Training and Classification using a Restricted Boltzmann Machine on the D-Wave 2000Q

Vivek Dixit[1,3], **Raja Selvarajan**[1,3], **Muhammad A.Alam**[2,4], **Travis S. Humble**[5], and ***Sabre Kais**[1,3,4]

[1] Department of Chemistry, Purdue University, West Lafayette, IN
[2] Department of Electrical and Computer Engineering , Purdue University, West Lafayette, IN
[3] Department of Physics and Astronomy, Purdue University, West Lafayette, IN
[4] Birck Nanotechnology Center, Purdue University, West Lafayette, IN
[5] Quantum Computing Institute, Oak Ridge National Laboratory, Oak Ridge, TN

E-mail: `kais@purdue.edu`

Abstract. Restricted Boltzmann Machine (RBM) is an energy based, undirected graphical model. It is commonly used for unsupervised and supervised machine learning. Typically, RBM is trained using contrastive divergence (CD). However, training with CD is slow and does not estimate exact gradient of log-likelihood cost function. In this work, the model expectation of gradient learning for RBM has been calculated using a quantum annealer (D-Wave 2000Q), which is much faster than Markov chain Monte Carlo (MCMC) used in CD. Training and classification results are compared with CD. The classification accuracy results indicate similar performance of both methods. Image reconstruction as well as log-likelihood calculations are used to compare the performance of quantum and classical algorithms for RBM training. It is shown that the samples obtained from quantum annealer can be used to train a RBM on a 64-bit 'bars and stripes' data set with classification performance similar to a RBM trained with CD. Though training based on CD showed improved learning performance, training using a quantum annealer eliminates computationally expensive MCMC steps of CD.

1. Introduction

Quantum computing holds promise for a revolution in the field of science, engineering, and industry. Most of the R&D work related to quantum computing is focused on gate based approach [1, 2, 3], an alternative to this is the adiabatic quantum computing (AQC) [4, 5, 6, 7]. In AQC, a system of qubits start with a simple Hamiltonian whose ground state is known. Gradually, the initial Hamiltonian evolves into a final Hamiltonian. The final Hamiltonian is designed in such a way that its ground state corresponds to the solution to the problem of interest. According to the quantum adiabatic theorem, a quantum system that begins in the non-degenerate ground state of a time-dependent Hamiltonian will remain in the instantaneous ground state provided the Hamiltonian changes sufficiently slowly [8, 9, 10, 11]. It has been shown theoretically that an AQC machine can solve QMA-complete problems [12]. QMA-

complete is quantum analogue of NP-complete. Thus, AQC can give solutions which are very difficult to find using classical methods.

For machine learning and optimization problems, the D-Wave's quantum annealer has been investigated by several researchers. Mott *et al* . [13] used D-Wave to classify Higgs-boson-decay signals vs. background. They showed that the quantum annealing-based classifiers perform comparable to the state-of-the-art machine learning methods. Mniszewski *et al* . [14] found that the results for graph partitioning using D-Wave systems are comparable to commonly used methods. Alexandrov *et al* . [15] used a D-Wave for matrix factorization. Lidar *et al* . [16] used a D-Wave for classification of DNA sequences according to their binding affinities. Kais *et al* . have used D-Wave's quantum annealer for prime factorization and electronic structure calculation of molecular systems [17, 18].

RBM is a widely used machine learning technique for unsupervised and supervised tasks. However, its training is time consuming due to the calculation of model dependent term in the gradient learning. RBMs are most commonly trained using a method known as Contrastive Divergence (CD). CD uses Markov chain Monte Carlo (MCMC) which requires long equilibration time. Further, CD does not follow the gradient of the log-likelihood [19], and is not guaranteed to give correct results. Therefore, better sampling methods can have a positive impact on RBM learning. In this regard, several researchers have investigated the D-Wave annealer for RBM applications. Adachi *et al* . [20] used quantum annealing for training RBMs, which were further used as layers of a two layered deep neural network and post-trained by back-propagation algorithm. The authors conclude that the hybrid approach results in faster training, although the relative effectiveness of RBM trained using a quantum-annealer vs. contrastive divergence has not been documented. Among other works related to the topic, Dumoulin *et al* . [21] assessed the effect of various parameters like limited connectivity, and noise in weights and biases of RBM on its performance. Koshka *et al* . explored the energy landscape of a RBM embedded onto a D-Wave machine, which was trained with CD [22, 23, 24, 25]. There has been growing interest in quantum machine learning including Boltzmann machines [26, 27, 28, 29], however training a quantum machine learning models on moderate or large dataset is challenging.

In this work, our objective is to train a RBM using samples from D-Wave quantum annealer and compare its performance with a RBM trained with contrastive divergence. The model dependent term in the gradient of log-likelihood can be estimated by using samples drawn from a quantum annealer. Trained models were compared based on the classification accuracy, image reconstruction and log-likelihood values. In order to carry out this study, bars and stripes (BAS) data set has been used. BAS is a binary data set with each record comprising of 64 bits.

