Learning Natural Image Statistics with Gaussian-Binary Restricted Boltzmann Machines

Schriftliche Prüfungsarbeit
für die Master-Prüfung des Studiengangs Angewandte Informatik
an der Ruhr-Universität Bochum

vorgelegt von

Jan Melchior

29. Mai 2012

Prüfer 1: Prof. Dr. Laurenz Wiskott
Prüfer 2: PD Dr. Rolf Würtz

Erklärung

Ich erkläre, dass das Thema dieser Arbeit nicht identisch ist mit dem Thema einer von mir bereits für eine andere Prüfung eingereichten Arbeit.
Ich erkläre weiterhin, dass ich die Arbeit nicht bereits an einer anderen Hochschule zur Erlangung eines akademischen Grades eingereicht habe.

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder dem Sinn nach entnommen sind, habe ich unter Angabe der Quellen der Entlehnung kenntlich gemacht. Dies gilt sinngemäß auch für gelieferte Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen.

Datum

Unterschrift

Acknowledgements

This thesis would not have been possible without the help and support of the kind and esteemed people around me.

Foremost, I want to express my special gratitude to my advisor and friend Nan Wang for his continuous support and the collaborative work I really enjoyed and still enjoy. My sincere thanks to my supervisor professor Laurenz Wiskott for his support and the opportunity to write this thesis. His talent of explaining complicated mathematical topics in a clear way was always motivating me. I also like to thank my second supervisor Dr. Rolf Würtz, who already supervised my bachelor thesis. Thanks to Asja Fischer, Oswin Krause and Kai Brügge for the inspiring discussions we had together with professor Laurenz Wiskott and Nan Wang in our regular meetings and in between.

The entire focus on this thesis would not have been possible without the support and forbearance of my beloved fiancée Kathrin Müller. I also like to thank the rest of my family, without them I would not be where I am today.

Finally, I like to thank those three school teachers who always believed in me and whose motivation for their subject and the willing to transfer their knowledge I really appreciate. Thanks to Mrs. Noé-Depiereux, Mrs. Dr. Hofer and Mr. Hofer.

List of Algorithms

List of Tables

List of Figures

List of Abbreviations

AIS	Annealed Importance Sampling
BB-RBM	Binary-Binary Restricted Boltzmann Machine
BM	Boltzmann Machine
CD	Contrastive Divergence
FPCD	Fast Persistent Contrastive Divergence
GB-RBM	Gaussian-Binary Restricted Boltzmann Machine
IC	Independent Component
ICA	Independent Component Analysis
i.i.d.	independent and identically distributed
IS	Importance Sampling
LL	(average) Log Likelihood
LOFS	Location, orientation and frequency selective
log-PDF	Logarithm of the Probability Density Function
MCMC	Markov Chain Monte Carlo
MLE	Maximum Likelihood Estimation
MML	Minimum Message Length
MoG	Mixture of Gaussians
MRF	Markov Random Field
PC	Principal Component
PCA	Principal Component Analysis
PCD	Persistent Contrastive Divergence
PDF	Probability Density Function
PLL	Pseudo Log Likelihood
PoE	Product of Experts
PT	Parallel Tempering
RBM	Restricted Boltzmann Machine
ZCA	Zero Phase Component Analysis

Contents

1 Introduction

Understanding our visual system is one of the major challenges in neuroscience and computer vision. It is amazing how fast the human's brain processes a visual input and analyses the scene. But the process as a whole, due to its complexity, is rarely understood in detail and therefore remains a challenging research area.
When analysing the visual system, a necessary task is to understand the first processing steps named early vision. Early vision is processed in the retina, the LGN and the primary visual cortex V1 and is, due to its mostly feed forward architecture relatively well understood. Since the visual system has evolved and been optimized over millions of years through evolution, it is assumed to process vision in a rather good way. Consequently, when modelling natural image statistics we assume that a good model models the visual input by extracting the same statistical properties as our visual system does.
The simple cells in the primary visual cortex have a localized receptive field focused on a local subregion of the visual input. These cells extract location, orientation and frequency selective features from the corresponding part of the input signal. They are often modelled directly using Gabor functions [23], but a more general and desired approach is to use a statistical model that learns these features automatically.

1.1 Previous and Related Work

Restricted Boltzmann machines (RBM) have gained popularity over the last decade as almost no other statistical model. This is mainly due to the fact that they can be stacked layer-wise to build deep neural networks [4, 16, 29] being capable of capturing higher order statistics. Beginning with training the first RBM on the input data, each following RBM is then trained on the output of the previous layer.
The original RBM has binary visible and hidden units, abbreviated by (BB-RBM). But a popular variant named Gaussian-Binary Restricted Boltzmann Machine (GB-RBM) [44] is capable of modelling continuous data, like natural images. A major disadvantage of RBMs is that they are known to be difficult to train [13], which seems to become even more critical when using GB-RBMs [43].

