Toward the Implementation of a Quantum RBM

Misha Denil
Department of Computer Science
University of British Columbia
Vancouver, BC
mdenil@cs.ubc.ca

Nando de Freitas
Department of Computer Science
University of British Columbia
Vancouver, BC
nando@cs.ubc.ca

Abstract

Quantum computers promise the ability to solve many types of difficult computational problems efficiently. It turns out that Boltzmann Machines are ideal candidates for implementation on a quantum computer, due to their close relationship to the Ising model from statistical physics. In this paper we describe how to use quantum hardware to train Boltzmann Machines with connections between latent units. We also describe the architecture we are targeting and discuss difficulties we face in applying the current generation of quantum computers to this hard problem.

1 Introduction

Boltzmann Machines are very general and powerful models of structure in data; however, despite being known for many years, there has been very little work done with these models due to the complexity of training [1]. For special cases of the general model (most notably the Restricted Boltzmann Machine) there are efficient training methods, and there has been extensive research into RBMs in recent years due to their power and tractability.

The key feature which makes RBMs tractable is that they are designed to have an advantageous conditional independence structure, which allows us to define efficient block Gibbs samplers to draw approximate samples from the model distribution. In contrast, the conditional distributions in a general Boltzmann Machine are quite complex (drawing exact samples is NP-hard in the worst case), which makes sampling extremely difficult.

The Boltzmann Machine is equivalent to a model from statistical physics known as the Ising model. While this is a fairly pedestrian fact mathematically (the equivalence is realized by a simple change of variables) it has important practical consequences. The Ising model is a mathematical description of a physical phenomenon, which suggests the following procedure for sampling from a Boltzmann Machine:

1. Transform the Boltzmann Machine into an Ising model.
2. Set up a physical system which realizes the transformed problem and "run the physics" to allow the system to equilibrate.
3. Measure the system to obtain a realization of states in the Ising model.
4. Transform the Ising samples into samples from the Boltzmann Machine.

This procedure transforms the difficult sampling step from a software problem into a physics problem. D-Wave Systems has developed a hardware system capable of realizing steps (2) and (3) in the above procedure [13]. The D-Wave hardware provides programmatic access to a highly specialized physical system, whose properties can be manipulated to correspond to a specified Ising model. The system exploits quantum mechanics to settle quickly into a low energy state, which can be measured

to obtain a sample from the original model. The D-Wave hardware is able to realize Ising models with complex graphical structure, which are very expensive to sample from in software.

The purpose of this paper is to explore methods of learning parameters for latent connections in Boltzmann Machines, in anticipation of realizing these algorithms on the quantum hardware. In order to maintain software tractability, we consider only small models and simple problems and focus our efforts on developing algorithms which are suitable for implementation on the D-Wave machine. We provide a description of the functionality provided by the D-Wave system, and discuss some practical difficulties we have encountered working with the current generation of hardware.

From a deep learning perspective the quantum hardware will enable us to sample from models with lateral connections. We anticipate such models will be better for many tasks than models which assume independence. This investigation is also important from a quantum computing perspective. D-Wave has demonstrated the quantum properties of a small version of their machine [13], and the big challenge now is to show that the hardware is able to solve hard problems in a more efficient way than a classical computer.

Finally, this paper also presents an efficient exact block sampling method for RBMs with chain and tree structured lateral connections.

2 Background

A Boltzmann Machine is a probabilistic graphical model defined on a complete graph of binary variables. We can partition the graph into "visible" units $\mathbf{v}$, where values are observed during training and "hidden" units $\mathbf{h}$, where values must be inferred. The probability of observing a state in the Boltzmann Machine is governed by its energy function

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{ij} W_{ij} v_i h_j - \sum_{jk} U_{jk} h_j h_k - \sum_{i\ell} L_{i\ell} v_i v_\ell$$

where $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{L}$ are matrices of parameters. This model is very general, but is not used in practice because it is very difficult to train [19]. Salakhutdinov [18] studied fully connected Boltzmann Machines and demonstrated a variational learning procedure which was effective with as many as 500 hidden units; however, applications of this model in its most general form are scarce.

An important special case of this model is the Restricted Boltzmann Machine [9], which disallows interactions between pairs of visible and pairs of hidden units. Restricting the model in this way creates conditional independence between the hidden and visible units, allowing efficient block Gibbs sampling. This model has seen great success and wide spread application in recent years.