2. Methods

2.1. Restricted Boltzmann Machine (RBM)

A Restricted Boltzmann Machine is an energy based model, inspired from Boltzmann distribution of energies of Ising model of spins. It models the underlying probability distribution of data set and can be used for machine learning applications. A RBM is comprised of two layers of binary variables known as visible and hidden layers. The variables or units in visible and hidden layers are denoted as $\{v_1, v_2, ..., v_N\}$ and $\{h_1, h_2, ..., h_M\}$, respectively. The variables in one layer interact with the variables in the other layer, however interactions between the variables in the same layer are not permitted. The energy of the model is given by:

$$E = -\sum_i b_i v_i - \sum_j c_j h_j - \sum_{i,j} v_i w_{ij} h_j, \tag{1}$$

where b_i and c_j are bias terms; w_{ij} represents the strength of interaction between variables v_i and h_j. Let us represent the variables in the visible layer collectively by a vector: $v \in \{0, 1\}^N$, similarly for the hidden layer: $h \in \{0, 1\}^M$. Using this representation equation (1) can be written as:

$$E(v, h) = -b^T v - c^T h - h^T W v, \qquad (2)$$

where b and c are bias vectors at the visible and hidden layer, respectively; W is a weight matrix composed of w_{ij} elements. The probability that the model assigns to the configuration $\{v, h\}$ is:

$$P(v, h) = \frac{1}{Z} e^{-E(v,h)}, \qquad Z = \sum_v \sum_h e^{-E(v,h)}, \qquad (3)$$

where Z is partition function. Substituting value of $E(v, h)$, from equation (2), we get:

$$Z = \sum_v \sum_h e^{b^T v + c^T h + h^T \cdot W \cdot v} = \sum_h e^{c^T h} \sum_v e^{b^T v + h^T \cdot W \cdot v} \qquad (4)$$

$$Z = \sum_h e^{c^T h} \sum_v e^{(b^T + h^T \cdot W)v} = \sum_h e^{c^T h} \sum_v e^{s \cdot v}, \qquad (5)$$

where s is:

$$s = b^T + h^T W = [s_1, s_2, ..., s_N]; \qquad (6)$$

N is the number of variables in the visible layer. Now, Z can be written as:

$$Z = \sum_h e^{c^T h} \prod_{j=1}^N (1 + e^{s_j}) \qquad (7)$$

From equation (7), we notice that the calculation of Z involves summation over 2^M configuration, where M is the number of variables in hidden layer. On the contrary, we need 2^{M+N} configurations to evaluate Z using equation (3).

2.2. Minimization of the log-likelihood

The partition function Z is hard to evaluate. The joint probability, $P(v, h)$, being a function of Z is also hard. Due to the bipartite graph structure of the RBM, the conditional distributions $P(h|v)$ and $P(v|h)$ are simple to compute,

$$P(h|v) = \frac{P(v, h)}{P(v)} \qquad (8)$$

where $P(v)$ is given by the following expression:

$$P(v) = \frac{\sum_h e^{-E(v,h)}}{Z}. \qquad (9)$$

Substituting values from equation (3) and equation (9) into equation (8) gives:

$$P(h|v) = \frac{exp\{\sum_j c_j h_j + \sum_j (v^T W)_j h_j\}}{Z'}, \qquad (10)$$

where

$$Z' = \sum_h exp(c^T h + h^T W v). \tag{11}$$

$$P(h|v) = \frac{1}{Z'} \prod_j exp\{c_j h_j + (v^T W)_j h_j\} \tag{12}$$

Let's denote

$$\widetilde{P}(h_j|v) = exp\{c_j h_j + (v^T W)_j h_j\} \tag{13}$$

Now, the probability to find an individual variable in the hidden layer $h_j = 1$ is:

$$P(h_j = 1|v) = \frac{\widetilde{P}(h_j = 1|v)}{\widetilde{P}(h_j = 0|v) + \widetilde{P}(h_j = 1|v)} = \frac{exp\{c_j + (v^T W)_j\}}{1 + exp\{c_j + (v^T W)_j\}} \tag{14}$$

Thus, the individual hidden activation probability is given by:

$$P(h_j = 1|v) = \sigma\left(c_j + (v^T W)_j\right), \tag{15}$$

where σ is the sigmoid function. Similarly, the activation probability of a visible variable conditioned on a hidden vector h is given by:

$$P(v_i = 1|h) = \sigma\left(b_i + (h^T W)_i\right). \tag{16}$$

A RBM is trained by maximizing the likelihood of the training data. The log-likehood is given by:

$$l(W, b, c) = \sum_{t=1}^{N} \log P\left(v^{(t)}\right) = \sum_{t=1}^{N} \log \sum_h P\left(v^{(t)}, h\right) \tag{17}$$

$$l(W, b, c) = \sum_{t=1}^{N} \log \sum_h e^{-E(v^{(t)}, h)} - N \cdot \log \sum_{v,h} e^{-E(v,h)}. \tag{18}$$

where $v^{(t)}$ is a sample from the training dataset. Denote $\theta = \{W, b, c\}$. The gradient of the log-likelihood is given by:

$$\nabla_\theta l(\theta) = \sum_{t=1}^{N} \frac{\sum_h e^{-E(v^{(t)}, h)} \nabla_\theta(-E(v^{(t)}, h))}{\sum_h e^{-E(v^{(t)}, h)}} - N \cdot \frac{\sum_{v,h} e^{-E(v,h)} \nabla_\theta(-E(v,h))}{\sum_{v,h} e^{-E(v,h)}} \tag{19}$$

$$\nabla_\theta l(\theta) = \sum_{t=1}^{N} \langle \nabla_\theta(-E(v^{(t)}, h)) \rangle_{P(h|v^{(t)})} - N \cdot \langle \nabla_\theta(-E(v,h)) \rangle_{P(v,h)}, \tag{20}$$
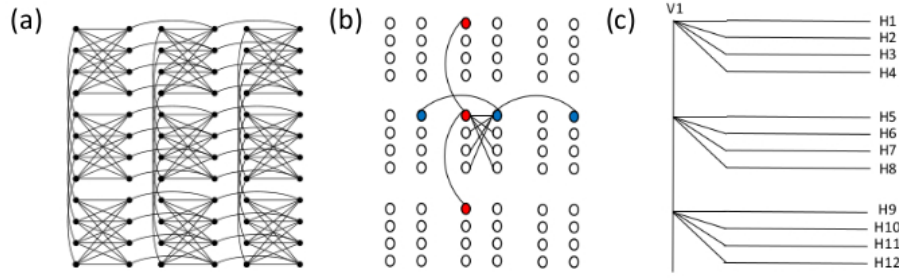
Figure 1. (a) C3 Chimera graph of qubits. (b) Three vertical (red) and horizontal (blue) qubits are chained to form a visible and a hidden unit. (c) Connectivity of visible unit (V1) with 12 hidden units. Here, each unit is formed by ferromagnetic couplings between three qubits.

