem

Bendik Samseth

June 8, 2018

Abstract

We apply a general reinforcement learning approach to solving quantum mechanical many-body problems. The specific learning model applied is the Restricted Boltzmann Machine (RBM), and we apply it to the system of two electrons trapped in an harmonic oscillator. We investigate the system properties both with and without considering the Coulomb interaction between them. We demonstrate that the RBM is able to learn a wavefunction for the non-interacting system with arbitrary precision. Including the inter-particle repulsion we obtain only approximate results, and are not able to improve accuracy beyond an error of about 10^{-1}a.u. Still, the physical system that the modelled wavefunction describes follows intuitive expectations, as well as following the predictions of the virial theorem, at least to the same degree of accuracy.

All material referenced in this report, including all code, is documented and available at `https://github.com/bsamseth/FYS4411`. All results presented can be readily reproduced using the notebook linked to from the above repository.

Contents

1 Introduction

A physicist faced with a quantum many body system is quickly forced to abandon any hope of solving the Schrödinger equation (or one of its even worse relativistic siblings) directly. Instead he must resort to one of two general workarounds: 1) make simplifying assumptions until the equation is solvable, or 2) make an educated guess for the solution. In either case, the desired quantity which solving the Schrödinger equation would yield is the quantum mechanical wavefunction. If we want to stay away from potentially faulty, or limiting assumptions, we must therefore somehow produce a guess for this wavefunction, without actually solving the equation.

The last project utilized the Variational Monte Carlo (VMC) approach, in which a specific, parametrized form is assumed for the wavefunction, with the subsequent optimization of the parameters.

The main limitation of VMC is that a specific form for the wavefunction is assumed. If this form contains the true optimum that works fine, but if not we are inherently limited in how accurate the final results can be.

Aiming to be overcome this issue, we may observe that the task at hand falls into the general field of function approximation. We wish

to approximate the function that takes as input the configuration of the system, and outputs a probability amplitude for the given state. Function approximation is a task which lends itself to techniques from machine learning (ML).

In this project we will attempt to represent the wavefunction with a specific type of ML-model, the Restricted Boltzmann Machine (RBM). The use of RBMs for such applications was presented recently by Carleo and Troyer [1], with encouraging results. We shall therefore attempt to apply the same techniques to a different type of system, namely that of two interacting quantum particles confined in a harmonic oscillator trap.

The choice of using the RBM as our model is somewhat arbitrary, and other model architectures could also be of interest. The general ML-approach which we employ, regardless of the specifics of the underlying model, is as follows:

1. Define the model to be trained.
2. Formulate the objective, or *loss function*.
3. Using some optimization scheme, optimize the model wrt. to the loss function.

2 Theory

2.1 The System

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

$$\begin{aligned}\hat{H} &= \sum_{i=1}^{P}\left(-\frac{1}{2}\nabla_i^2 + V_{ext}(\boldsymbol{r}_i)\right) + \sum_{i<j} V_{int}(\boldsymbol{r}_i,\boldsymbol{r}_j) \\ &= \sum_{i=1}^{P}\left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right) + \sum_{i<j}\frac{1}{r_{ij}},\end{aligned} \tag{1}$$

where natural units ($\hbar = c = m_e = 1$) are used with energies in atomic units (a.u.), P denotes the number of particles in the system, and ω is the oscillator frequency of the trap. Further, $\boldsymbol{r}_i$ denotes the position vector of particle i, with $r_i \equiv \|\boldsymbol{r}\|$ and $r_{ij} \equiv \|\boldsymbol{r}_i - \boldsymbol{r}_j\|$ defined for notational brevity.

In this project we limit ourselves to the case of $N = 2$ interacting electrons in a trap with a frequency such that $\hbar\omega = 1$. We do this because for this case we have exact, analytical solutions for the ground state energy. With the interaction term included, the ground state energy is $E_0 = 3\,\text{a.u.}$ [6]. This limitation is purely one of convenience, as having exact benchmarks makes for better verification of results. The methods discussed in the project should extend to other systems.

2.2 Simple Non-Interacting Case

If we omit the interacting terms in Equation 1 we have the standard harmonic oscillator Hamiltonian:

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

This Hamiltonian lends it self to analytical solutions, and the stationary single particle states are:

$$\phi_{n_x,n_y}(x,y) = AH_{n_x}(\sqrt{\omega}x)H_{n_y}(\sqrt{\omega}y)e^{-\frac{\omega}{2}(x^2+y^2)}, \tag{3}$$

for quantum numbers $n_x, n_y = 0, 1, \ldots$, and the Hermite polynomials H_n. The ground state, $n_x = n_y = 0$ is simply

$$\phi_{00}(x,y) = \sqrt{\frac{\omega}{\pi}}e^{-\frac{\omega}{2}(x^2+y^2)}. \tag{4}$$

Using this wavefunction we can calculate the ground state energy[1],

$$\epsilon_{00} = \frac{\langle\phi_{00}|\hat{H}_0|\phi_{00}\rangle}{\langle\phi_{00}|\phi_{00}\rangle} = \omega = 1\,\text{a.u.} \tag{5}$$

The ground state wavefunction for the (unperturbed) two-electron case is simply the product of the one-electron wavefunctions,

$$\begin{aligned}\Phi(\boldsymbol{r}_1,\boldsymbol{r}_2) &= \phi_{00}(\boldsymbol{r}_1)\phi_{00}(\boldsymbol{r}_2) \\ &= \frac{\omega}{\pi}e^{-\frac{\omega}{2}(r_1^2+r_2^2)}.\end{aligned} \tag{6}$$

The ground state energy can once again be evaluated analytically[1] and yields

$$E_0 = \frac{\langle\Phi|\hat{H}_0|\Phi\rangle}{\langle\Phi|\Phi\rangle} = 2\omega = 2\,\text{a.u.} \tag{7}$$

This result is not surprising, as adding one more particle, without any interactions, should simply double the energy. Another way to look at

[1] See `projects/python/Sympy.ipynb` in the Github repository for an explicit calculation of the ground state energy

it is that the simple harmonic oscillator solution gives $\omega/2$ per degree of freedom, so adding another two yields and extra ω.

When the two particles are electrons, we may say something about their total spin. As electrons are fermions, their total wavefunction must be anti-symmetric upon interchanging the labels 1 and 2. Equation 6 is obviously symmetric, and so the spin-wavefunction must necessarily be anti-symmetric. For the combination of two spin-1/2 particles, there is only one candidate, namely the spin-0 singlet:

$$\chi_0 = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle). \tag{8}$$

2.3 Considerations from the Virial Theorem

The Virial Theorem gives a general relation for the time-averaged kinetic energy $\langle K\rangle$ and the corresponding potential energy $\langle V_{pot}\rangle$ of a stable system of N particles. In general the theorem is stated as:

$$\langle K\rangle = -\frac{1}{2}\sum_{k=1}^{N}\langle \boldsymbol{F}_k \cdot \boldsymbol{r}_k\rangle\,, \tag{9}$$

where $\boldsymbol{F}_k$ denotes the combined forces acting on particle k, located at position $\boldsymbol{r}_k$. For a radial potential on the form $V(r) = ar^n$, such that the potential between any two particles in the system depends on some power of the inter-particle distance, the theorem takes the following form:

$$\langle K\rangle = \frac{n}{2}\langle V_{TOT}\rangle \tag{10}$$

where V_{TOT} denotes the sum of the potential energy $V(r)$ over all pairs of particles.

Although the harmonic oscillator potential does not depend on the *inter-particle* distance, but rather on each particles position, it works out to the same relation in our case. Computing the full relation for our Hamiltonian for two electrons in two dimensions, it even works out so that we can use the same relation on the HO potential and the Coulomb potential separately, and add the result.[2] This means that the Virial Theorem predicts the following [3]:

$$\langle K\rangle = \langle V_{ext}\rangle - \frac{1}{2}\langle V_{int}\rangle\,. \tag{11}$$

Note that this implies that we should consider the *total* kinetic energy, and the *total* external and internal potential energies.

3 Representing the Wavefunction with an RBM

Our machine learning model of choice is the Restricted Boltzmann Machine. It consists of two densely interconnected layers, the *visible* layer and the *hidden* layer. It is called restricted because we only allow connections between nodes in different layers - no visible-to-visible or hidden-to-hidden connections are included.

The RBM is a generative model, meaning it learns a probability distribution for its inputs. This means that a trained RBM can produce outputs which when viewed as a distribution, would be similar to the distribution of the inputs. For our case, this means learning the probability distribution for the space configurations of the electrons in our system. We may interpret this distribution as the wavefunction, as the wavefunction serves this same purpose.

In this case we do not have a training set of positions for any of the systems under consideration. This means that the most desired training regime, *supervised training*, is not relevant for our problem. Instead we shall use *reinforcement training*, were updates of the model are based on the variational principle: The true ground state wavefunction is the wavefunction for which the lowest ground state energy is obtained. We can treat the energy that a certain wavefunction (model configuration) produces as the penalty, and let the model adapt as to minimize the penalty it receives.