Several modifications have been proposed to overcome the problems during training, which usually prevent the model to learn meaningful features. In [28], the authors added a sparseness penalty on the gradient that forced the model to prefer sparse representations and seems to help learning meaningful features. Recently, the authors in [7] suggested that the training failure is due to the training algorithm and proposed several improvements to overcome the problem. The authors in [24] successfully trained a deep hierarchical network and concluded that a failure is mainly because of the existence of high-frequency noise in natural images, which prevents the model from learning the important structures. Other approaches modified the model such that it is capable of modelling higher order statistics directly [9, 36, 37]. All modifications showed that GB-RBMs are in principle capable of learning features comparable to the receptive fields in the early primary visual cortex V1, but in practice this is difficult to achieve.

To derive a better understanding of the limitations of the model, the authors in [26] evaluated its capabilities from the perspective of image reconstruction. In [40] the likelihood of the model is compared to classical machine learning methods. Although the model has been analysed to show the failures empirically, there are few works accounting for the failure analytically.

1.2 Aims and Contributions

The thesis has two major aims.

1. To give a consistent and comprehensive introduction to RBMs, and the related concepts. It therefore became a reference book, at least for myself and I hope it will be useful for other people too.

2. To analyse GB-RBMs for modelling natural image statistics. Therefore, the reader gets introduced briefly into natural images and why it is important to be able to model them. The model is then analysed concerning the way it models data and consequently, how its probability density function is structured. For this purpose, the property is used that GB-RBMs can be reformulated as a constrained Mixture of Gaussians (MoG). This has been analysed in [43] and has already been mentioned in previous studies [3, 40]. It presents a much clearer view on the models probability density function (PDF) than the Product of Experts formulation and shows that GB-RBMs are highly limited in the way they can represent data. Moreover, it allows to conclude how GB-RBMs should be trained more efficient and reliable on natural images. Considering the formulas similarity, GB-RBMs are compared to Independent Component Analysis (ICA), which can also be given as a Product of Expert formulation and is known to be a good model for natural images.

   Several experiments should show how GB-RBMs model natural images and that the learned features are similar to the features learned by ICA. Therefore, the results of GB-RBMs are compared to the results of the related models, i.e. ICA and MoGs. It is shown that the preprocessing of the data plays an important role and that the analysis allows to choose a training setup that leads to fast and stable training of GB-RBMs.

All of the research described in this thesis has been done in collaboration with my advisor Nan Wang, so that it is also a major part of his Ph.D. research. Consequently, this thesis is also a reference text of our collaborative research.

1.3 Structure of this Thesis

The first Chapter explains the motivation and structure for this thesis and gives an overview of the related work.

The second Chapter introduces the reader into natural image statistics and how our visual system is modelling it.

The third Chapter gives an introduction to Boltzmann machines (BM) in general and a detailed derivation of RBMs with binary visible and hidden units as well as Gaussian visible and binary hidden units. The training methods based on the maximum likelihood estimation and related concepts are explained in detail.

Chapter four shows the reformulation of GB-RBMs as an MoG and why the model is limited in its representational power compared to an unconstrained MoG. The limitations lead to the conclusion that whitening the input data is an important preprocessing step. Further, the model is compared to ICA.

Chapter five shows that only whitened data leads to the desired location, orientation and frequency selective features. The results are compared to ICA and MoGs and the model is trained for different numbers of hidden units. The results for training the variance are shown and the performance of the different training algorithms are compared. Finally, the involved hyperparameters are analysed.

1.4 Mathematical Notations

This work tries to follow the common mathematical notions. Special notations are rare and will be explained at their first occurrence. Although I think that the notations are self explaining, a brief introduction to the frequently used notations is given here, in order to prepare the reader for this work.

Vectors are per default column vectors!

Notation	Explanation	Example
Scalar	Lower-case letter	x
Vector	Lower-case, bold letter	$\mathbf{x}$
Matrix	Upper-case, bold letter	$\mathbf{X}$
PDF	Upper-case P or Q	$P\left(\cdot\right)$
Unnormalized PDF	P or Q with tilde	$\tilde{P}\left(\cdot\right)$
Partition function	Upper case Z	Z
Expectation value	Double lined E	$\mathbb{E}_{P(\cdot)}$
Average value	Angle braces	$\langle\cdot\rangle_x$
Euclidean Norm	Double lines	$\lVert\cdot\rVert$
Fraction bar	Denotes always a component wise division	$\frac{\mathbf{x}}{\mathbf{y}}$
Multiplication Symbol	The cross is used for highlighting a multiplication	×
Matrix/Vector transpose	Superscript T, do not confuse with the temperature parameter T	$\mathbf{x}^T$
Indexing	Indices are lower-case subscript and/or superscript letters	x_i^j
Matrix row selection	Denotes a column vector containing the values of the i^{th} row of matrix $\mathbf{X}$	$\mathbf{x}_{i*}$
Matrix column selection	Denotes a column vector containing the values of the j^{th} column of matrix $\mathbf{X}$	$\mathbf{x}_{*j}$
RBM's visible values	Size Nx1 containing the visible values x_i	$\mathbf{x}$
RBM's hidden values	Size Mx1 containing the hidden values h_j	$\mathbf{h}$
RBM's Weight matrix	Size NxM containing the weight values w_{ij}	$\mathbf{W}$
RBM's visible bias	Size Nx1 containing the visible bias values b_i	$\mathbf{b}$
RBM's hidden bias	Size Mx1 containing the visible bias values c_j	$\mathbf{c}$
GRBM's visible's variance	Size Nx1 containing the variance values σ_i	$\boldsymbol{\sigma}$

