Utilizing Restricted Boltzmann Machine for Fermions in a Harmonic Oscillator Trap

Herman Brunborg, Elise Malmer Martinsen and Noémie Fritz
(Dated: May 27, 2022)

We have utilized a restricted Boltzmann Machine (RBM) and the Variational Monte Carlo (VMC) method to compute the lowest upper limit for the local energy of two electrons in a two dimensional harmonic oscillator trap. During the energy simulation we emphasized how the parameters and different optimization algorithms impact the accuracy of our results. For two particles in two dimensions, we obtained the lowest absolute error using the Adam optimizer, $\omega = 1$ with a learning rate of 0.2 and two hidden layers. The errors were $7.3709 \cdot 10^{-5}$ for the non-interactive case and $1.7317 \cdot 10^{-1}$ for the interactive case. Using the importance sampler seems to give better performance than the Metropolis Hastings algorithm, although this was not always the case. In addition, it was not possible to find a set of exact parameters which performed best for all the different setups.

I. INTRODUCTION

In this project we will make use of restricted Boltzmann Machine and apply it to a system containing two electrons in a harmonic oscillator trap. The wave function of the system, the so-called *neural network quantum state* (NQS), can be represented using the RBM. We will study the local energy of the system, and see how a small perturbation, which in our case will represent the interactions between the electrons, will affect the system. To be able to find an upper limit of the local energy, we make use of the Variational Monte Carlo method combined with different optimization algorithms, such as the Stochastic Gradient Descent (SGD) method and Adam. We will then compare these two algorithms, and change the learning rate to see how this will affect our results.

Having a quantum mechanical system means that it can be difficult or impossible to solve the system analytically. Machine learning makes it possible to solve our quantum mechanical system, and therefore we use RBM which is also relatively easy to implement compared to a neural network, for instance. Combining machine learning and the VMC method to approximate the local energy, makes it possible to achieve relatively accurate results for quantum mechanical systems.

In section II we describe the Hamiltonian and the NQS of our system, which is represented by our RBM, and we also present the theory behind the Variational Principle. We then describe our methods to implement the RBM in section III. In section IV and V we present and discuss our results. Finally, we provide a short summary and outlook in section VI. This project is a continuation from earlier work, and we will therefore refer directly to our previous project [1] when needed.

II. THEORY

A. The Hamiltonian and our wave function

In this project we will represent our system by a NQS given by

$$\Psi(\boldsymbol{X}) = \sqrt{F_{rbm}(\boldsymbol{X})}, \tag{1}$$

where F_{rbm} is our joint probability distribution given by

$$F_{rbm} = \frac{1}{Z}\exp\left\{-\sum_i^M \frac{(X_i - a_i)^2}{2\sigma^2}\right\} \times \prod_j^N \left[1 + \exp\left\{\sum_i^M \frac{X_i\omega_{ij}}{\sigma^2} + b_j\right\}\right]. \tag{2}$$

Here X_i are the visible nodes in the RBM, corresponding to the position coordinates of a particle i, a is a visible bias, b is a hidden bias, ω_{ij} is a weight that describes the connection between a visible node and a hidden node. Z is the partition function given by

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

where $E(\boldsymbol{X}, \boldsymbol{H})$ is called a "Gaussian-binary", which is a version of a Restricted Boltzmann Machine (RBM) given by

$$E(\boldsymbol{X}, \boldsymbol{H}) = \sum_i^M \frac{(X_i - a_i)^2}{2\sigma_i^2} - \sum_j^N b_j H_j - \sum_{i,j}^{M,N} \frac{X_i \omega_{i,j} H_j}{\sigma_i^2}. \tag{4}$$

Here $\boldsymbol{H}$ are the hidden nodes.

We are going to consider a system of two electrons in two dimensions moving in an isotropic harmonic oscillator potential, where the Hamiltonian is given by

$$\hat{H} = \underbrace{\sum_{i=1}^N \left(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2\right)}_{\hat{H}_0} + \underbrace{\sum_{i<j} \frac{1}{r_{ij}}}_{\hat{H}_1}. \tag{5}$$

where N is the number of particles, r_i is the modulus of the position vector of particle i, ω is the angular frequency, and r_{ij} is the distance between the particles,

i.e. $r_{ij} = |\boldsymbol{r}_i - \boldsymbol{r}_j|$. The first term, $\hat{H}_0$, is the unperturbed part and describes the standard harmonic oscillator, while the second term, $\hat{H}_1$, describes the repulsive interaction between the electrons. Here we have used natural units, i.e. $\hbar = c = e = m_e = 1$. The system contains of two electrons in a quantum dot with frequency of $\hbar\omega = 1$.

To find the analytic expression for the local energy, we need to compute

$$E_L = \frac{1}{\Psi}\hat{H}\Psi \tag{6}$$

The calculation for this can be found in appendix B.

When we ignore the interaction between the electrons, we remove the repulsive part of the Hamiltonian, and the analytical energy of the system is then given by

$$E = \frac{1}{2}P \cdot D, \tag{7}$$

where P is the number of particles and D is the number of dimensions. For 1 electron in 1 dimension, the energy is $E = \frac{1}{2}$ a.u. (atomic units). For 2 electrons in 2 dimensions, the energy is $E = 2$ a.u. On the other hand, if we include the interactions between the electrons the energy is given by 3 a.u.

B. Variational Principle

We use the Variational Principle to evaluate the ground state energy of the electrons. Since we find an upper limit of the ground state energy, the Variational Principle is found to be a very useful method [2]. We can take an educated guess of a normalized trial wave function, Ψ, which depends on one or more variational parameters, and thereby minimize the expectation value of our Hamiltonian with respect to the variational parameters, and from this we find the lowest upper limit of the ground state energy, E_{gs}. This is described in equation 8.

$$E_{gs} \leq \langle\Psi(\alpha)|H|\Psi(\alpha)\rangle = \int \Psi^* H \Psi d\boldsymbol{r} \equiv \langle H\rangle, \tag{8}$$

where H is the Hamiltonian of the system, and our trial wave function depends on the variational parameters, $\alpha_i = a_1, ..., a_M, b_1, ..., b_N, \omega_{11}, ..., \omega_{MN}$. The expectation value of the energy can be interpreted as $\langle H\rangle$. These statements (8) are proven in Griffiths [2]. To find the lowest upper limit of E_{gs}, we must calculate

$$\frac{\partial\langle H\rangle}{\partial\alpha_i} = 0, \tag{9}$$

and solve with respect to α_i. To do this, we will also need to calculate

$$\frac{\partial\Psi}{\partial\alpha_i} = 0. \tag{10}$$

These calculations can be found in appendix C.

