VICTORIA UNIVERSITY OF WELLINGTON
Te Whare Wānanga o te Ūpoko o te Ika a Māui

School of Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

PO Box 600
Wellington
New Zealand

Tel: +64 4 463 5341
Fax: +64 4 463 5045
Internet: office@ecs.vuw.ac.nz

Richer Restricted Boltzmann Machines

Max Godfrey

Supervisors: Marcus Frean

Submitted in partial fulfilment of the requirements for
Bachelor of Engineering with Honours.

Abstract

TODO WRITE THE ABSTRACT TODO WORDING I or We????

Acknowledgments

Any acknowledgments should go in here, between the title page and the table of contents. The acknowledgments do not form a proper chapter, and so don't get a number or appear in the table of contents.

Contents

Figures

Chapter 1

Introduction

1.1 A Problem

Consider an image of a face. At least two systems are at play in the image of a face, the face itself and the illumination. Both face and illumination are complex and can vary greatly, but they are fundamentally acting independently of each other up until they compose to form the image. A form of Deep Nueral Networks, Deep Belief Networks, have achieved state of the art performance in facial recognition, but this is only possible with a large amount of training data. This suggests that these networks are missing a way to represent these *sources* independently. This project takes steps toward representing complex causes separately with the primary task of decomposing joint images (data).

Separating faces and illumination is too challenging for this project, there is no way to verify/evaluate that the new approach is working as the concept of a face without illumination cannot be visualised. Instead I will start with the modest task of working with images where the source images being combined are known.

1.2 Deep Belief Networks can achieve state of the art performance

Deep Belief networks (DBNs) are powerful models that have proven to achieve state of the art performance in tasks such as image classification, dimensionality reduction, natural language recognition, Document classification, Semantic Analysis.TODO CITE: Despite a DBN's expressiveness, there is no way to extract independant sources, the model instead learns how to represent the complex combination. The complex combination of sources is inheriently richer than the individual sources acting alone. The DBN may learn features that correspond to each source during it's training process, however the architecture or training algorithm make no attempt to enforce this.

DBNs are built of shallow networks called Restricted Boltzmann Machines.

1.3 A Proposed Solution and Contributions

Frean and Marsland propose an alternative architecture to that of an RBM, that aims to encode two complex sources independantly. Frean and Marsland also propose an algorithm to put this alternative architecture to use.

This project contributes:

- The first articulation of the architecture, presenting the context needed to understand the new architecture.

- A suite of graded evaluations/tests that explore how the architecture and source separation algorithm work in practice.

The evaluations will not address separating faces and illumination, instead only performing tasks such as separating two handwritten digits composed on each other (see figure 1.1).

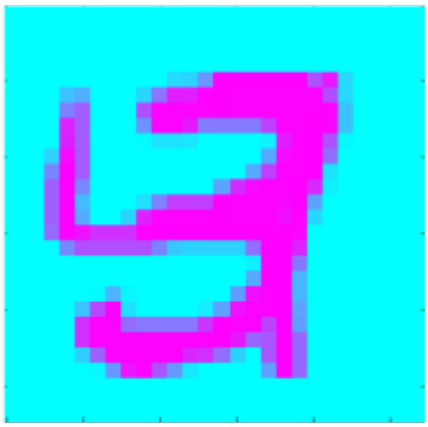

Figure 1.1: An example composition of a handwritten 4 and 3, illustrating a non-trival task where the ground truth is known.

Chapter 2

Backgroud

As the proposed architecture and algorithm extend existing work on RBMs, a substantial amount of background is required culminating with the new approach being derived. An robust understanding of these concepts is required for this project to implement and design appropriate evaluations for the proposed architecture new approach.

2.1 Generative Models

This project works with nueral network based generative models. Generative models are a powerful way to model data. The rational behind them being that we aim to learn a model that can both create the training data represent it, and reconstruct it using it's learned representation. Generative models can map input data from raw values to higher level features. Hinton gave a compelling argument why higher level features are desirable in the context of generative models [3].