Different restrictions of the Boltzmann Machine have also been studied. The Semi-Restricted Boltzmann Machine [16] relaxes the conditions imposed in the RBM by allowing the visible units to interact. In this model inference over the visible units is approximate; however, it has been observed [19] that training is still effective as long as the reconstructions have lower free energy than the data. While approximate inference works well for the visible units, it is important that the hidden unit states are sampled from their exact posterior [16]. This fact makes adding interactions between hidden units more difficult, since exact sampling in complex graphs is very difficult.

Schulz et al. [19] studied Restricted Boltzmann Machines with hidden interactions; however, they heavily restrict the connectivity between the visible and hidden units in order to enforce a topology in the visible layer of their model. Their model has very different structure than the ones we consider here, and likely has many different properties.

Our proposed method is surprisingly similar to work done in the early 90's on implementing neural networks in VLSI circuits [8, 3, 7, 6]. These works used a weight perturbation scheme very similar to the one we propose in this paper; however, the motivations are very different. In VLSI, weight perturbation schemes are desirable because they can be implemented more efficiently than traditional training algorithms [11]. Our use of perturbation is motivated by very different concerns, which we outline in Section 5.

3 P≠NP and why we're not crazy

Much like artificial intelligence in its early days, the reputation of quantum computing has been tarnished by grand promises and few concrete results. Talk of quantum computers is often closely flanked by promises of polynomial time solutions to NP-Hard problems and other such implausible appeals to blind optimism. In this section we attempt to placate the understandably sceptical reader by providing a brief description of the computational model on which the D-Wave machine is based, as well as some intuition for how it could help us solve hard problems even if **P≠NP**. Naturally this is far too large a topic to cover fully in this paper. More in depth discussions can be found in the references from this section, with [17] providing a brief but very accessible introduction.

The D-Wave machine is based on the idea of Adiabatic Quantum Computation. AQC is a formal model of quantum computation (in much the same way a Turing Machine is a formal model of classical computation) which has been shown to be universal [12, 2]. Intuitively, AQC works by starting with a problem which is easily solvable and then slowly transforming it into the problem of interest. If the transformation is carried out sufficiently slowly then a solution to the easy problem can be transformed into a solution to the problem of interest without ever solving the hard problem directly. Unfortunately, it can be shown that for NP-Hard problems, going "sufficiently slowly" requires exponential time [4].

The relationship between AQC and the D-Wave machine parallels the relationship between a Turning Machine and a desktop computer. A Turing Machine has an infinite tape on which to write symbols, while a desktop computer has finite memory. From this perspective we can view a Turing Machine as the "large memory" limit of a desktop computer. Analogically, AQC can be understood as the "long time" limit of the D-Wave architecture.

The D-Wave machine uses a process known as quantum annealing, which is essentially a fast, approximate version of AQC. Both procedures work by transforming the solution to an easy problem into the solution to a hard problem, but QA does this transformation quickly at the cost of not being able to guarantee an optimality. However, all is not lost, since the solutions produced by QA are not completely arbitrary but are in fact samples from a Boltzmann distribution (the shape of which is determined by the problem).

The energy landscape, which determines the probability distribution over states, is shaped so that better problem solutions have lower energy (and thus higher probability of occurring). This allows QA to be used for optimization, by selecting the lowest energy result after several runs; or for sampling, by aggregating the results of several runs into a population of samples.

Since this is essentially a quantum analog of simulated annealing one might wonder if there is any advantage using QA. However, there is experimental evidence which suggests that QA is superior to its classical counterpart [5], at least for certain types of problems.

4 Model

In this paper we consider Restricted Boltzmann Machines with two different patterns of latent connections. We first consider chain structures, which give rise to models which are simple enough that they can be simulated efficiently in software, but are also sufficiently complex that they can serve as a testbed when designing algorithms for the D-Wave hardware. We also consider a more complex graphical structure on the hidden units, which reflects a scaled down version of the connection patterns available on the D-Wave machine.

Although the hidden units in these models are no longer conditionally independent, for the chain model we can still obtain samples quickly using a backward filtering forward sampling algorithm. We can also use this algorithm to sample from general graph structures, but the complexity required to do so grows very quickly.

4.1 Backward filtering forward sampling

For simplicity, we describe the exact sampling algorithm only for the case where the hidden units have chain structured dependencies, and provide references for where the details of the general algorithm can be found.