III. METHOD

A. Restricted Boltzmann Machine

A restricted Boltzmann Machine is an artificial neural network containing two layers; a hidden layer with length N, and a visible layer with length M. The visible layer is represented by a vector $\boldsymbol{X}$ and gives us our results. The hidden layer is represented by a vector $\boldsymbol{H}$. Each of the nodes in $\boldsymbol{X}$ and $\boldsymbol{H}$ have corresponding biases which are represented by $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. The nodes in the visible layer and the hidden layer are connected by weights, ω_{ij}, stored in a matrix $\boldsymbol{W_{MN}}$ (with size $M \times N$), where the weight ω_{ij} represents the connection between the visible node x_i and the hidden node h_j.

Our RBM model is described by the joint probability distribution from equation 2, and our goal is to minimize the local energy of our system described in section II. To do this, we need to find the values of the weights and biases that gives the lowest local energy. This is done by implementing the SGD method (described in section III E), which will train the RBM to optimize the network's weights and biases. We will also use another optimization algorithm called Adam, which is described in section III F, to see how this will affect our results.

The Boltzmann Machine is restricted such that the nodes within a layer are not connected to each other. This is different from a regular Boltzmann Machine where all the nodes are connected. Figure 1 shows an illustration of a RBM with one hidden layer.

We want to implement a Gaussian-Binary restricted Boltzmann Machine since the values we want to estimate are continuous, and not binary. The Gaussian-Binary RBM is well suited for our case since the visible nodes, the elements in $\boldsymbol{X}$, can take any real value and are normally distributed. We will also use a RBM with different number of hidden layers.

B. Variational Monte Carlo and Blocking Method

We are going to use the Variational Monte Carlo Method to approximate the ground state energy numerically. To find the uncertainty of our simulation we use the Blocking method. These two methods are described in our previous project [1]. Here we will define a Monte Carlo iteration as one sampling step. We will utilize the drift force from [3] in our simulation.

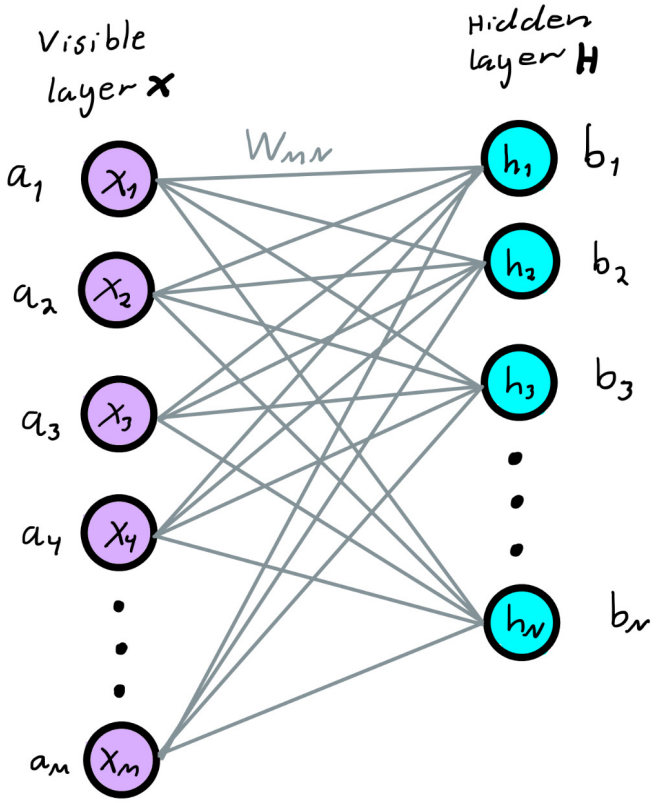

Figure 1. The figure illustrates a restricted Boltzmann Machine with one hidden layer.

C. Metropolis Hastings algorithm

Since our system consists of electrons, which are fermions, we have an anti-symmetric wave function under interchange of particles, and therefore the probability distribution will also be anti-symmetric. We therefore have to use the Metropolis Hastings algorithm, which is a Markov Chain Monte Carlo (MCMC) method. The electrons move based on an acceptance probability $A_{j\to i}$ which depends on the transition probability $T_{i\to j}$. The particles will move from an initial state i, to a new, random position j with some transition probability $T_{i\to j}$. The movement of the particles is either accepted or denied according to

$$A_{j\to i} = \min\left(1, \frac{p_i T_{i\to j}}{p_j T_{j\to i}}\right). \tag{11}$$

Here, p_i is our desired probability distribution that we obtain after running enough iterations. If the movement is accepted the particle will move to state j, and if it is denied, it stays in state i. Then, we calculate the energy of the particle, and repeat this for each Monte Carlo cycle. To generate random numbers that will contribute to either denying or accepting the movement of the particle, we use the pseudo random number generator Mersenne-Twister, which has a period of $2^{19937}-1$ [4].

D. Importance sampling

A more efficient algorithm to accept or deny movements is the Importance sampling algorithm. Importance sampling is based Fokker-Planck and Langevin equations which uses walkers that are biased by our wave function instead of having completely random walkers. From these two equations we obtain Green's function, which is our transition probability given by

$$G(x,y,\Delta t) = \frac{1}{(4\pi D\Delta t)^{3N/2}} \exp\left\{-\frac{(y-x-D\Delta t F(x))^2}{4D\Delta t}\right\}. \tag{12}$$

The acceptance probability used in the Metropolis Hastings algorithm then becomes

$$A(y,x) = \min(1, q(y,x)) \tag{13}$$

where $q(y,x)$ now is defined by

$$q(y,x) = \frac{G(x,y,\Delta t)|\Psi_T(y)|^2}{G(y,x,\Delta t)|\Psi_T(x)|^2}. \tag{14}$$

E. Stochastic Gradient Descent

To numerically find the upper limit of the ground state energy, using a few Monte Carlo cycles, we use the SGD algorithm. We start by giving an initial value of each RBM parameter and calculate the gradient of the expectation value of the energy $\langle E(\alpha_i)\rangle$ with respect to α_i. The gradient is given by

$$G_i = \frac{\partial\langle E_L\rangle}{\partial\alpha_i} = 2\left(\langle E_L \frac{1}{\Psi}\frac{\partial\Psi}{\partial\alpha_i}\rangle - \langle E_L\rangle\langle\frac{1}{\Psi}\frac{\partial\Psi}{\partial\alpha_i}\rangle\right). \tag{15}$$

For each step we correct the value of α_i, subtracting by the gradient of the expectation value of the energy which again is multiplied by a learning rate, γ. This is shown in equation 16.

$$\alpha_{i,k+1} = \alpha_{i,k} - \gamma_k G(\alpha_{i,k}), \quad k \geq 0, \tag{16}$$

where $\alpha_{i,k+1}$ and $\alpha_{i,k}$ are the new and old values of α_i, respectively. Equation 16 is then repeated until the value of α_i converges and we have reached the optimal value of α_i. These optimal values will then be used in a full scale Monte Carlo simulation.

