Information Component Analysis For Facial Images

Danny Wood

A dissertation submitted to The University of Manchester for the degree of Master of Science, in the Faculty of Engineering and Physical Sciences

2014

School of Computer Science

Contents

Word Count: 22,406

List of Figures

List of Tables

Abstract

This project aims to explore the state of the art in unsupervised representation learning for facial images. Facial images contain a great deal of information about their subjects, however, these *information components* are often entangled in highly non-linear ways. As such, disentangling information components can be a very difficult task. This project considers three models presented in the literature: the Gaussian Restricted Boltzmann Machine (GRBM), the spike-and-slab Restricted Boltzmann Machine (ssRBM) and the higher order spike-and-slab Restricted Boltzmann Machine (hossRBM), aiming to serve as a comprehensive guide to these models, as well as to show that they make it possible to learn representations, in an unsupervised fashion, in which information components become disentangled.

In experiments, promising results were obtained using both the Gaussian restricted Boltzmann machine and spike-and-slab restricted Boltzmann machine to learn meaningful representations of facial images. Meaningful representations of images utilising the higher order spike-and-slab restricted Boltzmann machine were not achieved due to instabilities of the model which occur during training. Possibile reasons for this instability are explored in the project, and attemps to overcome them using several different approaches are demonstrated and discussed.

Declaration

Intellectual Property Statement

Acknowledgements

First and foremost, I would like to thank my supervisor, Dr Ke Chen. I'm very grateful for the his guidance and advice during every stage of this dissertation. In particular, his ability and willingness to provide thorough feedback on difficult questions via email, often within minutes of being asked, proved invaluable in the completion of this project.

I would like express my gratitude to Guillaume Desjardins for making his code available at my request, as well as sharing some valuable insights on problems I had all but given up on.

Thanks also to my parents, and sister Kirsty, for their support and encouragement throughout this year.

Finally, special thanks to Becky Atkinson for proofreading early drafts of several chapters. Readers can rest assured that any typos still found within this text were added afterwards.

Chapter 1

Introduction

1.1 Description of Problem

When deciding on a representation for data to be used in machine learning tasks, it is important to consider that structures within the data may be the product of many interacting factors, or information components [2, 3]. For instance, a set of facial images may contain photographs taken of many different individuals, making different facial expressions and under different lighting conditions. The distribution of intensity values for each individual pixel is conditionally dependent upon all of these factors.

In many cases, not all of the available data is categorised by the factors of interest; as such, unsupervised learning techniques must be used in order to extract meaning from the data. Unsupervised learning methods often serve as a form of pre-processing for data in a classification pipeline [4], with the success of the classification algorithm determined by the ability of the unsupervised learning method to disentangle information components within the data [3, 5].

This is especially important when the information component of interest is not dominant in the raw pixel intensity based representation. In the example of facial images, the variation caused in the raw pixel representation due to difference in identity far outweigh those caused by difference in facial expression, and as such facial expression recognition on this representation is likely to yield poor results [2]. This is the problem which will be considered in this project: the problem of finding a representation of facial images in which the facial expression of the subject of an image becomes disentangled from the images more prominent features.

In the field of representation learning many different techniques have been attempted to solve problems of this nature. One particularly promising technique is the use of probabilistic graphical models. It is a subset of these models, restricted Boltzmann machines (RBMs), which are the subject of this project.

1.2 Aims and Contributions

It is hoped that this project will serve as a useful contribution to the literature in representation learning. To this end, the two primary objectives of this project are:

1. To act as a thorough and complete reference work on the subject of Restricted Boltzmann Machines (RBMs), including Gaussian (GRBM), spike-and-slab (ssRBM) and higher order spike-and-slab (hossRBM) variants. Ideally, it should be possible for someone with a solid knowledge of linear algebra, calculus and statistics to use this material in order to gain a reasonably comprehensive understanding of the models presented despite no previous knowledge of machine learning or graphical models.

2. To demonstrate the advantages of probabilistic models for learning representations of facial images. In particular, the aim is to prove that hossRBMs successfully disentangle information components of facial images, learning a representation which increases the accuracy of a linear classifier when classifying images by facial expression. Successfully demonstrating this requires that a significant improvement against both the raw pixel representation and representations learnt by other models can be achieved. In this way, this project aims to serve to replicate and confirm the findings of [6] and [2].

1.3 Structure of Project

The structure of this dissertation is as follows:

The first chapter gives a brief introduction to the problem domain and describes the aims and structure of the project.

The second chapter gives a brief overview of Markov Random Fields, using their properties to justify the designs of the models used in the project. The RBM, GRBM, ssRBM and hossRBM are introduced, with a strong emphasis on describing their structure and providing commentary on the motivations for any design choices. The concepts of sparsity and regularisation are also introduced along with a discussion of how they can be used when training RBMs.

The third chapter gives a description of how the models described in the previous chapter are trained. The topics of stochastic gradient descent, Gibbs sampling and contrastive divergence are discussed in relation to training RBMs.

The fourth discusses the details of the implementation used in the experiments completed in the course of the project. An overview is given of the major technologies used, noting any differences between the models described in the previous chapters and the implementations.

The fifth chapter is concerned with the main experiment conducted for this project. This experiment involves training several models on facial images. The representations learnt by each model are used as the input for a linear SVM, along with facial expression labels.

The sixth chapter provides a discussion of the outcomes of the project, presenting a more personal, informal reflection on the successes and failures of the project. This chapter also contains a discussion of possible directions of future work based on the outcomes of the project.

Finally, the seventh chapter provides a summary of the project as a whole and gives a few concluding remarks.

Chapter 2

Restricted Boltzmann Machines

This chapter provides an introduction to the Restricted Boltzmann Machine (RBM). The first section of the chapter provides a brief context and background for the model, exploring the current state of representation and discussing the advantages and disadvantages of various approaches currently being explored in the field. The next section discusses energy based graphical models in an abstract way, with the remaining sections describing the RBM itself, along with several variations of the model.

2.1 Representation Learning

Representation learning is described by Bengio et al. as "[The process of] learning transformations of data that make it easier to extract useful information when building classifiers or other predictors" [3]. The earliest representation learning techniques, such as Principal Component Analysis (PCA), were concerned primarily with dimensionality reduction [7]. In the case of PCA, the aim was to find a lower dimensional representation which maintained as much variance from the original data as possible, built on the assumption that directions of high variance corresponding to interesting structure in the data [8]. Such models made no attempt to separate explanatory factors within the data but could still prove useful in the preparation of data whose dimensionality prohibits the learning of more expanatory representations [3].

In their review, Bengio et al. outline three of the most popular approaches taken to representation learning: auto-encoders, in which the training algorithm is based on reducing the error in the reconstruction of training data from the representation; manifold learning, in which the emphasis is placed on finding representations which describe the data as being embedded in a lower dimensional manifold in the feature space and probabilistic models, which have the goal of learning the underlying distribution from which the training data is sampled [3].

Whilst all three methods have produced state-of-the-art classification results in recent years (auto-encoder: [9], manifold: [10], probabilistic: [11]), this project will be concerned solely with probabilistic models. In particular, it will explore an interesting approach introduced in [2], in which a probabilistic model, the higer order spike-and-slab restricted Boltzmann machine (hossRBM) is constructed with the primary purpose of learning to disentangle information components in an unsupervised manner.

In recent years a great deal of focus has been placed on *deep learning*, the process of learning multiple representations in a hierarchical manner, with each layer of features being more abstract from the raw data than the one before it. Success of deep learning architectures is very much dependent upon the models used in each layer, as such the investigation of novel approaches in single layer models is still an important area of research. Due to the increased difficulties involved in training deeper models, only single layer constructions will be considered within this project. It should, however, be noted that the application of the hossRBM to deep learning has been described in the original paper as “an important area of future research” [2].

2.2 Energy Based Graphical Models

Facial images are very complex, and can vary greatly in many ways. However, the set of all possible facial images is highly structured. Separating out different information components, such as identity and facial expression, is essentially the task of learning and understanding this structure. With no prior knowledge of the structure of the domain, a fruitful strategy is to attempt to learn the underyling distribution based on a finite number of training examples.

A powerful technique for modelling such a distribution is the use of hidden variables [12]. Using this approach, models are devised which consist of two sets of variables: v, the set of visible or input variables, and h, the set of hidden or latent variables. Alongside these variables are a set of parameters θ, which are chosen using algorithms based on the training data. Many such models easily lend themselves to graphical representation, with one such example being the Markov Random Field (MRF) [12].

The MRF is a model which was originally devised for use in statistical mechanics [13]. In particular, it was created as an attempt to construct a general probabilistic framework in which the random interactions of dipoles within ferromagnetic materials could be explained [13]. In order to succinctly define and describe the MRF, several definitions from graph theory must first be given [12, 14].

Definition 2.1. *A graph $G = (V, E)$, is a tuple consisting of a set of nodes V and a set of undirected edges E, where each edge is defined as an unordered pair of nodes $\{v_i, v_j\} \in E$.*

Definition 2.2. *Two nodes v_i and v_j are neighbours if $\{v_i, v_j\} \in E$. the neighbourhood of v, $\mathcal{N}_v$, is definied as the set of nodes connected to v, i.e. $\mathcal{N}_v = \{w \in V : \{w, v\} \in E\}$*

Definition 2.3. *A clique on a graph $G = (V, E)$ is a subset of V in which all pairs of distinct nodes are connected by a node in E. A clique is maximal if a larger clique cannot be made by adding more nodes from V.*

In a Markov random field, a random variable X_v is associated with each node in V, with X being the set of all such variables. For a clique $C \subset V$, let X_C denote the set of random variables associated with the nodes in C.

Definition 2.4. *A graph G along with associated random variables X forms a Markov random field if the probability distribution of X satisfies the following property for all v:*

$$P(X_v | X \setminus \{X_v\}) = P(X_v | \mathcal{N}_v)$$

That is, the distribution of X_v given the values variables on neighbouring nodes, is independent of all other variables.

It is now almost possible to introduce the concept of an energy function, which will be very important in the construction of models defined later on. However, it is first necessary to state a result known as the *Hammersley-Clifford Theroem*.

Theorem 2.5. *Let G be a graph with associated random variables X, let all variables in X have a strictly positive distribution, and let $\mathcal{C}$ be the set of all maximal cliques of G. The graph is a Markov Random Field if and only if there exist a set of non-negative functions $\{\psi_C\}_{C\in\mathcal{C}}$ such that ψ_C is dependent only on variables in C and*

$$p(X) = \frac{1}{Z}\prod_{C\in\mathcal{C}}\psi_C(X). \tag{2.1}$$

The normalisation constant Z, known as the partition function, is defined as

$$Z = \sum_X \prod_{C\in\mathcal{C}}\psi_C(X). \tag{2.2}$$

The proof of this theorem is beyond the scope of this project, but can be found in [15]. The alternative definition of an MRF this theorem gives is incredibly useful, as it yields the following implicit definition of the probability distribution:

$$\begin{aligned} p(X) &= \frac{1}{Z}\prod_{c\in\mathcal{C}}\psi_C X_C \\ &= \frac{1}{Z}\exp\left(\sum_{C\in\mathcal{C}}\ln\psi_C(X_C)\right) \\ &= \frac{1}{Z}\exp(-E(X)) \end{aligned} \tag{2.3}$$

Where $E = \sum_{C\in\mathcal{C}}\ln\psi_C(X_C)$ is the energy function. It is easy to see that this function behaves as desired; improbable outcomes on X correspond to high values of the function E, whereas low values will be induced by more probable configurations. Another important aspect of this derivation is that it allows for the possibility that, with some care, a desirable energy function can be constructed first, and then an MRF induced from this function [12].

An important observation is that the energy function defines a probability distribution on the possible configurations of variables in the network. If it is possible to create a function between a set of features of some dataset and configurations of the network, then the network can effectively assign a probability to examples from the dataset. The training of a model is the process of tuning the energy function to achieve a distribution approximating the distribution from which the training data is sampled from; this process is the subject of Chapter 3. Interestingly, once this has been achieved, the learnt distribution can be sampled from, effectively allowing new data to be generated. For this reason, this kind of model is sometimes referred to as a *generative model* [16].

2.3 The Restricted Boltzmann Machine

Let $v = \{v_1, v_2, \ldots, v_D\}$ and $h = \{h_1, h_2, \ldots, h_N\}$ be sets of binary random variables, arranged in as shown in Figure 2.1. It can be seen that, if this graph is an MRF, as will be proved shortly, all visible nodes are independent given the hidden nodes, and vice-versa. The joint probability distribution of the restricted Boltzmann Machine over these two sets of variables is given by the Gibbs distribution $p(v,h) = \frac{1}{Z}e^{-E(v,h)}$ with energy function:

$$E(v,h) = -\sum_{i=1}^{N}\sum_{j=1}^{D} W_{ij}h_i v_j - \sum_{j=1}^{D} b_j v_j - \sum_{i=1}^{N} c_i h_i, \tag{2.4}$$

At this point it is instructive to show that the RBM is an MRF. The approach usually taken in the literature is to explicitly demonstrate conditional independences by calculating the marginal distributions of hidden and visible variables [14, 17]. An alternative approach attempted here is to use the energy function to derive the potential functions of all cliques on the graph, then it can be claimed that the graph is an MRF using the Hammersley-Clifford theorem [14]. Whilst the first of these methods is the more useful, as it leads to the marginal distributions of individual nodes, the second will also be demonstrated for this model as a useful example of how a distribution can be made to factorise over an MRF.

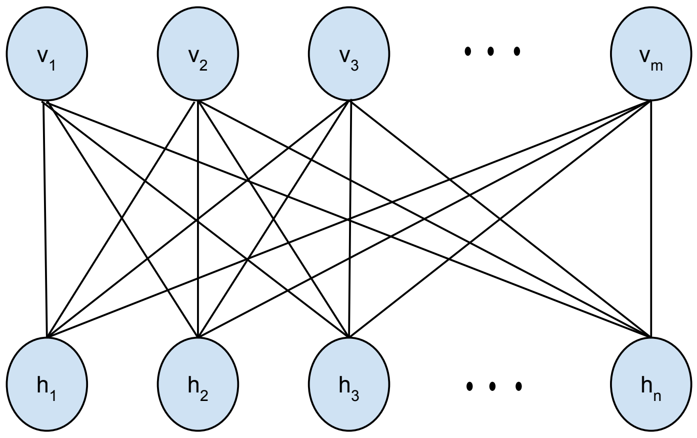


FIGURE 2.1: The Graphical Structure of the Restricted Boltzmann Machine

Approach 1

In order to show the RBM is an MRF it is necessary to show that the hidden variables are conditionally independent of each other, and similarly for the visible variables. This can be demonstrated directly from the joint distribution, by showing that the probability of a particular outcome in one variable is not dependent on variables of the same type. In doing so, the marginal distributions of the hidden and visible units are also obtained.

The approach to this calculation shown below is taken from [14]. This approach turns out to be very powerful, and will be used in many calculations performed for more sophisticated models. For this reason, an attempt is made to present the derivation in as general a way as possible before applying it to the specific energy function of the RBM. For a Markov Random Field with a set of visible variables v and a set of hidden variables h, define h_{-l} be the set of all hidden units except h_l. Define $\alpha_l(v)$ as the sum of the coefficients of h_l in all terms in the energy function which have h_l as a factor, with $\beta(v, h_{-l})$ being the sum of all the remaining terms.

Using these definitions, the energy function can be rewritten as

$$E(v,h) = \beta(h_{-l}, v) + h_l \alpha_l(v). \tag{2.5}$$

The advantage of this definition is that it separates out all terms which are dependent on h_l. This allows $p(h_l = 1 | h_{-l}, v)$ to be calculated as:

$$\begin{aligned}
p(h_l = 1 | v, h_{-l}) &= \frac{p(h_l = 1, h_{-l}, v)}{p(h_{-l}, v)} \\
&= \frac{\exp(-E(h_l = 1, h_{-l}, v))}{\exp(-E(h_l = 1, h_{-l}, v)) + \exp(-E(h_l = 0, h_{-l}, v))} \\
&= \frac{\exp(-\beta(h_{-l}, v) - 1 \cdot \alpha_l(v))}{\exp(-\beta(h_{-l}, v) - 1 \cdot \alpha_l(v)) + \exp(-\beta(h_{-l}, v) - 0 \cdot \alpha_l(v))} \\
&= \frac{\exp(-\beta(h_{-l}, v)) \exp(-\alpha_l(v)))}{\exp(-\beta(h_{-l}, v))(\exp(-\alpha_l(h_{-l}, v)) + 1)} \\
&= \frac{1}{1 + \exp(\alpha_l(v))} \\
&= \mathtt{sigm}(-\alpha_l(v))
\end{aligned} \tag{2.6}$$

Applying this principle to the energy function of the restricted Botlzmann machine gives the definitions of $\alpha_l(v)$ and $\beta(v_{-l}, h)$ as:

$$\alpha_l(v) = -\sum_{j=1}^{D} W_{lj} h_i - c_l$$

and

$$\beta(v_{-l}, h) = -\sum_{i=1, i \neq l}^{N} \sum_{j=1}^{D} W_{ij} h_i v_j - \sum_{j=1}^{D} b_j v_j - \sum_{i=1, i \neq l}^{N} c_i h_i$$

leading to a conditional probability of:

$$
\begin{aligned}
p(h_l|h_{-l}, v) &= \texttt{sigm}(-\alpha_l(v)) \\
&= \texttt{sigm}\left(\sum_{j=1}^{D} W_{lj}v_j + c_l\right) \\
&= p(h_l|v) \qquad (2.7)
\end{aligned}
$$

Here $\texttt{sigm}(x)$ is the sigmoid function, defined as $\texttt{sigm}(x) = \frac{1}{1+e^{-x}}$. From the equalities shown above, it is apparent that the variables in the hidden layer are independent given the visible layer. For a binary hidden variable h_i, the probability $p(h = 1)$ is sometimes referred to as the activation probability of the variable [18].

Furthermore, due to symmetries in the function, $p(h = 1|v)$ can be derived using nearly identical steps, yielding the result:

$$
p(h_i = 1|v) = sigm\left(\sum_{j=1}^{D} W_{ij}v_j + c_i\right) \qquad (2.8)
$$

As well as proving independence, the above gives the conditional probability distributions which will be necessary later for sampling from and training the model.

