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

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 Problem

1.1.1 Deep Belief Networks can achieve state of the art performance

Deep Belief networks are powerful models that have proven to achieve state of the art performance in many domains. For instance a non-exhaustive list is image classification, dimensionality reduction, natural language recognition, Document classification, Semantic Analysis and .

DBNs capture non-linear interactions between low level features, in the context of image classification the lower layers can capture image filters.

1.1.2 DBNs have no mechanism for separating sources

Despite a DBNs expressiveness, there is no way to extract these interactions. If an input has multiple sources then the complex combination is instead learnt, the network has no mechanism for extracting multiple causes. This is the motivation for this project, to be able to separate the sources of data in a new model.

[1]

1.1.3 Restricted Boltzmann Machines cannot separate sources either

Restricted Boltzmann Machines are two layer, fully connected, unsupervised nueral networks. DBNs are constructed by stacking RBMs. Being the building block of the powerful DBN, RBMs are a natural starting point for representing mutliple sources. RBMs make the assumption that the features of the input data are dependant in the prior, as they are independant in the posterior. The latter makes them tractable to use in practice, but also means they model/encode a single representation. Again using the example of images, an input image will map to a single representation, again there is a lack of mechanism for modelling sources that are acting independantly.

1.1.4 Sigmoid Belief Networks; Intractably rich in practice

The Sigmoid Belief Network, the parameterized version of a Bayesain/Belief network appears as a natural choice for modelling inpendant sources in that it makes a polar assumption to the RBM; – Warning Semicolon Use – Every feature has an independant cause. The sigmoid belief networks assumption could capture data that has multiple sources, but this is intractable in practice.

1.2 Solution

1.2.1 Trading tractibility for Source Separation

Frean and Marsland propose a generative model that aims to trade a small amount of the RBMs performance for richness, finding a middle ground between the sigmoid beleif network and the restricted boltzmann machine. Frean and Marsland also propose an algorithm to invert this model, seperating the sources of an input.

The new generative model, referred to onwards as an ORBM, uses an RBM to model each source and a sigmoid belief network to capture their combination to from data. This project explores the ORBM use for separating two causes.

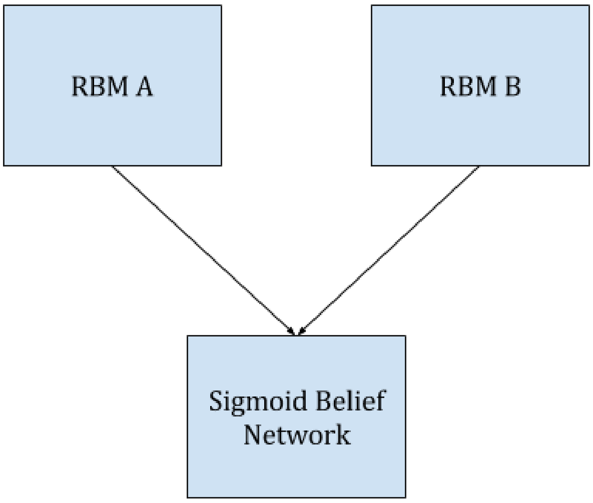

Figure 1.1: A graphical representation showing the proposed generative model for capturing two causes, the ORBM.

Given the proposed model and algorithm, this project answers the following questions:

- Can this model encode data comprised of more than one cause as it's constituted causes? That is, can the model and new algorithm for inverting it, perform source separation.
- Is the ORBMs two cause structure to rich to be tractible in practice?

1.3 Results

TODO Draft Points Still to write TODO

Spoil the results here. Good on smaller cases, less good on larger case

Chapter 2

Backgroud

As the ORBM builds on the previous work of Restricted Boltzmann Machines and Sigmoid Belief networks, the concepts and previous work in source separation, generative models as well as background on RBMs and Sigmoid Belief Networks need to be introduced.

2.1 Source Separation, nature can do it

2.1.1 An example, the cocktail party problem

A famous example that illustrates the idea of source separation is the cocktail party problem. At a cocktail party many conversations are taking part at the same time, creating noise, a composition of all the conversations. Despite this, a partygoer is able to separate their conversation from cacophany, separating the sources.

The applications of source separation are far wider than talkative partygoers. In the field of signal processing

TODO Draft Points Still to write TODO

2.2 Generative Models

Generative models are a powerful way to model data. TODO WORDING (TODO-GRAB-THOSE-GENERATIVE-MODEL-USES-CITATIONS) Basically justify generative models.

The ORBM proposed in this project aims to represent data generated by two indepedanlty acting causes and does so by combining two existing generative models, the Restricted Boltzmann Machine, and the Sigmoid Belief Network.

2.2.1 Terminology in Generative Models observable and hidden variables

Generative models are comprised of variables, often referred to as units. Some of these variables are observed, that is their state is known. These are often referred to as the ‘visible‘ units and are used to represent the training data. For example in the image domain, the visible units correspond to the pixels of the image.

The variables that are not observed, are latent variables, often referred to as ‘hidden units‘ as they are not observed.

Connections between units are used to encode relationships between the variables, where the relationship may be causal, such as in a Sigmoid Belief network or an encoding/representation in the Restricted Boltzmann Machine.

Collections of 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, the visible pattern would be the pixels of the image ravelled into a one dimensional vector.

2.2.2 PGMs as a tool reasoning about generative models

TODO Draft Points Still to write TODO

Probalistic Graphical Models or PGMs for short, are an expressive way to represent a collection of related, stochastic variables. If the graph is directed then the edges represent causation, this is also referred to a Bayesian network. Conversely, if the graph was undirected then edges represent a dependancy or mapping. Throughout this report, RBMs, Sigmoid Belief Networks and the proposed ORBM will be shown in this format.

Figure 2.1 shows an abstract example of a directed PGM, where B is the underlying cause of A, we cannot observe B directly, instead it's state is represented as a ‘belief‘ or a probability of being in a given state.

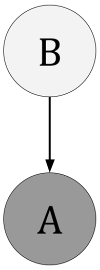

Figure 2.1: An example PGM, showing an observed variable ‘A‘ and it's hidden cause ‘B‘.