F. Adam

Adam is an optimization algorithm like the SGD algorithm, but instead of having one learning rate for all the parameters that needs to be optimized, Adam calculates a learning rate for each parameter. Adam is a combination of two other algorithms; Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp) [5]. This makes Adam very efficient and

it requires less memory. Adam can bee interpreted as SGD with momentum as it adjusts the learning rate for each parameter by using estimations of first and second moments. The $n-$th moment, m_n, is given by

$$m_n = E[X^n], \tag{17}$$

where X is a random variable which in our case is the gradient of the energy with respect to the RBM parameters. A momentum based SGD will look as follows,

$$\boldsymbol{v}_t = \gamma \boldsymbol{v}_{t-1} + \eta_t \nabla_\alpha E(\alpha_t) \tag{18}$$

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \boldsymbol{v}_t \tag{19}$$

We can now add the second moment of the gradient, which will give us the Adam algorithm,

$$\boldsymbol{g}_t = \nabla_\alpha E(\alpha_t) \tag{20}$$

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \tag{21}$$

$$\boldsymbol{s}_t = \beta_2 \boldsymbol{s}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2 \tag{22}$$

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t} \tag{23}$$

$$\hat{\boldsymbol{s}}_t = \frac{\boldsymbol{s}_t}{1-\beta_2^t} \tag{24}$$

$$\boldsymbol{\alpha}_{t+1} = \boldsymbol{\alpha}_t - \eta_t \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{s}}_t} + \epsilon} \tag{25}$$

where $\boldsymbol{g}_t$ is the gradient of the energy with respect to the parameters in $\boldsymbol{\alpha}$. $\boldsymbol{m}_t$ and $\boldsymbol{s}_t$ are the first and second momentum at iteration step t, respectively. $\hat{\boldsymbol{m}}_t$ and $\hat{\boldsymbol{s}}_t$ are the averages of the first and second moments, respectively, and β_1 and β_2 are typically set to be values right below one, and they describe the memory lifetime of the first and second momentum. The variance of $\boldsymbol{g}_t$ is then

$$\boldsymbol{\sigma}_t^2 = \boldsymbol{s}_t - (\boldsymbol{m}_t)^2. \tag{26}$$

IV. RESULTS

All the figures 2, 3, 4, 5, 6, 7, 8 and 9 show results for the different optimization algorithms, Adam and SGD, both with importance sampling and the brute force Metropolis Hastings algorithm. The ω was initialized to 1 for all simulations except where explicitly stated otherwise.

We start by looking at the impact of the learning rate on the absolute error of our model. We train the RBM in both the non-interactive and interactive case. The results in Figure 2 and 3 show the results for the non-interactive and interactive cases respectively. From these results, we select learning rate to be 0.2 for all the remaining results.

Figure 4 and 5 show the absolute error per optimizer step. The absolute error is here the energy per update of the RBM-weights. One optimizer step is done every $10,000$ Monte Carlo iteration.

Figure 6 and 7 show the absolute error as a function of ω with and without interactions, respectively, for two particles in two dimensions.

Figure 8 and 9 show the energy for two particles in two dimensions which is plotted against the number of hidden layers.

Figure 10 shows the one body density for both the interactive and non-interactive case.

Figure 11, 12, 13, and 14 show the energy for two particles in two dimensions per Monte Carlo iteration with and without interactions between the electrons. We have plotted for both the Adam and the SGD algorithm using either Metropolis Hastings algorithm or importance sampling.

Figure 15 and 16 show the energy per optimizer step. This means that it shows the average energy per 10000 sampling, since we used 10000 Monte Carlo steps per optimizer step.

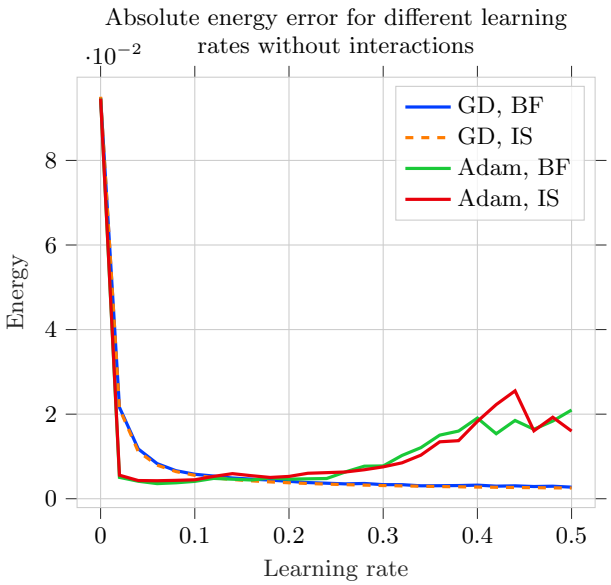

Figure 2. Absolute error compared to the analytical value, $E = 2$ a.u., of the energy as a function of learning rate for the Adam and the SGD algorithm without interactions for either Metropolis Hastings algorithm or importance sampling. BF stands for brute force and IS stands for importance sampling.