Approach 2

In order to determine a set of potential functions over cliques within the RBM graph, it is first necessary to identify the cliques within the graph. Consider a node v_j with $1 \leq j \leq D$, for each hidden node h_i with $1 \leq i \leq N$ the set $\{v_j, h_i\}$ forms a clique, since there is an edge connecting the two nodes. Furthermore, since the only edges in the graph consist of one visible and one hidden node, this clique is necessarily maximal (no other node can share an edge with both a hidden node and a visible one). From this it can be deduced that there are $n \times m$ cliques, one for each hidden/visible pair.

From the identity $E = \sum_{C\in\mathcal{C}} \ln \psi_C(X_c)$ an obvious approach to take is to devise a set of functions which sum to the energy function, each one dependent only on nodes within one clique, and then exponentiate these functions to get the actual potential functions. One way of doing this is:

$$\begin{aligned} & \ln \psi_{\{v_j,h_i\}}(v,h) = -W_{ij}v_jh_i - \frac{1}{D}b_jv_j - \frac{1}{N}c_ih_i \\ \Rightarrow \quad & \psi_{\{v_j,h_i\}}(v,h) = \exp(-W_{ij}v_jh_i - \frac{1}{D}b_jv_j - \frac{1}{N}c_ih_i) \end{aligned} \tag{2.9}$$

In this solution each clique $\{v_j, h_i\}$ is responsible for a single element in the first sum in Equation 2.4 and a fraction of a single element of each of the other two terms. Note that this is not a unique solution, in fact there are many possible sets of potential function for a given MRF [19].

2.4 Gaussian Visible Units

A disadvantage of the RBM as previously defined is the restriction to binary input. This is not ideal for the processing of natural images, which consist of pixels which can take a range of integer values, and therefore cannot be represented by binary units without a lot of data being lost. A simple trick for extending the RBM to deal with real valued visible units is simply to scale the training data to fit in an interval of unit length, and take the value of $p(v_j = 1|h)$ as the variable state, rather than a measure of the binary distribution.

A more nuanced approach is the use of *Gaussian visible units*. Using this method the energy function becomes: [18]

$$E(v,h) = -\sum_{i=1}^{n}\sum_{j=1}^{m}\frac{W_{ij}h_iv_j}{\sigma_j} - \sum_{j=1}^{m}\frac{(b_j - v_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n}c_ih_i,$$

Here each σ_j is an additional parameter representing the standard deviation of the Gaussian noise of the corresponding visible unit [18]. Similarly, the b_j variables correspond to the mean of the visible units v_j. In this way, these additional parameters serve to scale and translate the visible variables, meaning that the model can weight the influences of different features in a meaningful way.

The derivation of the conditional probability $p(h_l|v)$ is almost unchanged, requiring only trivial modifications to approach shown in Equation 2.6 to include the $\frac{1}{\sigma}$ term in the α_l function [1].

$$p(h_l = 1|v) = \mathtt{sigm}\left(\sum_{j=1}^{D} \frac{W_{lj} v_j}{\sigma_j} + c_l\right) \tag{2.10}$$

The derivation of $p(v_l|h)$, however becomes slightly more involved. The following derivation is taken from [20].

$$\begin{aligned}
p(v|h) &= \frac{exp(-E(v,h))}{\int_u \exp(-E(u,h))du} \\
&= \frac{exp(-\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{W_{ij}h_i v_j}{\sigma_j} - \sum_{j=1}^{m} \frac{(b_j - v_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n} c_i h_i)}{\int_u exp(-\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{W_{ij}h_i u_j}{\sigma_j} - \sum_{j=1}^{m} \frac{(b_j - u_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n} c_i h_i)du} \\
&= \frac{exp(-\sum_{i=1}^{n}\sum_{j=1}^{m} \frac{W_{ij}h_i v_j}{\sigma_j} - \sum_{j=1}^{m} \frac{(b_j - v_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n} c_i h_i)}{\prod_{j=1}^{m} \sigma_j\sqrt{2\pi}\left(\exp(\frac{1}{2}(\sum_{i=1}^{n} h_i w_{ij})^2) + \sum_{i=1}^{n} c_i h_i + \frac{1}{\sigma_j} b_j \sum_{i=1}^{n} h_j W_{ij}\right)}
\end{aligned}$$

[1]The β function also changes in order to ensure that Equation 2.6 still holds. However, this is not important, since it is made to disappear using the trick in Equation 2.6.

$$= \frac{\prod_{j=1}^{m} exp(-\sum_{i=1}^{n} \frac{W_{ij}h_i v_j}{\sigma_j} - \frac{(b_j - v_j)^2}{2\sigma_j^2} - \sum_{i=1}^{n} c_i h_i)}{\prod_{j=1}^{m} \sigma_j \sqrt{2\pi} \left(\exp(\frac{1}{2}(\sum_{i=1}^{n} h_i w_{ij})^2) + \sum_{i=1}^{n} c_i h_i + \frac{1}{\sigma_j} b_j \sum_{i=1}^{n} h_j W_{ij} \right)}$$

$$= \prod_{j=1}^{m} \frac{1}{\sigma_j \sqrt{2\pi}} \exp\left(-\frac{(v_j - b_j)^2}{2\sigma_j^2} - \frac{1}{2}\left(\sum_{i=1}^{n} h_i W_{ij}\right)^2 + \frac{1}{\sigma_j}(v_j - b_j)\left(\sum_{i=1}^{n} h_i W_{ij}\right)\right)$$

$$= \prod_{j=1}^{m} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma_j^2}(v_j - b_j)^2 + \sigma_j^2(\sum_{i=1}^{n} h_i W_{ij})^2 - 2\sigma_j(v_j - b_j)(\sum_{i=1}^{n} h_i W_{ij}))\right)$$

$$= \prod_{j=1}^{m} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(\frac{1}{2\sigma_j^2}(v_j - b_j - \sigma_j - \sum_{i=1}^{n} h_i W_{ij})^2 \right)$$

$$= \prod_{j=1}^{m} \mathcal{N}\left(b_j + \sigma_j + \sum_{i=1}^{n} h_i W_{ij}, \sigma_j^2 \right) \tag{2.11}$$

In the above, the step involving integration is done by completing the square for each u_i then using $\int_{-\infty}^{\infty} e^{-x^2} dx = \frac{1}{\sqrt{2\pi}}$.

2.5 The Spike and Slab Restricted Boltzmann Machine

Whilst the GRBM can be used to model real valued data, it is still not well suited to the task of learning from natural images [6, 21]. This is not entirely unexpected, as the representational power of the model is severely diminished by the move from a discrete domain to a continous one [14]. One possible reason for the failure of the GRBM is the binary nature of the hidden units, which are only guaranteed to accurately approximate binary distributions [14].

In an attempt to remove these shortcomings, the spike-and-slab restricted Boltzmann machine (ssRBM) was created [6]. In the ssRBM there are two distinct kinds of latent variables, the binary 'spike' variables and the real valued 'slab' variables. The name spike-and-slab is taken from the statistics literature, where it referred to a prior used in

Bayesian regression, in which the probability distribution is uniform across an interval, save for one elevated 'spike' value [22].

As before, there are D real valued visible variables represented by the vector $v \in \mathbb{R}^D$, N hidden units, with the ith hidden unit consisting of a spike variable $h_i \in \{0, 1\}$ and a real values slab variable $s_i \in \mathbb{R}$ or a vector of such variables $s_i \in \mathbb{R}^k$. The energy function the ssRBM defined over these variables is:

$$E(v, s, h) = -\sum_{i=1}^{N} v^T W_i s_i h_i + \frac{1}{2}\sum_{i=1}^{N} s_i^T \alpha_i s_i + \frac{1}{2} v^T \Lambda v - \sum_{i=1}^{N} b_i h_i$$

Where W_i a $D \times K$ weight matrix weighting the interactions between visible variables and the ith hidden variables, and b_i is the bias term for the ith spike variable. The diagonal matrices α_i and Λ are matrices responsible for decreasing the likelihood of configurations with large values of $||s_i||_2^2$ and $||v||_2^2$. The joint probability distribution is of the form:

$$p(v, s, h) = \frac{1}{Z}\exp(-E(v, s, h))\mathbb{I}(v)$$

where $\mathbb{I}(v)$ is a uniform distribution which over the ball of radius $R = \max ||v||_2$ centred at the origin, containing all the training data. The reason for the necessity of this addition to the energy function is that the increased complexity of the energy function means that it is no longer guaranteed that its integral over all possible configurations is bounded, meaning that the partition function would need to be infinite, this will be discussed in more detail in the next chapter. In practice the $\mathbb{I}(v)$ term is often disregarded and in the literature is often only mentioned in the first formal definition of the energy function [6, 23, 24]. This is a practice which will be adopted for the remainder of this chapter, with the reasons for doing so discussed in the section of the next chapter discussing the training of the ssRBM.

One of most important marginal distribution to calculate is $p(v|s,h)$. Within the bounts of $||v||_2 > R$; this distribution is calculated as:

$$\begin{aligned}
p(v|s,h) &= \frac{p(v,s,h)}{p(s,h)} \\
&= \frac{1}{p(s,h)}\frac{1}{Z}\exp\left(-E(v,s,h)\right) \\
&= \frac{1}{p(s,h)}\frac{1}{Z}\exp\left(-\frac{1}{2}v^T\Lambda v + \sum_{i=1}^{n}(v^T W_i s_i h_i + \frac{1}{2}s_i^T\alpha_i s_i + b_i h_i)\right) \\
&= \frac{1}{p(s,h)}\frac{1}{Z}\exp\left(\sum_{i=1}^{N}\frac{1}{2}s_i^T\alpha_i s_i + b_i h_i\right)\exp\left(-\frac{1}{2}v^T\Lambda v + \sum_{i=1}^{N}v^T W_i s_i h_i\right) \\
&= \frac{1}{Z'}\exp\left(\frac{1}{2}v^T\Lambda v + \sum_{i=1}^{N}v^T W_i s_i h_i\right) \\
&= \frac{1}{Z''}\exp\left((v-\Lambda^{-1}\sum_{i=1}^{N}W_i s_i h_i)^T\Lambda(v-\Lambda^{-1}\sum_{i=1}^{N}W_i s_i h_i)\right) \\
&= \mathcal{N}\left(\Lambda^{-1}\sum_{i=1}^{N}W_i s_i h_i, \Lambda^{-1}\right)
\end{aligned} \tag{2.12}$$

This derivation is an extended version of the three line derivation found in [23], with some extra steps added. Note that Z' and Z'' are used to denote the partition function absorbing terms which are independent of v, this will be common practice for subsequent derivations in this chapter. It is worth remembering that Λ is a diagonal matrix, and therefore its inverse is trivial to compute.

It may be tempting to try to marginalise out the slab variables and treat the model as having only binary latent variables, with the slabs serving as an elaborate set of parameters. However, attempting such a thing yields [23]

$$\begin{aligned}
p(v|h) &= \frac{1}{p(h)}\frac{1}{Z}\int \exp\left(\sum_{i=1}^{N}v^T W_i s_i h_i + \frac{1}{2}\sum_{i=1}^{N}s_i^T\alpha_i s_i - \frac{1}{2}v^T\Lambda v + \sum_{i=1}^{N}b_i h_i\right)ds \\
&= \frac{1}{p(h)}\frac{1}{Z}\left((2\pi)^{NK}\prod_{i=1}^{N}\det(\alpha_i^{-1})\right)^{\frac{1}{2}}\exp\left(\frac{1}{2}v^T\left(\Lambda - \sum_{i=1}^{N}h_i W_i \alpha_i^{-1}W_i^T\right)v\right) \\
&= \mathcal{N}\left(0, \left(\Lambda - \sum_{i=1}^{N}h_i W_i\alpha_i^{-1}W_i^T\right)^{-1}\right)
\end{aligned} \tag{2.13}$$

In this calculation, the integration over s is done by completing the square on s in the exponent in order to get the constant term $\left((2\pi)^{NK}\prod_{i=1}^{N}\det(\alpha_i^{-1})\right)$, with the remaining terms modelling a normal distribution on v.

This result makes treating the slab variables purely as parameters problematic, for a couple of reasons. Firstly, there is no guarantee that $\left(\Lambda - \sum_{i=1}^{N} h_i W_i \alpha_i^{-1} W_i^T\right)$ is positive definite, a necessary condition for it to be a valid covariance matrix. Secondly, even if it is positive definite, it is very computationally expensive to compute this matrix, since it is the inverse of a non-diagonal matrix [6].

It is however, possible to calculate $p(h_i = 1|v)$, marginalising out s [23]:

$$\begin{aligned}
p(h_i = 1|v) &= \frac{1}{p(v)}\frac{1}{Z}\int \exp(-E(v,s,h))ds \\
&= \frac{1}{p(v)}\frac{1}{Z}\int \left(\sum_{i=1}^{N} v^T W_i s_i h_i - \frac{1}{2}\sum_{i=1}^{N} s_i^T \alpha_i s_i - \frac{1}{2}v^T \Lambda v + \sum_{i=1}^{N} b_i h_i\right) ds \\
&= \frac{1}{p(v)}\frac{1}{Z}\exp\left(\frac{1}{2}h_i v W_i \alpha_i^{-1} W_i^T v + b_i h_i\right)\left((2\pi)^K \det(\alpha_i^{-1})\right)^{\frac{1}{2}} \\
&= \mathtt{sigm}(\frac{1}{2}v^T W_i \alpha_i^{-1} W_i^T v + b_i) \qquad (2.14)
\end{aligned}$$

The last step in the derivation of $p(h_i = 1|v)$ uses the trick shown in Equation 2.6, but with the β function being a constant zero since there are no terms in the exponent which are independent of h_i.

Finally, the distribution $p(s|v,h)$ is considered and shown to be Gaussian, this distribution is stated without proof in the literature [6, 23]:

$$
\begin{aligned}
p(s|v,h) &= \frac{1}{p(v,h)}\frac{1}{Z}\exp(-E(v,s,h)) \\
&= \frac{1}{p(v,h)}\frac{1}{Z} exp\left(\sum_{i=1}^{N} v^T W_i s_i h_i - \frac{1}{2}\sum_{i=1}^{N} s_i^T \alpha_i s_i - \frac{1}{2} v^T \Lambda v + \sum_{i=1}^{N} b_i h_i\right) \\
&= \frac{1}{p(v,h)}\frac{1}{Z'}\exp\left(-\frac{1}{2}\sum_{i=1}^{N}\left(s_i - h_i\alpha_i^{-1}W_i^T v\right)\right)\alpha_i\left(-\frac{1}{2}\sum_{i=1}^{N}\left(s_i - h_i\alpha_i^{-1}W_i^T v\right)\right) \\
&= \frac{1}{p(v,h)}\frac{1}{Z'}\prod_{i=1}^{N}\exp\left(-\frac{1}{2}\left(s_i - h_i\alpha_i^{-1}W_i^T v\right)\alpha_i\left(s_i - h_i\alpha_i^{-1}W_i^T v\right)\right) \\
&= \prod_{i=1}^{N}\mathcal{N}\left(h_i\alpha^{-1}W_i^T v, \alpha_i^{-1}\right) \qquad (2.15)
\end{aligned}
$$

The use of the pairing of two distinct kinds of latent variables allows for several choices of representation based on the output of the model. It is possible to use the products of the spike and slab variables as the representation, or similarly the product of the spike and the magnitude of the slab. Alternatively, it is possible to use just the spike variables, or the expectations thereof, using the fact that it is relatively easy to marginalise over s to find $p(h|v)$.

2.6 Higher Order Interactions

The ssRBM is much more suited to the task of learning representations of natural images than the GRBM, however, it still makes no attempts to model images as being generated from a distribution based on multiple explanatory factors. Using the approach outlined in [2], this can be done by associating each slab variable with multiple interacting spike variables.

The motivation behind this model is a notion regarding the relationship between the generative process and the process of creating a representation. In particular, that generating samples using a model is roughly the reverse of creating a representation for a specific input. Therefore, having a generative process in which information components become entangled leads to a model which exposes these components with data it is presented [2]. The energy function which gives rise to these interactions is:

$$
\begin{aligned}
E(v, s, f, g, h) = \frac{1}{2}v^T \Lambda v - \sum_k e_k f_k + \sum_{i,k} c_{ik} g_{ik} + \sum_{j,k} d_{jk} hjk + \frac{1}{2}\sum_{i,j,k} \alpha_{ijk} s_{ijk}^2 \\
+ \sum_{i,j,k} (-v^T W_{\cdot,ijk} s_{ijk} - \alpha_{ijk}\mu_{ijk} s_{ijk} + \frac{1}{2}\alpha_{ijk}\mu_{ijk}^2) g_{ik} h_{jk} f_k \qquad (2.16)
\end{aligned}
$$

The interactions between latent variables can be visualised in the way demonstrated in Figure 2.2. The parameters $g \in \{0,1\}^{M\times K}$ and $h \in \{0,1\}^{N\times K}$ are matrices, with the columns and rows respectively determining the interactions within a given block. The parameters in $f \in \{0,1\}^K$ are responsible for controlling the interactions between blocks. It is due to these higher order interactions between spike variables that the higher order spike-and-slab Restricted Botlzmann Machine (hossRBM) gets its name.

The other parameters are as follows:

- W: The $D \times M \times N \times K$ tensor weighting the interactions between the visible and latent variables.
- μ: The $M \times N \times K$ tensor describing the mean of the slab variables.
- α: The $M \times N \times K$ tensor describing the precision of the slab variables, punishing large values of s.
- Λ: The $D \times D$ matrix punishing large input values.
- c: The $M \times K$ matrix of bias terms for the g variables.
- d: The $N \times K$ matrix of bias terms for the h variables.
- e: The K vector of bias terms for the f variables.

The following conditional dstributions are all stated without proof in [2].