2.3 Sampling and inverting the model

Sampling is the process of drawing samples from a distribution. It is used when the distrubtion we want samples from is intractable to calculate analytically. Sampling is required to train generative models, as often the gradient to be climbed/descended involves calculating a probability in the generative model.

TODO Draft Points Still to write TODO

- Inference is the process of given reasoning about what we do not know given that of which we do know.
- In a Generative Model this amounts to the Posterior

2.3.1 Gibbs sampling, a subset of Markov Chain Monte Carlo

- The importance of Markov Chains and mixing time are crutial in this project

Gibbs sampling is a special case of Markov Chain Monte Carlo, a technique for drawing sampling from a complex distribution. The probability mass (the joint distribution) of a generative model is a common use case for Gibbs sampling.

Gibbs sampling explores the desired probability distribution, taking samples of that distributions state, allowing iterations of exploration between drawing of a sample to ensure that the samples are independant. The process of taking a step between states is referred to as a Gibbs iteration.

Gibbs sampling is used for performing inference in the RBMs, Sigmoid Belief Networks and 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 more than one may be required.

Mixing Time

MCMC methods require a 'mixing' phase to ensure convergence, that is that the sample is being drawn from a representative part of the desired distribution. This is part of the trade off the ORBM attempts to make, as a mixing time is introduced that is not present in the RBM.

2.3.2 Reconstructions, visualising what the model has learnt

Generative Models can create an internal representation given an input. They can also generate a faux input given an internal representation. Performing one Gibbs iteration, that is sampling from the hidden units given an input $P(\tilde{h}|\tilde{v})$ and then taking the generated hidden state and generating a faux input. The model tries to reconstruct the input.

Ancestral Sampling or fanstasies of the model

TODO Draft Points Still to write TODO

In the same way that a generative model uses reconstructions to try and recreate the supplied input based purely on how it's represented that input, performing many, many (greater than 100) Gibbs iterations with no input pattern clamped allows the reconstructions to explore the probability mass that the model has built up 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-PAPER-WITH-MNIST-DREAM-EVALUATION, they were crappy).

2.4 An intractable model for causes

2.4.1 Sigmoid Belief Networks

TODO Draft Points Still to write TODO

The ORBM relies on the Sigmoid Belief Network to capture the causation. The Sigmoid Belief Network is composed of units with weights and a sigmoid activation function, akin to that of a perceptron linear threshold unit/Perceptron. The probability of a node being 'on' is found by taking the weighted sum of all input to that node and applying a Sigmoid function or another activation function that ensures a values between 0 and 1.

Belief Networks appear to be an intuitive way to model data in machine learning, as rich dependancies often present in real data can be expressed in it's architecture. Nodes in the network represent binary variables which are dependent on ancestor nodes, the degree of which is encoded in a weight on a directed edge between them.

Performing inference in a Sigmoid Belief network would allow source separation in that each hidden unit could represent a cause. Meaning if a causes state could be inferred from an input item, individual causes could be examined for an input. For example if the input was an n by n image, the Sigmoid Belief Net makes the assumption that each pixel has an independent cause.

Despite the Sigmoid Belief Network being expressive and providing a succinct encoding of inter-variable dependancies, the expressiveness is too rich such that performing inference is intractable. There do exist algorithms for performing inference in Sigmoid Belief Networks. For instance, the Belief Propagation algorithm TODO CITE: The paper where BP/Sum Prod proposed operates on this encoding, calculating the probabilities of a given network state (i.e. the state of all the variables). Belief Propagation is intractable to use as the number of variables grow TODO CITE: the paper explaining intractable for belief prop.

This intractability arises from the Sigmoid Nets richness and the 'explaining away effect'. Inference is required for training generative models making Sigmoid Belief Networks impractical to train. TODO CITE: It has been done, link to paper where they do it.

2.4.2 Explaining Away creates a trade off between richness and tractability

TODO Draft Points Still to write TODO

The power of the Belief Network is also it's weakness, a rich structure that models a system of interest inherently has dependancies. In its minimal case explaining away can be seen in a 3 node network popularised by TODO CITE: AI-A-MODERN-APPROACH-TODO. TODO-GRAPHIC as shown in figure 2.2. Each of the nodes represents a binary state. For instance $Burglar = 1$ means that the person owning the `Alarm` has been burgled. Also note how the connections between the units have arrows, this is causal.

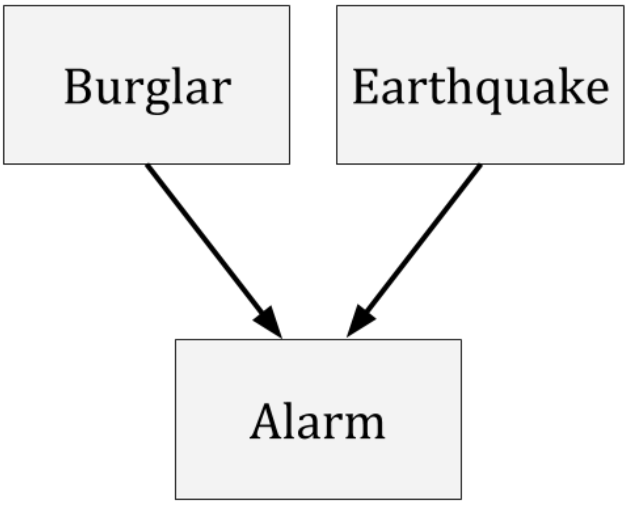

Figure 2.2: The famous Burglar, Earthquake, Alarm network showing a minimal case of explaining away.

In the network shown in figure 2.2, knowledge of the `Alarm` creates a dependance between `Burglar` and `Earthqaukes`. For instance, say the Alarm has gone off and we know an earthquake has occurred, our belief in being burgled decreases. 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`. TODO WORDING 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 is intractable.