where $\langle \cdot \rangle_{P(v,h)}$ is expectation value with respect to distribution $P(v,h)$. Since $\theta = \{W, b, c\}$, therefore the gradient with respect to θ can also be expressed in terms of its components:

$$\nabla_w l = \frac{1}{N} \sum_{t=1}^{N} \langle v^{(t)} \cdot h^{(t)} \rangle_{P(h|v^{(t)})} - \langle v \cdot h \rangle_{P(v,h)} \tag{21}$$

$$\nabla_b l = \frac{1}{N} \sum_{t=1}^{N} \langle v^{(t)} \rangle_{P(h|v^{(t)})} - \langle v \rangle_{P(v,h)} \tag{22}$$

$$\nabla_c l = \frac{1}{N} \sum_{t=1}^{N} \langle h^{(t)} \rangle_{P(h|v^{(t)})} - \langle h \rangle_{P(v,h)} \tag{23}$$

The first term in equation (20) is expectation value of $\nabla_\theta(-E(v^{(t)}, h))$ with respect to Boltzmann distribution, $v^{(t)}$ is a row vector from the training dataset and h is a hidden vector. Given $v^{(t)}$, h can be calculated via equation (15).

The second term in equation (20) is a model dependent term, expectation value of $\nabla_\theta(-E(v, h))$, v and h can be any possible binary vectors. This term is difficult to evaluate as it requires all possible combinations of v and h. Generally this term is estimated using contrastive divergence, where one uses many cycles of Gibbs sampling to transform the training data into data drawn from the proposed distribution. We used equation (15) and (16) to sample from hidden and visible layers repeatedly. Once we have the gradient of log-likelihood (equation (18)), weights and biases can be estimated using gradient accent optimization:

$$\theta_j^{new} = \theta_j^{old} + \epsilon \cdot \nabla_{\theta_j} l(\theta_j) \tag{24}$$

APP PROFONDIRE

where ϵ is the learning rate.

Alternatively, the second term can be calculated using samples drawn from the D-Wave quantum annealer, which is a much faster procedure than MCMC.

2.3. D-Wave Hamiltonian and arrangement of qubits

The Hamiltonian for a D-Wave system of qubits can be represented as:

$$h_{Ising} = -\frac{A(s)}{2}\Big(\sum_i \hat{\sigma}_x^i\Big) + \frac{B(s)}{2}\Big(\sum_i h_i\hat{\sigma}_z^i + \sum_{(i>j)} J_{ij}\hat{\sigma}_z^i\hat{\sigma}_z^j\Big) \tag{25}$$

where $\hat{\sigma}_{x,z}^{(i)}$ are Pauli matrices operating on qubit q_i. h_i and $J_{i,j}$ are the qubit biases and coupling strengths. s is called the anneal fraction. $A(s)$ and $B(s)$ are known as anneal functions. At $s = 0$, $A(s) \gg B(s)$, while $A(s) \ll B(s)$ for $s = 1$. As we increase s from 0 to 1, anneal functions change gradually to meet these boundary conditions. In the standard quantum annealing protocol, s changes from 0 to 1. The network of qubits starts in a global superposition over all possible classical states and after $s = 1$, the system is measured in a single classical state.

The arrangement of qubits on the D-Wave chip forms a C16 Chimera graph with 16×16 unit cells (2048 qubits are mapped into a 16×16 matrix of unit cells; each unit cell has 8 qubits). Figure 1(a) shows a C3 Chimera graph with 3×3 unit cells. Within each unit cell there are two sets of 4 qubits which are connected to each other in a bipartite fashion. As shown in the figure, each qubit in a unit cell is connected to four qubits of the unit cell and two qubits of other unit cells. Thus, each qubit can be connected to a maximum of 6 qubits. This connectivity can be enhanced by forming a strong ferromagnetic couplings between the qubits, which forces coupled qubits to stay in the same state.

2.4. RBM embedding onto the D-Wave QPU

Mapping an AQC algorithm on a specific hardware is nontrivial and requires creative mapping. There are several algorithms that can be used to map a graph to the physical qubits on adiabatic quantum computer[30, 31]. However, it is nontrivial to find a simple embedding when the graph size is large. Taking into consideration the arrangement of qubits on 2000Q processor, we found a simple embedding which utilizes most of the working qubits. In the present study, we used RBMs in two configurations, one with 64 visible unit and 64 hidden units, another with 64 visible units and 20 hidden units. Here, we will discuss embedding of the RBM with 64 units in both layers.