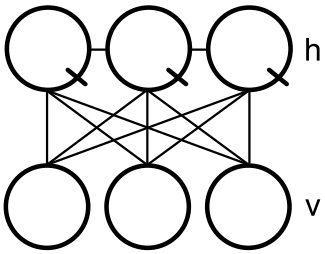

Figure 1: The model we study in this paper. The visible units are binary valued and are conditionally independent given the hiddens. The hidden units are also binary valued but have correlations in their conditional distribution. We consider both chain structured connections between the hidden units, as well as a more complex graph structure which reflects the architecture of the D-Wave machine. Units which are intended to be realized on the quantum hardware are marked with a small diagonal bar.

Suppose we have a Restricted Boltzmann Machine whose energy function is given by

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{ij} W_{ij} v_i h_j - \sum_{|j-k|<2} U_{jk} h_j h_k - \sum_i L_{ii} v_i \ ,$$

where we have allowed hidden units to interact with their neighbours (i.e. the hidden connections form a chain). Although we can still sample from $P(\mathbf{v}|\mathbf{h})$ directly, the conditional distribution over the hidden units is not so simple. The following procedure allows us to draw samples from this more complex distribution efficiently.

Because of how the hidden connections have been restricted, the conditional distribution over the hidden units factors as

$$p(\mathbf{h}|\mathbf{v}) = p(h_1 = 1|\mathbf{v}) \prod_{j=2}^{|\mathbf{h}|} p(h_j = 1|h_{1:j-1}, \mathbf{v}) = p(h_1 = 1|\mathbf{v}) \prod_{j=2}^{|\mathbf{h}|} p(h_j = 1|h_{j-1}, \mathbf{v}) \ .$$

The conditional distributions in this product are easy to sample from, the only problematic term is the marginal $p(h_1 = 1|\mathbf{v})$. If we could sample $h_1 \sim p(h_1 = 1|\mathbf{v})$ from the marginal then we could use the conditional distributions to sample sequentially from the joint.

The marginal distribution for $p(h_1 = 1)$ can be found via a backwards pass of belief propagation. Starting at the end of the chain, for each $j = |\mathbf{h}| - 1, \ldots, 1$ we compute

$$m_{j+1 \to j}(h_j) = \sum_{h_{j+1}} \phi_{j+1}(h_{j+1}) \psi_{j+1}(h_j, h_{j+1}) m_{j+2 \to j+1}(h_{j+1}) \tag{1}$$

(where $m_{|\mathbf{h}|+1 \to |\mathbf{h}|}(\cdot) \triangleq 1$). In the above expression the ϕ's are unary potentials of the conditional distribution, and the ψ's are binary potentials. The marginal distribution $p(h_1 = 1)$ is then given by

$$p(h_1 = 1) \propto \phi_1(h_1) m_{2 \to 1}(h_1)$$

and we can find the normalizing constant by summing over h_1. In our case the binary potentials in Equation 1 take the form $\psi_{j+1}(h_j, h_{j+1}) = (U_{j+1,j} + U_{j,j+1}) h_j h_{j+1}$ and the unary potentials are given by

$$\phi_j(h_j) = \sum_i W_{ij} v_i + U_{jj}$$

as in an ordinary RBM.

The backward filtering forward sampling method we have described can be extended from chains to trees in a straightforward way; however for general graph structures (which may include cycles) the method described here cannot be employed directly. For general graphs we can form a junction tree and do the backwards filtering pass there to compute the marginal distribution for one of the nodes. We can then sample from the whole graph using conditional distributions in much the same way as we have described here. The transformation to a junction tree is non-trivial, but is a fairly standard operation, and we refer the interested reader to [10, 14] for the details. Unfortunately, in this setting the backward filtering step has complexity which is exponential in the tree width of the graph, so in practice we are restricted to either very small graphs or very simple connectivity patterns.