2 Natural Images

The biological visual system has evolved over millions of years by adapting to the sensory input from the individuals' environment. Accordingly, we assume our visual system to be adapted to the environment our ancestors have been living in over millions of years. This environment is obviously the natural environment and not our mostly artificial environment, like cities, in which we are living today.
The term ”natural images” [21] denotes all photographs showing typical scenes of the natural environment. Figure 1 shows two examples, which are the kind of sensory inputs that we assume our visual system has been adapting to.

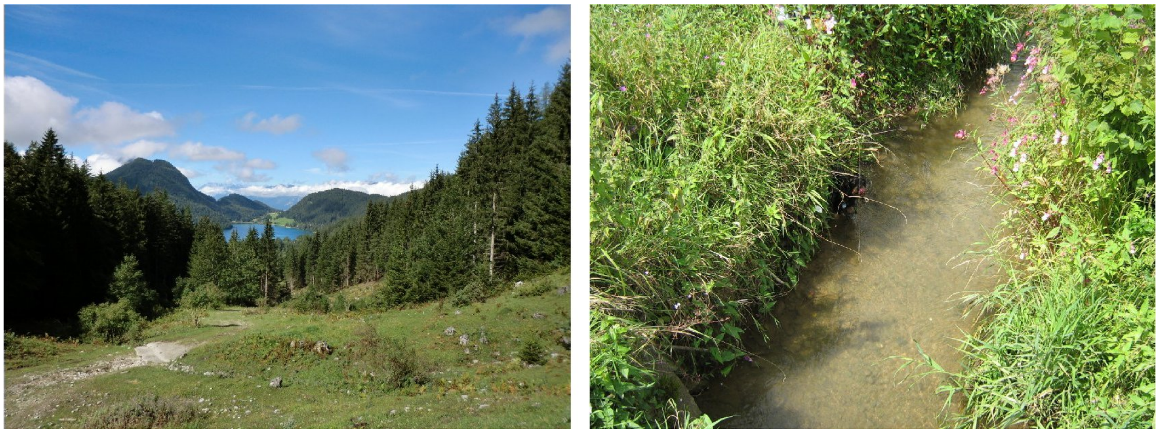


Figure 1: Two natural images.

Although the sensory input for our visual system is a continuous signal, our retina as well as digital cameras quantize the signal, which somehow justifies the work with digital images. Additionally, we assume the information used for analysing a scene to be mainly independent of color information. This becomes clear if you compare Figure 1 to Figure 2, which shows the same image but converted to grey scale. Apart from color depending classification tasks we are able to analyse the scenes without using color information. Furthermore, using grey scale images reduces the data dimensionality by a factor of three, which gives a computational advantage.

Let us now consider the set of digital images that show all possible scenes you can imagine. It contains all possible natural scenes as well as all scenes all humans have seen in their lifetime and many more. Although this set is already incredibly big it is only a very small subset of all images that are possible. The biggest subset are

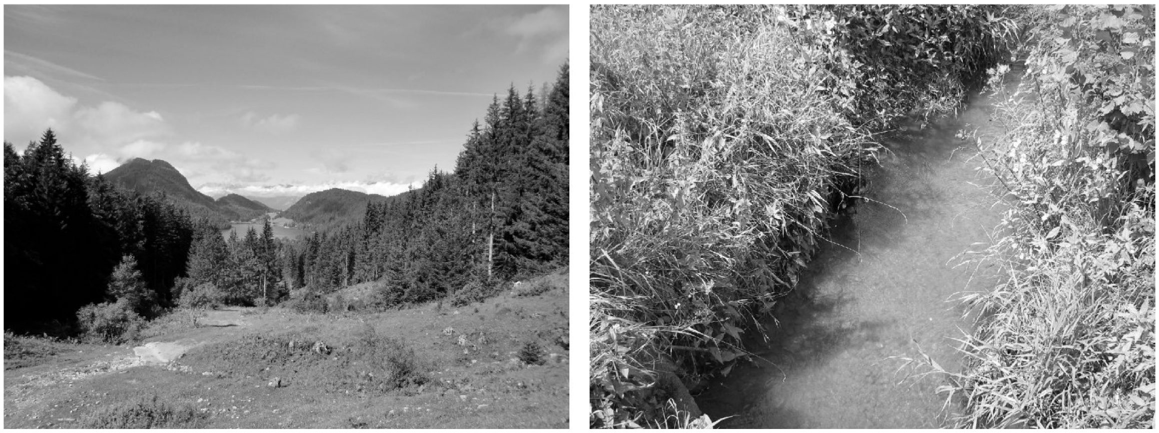


Figure 2: The two natural images of Figure 1 converted to grey scale.

noise images and images that look like noise. How big the number of those images is, compared to images that we denote as meaningful, becomes clear if we generate random images by setting every pixel randomly and independently. Then all possible images are equally likely to appear but what we get are only noise images similar to the image shown in Figure 3.