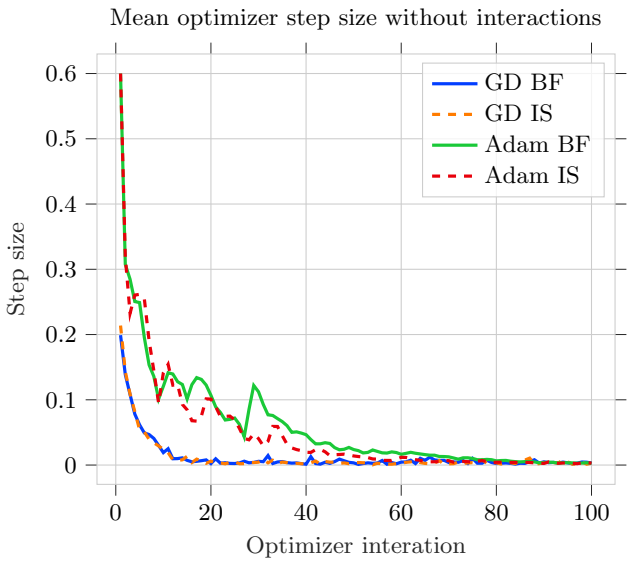

Figure 4. The mean step size over all parameters per optimizer iteration without interactions for the Adam and gradient descent algorithm (GD) and for either brute force sampling (BF) or importance sampling (IS).

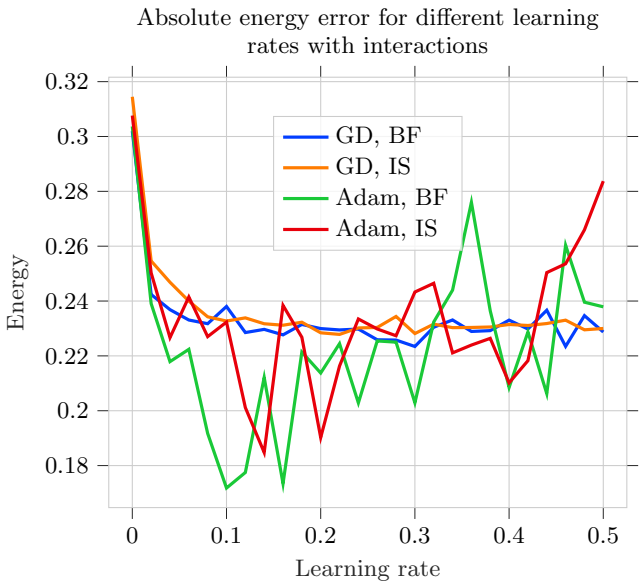

Figure 3. The absolute error compared to the analytical value, $E = 3$ a.u., of the energy as a function of learning rate for the Adam and the SGD algorithm with interactions for either Metropolis Hastings algorithm or importance sampling. BF stands for brute force and IS stands for importance sampling.

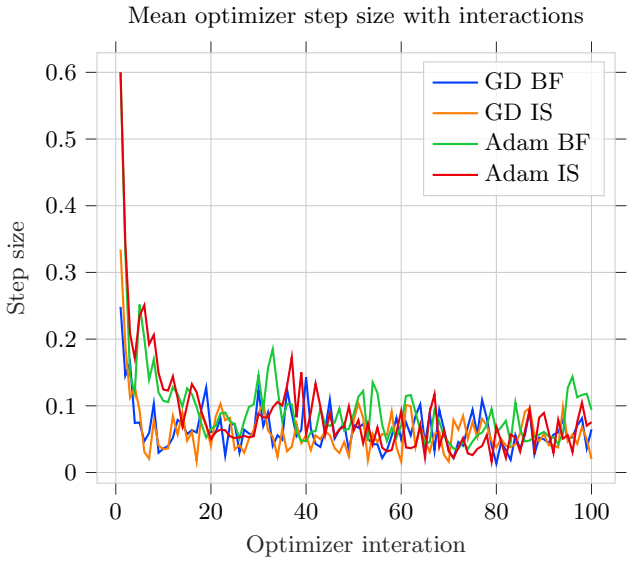

Figure 5. The mean step size over all parameters per optimizer iteration with interactions for the Adam and gradient descent algorithm (GD) and for either brute force sampling (BF) or importance sampling (IS).

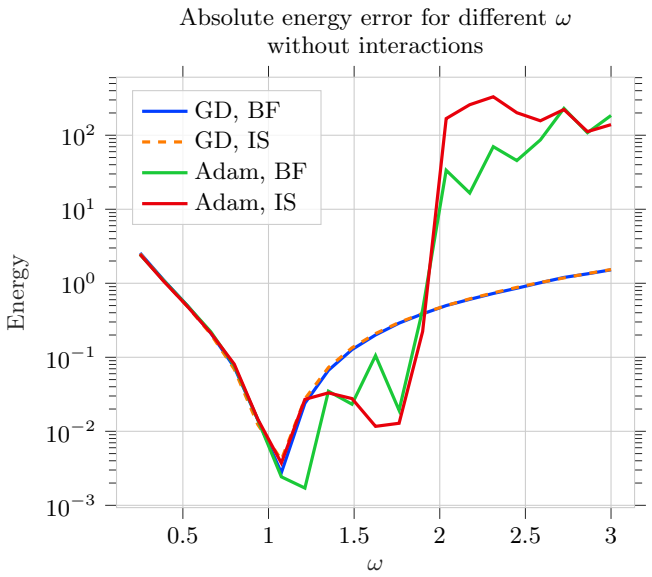

Figure 6. The absolute errors compared to the analytical value, $E = 2$ a.u., of the energy as function of the RBM weights for the Adam and the SGD algorithm with interactions for either Metropolis Hastings algorithm or importance sampling when interactions are not included. We have used a learning rate equal to 0.2. BF stands for brute force and IS stands for importance sampling.

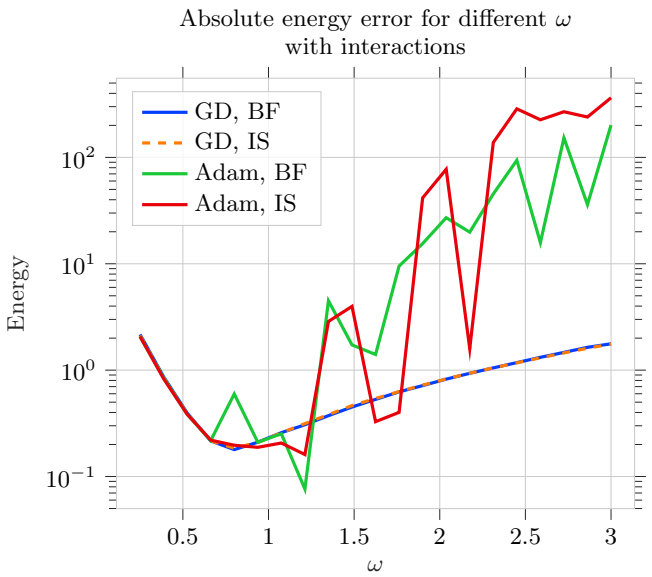

Figure 7. The absolute errors compared to the analytical value, $E = 3$ a.u., as function of the RBM weights for the Adam and the SGD algorithm with interactions for either Metropolis Hastings algorithm or importance sampling when interactions are included. We have used a learning rate equal to 0.2. BF stands for brute force and IS stands for importance sampling.

