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 A Problem

1.1.1 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.

The shared views of four research groups, including the prolific Geoffrey Hinton, is that DBNs are favoured for use in speech recognition tasks over other deep learning approaches [1]. For instance `Stacked Denoising Autoencoders` and `Deep Convolutional Networks`.

1.1.2 Aspirational Examples as a motivation of the project

Now some thought experiments to highlight a fundamental weakness in DBNs.

DBNs can capture rich non-linearity, making them ideal for tasks such as facial recognition, where pixels in the images change drastically depending on pose and illumination. These variations impede the use of classical feature based on geometry, such as Eigenfaces or Fischer-face [5]. There are at least two systems at play in an image of a face, the face itself that has a shape and colour, and the illumination which can vary greatly, casting shadows and highlighting parts of the face. Despite this, a DBN can achieve excellent classification performance on the UMIST dataset of 576 multi-viewed faced images of 20 individuals [5].

Consider another famous example that illustrates the idea of multiple sources acting independently to form some data, referred to as the `Cocktail Party Problem`. At a cocktail party many conversations are taking part at the same time, yet despite the cacophony, a partygoer is able to take part in their conversation, separating the sources.

Both these examples exhibit independent sources arising and interacting in a non-linear way to generate a visible artifact, be it an image of a face or a audio recording. This project does not address these aspirational examples, they set the scene for it's motivation. Despite a DBN's expressiveness, there is no way to extract these sources, the model instead learning the how to represent the combination. The DBN might fully well learn features that correspond to each source during it's training process, however the architecture or training algorithm make no attempt to enforce this.

1.1.3 Restricted Boltzmann Machines cannot separate sources either

DBNs are composed of a models with a shallow architecture, Restricted Boltzmann Machines (RBMs). RBMs provide a natural starting point for representing data causes by the combination of two sources.

RBMs are two layer, fully connected, unsupervised neural networks. RBMs do not model causation, instead they model dependancies.

Again using the example of images, an input image will map to a single representation. There lacks a mechanism for modeling sources that are acting independantly, instead modeling inputs as a single source.

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 independent 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 A proposed 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 Belief network and the Restricted Boltzmann Machine. Frean and Marsland also propose an algorithm to invert this proposed generative 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.

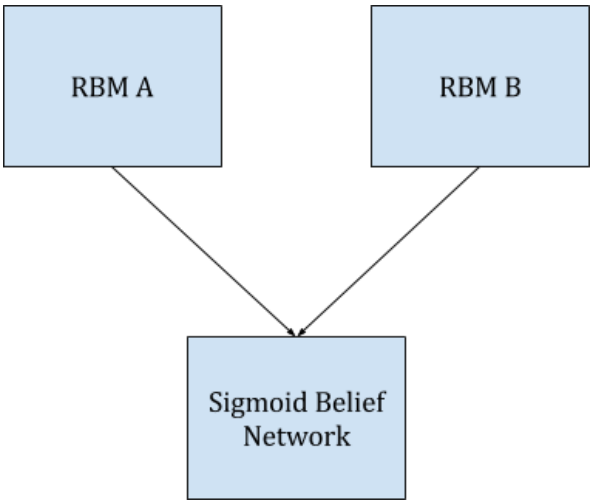


Figure 1.1: A graphical representation showing the proposed generative model for capturing two causes, the ORBM.

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

- TODO WORDING 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 and Contribution

This project provides a suite of evalautions, in the form of tests of increasing complexity to answer the above questions. The aim being to incrementally verify the new generative model and inference algorithm.

The tests give reasons for optimism, especially in the smaller cases. Challenges have been exposed in the evaluation that will need to be addressed going forward, to make the algorithm and model scalable.

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 Generative Models and Source Separation

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 and represent it. Input data, say images, could be mapped from raw values into some higher level features. Hinton gave a compelling argument why higher level features are desirable in the context of generative models [2].

> 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. In cases like this, it makes sense to start by using unsupervised learning to discover latent variables (i.e. features) that model the structure in the ensemble of training images.

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

2.1.1 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.

2.1.2 PGMs as a tool reasoning about generative models

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.

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. Throughout this report, RBMs, Sigmoid Belief Networks and the proposed ORBM will be shown in this format.

2.2 Sampling and inverting the model

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. 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.

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 algorithm introduced and evaluated in this report aims to invert the ORBM, to generate hidden representations given an input.

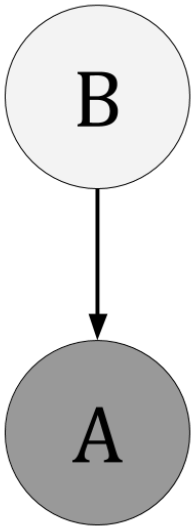


Figure 2.1: An example PGM, showing an observed variable 'A' and it's hidden cause 'B'.