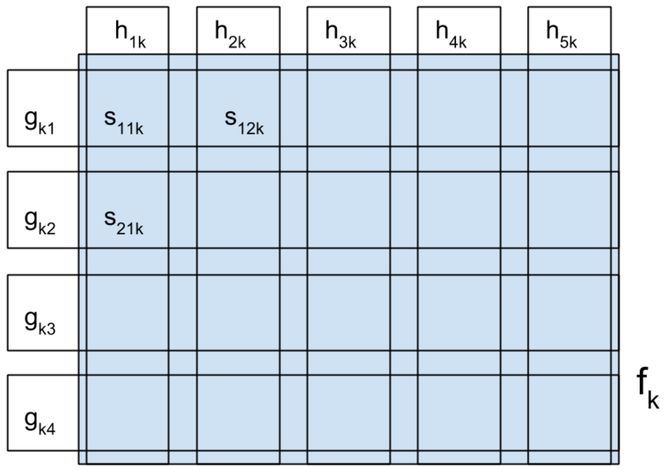


FIGURE 2.2: Visualisation of the interactions of spike and slab variables with the block controlled by f_k. Based on diagram from [2].

As with the ssRBM, the probabilities of the visible variables given the hidden variables can be shown to be Gaussian:

$$\begin{aligned}
p(v|s,f,g,h) &= \frac{1}{p(v,s,f,g,h)}\frac{1}{Z}\exp(-E(v,s,f,g,h)) \\
&= \frac{1}{p(v,s,f,g,h)}\frac{1}{Z'}\exp\left(\frac{1}{2}v^T\Lambda v - \sum_{i,j,k} v^T W_{\cdot ijk}s_{ijk}g_{ik}h_{jk}f_k\right) \\
&= \frac{1}{p(v,s,f,g,h)}\frac{1}{Z''}\exp\left(\frac{1}{2}\left(v - \Lambda^{-1}\sum_{i,j,k}W'_{\cdot ijk}\right)\Lambda\left(v - \Lambda^{-1}\sum_{i,j,k}W'_{\cdot ijk}\right)\right) \\
&= \mathcal{N}\left(\sum_{i,j,k}\Lambda^{-1}W_{\cdot ijk}s_{ijk}g_{ik}h_{jk}f_k, \Lambda^{-1}\right) \qquad (2.17)
\end{aligned}$$

Here $W'_{\cdot ijk} = W_{\cdot ijk}s_{ijk}g_{ik}h_{jk}f_k$ and all terms of the energy function that do not have v as factor are absorbed by the partition function $\frac{1}{Z'}$.

As with the ssRBM it is insightful to show the conditional distribution of the visible variables over the spike variables, marginalising out the slab variables. Due to the complexity of the energy function, proof of this is very fiddly, in order to partially alleviate this, some new notation is introduced

$$\hat{\alpha} = \frac{1}{2}v^T\Lambda v - \sum_k e_k f_k + \sum_{i,k} c_{ik}g_{ik} + \sum_{j,k} d_{jk}g_{jk} + \frac{1}{2}\alpha_{ijk}\mu_{ijk}^2 f_k g_{ik} h_{jk} \tag{2.18}$$

$$\hat{\beta}_{ijk} = \frac{1}{2}\alpha_{ijk} \tag{2.19}$$

$$\hat{\gamma}_{ijk} = -(v^T W_{ijk}\alpha_{ijk}\mu_{ijk}) f_k g_{ik} h_{jk}) \tag{2.20}$$

Using these definitions, the energy function becomes:

$$E(v,s,f,g,h) = \hat{\alpha} + \sum_{i,j,k} \hat{\beta_{ijk}} s_{ijk}^2 + \sum_{i,j,k} \hat{\gamma}_{ijk} s_{ijk} \tag{2.21}$$

Marginalising over the s variables gives:

$$\begin{aligned}\int p(v,s,f,g,h)ds &= \frac{1}{Z}\int \exp(-(\hat{\alpha} + \sum_{i,j,k}\hat{\beta_{ijk}} s_{ijk}^2 + \sum_{i,j,k}\hat{\gamma}_{ijk}s_{ijk}))ds \\ &= \frac{1}{Z}\exp(-\hat{\alpha})\int \exp(-(\sum_{i,j,k}\hat{\beta_{ijk}} s_{ijk}^2 + \sum_{i,j,k}\hat{\gamma}_{ijk}s_{ijk}))ds \\ &= \frac{1}{Z}\exp\,(-\hat{\alpha} + \sum_{i,j,k}\hat{\gamma}_{ijk}^2\hat{\beta}_{ijk}^{-1})\int \exp(-(\sum_{i,j,k}\hat{\beta_{ijk}}(s_{ijk} + \frac{1}{2}\sum_{i,j,k}\hat{\gamma}_{ijk}\hat{\beta}_{ijk}^{-1})^2)ds \\ &= \frac{1}{Z'}\exp\,(-\hat{\alpha} + \sum_{i,j,k}\hat{\gamma}_{ijk}^2\hat{\beta}_{ijk}^{-1}) \end{aligned} \tag{2.22}$$

From the above, the marginal distribution given s can be thought of as having an energy function of the form:

$$\begin{aligned}\hat{E}(v,f,g,h) =& \frac{1}{2}v^T\Lambda v - \sum_k e_k f_k + \sum_{i,k} c_{ik}g_{ik} + \sum_{j,k} d_{jk}hjk + \frac{1}{2}\alpha_{ijk}\mu_{ijk}^2 g_{ik}h_{jk}f_k \\ &+ \sum_{i,j,k}\left(v^T W_{\cdot ijk} - \alpha_{ijk}\mu_{ijk}\right) f_k g_{ik} h_{jk}\end{aligned} \tag{2.23}$$

This energy function can then be used to calculate $p(v|f,g,h)$:

$$\begin{aligned}
p(v|f,g,h) &= \frac{1}{Z}\int exp(-\hat{E}(v,f,g,h))dv \\
&= \frac{1}{Z'}\exp(-(\frac{1}{2}v^T(\Lambda - \sum_{i,j,k} W_{\cdot ijk}W^T_{\cdot ijk}\alpha^{-1}_{ijk}f_k g_{ik}h_{jk})v - v\sum_{i,j,k} W_{\cdot ijk}\mu_{ijk}f_k g_{ik}h_{jk})) \\
&= \frac{1}{Z'}\exp\left(-\left(\left[v - \sum_{i,j,k} C_{v|f,g,h}W'_{\cdot ijk}\right] C^{-1}_{v|f,g,h}\left[v - \sum_{i,j,k} C_{v|f,g,h}W'_{\cdot ijk}\right]\right)\right) \\
&= \mathcal{N}\left(\sum_{i,j,k} C_{v|f,g,h}W_{\cdot ijk}\mu_{ijk}f_k g_{ik}h_{jk}, C_{v|f,g,h}\right) \qquad (2.24)
\end{aligned}$$

In the above the notation $W'_{\cdot ijk} = W_{\cdot ijk}\mu_{ijk}f_k g_{ik}h_{jk}$ is used for readability and

$$C_{v|f,g,h} = (\Lambda - \sum_{i,j,k} W_{\cdot ijk}W^T_{\cdot ijk}\alpha^{-1}_{ijk}g_{ik}h_{jk}f_k^{-1})^{-1} \qquad (2.25)$$

This result differs slightly from that found in the original paper [2], in which the mean is given without proof as $\sum_{i,j,k}\Lambda^{-1}W_{\cdot ijk}\mu_{ijk}f_k g_{ik}h_{jk}$ [2].

From this distribution, it can be seen that f, g and h contribute not just to the conditional mean of the visible variables, but also the conditional covariance. This ability to model the conditional covariance in a more sophisticated way is argued to be a reason that this model should offer an improvement over the traditional RBM model in handling natural images [2].

The conditional distributions of the spike variables can be calculated as:

$$p(f_k = 1|v,g,h) = \sigma(e_{ik} + \sum_{i,j} v^T W_{\cdot ijk}\mu_{ijk}g_{ik}h_{jk} + \frac{1}{2}\sum_{i,j}\alpha^{-1}_{ijk}(v^T W_{\cdot ijk})^2 g_{ik}h_{jk}) \qquad (2.26)$$

$$p(g_{ik} = 1|v,f,h) = \sigma(c_{ik} + \sum_{j} v^T W_{\cdot ijk}\mu_{ijk}h_{jk}f_k + \frac{1}{2}\sum_{j}\alpha^{-1}_{ijk}(v^T W_{\cdot ijk})^2 f_k) \qquad (2.27)$$

[2]Contact has been made with one of the original authors regarding this, but so far he has neither refuted nor confirmed this as a mistake.

$$p(h_{jk}=1|v,g,h)=\sigma(d_{jk}+\sum_i v^T W_{\cdot ijk}\mu_{ijk}g_{ik}f_k+\frac{1}{2}\sum_{i,j}\alpha_{ijk}^{-1}(v^T W_{\cdot ijk})^2 g_{ik}f_k) \tag{2.28}$$

The derivations of the above equations are all very similar, and all heavily dependent on the trick used in equation 2.6. For $p(g_{ik}=1|v,f,h)$ the distribution can be demonstrated by defining:

$$\gamma_{i'k'}=c_{i'k'}+\sum_j v^T W_{\cdot i'jk'}\mu_{i'jk'}h_{jk'}f'_k+\frac{1}{2}\sum_j \alpha_{i'jk'}^{-1}(v^T W_{\cdot i'jk'})^2 f_{k'} \tag{2.29}$$

With β defined defined as the sum of all energy terms in $\hat{E}$ which do not have g_{ik} as a factor. The notation $\gamma_{i'k'}$ is being used rather than $\alpha_{i'k'}$ in order to avoid confusion with the precision tensor α. The results for $p(g_{ik}=1|v,f,h)$ and $p(f_k=1|v,g,h)$ can be found similarly.

Finally, the conditional distribution of the slab variables can be shown to be Gaussian:

$$\begin{aligned}
p(s_{ijk}|v,f,g,h) &= \frac{1}{p(s,h)}\frac{1}{Z}\exp(-E(v,s,f,g,h)) \\
&= \frac{1}{p(s,h)}\frac{1}{Z'}\exp\big(-\frac{1}{2}\alpha_{ijk}s_{ijk}^2 \\
&\quad + (v^T W_{\cdot ijk}s_{ijk}+\alpha_{ijk}\mu_{ijk}s_{ijk})f_k g_{ik}h_{jk}\big) \\
&= \frac{1}{p(s,h)}\frac{1}{Z''}\exp\left(-\frac{(s_{ijk}-\alpha_{ijk}W_{\cdot ijk}s_{ijk})^2}{2\alpha_{ijk}^2}\right) \\
&= \mathcal{N}\left((\alpha_{ijk}^{-1}v^T W_{\cdot ijk}+\mu_{ijk})f_k g_{ik}h_{jk},\alpha_{ijk}^{-1})\right)
\end{aligned} \tag{2.30}$$

2.7 Weights and Visualisation

When RBMs and their variants are used on images in the literature, a common method of demonstrating properties of a trained model is the visualisation of that model's weights. The weight matrix, or tensor, describes the conditional relationships between the visible variables and hidden ones. An advantage of this is that vectors from the matrix correspond to patterns of activation of visible units within the model. As such, the weights can be visualised, and used to gain an insight into the kinds of features the model has learnt. Furthermore, these features can be ranked by importance, with the features corresponding to latent variables with higher activations being those with the greatest influence on the generative process [16].

2.8 Chapter Summary

In this chapter the restricted Boltzmann machine has been introduced. Some general background has been given regarding where the models fit in the field of representation, as well as some detail given about MRFs in general, showing the mathematical underpinnings of the models. The structure of the restricted Boltzmann machine is described, and three variants of the RBM have been introduced. The motiviation behind each variant and the structure of each have been given, including several proofs relating to the distributions of variables which have previously been omitted from the literature.

Chapter 3

Training Restricted Boltzmann Machines

In the previous chapter, model parameters were introduced freely whenever they were required. However, no effort was made to describe what values these parameters should take. It is, in fact, the values of these parameters which determine the efficacy of the model in capturing the distribution of the data. This chapter discusses the training of restricted Boltzmann machines, that is, the process of finding values for these parameters which lead to meaningful representations of the data.

3.1 Maximum Likelihood

The primary goal of training a model is to find a distribution for the model which closely approximates the distribution from which the training data is sampled. A measure of this difference is the *Kullback-Leibler* divergence (KL-divergence). If the training data is from the distribution q and the model's distribution is p, then the KL-divergence is given by:

$$KL(q||p) = \sum_v q(v) \ln \frac{q(x)}{p(x)} = \sum_v q(v) \ln q(v) - \sum_v q(v) \ln p(v)), \tag{3.1}$$

where the sums are over all possible configurations of the visible variables, whilst marginalising out the hidden ones. Since the first term in the sum is dependent only upon the unknown distribution and not on the model, it can be disregarded [14]. This formulation will become useful shortly, but for now an alternative, equivalent formulation is considered: maximising log-likelihood.

Define θ as the set of all parameters for a given model and define $V = v_1, v_2, \dots v_l$ as the set of training data for the model. The likelihood of the model given the data is a function $\mathcal{L} : \theta \to \mathbb{R}$ defined as:

$$\mathcal{L}(\theta|V) = \prod_{i=1}^{l} p(v_i|\theta)$$

Generally, the log of this measure, the log-likelihood is used instead, this is given by

$$\ln \mathcal{L}(\theta|S) = \ln \prod_{i=1}^{l} p(v_i|\theta) = \sum_{i=1}^{l} \ln p(v_i|\theta)$$

The log-likelihood is a measure of the how likely the model is to generate the training data, it is therefore desireable that the parameters of the model be chosen in order to maximise this function. Analytic solutions are rarely possible for complex datasets, and therefore the usual approach to finding parameter values is the use of gradient ascent on the log-likelihood [14] or equivalently gradient descent on the negative log-likelihood [25].

3.2 Stochastic Gradient Ascent

Gradient ascent is an algorithm in which parameters are tuned by iteratively looking at the gradients of some function (in this case the log-likelihood function) at the current position in the parameter space, and changing the parameters by some amount, the learning rate, in the direction of the greatest increase in gradient. Since the training set can be very large, it would be costly to approximate the true log-likelihood gradient of the set at each iteration; a stochastic approach is usually used instead. Using stochastic gradient ascent (SGA), the gradient is estimated using a small batch of examples from the training distribution [26]. A version of the stochastic gradient ascent can be seen in Algorithm 1.

It is worth noting that there are several useful modifications which can be made to SGA. Within the scope of this project, it will be useful to note that the learning rate can be a function of the number of iterations which have currently been performed rather than a constant, which can allow the algorithm to avoid local minima in the initial stages, whilst allowing finetuning once the region containing the true minimum is found [18]. Additional features, such as weight decay, which punishes large parameter values, and momentum, which prevents some oscillations in parameter values by making the next iteration dependent on the previous as well as the current one, will not be used in this project, but are discussed in the context of RBMs in [14].

Algorithm 1: SOCHASTIC GRADIENT ASCENT

```
Input: Set of parameters $\theta$, Set of training data $V$, Number of Iterations $k$,
       Learning-rate $\eta$, Batch size $n$
Output: Updated parameters $\theta$
for $i$ in $1 \dots k$ do
    Choose $S$,a set of $n$ samples from $V$
    foreach sample $v$ from $S$ do
        $\theta \leftarrow \theta + \eta \frac{\partial}{\partial \theta} (\ln \mathcal{L}(\theta|x))$
    end
end
return $\theta$
```

In order to apply this algorithm to a model, it is necessary to be able to calculate $\frac{\partial}{\partial \theta} (\ln \mathcal{L}(\theta|v))$ for a given training example. Taking the restricted Boltzmann machine's energy function as an example, the log-likelihood of a single training sample v is given by [14]:

$$\begin{aligned} \ln \mathcal{L}(\theta|v) &= \ln(p(v|\theta) = ln \frac{1}{Z} \sum_h e^{-E(v,h)} \\ &= \ln \frac{\sum_h e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}} = \ln \sum_h e^{-E(v,h)} - \ln \sum_{v,h} e^{-E(v,h)} \end{aligned} \tag{3.2}$$

where $\sum_h$ is the sum over all possible configurations for the hidden variables and $\sum_{v,h}$ is the sum over all configurations of hidden and visible variables. From this, derivatives of the log-likelihood with respect to trained parameters can be found [14]:

$$\begin{aligned}\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta} &= \frac{\partial}{\partial \theta}\left(\ln \sum_h e^{-E(v,h)}\right) - \frac{\partial}{\partial \theta}\left(\ln \sum_{v,h} e^{-E(v,h)}\right) \\ &= -\frac{1}{\sum_h e^{-E(v,h)}} \sum_h e^{-E(v,h)} \frac{\partial E(v,h)}{\partial \theta} \\ &\quad + \frac{1}{\sum_{v,h} e^{-E(v,h)}} \sum_{v,h} e^{-E(v,h)} \frac{\partial E(v,h)}{\partial \theta} \\ &= -\sum_h p(h|v) \frac{\partial E(v,h)}{\partial \theta} + \sum_{v,h} p(v,h) \frac{\partial E(v,h)}{\partial \theta} \end{aligned} \tag{3.3}$$

An important observation gained from the above is that the derivative can be written as the difference between two expectations: the expectation of $\frac{\partial E(v,h)}{\partial \theta}$ under the distribution conditioned on v and the expectation of the same derivative under the model distribution. These two expectations are referred to as the positive phase and the negative phase respectively [25]. During training, the first term increases the likelihood of the model generating the training data, whilst the second term decreases the likelihood under the model of samples that it generates [25].

Calculation of the derivatives of the energy function with respect to model parameters, as required by Equation 3.3 turns out to be fairly easy. For the GRBM, the derivatives were found in the literature, specifically in [20]:

$$\frac{\partial E}{\partial W_i j} = \frac{v_j}{\sigma_j^2} h_i \tag{3.4}$$

$$\frac{\partial E}{\partial b_j} = \frac{v_j - b_j}{\sigma_j^2} \tag{3.5}$$

$$\frac{\partial E}{\partial c_i} = -h_i \tag{3.6}$$

$$\frac{\partial E}{\sigma_j} = -\left(\frac{v_j - b_j}{\sigma_j \sqrt{\sigma_j}}\right)^2 + \left(\frac{2v_j}{\sigma_j^3}\right)^T W_i j h i \tag{3.7}$$

For the ssRBM, explicit calculations of the gradients could not be found in the literature. The closest that could be found is [6], where the expectations for the positive phase terms are given. Using these values to verify calculations, the derivatives are found to be:

$$\frac{\partial E}{\partial W_{ij}} = v_j s_i h_i \tag{3.8}$$

$$\frac{\partial E}{\partial \Lambda_j} = \frac{1}{2} v_j^2 \tag{3.9}$$

$$\frac{\partial E}{\partial b_i} = h_i \tag{3.10}$$

$$\frac{\partial E}{\partial \alpha_i} = \frac{1}{2} s_i^2 \tag{3.11}$$

In this case, $\frac{\partial E}{\partial W_{ij}}$ and $\frac{\partial E}{\partial \alpha_i}$ are vectors of the length of the slab vectors s^k, or scalar values in the case $k = 1$.