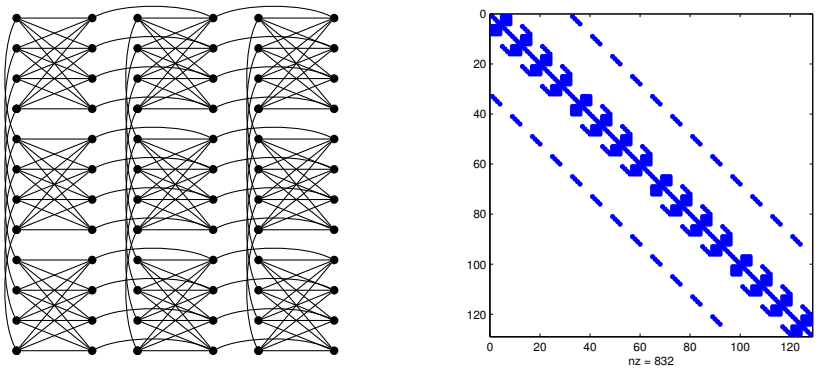

Figure 2: Some examples of Chimera graphs. **Left:** A line diagram of a Chimera(3,3,4) graph showing the connections between qubits. **Right:** The connectivity matrix for a Chimera(4,4,4) graph. For a general Chimera(M, N, L) graph the connectivity matrix is given by

$$\mathbf{A} = \mathbf{I}_M \otimes \mathbf{I}_N \otimes \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \otimes \mathbf{1}_L + \mathbf{L}_M \otimes \mathbf{I}_N \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \mathbf{I}_L + \mathbf{I}_M \otimes \mathbf{L}_N \otimes \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \mathbf{I}_L$$

where $\otimes$ is the Kronecker product, $\mathbf{I}_n$ is the $n \times n$ identity matrix, $\mathbf{1}_n$ is the $n \times n$ matrix of 1's and $\mathbf{L}_n$ is the adjacency matrix for a chain of length n.

4.2 Chimera graphs

Although the D-Wave hardware is not able to realize a fully connected Ising model it is capable of handling graph structures more general than chains or trees. The architecture is able to realize structures from a family of graphs called Chimeras. Chimera graphs are built by interconnecting a two dimensional grid of small bipartite graphs. This family of graphs can be described succinctly by three parameters; a Chimera(M, N, L) graph is assembled from an $M \times N$ grid of $L \times L$ dense bipartite graphs, connected as shown in Figure 2.

Although the Chimera graphs are sparse, they are non-planar and are of moderately high tree width (a Chimera(N, M, L) graph has tree width $L \min(M, N)$), which makes them difficult to sample from in software. The current version of the D-Wave hardware implements a Chimera(4,4,4) structure, and future generations of hardware will provide larger graphs.

5 Parameter Warping

The D-Wave hardware enables us to draw samples from graph structures which would be otherwise intractable, but there are still some practical difficulties to overcome. This is not surprising, since the hardware is still very new. We expect these difficulties to be resolved in time, but for the moment they are a serious concern.

The most difficult problem we have encountered is something we call parameter warping. In order to use the D-Wave hardware to draw samples, it is necessary to program the machine with parameters for the desired distribution. Currently there is a systematic warping phenomenon which occurs when we program the machine. This warping perturbs the parameters from their desired settings, so that the distribution realized by the hardware is actually not the distribution we specify, but is instead a nearby distribution whose parameters are slightly different. Concretely, given a distribution $P(\cdot|\boldsymbol{\theta})$ from which we would like to draw samples, realizing this distribution in hardware produces samples from the nearby distribution $P(\cdot|W(\boldsymbol{\theta}))$ for some unknown, systematic warping function W.

Computing the maximum likelihood gradients in an RBM requires evaluating expressions of the form

$$\mathbb{E}_{\mathcal{D}}\left[\frac{\partial}{\partial \boldsymbol{\theta}} \log P(\mathcal{D}|\boldsymbol{\theta})\right] = -\mathbb{E}_{\mathcal{D}}\left[\frac{\partial}{\partial \boldsymbol{\theta}} E(\boldsymbol{\theta})\right] + \mathbb{E}_{\mathcal{M}}\left[\frac{\partial}{\partial \boldsymbol{\theta}} E(\boldsymbol{\theta})\right] \tag{2}$$

where the first expectation on the right hand side is taken over the data distribution and the second expectation is over the distribution defined by the model. In our case, parameter warping replaces the $E(\boldsymbol{\theta})$ terms in the above expression with $E(W(\boldsymbol{\theta}))$. If W were known then this would pose no obstacle, since we could simply expand the gradients in Equation 2 using the chain rule and proceed as before. Unfortunately, this is not the case.