2.2.1 Gibbs sampling, a subset of Markov Chain Monte Carlo

Gibbs sampling is a special case of Markov Chain Monte Carlo, 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.

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 independent. The process of taking a step between states is referred to as a `Gibbs iteration`.

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 more than one may be required.

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 likihood of samples being from the desired distribution.

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 [3], one Gibbs step was sufficient in practice for training and using an RBM. The ORBM is not so fortunate.

2.2.2 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).

2.3 An intractable model for causes

2.3.1 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.

2.3.2 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.

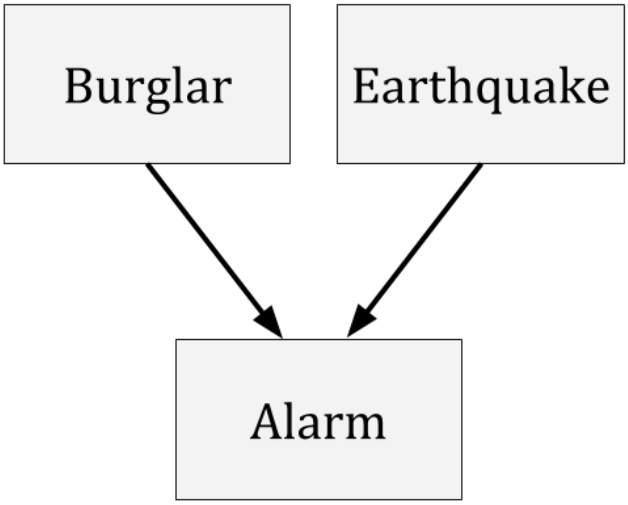


Figure 2.2: The famous Burglar, Earthquake, Alarm network showing a minimal case of explaining away.

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. Neal showed this by comparing the number of gibbs iterations required for small enough error rates in [6].

2.3.3 Boltzmann Machines

A Boltzmann machine TODO CITE: Cite the Harmonium, markov field and Hopfield for deterministic example... 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.

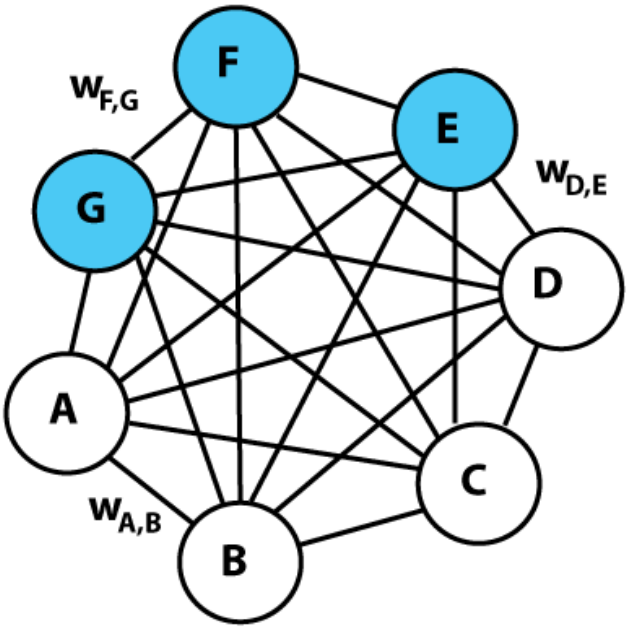


Figure 2.3: A Boltzmann Machine, the blue shaded nodes representing the observed variables, and the non-shaded nodes the latent variables.

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.

2.4 Restricted Boltzmann Machines: A Strong assumption

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 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 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`.

The effect of this restriction is inference 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 dependant on h_2 wether or not we know anything about the visible

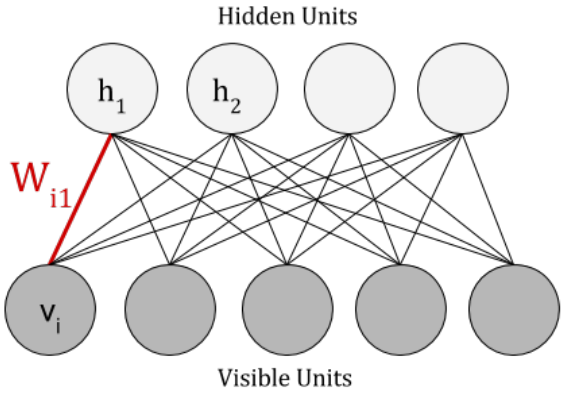


Figure 2.4: An example Restricted Boltzmann Machine with four hidden units, and five visible units. Note that the edges between units are not directed - representing a dependency not a cause.