> Consider, for example, a set of images of a dog. Latent variables such as the position, size, shape and color of the dog are a good way of explaining the complicated, higher-order correlations between the individual pixel intensities, and some of these latent variables are very good predictors of the class label.

Generative models model a distribution over a collection binary variables X, where X is comprised of variables which can be observed or unobserved. The observed variables are referred to as `visible` variables or units (v). Conversely, unobserved variables correspond to the hidden units of the neural network (h). With these terms defined the joint distribution that generative models model can be expressed as $P(X)$ where X is comprised of h, v. Collections of these units, are often referred to as 'patterns' or 'vectors' in that they are represented by a vector or pattern of bits. For instance in the context of an image, a visible pattern is the pixels of the image raveled into a one dimensional vector.

2.2 Directed PGMs

A Directed PGM, or in full, a Directed Probabilistic Graphical Model (DPGMs), is a language for reasoning about generative models where connections between units express causation [11]. They provide an expressive way to represent a collection of related, stochastic variables. See figure 2.1 for a simple, abstract example where variable A is dependent on B. As a result this network often referred to as a Belief Network or Bayesian Network where it causal dependencies are expressed as a conditional probability table. For example in figure 2.1 the

probabilities of A being in a given state are dependent on B, and as a result the joint distrubution over A and B is $P(A,B) = P(A|B)P(B)$.

Given X, the variables in the generative model, and $parent_i$ the parent unit of x_i, the distribution over X in a directed PGM is defined by the following factorisation:

$$P(X) = \prod_i P(x_i|parent_i)$$

Note that a normalisation is not needed in DPGMs as the conditional probabilities enforce a value between 0 and 1.

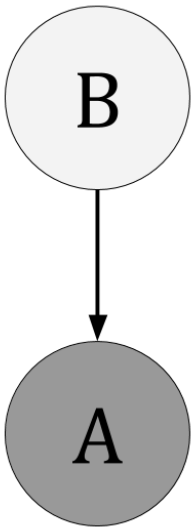

Figure 2.1: An example PGM, showing an observed variable 'A' and it's hidden cause 'B'.

2.2.1 Explaining Away in DPGMs

A common task in generative models is given a known parent unit (a cause), is inferring the probability of children variable dependent on that parent being in a given state. In directed PGMs this proves trivial to calculate. The opposite task of inferring the state of causes, given the effect of that cause is also a desirable taskTODO CITE: . It becomes problematic in DPGMs as these causal relationships gives rise to the effect of 'Explaining Away'. The canonical example `Burglar, Earthquake, Alarm Problem` is a exemplifies this effect effect[1] and is illustrated in figure 2.2. Knowledge of the state of the `alarm` makes `burglar` and

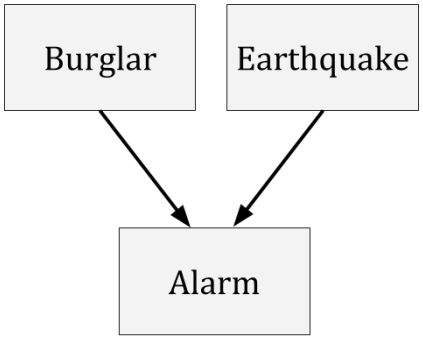

Figure 2.2: The famous Burglar, Earthquake, Alarm network showing a minimal case of explaining away.

`earthquake` dependent. The `alarm` is the observable variable here (v) and the `burglar` and `earthquake` are the hidden 'causes' (h). For example if the `alarm` is true, and we see news of earthquake in the area, our belief that we have been burgled decreases. Expressed in probabilities where A, B and E are the states of `alarm`,`burglar` and `earthquake` respectively:

$$P(A,B,E) = P(A|B,E)P(B)P(E)$$

2.2.2 DPGMs in Neural Networks: The Sigmoid Belief Network

A Belief network can be expressed as a neural network, where conditional probabilities are parameterised as weights. This network is called a Sigmoid Belief Network (SBN) as the