Finally the gradients of the positive phase in the directions of the parameters of the hossRBM are given, these calculations do not appear in previous literature for the model.

$$\frac{\partial E}{\partial e_k} = f_k + \sum_{i,j} \left(-v^T W_{\cdot ijk} s_{ijk} - \alpha_{ijk} \mu_{ijk} s_{ijk} + \frac{1}{2} \alpha_{ijk} \mu_{ijk}^2 \right) g_{ik} h_{jk}$$

$$\frac{\partial E}{\partial c_i k} = g_i k + \sum_{j} \left(-v^T W_{\cdot ijk} s_{ijk} - \alpha_{ijk} \mu_{ijk} s_{ijk} + \frac{1}{2} \alpha_{ijk} \mu_{ijk}^2 \right) f_k h_{jk}$$

$$\frac{\partial E}{\partial d_j k} = d_j k + \sum_{i} \left(-v^T W_{\cdot ijk} s_{ijk} - \alpha_{ijk} \mu_{ijk} s_{ijk} + \frac{1}{2} \alpha_{ijk} \mu_{ijk}^2 \right) f_k g_{ik}$$

$$\frac{\partial E}{\partial \Lambda_i} = \frac{1}{2} v_i^2$$

$$\frac{\partial E}{\partial \alpha_{ijk}} = \frac{1}{2} s_{ijk}^2 + (-\mu_{ijk} s_{ijk} + \frac{1}{2} \mu_{ijk}^2) g_{ik} h_{jk} f_k$$

$$\frac{\partial E}{\partial \mu_i j k} = (-\alpha_{ijk} s_{ijk} + \alpha_{ijk} \mu_{ijk}) g_{ik} h_{jk} f_k$$

$$\frac{\partial E}{\partial W_{lijk}} = -v_l s_{ijk} g_{ik} h_{jk} f k$$

The difficulty which emerges when computing the gradient of the log-likelihood is not in any of the above calculations, the difficulty is that in the current form Equation 3.3 requires the computation of each of these terms for *every* possible configuration of the model.

3.3 Positive Phase of GRBM and ssRBM

At first the calculation of the expection for the positive phase of GRBM and ssRBM may seem problematic. For an arbitrary parameter θ the term for the positive phase is:

$$\sum_h p(h|v)\frac{\partial E(v,h)}{\partial \theta} \tag{3.12}$$

This appears to require the summing over all possible configurations of hidden variables, or worse for continuous variables, where it is possible for each variable to take infinitely many values. Fortunately, it is possible to greatly simplify this calculation.

Using the notation used in [14] and the derivations from [20], the positive phase for the derivative with respect to W_{ij} becomes:

$$\begin{aligned}
\sum_h p(h|v)\frac{\partial E}{\partial W_{ij}} &= \sum_h p(h|v)\frac{h_i v_j}{\sigma^2} \\
&= \sum_h \prod_{k=1}^{n} p(h_k|v)\frac{h_i v_j}{\sigma_j^2} \\
&= \sum_h \sum_{h_{-1}} p(h_i|v)p(h_{-i}|v)\frac{h_i v_j}{\sigma_j^2} \\
&= \sum_{h_i} p(h_i|v)\frac{h_i v_j}{\sigma_j^2} \\
&= p(h_i=1|v)\frac{v_j}{\sigma_j^2}
\end{aligned} \tag{3.13}$$

where $\sum_{h_{-i}}$ is the sum over all configurations of the set of hidden variables less the variable h_i. For b_j and c_i the derivatives are even simpler using the same trick to eliminate all terms which do not share the index to compute:

$$\begin{aligned}\sum_h p(h|v)\frac{\partial E}{\partial b_j} &= \sum_h p(h|v)\frac{v_j - b_j}{\sigma_j^2} \\ &= \frac{v_j - b_j}{\sigma_j^2}\end{aligned}$$

and

$$\begin{aligned}\sum_h p(h|v)\frac{\partial E}{\partial c_i} &= -\sum_h p(h|v)h_i \\ &= -\sum_{h_i} p(h_i|v)h_i \\ &= p(h_i = 1|v) \qquad (3.14)\end{aligned}$$

Finally, the derivative with respect to σ_j is also amenable to simplification.

$$\begin{aligned}\sum_h p(h|v)\frac{\partial E}{\partial \sigma_j} &= \sum_h p(h|v)\left(\frac{v_j - b_j}{\sigma_j\sqrt{\sigma_j}}\right) + \sum_h p(h|v)W_{ij}h_i \\ &= \left(\frac{v_j - b_j}{\sigma_j\sqrt{\sigma_j}}\right) + p(h_i = 1|v)W_{ij} \qquad (3.15)\end{aligned}$$

Applying the same technique to the positive phase of the gradient function of the ssRBM is also possible, but slightly more complicated. The results below are stated without proof in [6].

$$\begin{aligned}
\int_s \sum_h p(h,s|v)\frac{\partial E}{\partial b_i}ds &= \int_s \sum_h p(s|h,v)h_i ds \\
&= \sum_{h_{-i}} p(h_{-i}|v) \sum_{h_i} p(h_i|v)h_i \int_s p(s|h,v)h_i ds \\
&= \sum_{h_i} p(h_i|v)h_i \int_s p(s|h_i,v)ds \\
&= p(h_i = 1|v) \int_s p(s|h_i,v)ds \\
&= p(h_i = 1|v) \qquad (3.16)
\end{aligned}$$

For the derivative in the direction of Λ_j, the derivative is independent of both s and h, giving:

$$\begin{aligned}
\int_s \sum_h p(h,s|v)\frac{\partial E}{\partial \Lambda_j} &= \int_s \sum_h p(h,s|v)\frac{1}{2}v_j^2 \\
&= \sum_h p(h|v) \int_s p(s|h,v)\frac{1}{2}v_j^2 ds \\
&= \sum_h p(h|v)\frac{1}{2}v_j^2 \\
&= \frac{1}{2}v_j^2 \qquad (3.17)
\end{aligned}$$

Due to the dependence of $\frac{\partial E}{\partial W_{ij}}$ on both s and h, the calculation is slightly more difficult.

$$\begin{aligned}
\int_s \sum_h p(h,s|v)\frac{\partial E}{\partial W_{ij}}ds &= \int_s \sum_h p(h,s|v)v_j s_i h_i ds \\
&= v_j \sum_h p(h|v)h_i \int_s p(s|v,h)s_i ds \\
&= v_j \sum_{h_i} p(h_i|v)h_u \int_{s_i} p(s_i|v,h_i)s_i ds_i \\
&= v_j p(h_i = 1|v) \int_{s_i} p(s_i|v,h_i = 1)s_i ds_i \\
&= v_j p(h_i = 1|v)\alpha^{-1}W_i^T v \qquad (3.18)
\end{aligned}$$

The last line of the proof above comes from the fact that the integral will be evaluated as the mean of the distribution of s, which is known from 2.15 to be $h_i\alpha_i^{-1}W_i^T v$.

Finally, the explicit calculation for the positive phase of the gradient function with respect to α is given. This does not appear at all in the literature, but an attempt at derviing the gradient is given here [1].

$$\begin{aligned}
\int_s \sum_h p(h,s|v)\frac{\partial E}{\partial \alpha_i} &= \int_s \sum_h p(s,h|v)\frac{1}{2}s_i^2 ds \\
&= \sum_h p(h|v)\int_s p(s|h,v)\frac{1}{2}s_i^2 ds \\
&= \sum_h p(h|v)\int_s p(s|v,h)\frac{1}{2}s_i^2 ds \\
&= \sum_h p(h|v)\left(\int_{s_i} p(s_i|v,h)\frac{1}{2}s_i^2 ds\right) \\
&= \sum_h p(h|v)\left(\int_{s_i} s_i^2 \frac{\exp\left(\frac{(s_i-\mu)^2}{2\sigma^2}\right)}{\sigma\sqrt{2\pi}} ds_i\right) \\
&= \sum_h p(h|v)\left(\frac{1}{\sigma\sqrt{2\pi}}\int_u (\sqrt{2}\sigma u+\mu)^2 e^{-u^2} du\right) \\
&= \sum_h p(h|v)\bigg(\frac{1}{\sigma\sqrt{2\pi}}\int_u 2\sigma^2 u^2 e^{-u^2} du + \frac{1}{\sigma\sqrt{2\pi}}\int_u 2\sqrt{2}\sigma u\mu e^{-u^2} du \\
&\qquad + \frac{1}{\sigma\sqrt{2\pi}}\int_u \mu^2 e^{-u^2} du\bigg) \\
&= \sum_h p(h|v)\bigg(\frac{2\sigma^2}{\sigma\sqrt{2\pi}}\left[\frac{1}{4}\sqrt{\pi}\texttt{erf}(u) - \frac{1}{2}e^{-u^2}\right]_{-\infty}^{\infty} + \frac{2\sqrt{2}\sigma\mu}{\sigma\sqrt{2\pi}}\left[\frac{1}{2}e^{-u^2}\right]_{-\infty}^{\infty} \\
&\qquad + \frac{2\sigma^2}{\sigma\sqrt{2\pi}}\left[\frac{\sqrt{\pi}}{2}\texttt{erf}(u)\right]_{-\infty}^{\infty}\bigg) \\
em &= \sum_h p(h|v)\left(\frac{\sqrt{\pi}}{\sigma\sqrt{2}}\left[(\sigma^2+\mu^2)\texttt{erf}(u)\right]_{-\infty}^{\infty} + \left[\frac{1}{\sqrt{2}}(\sqrt{2}\mu+\sigma u)e^{-u^2}\right]_{-\infty}^{\infty}\right)
\end{aligned}$$

[1] In the original ssRBM paper [6] the parameter α is given a fixed value, the alpha parameter is allowed to vary in [24], but no calculations of the gradient are given

$$
\begin{aligned}
&= \sum_h p(h|v) \frac{\sqrt{\pi}}{\sqrt{2}\sigma}(\mu^2 + \sigma^2) \\
&= \sum_h p(h|v) \frac{\sqrt{\pi}}{\sqrt{2}\alpha_i^{-1}}((h_i \alpha_i^{-1} W_i^T v)^2 + \alpha_i^{-2}) \\
&= p(h_i|v)\sqrt{\frac{\pi}{2}}\alpha_i^{-1} W_i^T v v^T W_i + \sqrt{\frac{\pi}{2}}\alpha_i^{-1} \qquad (3.19)
\end{aligned}
$$

where μ and σ are the mean and standard deviation of the normal distribution defined for $p(s|v,h)$ given in 2.15 and $u = \frac{s_i - \mu}{\sigma}$. The evaluation of the above integral was done using a table of standard integration results [27], with $\texttt{erf}(x)$ being the error function, defined as

$$\texttt{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt.$$

3.4 Positive Phase for hossRBM

The calculation of the negative phase of the gradient has now been discussed for all the relevant models. However, a complication exists within the positive phase for the gradient of the log-likelihood for the hossRBM model. The form of the positive phase for a generic parameter θ is:

$$-\sum_{f,g,h} \int_s p(f,g,h,s|v) \frac{\partial E(v,s,f,g,h)}{\partial \theta} ds \qquad (3.20)$$

Due to the interactions of the slab variables in the energy function, attempts to compute $p(f,g,h,s|v)$ will require the computation $p(f|v)$, $p(g|v)$ or $p(h|v)$ [2]. Any attempts at computation of any of these terms will involve the summing over all possible configurations of the two other spike variables. As such, the computation is intractable, and a different approach must be taken.

[2]Assuming that s has already been marginalised out

The approach chosen in [2] is to assume that the joint conditional distribution $p(f,g,h|v)$ can in fact be approximate by a distribution in which f, g and h are conditionally independent given v. This method is known as mean-field approximation, and is another strategy taken from the physics, particularly, statistical mechanics [28], and has been shown to be very successful for use in deeper graphical models [29].

The idea is that the joint conditional distribution $p(f,g,h|v)$ can be approximated by some function $Q_v(f,g,h) = Q_v(f)Q_v(g)Q_v(h)$. The KL-divergence $KL(Q_v(f,g,h)||p(f,g,h|v)$ is used to give a measurement of the quantity to be minimised. This gives the desired distribution as:

$$\max \mathcal{L}(Q_v) = \max_{Q_v} \sum_{f,g,h} Q_v(f)Q_v(g)Q_v(h) \log\left(\frac{p(f,g,h|v)}{Q_v(f)Q_v(g)Q_v(h)}\right)$$

This can be found by using the conditional distributions $p(f|g,h,v)$, $p(g|f,h,v)$ and $p(h|f,g,v)$ to form a set of fixed point equations, which are iterated over until a stationary point is reached:

$$\hat{f}_k = \sigma(e_{ik} + \sum_{i,j} v^T W_{.ijk}\mu_{ijk}\hat{g}_{ik}\hat{h}_{jk} + \frac{1}{2}\sum_{i,j}\alpha^{-1}_{ijk}(v^T W_{.ijk})^2\hat{g}_{ik}\hat{h}_{jk}) \tag{3.21}$$

$$\hat{g}_{ik} = \sigma(c_{ik} + \sum_{i,j} v^T W_{.ijk}\mu_{ijk}\hat{f}_k\hat{h}_{jk} + \frac{1}{2}\sum_{i,j}\alpha^{-1}_{ijk}(v^T W_{.ijk})^2\hat{f}_k\hat{h}_{jk}) \tag{3.22}$$

$$\hat{h}_{jk} = \sigma(d_{ik} + \sum_{i,j} v^T W_{.ijk}\mu_{ijk}\hat{g}_{ik}\hat{f}_k + \frac{1}{2}\sum_{i,j}\alpha^{-1}_{ijk}(v^T W_{.ijk})^2\hat{f}_k\hat{h}_{jk}) \tag{3.23}$$

Assuming the absence of significant local maxima, the stationary points will correspond to the distribution with minimum distance from the desired distribution [3].

3.5 Gibbs Sampling

The ability to obtain unbiased samples from the model distribution is key when computing the negative phase of the log-likelihood gradient. This task is accomplished with the use of Gibbs Sampling, a technique which makes it possible to sample from the model's distribution without having to explicitly calculate the distribution's density [30].

This section closely follows the material found in [14], although attempts have been made to include additional commentary where it is constructive to do so. In particular, a brief discussion will be presented at the end of this section about the application of Gibbs Sampling to RBMs, in contrast with the far more general context of the original paper. Additionally, since this section relies heavily on the use of Markov chains, a primer on the subject is contained in Appendix B.

Let $X = \{X_1, \dots X_N\}$ be an MRF on a graph $G = (V, E)$. For convenience each random variable takes values from the same finite state space Λ. Let the joint probability distribution be defined as $\pi(x) = \frac{1}{Z}e^{-E(x)}$. A Markov chain $Y = \{X^{(k)} | k \in \mathbb{N}\}$ can be constructed, taking values in $\Omega = \Lambda^N$, with $X^{(k)} = (X_1^{(k)}, \dots, X_N^{(k)})$ being the state of the MRF at time k.

In order to define the transition probabilities for the chain, an arbitrary strictly positive distribution q on V is first defined, with $q(i)$ being the probability of selecting the variable X_i for $1 \leq i \leq N$. The new value for the chosen variable is sampled from the conditional distribution of the variable given the states of all the other variables in the current configuration. This gives the transition probability between two states x,y of the MRF Y as:

$$p_{xy} = \begin{cases} q(i)\pi(y_i|(x_v)_{v\in V\setminus i}) & \text{if } \exists i \in V \text{so that} v \neq i \Rightarrow x_v = y_v \\ 0 & \text{otherwise} \end{cases}$$

It can be shown that π is the stationary distribution of the MRF. This is done by showing that the detailed balance condition holds for the distribution. If $x = y$ then the detailed balance condition trivially holds. If x and y differ by more than one variable in X then $p_{xy} = p_{yx} = 0$ and once again the detailed balance condition trivially holds. Finally, the case that x and y differ by a single variable is considered. For some i let $x_j = y_j$ and $x_i \neq y_i$ and for all $j \neq i$. In this case it holds that

$$\begin{aligned}
\pi(x)p_{xy} &= \pi(x)q(i)\pi(y_i|(x_v)_{v\in V\backslash i}) \\
&= \pi(x_i,(x_v)_{v\in V\backslash i})q(i)\frac{\pi(y_i,(x_v)_{v\in V\backslash i})}{\pi((x_v)_{v\in V\backslash i})} \\
&= \pi(y_i,(x_v)_{v\in V\backslash i})q(i)\frac{\pi(x_i,(x_v)_{v\in V\backslash i})}{\pi((x_v)_{v\in V\backslash i})} \\
&= \pi(y)q(i)\pi(x_i|(x_v)_{v\in V\backslash i}) \\
&= \pi(y)p_{yx}. \qquad (3.24)
\end{aligned}$$

Hence the detailed balance condition holds. In order to show that π is the unique stationary distribution of the model, all that remains is to show that the Markov chain is both irreducible and aperiodic. Aperiodicity simply comes from the fact that $p_{xx} > 0$ for all possible states. Since π is the distribution of an MRF it is of the form $e^{-E(X)}$, and therefore is strictly positive. A necessary condition for this is that the distribution of each variable in the MRF is also strictly positive. The probability of a variable X_i reaching state x_i in a single transition is non-zero, and therefore any state can be reached with N steps, giving irreducibility.

A couple of important amendments to Gibbs sampling as defined above must be made before it is ready to be used in the training of RBMs. Firstly, in practice the variable to be updated is not selected at random, but instead the variables are updated in a prefined sequence. This is not a problem, as long as all variable are included in the sequence. Secondly, it is very common for several conditionally independent variables to be updated in parallel. In particular the hidden variables of the RBM are all updated in parallel, as are the visible ones. This modificiation is known as *block Gibbs sampling*

As discussed in the previous chapter, the energy function for the ssRBM is actually a truncated Gaussian, rather than a full Gaussian as is the case for visible variables in the GRBM [6]. In order to implement this rejection sampling is used. In rejection sampling, a sample is drawn from the full Gaussian described by the energy function, but is accepted as a valid sample if and only v is contained with the bounded domain, otherwise another sample is drawn from the distribution. This process is repeated until a sample from within the bounds is drawn.