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 but makes the RBM useable in practice. TODO CITE: The paper about Boltzmann Machines being really good when actually left to find solution.

2.4.1 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 $\tilde{h}$ and $\tilde{v}$ 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 [4], presented this energy as defined by the function as

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

The probability of the RBM being in a given configuration is the joint probability of $\tilde{h}$ and $\tilde{v}$.

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

If we move to log space that becomes

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

$\frac{1}{Z}$ 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. So we arrive at

$$P^{\star}(\tilde{h}, \tilde{v}) = \log \sum_{j,i} h_j W_{ji} v_i$$

This can be illustrated by re-examining how we represent the joint distribution in this restricted architecture, in log space

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

2.4.2 Gibbs Sampling in RBMs

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.

To perform inference, create reconstructions, and train RBMs, Gibbs sampling is used to draw samples from $P(h|v)$ (referred to as sampling from the posterior) and $P(v|h)$ respectively. It is introduced here to be reference later in Gibbs in an ORBM.

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 process can be calculated in one step for all units in the target layer as they are independant on one another. The process of updating the hidden, then visible layers forms what is referred to as the `Gibbs Chain` TODO CITE: Feel like I could cite this again in like a CD paper or something. This process is visualised at a layer level in figure 2.5

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 is: TODO CITE: Gibbs sampling would be good here.

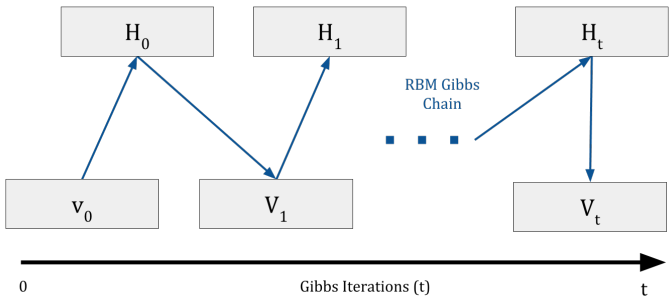


Figure 2.5: A figure illustrating a Gibbs chain where left to right indicates a Gibbs iteration. Note this is not a PGM.

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

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 2.6 illustrates psi for an example RBM.

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

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

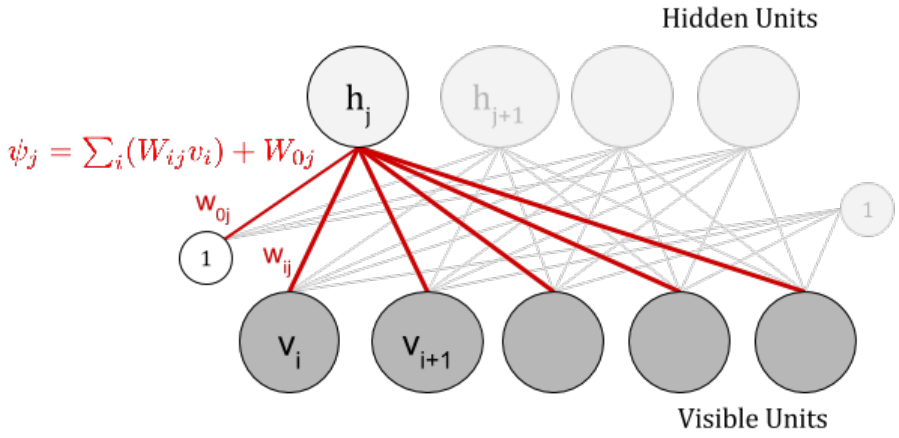


Figure 2.6: A diagram showing ψ_j, the weighted sum into the jth hidden unit. Note that W_{oj} is the hidden bias, represented as a unit that is always on with a weight into each hidden unit.

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 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.

2.4.3 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)

2.4.4 Inference

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

2.4.5 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).
- We can train a classifier on the RBMs hidden representation. This can be compared for a ORBM and RBM.