This approach should work in general, but there is one potential issue with this. In order to evaluate how good (or bad) a proposed model work, we need to evaluate the ground state energy. In order to do this we, as in project 1, use the proposed wavefunction to sample positions to be used in the evaluation of the energy. This kind of almost circular reasoning will only work

[2] We can (and have) work this out via Equation 9 directly using the corresponding forces for the harmonic oscillator and Coulomb potential. A write-up of the derivation is not given though, as this is a well known result.

well if the model is somewhat decent at modeling the true wavefunction. If the proposed model is very far off, the resulting energy evaluations will be very unstable, and we could end up with large errors in the computed gradients. This could in turn lead to an even worse model, and this could go on until something crashes. This issue is similar to the *exploding gradient* problem which often occurs in neural network models, if not addressed properly. We will not address this directly in this project, and instead lean on the fact that we probably have rather decent starting guesses, as we have exact solutions to the ideal case. It is, however, a potential issue if this approach should be applied to systems for which we have a very poor sense of the underlying wavefunction form.

3.1 The Math

In the following, $\boldsymbol{x}$ denotes the values of the visible layer (our position coordinates), and $\boldsymbol{h}$ denotes the values of the hidden layer.

The joint probability distribution over $\boldsymbol{x}$ and $\boldsymbol{h}$ is:

$$F_{RBM}(\boldsymbol{x}, \boldsymbol{h}) = \frac{1}{Z} e^{-\frac{1}{T_0} E(\boldsymbol{x}, \boldsymbol{h})}, \tag{12}$$

where Z is the partition function ensuring that F_{RBM} is normalized. In accordance with common norm, we set $T_0 = 1$. The function $E(\boldsymbol{x}, \boldsymbol{h})$ is known as the energy of a configuration of the nodes, not to be confused with the energy of the quantum system. It encodes the probability of a given configuration - high energy configurations are less likely.

3.1.1 Gaussian-Binary RBM

The type of energy function we will use is called Gaussian-Binary, meaning our inputs (the position coordinates) are Gaussian (continuous and normal), while the hidden nodes take binary values $h_j \in \{0, 1\}$. It looks as follows:

$$E(\boldsymbol{x}, \boldsymbol{h}) = \frac{\|\boldsymbol{x} - \boldsymbol{a}\|}{2\sigma^2} - \boldsymbol{b}^T \boldsymbol{h} - \frac{\boldsymbol{x}^T \boldsymbol{w} \boldsymbol{h}}{\sigma^2}, \tag{13}$$

where $\boldsymbol{a} \in \mathbf{R}^M, \boldsymbol{b} \in \mathbf{R}^N$ are bias vectors for the visible and hidden layers respectively, and $\boldsymbol{w} \in \mathbf{R}^{M \times N}$ is a matrix encoding the weights for every connection between the two layers. In our case, $M = PD$ for P particles in D dimensions, while N will be chosen freely by us.

3.1.2 The Wavefunction

The wavefunction shall be the probability amplitude of any given system configuration. We obtain this by tracing out the hidden values:

$$\begin{aligned}
\Psi(\boldsymbol{X}) = F_{RBM}(\boldsymbol{x}) &= \sum_{\boldsymbol{h}} F_{RBM}(\boldsymbol{x}, \boldsymbol{h}) \\
&= \frac{1}{Z} e^{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}} \prod_j^N \left(1 + e^{b_j + \sum_i^M \frac{X_i w_{ij}}{\sigma^2}}\right) \\
&\equiv \frac{1}{Z} e^{-\sum_i^M u_i} \prod_j^N (1 + e^{v_j})
\end{aligned} \tag{14}$$

where u_i and v_j are defined for convenience. We will treat a_i, b_i and w_{ij} as tunable parameters, while σ^2 will be taken as some constant, specifically the value we would have for the ideal wavefunction for the non-interacting case. Looking at Equation 6 in comparison to Equation 14 we see that $\sigma^2 = \omega^{-1}$ would yield the ideal non-interacting wavefunction when all parameters $\boldsymbol{\alpha} = \mathbf{0}$.

4 Learning the Wavefunction

4.1 The Cost Function - Energy

The cost function we use to train the network shall be the expectation value of the Hamiltonian, under the wavefunction modeled by the network. Minimizing the energy will yield the best possible wavefunction obtainable within the model. The energy is expressed as

$$Q = E[H] = \langle H \rangle = \frac{\int \mathrm{d}\boldsymbol{x}\, \Psi^* \hat{H} \Psi}{\int \Psi^* \Psi}. \tag{15}$$

where $\boldsymbol{x}$ is the vector containing all the positions of the particles in the system, $\boldsymbol{x} = [x_1, y_1, \ldots, x_n, y_n]$. In order to numerically evaluate this integral we first manipulate it a bit. The probability density at position $\boldsymbol{x}$, under the trial wave function, is

$$P(\boldsymbol{x}) = \frac{|\Psi|^2}{\int \mathrm{d}\boldsymbol{x}\, |\Psi|^2}. \tag{16}$$

We finally define a new quantity, called **the local energy**:

$$E_L(\boldsymbol{x}) = \frac{1}{\Psi} \hat{H} \Psi \tag{17}$$

Combining these two definitions we can now rewrite $\langle H \rangle$ as follows:

$$\begin{aligned}\langle H \rangle &= \int \mathrm{d}\boldsymbol{x}\, P(\boldsymbol{x}) E_L(\boldsymbol{x}) \\ &\approx \frac{1}{n}\sum_{i=1}^{n} E_L(\boldsymbol{x}_i),\end{aligned} \tag{18}$$

where $\boldsymbol{x}_i$ are n randomly drawn positions from the PDF $P(\boldsymbol{x})$. We have therefore that estimating the average value of E_L yields an approximated value for $\langle H \rangle$.

4.2 Optimization

In order to train the model to minimize $\left\langle \hat{H} \right\rangle$ we need to know how to adjust the parameters $\boldsymbol{\alpha} = (a_1, \ldots, a_M, b_1, \ldots, b_N, w_{11}, \ldots, w_{MN})$. We do this using some optimization algorithm, which will in turn be based on the partial derivatives of the expectation of the local energy [2]:

$$G_i = \frac{\partial \langle E_L \rangle}{\partial \alpha_i} = 2\left(\left\langle E_L \frac{1}{\Psi}\frac{\partial \Psi}{\partial \alpha_i} \right\rangle - \langle E_L \rangle \left\langle \frac{1}{\Psi}\frac{\partial \Psi}{\partial \alpha_i} \right\rangle \right) \tag{19}$$

These partial derivatives are trivial to compute analytically, and come out as follows:

$$\frac{1}{\Psi}\frac{\partial \Psi}{\partial a_k} = \frac{x_k - a_k}{\sigma^2} \tag{20}$$

$$\frac{1}{\Psi}\frac{\partial \Psi}{\partial b_k} = \frac{1}{1 + e^{-v_k}} \tag{21}$$

$$\frac{1}{\Psi}\frac{\partial \Psi}{\partial w_{kl}} = \frac{1}{1 + e^{-v_l}}\frac{x_k}{\sigma^2} \tag{22}$$

The expression for the local energy it self is a bit less trivial to work out, but still doable. And as we shall need to compute the local energy often, it will be useful to do the differentiation analytically, so as to speed up the computation compared to doing the second derivative in $\hat{H}$ numerically. The details are laid out in Appendix A.

The final expression we shall use for the local energy is:

$$E_L = \sum_{i=1}^{M}\frac{1}{2}x_i^2 + \sum_{i<j}^{P}\frac{1}{r_{ij}} - \frac{1}{2}\sum_{k=1}^{M}\frac{1}{\Psi}\frac{\partial^2}{\partial x_k^2}\Psi, \tag{23}$$

where,

$$\begin{aligned}\frac{1}{\Psi}\frac{\partial^2}{\partial X_k^2}\Psi = &-\frac{1}{\sigma^2} + \sum_{j}^{N}\frac{w_{kj}^2}{\sigma^4}\frac{e^{-v_j}}{(1+e^{-v_j})^2} \\ &+ \frac{1}{\sigma^4}\left(a_k - x_k + \sum_{j}^{N}\frac{w_{kj}}{1+e^{-v_j}}\right)^2\end{aligned} \tag{24}$$

The complexity of E_L is $\mathcal{O}(M + P^2 + MNM) = \mathcal{O}(P^2 + M^2 N)$. This can be optimized slightly by computing all the v_j terms at once (as opposed to on demand within the sums), which brings this down to $\mathcal{O}(P^2 + MN)$. It still scales quadratically with additional particles (and dimensions), and linearly with the number of hidden nodes.

Comparing this with the results from project 1, where the complexity of a local energy evaluation was $\mathcal{O}(P^3)$ when a similar interaction was considered, we see a significant difference. *Assuming* we can obtain good results using an RBM where N is comparable to P in size, this new approach has much better time-complexity and therefore looks much more promising for use with large systems. We will not pursue this large P regime much further in this project, but this is something to note, if the RBM prove to be fruitful.