Although we do not have a good model for the warping function W, we do have some idea of its general structure. The warping appears to be caused by interactions between nearby parameter realizations in the hardware. This makes the warping function local, so that parameters whose realizations are physically separate on the hardware do not interact in the warping function. On the other hand, parameters which do interact appear to do so in a non-linear way, which makes building a model of W difficult.

A possible approach to this problem is to estimate W for many parameter values and build a regression model, but this is a poor solution since W is a very high dimensional function ($W : \mathbb{R}^{832} \rightarrow \mathbb{R}^{832}$ on the current generation of hardware), so the model is likely to be poor. Additionally, even if we could estimate W to high accuracy, the relevant quantity which affects our optimization is the gradient of W, and even a very good model for the value of W is unlikely to provide a good model for its gradients.

It turns out that not knowing W makes this problem quite difficult, since it precludes the use of many standard techniques. For instance, one might try to take advantage of the fact that the distance between $P(\cdot|\boldsymbol{\theta})$ and $P(\cdot|W(\boldsymbol{\theta}))$ should be fairly small in order to do importance or rejection sampling with $P(\cdot|W(\boldsymbol{\theta}))$ as a proposal for $P(\cdot|\boldsymbol{\theta})$. Unfortunately, computing the importance weights for this procedure requires the ability to evaluate W, which we do not have.

In order to overcome this problem, we have designed a procedure which optimizes the connection strengths between hidden units as a black box function. We cannot use the maximum likelihood objective, for the reasons we have outlined above, but we can instead adjust the latent connection strengths to optimize one-step reconstruction error while training the visible-hidden connection weights with the ordinary rule. Exact gradients of this objective still require knowledge of W, but as we demonstrate, we can optimize this objective by treating the reconstruction error as a black box function of $\boldsymbol{\theta}$ and approximating the gradients empirically. Computing empirical gradients requires many evaluations of the objective function, making it time consuming in high dimensions. We employ a stochastic approximation to finite differencing to reduce the computational cost to a manageable level.

Unfortunately, under our procedure we never recover the parameters of the model we are using. The result of our approach is a vector $\boldsymbol{\theta}$ such that the warped vector $W(\boldsymbol{\theta})$ contains the parameters of our model, which means we cannot decouple our model from the machine.

5.1 Simultaneous Perturbation Stochastic Approximation

Simultaneous Perturbation Stochastic Approximation (SPSA) is an algorithm for approximate gradient based optimization of noisy, differentiable, black box functions [20]. SPSA requires that the objective be differentiable, but unlike traditional stochastic gradient methods it does not require explicit access to the objective gradient.

SPSA estimates the objective gradient at each step using a stochastic variant of the finite difference approximation. Forming the ordinary finite difference approximation is expensive in high dimensions, since it requires $2d$ evaluations of the objective (where d is the dimensionality of the problem). In contrast, SPSA is able to function with exactly 2 objective evaluations at each step, regardless of the dimensionality.

Formally, given an objective function $J(\boldsymbol{\theta})$ and an initial value $\boldsymbol{\theta}_0$, at time t during the optimization SPSA preforms the update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - a_t g_t(\boldsymbol{\theta}_t) \ ,$$

where $g_t(\boldsymbol{\theta}_t)$ is a stochastic estimate of the gradient of $J(\boldsymbol{\theta}_t)$ given by

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}_t) \approx g_t(\boldsymbol{\theta}_t) = \frac{1}{2c_t}(J(\boldsymbol{\theta}_t + c_t\boldsymbol{\Delta}_t) - J(\boldsymbol{\theta}_t - c_t\boldsymbol{\Delta}_t))\boldsymbol{\Delta}_t^{-1} \ .$$

In the above equation, $\mathbf{\Delta}_t^{-1}$ denotes the elementwise inverse of the vector $\mathbf{\Delta}_t$, which is chosen randomly at each time step from a suitable distribution. Many distributions are appropriate here (although some obvious choices such as Gaussian or uniform are not valid), but the simplest choice is to sample the elements of $\mathbf{\Delta}_t$ independently from $\{-1, 1\}$ with equal probability.

The optimization is controlled by two gain sequences a_t and c_t, good selection of which is critical for the optimization to preform well.[1] It can be shown that, given some reasonably general conditions, SPSA is able to achieve the same level of accuracy as ordinary finite difference methods for a fixed number of iterations, even if the objective function is noisy.