2.4.3 Boltzmann Machines

A Boltzmann machine TODO CITE: Cite the Harmonium, markov field has qualities in common with Belief Networks. Both are generative models with their nodes having probabilities of being active based on neighboring nodes. Connections between nodes have associated weights as shown in figure 2.3. These weights are symmetric. TODO WORDING Unlike a Belief Network, a Boltzmann Machine is a undirected network that allows cycles and thus more complex data can be captured.

TODO WORDING Connections between nodes no longer encode causal information, instead a depedancy, the difference being that a connection encodes a relationship as they are not directed.

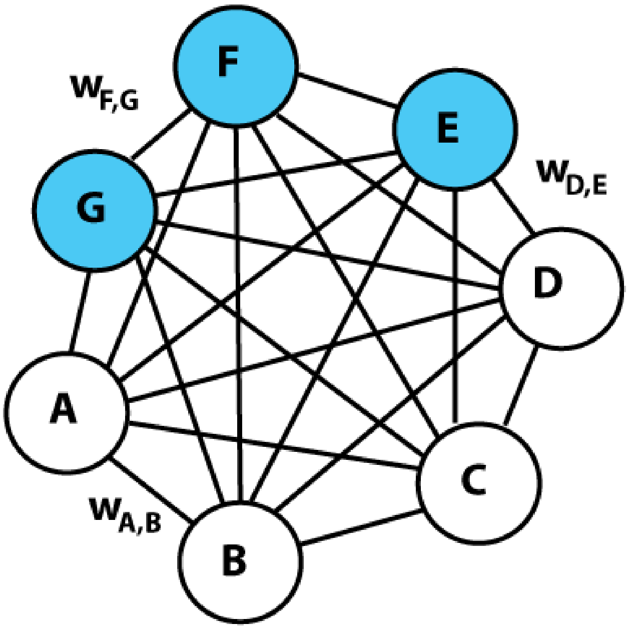

Figure 2.3: A Boltzmann Machine, the blue shaded nodes representing the observed variables, and the non-shaded nodes the latent variables.

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.

TODO CITE: TODO-REFENCE-PAPER-OF-THIS The Boltzmann Machine was shown, given an unreasonable amount of time, to be able to perform better than the state of the art at the time.

2.5 The Current Approach: A Strong assumption

2.5.1 Restricted Boltzmann Machines

While Boltzmann Machines are impractical to train and sample from as networks grow in size TODO CITE: Need to cite this... their architecture can be altered to alieviate these shortcomings. The restriction, proposed by TODO CITE: Hinton, a proper cite requires the network to be bipartite, where connections are forbidden between the layer of hidden units and the layer of visible units repectively.

An example Restricted Boltzmann Machine architecture is shown in figure 2.4. The affect of the restriction is that inference can be tractably computed, as the latent variables no longer become dependant given the observed variables.

For example in figure 2.4 the hidden unit h_1 is not dependant 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. The RBMs nature removes the recurence present in Boltzmann Machines. This reduces the expresiveness of the network but makes the RBM useable in practice. TODO CITE: The paper about Boltzmann Machines being really good when actually left to find solution.

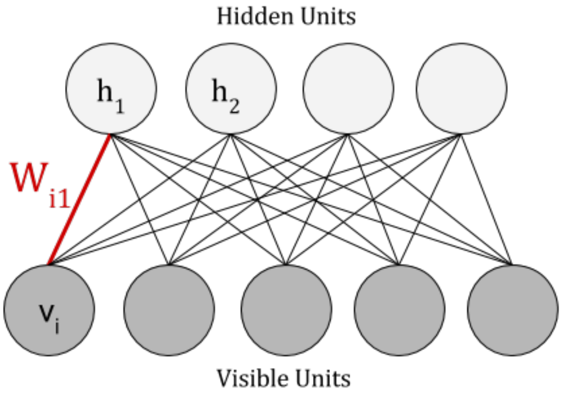

Figure 2.4: An example Restricted Boltzmann Machine with four hidden units, and five visible units. Note that the edges between units are not directed - representing a dependency not a cause.

TODO Draft Points Still to write TODO

Talk about how Gibbs sampling in RBMs allows us to approximately sample from the hidden (unknown/representation) given the visible (known/input data).

TODO Draft Points Still to write TODO

Hinton TODO-REFERENCE-THE-PAPER proposed a restriction by way of assumption to the Boltzmann Machine that makes it tractable to sample from and therefore train. Boltzmann Machines of this architecture are referred to as Restricted Boltzmann Machines, or RBMs for short.

The assumption being that the observable and latent variables are independant respectively, enforcing a two layer, fully connected bipartide structure.

Tractable Training - Contrastive Divergence

Hinton TODO-CITE-CLASSIC-PAPER proposed Contrastive Divergence as a method for training RBMs efficiently. The algorithm leverages the now tractable wake phase because $P(h|v)$ is efficeint to compute. However the free or sleep phase required another restriction where the network is only left to its own dynamics can be limited to only one iteration and still perform well. TODO-CITE-CD-PAPER

The observed variables are often referred to as the visible units, and will be so forth in this report. The latent variables are often referred to as the hidden units, and will be so forth in this report. Therefore the Restricted Boltzmann machine transforms some visible unit into a hidden representation. These two layers of units can be thought of as vectors of binary values, referred to as v and h for visible and hidden layers respectively.

This restriction allows an efficient calculation of the Wake Phase of generative model learning, as the $P(h|v)$ can be calculated as a simple weighted sum passed through a sigmoid followed by a bernouli trial where the probability of being 1 is equal to the result of sigmoid.

2.5.2 Deep Learning

- Discuss deep learning as there are clear parallels to Deep Belief Networks and the new approach
- in paritcular how the deep networks have this process of freezing the weights and creating a sigmoid belief layer instead. There seem to be clear parallels between a deep network with one RBM to the ORBM.
- Unrolling the gibbs chain and we are in effect training an infinite depth sigmoid beleif net (TODO-REFERENCE-HINTONS-PAPER-HERE)

2.5.3 Inference

One of the reasons the Restricted Boltzmann Machine is effective in practice is inference can be performed efficiently. Inference being computing the posterior.