4.2.1 Optimization Schemes

Now equipped with a gradient, the general approach of optimization goes as follows:

Algorithm 1 General Optimization Routine

Require: Cost function Q
Require: Update function O
Ensure: $\boldsymbol{\alpha}$ minimizes O
  Initialize $\boldsymbol{\alpha}$ with random values.
  while $\boldsymbol{\alpha}$ not converged **do**
    $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + O(\nabla Q)$
  end while

What remains to plugged into Algorithm 1 is some update function which should return a suitable update to apply to our parameters. There are a myriad of versions for how to do this, with the most simple being *Stochastic Gradient Decent* (SGD). The basic algorithm for SGD is shown in Algorithm 2.

Algorithm 2 The Stochastic Gradient Decent Algorithm

Require: Cost function gradient ∇Q
Require: Learning rate $\eta > 0$
  return $-\eta \nabla Q$

Many, many additions can be made to SGD in order to improve convergence speed and stability. Examples of such modifications are using a decaying learning rate η, including previous updates in the calculation of new ones (momentum), and many other variations. We will limit this project to implementing two update functions, the aforementioned SGD, and ADAM [4]. The ADAM optimizer is widely used in ML and consistently performs on par or better than any other scheme currently known. The pseudo-code for the algorithm is given in the original paper [4], and the implementation can be found on the Github repository under the filename `src/optimizer.cpp`

4.3 Regularization

We may wish to impose some regularization in the cost function as well. Often times this can help guide the optimization out of local minima, as well as shaping ill-formed cost functions. We can modify the cost function as follows:

$$Q = \langle H \rangle + \gamma \|\boldsymbol{\alpha}\|_d^d, \tag{25}$$

where $\gamma > 0$ is a hyper-parameter controlling the amount of regularization, and $\|\cdot\|_d^d$ is the L_d norm. Two of the most widely used types of regularization are Ridge and Lasso, which use $d = 2$ and $d = 1$, respectively. In order to keep the gradient simple, we will only implement Ridge loss. We obtain then a slightly modified form for G_i:

$$G_i = \frac{\partial \langle E_L \rangle}{\partial \alpha_i} + 2\gamma\alpha_i, \tag{26}$$

5 Sampling Algorithms

We will continue to use the Metropolis and Metropolis-Hastings algorithms presented in project 1 for this project also. For the sake of conciseness, we shall not repeat the derivations or motivations for these algorithms here, as they are easily adapted to work with this new model. However, the RBM model enables us to employ another sampling technique, *Gibbs sampling*.

5.1 Gibbs Sampling

For the particular case of systems like the ones we are currently considering, we know that the true wavefunction is positive definite. This allows us to use (yet) another sampling algorithm: Gibbs sampling. In order for us to make efficient use of Gibbs sampling, we have to change our wavefunction representation a bit,

$$\Psi_G(\boldsymbol{x}) = \sqrt{\Psi(\boldsymbol{x})} = \sqrt{F_{RBM}(\boldsymbol{x})}, \tag{27}$$

such that F_{RBM} models the probability $|\Psi_G|^2$ directly, instead of the probability amplitude Ψ_G. This is done in order to have efficient, direct expressions to sample from later (more in next section). This transformation is only valid in general when the wavefunction is positive definite, while the original definition Ψ remains valid in general.

5.2 The Algorithm

Gibbs sampling comes up whenever we are dealing with a *joint* PDF from which we cannot easily (or at all) sample values. In our case we are dealing with the PDF from Equation 12, which does not have a standard, direct sampling strategy. In general, given a PDF $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$, Gibbs relies on the conditionals $P(X_i = x_i | X_j = x_j \forall j \neq i)$. *If* these conditionals are easy to sample from (i.e. can be expressed as one of the known distributions for which direct sampling is possible), then we may obtain approximate samples from the joint PDF by iteratively sampling individual values from the conditionals. Going back to Equation 12, the algorithm becomes:

Algorithm 3 Gibbs sampling of $\boldsymbol{x}$ from $F_{RBM}(\boldsymbol{x}, \boldsymbol{h})$

Require: n number of samples to yield.
Ensure: $\boldsymbol{x}, \boldsymbol{h} \overset{\text{d}}{=} P(\boldsymbol{x}, \boldsymbol{h})$.
  Initialize $\boldsymbol{x}$ randomly (e.g. normally).
  for n iterations **do**
    $\boldsymbol{h} \overset{\text{d}}{\leftarrow} P(\boldsymbol{h}|\boldsymbol{x})$
    $\boldsymbol{x} \overset{\text{d}}{\leftarrow} P(\boldsymbol{x}|\boldsymbol{h})$
    Save $\boldsymbol{x}$ as the i-th sample.
  end for

From Algorithm 3 is is clear that we need to be able to sample from both $P(\boldsymbol{x}|\boldsymbol{h})$ and $P(\boldsymbol{h}|\boldsymbol{x})$. These are simply stated here without

further derivation, and follow from the form of the RBM:

$$P(\boldsymbol{x}|\boldsymbol{h}) = \mathcal{N}\Big(\boldsymbol{x}; \boldsymbol{a} + \boldsymbol{w}\boldsymbol{h}, \sigma^2\Big), \tag{28}$$

$$P(\boldsymbol{h}|\boldsymbol{x}) = \prod_j^N \frac{e^{v_j h_j}}{1 + e^{v_j}} \tag{29}$$

The first conditional is just the normal distribution, which there are standard tools to sample from. The second seems a little harder, until we look at it for one single h_i:

$$P(h_j|\boldsymbol{x}) = \frac{e^{v_j h_j}}{1 + e^{v_j}} \Rightarrow P(h_j = 1|\boldsymbol{x}) = \frac{1}{1 + e^{-v_j}}. \tag{30}$$

So to sample h_i we set it to 1 with probability $p = (1 + \exp(-v_j))^{-1}$, and 0 otherwise.

5.3 New Expressions for Ψ_G

Using Ψ_G as our wavefunction will slightly change all the expressions for the derivatives. Luckily, not much work is needed to redo them all using the following observations:

$$\ln \Psi_G = \ln \sqrt{\Psi} = \frac{1}{2} \ln \Psi, \tag{31}$$

and

$$\frac{1}{\Psi}\frac{\partial \Psi}{\partial x_k} = \frac{\partial \ln \Psi}{\partial \Psi}\frac{\partial \Psi}{\partial x_k} = \frac{\partial \ln \Psi}{\partial x_k} \tag{32}$$

$$\Rightarrow \frac{1}{\Psi_G}\frac{\partial \Psi_G}{\partial x_k} = \frac{1}{2}\frac{1}{\Psi}\frac{\partial \Psi}{\partial x_k}. \tag{33}$$

Propagating this factor of a half through all the expressions quite easily yields:

$$\frac{1}{\Psi_G}\frac{\partial \Psi_G}{\partial a_k} = \frac{x_k - a_k}{2\sigma^2} \tag{34}$$

$$\frac{1}{\Psi_G}\frac{\partial \Psi_G}{\partial b_k} = \frac{1}{2(1 + e^{-v_k})} \tag{35}$$

$$\frac{1}{\Psi_G}\frac{\partial \Psi_G}{\partial w_{kl}} = \frac{1}{1 + e^{-v_l}}\frac{x_k}{2\sigma^2} \tag{36}$$

$$\frac{1}{\Psi_G}\frac{\partial^2}{\partial X_k^2}\Psi_G = -\frac{1}{2\sigma^2} + \sum_j^N \frac{w_{kj}^2}{2\sigma^4}\frac{e^{-v_j}}{(1 + e^{-v_j})^2} + \frac{1}{4\sigma^4}\left(a_k - x_k + \sum_j^N \frac{w_{kj}}{1 + e^{-v_j}}\right)^2 \tag{37}$$

6 Results

6.1 One Particle in One Dimension

As a first test of the RBM, we apply it to the very simple simple case of a single particle in one dimension. We will use the standard metropolis sampling algorithm to begin with. Figure 1 shows the learned wavefunction after $2 \cdot 10^4$ iterations, using a learning rate of 0.9, and no regularization. The energy produced by this wavefunction comes out as $0.49999985 \pm 2 \cdot 10^{-6}$a.u., compared to the exact value of 0.5 a.u.. The error given here is the standard error of the mean local energy, as calculated with the Blocking method discussed in project 1, using 2^{20} samples. If we allow for more training, the energy is lowered more, although the gains diminish as E_L approaches its ideal value. In any case, this precision is more than acceptable.

Applying regularization in this case, for almost any reasonable choice for γ (e.g. 0.1), the RBM converges *very* quickly to the exact wavefunction, by setting all parameters as small as possible. The effect of regularization is artificially magnified in this situation, as it so happens that the minimizing choice of parameters coincide with the minimizing choice for the regularization term, i.e the trivial $\boldsymbol{\alpha} = \mathbf{0}$ solution. We cannot expect this to work as well for less trivial tests, but it serves as a nice sanity check for the developed codes.