6 Experiments

In this section we describe the results of two experiments designed to test the procedure outlined in the previous sections. The results demonstrate that our proposal to adjust the hidden connection weights to minimize reconstruction error can be applied successfully in software.

We trained four different RBM models with varying latent structure:

1. The *vanilla* model is an ordinary RBM trained using stochastic maximum likelihood, which we include here as a reference model.
2. The *chain grad* model is an RBM with chain structured connections between the hidden units. We train all the parameters in this model using stochastic maximum likelihood.
3. The *chain spsa* model is again an RBM with chain structured hidden connections; however, the latent connections in this model are trained to optimize reconstruction error using SPSA as described in Section 5.1. The visible-hidden connections are trained using stochastic maximum likelihood.
4. The *chimera spsa* model is trained using the same procedure as in *chain spsa*, but the latent connections in this model are connected in a Chimera(3,3,3) pattern.

In the interest of consistency, all the models we consider here have 54 hidden units. We have restricted ourselves to small models in order to make software sampling in the Chimera graph feasible. Even with only 54 units, training the *chimera spsa* model in software takes many hours.

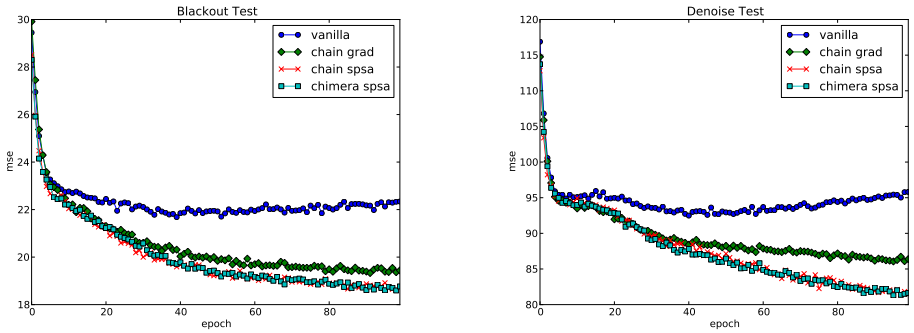

Figure 3: **Left:** One step reconstruction error for filling in a partially blacked out image during training. The *vanilla* model is an ordinary RBM with no special latent structure, which we show here for comparison. The *chain grad* model is an RBM with chain structured latent connections trained using the stochastic maximum likelihood gradient. The *chain spsa* model has the same structure as *chain grad*, but the latent connections are trained to minimize reconstruction error using SPSA. Finally, the *chimera spsa* model is also trained to minimize reconstruction error, but the hidden units are connected in a Chimera(3,3,3) pattern. All of the models shown here have 54 hidden units. **Right:** Reconstruction error for the denoising task during training. Each test image was corrupted with 20% random Bernoulli noise and the one step reconstruction error is shown. The models in this figure are the same as in the left panel.

[1] See [20] for a discussion on how to set these sequences.

We train our models using the 16×16 pixel version of the Caltech 101 Silhouettes[2] data set [15]. This data set is a binarized version of Caltech 101, where a silhouette of the target object in each of the original images has been rendered in solid black on a white background. We use the provided train and test split and do no additional pre-processing before learning.

The left plot in Figure 3 shows squared reconstruction error of the test set for each of our models as training progresses. To produce this figure we blacked out a 64 pixel region from each of the test images and compute one step reconstruction error in the blacked out region. From this plot we can see that the models with latent connections are able to achieve lower error rates than the baseline RBM. Importantly, the performance of the *chain grad* model is much better than the performance of the *vanilla* RBM. This demonstrates how latent structure can proved benefits on even this simple problem.

The right plot in Figure 3 shows a different experiment with the same set of models. For this experiment we randomly corrupted the test images with Bernoulli noise and computed the reconstruction error over the entire image. Performance of the various models is similar to their performance in the blackout experiment.

It is interesting that in both experiments *chimera spsa* is not able to achieve a lower error rate than *chain spsa*, despite being a more flexible model; however, the tasks we have considered are fairly simple, and our models are quite small, so it is likely that the additional hidden connections are simply not beneficial in this setting.