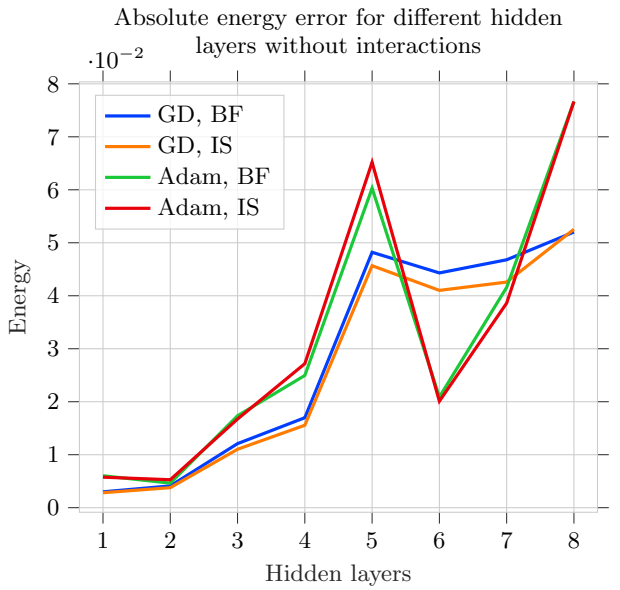

Figure 8. The absolute errors compared to the analytical value, $E = 2$ a.u., of the energy as a function of numbers of hidden layers for the Adam algorithm and the SGD algorithm, either using importance sampling or the brute force Metropolis Hastings algorithm for two particles in two dimensions without interactions. We have used a learning rate equal to 0.2. BF stands for brute force and IS stands for importance sampling.

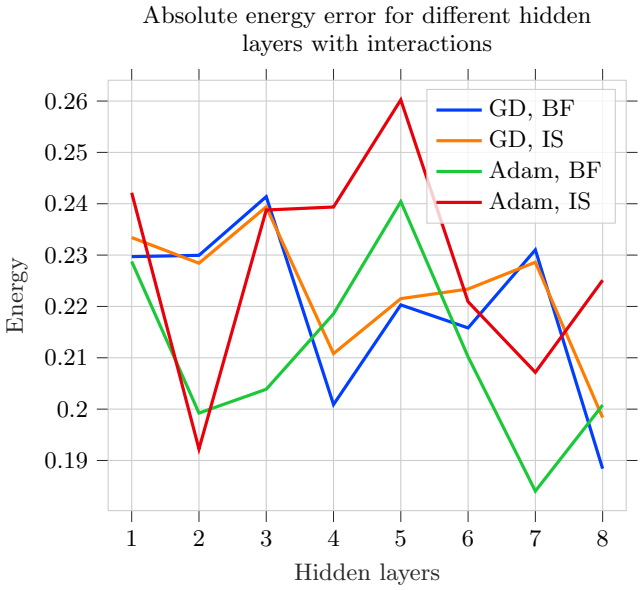

Figure 9. The absolute errors compared to the analytical value, $E = 3$ a.u., of the energy as a function of numbers of hidden layers for the Adam algorithm and the SGD algorithm, either using importance sampling or the brute force Metropolis Hastings algorithm for two particles in two dimensions with interactions. We have used a learning rate equal to 0.2. BF stands for brute force and IS stands for importance sampling.

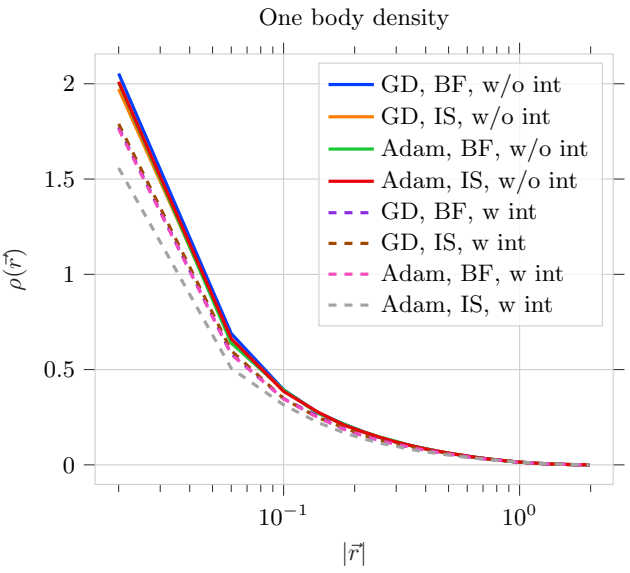

Figure 10. The one body density for the gradient descent algorithm (GD) and Adam optimizer, for both the brute force sampler (BF) and importance sampler (IS) and ran both with and without interactions. The x-axis is log-scale

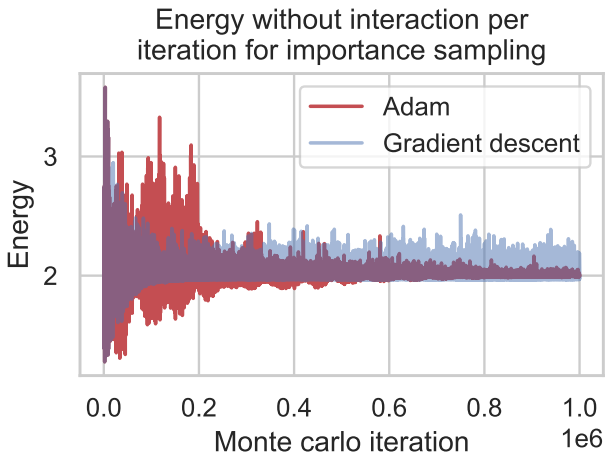

Figure 12. Energy per sampler iteration without interactions between the electrons for the Adam and SGD algorithms, with importance sampling. We have used a learning rate equal to 0.2. The purple area means the energy from the adam and gradient descent overlap.