probability of a variable x_i that is dependent on a ancestor variable $parent_i$ the weighted sum into x_i, ϕ_i passed through the sigmoid function ($\sigma(x) = 1/(1+e^{-x})$). This is equivalent to a perceptron using a sigmoid activation function and ensures that the output is a valid probability (between 0 and 1). SBNs take a naive approach to causes, where each hidden unit represent a single, simple cause. Formally, ϕ_i is

$$\phi_i = \sum_{j \in parent_i} W_{ij} x_j$$

and the factorisation of a DPGM is defined as:

$$P(x_i | parent_i) = \sigma(\phi_i)$$

2.3 Undirected PGMs:

Unlike DPGMs Undirected PGMs (UPGMs) do not represent causation, instead capturing a dependency between two units. These pairwise dependencies change the structure of the factorisation, requiring a normalisation constant Z, a factor Φ between two variables x_i, x_j, resulting in the factorisation:

$$P(X) = \frac{1}{Z} \prod_i \Phi(x_i, x_j)$$

The introduction of the normalisation Z (often referred to as the partition function) adds nontrival complexity to the model as to compute Z, a sum over all configurations of x_i and x_j is required for all i and j.

As UPGMs do not capture causal data, calculating the state of a variable given another is no longer hampered by the effect of explaining away, however their reccurent structure, while expressive introduces an intractability in practice.

2.3.1 UPGMs in Neural Networks: The Boltzmann Machine

A UPGM expressed as nueral network is referred as Boltzmann Machine or Markov Field, where connections encode dependancies with an associated weight. We see this where W_{ij} is the weight between variables x_i and x_j the factor Φ is expressed as:

$$\Phi = e^{x_i, x_j, W_{ij}}$$

The Boltzmann machine has proposed in various forms, from different domains throughout the years, for instance it was presented in a non-stochastic context of the Hopfield network in [6]. Hinton and Sejnowski also proposed the Boltzmann machine in [5]. An example Boltzmann Machine is shown in figure 2.3. As shown in this figure 2.3, the Boltzmann Machine can be recurrent, expressing complex dependencies between variables. This recurrence makes inferring the state of a subset variables based on knowledge of another subset non-trivial as the size of the network grows to be practical TODO CITE: .

2.3.2 Restricted Boltzmann Machines

A Boltzmann Machine's architecture can be altered to alleviate inference shortcoming. The restriction, originally proposed by [12], and then later revitalised with a training algorithm that operates on the deeper architecture of the DBN [5]. The restriction requires the network to be a two layer bipartite network, each layer corresponding to the observed (visible) and latent (hidden) units. Connections are forbidden between the layer of hidden units and the

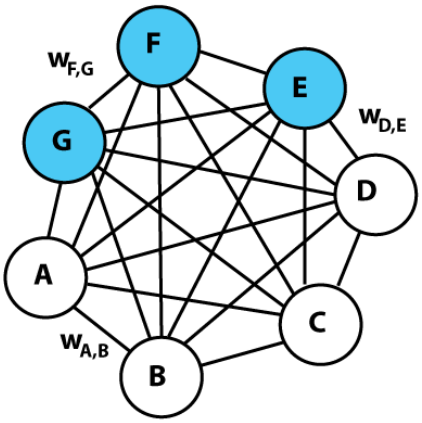

Figure 2.3: A Boltzmann Machine, the blue shaded nodes representing the observed variables, and the non-shaded nodes the latent variables.

layer of visible units respectively. An example Restricted Boltzmann Machine architecture is shown in figure 2.4. The collection of hidden units, forming a layer are referred to as the `hidden layer`. The collection of visible units are referred to as the `visible layer`.

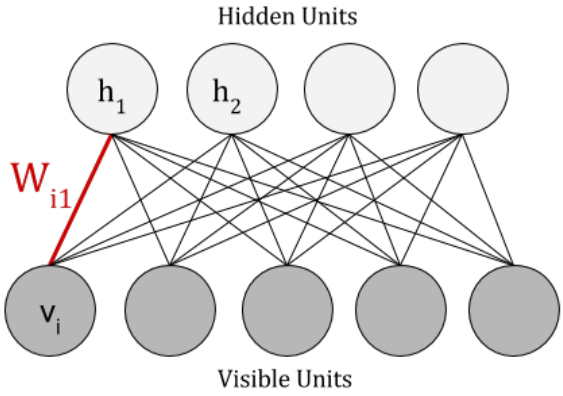

Figure 2.4: An example Restricted Boltzmann Machine with four hidden units, and five visible units. Note that the edges between units are not directed - representing a dependency not a cause.