Each unit of the RBM is connected to 64 other units, but in the D-Wave each qubit only connects to 6 other qubits. To enhance the connectivity qubits can be coupled together or cloned by setting $J_{i,j} = -1$. This forces the two qubits to stay in the same state. In our embedding one unit of RBM is formed by connecting 16 qubits. The D-Wave processor has qubits arranged in 16×16 matrix of unit cells. Each unit cell has two sets of 4 qubits arranged in bipartite fashion. Each qubit in the left column of the unit cell can be connected to one qubit of the unit cell just above it and one just below it. There are 16 unit cell along one side, so a chain of 16 qubits can be formed. This chain represents one visible unit of the RBM. Figure 1(b) shows procedure to couple 3 qubits to form a chain which represents a visible unit for a 3×3 chimera graph. Qubits that form one visible unit are shown in red forming a vertical chain. Since, there are 4 qubits in left column of the unit cell, four chains can be formed resulting in 4 visible units of RBM. Four qubits that form the right column of the unit cell can be connected to form horizontal chains, as shown in Figure 1(b). These horizontal chains form the hidden units. There are 16 unit cells along the horizontal direction, therefore each horizontal chain is also composed of 16 qubits. Exploiting the arrangement of qubits of the D-Wave QPU, 64 vertical and 64 horizontal chains can be formed representing the 64 visible and 64 hidden units of RBM. Figure 1(c) shows the scheme that we used to connect one visible unit (V1) to 12 hidden units. In this fashion one can embed a RBM with 64 visible and 64 hidden units on a C16 Chimera graph. Of course care must be taken for any inaccessible qubits to form a further restricted
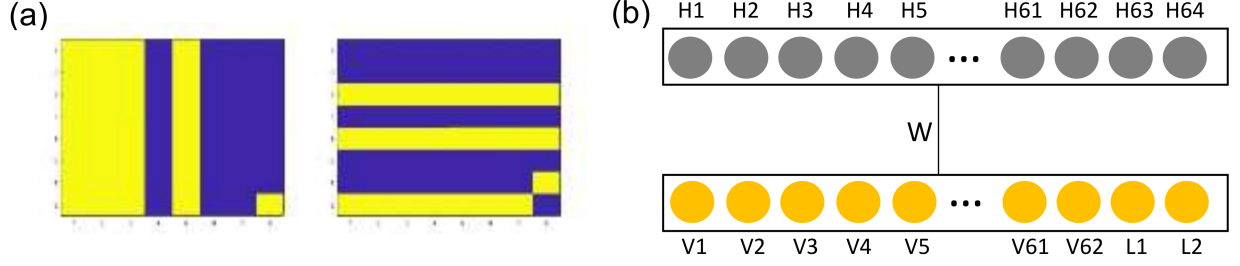
Figure 2. (a) Example of bar (left) and stripe (right) pattern of size 8×8. A blue cell represents a 0 while a yellow cell a 0. Last two bits (bottom right) are labels; bar = 01, stripe = 10. (b) RBM for classification. Box with yellow units is the visible layer. Hidden layer is shown by box with grey units. A record of size 1×64 is applied to the visible layer. Units L1 and L2 are the labels.

RBM. In our experiments, we found that absence of few qubits does not affect the performance of the resulting network. Employing this embedding we were able to connect most of the RBM units to 64 units of the opposite layer.

2.5. Classification and image reconstruction

Each record of the bar and stripes (BAS) data set has 64 bits. The last two bits are for labeling the pattern: 01 for a bar and 10 for a stripes pattern (Figure 1(a)). If the last two bits are 00 or 11, prediction by RBM is incorrect. Once we obtained the weights and biases of the RBM from the training step, RBM can be used for classification or image reconstruction. To predict the class of a test record we apply its first 62 bits at the visible layer (Figure 1(b)). For the last two classifying bits (L1 and L2), we randomly input either zero or one. We then run 50 Gibbs cycles, keeping the 62 visible units clamped at the values of the test record. At the end of 50 Gibbs cycles label units 63 and 64 are read. $L1 = 0$ and $L2 = 1$ indicates a bar pattern, while $L1 = 1$ and $L2 = 0$ suggests a stripe pattern. For the problem of image reconstruction, the goal is to predict value of missing part of the image. Similar procedure can applied for image reconstruction where a trained RBM is used to predict values of the missing units, while in the classification process one only predicts the classifying labels. In this case we clamp the visible units where value are given, run few Gibbs cycle and then sample the units where values are missing.

A different procedure is used for classification and image reconstruction using a quantum annealer. First step is to embed a trained RBM onto the D-Wave QPU. Then a field 'h' which is proportional to the test record (or corrupted image) is applied to the visible units. Finally, quantum annealing step is performed which results in predicting values of missing visible units.

3. Results and Discussion

In the present work, we have used bars and stripes (BAS) data set, which is widely used as a benchmark for training and classification using RBMs. BAS is a binary data set consisting of records 64 bit in length, with the last 2 bits representing the label of the record: 01 for a bar and 10 for a stripe pattern. Our BAS data set is comprised of 500 unique records. The number of unique samples used for training is 300, with the remaining 200 samples were held for testing.
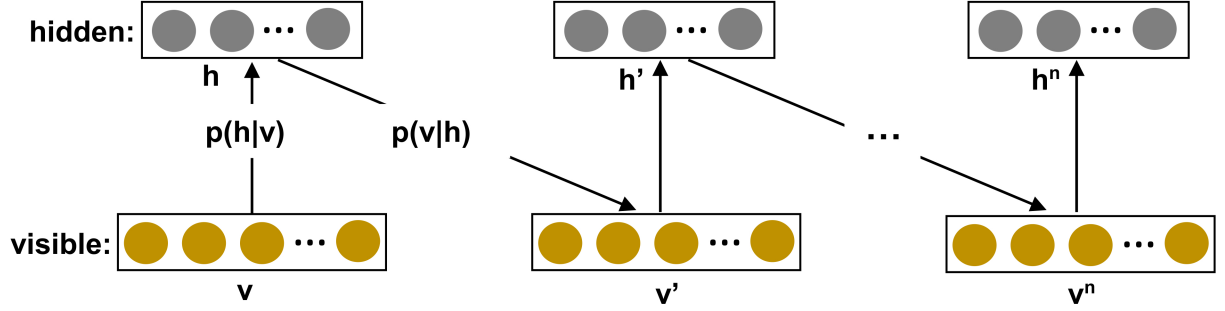
Figure 3. Illustration of the contrastive divergence algorithm for training restricted Boltzmann machines (RBMs). If sampling stops at n^{th} Gibbs step of the Markov Chain, the procedure is known as CD-n.