Figure 2 shows the error we make as a function of training iterations for the three different samplers discussed. We can observe a couple of interesting points from this plot. First, brute force Metropolis sampling has, as expected, a significantly higher variance in energy compared with Metropolis-Hastings. Interestingly that does not appear to affect the learning much, as both algorithms follow roughly the same curve. However, we would prefer to use Metropolis-Hastings here due to the stability it provides.

Gibbs sampling affects the training in a noticeably different way. We see it starting out with slower convergence compared the others, but as we continue to train, the Gibbs sampling curve holds a steeper decline, and surpassing the

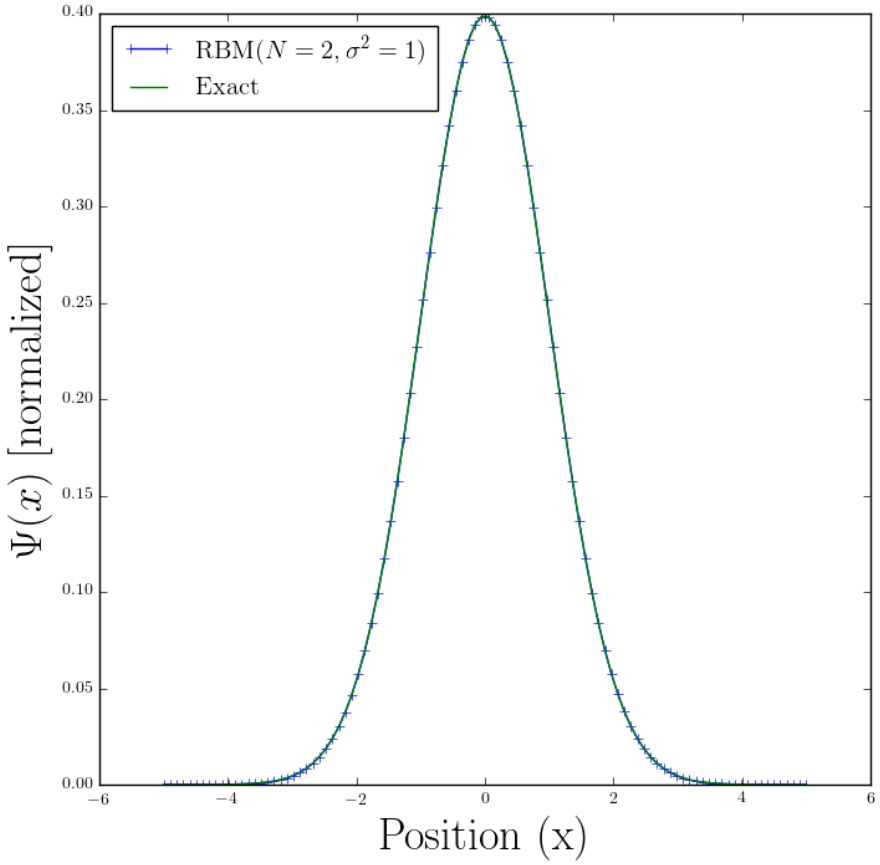

Figure 1: Ideal and learned wavefunction for one particle in one dimension.

results obtained by the Metropolis variants. The effect is small though, keeping in mind the logarithmic y-axis which acts to amplify such differences. In fact, only the slightly slower start is always reproduced, as we sometimes get stuck in a local minimum before Gibbs can catch up fully.

One thing to note is that the constant value assumed for σ^2 here was $\sigma^2 = 1$ for the Metropolis variants, and $\sigma^2 = 1/2$ for Gibbs. Both of these were chosen so that setting $\boldsymbol{\alpha} = \mathbf{0}$ would correspond to the ideal wavefunction. Using $\sigma^2 = 1$ with Gibbs makes it significantly worse. While this could potentially be helped by adding more parameters to the model (i.e. increasing N), this illustrates the potential issue with this general approach, namely the dependence on somewhat decent starting points.

Figure 3 shows a similar comparison but for various optimizers. We can see that choosing a smaller learning rate for SGD results in slower convergence. Ideally, given sufficient training iterations we expect the smaller η to result in slightly better accuracy, as it allows for more fine tuned adjustments. However, this is only guaranteed for ideally convex cost functions. Here we observe the small-η optimizer getting stuck at some local minimum, unable to get out due to the small learning rate. This goes to show the importance of choosing a suitable learning rate when using vanilla SGD. Although not shown here, choosing to large values can result in catastrophic results.

Better than both other methods is the Adam optmizer, here used with the default parameters given by the original authors [4]. Although the large-η SGD variant is initially somewhat faster, Adam ends up close to an order of magnitude better. Interestingly we appear to reach a point were further training reduces the energy

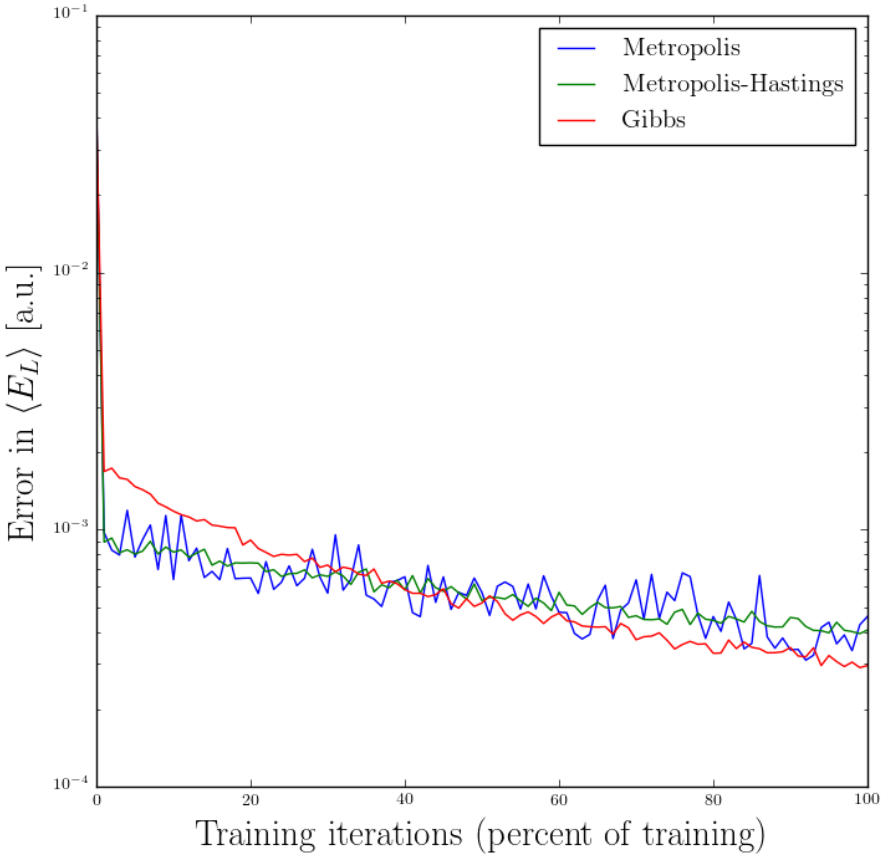

Figure 2: Learning progression, comparing the three different sampling algorithms. We see a slight difference, especially in the jaggedness of the curve, indicating a difference in variance.

slightly, while the variance remains roughly the same (causing an apparent increase in variance, due to the log-axis). In addition to the improved results, the main appeal of Adam is how well it performs without hyper-parameter tuning. Due to its adaptive nature, it performs well with the defaults in all scenarios attempted.

Figure 4 shows yet again a similar plot of training progression, this time for various values for N. Here we see that increasing N did not help, and only increased the run-time. Also, depending on the random starting point of the parameters, the higher N RBMs sometimes get stuck at local minima, and convergence stops. This results makes sense, considering that the ideal case has an optimum for $\boldsymbol{\alpha} = \mathbf{0}$. Adding more parameters just means it gets harder for the optimizer to locate the minimum, and increases the density of local, non-optimal minima. Increasing N might be more helpful when we consider the non-trivial interacting case, which does not have this trivial solution.