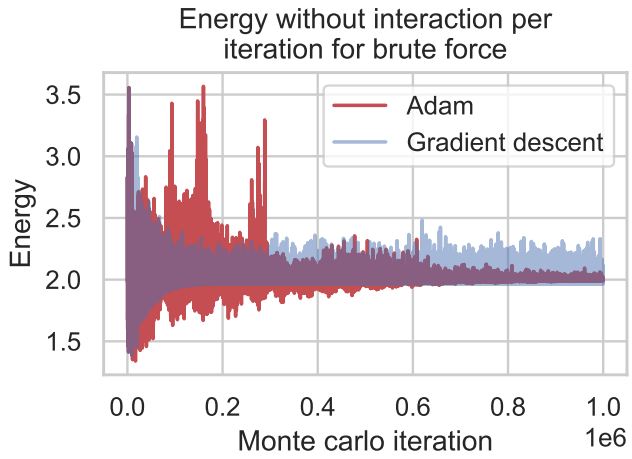

Figure 11. Energy per sampler iteration without interactions between the electrons for the Adam and SGD algorithms, with brute force sampling. We have used a learning rate equal to 0.2. The purple area means the energy from the adam and gradient descent overlap.

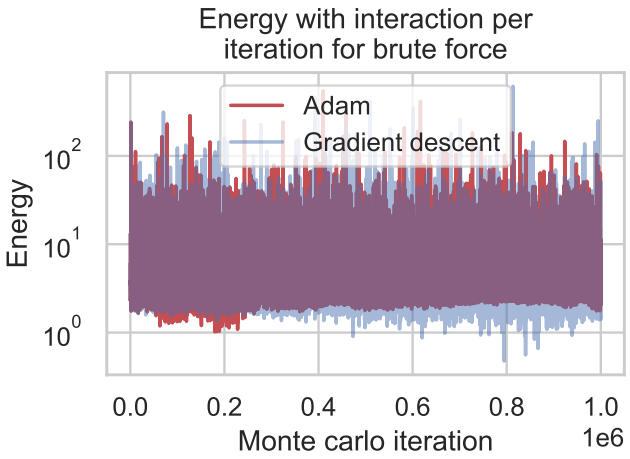

Figure 13. Energy per sampler iteration with interactions between the electrons for the Adam and SGD algorithms, with brute force sampling. We have used a learning rate equal to 0.2. The purple area means the energy from the adam and gradient descent overlap. The y-axis is log-scale.

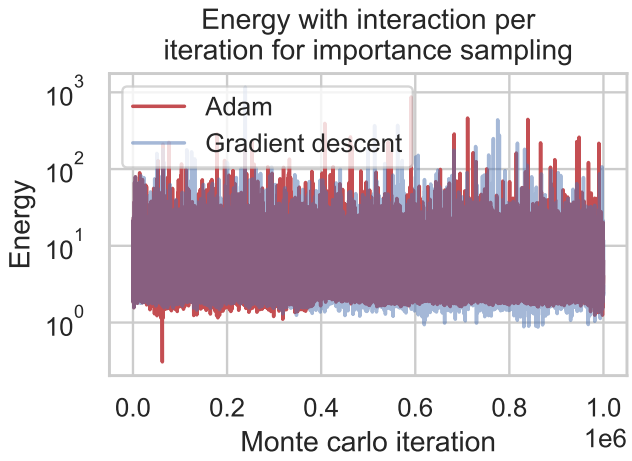

Figure 14. Energy per sampler iteration with interactions between the electrons for the Adam and SGD algorithms, with importance sampling. We have used a learning rate equal to 0.2. The purple area means the energy from the adam and gradient descent overlap. The y-axis is log-scale.

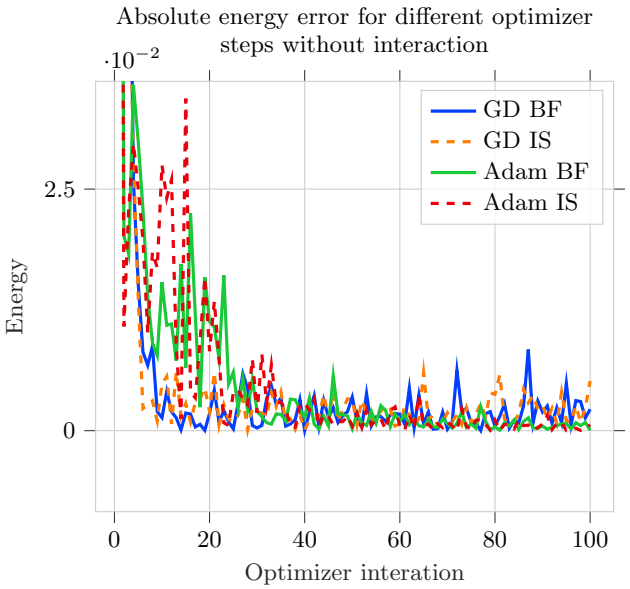

Figure 15. Absolute error compared to the analytical value, $E = 2$ a.u., of the energy per optimizer step without interactions between the electrons for Adam and SGD algorithms, with either brute force or importance sampling. We have used a learning rate equal to 0.2. Here BF stands for brute force and IS stands for importance sampling.

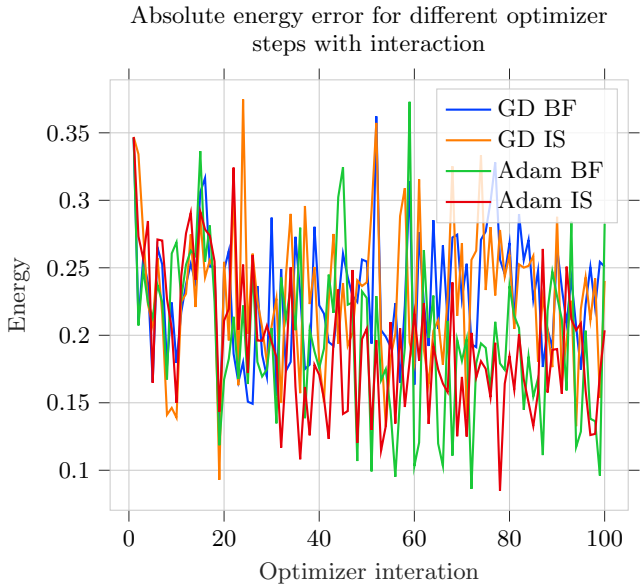

Figure 16. Loss function for the energy per optimizer step with interactions between the electrons for Adam and SGD algorithms, with either brute force or importance sampling. We have used a learning rate equal to 0.2. BF stands for brute force and IS stands for importance sampling.