2.5.4 Evaluating Restricted Boltzmann Machines

- Being unsupervised makes it difficult to evaluate RBMs. Often used as part of a deeper network, feature extractor, autoencoder
- Hinton Diagrams allow visualisation of hidden unit utlisation (TODO-SOME-SORT-OF-CITE). The weights out of a given hidden unit can be visualised in visible data space. The weights should exhibit some structure if they are being utlisied. This is a good smoke tests for non-tulised hidden units will look very similar to units with random initial weights.
- Small Cases
- In trivial cases an RBM can inspected analytically. Reconstructions of the dataset should match the dataset with approximately the correct proportion. For instance training RBMs on 2 bit XOR should result in mostly [1,0] and [0,1] but not [1,1] and [0,0].
- Hand craft weights can be used to perform inference in a 'perfect model'. For instance an RBM that can capture two bit, logical XOR can be represented as :TODO-INSERT-PIC
- Large Cases
- In non-trivial cases, with larger datasets, reconstructions can be compared to the dataset but given the unsupervised nature of RBMs emperically detecting if a model is trained is difficult.
- The log liklihood of the RBMs generative model exbiting the dataset is a good measure that can be approximated (because we have to sample).

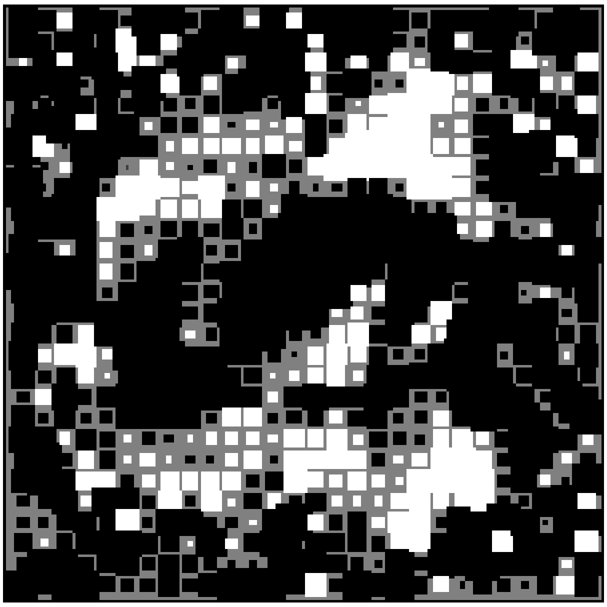

Figure 2.5: Good Hinton Diagram

- We can train a classifier on the RBMs hidden representation. This can be compared for a ORBM and RBM.

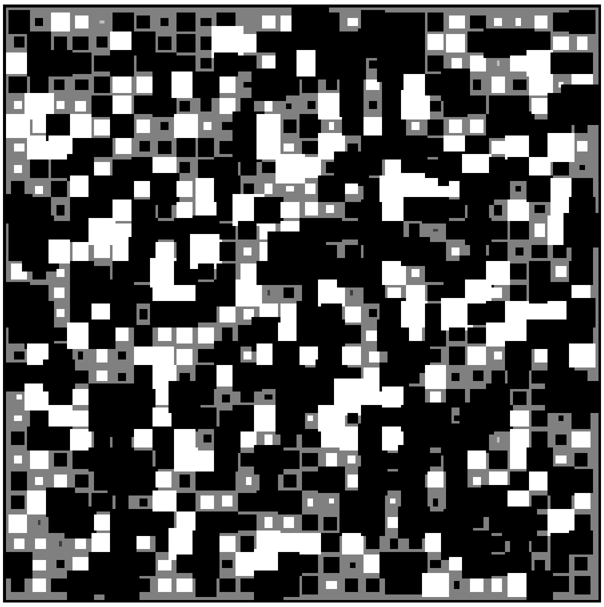

Figure 2.6: Bad Hinton Diagram

2.6 A New Approach - The ORBM

- Frean and Maslands new approach combines the Restricted Boltzmann Machine and the Sigmoid Belief Network, the RBMs allowing the rich complex causes to be encoded independantly, and the Sigmoid Belief Network modelling the combination of the causes to form the observable data.'
- By building on these existing methods can leverage exisitng algorithms
- Can verify the inference algorithm with pre trained RBMs for each cause.
- Like the RBM leveraged in the ORBM, it is difficult to evaluate, But similar techniques can be leveraged

2.6.1 Architecture

- Two RBMs, on for each cause, then they combine via a Sigmoid Beleif Layer. Weights between the RBM and Sigmoid Layers are shared. (TODO-A-DIAGRAM)
- Diagram of the ORBM Architecture Including U Layer. Make sure I'm explaining the U layer.
- In fact Ua, Ub are like mirrors of the visible.

2.7 Inference In the ORBM

2.7.1 The Gibbs Chain

TODO WORDING To describe the ORBM and how to perform inference, traditional RBMs need to be thought of in a different yet equivalent way.TODO CITE: This is similar to Hinton infinitely deep sigmoid belief networks.

For the following discussion, the hidden units are indexed by j and the visible units are indexed by i.

To perform Gibbs sampling in an RBM one must sample from $P(\tilde{h}|\tilde{v})$ giving a hidden state $\tilde{h}$. Using this hidden state a visible state is then generated, $\tilde{v}'$, by sampling from $P(\tilde{v}'|\tilde{h})$ and so on. This forms the Gibbs Chain TODO CITE: Feel like I could cite this again in like a CD paper or something. This process is visualised in figure 2.7

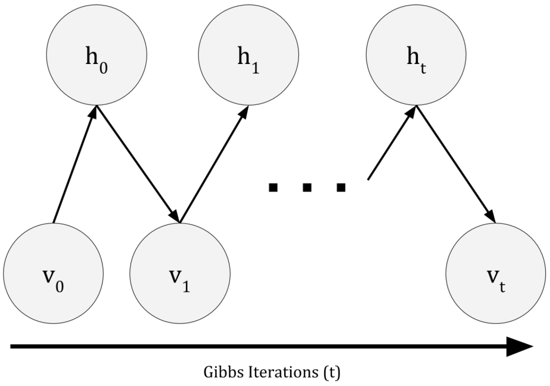

Figure 2.7: A figure illustrating a Gibbs chain where left to right indicates a Gibbs iteration. Note this is not a PGM.