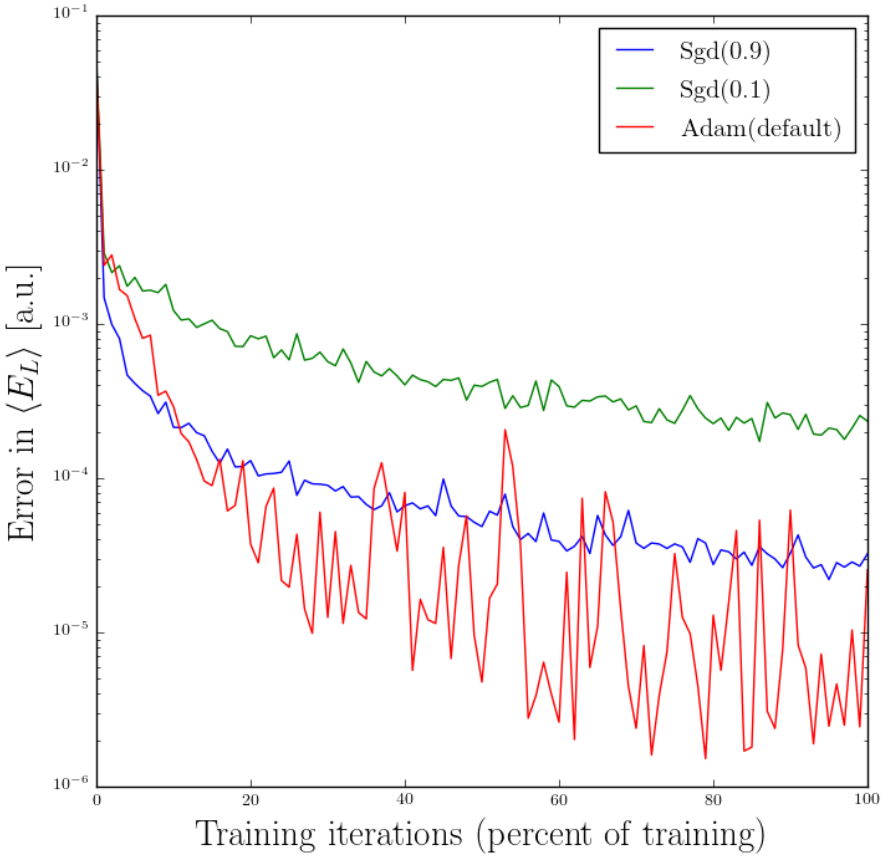

Figure 3: Learning progression, comparing different optimization algorithms. We observe Adam outperforming SGD, with the small learning rate converging very slowly. Training was performed using $4 \cdot 10^4$ iterations with 100 samples per iterations.

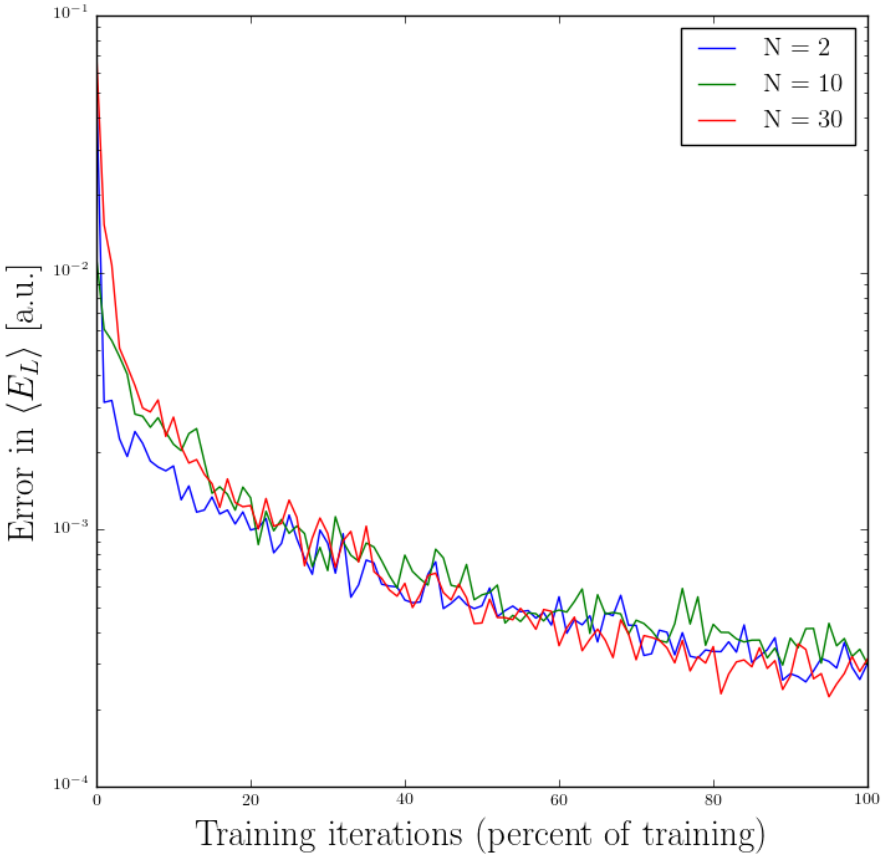

Figure 4: Learning progression, comparing different number of hidden neurons. Training was performed using $4 \cdot 10^4$ iterations with 100 samples per iterations, using Metropolis-Hastings sampling and Adam optimizer.

6.2 Two Particles in Two Dimensions

We now scale up to two particles in two dimensions, but not including the interacting for now. We are once again able to obtain very good results here. Using Metropolis-Hastings sampling, Adam optimizer, $N = 2$ and $\sigma^2 = 1$, running for $4 \cdot 10^4$ iterations with 100 samples per iteration, we obtain the following results (all energies in atomic units):

$\langle E_L \rangle$	$SE_{\langle E_L \rangle}$	$\langle r_{12} \rangle$
$2 + 8 \cdot 10^{-8}$	$4 \cdot 10^{-8}$	1.25247
$\langle K \rangle$	$\langle V_{pot} \rangle$	$\langle K \rangle_{\text{virial}}$
0.9998	1.0002	1.0002

Once again, the energies produced are very accurate. From the mean potential and kinetic energy we see both coming out to about half of the total energy each, as we would expect from Equation 11. Our results fit the theory well.

Figure 5 shows yet another indication that the learned wavefunction is a very close approximation to the ideal, this time through the one-body density. The expected, ideal curve drawn was derived in the report for project 1, and the technique used to calculate it is also as described there.

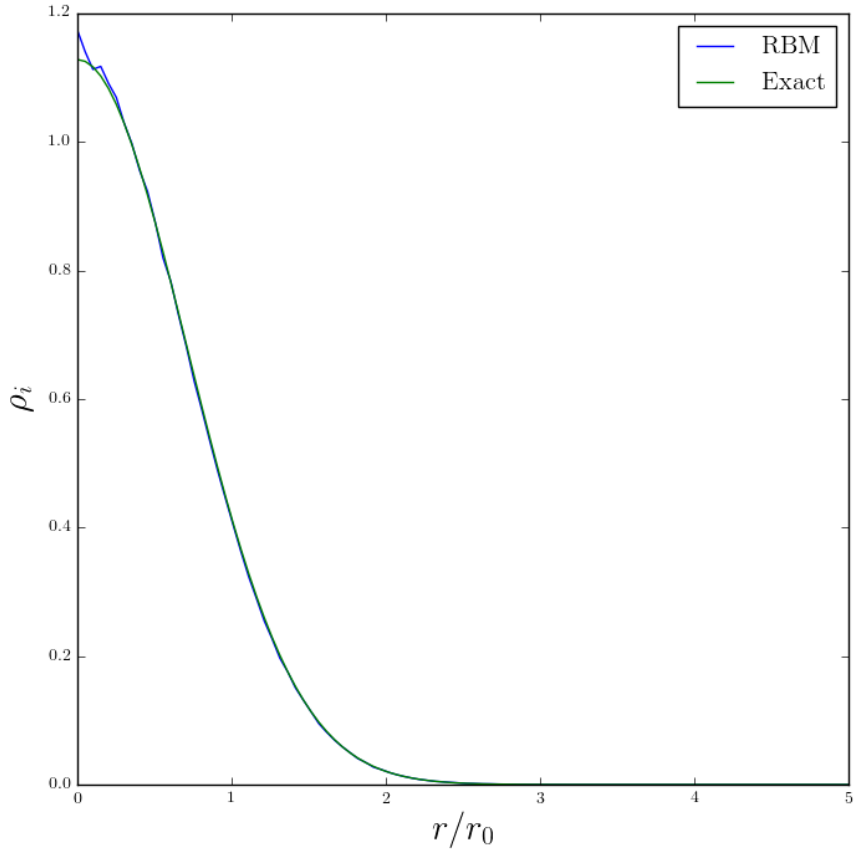

Figure 5: One-body density for two non-interacting particles in 2D. The numerical approximation closely follows the exact form.

6.3 With Interaction

We turn now to the most interesting, and challenging part - including the Coulomb interaction between the two particles. As mentioned before, we expect the ground state energy to be 3 a.u. in this case.

One of the most attractive features of using the RBM as our wavefunction model is that we need not change anything about the model in

order to learn the new wavefunction. The only thing that is changed is the Hamiltonian. As long as the model has sufficient flexibility (i.e. number of hidden units), the hope is that it is able to adjust and optimize its response.

While experiments have been made with all sampling strategies discussed, the results given here are using the Metropolis-Hastings algorithm. Little difference has been seen between the three candidates, and Metropolis-Hastings has been deemed most appropriate only due to its slightly less variant results.

The first thing we note when training various configurations is the issue mentioned previously, namely that training is far less stable due to us having worse initialing guesses for the parameters to use, as well as the increased complexity of the Hamiltonian it self. We find that using 100 sample points as before to compute the gradient at each iteration is far to little for stable learning, and the resulting learning plots in Figure 6 are quite messy. Inspecting the plot with a positive attitude we see some notion of improvement from start to finish, despite the variance being far to high for us to make any quantitative observations.