Table I. Table showing the absolute error (Abs err) compared to the analytical value and the standard deviation (σ_{block}) for both Adam and SGD algorithm using either Metropolis Hastings algorithm (BF) or importance sampling (IS). Here, N is the number of particles, D is the number of dimensions, "Opti" indicates what optimization algorithm that has been used, and "Spl" shows what kind of sampler is used. "Inter" shows if the interactions are included or not, so "w/o" stands for "without interactions". All results were generated with learning rate of 0.2

N	**D**	**Inter**	ω	**Opti**	**Spl**	**Abs err**	σ_{block}
1	1	w/o	1	GD	BF	1.4513e-04	8.4560e-05
1	1	w/o	1	GD	IS	1.0735e-04	8.4046e-05
1	1	w/o	1	Adam	BF	3.4801e-05	3.4511e-05
1	1	w/o	1	Adam	IS	5.4730e-05	4.2549e-05
2	2	w/o	1	GD	BF	9.5556e-04	3.2089e-04
2	2	w/o	1	GD	IS	3.5216e-04	3.1685e-04
2	2	w/o	1	Adam	BF	2.4907e-04	1.5297e-04
2	2	w/o	1	Adam	IS	7.3709e-05	1.3704e-04
2	2	with	0.75	GD	BF	1.7760e-01	5.9627e-03
2	2	with	0.75	GD	IS	1.7197e-01	6.1799e-03
2	2	with	0.75	Adam	BF	1.8957e-01	9.0860e-03
2	2	with	0.75	Adam	IS	1.7475e-01	6.2763e-03
2	2	with	1	GD	BF	2.3851e-01	6.0561e-03
2	2	with	1	GD	IS	2.2943e-01	6.0229e-03
2	2	with	1	Adam	BF	1.7944e-01	6.2475e-03
2	2	with	1	Adam	IS	1.7317e-01	5.9029e-03

V. DISCUSSION

From figure 2 and 3, we see that finding the correct learning rate seems to be important to get a good performance with respect to the absolute error, but also here there are no clear trends showing that some learning rates perform better than others across the different optimizer algorithms and samplers, both for the interactive and non-interactive cases. We also see that even for a specific sampler, optimizer and interactive/non-interactive case, the absolute error does not seem to be convex, with one clear optimal learning rate. This might indicate that there is some randomness associated with the error. We also see that the Adam optimizer seems to be more impacted by changes in learning rate as compared to the SGD algorithm, which molds a much more stable error for the different learning rates.

In figure 6 and 7, we see that there is a clear correlation between ω and the absolute error. It also seems like the optimizer and sampler do not have a huge impact on what ω performs best, and we see a very clear trend that values away from the optimal ω perform worse. Another interesting finding is that the optimal ω seems to be different for the interactive and non-interactive case, with the optimal ω for the non-interactive case being 1, while it is somewhere around 0.75 for the interactive case. One important aspect to note, is that even though the absolute error is smaller for $\omega = 0.75$, this does not mean that $\omega = 0.75$ is the optimal parameter, and it might also be $\omega = 1$, even though we get a bigger absolute error. If we for instance were to multiply all energies by 0.95, it would give us a seemingly better energy, as it was closer to $E = 3a.u.$, but the model would still be worse. Therefore, more investigation would be required to find out what the best ω is, and is also the reason for why $\omega = 1$ was used throughout the rest of the simulations.

Another trend which can be seen from figure 6 and 7, is that the SGD algorithm seems to be more stable for the different ω values, meaning that varying ω impacts the absolute error less than for the Adam method. This means that the SGD algorithm might be better suited for finding the best parameters, like ω even though the performance of the Adam optimizer might be slightly better.

When we look at the one body density from figure 10, we see that the particles seem to be more close to the middle for the non-interactive case when compared to the interactive case. This indicates that when interactions are enabled, the particles repel each other, and the closer they are to the middle, the closer they are to each other, and the more they repel each other. We also see that the one body density behaves quite similarly for all the setups without interactions. With interactions, they also behave very similarly, except for the Adam optimizer and importance sampling, which seems to be more spread out than the others. Since one of the most interesting aspects of the one body density is the difference between the interactive and non-interactive cases, they were plotted in the same plot, even though that made it challenging to see the exact distribution for each individual setup.

From figure 11, 12, 13 and 14 we see that the difference between using the brute force Metropolis Hastings algorithm and importance sampling seems to be quite small, with similar behaviour per optimizer. This does not appear to be the case for the different optimizers. It is especially noticeable for the non-interactive case, where the Adam optimizer converges better than the SGD optimizer. One interesting observation is that the energy for the Adam optimizer varies more in the beginning of the optimization, with the SGD method converging faster to the area around $E = 2a.u.$. For the interactive case, we see large spikes for both the Adam optimizer and the SGD methods, and we can therefore not find one clear winner among the two methods from this plot.

Looking at figure 15 we see that the SGD algorithm converges faster than Adam algorithm, but that the Adam algorithm seems to converge to a higher degree. In light of plot 11 and 12, this is expected. We also see that both converge, and that they seem to converge towards the analytical value $E = 2a.u.$. There also seems to be a slight difference between the Metropolis Hastings algorithm and importance sampling, but with no clear winner among the two.

A possible explanation for the slower but seemingly better convergence for the Adam optimizer compared to the SGD optimizer for the non-interactive case, might be explained by figure 4. Here, we see that the steps are bigger for quite some time for the Adam optimizer as compared to the SGD method. This might mean that when training with the Adam optimizer, we are able to find better weights, but that these are challenging to find, explaining why it both uses more time and why the resulting performance is better. We see a similar trend for the interactive case in figure 5, with bigger optimizer steps in the beginning. Here the implications on the absolute error per iteration are not as clear, since the errors are not converging to the same degree as for the non-interactive case, but it might explain why the Adam optimizer generally performs better than the SGD method also for the interactive case. Once again, these are only empirical observations, and would need to be investigated more thoroughly for this hypothesis to be verified.

In figure 16, we see no clear convergence. However, it seems that the Adam optimizer has a lower absolute error on average once the RBM has trained for a bit. This means that once again, the ADAM optimizer might be preferred, but this is only an indication, and we can therefore not conclude that the Adam method is better than the SGD for our case based on this plot.