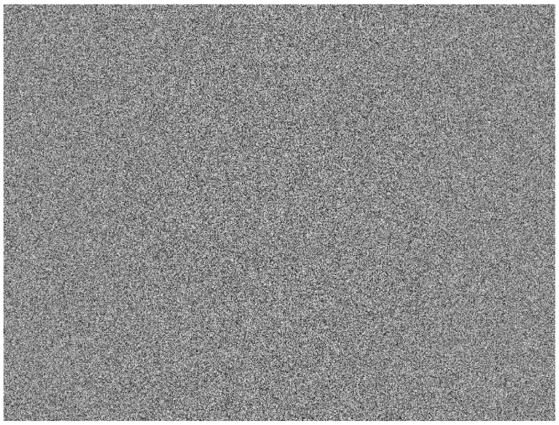


Figure 3: A noise grey scale image generated by setting each pixel independently to a random value.

2.1 Optimal Codes

From an information theoretical point of view we can now argue that a pixelwise representation of natural images is an inefficient code. This can be formalized using the Minimum Message Length [42] (MML). It postulates that given an optimal code C_{opt}, which uses a representation under base b, that the length of the code for an event E is equal to the negative logarithm of its probability given by:

$$| \, C_{opt} \, (E) \, |_b \; = \; - \log_b \left(P \left(E \right) \right) \; , \tag{1}$$

where $| \cdot |_b$ denotes the length of the code under base b. Accordingly, a good code produces short codes for likely events, long codes for unlikely events and it becomes optimal if $P(E)$ is the true distribution for all E. Consequently, the search for the shortest average code length is equivalent to the search for the true PDF.
The average code length is given by:

$$\begin{aligned} \langle | \, C_{opt} \, (E) \, | \rangle_{P(E)} &= \int P(E) \, | \, C_{opt} \, (E) \, |_b \, \mathrm{d}E \, , && (2) \\ &= - \int P(E) \, \log_b \left(P \left(E \right) \right) \, \mathrm{d}E \, , && (3) \\ &= H_b[E] \, , && (4) \end{aligned}$$

where $H_b[E]$ denotes the entropy of event E under base b, which is a measure for uncertainty or unpredictability. Consequently, reducing the average code length is equivalent to reducing the entropy by choosing a better PDF. Since we are usually not able to represent the true PDF, a good estimations should represent the most important structure of the data, which is equivalent to choosing a code with less redundancy.

We can show that the pixelwise representation of natural images is an inefficient code, which is the basis for all compression algorithms. Using a code with fixed length M for all events implies that we assume the events being distributed uniformly given by:

$$\begin{aligned} & - \log_2 \left(P \left(E \right) \right) = M \, , \forall \, E \, , && (5) \\ \Leftrightarrow \quad & P \left(E \right) = 2^{-M} \, , \forall \, E \, , && (6) \\ \Leftrightarrow \quad & P \left(E \right) = \frac{1}{2^M} \, , \forall \, E \, , && (7) \end{aligned}$$

where we used base two because of the computer's binary representation. If we now assume the total number of all natural images to be 2^M and the number of all other possible images to be 2^N, then we use $N + M$ bits to represent each image. But we would only need M bits if we consider only the natural images, which implies N bits redundant information if we use a code of length $N + M$.

The assumption that the natural images are uniformly distributed is obviously incorrect and therefore a fixed code length cannot be optimal. But as already mentioned, we are able to generate a better code by finding a PDF that represents the data better.

A good estimation of a PDF represents the important structure of the data and therefore the goal becomes to identify the important structure present through all images. It is obvious to see that neighbouring pixels tend to have a similar color or grey value. These pixels are dependent and consequently contain redundant information. Therefore, a representation where each component is independent of each other promises to be a more efficient code.

2.2 Independent Components

Two random variable x_i and x_j are statistically independent if their joint probability is equivalent to the product of their marginal probabilities given by:

$$P(x_i, x_j) \;=\; P(x_i)\,P(x_j)\,. \tag{8}$$

Informally speaking, knowing the value x_i does not give us any information about x_j and vice versa. Formally this is denoted by $x_i \perp\!\!\!\perp x_j$.

Assume we have two independent variables $x'_i \in [0, 1]$ and $x'_j \in [0, 1]$, their joint probability will be uniform as shown in Figure 4 on the left.

Now consider the same distribution rotated by 30° around the origin as shown in Figure 4 on the right. The new variables x_i and x_j are not independent any more since a high value for x_i implies a small value for x_j and vice versa. Consequently, the marginal distributions shown beside the axis are not uniform any more. They look more like a Gaussian distribution, which comes from the central limit theorem. It states that the sum or mixture of N independent and identically distributed (i.i.d.) random variables will become more Gaussian as more variables we add. Therefore, the directions which are most independent are the directions where the marginal distributions are most non-Gaussian.