Ideally we fix this issue by increasing the number of samples per iteration. For instance, using 10^6 samples more or less removes the issue of unstable results, but at the expense of increasing training time by four orders of magnitude. The mitigate this, we can cut down on the number of iterations. Considering the experiences from the ideal case, most of the improvements happen early, with diminishing returns on further training. However, a decent amount of iterations is still needed, somewhat dependent on the optimization scheme used. Some balance must be struck here, and experiments tend toward a preference for many training iterations.

We present first some results for different number of hidden units. The following table was obtained using 10^4 iterations of 10^3 samples, and the means and standard errors were computed on the resulting models using 2^{20} samples. As before, the standard error is as calculated using the blocking method.

N	$\langle E_L \rangle$	$SE_{\langle E_L \rangle}$
1	3.14811	0.00287
2	3.07651	0.00175
4	3.07969	0.00187
8	3.08267	0.00205
12	3.08562	0.00206

Several interesting things can be observed. Firstly, using only a single hidden unit seems to be inadequate to describe the system properly, as this model has converged quite far of, compared to the rest. Increasing the number to $N \geq 1$ we see a better result, although far from perfect. Interestingly, using $N > 2$ does not yield more accurate results, indicating that $N = 2$ is sufficient. As a consequence we see a slight trend of decreasing accuracy when N is increased. This fits earlier results, were having N excessively high not only increases the run-time, but also makes training converge slightly slower and/or increases chances of local minima. As such, it seems there is no point in using $N > 2$.[3]

As such, a model using $N = 2$ was trained using $2 \cdot 10^4$ training iterations of 10^3 sample points using the Adam optimizer with default parameters. The obtained energy estimate obtained saw no improvement over the above listed results, and so we conclude that this accuracy as about as good as we can get it.

Using this model, the mean distance between the particles comes out as 1.65 for this oscillator strength, which is notably larger than what we had in the ideal case. This follows our intuition, as adding a repelling force should act to increase the distance between particles. Figure 7 shows the one-body density that the learned wavefunction produces, with the exact ideal case curve drawn for reference. It shows the same effect graphically, as we see that the density is somewhat more spread out as a result of the particles hindering each other from staying near the origin.

We have also experimented with the regularization parameter. Despite the ideal results seen by using regularization in the non-interacting system, applying it to this case had very poor results. It seems that the problem at hand does not suffer from the issue that regularization aims to fix, so rather we just end up hampering the learning without any benefit. All results pre-

[3] It could be, given only these results, that training for longer would increase the accuracy for high N, and that the results shown are simply due to using too few iterations. This has, however, been seen to not be the case. Increasing the number of training iterations on high N models does not yield improved results.

sented are therefore obtained without regularization added, and no further discussion of regularization is included in this report. As a footnote to future work, regularization should not be abandoned completely, as other ML-models with different architectures and flexibilities might still benefit.

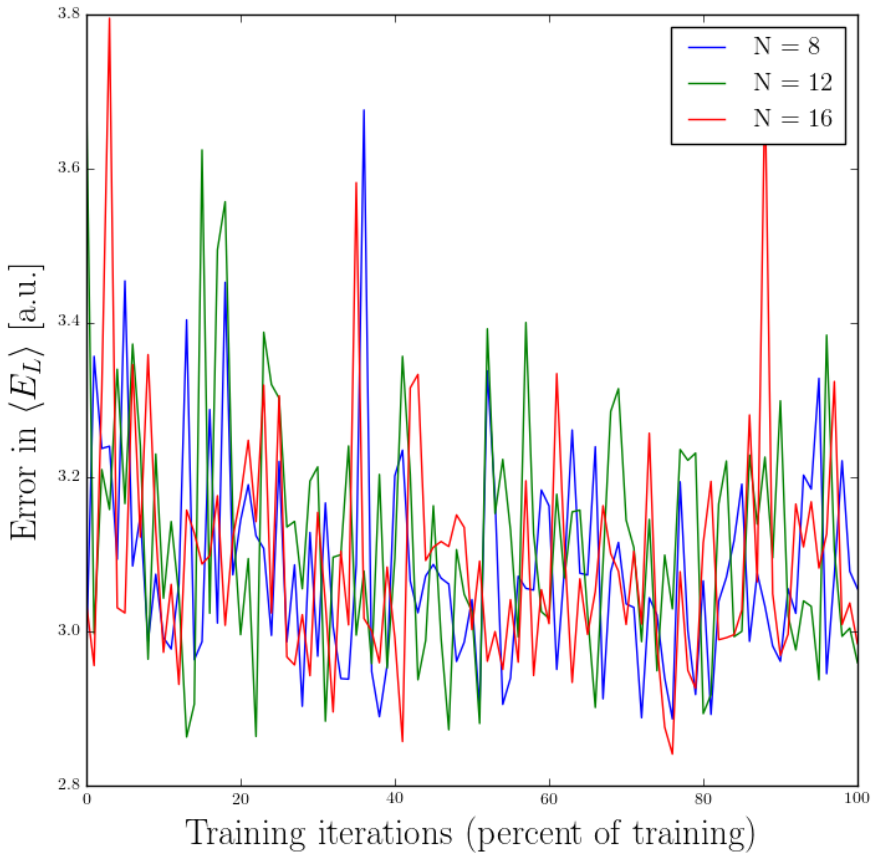

Figure 6: Learning progression for some different RBMs, all using Metropolis-Hastings for sampling and Adam with default settings for optimization. The models were trained for $2 \cdot 10^4$ iterations using 100 sample points per iteration.

We now turn to investigate the effect of changing the trap strength ω. Table 3 was made by training $N = 2$ models the same way as before, but a separate model for each value of ω. Without knowing the exact expected energies for all ω, it is hard to say much about the accuracy. We know that the ideal case would have $E_0 = 2\omega$, which we see all results overshooting, as one would expect. For the low-ω values we also see a relative difference from 2ω which is much larger than when $\omega = 1$. From [5] we have numerical results for $\omega = 0.5$, which indicate that the exact energy should be $E_0 \approx 1.67$ a.u. The results from the RBM are close, but once again off by about the same magnitude as seen for $\omega = 1$.

Looking at the mean distance between the particles we see that it goes down drastically with increasing ω values. This follows our intuition, as increasing the trap strength should force the particles closer together, with the opposite being true for the reverse.

Lastly, we can see how the results perform with respect to the predictions from the virial theorem. For $\omega = 0.05$ and $\omega = 0.10$ we are more or less exactly correct. The other results are off by a little more. However, considering $\omega = 1$ and $\omega = 0.5$, for which we know the de-

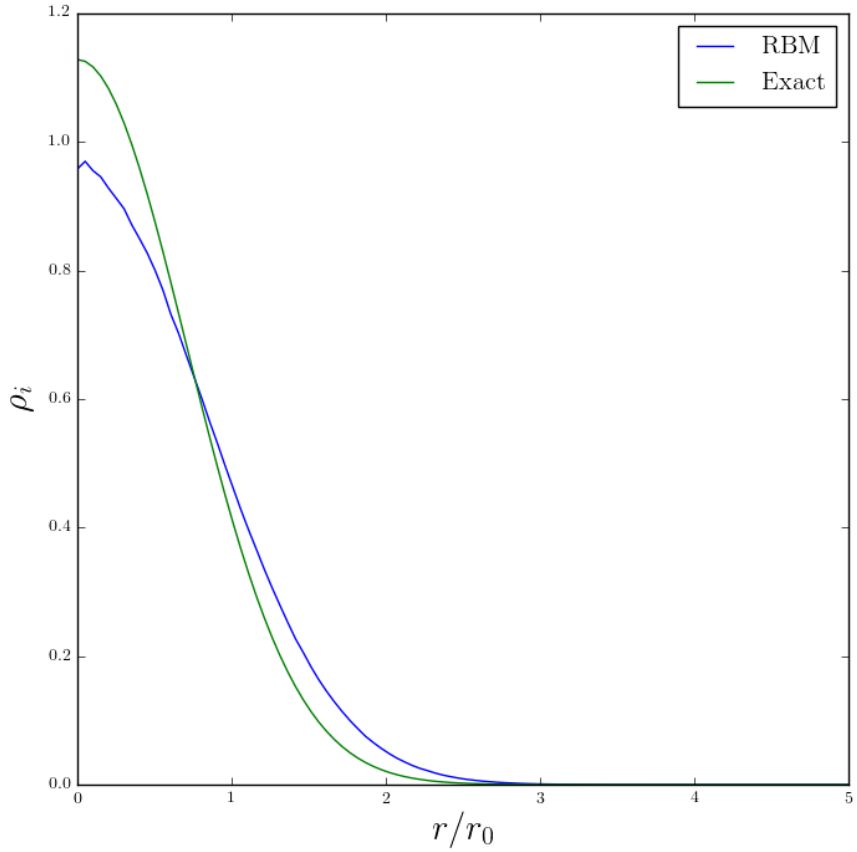

Figure 7: One-body density for the interacting case of two electrons in 2D, and $\omega = 1$, as estimated using the trained $N = 2$ RBM. The exact curve for the non-interacting case is drawn for reference.

sired results, we see that the error in the energy is very comparable to the error in the kinetic energy vs the virial prediction. In other words, the model respects the virial theorem to about the same accuracy as it reproduces the correct energy. We consider this a very positive result, showing that not only has the RBM been able to produce energies that are comparable to the true values, but it has done so in a way that does not conflict with the rest of the laws of physics. One could perhaps argue that the RBM simply learned *some* way to produce good energies, but that the form of the wavefunction did not behave as the real world would have. This is thankfully evidence against that fear.