The introduction of spike and slab variables complicates the process of Gibbs sampling due to the introduction of another set of variables. However, it is still amenable to Gibbs sampling if the distributions $p(h|v)$, $p(s|v,h)$ and $p(v|s,h)$ are used[6], in this case the structure of the k-step Gibbs chain is:

$$\begin{aligned}
&h_0 \sim p(h|v_0) \\
&s_0 \sim p(s|v_0, h_0) \\
&v_1 \sim p(v|s_0, h_0) \\
&\dots \\
&v_k \sim p(v|s_{k-1}, h_{k-1}) \\
&h_k \sim p(h|v_k) \\
&s_k \sim p(s|v_k, h_k)
\end{aligned}$$

where v_0 is a sample taken from training data. The negative phase of the hossRBM is also amenable to Gibbs sampling, with the Gibbs chain taking a similar form to that of the ssRBM, sampling from $p(f|v,g,h)$, $p(g|v,f,h)$, $p(h|v,f,g)$, $p(s|v,g,f,h)$ and finally $p(v|f,g,h,s)$. The initial value for the visible variables v_0 is taken from training data as before, and f_0,g_0 and h_0 taken at random [31].

3.6 Contrastive Divergence

In order to approximate sampling from the model distribution with great accuracy using Gibbs sampling requires a large number of steps for the distribution of the chain to converge to that of the model. However, in order to make it possible to train the model in a reasonable amount of time, Contrastive Divergence (CD) is often used.

In contrastive divergence the second term of the log-likelihood gradient is approximated by running a short Gibbs chain from a training sample. In particular, for k-step contrastive divergence (CD-k) a Gibbs chain is initialised from a training example $v^{(0)}$, then for each step t from 1 to k $h(t)$ is sampled from the conditional distribution $p(h|v^{(t)}$ then $v^{(t+1)}$ is sampled from $p(v|h^{(t+1)})$ [14]. Using Equation 3.3 and $p(v|h^{(t+1)})$ as an approximation of the probability of v under the model distribution, $CD_k(\theta, v^{(0)})$, and approximation of $\frac{\partial \ln \mathcal{L}(\theta|v)}{\partial \theta}$ is obtained as:

$$CD_k(\theta, v^{(0)}) = -\sum_h p(h|v^{(0)}) \frac{\partial E(v^{(0)}, h)}{\partial \theta} + \sum_h p(h|v^{(k)}) \frac{\partial E(v^{(k)}, h)}{\partial \theta} \tag{3.25}$$

The derivatives in the direction of a given parameter can be estimated by the derivative of that parameter with respect to a single example, or a small batch of examples.

The algorithm for k-step Contrastive Divergence (CD-k) applied to a GRBM is shown in Algorithm 2. This algorithm closely follows the algorithm presented in [14], but is applied to the GRBM instead of the RBM, in order to make it more relevant to the models in this project.

It is important to note that the samples gained from contrastive divergence are not truly samples from the stationary distribution of the model. This is due to the fact that using contrastive divergence does not maximise the likelihood of the data under the model, but instead maximises the difference between the two KL-divergences [14]:

$$KL(q|p) - KL(p_k|p)$$

This is not a problem when k is large, as in this case $KL(p_k|p) \to 0$ [14]. However, even for small values of k it has been found that contrastive divergence leads to reasonable approximations [14, 20].

Algorithm 2: k-step Contrastive Divergence.

```
Input: $v_1, v_2, \dots v_m, h_1, h_2 \dots h_n$, training data batch $B$, Number of steps $k$
Output: Estimates of the gradient in direction of model parameter
        $\Delta W_{ij}$,$\Delta b_j$,$\Delta c_i$,$\Delta\sigma_j$ $1 \leq i \leq n$, $1 \leq j \leq m$
init $\Delta W_{ij}$,$\Delta b_j$,$\Delta c_i$ for all $i$,$j$ for $v \in B$ do
    $v^{(0)} \leftarrow v$
    for $t = 0$ to $k - 1$ do
        for $i = 0$ to $n$ do
            sample $h_i^{(t)} \sim p(h_i|v^{(t)})$
        end
        for $j = 0$ to $m$ do
            sample $v_j^{(t+1)} \sim p(v_h|h^{(t)})$
        end
    end
    for $i = 1$ to $n$, $j = 1$ to $m$ do
        $\Delta W_{ij} \leftarrow \Delta w_{ij} + p(h_i = 1|v^{(0)})v_j^{(0)} - p(h_i = 1|v^{(k)})v_j^{(k)}$
        $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
        $\Delta c_i \leftarrow \Delta c_i + p(h_i = 1|v^{(0)}) - p(h_i = 1|v^{(k)})$
        $\Delta\sigma_j \leftarrow \Delta\sigma_j + \frac{v_j^{(0)}}{\sigma_j\sqrt{\sigma_j}} + p(h_i = 1|v_j^{(0)}) - + \frac{v_j^{(k)}}{\sigma_j\sqrt{\sigma_j}} - p(h_i = 1|v_j^{(k)})$
        /* Since the $\theta$ parameter updates are inter-dependent with the
           updates of other parameters, these updates must be done in
           parallel.  This is not the case for the RBM.                  */
    end
end
return $\Delta w_{ij}, \Delta c_i, \Delta b_j, \sigma_j$ for $1 \leq i \leq n$, $1 \leq j \leq m$
```

In many cases $k = 1$ is sufficient to train a model well [17]. In [14] it is demonstrated that the maximum of the log-likelihood of the training data over many iterations of gradient ascent is improved by increasing k, but the rate of improvement in the early stages of training is comparable. Care should be taken when drawing conclusions from this, however, monitoring the progress of the training of a model using log-likelihood can be problematic [18].

A potential modification to the way contrastive divergence is used is to store the final states of the chains used to compute the gradient, then use those states as the initial states for the contrastive divergence subroutine in the next iteration of the stochastic gradient ascent algorithm. This technique is known as persistent contrastive divergence [32], and is a technique which will be utilised by the implementations of models used for experiments later on. The use of this model can be problematic, as using the output which persists from the previous chain approximates sampling from the previous models distribution, not the updates one. For this reason, it is important to use a reduced learning rate when using PCD, as this increases the likelihood that high-probability states in one iteration will remain high-probability in the next [32]

3.7 Sparsity

In the literature, sparsity constraints are used on a variety of RBMs and RBM like models [18, 33]. These contraints are designed to ensure that the expected values of binary latent variables are very small, i.e., the average probability of a latent variable being activated by an input is close to zero. In order to accomplish this, a small probability known as a sparsity target is chosen, with the goal of having the probability of activation of each hidden unit being close to the sparsity target[18].

This penality is formulated as the negative cross entropy between the desired activation probability q and the actual activation probability of the current model p [18]:

$$\texttt{Sparsity Penalty} = A(q \log p + (1-q)\log(1-p)),$$

where A is the sparsity weight, a hyperparameter which decides the relative impacts of the log-likelihood and sparsity costs in the training of the model. Explicit computation of p would be very costly, so an alternative approach is used, p can be estimated by using the mean activation of hidden units based on the current batch used in the SGA algorithm [31]. In order to train a model with a sparsity target, a sparsity penality is introduced in a weighted sum with the log-likelihood to the function being maximised by the gradient ascent [33]. In this way, the function being maximised becomes:

$$\sum_{i=1}^{l} \ln p(v_i|\theta) + A\left(\frac{1}{N}\sum_{j=1}^{N}(q\sum_{i=1}^{l}\ln(p(h_i|v_l)) + (1-q)\sum_{i=1}^{l}(1-p(h_i|v_l))\right), \qquad (3.26)$$

For a model with l training examples, parameters θ and N hidden variables.

Setting sparsity targets can in some cases serve the opposite purpose to that discussed above. In cases where the problem is that most units are inactive the majority of the time, it can be advantageous to set a high sparsity target, perhaps around 0.5. Setting such a target encourages *more* hidden units to be active for a typical input [18].

3.8 Regularisation

Another modification that can be made to the cost function is the addition of a regularisation term. The purpose of regularisation is to punish large weights in the weight matrix, and therefore encourage a greater uniformity of weight values [35]. The use of regularisation can have a significant impact on the quality of the learnt representation of a model[36]. There are two type of regularisation that can be applied, L1 and L2. The penalties used in each case are defined as :

$$\begin{aligned} L1 &= \sum_i |\theta_i| \\ L2 &= \sum_i \theta_i^2 \end{aligned} \tag{3.27}$$

Where i indexes all weight parameters to which the regularisation applied. The key difference between the two kinds of regularisation is that L2 issues a far greater punishment to large values weight parameters. whilst punishing small values less. In practice, this difference causes L1 regularisation to lead to the model learning filters which are far more localised in the visible domain, especially when the number of latent variables is large; L2 regularisation leads to less localised, but smoother features [35]

Like the sparsity penalty of the previous section, the regularisation penalties from a weighted sum with the log-likelihood, of the form $A\texttt{L1} + B\texttt{L2} + \ln \mathcal{L}(\theta|S)$, which serves as a new cost function to be optimised with SGA, with A and B being referred to as the L1 and L2 regularisation penalties respectively.

3.9 Chapter Summary

This chapter has covered the details of training restricted Boltzmann machines. Firstly, the goal of training, the learning of the distribution of the training data, was described, motivating a discussion of log-likelihood. The method of stochastic gradient ascent was introuduced, with a demonstration of how this can be applied to the models discussed in the previous chapter. The problem of calculating the gradient terms for the various models was discussed, motivating the introduction of Gibbs sampling in order to obtain unbiased samples from the models distribution. Finally, the concepts of sparsity and regularisation were introduced, along with how these terms fit into the training process.

Chapter 4

Implementation

This chapter provides a brief discussion of the technologies used in the experiments carried out during this project. The main technologies used in the implementation are discussed, in increasing level of abstraction. The classifier used in classification problems within the proect is also discussed.

4.1 Model Implementation

The implementations of ssRBM and hossRBM models used in this project are the implementations released to accompany the original paper on the hossRBM [2, 31]. For the GRBM, the implementation which exists within Pylearn2 and the code accompanying the hossRBM were both considered and tested in the early stages of this project. It was decided to use the latter, as it was thought it would be easier to make like-for-like comparisons between models.

In both available implementations of the GRBM, there is no parameter for the mean of the visible variables. This is similar in some ways to [39], where the σ parameters are removed from the model, with the caveat added in the text that the energy function assumes a unit variance on all dimensions of the data. Similarly, if the data is pre-processed in order to get a zero mean, then the energy function will prove adequate.

One key difference between the hossRBM as described in Chapter 3 and the model implemented in the code is the removal of f variables gating the interactions between the blocks. In the case where only a single f block exists, the two models are equivalent, since the binary f variable would learn to be active with a probability of one for all inputs. For models with a larger number of blocks, this difference means that the implementation can be thought of as having all f variables permanently set to be on. It is argued that these variables are somewhat unnecessary [40], presumably because the representational power which being able to turn off and on whole blocks is minor compared to that given by the control of all variables within the blocks.

4.2 Pylearn2 and Theano

The implementations used in this project are highly dependent upon Pylearn2 [41] and Theano [42]. For this reason, the key features of both are briefly described.

Pylearn2 is a machine learning research library, designed to be used in a wide variety of experiments in the field. In the Pylearn2 framework, the model, data and training algorithm are all python objects. Experimental configurations are created in a YAML file, in which the objects are declared, along with any initialisation arguments they require. This leads to a system where learning algorithms can be thought of as consisting of a set of components, which can easily be interchanged or modified between experiments [41].

Theano is a python library used to compile and evaluate mathematical functions and is used for a lot of the computations in training the model. Of particular importance for the models in this project, is the ability of theano to perform symbolic diffferentiation and compile the resulting expression into an executable function which can be called by the code within a machine learning model. Using this to compute the gradients of log-likelihood function for energy-based models leads to simpler code; in [42] it is described as " combin[ing] the convenience of numpy sytax with the speed of optimized native machine language" .

4.3 SVM Implementation

For the linear classifier linearSVC is used from scikit-Learn[43]. linearSVC is an implementation of a linear Support Vector Machine(SVM), which can be applied to multi-label data. It uses a one-vs-rest classification, where for each possible label a two-class SVM is trained using with datat with that label being in the positive class, and the rest in the negative class. In order to classify an example, it is scored by each classifier, and is predicted to belong to the class whose classifier shows the most confidence that the example belongs in that class.

Some problems were encountered with non-deterministic behaviour within the classifier, with two classifiers trained on exactly the same data giving slightly different results. These differences were often with a percentage point, but could occaisonally be more extreme. In accordance with the documentation for the LinearSVC module [43], the tolerance, the criterion used to determine when to terminate the training of the model, was reduced by an order of magnitude from 10^{-4} to 10^{-5}.

Chapter 5

Experiments on Facial Images

This chapter is concerned with experiments carried out during the course of this project. The dataset used is introduced, and a general methodology for experiments is laid out. The findings made during the process of carrying out the experiments are described, and the data collected during the experiments is displayed.

5.1 Image Dataset

The dataset chosen for the experiments in this project is the Facial Recognition 2013 (FER-2013) dataset [1]. The dataset consists of 35,887 greyscale images of human faces. Each image is 48x48 pixels and is labelled with one of seven categories corresponding to the facial expression of its subject. These labels are: 0 = Angry, 1 = Disgust, 2 = Fear, 3 = Happy, 4 = Sad, 5 = Surprise, 6 = Neutral. Tests conducted by the compilers of the dataset have put human accuracy at identifying the facial expressions of the subjects at $65 \pm 5\%$

The dataset is of a similar format to the Toronto Face Dataset (TFD), a dataset frequently used for experiments in the literature [2, 9, 44]. However, the method of collecting the data is significantly different. Whereas the TFD is constructed by aggregating images from smaller datasets, most of which are created from images taken under ideal conditions, the FER-2013 dataset is constructed using Google's image search API, using facial images found using keywords relating to each emotion (e.g. "enraged" for anger), examples of images for facial expression are shown in Figure 5.1. The TFD dataset is therefore more highly structured, containing multiple images of the same person under different conditions; this structure is potentially useful in both the unsupervised and supervised stages of the classification task, and therefore higher accuracy might be expected from the TFD than the FER-2013.

Limited preprocessing was done on to the dataset. Scaling was applied to each pixel so as to give the training data a standard deviation of one. The choice to scale each pixel has a couple of advantages. Firstly, it means that the standard deviation of the data is uniform, meaning that the speed that the weights in the model corresponding to each visible variable train at the same speed. Secondly, it gives an easy way to select what the initial values for the parameters measuring precision of the visible variables should be, and makes it easy to see that something is going wrong in the model if these parameters are learnt to be significantly different from these values. The pixel values for each image are also shifted, so that the mean value for each pixel in over the training set is zero. This is necessary, since the GRBM implementations used in the experiments are not capable of learning the mean of individual pixels [31].

For the experiments, the dataset was split into three smaller sets: the training set, consisting of 28,709 images; the validation set, consisting of 3,589 images and the test set, also consisting of 3,589 images. The size chosen for these sets is the same as the size originially chosen for the competition for which the dataset was originally published. A validation set is used instead of using cross validation, since the amount of time required to train models is prohibitive, and the size of the training set is sufficiently large that the loss of some samples for use in a validation set should not be problematic.

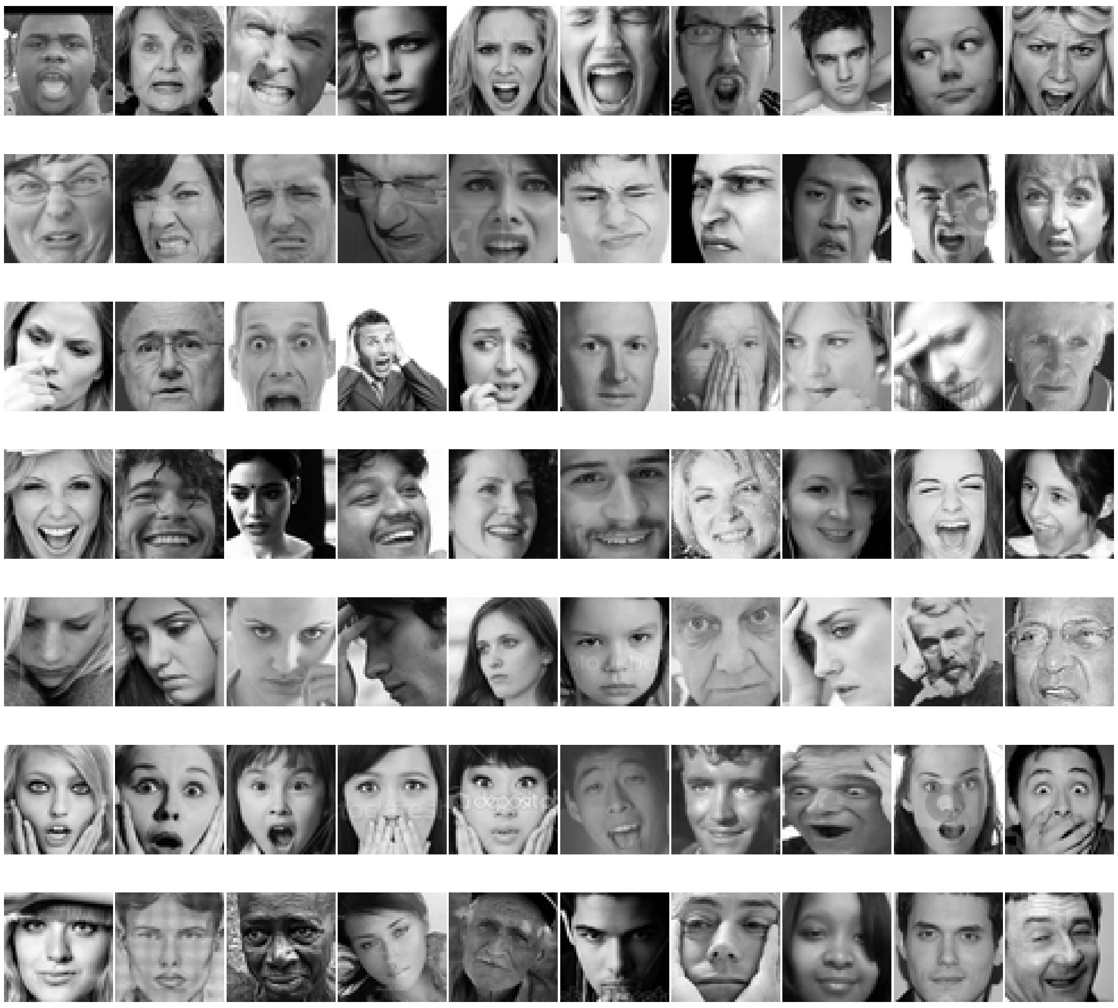


FIGURE 5.1: Examples of images taken from the facial image data set [1]. From top to bottom: Anger, Disgust, Fear, Happy, Sad, Surprise, Neurtal.