Given the rotated data we are able to restore the unrotated version if we know the transformation that rotates the data to the directions of most non-Gaussianity. This is known as Independent Component Analysis. But we want to be able to identify

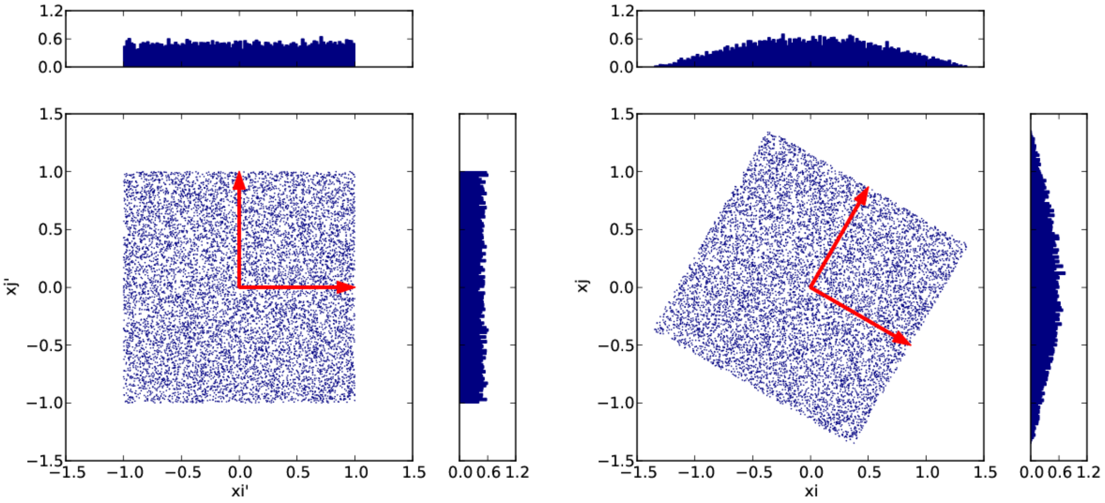


Figure 4: (left) Illustration of two random variables drawn independently from a uniform distribution between $[-1, +1]$. The histograms along the axis represent the marginal PDFs. (right) The same distribution rotated by 30°, where the histograms show that the variables are distributed more Gaussian like, which implies dependence.

the IC's under all affine transformations and we know that the IC's in the independent representation are orthogonal. So if we could guarantee that the IC's are still orthogonal in the transformed version of the data, then the problem reduces to finding a rotation matrix as described before.
Luckily there is a transformation which does this, named whitening. Whitening removes the mean of the data and transforms it such that it has unit variance in all, not only the coordinate axis directions. This causes the IC's to be orthogonal. Therefore, whitening is an important preprocessing step for almost all ICA algorithms and will be discussed in Chapter 4. Figure 5 shows the IC's for small natural images patches of size 14 times 14 pixels, which show localization, orientation and frequency selective (LOFS) structures.

2.3 Early Vision

The discussion so far was motivated by finding a good representation for natural images. We already mentioned that we assume the visual system to be adapted in an evolutionary process to natural images. It is therefore most natural to have a

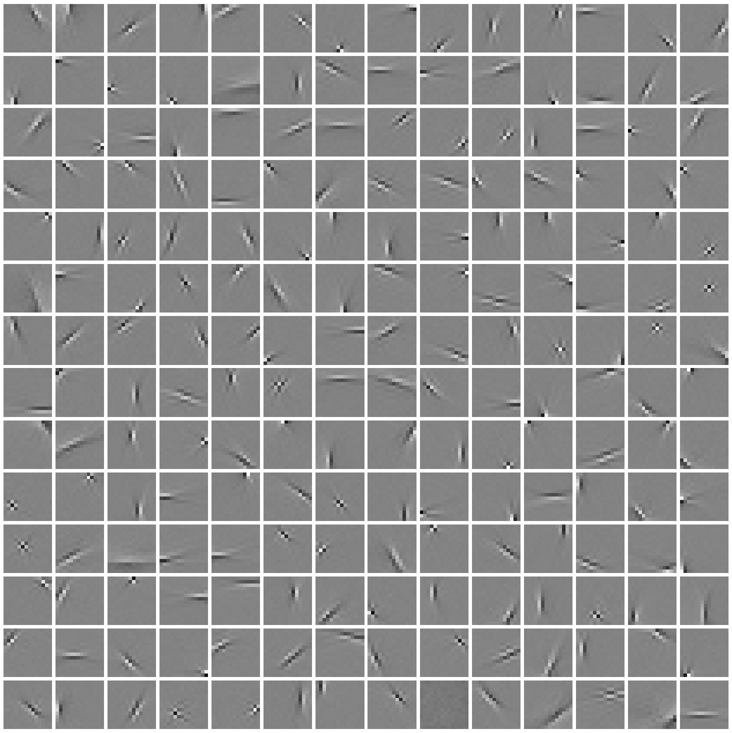


Figure 5: Independent components of small natural image patches of size 14x14.

closer look to the image processing in our brain.