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

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

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 2.8 TODO WORDING shows psi for an example RBM.

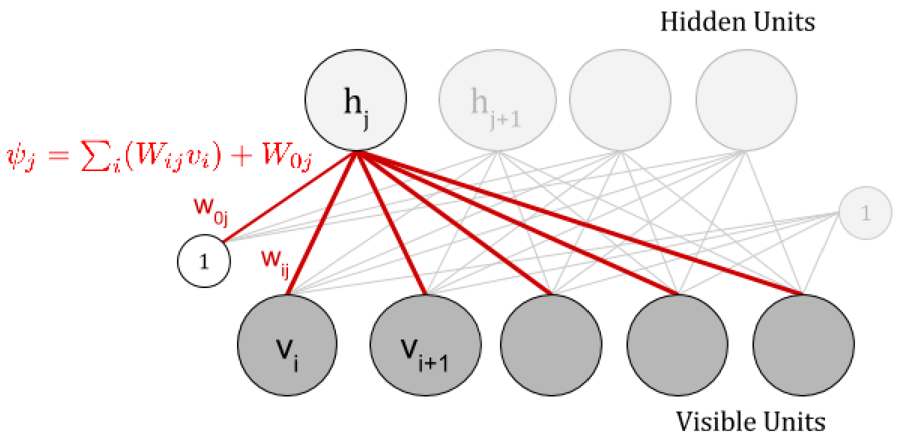

Figure 2.8: A diagram showing ψ_j, the weighted sum into the jth hidden unit.

As the weights are symmetric, sampling from the visible layer, given a hidden state is similar. That is $P(v_i = 1|\tilde{h})$, where $\tilde{h}$ is the entire hidden vector is given by:

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

Where ϕ_i is the weighted sum into the ith visible unit.

$$\phi = \sum(W_{ji} h_j) + W_{0i}$$

2.7.2 Unrolling a Gibbs Chain, a different perspective on RBM inference

Before the ORBMs architecture can be introduced, Gibbs sampling in an RBM must be presented in a different way. TODO CITE: Hintons paper on unrolling the Gibbs chain. He showed that RBM is infinitely deep belief network with tied weights. This unrolling a single Gibbs iteration is illustrated in figure 2.9, the U layer corresponding to the V layer after one Gibbs iteration. To clarify, the U layer corresponds to v_1 in figure 2.7.

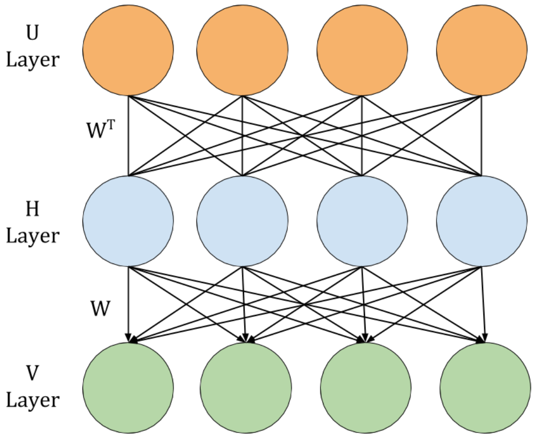

Figure 2.9: A diagram illustrating ‘unrolling’ an RBM by one Gibbs iteration.

Note the weights between the H and U layer are the transpose of the weights between the V and H layer.

We can show that Gibbs sampling in this RBM unrolled with a Sigmoid belief network is equivalent to Gibbs Sampling in a standard RBM.

Ancestral Sampling in this equivalent network

Ancestral sampling in an RBM TODO CITE: as described in the section where I talk about dreams.... This network behaves a similar way, where a Gibbs chain is run between the top two layers, the U and H layer, until the last iteration. At the last iteration a hidden pattern is sampled and then is pushed through the Sigmoid belief network between H and V. This is illustrated in figure 2.10.

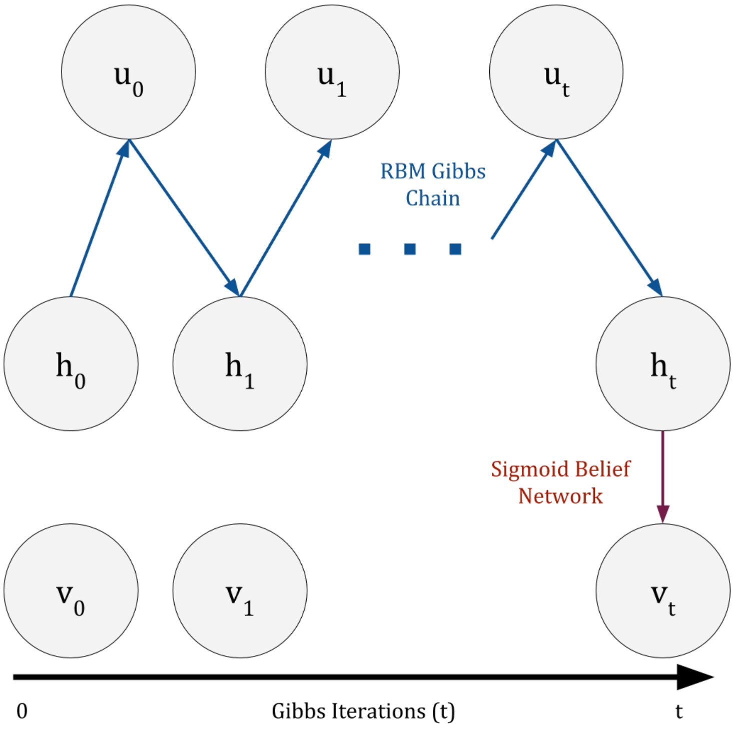

Figure 2.10: A diagram showing Ancestral sampling in the equivalent network, where normal sampling in the top 2 layers is performed until Gibbs iteration t the hidden state is pushed through the bottom layers Sigmoid Network.