When we look at the impact of different number of hid-

den layers, from figure 8 and 9, we again see that there is no clear common trend between the interactive and non-interactive case, when it comes to how many hidden layers produce the best results. For the non-interactive case, we see that the absolute error seems to increase with the number of hidden layers. This might indicate that using more hidden layers will result in overfitting. For the interactive case, there is no clear correlation between the number of hidden layers and the absolute error, which indicates that increasing the complexity of the model does not seem to improve the performance. This might indicate that if we want to obtain a lower absolute error, one would need to change the underlying model rather than adjusting the parameters to be able to approximate more complex functions built on top of the existing model. There might also be a slight trend for more complex models performing better, but more testing would be required to conclude that.

From table I, we see that the Adam optimizer performs best for most cases, and once again, the difference between our two sampling methods is small. For the non-interactive case, we see that the standard error from the blocking method makes sense, when seen in conjunction with the absolute errors. This means that the actual value is reasonably close to the predicted value when seen in conjunction with the standard deviation from the blocking. For the interactive case however, we see that the error is quite low compared to the absolute error. This means that the error here does not only come from the sampler, but this also indicates that our underlying model is not entirely correct.

We see form table I that the importance sampling performs better than the brute force sampler for seven out of the eight parameter configurations. This contradicts with the fact that the performance is quite similar both the Metropolis Hastings algorithm and the importance sampler, which is also the case for our different optimizers.

VI. CONCLUSION

In our findings, using a learning rate of around $0.1 - 0.2$, $\omega = 1$ for the non-interactive and $\omega = 0.75$ for the interactive case with two hidden layers seems to produce the most stable and best results. We have also seen that the Adam optimizer seems to perform better than the SGD method, while the SGD generally is more stable. Another interesting finding is the fact that not all of the parameters have a strictly convex nature with one value for the parameter being clearly better than the others, although we could not get to the bottom of whether the reason for this is due to randomness or if there just is no strictly best value for all the parameters.

In addition, we found that there is no set of parameters that suits all of our cases. It varies if it is the Adam or the SGD algorithm performs best. Therefore, one cannot simply find the best parameters for a simple case, and expect the same parameters to also fit another problem as well. It therefore seems like one would need to spend extra energy fine-tuning the parameters for the exact case of interest.

For further works, we propose to explore alternatives to the RBM, eg. a neural network as well as looking at other optimization algorithms. Another possibility is trying a similar setup, for instance simulating bosons as opposed to fermions. One could also try to do a more thorough investigation about the best parameters, like ω.

ACKNOWLEDGMENTS

We would like to thank Morten Hjorth-Jensen and Øyvind Sigmundson Schøyen for answering all of the 10^{∞} questions we asked from dusk till dawn.

[1] H. Brunborg, N. Fritz, and E. Martinsen, Variational monte carlo simulations for elliptical and spherical bose gas, accessed 20.04.22.
[2] D. J. Griffiths and D. F. Schroeter, *Quantum Mechanics 3rd Edition.* (Cambridge Univesity Press, 2018).
[3] M. Hjorth-Jensen, Advanced topics in computational physics, section 14.7, accessed 14.05.22.
[4] W. J. Buchanan, Mersenne twister, accessed 25.03.22.
[5] J. Brownlee, Gentle introduction to the adam optimization algorithm for deep learning, accessed 28.04.22.

Appendix A: Source code

Our code can be found here: https://github.com/elisemma/FYS4411/tree/main/project2

Appendix B: Analytic expression for local energy

The local energy is given by

$$E_L = \frac{1}{\Psi} H\Psi, \tag{B1}$$

We have that

$$\frac{1}{\psi}\frac{\partial \psi^2}{\partial x^2} = \Big(\frac{\partial \ln \psi}{\partial x}\Big)^2 + \frac{\partial^2}{\partial x^2}\ln\psi. \tag{B2}$$

To find the analytical expression of the local energy we use equation 1, 5 and B2. We then obtain

$$E_L = \frac{1}{\psi}\Big[\sum_i^N(-\frac{1}{2}\nabla_i^2 + \frac{1}{2}\omega^2 r_i^2) + \sum_{i<j}\frac{1}{r_{ij}}\Big]\psi \tag{B3}$$

$$= \frac{1}{\psi}\sum_i^N -\frac{1}{2}\nabla_i^2\psi + \sum_i^N \frac{1}{2}\omega^2 r_i^2 + \sum_{i<j}\frac{1}{r_{ij}} \tag{B4}$$

$$= -\frac{1}{2}\sum_i^N\sum_d^D\Big[\Big(\frac{\partial}{\partial x_{id}}\ln\psi\Big)^2 + \frac{\partial^2}{\partial x_{id}^2}\ln\psi\Big] + \frac{1}{2}\sum_i^N\omega^2 r_i^2 + \sum_{i<j}\frac{1}{r_{ij}}. \tag{B5}$$

Since we have $\Psi(\boldsymbol{X}) = \sqrt{F_{rbm}(\boldsymbol{X})}$, we obtain

$$\frac{\partial}{\partial x_{id}}\ln\psi = \frac{x_{id}+a_i}{2\sigma^2} + \frac{1}{2\sigma^2}\sum_i^N \frac{\omega_{ij}}{\exp\Big\{-b_j - \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\Big\}+1} \tag{B6}$$

and

$$\frac{\partial^2}{\partial x_{id}^2}\ln\psi = \frac{1}{2\sigma^2} - \frac{1}{2\sigma^4}\sum_i^N \omega_{ij}^2 \frac{\exp\Big\{-b_j - \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\Big\}}{[\exp\Big\{-b_j - \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\Big\}+1]^2}. \tag{B7}$$

Appendix C: Analytic expression of the gradient of the local energy

The gradient of the local energy with respect to the Restricted Boltzmann Machine parameters is given as in equation 15. We then compute

$$\frac{\partial \ln\Psi}{\partial a_i} = \frac{1}{\sigma^2}(X_i - a_i), \tag{C1}$$

$$\frac{\partial \ln\Psi}{\partial b_i} = \frac{1}{\exp\Big\{b_j + \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\Big\}+1}, \tag{C2}$$

and

$$\frac{\partial \ln\Psi}{\partial \omega_{ij}} = \frac{X_i}{\sigma^2\exp\Big\{-b_j - \sum_i^M \frac{X_i\omega_{ij}}{\sigma^2}\Big\}+1}. \tag{C3}$$