When neuroscientists analysed the primary visual cortex they found mainly so called simple cells, which process the visual input received on the retina and preprocessed in the lateral geniculate nucleus (LGN). These cells have a localized receptive field, meaning that they are focused on a local subregion of the input signal. A lot of cells are connected to the same subregion so that all these cells together will represent the corresponding part of the signal. Consequently, there exist a group of cells for each subregion. The scientists discovered that the receptive fields within a group connected to the same subregion, have a similar structure as the IC's shown in Figure 5. For comparison see Figure 6, which shows the receptive fields recorded from a Macaque monkey, which are assumed to be similar to the human simple cell receptive fields. Each group of cells has in principle the same receptive fields since they need to be able to model the same input signal. This is motivated by the fact that the input signal can be shifted around by moving the eyes or the head.

These LOFS receptive fields are often modelled using a two dimensional Gabor function [23], which is basically a harmonic function multiplied with a Gaussian function

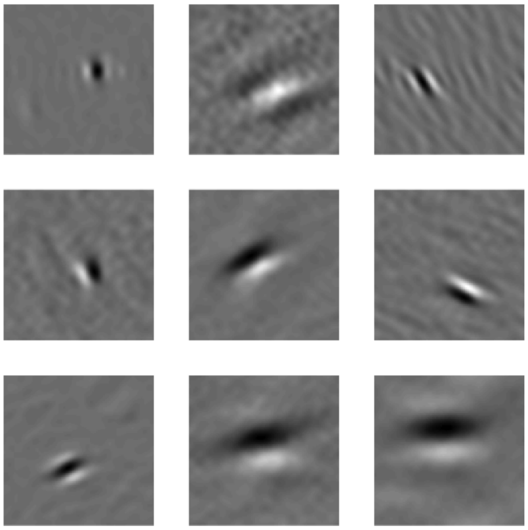


Figure 6: Receptive fields of simple cells in a Macaque monkey's brain. Courtesy of Dario Ringach, UCLA.

given by:

$$g_{\lambda,\theta,\phi,\sigma,\gamma}(x,y) \;=\; \exp\left(-\frac{\tilde{x}^2+\gamma^2\tilde{y}^2}{2\sigma^2}\right)\cos\left(2\pi\frac{\tilde{x}}{\lambda}+\phi\right)\,, \tag{9}$$

with

$$\tilde{x} = x\cos\theta + y\sin\theta\,, \tag{10}$$

$$\tilde{y} = -x\sin\theta + y\cos\theta\,, \tag{11}$$

where λ is the wavelength of the sinusoidal, θ represents the orientation, ϕ is the phase offset, σ is the standard deviation of the Gaussian, γ specifies the ellipticity of Gabor function.
Figure 7 shows four exemplary receptive fields produced by the Gabor function with different orientations, frequencies and scales, on the right and the illustration of the receptive field structure on the left. Dark regions correspond to a negative activation while light regions correspond to a positive activation. Gabor functions are a very popular approach for face recognition [14, 45].

The neuroscientist also discovered that the simple cells are rarely active, which means given an input signal only a few cells are firing at the same time. So that we assume the IC of natural images to be distributed sparsely [33]. Consequently, also sparse coding leads to filters similar to the receptive fields of simple cells as shown in [34].

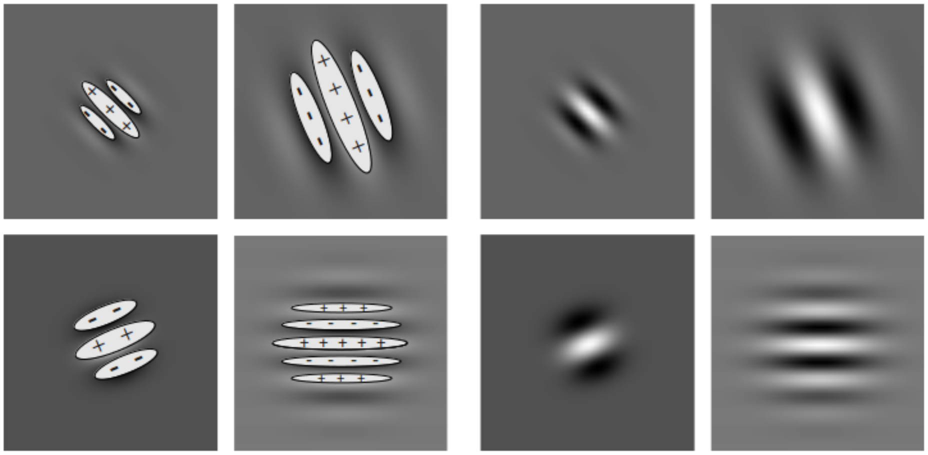


Figure 7: Illustration of four exemplary Gabor wavelets of different orientation, frequency and scale.

Simple cells can also be modelled using a standard model neuron, given by:

$$\begin{aligned} \mathbf{y} &= \sigma\left(\sum_{i}^{N} w_i x_i\right) && (12) \\ &= \sigma\left(\mathbf{w}^T\mathbf{x}\right) , && (13) \end{aligned}$$

where w_i are weights, one for each input signal x_i and $\sigma(\cdot)$ is an activation function, which is usually chosen to be a non linear function like the sigmoid function $\frac{1}{1+\mathrm{e}^{-x}}$. The weights describe the receptive field of this cell, so that we can model a simple cell by choosing the weights to be a receptive field as shown in Figure 6. However, it is more interesting to see whether model neurons learn filters showing LOFS structures in an unsupervised way. We therefore need some unsupervised neural network model that consists of model neurons.