TODO WORDING We know (from above) how to generate the h sample: Gibbs sampling from the RBM will do it. Then, the conditional probability of v under a sigmoid belief net is (by definition) $p(v_i = 1|h) = \sigma_i(h)$. Thus Gibbs sampling from a simple RBM ending in a sample for v is the same as sampling h from the same RBM and then using a sigmoid belief net for the last step.

TODO WORDING However, there's another way to draw such samples. Write (product rule) $\log P^{\star}(h,v) = \log P^{\star}(h) + \log P(v|h)$. We have the second term already:

$$\log P(v|h) = \sum_i v_i \log \sigma_i(h) + (1 - v_i) \log(1 - \sigma_i(h))$$

To find $P^{\star}(h)$ we need to marginalise that joint over all $\mathbf{v}$ configurations:

$$\begin{aligned}
P^{\star}(h) &= \sum_{v_1=0}^{1} \cdots \sum_{v_n=0}^{1} \exp \left[\log P^{\star}(h,v) \right] \\
&= \sum_{v_1=0}^{1} \cdots \sum_{v_n=0}^{1} \exp \left[\sum_i \sum_j h_j W_{ji} v_i \;+\; \sum_i W_{0i} v_i \;+\; \sum_j W_{j0} h_j \right] \\
&= \sum_{v_1=0}^{1} \cdots \sum_{v_n=0}^{1} \exp \left[\sum_i v_i \phi_i(h) \;+\; \sum_j W_{j0} h_j \right] \\
\text{where } \phi_i(h) &= \sum_j W_{ji} h_j + W_{0i} \\
&= \exp \left[\sum_j h_j W_{j0} \right] \;\times \sum_{v_1=0}^{1} \cdots \sum_{v_n=0}^{1} \prod_i \exp \left[v_i \phi_i(h) \right] \\
&= \exp \left[\sum_j h_j W_{j0} \right] \;\times \prod_i \left(1 + e^{\phi_i(h)} \right) \\
\text{and so} \log P^{\star}(h) &= \sum_j h_j W_{j0} \;+ \sum_i \log \left(1 + e^{\phi_i(h)} \right) \\
&= \sum_j h_j W_{j0} \;+\; \sum_i \phi_i(h) \;-\; \sum_i \log \sigma_i(h)
\end{aligned}$$

So far we've figured out $\log P^{\star}(h)$ for the RBM that is the 'top layer'.

Therefore another way to write $\log P^{\star}(h,v)$ is

$$\log P^{\star}(h,v) = \underbrace{\sum_j h_j W_{j0} \;+\; \sum_i \phi_i(h) \;-\; \sum_i \log \sigma_i(h)}_{\log P^{\star}(h)} \;+\; \underbrace{\sum_i v_i \log \sigma_i(h) + (1 - v_i) \log(1 - \sigma_i(h))}_{\log P(v|h)}$$

By collecting terms and simplifying one can readily those that this matches the earlier form TODO CITE: The earlier equation.

2.7.3 The ORBM architecture is derived from the equivalent RBM architecture

This equivalent network can be extended to form the ORBMs architecture.

The design of the ORBM is such that two RBMs are used to model independant causes. As a generative model, the RBMs act independanlty, combining to form the visible pattern $\tilde{v}$. This is shown in figure 2.11. This architecture is simpler in practice as the U layers are factored out during the derivation.

2.7.4 Gibbs in a two cause model

For reasoning about the two cause model of the ORBM it is useful to work in terms of RBM A and consider the input of the other RBM as ϵ.

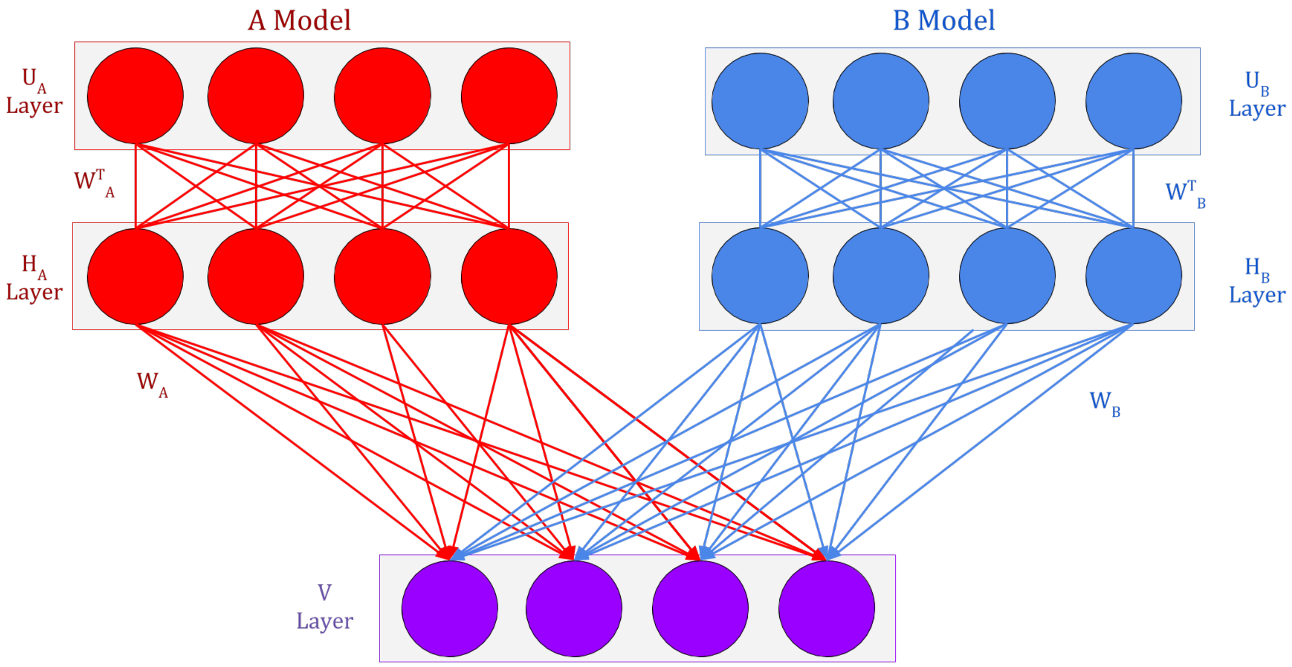

Figure 2.11: The full ORBM architecture, where A and B are the two causes that combine to form the data.