It is comforting that the *spsa* models achieve the lowest error rate in both experiments, but one should be careful not to assign too much weight to this fact. The *grad* models are trained using stochastic maximum likelihood for all connections, which maximizes the probability of observing the training data under the model. In contrast, the objective used in the *spsa* models optimizes the latent connection terms to explicitly minimize reconstruction error. In that sense our comparison between the *grad* and *spsa* models in Figure 3 is unfair; however, our intention is not to claim that the SPSA objective is a superior method for training RBMs in general, but rather to demonstrate that our objective produces reasonable results on some common tasks.

7 Conclusion

In this paper we described how the quantum computer developed by D-Wave Systems can be used as a tool to train Boltzmann Machines. The ability of the hardware to sample from complex Ising models presents exciting opportunities for training Boltzmann Machines with more connections than is typically seen in applications. We described a parameter warping phenomenon which is present in the current generation of hardware, and discussed the implications it has for using the D-Wave hardware as a tool for training deep models.

To deal with the parameter warping issue we have designed a procedure for tuning the latent connections in a Boltzmann Machine to minimize reconstruction error on the training set. Our optimization method works in this setting since it is able to operate without knowledge of the relationship between parameter and objective values. We have demonstrated that our procedure is effective in software, and we hope extend this demonstration to training RBMs directly on the quantum hardware in the near future.

Acknowledgments

We would like to thank Firas Hamze and Ziyu Wang for their helpful input. This work was supported by CIFAR-NCAP and a MITACS/D-Wave Systems grant.

References

[1] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A Learning Algorithm for Boltzmann Machines. *Cognitive science*, 9(1):147–169, 1985.

[2] Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SCIAM Review*, 2008. `http://dx.doi.org/10.1137/080734479`.

[2] `http://www.cs.umass.edu/~marlin/data.shtml`

[3] Joshua Alspector, Ronny Meir, B. Yuhas, A. Jayakumar, and D. Lippe. A parallel gradient descent method for learning in analog VLSI neural networks. In *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, pages 836–844, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[4] M. H. S. Amin. Effect of local minima on adiabatic quantum optimization. *arXiv*, 2008. arXiv:0709.0528v2 [quant-ph].

[5] J. Brooke, D. Bitko, G. Aeppli, et al. Quantum annealing of a disordered magnet. *Science*, 284(5415):779, 1999.

[6] Gert Cauwenberghs. A fast stochastic error-descent algorithm for supervised learning and optimization. In *In*, pages 244–251. Morgan Kaufmann, 1993.

[7] Gert Cauwenberghs. A learning analog neural network chip with continuous-time recurrent dynamics. In *In*, pages 858–865. Morgan Kaufmann Publishers, 1994.

[8] A. Dembo and T. Kailath. Model-free distributed learning. *Neural Networks, IEEE Transactions on*, 1(1):58 –70, mar 1990.

[9] G. E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 1554:1527–1554, 2006.

[10] Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15:225–263, 1996.

[11] M. Jabri and B. Flower. Weight perturbation: an optimal architecture and learning technique for analog vlsi feedforward and recurrent multilayer networks. *Neural Networks, IEEE Transactions on*, 3(1):154 –157, jan 1992.

[12] Peter J. Love Jacob D. Biamonte. Realizable hamiltonians for universal adiabatic quantum computers. *arXiv*, 2011. arXiv:0704.1287v2 [quant-ph].

[13] MW Johnson, MHS Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, AJ Berkley, J. Johansson, P. Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.

[14] D. Koller and N. Friedman. *Probabilistic graphical models: principles and techniques*. The MIT Press, 2009.

[15] B.M. Marlin, K. Swersky, B. Chen, and N. de Freitas. Inductive principles for restricted boltzmann machine learning. In *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS10)*, volume 9, pages 509–516, 2009.

[16] Simon Osindero and G. E. Hinton. Modeling image patches with a directed hierarchy of markov random elds. *Neural Networks*, pages 1–8, 2008.

[17] Geordie Rose and William Macready. An Introduction to Quantum Annealing. Technical report, D-Wave Systems, 2007. http://dwave.files.wordpress.com/2007/08/20070810_d-wave_quantum_annealing.pdf.

[18] Ruslan Salakhutdinov. Learning and evaluating boltzmann machines. Technical report, University of Toronto, 2008.

[19] Hannes Schulz, M Andreas, and Sven Behnke. Exploiting Local Structure in Stacked Boltzmann Machines. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2010.

[20] J.C. Spall. *Introduction to stochastic search and optimization: estimation, simulation, and control*. John Wiley & Sons, 2003.