2.3.3 Energy, and the log likelihood of the joint

An RBM models the joint distribution of hidden and visible states. The RBM assigns to every configuration of h and v an `Energy`, where the lower the energy, the more likely the RBMs configuration is to *fall* into that state. Hopfield, in the context of what is now called the Boltzmann Machine [6], presented this energy as defined by the function as:

$$E(v,h) = -\sum_{i \in visible} W_{0i} v_i - \sum_{j \in hidden} W_{0j} h_j - \sum_{i,j} v_i h_j W_{ji}$$

The probability of the RBM being in a given configuration is the joint probability of h and v.

$$P(h, v) = \frac{1}{Z} \prod_{j,i} e^{h_j W_{ji} v_i}$$

Taking logs this becomes:

$$\log P(h, v) = \frac{1}{Z} \log \sum_{j,i} h_j W_{ji} v_i$$

$\frac{1}{Z}$ is the partition function, which normalises the probability of the joint. Calculating this would require summing over all possible configurations of h and v, which is intractable for practical numbers of units. For instance a 28 by 28 image corresponds to 784 visible units, and for, say 10 hidden units this would amount to $2^{784} * 2^{10}$ possible configurations. We opt to work in terms of $P^\star$ which is the non-normalised probability of the joint over h and v. So we arrive at

$$\log P^\star(h, v) = \sum_i \sum_j e^{h_j v_i W_{ji}} \tag{2.1}$$

2.4 Sampling in Generative Models

2.4.1 Why sampling is important

Sampling is the process of drawing samples from a distribution. It is used when the distribution we want samples from is intractable to calculate analytically. As mentioned in 2.1, the power of generative models is their ability to represent, reconstruct and be trained on data. These tasks all require sampling from configurations of the hidden and visible units, often conditioned one or the other — for instance sampling from $P(h|v)$ allows a hidden representation to be created from a given input.

This is required to train generative models, as often the gradient to be climbed/descended involves calculating a probability over all the units in the generative model. Training a nueral network based generative model involves calculating a weight update, which in turn requires inferring a hidden representation given a training item. The converse, sampling from a $P(v|h)$ is also required. TODO CITE: EM

2.4.2 The sampling technique: Gibbs Sampling

Gibbs sampling is a special case of Markov Chain Monte Carlo [2], a technique for drawing sampling from a complex distribution. Sampling from the probability mass (or 'joint distribution') of a generative model is a common use case for Gibbs sampling [10].

Gibbs sampling explores the desired probability distribution, taking samples of that distribution's state, allowing iterations of exploration between drawing of a sample to ensure that the samples are independentTODO CITE: . The process of taking a step between states is referred to as a `Gibbs iteration` or a `Gibbs Step`. Formally the algorithm is descibed in algorithm 1.

Mixing Time

MCMC methods aim to approximate a distribution, by exploring likely states. As we often start this process from a random state, it's important that enough Gibbs steps are taken before a sample is drawn. This is because the random state may not be close any part of the true distribution we want to sample from, so by running the chain for many iterations we increase the likelihood of samples being from the desired distribution.

Data: A vector x indexed by j.
Result: Gibbs sampling algorithm
Let $x_{\setminus j}$ be all components that make up x vector except x_j;
initialization, begin with x, we are going to get a sample x';
for *k many iterations* **do**
  for *each component in* x, x_j **do**
    Draw a sample, x'_j from $P(x_j|x_{\setminus j})$;
    Update the current value of x_j in x with x'_j;
  end
end

Algorithm 1: The Gibbs Sampling Algorithm

This process of waiting for enough steps to before drawing samples is referred to as the `Mixing Time`. When Hinton when proposed a fast training for algorithm for RBMs and DBNs [4], Gibbs sampling is used for performing inference in RBMs and as result also in the ORBM. The `mixing time`, that is how many Gibbs iterations are needed to reach a satisfactory sample is an important part issue in the ORBM, in that one Gibbs step was sufficient in practice for training and using an RBM. The new generative model is not so fortunate.