5.2 Methodology

A common problem, especially in the cases of ssRBM and hossRBM models is that the majority of the parameter space leads to models in which weight values tend rapidly towards infinity. In order to counter this for simpler RBM models, it can be sufficient to simply decrease the learning rate [18]. However, significantly more exploration of the parameter space was required for the more complex models. The restrictions, compromises and additional constraints for each model will be discussed in individual sections. The general strategy adopted for the experiments was to find suitable values for hyper-parameters using visual inspection of the filters learnt by each model. Where a relatively broad range is found, the most appropriate value is decided by training multiple models using different values and testing their classification accuracy on the validation set.

Classification is done using a multi-class support vector machine (SVM). As well as the tolerance parameter discussed in the previous chapter, the classifier also has a penalty parameter, C, which decides the penalty in the SVM's cost function for incorrect classifications. Whilst it would be possible to use the validation set again here to find the optimum value of this parameter, it was decided that this is unnecessary, or even counter-productive. This is because the aim of experiments in this project is to measure the effectiveness of unsupervised models in improving classification accuracy with a linear classifier, not just improving classification accuracy as much as possible. Increasing the complexity of the classification step itself does little to help achieve this goal.

Confidence intervals for classifier accuracy will be calculated under the assumption that each test sample is independent, and in line with the methods described in Appendix A.

The number of iterations for which the the gradient ascent algorithm is run is a very important decision in training a model. It was decided that each model would be trained for 50,000 iterations of the gradient descent algorithm, divided into epochs of 1,000 steps, when monitoring statistics will be collected. This number of iterations is within the range of iterations used in the literature [6, 20] and should allow the models enough time to learn the structure of the dataset whilst low enough that the amount of time to train a model is not too prohibitive [1]

Another decision which was made was to have the learning rate decrease linearly with each epoch, ending an order of magnitude lower than its original value. The reasoning behind this, inspired by [18], is that the large learning rate in the earlier stages allows the model to avoid local minima in the vicinity of the initial values of parameters, whilst decreasing later on allows the model to fit more snugly in the minimum in which it finds itself.

[1] Depending on the model and number of hidden variables, experiments performed in this project took anywhere between two and twenty hours to run

5.3 Raw Pixel Representation

One of the primary purposes of many unsupervised learning techniques, including the ones examined in this project, is to serve as a form of preprocessing for data in supervised learning tasks [3]. As such, for a representation to be truly useful it must serve as a more effective input to a classification model than the raw pixel representation. In order to find this benchmark, the classifier was trained and tested using this representation.

Training a multiclass support vector machine on the raw pixel representation of the training data provides a model with an accuracy of 33.0% $\pm$ 0.7 on the training data and 27.0% $\pm$1.4 on the test data, the discrepancy being due to the model overfitting the data. It should be noted that both these figures are significantly better than chance (14.3%), but well below the human benchmark of 65.5% $\pm$ 5.

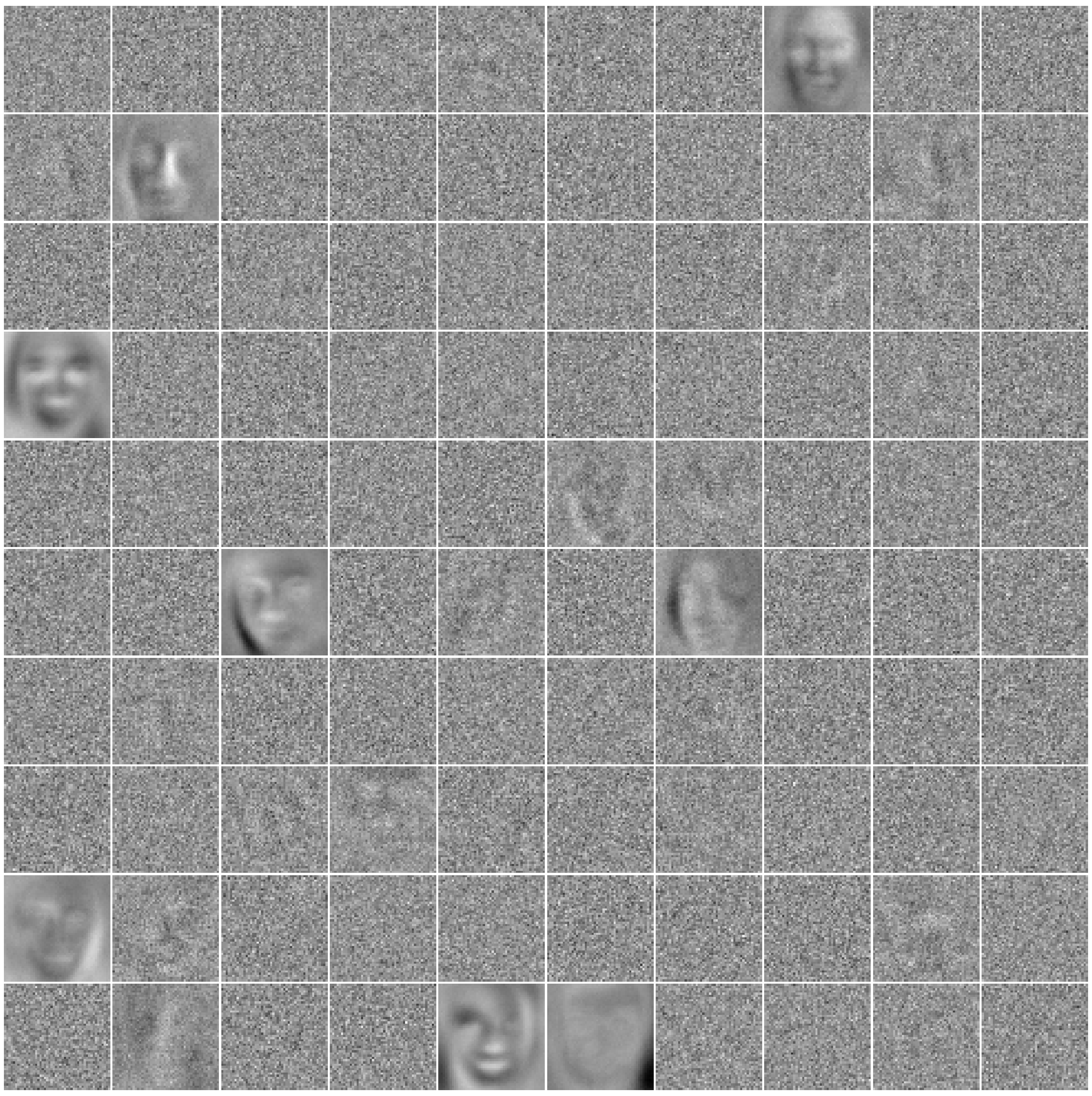
5.4 Experiments using GRBMs

Preliminary Experiments

An interesting phenomenon observed when training GRBMs, is the tendancy for a large number of weights to get stuck in a relative noisy state, as shown in figure 5.2. Using a model which has learnt these filters for the purposes of classification leads to relatively poor results, only just better than chance. Several possible causes to this were explored. Ideally, classification accuracy on the validation set would be used for each of the models trained in this preliminary investigation, however, this option proved to be too time-consuming to be practical. In many cases the quality of the filters learnt by the model are clear from visual inspection alone, with noisy unstructured filters clearly indicating a poor representation; for such cases no classifiers were trained.

The first thing which was attempted was to increase the initial learning rate. The idea being that the reluctance of the model to leave these noisy filters was due to them being contained within local minima of the cost function. If this was the case, then increasing

FIGURE 5.2: Examples of weights learnt by an GRBM with 500 hidden units



the learning rate at the beginning of training would make it more likely that the model would be able to leave these minima in order to find a better configuration closer to the global minimum. However, these attempts proved to be in vain, as such models would produce models which were either qualitively very similar, or within which the weights would explode off to infinity, rendering the model useless. Interestingly, the difference between a learning rate leading to a good representation and one in which the weights exploded appears to be very small. In the experiments carried out for this project, a learning rate of 0.001 would lead to a reasonably successful model, whereas a learning rate of 0.002 would return an error after less than one epoch.

In accordance with the advice given in [18], the weight matrix is initialised from a zero-mean normal distribution with a small standard deviation. Initially, a standard deviation of 0.1 was used. However, the results achieved with a standard distribution of 0.01 appeared to be far less noisy, giving filters that appear cleaner and more structured.

The one hyper-parameter which was not changed is the initialisation of the variances. This is because there is a good justification for setting this to a specific value. Since the precision parameters describe the approximate precision of the inputs in a given dimension, and the dataset has unit precision in each direction, it is logical to set these parameters to initially be one.

Introducing a regularisation term proved to yield interesting results. Using L1 regularisation on the weight matrix produced the filters shown in Figure 5.3. In some ways this matrix looks like it provides a poorer set of filters to learn representations of the data, with less clearly defined facial structures. However the filters are also a lot smoother, and appear to deal with more localised structures within the image space. This is in line with the results of L1 regularisation found in [35]. L2 regularisation was also tried, demonstrating similar quality filters for small values of the regularisation penalty, however, neither kind of regularisation encouraged more filters to contain useful structure, and so did not provide a solution to the problem.

A wide variety of values for the initial bias terms for the hidden variables were tried. Positive or zero values initial values would cause the model to learn minimal structure. Large negative values also generate filters with no discernable structure. Some success was achieved with fairly small negative values. The filters learnt with initial h of -5 and -10 both looked promising, the first being noisier, but with a greater number of filters showing meaningful structure than the prior (An attempt to show the difference can be seen in Figure 5.4). A large portion of the filters still show little structure, but this number is reduced. In order to determine which configuration to use for the main experiment, both options were used to train a model and a classifier trained on the validation set for each. The results are shown in Table 5.1; based on these results, the initial biases for the hidden units in further experiments was chosen to be -10

Initial h values	Training Set	Validation Set
-5	31.9% $\pm$ 0.6	28.1% $\pm$ 1.5
-7.5	29.1% $\pm$ 0.5	26.6% $\pm$1.4
-10	35.6% $\pm$ 0.6	33.4% $\pm$ 1.5

TABLE 5.1: Table showing the classification accuracy of GRBMs with different initial bias values for hidden units

	L1 Regularisation		L2 Regularisation	
Regulaisation Penalty	Training Set	Validation Set	Training Set	Validation Set
0	31.9% $\pm$ 0.6	28.1% $\pm$ 1.5	31.9% $\pm$ 0.6	28.1% $\pm$ 1.5
0.001	21.8 % $\pm$ 0.5	22.4 % $\pm$ 1.5	23.9% $\pm$ 0.5	24.2% $\pm$ 1.5
0.01	25.4 % $\pm$ 0.5	25.2 % $\pm$ 1.5	17.3% $\pm$ 0.5	16,8% $\pm$1.4

TABLE 5.2: Table showing the classification accuracy of GRBMs with different amounts of L1 and L2 regularisation applied

Number of hidden units	Training Set	Test Set
500	30.5% $\pm$ 0.5	27.4% $\pm$ 1.5
1000	39.6% $\pm$ 0.7	34.5 % $\pm$1.6
1500	42.1% $\pm$ 0.7	32.5% $\pm$1.5
2000	30.2% $\pm$ 0.5	27.8% $\pm$1.5

TABLE 5.3: Table showing the classification accuracy of GRBMs with different numbers of hidden units

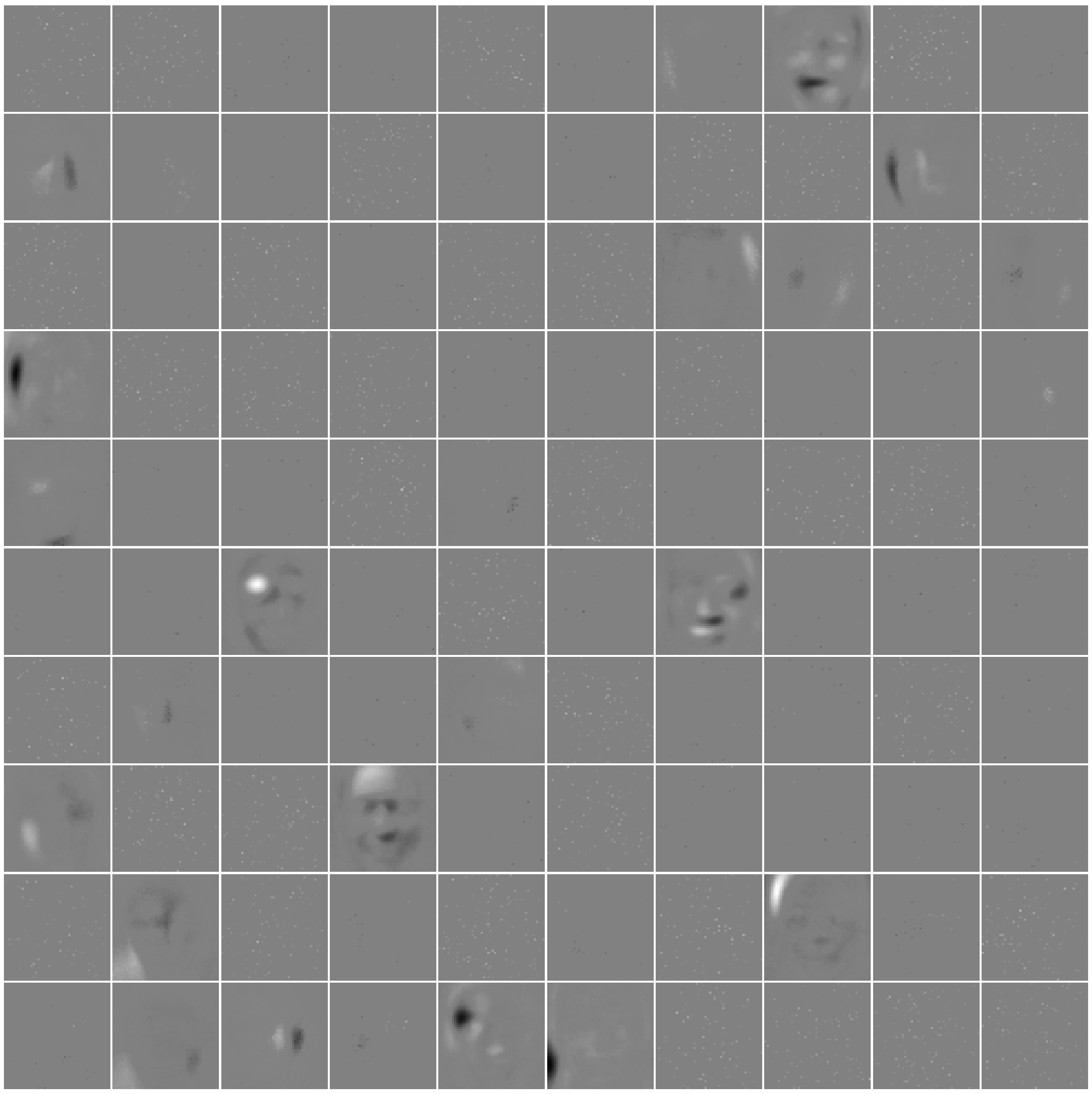
Having established a suitable set of hyper-parameters, L1 and L2 regularisation was attempted again, to see if their ability to produce smoother filters (as has been demonstrated in [35]) would have a beneficial effect on classification accuracy on the vaidation set. The results of these experiments can be seen in Table 5.2, based on these results, it was decided not to use regularisation for the main experiment.

GRBM Main Experiment

Having established a suitable set of hyper-parameters, different numbers of hidden units were experimented with, this time using the test dataset. The results can be seen in Table 5.3.

Since in all cases tested for the GRBM the dimensionality of the learnt representation has been less than the dimensionality of the raw pixel representation, GRBM can be considered to have been used as a method of dimensionality reduction. As such, it is interesting to compare the classification accuracies found against classification accuracies

FIGURE 5.3: Examples of weights learnt by an GRBM with 500 hidden units and an L2 regularisation penalty



which are achieved using PCA on the raw pixel representation. This data can be found in Table 5.4. There is no clear winner in the comparison, with both achieving significance over the other in some cases. However, it is possible that with better hyperparameters (ones for which training results in all filters showing meaningful structure), that the classification accuracy of GRBM could be greatly improved.