Another note to make about the virial predictions is that the expected values are, in the same way as every other numerical results here, computed using positions sampled from the learned wavefunction. This means that the *predicted* values for the total kinetic energy given in Table 3 are inaccurate as well. They still do show how the virial theorem would have the energy distributed, and so it should still be sensible to compare our results to these values.

The corresponding one-body density plots for the models trained are shown in Figure 8. We see the clear trend indicated by the mean distances in Table 3. As ω decreases and the pull towards the origin diminishes the electrons tend to drift further and further away. As such, the plot once again follows our intuition, and gives more quantitative evidence that the RBMs have successfully learned the constraints imposed by the quantum laws, at least to a certain extent.

Table 3: Results from trained RBMs for selected values of ω.

ω	$\langle E_L \rangle$	$SE_{\langle E_L \rangle}$	$\langle r_{12} \rangle$	$\langle K \rangle$	$\langle V_{pot} \rangle$	$\langle K \rangle_{\text{virial}}$
0.01	0.1445	0.00166	12.64	0.0099	0.1327	-0.0510
0.05	0.2776	0.00054	11.31	0.0508	0.2268	0.0611
0.10	0.4856	0.00068	7.15	0.0994	0.3857	0.1060
0.50	1.7207	0.00099	2.55	0.3809	1.3396	0.4557
1.00	3.0769	0.00114	1.65	0.7996	2.2775	0.8877

7 Conclusion

Using the RBM as a model for the many-body wavefunction has proven to work without issue in the non-interacting case. All results produced were in strong agreement with the analytical expectations. Accuracy could be made almost arbitrarily high using sufficient training iterations. We could also achieve perfect results by using regularization, although this results was artificially good due to the trivial nature of the ideal wavefunction. The model was able to extend to a higher number of degrees of freedom without issue.

In the full interacting system we saw that the RBMs had a much harder time converging to the correct energies. However, we were still able to produce results within respectable bounds. In addition, we saw how the physical properties of the system, such as mean inter-particle distance, distribution of kinetic and potential energy, and one-body density all followed our expectations to within the same degree of accuracy.

On the pessimistic side, we seemed to run up against a hard barrier on how accurate we could make the results. Ideally we would have seen a trend indicating that allowing sufficient flexibility of the model (i.e. high N) and enough training, we could obtain arbitrary accuracy. We did not see such a trend. However, this has not been disproved by this project, and it is completely feasible that throwing sufficient computing power at this (i.e. use many more samples per iteration, stabilizing the training, and letting it go for a long as needed) could potentially improve the results somewhat.

Despite the obvious lack of absolute accuracy, the approach used in this project has one major benefit. That lies in its adaptability. If we would like to investigate other systems, we only need to supply the Hamiltonian describing it. The model can be left untouched, and deployed on new problems without any hassle. This could prove to be a very useful tool, if we first and foremost want to gain some higher level understanding of the system, and (at least at first) do not have stringent requirements on the absolute accuracy of the results. As we learn the wavefunction in this approach, this gives us maximum information about the system, as apposed to just a single property, and so we can investigate any property of the system we would like.

8 Further Work

The general approach employed here, applying the variational principle in a reinforcement learning scheme, learning the wavefunction through some model, has shown promise both in previous publications as well as in this project. As such, an interesting point of further study would be to test different models on this, and other systems of interest. Hopefully, deploying some different architectures might help to overcome the apparent barrier which the simple RBM seems to have run up against. Thankfully, the pool of possible models to try is vast, and it only remains to find efficient ways to apply them to the quantum problems.

References

[1] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325 (2017), pp. 602–606. ISSN: 0036-8075. DOI: 10.1126/science.aag2302. eprint: http://science.sciencemag.org/content/355/6325/602.full.pdf. URL: http://science.sciencemag.org/content/355/6325/602.

[2] Morten Hjorth-Jensen. *Computational Physics 2: Variational Monte Carlo methods, Lecture Notes*. Spring 2018. URL: http://compphysics.github.io/ComputationalPhysics2/doc/web/course.

[3] Jacob Katriel and Henry Montgomery. "Hund's rule in the two-electron quantum dot". In: *The European Physical Journal B* 85 (Dec. 2012).

[4] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2014). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980.

[5] M. Pedersen Lohne et al. "Ab initiocomputation of the energies of circular quantum dots". In: *Physical Review B* 84.11 (Sept. 2011). DOI: 10.1103/physrevb.84.115302. URL: https://doi.org/10.1103/physrevb.84.115302.

[6] M. Taut. "Two electrons in an external oscillator potential: Particular analytic solutions of a Coulomb correlation problem". In: *Phys. Rev. A* 48 (5 Nov. 1993), pp. 3561–3566. DOI: 10.1103/PhysRevA.48.3561. URL: https://link.aps.org/doi/10.1103/PhysRevA.48.3561.

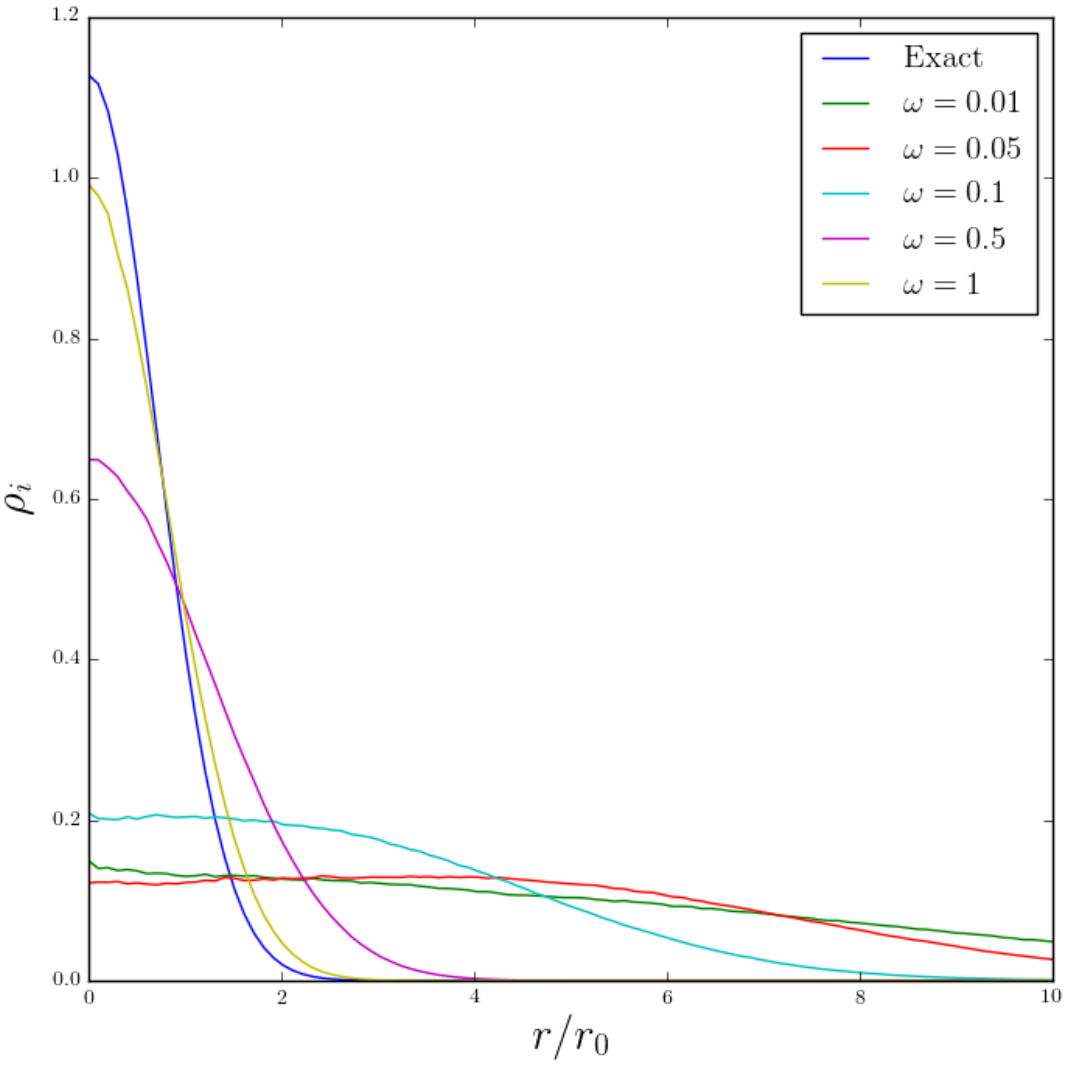

Figure 8: One-body densities for selected values of ω, as produced from RBMs trained as described in the text.