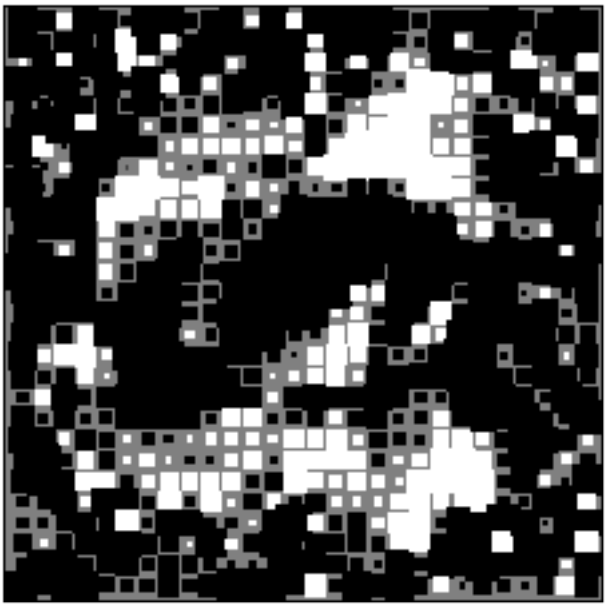


Figure 2.7: Good Hinton Diagram

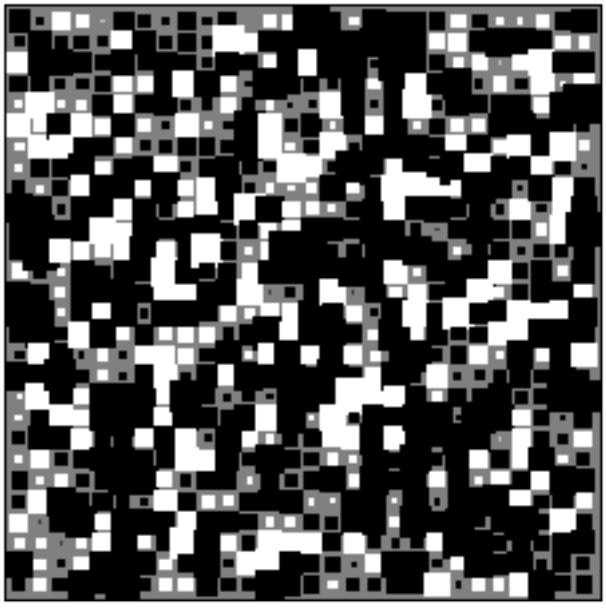


Figure 2.8: Bad Hinton Diagram

2.5 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. TODO WORDING Introduce the idea of plug and play here
- Like the RBM leveraged in the ORBM, it is difficult to evaluate, But similar techniques can be leveraged

2.5.1 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.

2.6 Inference In the ORBM

2.6.1 The Gibbs Chain

2.6.2 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 2.5.

Note the weights between the H and U layer are shared between the V and H layers.

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

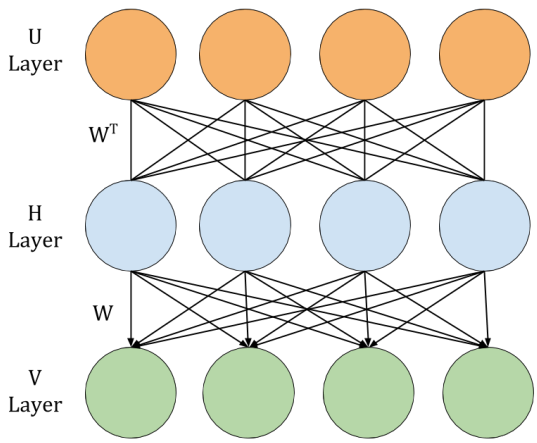


Figure 2.9: A diagram illustrating 'unrolling' an RBM by one Gibbs iteration.

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.

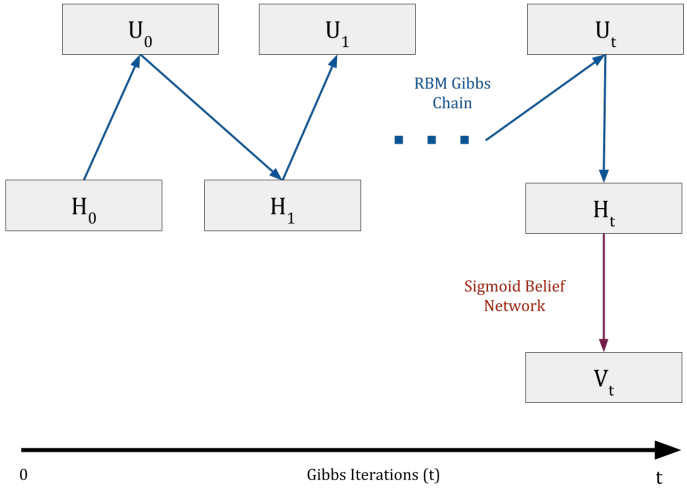


Figure 2.10: A diagram showing Ancestral sampling in the equivalent network, where normal sampling in the top 2 layers is performed until Gibbs iteration t the hidden state is pushed through the bottom layers Sigmoid Network.

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 **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.

2.6.3 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.

2.6.4 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.