3 Restricted Boltzmann Machines

This chapter introduces the reader to an unsupervised artificial neural network named Boltzmann machines. The chapter begins with a general introduction and a detailed derivation of their more popular variant restricted Boltzmann machines. This is followed by a detailed discussion of the original restricted Boltzmann machines, which works on binary data and a variant that allows to handle continuous data like images, named Gaussian-binary restricted Boltzmann machines. Finally the training and the related concepts are discussed in detail.

3.1 Boltzmann Machines

A BM [17] is an undirected probabilistic graphical model [5] with stochastic continuous or discrete units. It is often interpreted as a stochastic recurrent neural network where the state of each unit depends on the units it is connected to. The original BM has a fully connected graph with binary units, which turns into a Hopfield net if we choose deterministic rather than stochastic units. But in contrast to Hopfield nets, a BM is a generative model that allows to generate new samples from the learned distribution.

Usually the graph of a BM is assumed to be divided into a set of observable visible units $\mathbf{x}$ and a set of unknown hidden units $\mathbf{h}$ called visible and hidden layer, respectively. Additionally, the graph has a visible and hidden bias that are units having a constant input of one.

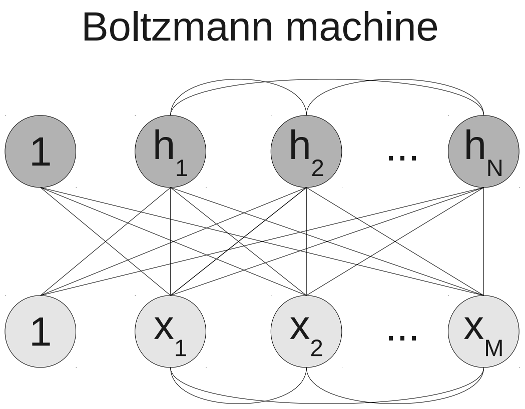


Figure 8: The graphical structure of a Boltzmann machine given as a complete undirected graph, with N visible units, M hidden units, visible and hidden bias.

A graph of a BM with N visible units, M hidden units, visible and hidden bias is shown in Figure 8. The values for the visible layer are considered to be known data points, while the hidden units are latent variables forming a conditional hidden representation of the data. This allows to transfer a given visible state to a hidden representation and vice versa.

An important property of BMs is that they are stackable, which means that we are able to train a BM on the hidden representation of another BM. This allows to construct deep networks [3] for learning complex probability densities where the layers can be trained one after each other, which makes them become very popular in the field of deep learning. But BMs are also popular in the field of feature extraction [24] and dimensionality reduction [16].

3.1.1 Product of Experts and Markov Random Fields

A BM as we will see, is a special case of a Markov Random Field (MRF) [5], which itself is a special case of a Product of Experts (PoE) [18]. Thus understanding MRFs, PoEs and how they are related is important for a profound understand of BMs.

A PoE with input variable $\mathbf{x}$ and latent variables $\mathbf{h}$, defines a PDF over the given input space, $\mathbf{x}, \mathbf{h} \in \mathbf{X}, \mathbf{H}$ by taking the product of individual components $\phi_c(\mathbf{x}, \mathbf{h})$. These components, named experts, are themselves not necessarily normalized probabilistic models, but their product needs to be normalized in order to form a valid PDF. This is achieved by the normalization constant Z^{PoE}, named partition function, which integrates over all possible states $\tilde{\mathbf{x}}, \tilde{\mathbf{h}} \in \tilde{\mathbf{X}}, \tilde{\mathbf{H}}$. The PoE is defined as:

$$P^{PoE}\left(\mathbf{x}, \mathbf{h}\right) = \frac{1}{Z^{PoE}} \prod_{c}^{C} \phi_c\left(\mathbf{x}, \mathbf{h}\right), \tag{14}$$

with partition function,

$$Z^{PoE} = \int \int \prod_{c}^{C} \phi_c(\tilde{\mathbf{x}}, \tilde{\mathbf{h}}) \, \mathrm{d}\tilde{\mathbf{x}} \, \mathrm{d}\tilde{\mathbf{h}}\,, \tag{15}$$

where the corresponding integral turns into a sum over all possible states in the case of discrete visible or hidden units.
Since we multiply the individual probabilities of the experts, it is obvious that we

only get a high overall probability if all experts assign high individual probabilities. The PoE can therefore be interpreted as a council that judges a presented sample as being important if the judgement is unanimous. This stays in contrast to a mixture model [5] where the individual probabilities for a presented sample are summed up. Consequently, in a mixture model an expert or mixture can possibly overrule the others and the overall probability will only be low if all mixtures assign low probability.