Classification of bars and stripes, image reconstruction and log-likelihood were used to compare the performance of trained RBMs.

3.1. RBM Training

Equation (18) has been used to train the RBMs. The first term in this equation is data dependent. This term can be exactly calculated using the conditional probabilities $P(h|v)$ and $P(v|h)$ given by equation (13) and (14). The second term is model dependent, which requires expectation value over all possible hidden h and visible v vectors, which is clearly intractable. Typically, the model dependent term is approximately estimated using a method known as contrastive divergence (CD). In this approach, samples needed to calculate the model dependent term are obtained by running Gibbs chain starting from a sample from the training data (figure 3). If n Gibbs steps are performed, method is known as CD-n. It is shown by Hinton that $n = 1$ could be sufficient for convergence (CD-1) [32]. In CD-1, first a data sample is applied at the visible layer, then equation (13) is used to generate corresponding hidden vector at the hidden layer. Now, this hidden vector is used to generate a new visible vector using equation (14), which is in turn used to generate a new hidden vector. These new visible and hidden vectors are used to calculate model expectation value. This process is repeated for each record in the data set. A detailed description of RBM training using CD is given in a review article by Hinton [33]. The model dependent term can also be calculated using samples (v and h) obtained from a RBM mapped on the D-Wave. From equation (17) we notice that in the second term expectation value should be calculated with respect to $e^{-E(v,h)}$ distribution, while samples from the D-Wave follow a distribution of $e^{-\frac{E(v,h)}{kT}}$. Following the approach used by Adachi *et al* . [20], we used a empirical parameter, S, such that for the model dependent term, we sample from $e^{\frac{-E(v,h)}{SkT}}$ distribution. Here, S is a parameter, which is determined by trial and error method. The optimal condition corresponds to the case when $SkT = 1$. A different approach was taken by Benedetti *et al* . [34]. They calculated effective temperature during each epoch. Their approach is difficult to apply in the present case of 64 bits record length BAS data set. BAS data set that they used had 16-bit records. A complex data set leads to a complicated distribution, which makes training with a variable temperature difficult.

In order to train a RBM using D-Wave, model parameters (w_{ij}, b_i and c_j) were initialized with random values, first term of equation (18) was calculated exactly using these weights and biases and the training dataset. Weights and biases were used to embed the RBM onto the D-Wave QPU, and quantum annealing was performed. Once annealing was complete, the D-
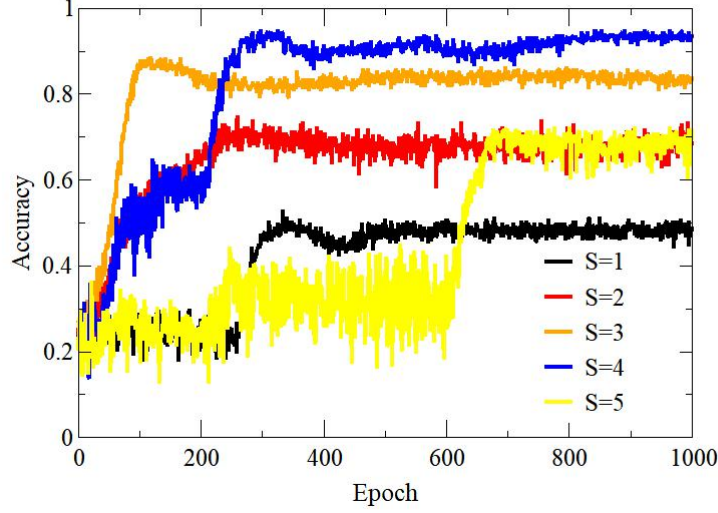
Figure 4. Classification accuracy vs. epoch graph on BAS dataset. The plot shows the effect of empirical parameter S on the accuracy. The best results were obtained for $S = 4$.

Wave returned low energy solutions. Based on the mapping of the RBM, visible v and hidden h vectors were obtained from the solutions returned from the D-Wave. These v and h samples were used to calculate the model dependent expectation value which in turn gives the gradient of log-likelihood. The gradient was further used to calculate new weights and biases. The whole process was repeated until some convergence criterion was achieved. An optimal value of the empirical parameter S is important for the correct sampling of v and h vectors. The effect of change in S on the classification accuracy is shown in figure (4), a plot between accuracy and epoch. The term epoch means a full cycle of iterations, with each training pattern participating only once. Accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{26}$$

The classification accuracy is maximum for $S = 4$. The performance of the model during the training process can be visualized by plotting classification accuracy with epochs. Figure (5) shows the plot of classification accuracy vs. epochs for bars (left) and stripes (right) patterns. This calculation was performed on the test data set. As the number of epoch increases from 0 to 400, the classification accuracy increases after that it stays constant. Based on these results, we conclude that performance of RBM trained with quantum annealing is similar to that using CD-1. However, the classification accuracy from CD-1 based training shows higher fluctuations.