TODO WORDING We're interested in how this will affect the Gibbs updates to the hidden units h in the first RBM.

TODO WORDING Note: ϵ_i is in general going to be a weighted sum of inputs from the second RBM's hidden layer, plus a new bias arising from the second RBM. Although the two biases both going into the visible units seems (and might be) redundant, in the generative model it seems sensible that visible activations under the two "causes" would have different background rates if taken separately (ie. different biases). So for now I think we should leave them in, but maybe hope to eliminate merge if possible in future, if it helps anything...

TODO WORDING Before going on to the Gibbs Sampler version in which visible units are clampled, consider "ancestral" sampling from the model: each RBM independently does alternating Gibbs Sampling for a (longish) period, and then both combine to generate a sample v vector, by adding both their weighted sums (ϕ) and adding in both their visible biases too . That's a much more efficient way to do the "sleep" phase than doing what follows (which is mandatory for the "wake" phase samples however).

The Gibbs update step is given by $p_j = \sigma(\phi_j)$ with $\psi_j = \log P^*(h, v; h_j = 1) - \log P^*(h, v; h_j = 0)$.

However this time we don't have exact correspondence with an RBM because only the final step involves ϵ, not the reverberations in the RBM above it that generates h. So it's not enough to consider just the RBM alone, with it's joint being just the product of factors in the first line of math above. We need to incorporate the last step explicitly, with its slight difference in the form of ϕ_i^B. We know the joint decomposes into this:

$$\log P^*(h, v) = \log P^*(h) + \log P(v|h)$$

where the first term is the vanilla RBM probability but the second is the final layer's probability, now given by

$$\log P(v|h^A, h^B) = \sum_i v_i \log \sigma(\phi_i^A(h) + \phi_i^B) + (1 - v_i) \log(1 - \sigma(\phi_i^A(h) + \phi_i^B)$$

To carry out Gibbs sampling in the hidden layer of this architecture we need to calculate $\psi_j^A = \log P^*(h, v; h_j^A = 1) - \log P^*(h, v; h_j^A = 0)$. We'll use the fact that $\phi_i^A(h; h_j^A = 1) = \phi_i^A(h; h_j^A = 0) + W_{ji}^A$.

And we'll abbreviate $\phi_i^A(h; h_j = 0)$ to ϕ_i^{Aj0}.

We obtain:

$$\psi_j^A = \sum_i v_i \log \left(\frac{1 + e^{-\phi_i^{Aj0} - \phi_i^B}}{1 + e^{-\phi_i^{Aj0} - W_{ji} - \phi_i^B}} \frac{1 + e^{\phi_i^{Aj0} + W_{ji} + \phi_i^B}}{1 + e^{\phi_i^{Aj0} + \phi_i^B}} \right) \;+\; \sum_i \log \left(\frac{1 + e^{\phi_i^{Aj0} + W_{ji}}}{1 + e^{\phi_i^{Aj0}}} \frac{1 + e^{\phi_i^{Aj0} + \phi_i^B}}{1 + e^{\phi_i^{Aj0} + W_{ji} + \phi_i^B}} \right)$$

Now $\phi = \log \frac{1+e^{\phi}}{1+e^{-\phi}}$ (Marcus' magic identity), which is= $\log \frac{\sigma(\phi)}{\sigma(-\phi)}$. So the first term simplifies to $\sum_i v_i W_{ji}$, which is the same as that in a "vanilla RBM".

The second term can also be simplified, using the identity $\log(1 - \sigma(\phi)) = \phi - \log(1 + e^{\phi})$.

This leads to the following Gibbs Sampler probability of the j-th hidden unit in network A being 1: $p_j = \sigma(\psi_j^A)$ with

$$\psi_j^A = \underbrace{\sum_i W_{ji}^A v_i}_{\text{vanilla RBM}} \;+\; \underbrace{\sum_i C_{ji}^A}_{\text{correction}}$$

where

$$\begin{aligned} C_{ji}^A &= \log \left[\frac{\sigma(\phi_i^{Aj0})}{\sigma(\phi_i^{Aj0} + W_{ji}^A)} . \frac{\sigma(\phi_i^{Aj0} + W_{ji}^A + \phi_i^B)}{\sigma(\phi_i^{Aj0} + \phi_i^B)} \right] \\ &= \log \sigma(\phi_i^{Aj0}) \;+\; \log \sigma(\phi_i^{Aj0} + W_{ji}^A + \phi_i^B) \;- \log \sigma(\phi_i^{Aj0} + W_{ji}^A) \;-\; \log \sigma(\phi_i^{Aj0} + \phi_i^B) \\ &= \log \left[\frac{\sigma(\phi_i^A - h_i^A W_{ji}^A)}{\sigma(\phi_i^A + (1 - h_i^A) W_{ji}^A)} \right] \;-\; \log \left[\frac{\sigma(\phi_i^{AB} - h_i^A W_{ji}^A)}{\sigma(\phi_i^{AB} + (1 - h_i^A) W_{ji}^A)} \right] \end{aligned}$$

where $\phi_i^{AB} = \phi_i^A + \phi_i^B$.

Notice that, weirdly enough, v plays no role in this correction!

It is clear that adding ϕ_i^B has introduced a dependency between the whole of h, which is "a bit of a worry".

TODO WORDING The majority of this...