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

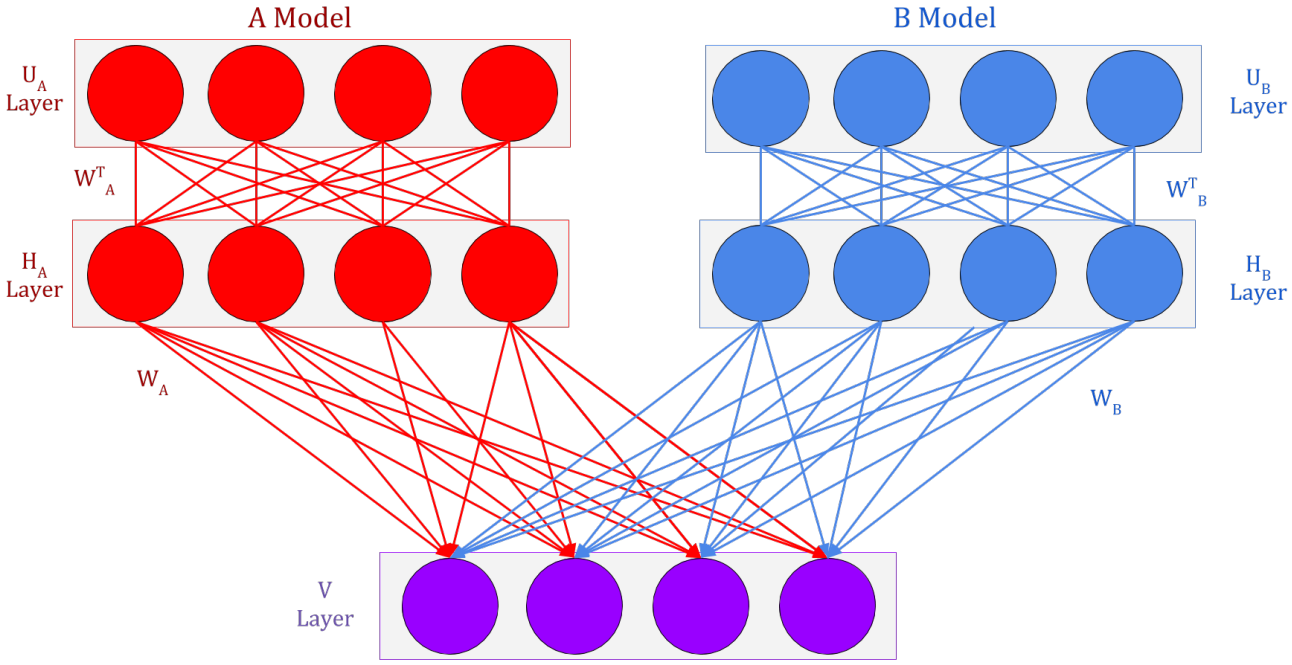


Figure 2.11: The full ORBM architecture, where A and B are the two causes that combine to form the data.

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...

2.6.5 Approximating the Correction

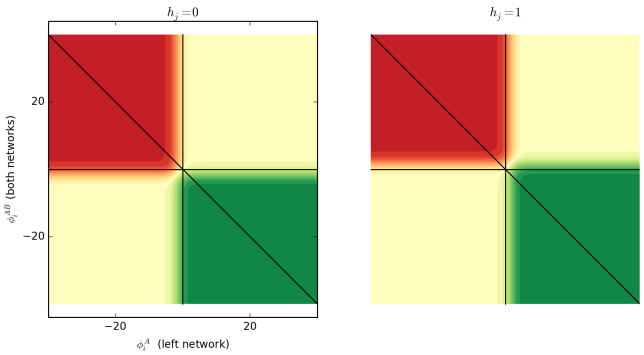


Figure 2.12: A diagram what the correction function looks like for ...

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 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.

3.1.2 Implementation Design

- Sampler, Trainer, RBM concepts for plug and play.
- Library choice meant efficiency with benefits of python for quick dev

- Easy to deploy to grid (versus java higher risk, lack of expeirence)
- Ordering of evaluation tasks made unit testing, with hand made test cases possible. Also less risk, remove the uncertainty around the RBM.

Chapter 4

Evaluation

4.1 Evaluation Design

4.1.1 Mixing Time Evaluation

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

4.1.2 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.

4.2 Evaluation Methods

My TODO WORDING x experiements all followed the roughly the same process, ultimately working from trivial cases to more challenging tasks. The reasoning behind this was that trivial cases make unit testing feasible, therefore ensuring conclusions can be drawn with regard to the algorithm and model, not an incorrect implementation.

The following evaluations aimed to evaluate the ORBM and it's inference algorithm by examining reconstructions given a multi-cause input. An example of a muli-cause input with two quadrilaterals is shown in figure 4.1. The optimal reconstructions are equivalent to the two images that combined form the input.