3.2. Image reconstruction

For classification task both training methods (quantum annealing and CD-1) showed similar results. Classification task requires prediction of target labels based on the features in the data set. Input data record is applied at the visible layer and the target labels are reconstructed.
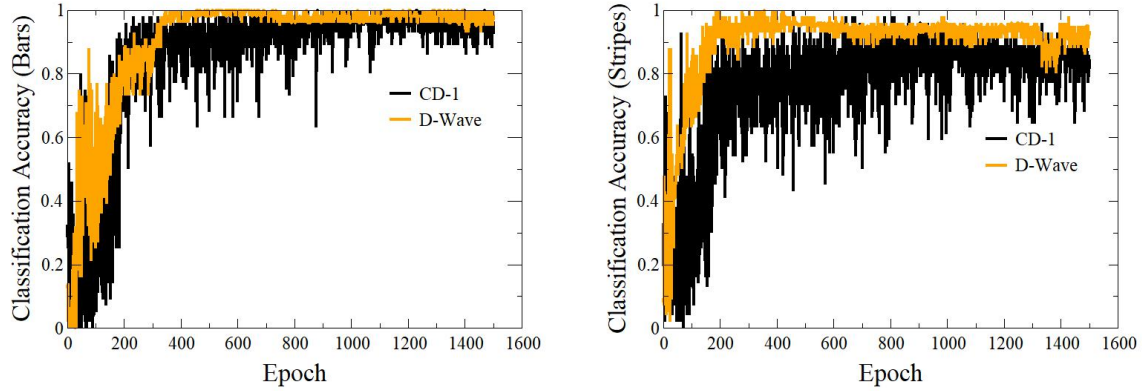
Figure 5. Plots showing classification accuracy of individual classes with epoch. Gradient of log-likelihood was calculated using samples generated via contrastive divergence and D-Wave's quantum annealing. Comparison of accuracy of both methods are presented for bars (left) and stripes (right) patterns.

More difficult task would be the reconstruction of not just the target labels, but also some other bits of the record. We call this task - image reconstruction. Here, we take a 64 bit record from the test data set, corrupt some of its bits, and then apply this modified test record to the visible layer of a trained RBM. We follow the procedure explained earlier for image reconstruction using RBM trained with CD-1 and quantum annealing. The results of image reconstruction are presented in figure (6). In figure 6(a) only target labels were corrupted/reconstructed. We notice that both training methods correctly reproduced the classifying labels. In the second case, figure 6(b), 16 bits of the original data record were corrupted. The RBM trained using CD-1 correctly predicted all the bit, while two bits were incorrectly predicted by the RBM trained with quantum annealing. In the third case, figure 6(c), a completely random 64 bit input vector (all bits corrupted) was fed to the both RBMs. In the case of CD-1, the output is a bar pattern, where as the D-Wave trained RBM resulted in a stripes pattern with many bits incorrectly predicted. From these results, we infer that the RBM trained using CD-1 performs slightly better than the RBM trained using quantum annealing. The first example of image reconstruction (figure 6(a)) where only target labels were predicted and the plot of classification accuracy vs epochs (figure (5)) indicate that it is easier to predict the classifying labels of a record compared to the prediction of other bits. Similar results were obtained (not shown here) when other records of the test dataset were investigated for image reconstructions.

3.3. Log-likelihood comparison

The classification accuracy results indicated similar performances of both methods (CD-1 and quantum annealing). However, image reconstruction suggests improved performance of CD-1. In order to further compare and quantify the performances of these two models log-likelihood of training data was calculated. Several researchers have used 'log-likelihood' in order to compare different RBM models [35, 36]. The log-likelihood has been computed using equation (16). It involves the computation of partition function Z. If number of units in the hidden layer is not too large, Z can be exactly calculated using equation (7). To calculate the log-likelihood, the number of hidden units was set to 20. The log-likelihood of both models were computed at various epochs. The results are presented in Figure 7. We notice that the log-likelihood is higher
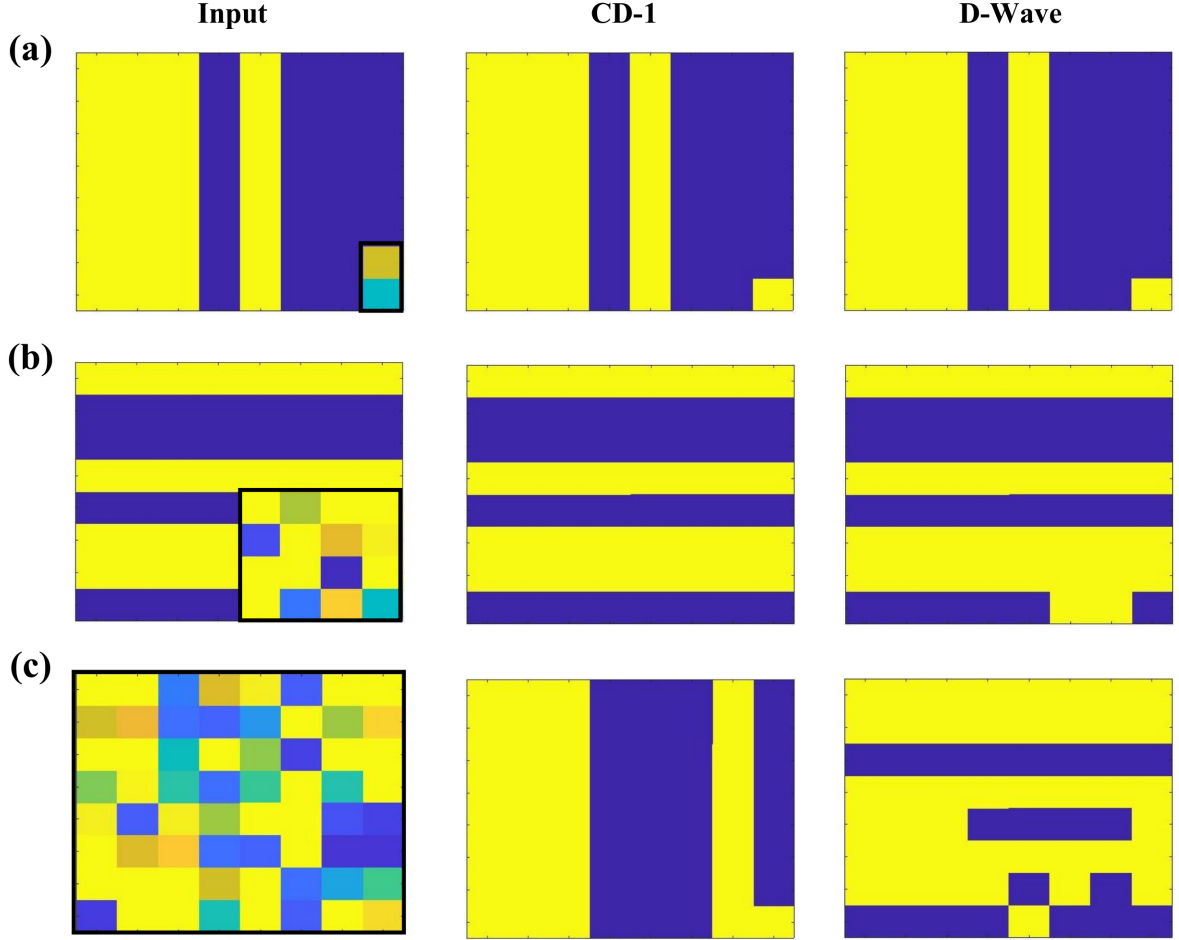
Figure 6. Image reconstruction. Original data was first corrupted, then reconstructed. The images in the left column are data fed to the RBMs; output from the RBMs trained using CD-1 and quantum annealing (D-Wave) are shown in the middle and right column, respectively. Corrupted bits in the input images are enclosed by black rectangles.