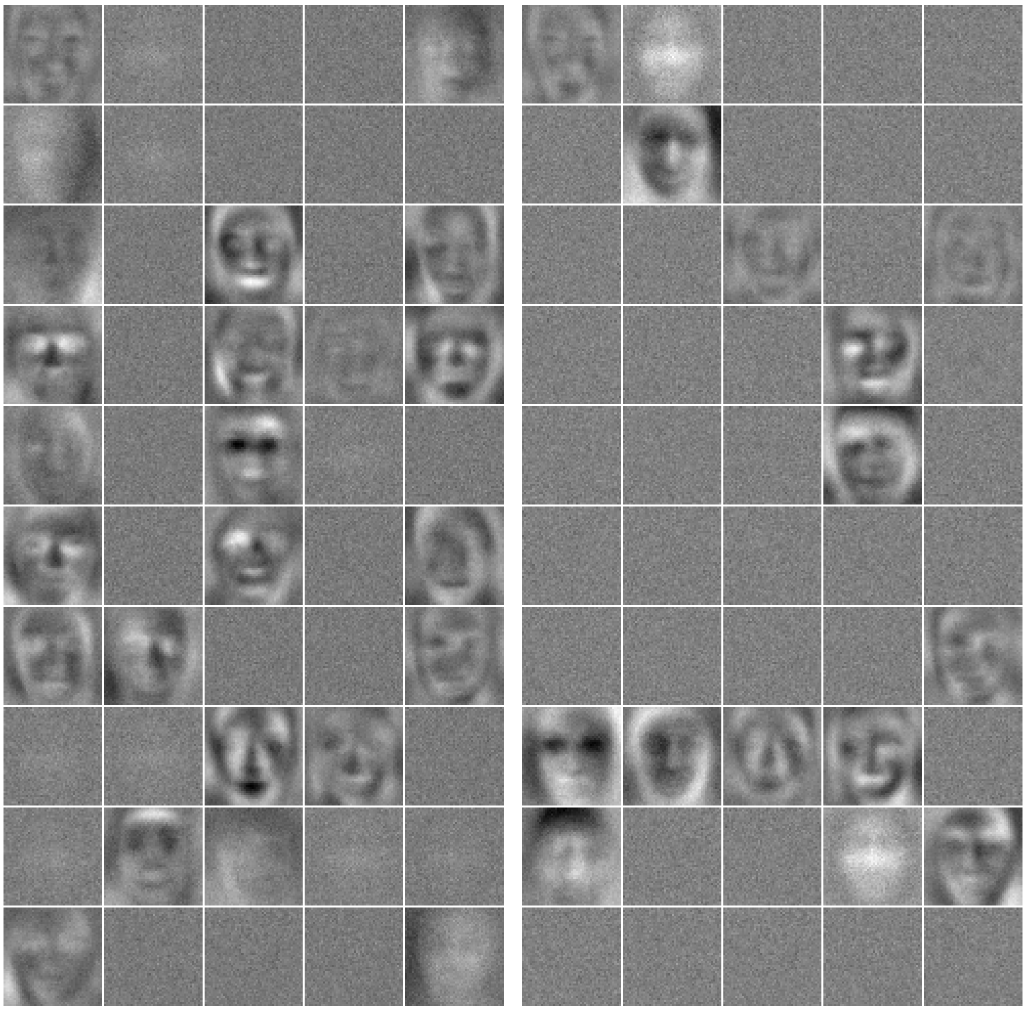


FIGURE 5.4: Examples of filters learnt by two GRBMs with different initialisation of biases for hidden units. Left: -5, Right: -10

Number of Dimensions	Training Set	Test Set
500	34.9% ± 0.7	32.1% ±1.5
1000	34.1% ± 0.7	30.4 % ± 1.5
1500	35.2 % ± 0.7	30.1% ± 1.5
2000	36.9 % ± 0.7	27.8% ± 1.5

TABLE 5.4: Table showing the classification accuracy of SVM on PCA data (retaining dimensions containing the most variance)

5.5 Experiments Using ssRBMs

Preliminary Experiments

Finding suitable hyper-parameters for the ssRBM proved to be an extremely difficult task. Many different sets of parameters were attempted before any model was found which successfully learnt any meaningful filters at all. The primary problems were that moderate to large values of the learning rate would cause the model to find extreme difference in gradient, thus causing the weight values to explode, i.e., grow exponentially quickly and rapidly produce 'nan' errors; conversely, small learning rates would fail to learn any interesting features at all, similar to the problems encountered with using the GRBM in [45].

In order to prevent a weight explosion, many configurations of initial conditions were tried with a very small learning rate (i.e. 10^{-6} to 10^{-7}). This prevented the weights from exploding in all cases, however, it became clear that such a low learning rate was preventing meaningful filters from being learnt.

Eventually, with some expert advice [40], a successful configuration was found. The most important hyper-parameters appear to be the initial size of elements the precision matrix α. The key with these value seems to be that they need to be very large, severely punishing excessively large values of the slab variables. Another constraint which proved important to add to the model was the truncation of the of the distribution of v variables. The apparent necessity of this constraint is contrary to the claims in the literature, discussed in Chapter 3, about the model automatically tending towards regions of stability, and therefore the model not requiring artificial bounds on the distribution of the visible variables [24]. With these parameters discovered, a learning rate of 10^{-4} to 10^{-3} proved sufficient to train the model.

ssRBM Main Experiment

A selection of weights learnt by the ssRBM can be seen in Figure 5.5. Many of these clearly exhibit strong resemblence to human faces. If the number of weights could successfully be increased in the model, perhaps to several times the number of input features, it would be expected that the weights learnt would be more localised, with many filters being concerned with a single facial feature [16]

For each model a linear SVM is trained on two representations, one treating each of the spike variables and each slab variables as a separate dimension (labelled as 'separated'),

# of units (# slabs per unit)	Separated		Product	
	Training Set	Test Set	Training Set	Test Set
500 (1)	36.9% ± 0.6	34.7% ± 1.6	27.41% ± 0.5	26.41 % ± 1.4
1000 (1)	33.7% ± 0.5	32.2% ± 1.5	36.7% ± 0.6	34.1% ± 1.6
500 (2)	30.5 % ± 0.5	30.0 % ± 1.5	35.5% ± 0.6	32.6% ± 1.5
1000 (2)	33.7 ± 0.5%	32.3% ± 1.5	33.9 % ± 0.6	31.1 % ± 1.5

Table 5.5: Classification accuracies of ssRBM models with different numbers of hidden variables

the other being the expectation of product of the spike variables with their corresponding slab variables (labelled as 'product'). The results of these experiments are shown in Table 5.5. In experiments, neither of the two representations consistently outperforms the other in terms of classification accuracy. This suggests that for real-world applications, the choice of representation would be best decided through the use of a validation set.

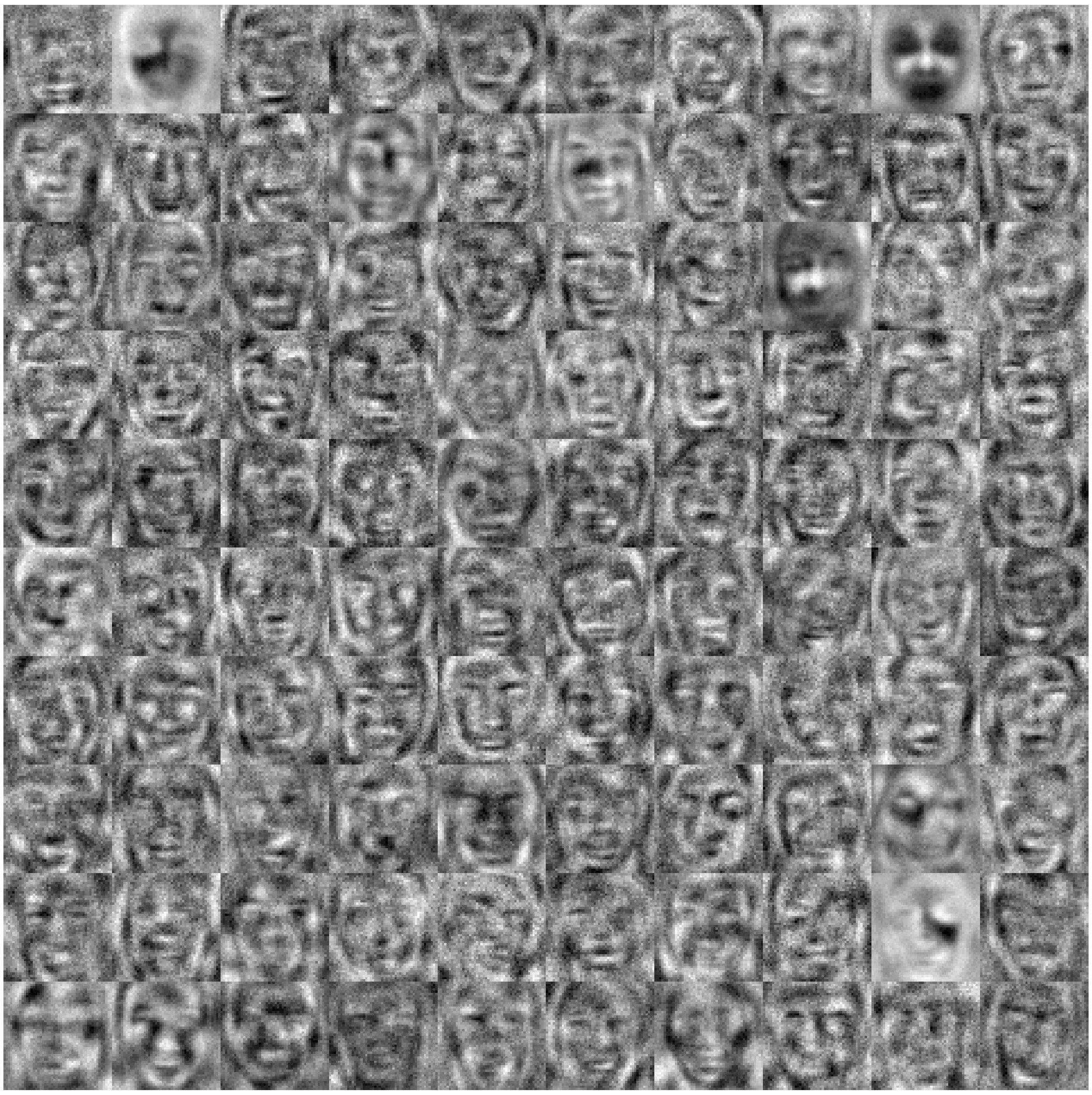
5.6 hossRBM

There were several difficulties in training the hossRBM. It appears that the log-likelihood of the cost function is very unstable. Many attempts to train the model had teh gradient descent contained to a small volume of the parameter space for prolonged periods of time before suddenly become uncontrollably large. An example of this can be seen Figure 5.6. As a result of this, no successfully trained models were created. This section discusses the various experiments performed whilst searching for an initial configuration of the model leading to meaningful representations of the data.

In the experiments described below, a single block structure was used, with 6×4 slab variables. Whilst multi-block structures and structures with more latent variables were briefly experimented with early on, it was decided that the best approach would be to get the model working for a small single block structure before moving on to more complex configurations.

In a departure from the training strategy for the previous two models, only twenty epochs of training were used (in cases where the weight explosion does not occur before twenty epochs). This should be more than sufficient to for structure to appear in the

FIGURE 5.5: Examples of weights learnt by an ssRBM with 500 hidden units

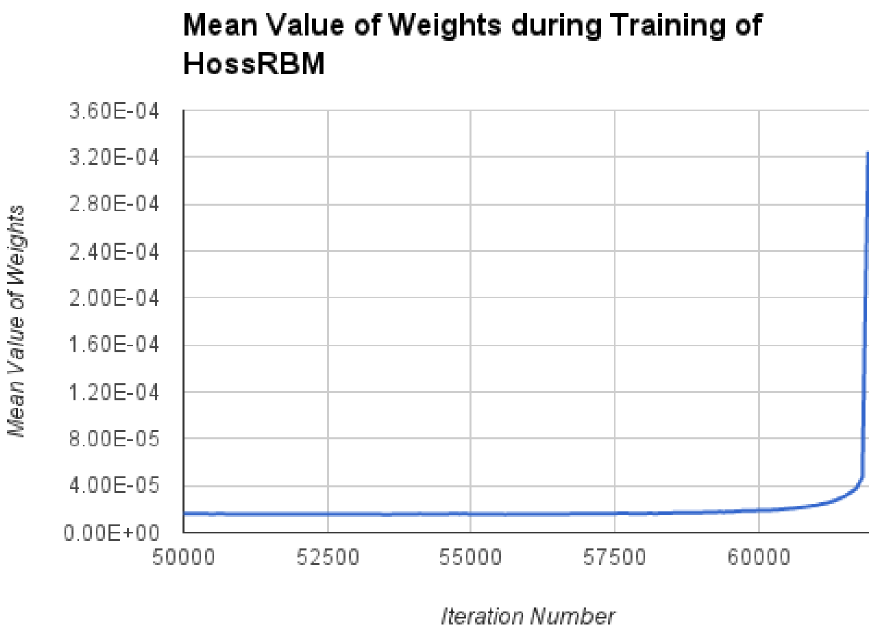


model (in previous models visible structure would often appear within a single epoch), this was necessary, as even with the reduction in the number of iterations the models would take several hours to train.

Changing Learning Rate

Attempts were made to mitigate this by changing the learning rate. Increasing the learning rate above the range of 0.01 would cause the weight explosion to happen much sooner, often with 10,000 iterations. Decreasing the learning rate causes the model

FIGURE 5.6: Graph showing an example of a model become unstable after a prolonged period of training.



to 0.001 or below, weight explosion does not occur but the model fails to learn any meaningful filters.

Changing Initial Weights and Bias Values

It was observed that during training, it was often the case that only one or two spike variables would be active in the samples in the monitoring batches. Encouraging activations of these spike variables by initialising their biases to be strongly negative was attempted, but yielded no success.

Since the problem is that the weights growing very large, an obvious thing to attempt is to decrease the size of the initial weights. Initial weights are sampled from a zero-mean normal distribution, reducing the standard deviation of this distribution from 0.1 to 0.05 makes the weight explosion happen later, but does not prevent it. In light of this observation, it seems plausible that the sudden explosion of weights occurs not after a period of stability, but is in fact preceded by a period of slow growth. This line of enquiry seems initially promising, plotting the change in the mean of weights when the weights are initialised from a zero-mean distribution of with standard deviation 0.01 appears to increase linear, as shown in Figure 5.7. However, zooming in on the first half of the graph in Figure 5.6, gives the graph shown in 5.8, showing a far more meandering and stable path. Whilst the possibility still exists that this a very noisy gradual increase, it is a lot less clear cut.

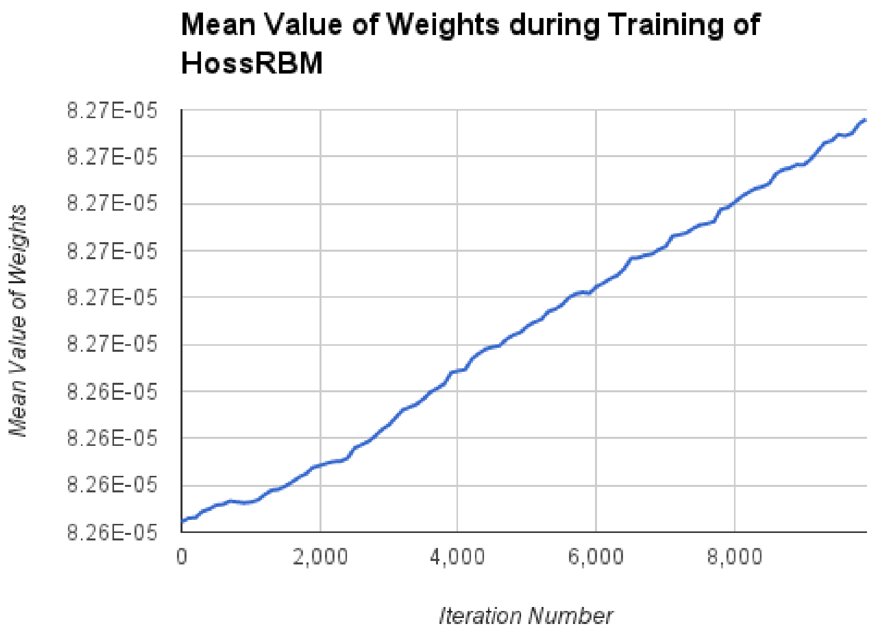


FIGURE 5.7: Graph showing the mean value of weight tensor during first epoch of training of hossRBM.

Adding Sparsity Conditions

Enforcing sparsity constraints appears to have little effect on the training of the model. Since there appears to be a problem with under-activation of units, the sparsity target was chosen to be 0.5 for both g and h variables, with the goal of driving up the activation probability of units. This however, appeared to have no discernable effect, suggesting that the low number of activations of hidden units is a symptom, rather than a cause of the problems found in training the model.

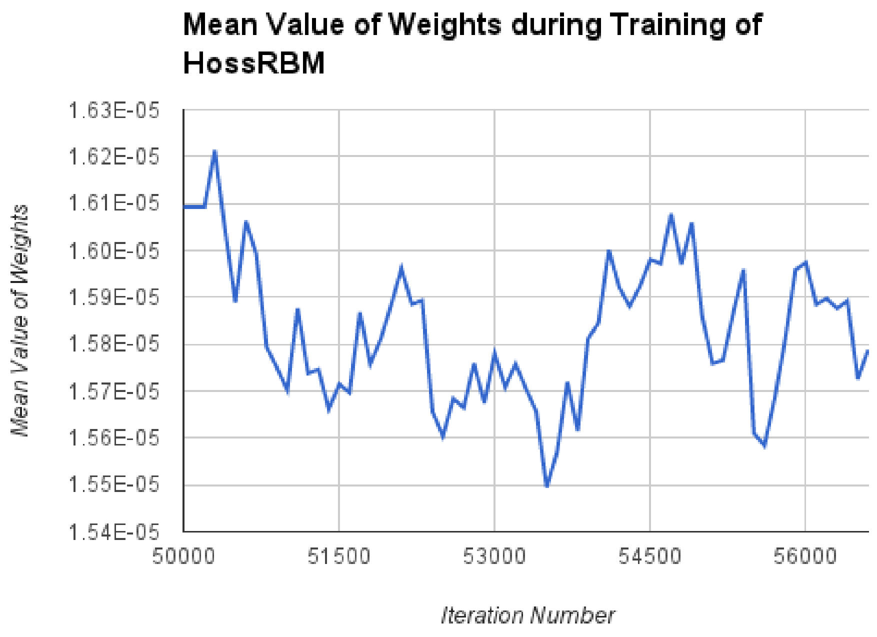


FIGURE 5.8: Graph showing period of stability of weight values before weight explosion during hossRBM training.

Adding Regularisation

Since the problem is the magnitude of weight values become far too large, including a regularisation term, in order to punish increases large values in the weight tensor, seems like it might provide a solution to the problem. Due to the nature of the problem, L2 regularisation seems like a far more obvious choice of solution than L1, since it more heavily punishes large values in the weight matrix. Using L2 regularisation some structure is sporadically learnt by the model, but the model does not seem to converge as completely as the ssRBM and GRBM did. An example of this can be seen in Figure 5.9. A pecular phenomenon is observed with larger regularisation penalties, where some of the filters learnt in early epochs are recognisably based on single training examples.