2.7.5 Approximating the Correction

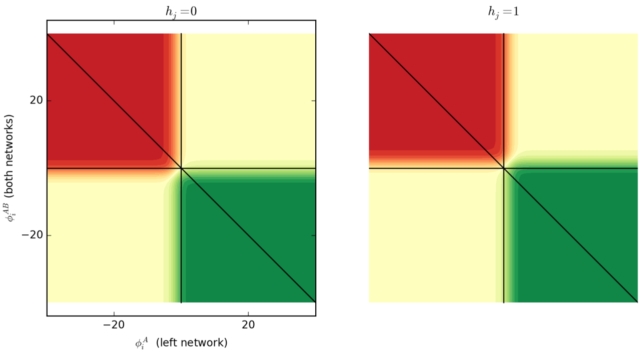

Figure 2.12: A diagram what the correction function looks like for ...

Chapter 3

Work Done

TODO WORDING Make sure you talk about why the sigmoid is the most appropriate atrication function! Maybe this should go in background..

3.1 Design

3.1.1 Evaluation Design

Mixing Time Evaluation

- List the types Traceplot, Densityplots, Gelbin and Rubin multiple sequence diagnostics
- Performed Geweke Diagnostic

Reconstructions As a measure of Performance

- Cite literature where reconstructions are used, starting from early RBM training papers by Hinton, up to more recent in deeper networks - the goal being to show the reader that reconstructions are used well in practice.

Evaluation Methods

This is the way I will gain the reconstructions.

- Non-trivial to evaluate
- 2 Bit XOR - Minimal Case
  - Method
  - Can check the inference algorithm by using RBMs with hand trained weights
  - Using a single RBM, copied.
  - Diagram with small explaination why RBMs with these weights will work.
    - 2 Hidden Units, 2 Visible Units. Weights Matrix of [-1,1],[1,-1]
    - This ensures that if one visible unit is 1, the other will get set to off.
  - Look at the reconstructions of these RBMs, should match XOR patterns.
    - 100000 free phase samples from RBMs resulted in [1,0] and [0,1] most of the time. (TODO-SHOW-GRAPHs). 100000 free phase samples is quite quick to calculate for only 4 weights.

        - 2 sets of 100000 reconstructions given the patterns [1,0] and [0,1] resulted in all reconstructions for each set producing [1,0] and [0,1].
    - Hypothesis
    - Clamp the visible to patterns of interest. Can see the dynamics of the system.
        - 1,1 This should result in reconstructions matching XOR - [1,0] and [0,1]
        - 1,0 This should result in reconstructions of just - [1,0]
        - 0,1 Should behave symmertrically to [1,0].
        - 0,0 Under the dynamics of the generative model should fall into [0,0].

- Y Bit pattern, X bits On
    - Have to train an RBM to recognise X neighbouring bits on at a time in a Y bit pattern. e.g. X = 2, Y = 5, [0,0,1,1,0]
    - num Y hiddens, or more.
    - While Y less than 10 it is easy to visualise if separation is occurring.

- 2D Pattern, Square Separation, Single Model
    - Dataset:
        - 2x2 square in a 5x5 image. Dataset of every possible configuration of said square.
    - Method:
        - train a single RBM to represent 2x2 squares in 5x5 space. Nice and small, can still inspect reconstructions, and dreams.
        - Compose two square images with each other, can the ORBM architecture separate the images.

- 2D Pattern, Different Rectangle Separation
    - Dataset
        - n by m rectangles in 5x5 Images (TODO-IMAGE-IN-HERE) where n and m are not always equal.
    - Method
        - Superimpose two images as the same way as before. How well can they separate.

- MNIST Digits
    - Dataset
        - 500, 28 by 28 pixel, 0-1 valued, handwritten digit images form the training set.
    - Method
        - Train an single RBM per digit (0-9) in the dataset. Each RBM having 100 hidden units, (TODO-CITE-COOKBOOK) for reasoning.
        - For all 500 training digit images, compose it with every other digit, and itself. This way the ground truth, the sources are known, therefore making evaluation possible.
        - For all of these compositions us the RBMs trained of the corresponding sources to:

- Compute the RBM reconstructions for the two sources given their composite input. (TODO-SHOW-DIAGRAM)
- Compute the ORBM reconstructions, the ORBM using the same RBMs as in the RBM evaluation.
- Because 28 by 28 pixels equates to 784 features, emperically examining all reconstructions is non-trivial. Hence some scoring functions are required.
- Scoring
- Cross Entropy - Approximate the Log likelihood of the underlying dataset.
- Pixel Difference Score - Absolute difference in pixels of the image
- Cosine Angle - between reconstruction and target vectors. This has the benefit of also being somewhat magnitude agnostic. (TODO-CITE-AS-MEASURE-WITH-JUSTIFICATION).
- Repeat this process 30x to give confidence in findings. Scores calculated can then be meaned over the 30 runs.

Classifciation Accuracy on Hidden Representation

Not sure about this one.

3.1.2 Dataset Choice

- 2 Bit XOR - It's the minimal case. Can examine reconstructions and dreams/fantansy emperically. Also can check that code performs as expect by manually calcating expected values for the algorithm.

- X Neighbouring Bits On in Y Bit Pattern - A natural next step from 2 bit XOR. Still trivial to train an RBM to represent, quick feedback to ensure the algorithm works. Allows comparing the different approximations tractable as number of hidden units is small enough.

- Square Patterns in a 5 by 5 Pixel Image - Another natural step from neighbouring bits, trival to construct a dataset. Very easy to compose. Can use the same model for both models in the ORBM Architecture.

- Rectangles in a 5 by 5 Pixel Image - Next step, builds on the previous test but adds a different model for each source. Also different shapes of model can be used.

- Continous Rectangles (Pixel values 0 to 1) in a 5 by 5 Image - Another variation working towards the MNIST dataset, same as the previous but has continous visible pixels. Allows to different intensitities, interesting cases where the images overlap and increase in intensity.

- MNIST Handwritten Digit Dataset - Pixel values from 0 to 1, 28 by 28 pixel image (784 pixel features.) Used extensiviely in previous studies. Gives me confidence that RBMs can learn these representations. Also allows metaparameters don't have to be tweaked as much as studies already have found reasonable values. Non-trivial to evaluate, but more of a real world case.

Bibliography

[1] History of SciPy. `http://wiki.scipy.org/History_of_SciPy/`. Accessed: 2015-07-20.