2.4.3 Gibbs Sampling in a Sigmoid Belief Network

The dependance in belief networks means that sampling from the network requires a longer Markov Chain to mix, as changing the value of `Earthquake`, effects the value of `Burglar`. Formally, Gibbs sampling in a SBN, where our visible node is observed is expressed by:

$$P(h_1 = 1|v = 1) = \left[1 + \frac{(1 - f(b_1))f(w_2)}{f(b_1)f(w_1 + w_2)}\right]^{-1}$$

In a network with many connected nodes the dependence introduced makes sampling take longer. In the context of images, where there may be upwards of 1000 observable values, all with different dependancies this becomes intractable. Neal showed this by comparing the number of gibbs iterations required for small enough error rates in [8].

2.4.4 Gibbs Sampling in a Boltzmann Machine

Performing Gibbs sampling appears trivial in a Boltzmann Machine, in that to find the probability of a given unit being active a weighted input to that node is passed through a sigmoid function. However, in practice the recurrent nature of Boltzmann Machines makes sampling intractable as updating a node will change the probabilities of those connected. However, it was shown that given unlimited training time Boltzmann Machines could be trained, out performing the state of the art models of the time TODO CITE: This.

Recall that $x_{\setminus j}$ be all components that make up x vector except x_j and that a Boltzmann Machine has symmetric weights ($W_{ji} = W_{ij}$),

$$P(x_j = 1, x_{\setminus j}) = \frac{1}{1 + e^{-\sum_i w_{ji} x_i}}$$

That is, Gibbs sampling in a Boltzmann Machine amounts to use the Sigmoid function of the weighted inputs.

TODO CITE: neal1992:connectionist

2.4.5 Gibbs Sampling in RBMs

In order to describe Gibbs sampling in the new architecture proposed, it must first be explained for a standard RBM — The process of Gibbs sampling is as follows:

- One must sample from $P(h|v)$ giving a hidden state $\tilde{h}'$
- Using this hidden state, a visible state is then generated, $\tilde{v}'$, by sampling from $P(\tilde{v}'|\tilde{h}')$. This process of generating a hidden pattern, and subsequent visible pattern is referred to as a Gibbs step.
- This chain of Gibbs steps between sampling from $P(h|v)$ and $P(v|h)$ can then be repeated as desired, the longer the chain the closer the samples will be to the true joint distribution that the model has learnt. For training an RBM HintonTODO CITE: CD-1Paper showed that 1 step is often enough in practice, as one step is enough to infer a direction to adjust the weights in.

The process of updating the hidden, then visible layers forms what is referred to as the `Gibbs Chain` and is visualised at layer level in figure 2.5.

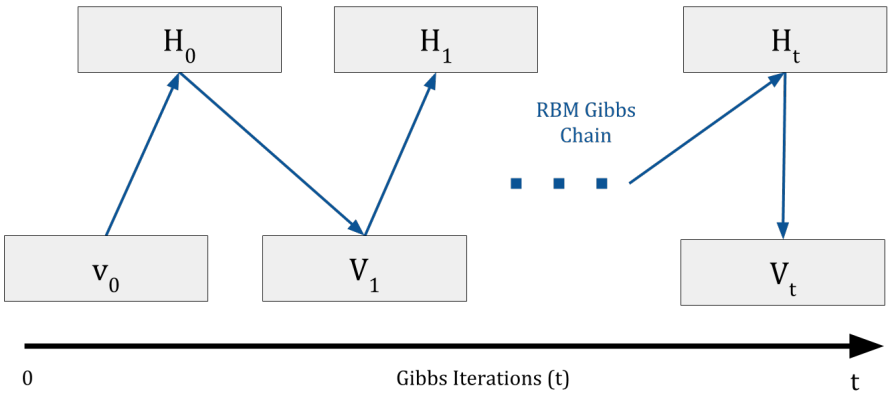

Figure 2.5: A figure illustrating a Gibbs chain where left to right indicates a Gibbs iteration. Note this is *not* a PGM.