for RBM trained using CD-1 compared to quantum annealing. A lower value of the log-likelihood for the D-Wave trained model could be attributed to a restricted range of allowed values for the bias field h and the coupling coefficients J. Another reason could be an instance (each set of h and J) dependent temperature variation during the RBM training [34]. This disturbs the learning of embedded RBM. Figure 7 also compares the log-likelihood values calculated using new lower-noise D-Wave 2000Q processor and an earlier 2000Q processor. D-Wave's lower-noise machine shows slightly improved log-likelihood values over the entire training range.

4. Conclusions

In this work, we presented an embedding that can be use to embed a RBM with 64 visible and 64 hidden units. We trained a RBM by calculating the model dependent term of the log-likelihood using samples obtained from the D-Wave quantum annealer. A trained RBM was embedded onto the D-Wave QPU for classification and image reconstruction. We also showed that new lower-noise quantum processor gives improved results. The performance of the RBM was compared with a RBM trained with commonly used method called contrastive divergence
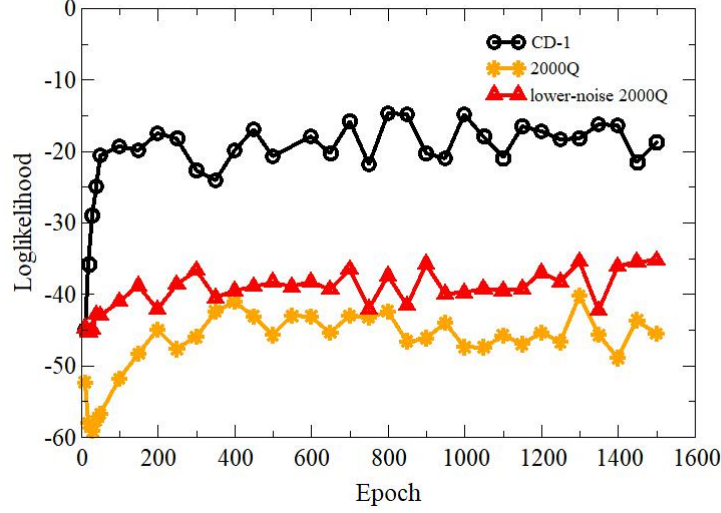
Figure 7. Plot showing variation in log-likelihood with training epochs. Label CD-1 represents RBM training employing contrastive divergence, while label 2000Q indicates training using samples are generated from D-Wave. The lower-noise 2000Q processor results in slightly improved log-likelihood

(CD-1). Though both methods resulted in comparable classification accuracy, CD-1 training resulted in a better image reconstruction and the log-likelihood values. In our future work, we will improve the RBM training using quantum annealer by finding effective ways to estimate instance dependent temperature of the quantum annealer and by incorporating this temperature in the RBM training procedure. Our present method relies on the estimation of a empirical parameter, which eliminates the need to find D-Wave QPU temperature during each iteration. RBM training using the samples from a quantum annealer removes the need time consuming MCMC steps during training and classification procedures. These computationally expensive MCMC steps are essential part of training and classification with CD-1.

References

[1] Rieffel E and Polak W 2014 *Quantum Computing: A Gentle Introduction* (MIT Press, Cambridge) ISBN 9780262526678
[2] Nielsen M A and Chuang I L 2010 *Quantum Computation and Quantum Information: 10th Anniversary Edition* (Cambridge University Press)
[3] Kais S 2014 *Quantum Information and Computation for Chemistry* (John Wiley & Sons) ISBN 9781118495667

[4] Farhi E, Goldstone J, Gutmann S and Sipser M 2000 Quantum computation by adiabatic evolution (*Preprint* `quant-ph/0001106`)

[5] Aharonov D, van Dam W, Kempe J, Landau Z, Lloyd S and Regev O 2007 *SIAM Journal on Computing* **37** 166–194 (*Preprint* `https://doi.org/10.1137/S0097539705447323`) URL `https://doi.org/10.1137/S0097539705447323`