In the smaller dimenisonal cases the reconstructions are inspected by hand, manually plotting the reconstructions and the frequency with which they occurred after a large amount of repetitions. In the larger dimensional tasks, two 'scores' were used to evaluate reconstructions against the training set.

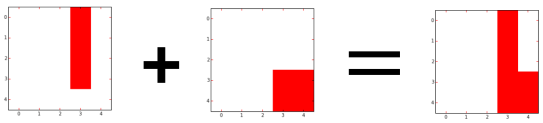


Figure 4.1: A figure illustrating two five by five pixel images combining to form a composite/multicause input. The images are binary.

To be able to use this reconstruction based evaluation, knowledge of the ground truth was required. This was so:

h

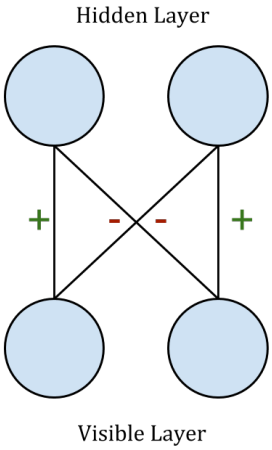


Figure 4.2: The two bit RBM for modelling a single bit on at a time in a two bit pattern. The diagonal weights are negative and the non-diagonal weights are positive.

- RBMs could be trained and then plugged into the ORBM network.
- Reconstructions could be compared directly (by score) to the ground truth.

4.3 Starting from the minimal case, two bit XOR

Description

The starting point for the evalatuion was examining the ORBM reconstructions in a two bit pattern, where two of the same model were combining to form the data. This model made one bit on in a two bit pattern at a time. That is it could either make $[1,0]$ or $[0,1]$. The training set is the two bit XOR truth table.

Architecture and Parameters

Figure 4.3: Example reconstructions for the Hand-crafted RBM, For 10000 independant reconstructions given the input $[1,0]$.

Being a trivial dimensionality, the RBMs weights were constructed by hand, meaning that only the ORBMs inference algorithm was being evaluated. This network is picture in figure 4.2, TODO WORDING with weights greater than 5 the networks stable state results in either $[1,0]$ or $[0,1]$. The RBMs had 2 hidden units and 2 visible units.

Method

This RBM was checked to ensure that it behaved in practice

Visible Input	Expected Reconstructions	
$[1,1]$	$[1,0]$	$[0,1]$
$[1,0]$	$[1,0]$	$[1,0]$
$[0,1]$	$[0,1]$	$[0,1]$

Table 4.1: A Table showing the expected reconstructions from performing ORBM inference with various input patterns. The left and right hand column of the Expected Reconstructions columnn indicate the reconstructions from the left and right RBMs in the ORBM.

by ensuring it's reconstructions matched the input for a large frequency of reconstructions. The results of trying this with the input $[1,0]$ are showing in figure 4.3. The most common reconstruction is $[1,0]$, which matches the input. We see that 500 reconstructions out of the $10,000$ reconstructions were incorrect, corresponding to $[0,1]$. This is due to the stochastic nature of the RBM.

In a similar way to the reconstructions, ancestral samples can be taken from the model and evaluated. TODO WORDING In larger dimensions Hinton has shown RBMs to be poor generative models without the extra layers above... however in the small dimensions of this task the dreams should match the training set:

A histogram of frequencies of reconstructions given the RBM is in the free phase, is shown in figure 4.4. Again the model behaves as expected, generating dream patterns that match the training dataset the majority of the time.

Given a good model, the XOR RBM was duplicated and plugged into the ORBM structure as picture in figure 4.5.

The inference algorithm was run in the ORBM architecture for various visible inputs, giving two hidden vectors for the two representations h^A and h^B. For each pair of hidden vectors, a reconstruction was created. This process was repeated in a similar way to how the reconstructions were evaluated in the lone RBM, counting the frequency each reconstruction occurs over TODO WORDING a number here many runs,

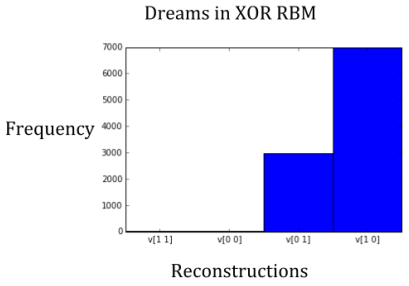
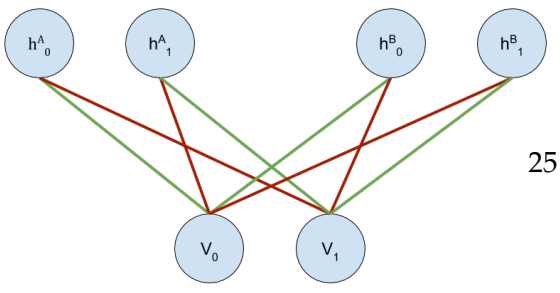


Figure 4.4: Dreams in the XOR RBM, note how only the training data is present.