The Gibbs update

In a standard RBM, updating a hidden unit h_j when performing Gibbs sampling is calculated by finding $P(h_j = 1|v)$ where v is an input pattern. In the context of an image, v would be the pixel values where each pixel corresponds to a visible unit, v_i. The probability of a given hidden unit activating is: TODO CITE: Gibbs sampling would be good here.

$$P(h_j = 1|v) = \sigma(\psi_j) \tag{2.2}$$

Where ψ_j is the weighted sum into the jth hidden unit and $\sigma()$ is the Sigmoid function, or it also known as the Logistic function $\sigma(x) = 1/(1 + e^{-x})$. Figure 2.6 illustrates ψ_j for an example RBM. As the weights are symmetric, sampling from the visible layer, given a hidden state is similar. That is $P(v_i = 1|h)$, where h is the entire hidden vector is given by:

$$P(v_i = 1|h) = \sigma(\phi_i) \tag{2.3}$$

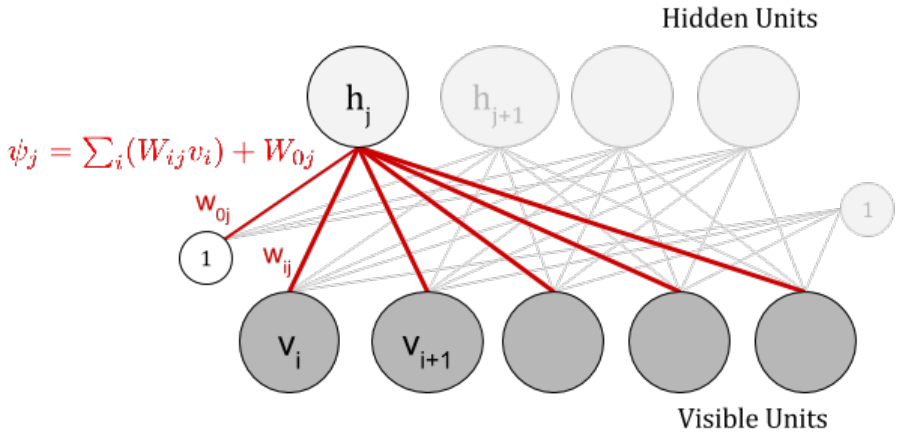

Figure 2.6: A diagram showing ψ_j, the weighted sum into the jth hidden unit. Note that W_{oj} is the hidden bias, represented as a unit that is always on with a weight into each hidden unit.

Where ϕ_i is the weighted sum into the ith visible unit, which is: $\phi_i = \sum(W_{ji}h_j) + W_{0i}$. Both ϕ_j and ψ_i can be expressed in alternative, but useful way:

$$\phi_j = \log P^\star(v, h|v_i = 1) - \log P^\star(v, h|v_i = 0) \tag{2.4}$$

$$\psi_i = \log P^\star(h, v|h_j = 1) - \log P^\star(h, v|h_j = 0) \tag{2.5}$$

2.5 The cost of sampling from $P(h|v)$

2.5.1 Inverting the Generative Model

Inverting a generative model can be referred to as inference, the process of reasoning about what we do not know, given that of which we do know. In generative models this is the 'posterior', the probability distribution of the hidden (latent) variables, given we know the state of the observable (visible) units. The process is called 'Inverting' because instead of the model generating the data (sampling from $P(v|h)$) we instead try to infer a representation given data $P(h|v)$.