Preliminary attempts to train a classifier with representations learnt from these models[2] show that the representations are the same regardless of input data, leading to a classifier which just chooses the most common class in the training data.

[2]For this experiment, the default classifier using just the spike variables was used

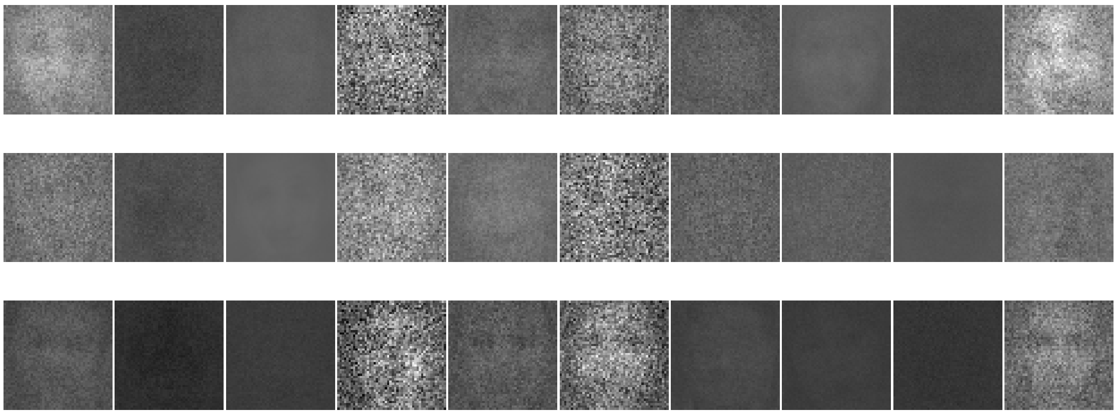


FIGURE 5.9: Examples of weights learnt by hossRBM with a regularisation penalty of 0.1. From top to bottom: epoch 1, epoch 2 and epoch 5

5.7 Summary of Results

In this chapter several experiments have been performed using each of the various models.

The GRBM, has proven to be an effective tool for learning useful representations of facial images, with accuracies which in the worst case match (within error bounds) of the raw pixel representation, and in best cases significantly outperform it.

Using the ssRBM was also demonstrated as a technique with the potential of increasing the classification accuracy of a linear classifer. Although in some cases, the different representations yield significantly different classification accuracies, there does not appear to be a consistent winner. For practical applications, it would therefore be wise to train classifiers using multiple representations available from the model, and select the best representation using classification accuracy on a validation set held out of the training data.

No meaningful representations were learnt using the hossRBM, with the majority of initial conditions leading to a weight explosion, often after a long period of relative stability or otherwise failing to learn any meaningful features at all. The weight explosion can be prevented through regularisation of values in the weight tensor, although it is still not possible to learn a truly useful representation. Trying more variations of hyper-parameters or finding some new insight is required in order to successfully train the hossRBM.

The parameters found in these experiments are by no means optimal, making it hard to do direct comparisons between the models. Both the GRBM and ssRBM perform above the raw pixel representation, although it is not clear which is better, since the best results from each model (34.5% $\pm$ 1.6 and 34.1 % $\pm$ 1.6 respectively) are well within the error bounds of each other.

A primary theme which emerged during these experiments is that RBMs are very sensitive to changes in hyper-parameters, and a great deal of fine-tuning is required to gain useful representations from them.

Chapter 6

Discussion

This chapter serves to provide a discussion of the project. A personal reflection is given, discussing what I believe to be the strengths and weaknesses of the project, the lessons learnt and what would be done differently if I were to carry out a similar project in future. Several suggestions for directions of future work are given along with a brief justification for each.

6.1 Personal Reflection

The models presented in this project have proved to be deeply interesting to me. I find the idea that a statistical model can be made to 'understand' something as complex as facial images very intriguing, and it is topic that I hope to have the chance to investigate further.

Initially, the task of learning to understand the models involved was quite daunting, but the exercise proved to be very enjoyable. Through a process of extensive literature research, as well as spending time playing with the underlying mathematics, I feel I was able to devise a fairly logical and complete exposition of the models used in this project.

I spent a significant amount of time attempting to write an implementation of the hossRBM myself. In retrospect, this was definitely over-ambitious and proved costly, restricting the amount of time remaining for experiments after an implementation was obtained from the authors of the original paper on the model [31]. Whilst it is hard to say whether this extra time would have allowed me to find a set of suitable hyper-parameters for the hossRBM, it would have been nice to be able to investigate some aspects of the model more thoroughly. Similarly, it would have been nice to be able to use validation on some of the parameters in ssRBM, this wasn't done in order to ensure sufficient time was left to train models with different numbers of hidden units.

The training of all models presented in this project proved to be far more problematic than anticipated. In particular, the failure to successfully train the hossRBM model has been extremely disappointing. More generally, it was interesting to discover that something mentioned only passingly in the literature, the difficulty of finding suitable hyper-parameters [20, 45] could prove to be such a major hurdle. Whilst I had originally intended to focus far more on different aspects of the training process (such as the effects of different values of k in the CD-k algorithm, and the likelihood of particular batches of images during training), the difficulties encountered introduced the possibility of focussing more on the effects of hyper-parameters and their effects on training, something far less commonly discussed in the literature (with the notable exception being [18]).

6.2 Future Work

An obvious direction of future work is further exploration of the parameters spaces for each model. Whether this would be performed by carrying out further random sampling of combinations parameters from within a predetermined range, as in [6], or by carrying out a more systematic approach. For this significantly more computational resources would need to be utilised, as the training of models, especially those with a large number of hidden units, is very computationally expensive.

The number of iterations of the gradient ascent algorithm used when training a model in this project was based on numbers used in the literature and time constraints. In

[18] it is suggested that number of iterations can be determined by holding out cross-validation set, with the termination condition being that the probability of training data becomes significantly greater than that of the validation set, this condition terminates the training of the model at the early signs of overfitting. This modification would remove the guesswork involved in determining when to terminate the training of a model.

A weakness of the data set used in this project is that it contains only a single label. The justification for the use of this dataset is that lighting conditions and identity are both far greater causes of variation in an image than facial expression, and therefore an unsupervised learning method which increases the prominence of facial expression in the representation must be in some way disentangling it from the other information components. In order to show this disentangling more rigorously, a multi-label dataset could be used, this would make it possible to test the representation more thoroughly, and discover if any useful information components are being discarded.

For this project, a deliberate decision was made to perform only minimal preprocessing on the dataset, closely following the methodology of [2]. In the literature, there are numerous examples of more complex preprocessing techniques being used. A relatively common practice is the use of either ZCA or PCA algorithms being applied to the training data. These have the effect of removing a significant amount of the noise within the data [20, 45].

Chapter 7

Conclusion

This dissertation was about the ability of three probabilistic models to disentangle information components within facial images in an unsupervised manner. It covered the necessary background material and gave a detailed description of the models, including a significant amount of detail not currently present in the literature.

Promising results were demonstrated with both the Gaussian restricted Boltzmann machine and the spike-and-slab restricted Boltzmann machine. The results achieved with both models demonstrate significant improvements in classification accuracy over the raw pixel representation, with significant reason to believe that further improvement can be achieved from each model. No successful models were trained using the hoss-RBM, but several possible reasons for this failure were investigated and reported. For all models, several directions of future work were suggested based on the results and lessons taken from experiments.

Appendix A

Markov Chains for Gibbs Sampling

This appendix contains the material regarding Markov Chains which is necessary in order to understand Gibbs Sampling. The majority of the material taken from [14], with references gven only for information taken from other sources.

Definition of Markov Chain

A Markov chain is a random process in which the next state is conditionally independent of all previous states, given the current state. Formally, a sequence of random variables $X_1, X_2, \ldots$ sharing a state space Ω are form a Markov chain if

$$P(X^{(k)} = x | X^{(1)} = x_1, X^{(2)} = x_2, \ldots, X^{(n-1)} = x_{n-1}) = P(X^{(n)} = x | X^{(n-1)} = x_{n-1}) \tag{A.1}$$

for all $n \geq 0$ and all $x, x_1, \ldots x_{n-1} \in \Omega$.

The set of probabilities p_{ij}^k form a transition matrix for the kth transition. If for all k the transition matrix p_{ij}^k is the same, the Markov chain is homogeneous. For the purposes of this project, only homogeneous Markov chains will be considered.

Let μ^0 be the distribution of X_0. For a homogeneous Markov chain, the distribution of X_k, μ^k, can be defined as $\mu^0 P^k$.

Stationary Distributions and the Detailed Balance Condition

A stationary distribution of a Markov chain is a distribution π for which $\pi^T = \pi^T P$. A sufficient condition for a Markov chain being stationary is that

$$\pi(i)p_{ij} = \pi(j)\pi_{ji} \tag{A.2}$$

This is known as the *detailed balance condition*. An important property of stationary distributions is that once a distribution reaches a stationary distribution, it will not leave it. That is, if a Markove chain has reached a stationary distribution by time k, we not only get that $\mu^k = \pi$ but also that $\mu^{k+n} = \pi$ for all $n \in \mathbb{N}$

A Markov chain is irreducible if it is always possible to transition from one arbitrary state to any other arbitrary sate in a single transition, that is $\forall i, j \in \Omega\ \exists k > 0$ such that $P(X^{(k)} = i | X^{(0)} = j) > 0$.

The period d_i of a state $i \in \Omega$ is defined as $d_i = \gcd(n \geq 1 | p_{ii}^n > 0)$ [46]. A Markov chain is aperiodic if $d_i = 1$ for all $i \in \Omega$. A sufficient, but not necessary condition for this is that it is always possible to reach a state from itself, that is $p_i i > 0$ for all $i \in \Omega$.

If a Markov chain is both irreducible and aperiodic, then it can be shown that it has a unique stationary distribution. Furthermore, irreducible and aperiodic Markov chains converge to their stationary dstributions. That is, for any initial distribution μ:

$$lim_{k\to\infty} d_v(\mu^T P^k, \pi^T) = 0 \tag{A.3}$$

where d_v is the distance of variation, a measure of the distance between two distributions defined as:

$$d_v(\alpha, \beta) = \frac{1}{2}|\alpha - \beta| = \frac{1}{2}\sum_{x \in \Omega} |\alpha(x) - \beta(x)| \tag{A.4}$$

Appendix B

Confidence Intervals for Classification Accuracy

This appendix pertains to the process of finding confidence intervals for classifier accuracy. The majority of material in this appendix is based on [47].

In order to satisfactorily compare the classification accuracy of multiple classifiers, it is not enough to just obtain the experimental classification accuracy of each, as the figures obtained are only approximations to the true classification accuracy over data sampled from the underlying distribution. For this reason, it is important to be able to devise a measurement of the uncertainty associated with a result, and use the knowledge of this uncertainty to construct a confidence interval.

An experiment to test classification accuracy consists of a series of independent outcomes, with the outcomes being either correct, when the classification label matches the data label, or incorrect, when it does not. The probability of a correct classification is some fixed value p. The number of misclassifications therefore follows the binomial distribution, with the probability of r misclassifications in n experiments being:

$$P(X = r) = \frac{n!}{r!(n-r)!} p^r (1-p)^r$$

Whilst the true expectation is not known, it can be approximated using the number of correct classifications:

$$p = \frac{1}{n} \sum_{i=1}^{n} x_i,$$

where $x_i = 1$ if the ith experiment is correctly classified and is 0 otherwise.

From this s, the standard deviation is calculated as:

$$s = \sqrt{\frac{p(1-p)}{n}}$$

Using these values, and the fact that for large n the binomial distribution is closely approximated by the normal distribution [48], the a confidence interval for classifier accuracy can be constructed as:

$$p \pm z_N s$$

where z_N is the distance from the mean of a normal distribution with unit standard deviation within which N% of area under the distribution curve lies. In this project N is chosen to be 95%, which leads gives $z_N = 1.96$ [47]

Bibliography

[1] Ian Goodfellow, Dumitru Erhan, Pierre-Luc Carrier, Aaron Courville, Mehdi Mirza, Ben Hamner, Will Cukierski, Yichuan Tang, David Thaler, Dong-Hyun Lee, Yingbo Zhou, Chetan Ramaiah, Fangxiang Feng, Ruifan Li, Xiaojie Wang, Dimitris Athanasakis, John Shawe-Taylor, Maxim Milakov, John Park, Radu Ionescu, Marius Popescu, Cristian Grozea, James Bergstra, Jingjing Xie, Lukasz Romaszko, Bing Xu, Zhang Chuang, and Yoshua Bengio. Challenges in representation learning: A report on three machine learning contests, 2013. URL http://arxiv.org/abs/1307.0414.

[2] Guillaume Desjardins, Aaron Courville, and Yoshua Bengio. Disentangling factors of variation via generative entangling. *The Computing Research Repository (CoRR)*, 2012.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. 35(8): 1798-1828, 2013.

[4] Christopher Poultney Marc'Aurelio Ranzato, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In *Proceedings of NIPS*, 2007.

[5] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[6] Aaron C Courville, James Bergstra, and Yoshua Bengio. A spike and slab restricted boltzmann machine. In *International Conference on Artificial Intelligence and Statistics*, pages 233–241, 2011.

[7] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2 (11):559–572, 1901.

[8] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010.

[9] Salah Rifai, Yoshua Bengio, Aaron Courville, Pascal Vincent, and Mehdi Mirza. Disentangling factors of variation for facial expression recognition. In *Computer Vision–ECCV 2012*, pages 808–822. Springer, 2012.

[10] Salah Rifai, Yann N Dauphin, Pascal Vincent, Yoshua Bengio, and Xavier Muller. The manifold tangent classifier. In *Advances in Neural Information Processing Systems*, pages 2294–2302, 2011.

[11] Abdel-rahman Mohamed and Geoffrey Hinton. Phone recognition using restricted boltzmann machines. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4354–4357. IEEE, 2010.

[12] Brendan J Frey. *Graphical models for machine learning and digital communication.* MIT press, 1998.

[13] Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.

[14] Asja Fischer and Christian Igel. An introduction to restricted boltzmann machines. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 14–36. Springer, 2012.

[15] Samson Cheung. Proof of hammersley-clifford theorem. Technical report, Korea Advanced Institute of Science (KAIS), 2008.

[16] Joshua Matthew Susskind. *Interpreting faces with neurally inspired generative models*. PhD thesis, University of Toronto, 2011.

[17] Yoshua Bengio. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[18] Geoffrey Hinton. A practical guide to training restricted boltzmann machines. *Neural Networks: Tricks of the Trade (2nd ed.) 2012: 599-619*, 9(1).

[19] Oskar Sandberg. Markov random fields and gibbs measures. 2004. Chalmers University of Technology.

[20] Jan Melchior. *Learning natural images statistics with Gaussian-Binary Restricted Boltzmann Machines*. PhD thesis, Institut fur Neuorinformatik, 2012.

[21] M Ranzato and Geoffrey E Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2551–2558. IEEE, 2010.

[22] Toby J Mitchell and John J Beauchamp. Bayesian variable selection in linear regression. *Journal of the American Statistical Association*, 83(404):1023–1032, 1988.

[23] Yoshua Bengio, Aaron C Courville, and James S Bergstra. Modeling natural image covariance with a spike and slab restricted boltzmann machine. 2011.

[24] Yoshua Bengio, Aaron C Courville, and James S Bergstra. Unsupervised models of images by spike-and-slab rbms. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1145–1152, 2011.

[25] A tutorial on restricted boltzmann machines. http://www.deeplearning.net/tutorial/rbm.html. Accessed: 2014-06-30.

[26] Léon Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, pages 421–436. Springer, 2012.

[27] Bruce Shapiro. Table of integrals. http://integral-table.com/downloads/single-page-integral-table.pdf. Accessed: 2014-8-17.

[28] I Vilfan. Methods of statistical mechanics. University Lecture Notes. http://www-f1.ijs.si/ vilfan/SM/.

[29] Ruslan Salakhutdinov and Geoffrey E Hinton. Deep boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pages 448–455, 2009.

[30] B Walsh. Markov chain monte carlo and gibbs sampling. *Lecture Notes, MIT.*

[31] Guillaume Desjardin. Code for hossrbm. https://github.com/gdesjardins. Accessed: 2014-06-30.

[32] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM, 2008.

[33] Honglak Lee, Chaitanya Ekanadham, and Andrew Y Ng. Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems*, pages 873–880, 2008.

[34] Nghia Ho. Rbm, l1 vs l2 weight decay penalty for images. nghiaho.com/?p=1817, . Accessed: 2014-08-16.

[35] Nghia Ho. Rbm, l1 vs l2 weight decay penalty for images. http://nghiaho.com/?p=1796, . Accessed: 2014-07-28.

[36] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798. ACM, 2007.

[37] Jason Yosinski and Hod Lipson. Visually debugging restricted boltzmann machine training with a 3d example.

[38] Nan Wang, Jan Melchior, and Laurenz Wiskott. Gaussian-binary restricted boltzmann machines on modeling natural image statistics. *CoRR*, abs/1401.5900, 2014.

[39] George Dahl, Abdel-rahman Mohamed, Geoffrey E Hinton, et al. Phone recognition with the mean-covariance restricted boltzmann machine. In *Advances in neural information processing systems*, pages 469–477, 2010.

[40] Guillaume Desjardins. Personal Communication, 2014-08-05. University of Montreal.

[41] Ian J Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. 2013.

[42] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[43] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[44] Joshua Susskind, Adam Anderson, and Geoffrey Hinton. The toronto face dataset. *UTML TR 2010-001*, 2010.

[45] Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[46] Andrew Ng. Markov chain properties: Irreducibility and aperiodicity. *Chalmers Institute of Technology, Lecture notes.*

[47] Jim Tian. Evalutation of classifiers. *Lecture Notes*, CS573, 2012. University of Iowa, http://www.cs.iastate.edu/ jtian/cs573/WWW/Lectures/lecture06-ClassifierEvaluation-2up.pdf.

[48] Laurence D Brown, T Tony Cai, and Anirban Dasgupta. Confidence intervals for a binomial proportion and asymptotic expansions. *The Annals of Statistics*, 30(1): 160–201, 2002.