[6] Kadowaki T and Nishimori H 1998 *Phys. Rev. E* **58**(5) 5355–5363 URL `https://link.aps.org/doi/10.1103/PhysRevE.58.5355`

[7] Finnila A, Gomez M, Sebenik C, Stenson C and Doll J 1994 *Chemical Physics Letters* **219** 343 – 348 ISSN 0009-2614 URL `http://www.sciencedirect.com/science/article/pii/0009261494001170`

[8] Santoro G E and Tosatti E 2006 *Journal of Physics A: Mathematical and General* **39** R393–R431

[9] Das A and Chakrabarti B K 2008 *Rev. Mod. Phys.* **80**(3) 1061–1081 URL `https://link.aps.org/doi/10.1103/RevModPhys.80.1061`

[10] Albash T and Lidar D A 2018 *Rev. Mod. Phys.* **90**(1) 015002 URL `https://link.aps.org/doi/10.1103/RevModPhys.90.015002`

[11] Boixo S, Ronnow T and Isakov S 2014 *Nature Physics* **10** 218224

[12] Biamonte J D and Love P J 2008 *Physical Review A* **78** ISSN 1094-1622 URL `http://dx.doi.org/10.1103/PhysRevA.78.012352`

[13] Mott A, Job J, Vlimant J R, Lidar D and Spiropulu M 2017 *Nature* **550** 375–379 URL `https://doi.org/10.1038/nature24047`

[14] Ushijima-Mwesigwa H, Negre C F A and Mniszewski S M 2017 *Proceedings of the Second International Workshop on Post Moores Era Supercomputing* PMES'17 (ACM) pp 22–29 ISBN 978-1-4503-5126-3 URL `http://doi.acm.org/10.1145/3149526.3149531`

[15] O′Malley D, Vesselinov V V, Alexandrov B S and Alexandrov L B 2018 *PLOS ONE* **13** 1–12 URL `https://doi.org/10.1371/journal.pone.0206653`

[16] Li R Y, Felice R D, Rohs R and Lidar D A 2018 *NPJ Quantum Information* **4** URL `https://doi.org/10.1038/s41534-018-0060-8`

[17] Jiang S, Britt K A, McCaskey A J, Humble T S and Kais S 2018 *Nature Scientific reports*

[18] Xia R, Bian T and Kais S 2018 *The Journal of Physical Chemistry B* **122** 3384–3395 URL `https://doi.org/10.1021/acs.jpcb.7b10371`

[19] Hinton G E 2002 *Neural Computation* **14** 1771–1800 (*Preprint* `https://doi.org/10.1162/089976602760128018`) URL `https://doi.org/10.1162/089976602760128018`

[20] Adachi S H and Henderson M P 2015 Application of quantum annealing to training of deep neural networks (*Preprint* `1510.06356`)

[21] Dumoulin V, Ian J Goodfellow A C and Bengio Y 2014 *AAAI Publications, Twenty-Eighth AAAI Conference on Artificial Intelligence* URL `https://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/viewPaper/8608`

[22] Koshka Y and Novotny M A 2018 *IEEE Transactions on Emerging Topics in Computational Intelligence* 1–11 ISSN 2471-285X

[23] Koshka Y and Novotny M A 2018 *2018 International Joint Conference on Neural Networks (IJCNN)* pp 1–8 ISSN 2161-4407

[24] Koshka Y, Perera D, Hall S and Novotny M A 2017 *Neural Computation* **29** 1815–1837

[25] Koshka Y, Perera D, Hall S and Novotny M A 2016 *2016 International Joint Conference on Neural Networks (IJCNN)* pp 1948–1954 ISSN 2161-4407

[26] Amin M H, Andriyash E, Rolfe J, Kulchytskyy B and Melko R 2018 *Phys. Rev. X* **8**(2) 021050 URL `https://link.aps.org/doi/10.1103/PhysRevX.8.021050`

[27] Lloyd S, Mohseni M and Rebentrost P 2013 Quantum algorithms for supervised and unsupervised machine learning (*Preprint* `1307.0411`)

[28] Rebentrost P, Mohseni M and Lloyd S 2014 *Phys. Rev. Lett.* **113**(13) 130503 URL `https://link.aps.org/doi/10.1103/PhysRevLett.113.130503`

[29] Wiebe N, Kapoor A and Svore K M 2014 Quantum deep learning (*Preprint* `1412.3489`)

[30] Goodrich T D, Sullivan B D and Humble T S 2018 *Quantum Information Processing* **17** URL `https://doi.org/10.1007/s11128-018-1863-4`

[31] Date P, Schuman C and Potok T 2019 *Quantum Information Processing* **18** URL `https://doi.org/10.1007/s11128-019-2236-3`

[32] Hinton G E 2002 *Neural Computation* **14** 1771–1800

[33] Hinton G E 2012 *A Practical Guide to Training Restricted Boltzmann Machines* (Springer Berlin Heidelberg) pp 599–619 ISBN 978-3-642-35289-8

[34] Benedetti M, Realpe-Gómez J, Biswas R and Perdomo-Ortiz A 2016 *Phys. Rev. A* **94**(2) 022308 URL `https://link.aps.org/doi/10.1103/PhysRevA.94.022308`

[35] Tieleman T 2008 *Proceedings of the 25th International Conference on Machine Learning* ICML '08 (New York, NY, USA: ACM) pp 1064–1071 ISBN 978-1-60558-205-4 URL `http://doi.acm.org/10.1145/1390156.1390290`

[36] Cho K, Raiko T and Ilin A 2010 *The 2010 International Joint Conference on Neural Networks (IJCNN)* pp 1–8 ISSN 1098-7576