Appendices

A Analytic Expression for the Local Energy

To get started we will be needing a couple of partial derivatives:

$$\frac{\partial}{\partial X_k}(1+e^{v_j}) = e^{v_j}\frac{\partial}{\partial X_k}v_j = e^{v_j}\frac{w_{kj}}{\sigma^2} \tag{38}$$

$$\begin{aligned}\frac{\partial}{\partial X_k}e^{-\sum_i^M u_i} &= e^{-\sum_i^M u_i}\frac{\partial}{\partial X_k}\left(-\sum_i^M u_i\right)\\ &= e^{-\sum_i^M u_i}\frac{\partial}{\partial X_k}(-u_k)\\ &= -e^{-\sum_i^M u_i}\frac{x_k-a_k}{\sigma^2}\end{aligned} \tag{39}$$

We are now ready to compute the first derivate:

$$\begin{aligned}\frac{\partial}{\partial X_k}\Psi &= \frac{1}{Z}\left[\prod_j^N(1+e^{v_j})\frac{\partial}{\partial X_k}e^{-\sum_i^M u_i}\right.\\ &\qquad \left.+e^{-\sum_i^M u_i}\frac{\partial}{\partial X_k}\prod_j^N(1+e^{v_j})\right]\\ &= -\frac{x_k-a_k}{\sigma^2}\Psi + \Psi\sum_j^N\left(\frac{1}{1+e^{v_j}}\frac{\partial}{\partial X_k}(1+e^{v_j})\right)\\ &= -\frac{x_k-a_k}{\sigma^2}\Psi + \Psi\sum_j^N\frac{1}{1+e^{v_j}}\left(\frac{w_{kj}}{\sigma^2}e^{v_j}\right)\\ &= \frac{1}{\sigma^2}\left(a_k - x_k + \sum_j^N\frac{w_{kj}}{1+e^{-v_j}}\right)\Psi.\end{aligned}$$

Before jumping into the second derivative, one more helpful derivative:

$$\begin{aligned}\frac{\partial}{\partial X_k}\left(\frac{w_{kj}}{1+e^{-v_j}}\right) &= -w_{kj}\frac{e^{-v_j}}{(1+e^{-v_j})^2}\frac{\partial(-v_j)}{\partial X_k}\\ &= \frac{w_{kj}^2}{\sigma^2}\frac{e^{-v_j}}{(1+e^{-v_j})^2}\end{aligned} \tag{40}$$

Now, finally the second derivative:

$$\begin{aligned}\frac{1}{\Psi}\frac{\partial^2}{\partial X_k^2}\Psi &= \frac{1}{\Psi}\frac{\partial}{\partial X_k}\left[\frac{1}{\sigma^2}\left(a_k - x_k + \sum_j^N\frac{w_{kj}}{1+e^{-v_j}}\right)\Psi\right]\\ &= \frac{1}{\sigma^2}\left[\frac{\partial}{\partial X_k}\left(a_k - x_k + \sum_j^N\frac{w_{kj}}{1+e^{-v_j}}\right)\right.\\ &\qquad \left.+\left(a_k - x_k + \sum_j^N\frac{w_{kj}}{1+e^{-v_j}}\right)\frac{1}{\Psi}\frac{\partial}{\partial X_k}\Psi\right]\\ &= -\frac{1}{\sigma^2} + \sum_j^N\frac{w_{kj}^2}{\sigma^4}\frac{e^{-v_j}}{(1+e^{-v_j})^2}\\ &\qquad + \frac{1}{\sigma^4}\left(a_k - x_k + \sum_j^N\frac{w_{kj}}{1+e^{-v_j}}\right)^2\end{aligned} \tag{41}$$

The final expression we shall use then for the local energy is:

$$E_L = \sum_{i=1}^P\frac{1}{2}r_i^2 + \sum_{i<j}^P\frac{1}{r_{ij}} - \frac{1}{2}\sum_{k=1}^M\frac{1}{\Psi}\frac{\partial^2}{\partial x_k^2}\Psi, \tag{42}$$

substituting in Equation 41.

B Example of Results of Bad σ^2 choices

In earlier attempts, we tried producing Figure 8 using $\sigma^2 = 1$ in all models. This lead to very interesting one-body-density plots, shown in Figure 9, showing distinct peaks away from the origin. This came as a result of σ^2 being way to small for the weak traps, leading to overly specific selection of positions. Although this plot might look more appealing in some sense, the energy predictions suffer greatly as a result. This serves as an example of how the assumptions we put into the model have a great deal to say on the final results, especially when the model is as simple as the RBM. A more flexible model would perhaps (given time) been able to adapt to bad starting points. The model as presented here however did not show this capability, resulting in strange results such as this.

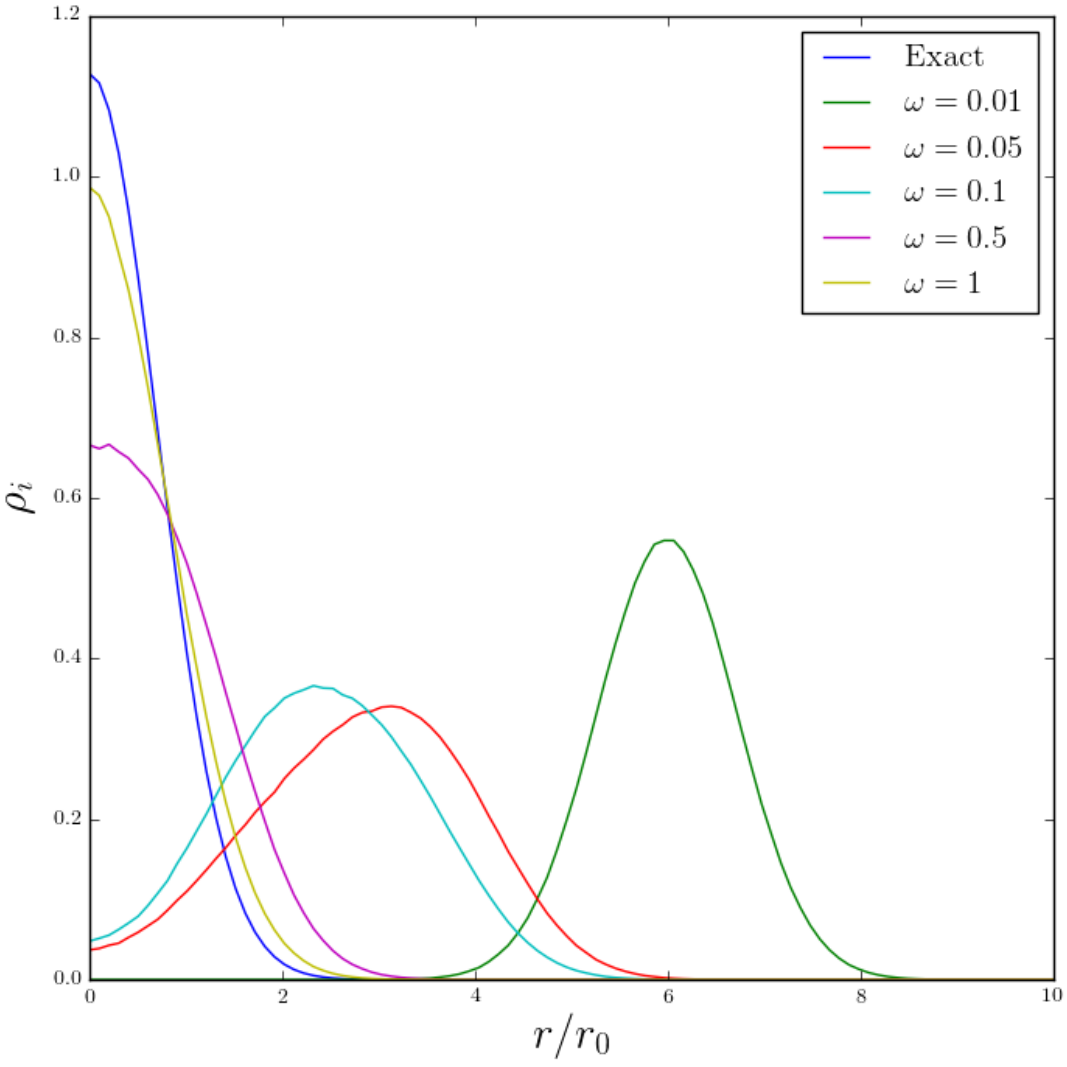

Figure 9: One-body density plots for various ω, obtained by training RBMs in a similar fashion to those used in Figure 8, only with $\sigma^2 = 1$ for all models. The overly specific spread parameter leads to peaks in the plot, which are not observed before. The corresponding energies produced are severely increased as ω gets smaller, indicating that these results are much less accurate.