2.5.2 Inverting a Sigmoid Belief Network

Performing inference in a Sigmoid Belief network would allow source separation, as each hidden unit could represent a simple cause. The SBN would be shown an input, and could extract the seperate causes that likely gave rise to that input. Despite the Sigmoid Belief Network being expressive and providing a succinct encoding of inter-variable dependancies, performing inference is intractable for a network of practical size [7].

There do exist algorithms for performing inference in Sigmoid Belief Networks. For instance, the Belief Propagation algorithm proposed by Judea Pearl [9] operates on this encoding, calculating the probabilities of a given network state (i.e. the state of all the variables). As well as constraining the architecture to be Directed Acyclic Graph, Belief Propagation is intractable to use as the number of variables grow. This intractability arises from the SBNs dependencies and the 'explaining away effect' [4]. Being able to efficiently sample from $P(h|v)$ is required for training generative modelsTODO CITE: em making Sigmoid Belief Networks impractical to train.

2.5.3 Inverting a Restricted Boltzmann Machine

A big payoff for the restriction in an RBM is inverting the model becomes tractable, as the latent variables no longer become dependant given the observed variables. This is illustrated in figure 2.4 the hidden unit h_1 is not dependent on h_2 wether or not we know anything about the visible units. This is the opposite of a Sigmoid Belief Network where knowledge of the visible units makes the hidden units dependant. By removing the recurence present in Boltzmann Machines, it reduces the expresiveness of the RBM network while making the RBM useable in practice as the Gibbs sampling process can stop after one Gibbs stepTODO CITE: .

Reconstructions, visualising what the RBM has learnt

RBMs create an internal representation given an input by sampling from $P(h|v)$. They can also generate a faux input given an internal representation. Performing one Gibbs iteration, that is, sampling from the hidden units given a 'clamped' input $P(h|v)$ and then taking the generated hidden state and generating a faux input (sampling from $P(v|h_{sampled})$) results in a `reconstruction`. Clamped input is where the visible units are set to be an input pattern. The model tries to reconstruct the input based on the internal representation it has learnt to model. This has applications in increasing the size of a dataset by introducing variation by generatin these faux inputsTODO CITE: .

RBM Fanstasies: The Free-Phase of a Generative model

In the same way that a Generative model uses reconstructions to try and recreate the supplied input vector, performing many, many (greater than 100) Gibbs iterations with no input pattern clamped allows the reconstructions to explore the probability mass that has been built by the model during training. Sampling from these wanderings creates what are refferred to as 'fantasies' or 'dreams'. These give a sense of what the model has learnt, and can act a smoke test for if the model has actually capture anything. TODO CITE: (TODO-CITE-PAPER-WITH-MNIST-DREAM-EVALUATION, they were crappy).

Bibliography

[1] BARBER, D. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, New York, NY, USA, 2012.

[2] HASTINGS, W. K. Monte carlo sampling methods using markov chains and their applications. *Biometrika 57*, 1 (1970), 97–109.

[3] HINTON, G. E. To recognize shapes, first learn to generate images. *Progress in brain research 165* (2007), 535–547.

[4] HINTON, G. E., OSINDERO, S., AND TEH, Y.-W. A fast learning algorithm for deep belief nets. *Neural Comput. 18*, 7 (July 2006), 1527–1554.

[5] HINTON, G. E., AND SEJNOWSKI, T. J. Analyzing cooperative computation. *Proceedings of the 5th Annual Congress of the Cognitive Science Society* (may 1983).

[6] HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences 79*, 8 (1982), 2554–2558.

[7] JENSEN, C. S., AND KONG, A. Blocking gibbs sampling in very large probabilistic expert systems. *Internat. J. HumanComputer Studies 42* (1995), 647–666.

[8] NEAL, R. M. Connectionist learning of belief networks. *Artificial intelligence 56*, 1 (1992), 71–113.

[9] PEARL, J. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the American Association of Artificial Intelligence National Conference on AI* (Pittsburgh, PA, 1982), pp. 133–136.

[10] PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[11] PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[12] SMOLENSKY, P. *Foundations of harmony theory: Cognitive dynamical systems and the subsymbolic theory of information processing*. Parallel distributed processing: Explorations in the ..., 1986.