XOR ORBM Analysis

The results of this process are shown in figure 4.6. The ORBM is able to separate the causes, as the model is being duplicated, it produces $[1,0]$ and $[0,1]$ roughly half the time. The reconstructions are of the form v_a$[x,y]$ v_b$[n,m]$ where $[x,y]$ is the reconstruction created model A and $[n,m]$ is the reconstruction created by model B.

These reconstructions were compared to one of the RBMs trained to recognised a single pattern being on in two bits. As expected a machine that has been trained to recognise one bit,



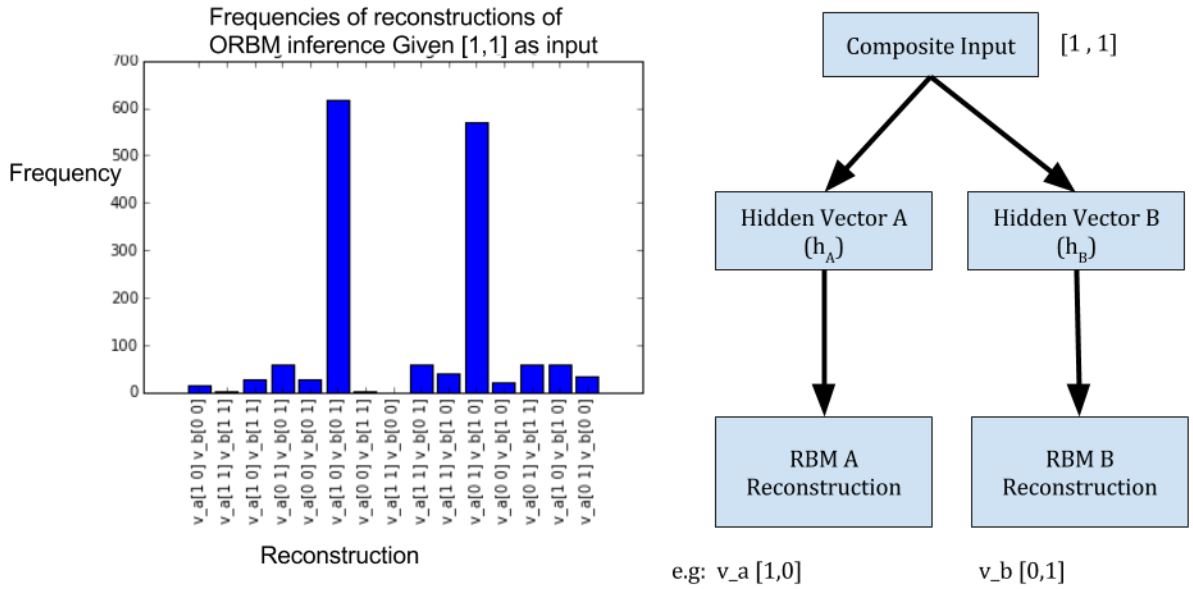


Figure 4.6: A figure with the results of running ORBM inference 1000 times, showing the frequency for which a given reconstruction occurred. The process is shown in the diagram on the right hand side.

has no concept of two bits being on and hence the reconstructions show no mechanism for seperating the sources. This is illustrated in figure 4.8.

The results of repeating this process with the input $[1, 0]$ yielded successful results. We would hope that ORBM architecture could also handle inferenece with just a single subject. The results of this are shown in figure 4.8.TODO WORDING Do the markers care that it works in this case... it's so trivial does it even mean anything for larger images..

Two bit XOR Conclusion

The ORBM was able to extract the target patterns $[1, 0]$ and $[0, 1]$ given the input $[1, 1]$.

4.4 Y bit pattern, X neighbouring bits on

Description

A natural next step from a single bit on in a 2 bit pattern, is moving up to X bits side by side on in a Y bit pattern is a next step. For example if $Y = 4$ and $X = 2$ then a valid inputs are the rows of the following:

$$dataset = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

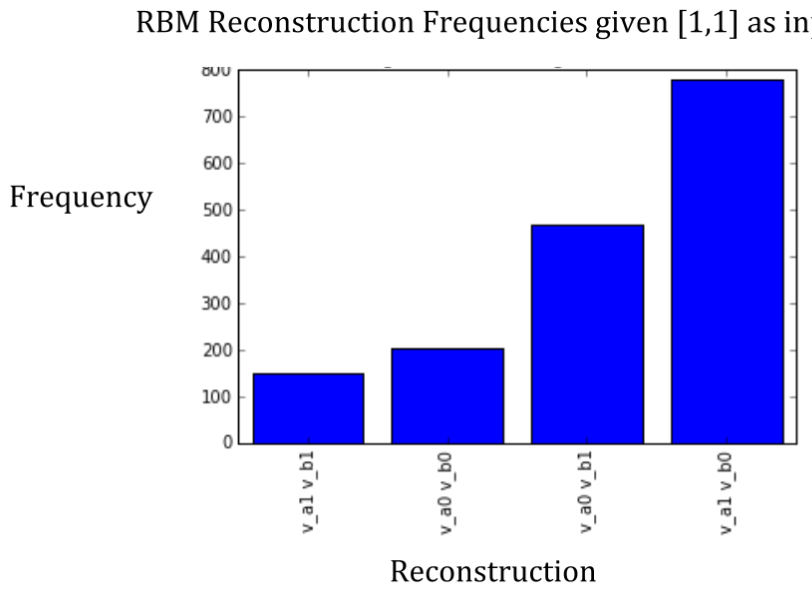


Figure 4.7: A figure showing the result of generating 1000 reconstructions with an RBM. It is clear the RBM has no mechanism to separate the sources. Hence showing it [1,1] when it has only been trained on XOR results in no separation, i.e. reconstructions including $[1, 1]$.

Architecture and Parameters

- $|Y|$ Hidden units per RBM
- Random starting weights. RBM trained with 1000 epochs and a learning rate of 0.002.
- A sample of 10,000 dreams were sampled from the RBMs, and then plotted on a histogram in a similar fashion to figure **??**. The frequency of these dreams should be roughly equal and the dreams should directly match the training set. This is feasible as allow possible data items are known.

Method

For values of Y where $y \epsilon Y | 2 < y < 7$ and all values of X where $x \epsilon X | 2 < x < y < 7$ The process below is repeated.

The inference algorithm was run for 1000 reconstructions in the ORBM architecture for all unique compositions of the training set. The result was 1000 reconstructions per composition of the dataset above. TODO WORDING Need a way to describe the dataset a bit better.

TODO CITE: Need to grab the images out of the ipython notebook.

4.5 2D Patterns in a small dimensional space

- Dataset:
  - 2x2 square in a 5x5 image. Dataset of every possible configuration of said square.

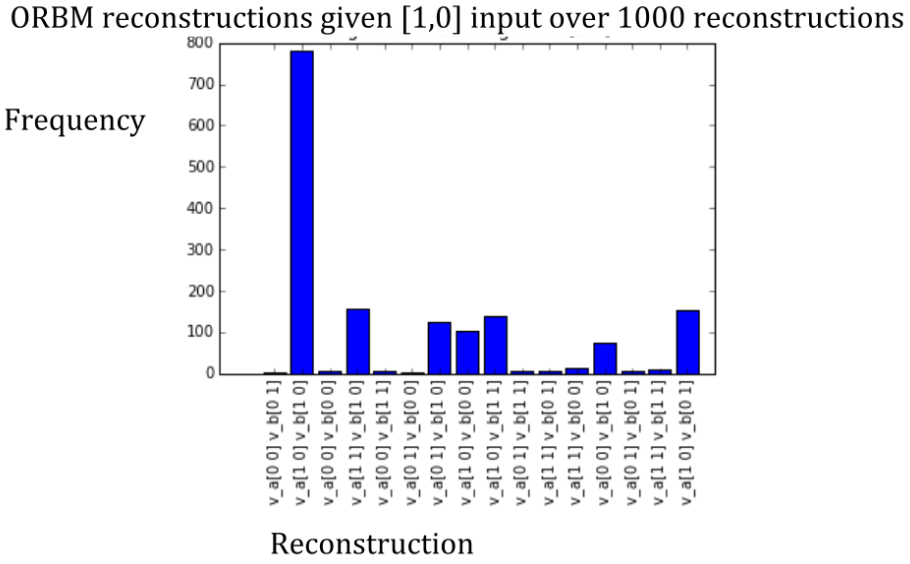


Figure 4.8: A figure showing the result of generating 1000 reconstructions with the ORBM with $[1, 0]$ as input.

- 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.

4.6 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.

4.7 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.

Bibliography

[1] HINTON, G., DENG, L., YU, D., DAHL, G., RAHMAN MOHAMED, A., JAITLY, N., SENIOR, A., VANHOUCKE, V., NGUYEN, P., SAINATH, T., AND KINGSBURY, B. Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine* (2012).

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

[3] 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.

[4] 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.

[5] LIN, M., AND FAN, X. Low resolution face recognition with pose variations using deep belief networks. In *Image and Signal Processing (CISP), 2011 4th International Congress on* (Oct 2011), vol. 3, pp. 1522–1526.

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