We now consider the particular case where the experts are chosen from the family of exponential functions defined by:

$$\phi_c^{MRF}(\mathbf{x}, \mathbf{h}) = \mathrm{e}^{-\frac{1}{T}\,\psi_c(\mathbf{x},\mathbf{h})}\,, \tag{16}$$

where the potential function $\psi_c\,(\mathbf{x}, \mathbf{h})$ defines the interaction between visible and hidden units of expert ϕ_c^{MRF}. It can be regularized by the constant $T \in [1, \infty)$ known as temperature. If we substitute (16) in (14) it turns out that a PoE model with exponential experts is an MRF with input variables $\mathbf{x}$ and latent variables $\mathbf{h}$, which is expressed by the Hammersley and Clifford theorem [13]. An MRF is defined by a Gibbs distribution also known as Boltzmann distribution by:

$$\begin{aligned} P^{MRF}\,(\mathbf{x}, \mathbf{h}) \quad &\overset{(14),(16)}{=} \quad \frac{1}{Z^{MRF}} \prod_{c}^{C} \mathrm{e}^{-\frac{1}{T}\,\psi_c(\mathbf{x},\mathbf{h})}\,, && (17)\\ &= \quad \frac{1}{Z^{MRF}}\,\mathrm{e}^{-\frac{1}{T}\,\sum_c^C \psi_c(\mathbf{x},\mathbf{h})}\,, && (18)\\ &= \quad \frac{1}{Z^{MRF}}\,\mathrm{e}^{-\frac{1}{T}\,E(\mathbf{x},\mathbf{h})}\,, && (19) \end{aligned}$$

and the partition function becomes,

$$Z^{MRF} \quad \overset{(15),(16)}{=} \quad \int\!\!\int \mathrm{e}^{-\frac{1}{T}\,E\left(\tilde{\mathbf{x}},\tilde{\mathbf{h}}\right)}\ \mathrm{d}\tilde{\mathbf{x}}\ \mathrm{d}\tilde{\mathbf{h}}\,. \tag{20}$$

The function $E\,(\mathbf{x}, \mathbf{h})$ known as energy between $\mathbf{x}$ and $\mathbf{h}$, is equivalent to the sum over the potentials and defines, which and how units interact. It therefore defines the complexity of the model, which is usually interpreted as a graph.

3.1.2 Boltzmann Machines

While an MRF is a particular case of a PoE, a BM is an MRF with a particular energy function that leads to a complete undirected graph as shown in Figure 8.

This implies a fully connected network where the pairwise communication between two units is symmetrical. The activation of each node is given by the sum over all values of its incoming connections. A general definition for BMs [44] can therefore be given by:

$$\begin{aligned} E^{BM}(\mathbf{x}, \mathbf{h}) \;=\; & -\sum_{i,a}^{N,A} b_i^a \, \alpha_i^a(x_i) - \sum_{j,d}^{M,D} c_j^d \, \beta_j^d(h_j) - \sum_{i,j,a,d}^{N,M,A,D} \alpha_i^a(x_i) \, w_{ij}^{ad} \, \beta_j^d(h_j) - \\ & - \sum_{i,k=i+1,a}^{N,N,A} \alpha_i^a(x_i) \, v_{ik}^a \, \alpha_k^a(x_k) - \sum_{j,l=j+1,d}^{M,M,D} \beta_j^d(h_j) \, u_{jl}^d \, \beta_l^d(h_l) \,. \end{aligned} \tag{21}$$

Where $\alpha_i^a(x_i)$ and $\beta_j^d(h_j)$ are one dimensional transfer functions, mapping a given input value to a desired feature value. They are the sufficient statistics of the model and can be arbitrary non-parametrized functions of the input variable x_i or h_j, respectively, but they need to be independent of the parametrization. The index a and d denote that there can be multiple transfer functions per variable. The first sum only depends on the visible units and the second term only depends on the hidden units, so that b_i^a and c_j^d could be interpreted as the corresponding visible and hidden bias, respectively. The inter layer connection term w_{ij}^{ad} connects the visible units with the hidden units. The intra layer connection term v_{ij}^a connects the visible units with each other and u_{ij}^d connects the hidden units with each other, respectively.

This formalism allows to define even complexer BMs where more than two units interact with each other, named higher order BMs [36]. But a major disadvantage of BMs in general is that it is usually intractable to calculate the partition function since the integration over all possible states is only computable for small toy problems.

Therefore, training BMs is usually done by approximations using sampling methods [5], which will be described in detail later. So far it is just important to note that for those sampling methods we need to be able to calculate the conditional probability of the visible units given the hidden units and vice versa. Using Bayes theorem we can derive the conditional probability of the hidden units given the visible values given by:

$$
\begin{aligned}
P^{BM}(\mathbf{h}|\mathbf{x}) &= \frac{P^{BM}(\mathbf{x},\mathbf{h})}{\int P^{BM}(\mathbf{x},\tilde{\mathbf{h}})\,\mathrm{d}\tilde{\mathbf{h}}}\,, && (22)\\
&\overset{(19),(21)}{=} \frac{\frac{1}{Z^{BM}}\mathrm{e}^{E^{BM}(\mathbf{x},\mathbf{h})}}{\frac{1}{Z^{BM}}\int \mathrm{e}^{E^{BM}(\mathbf{x},\tilde{\mathbf{h}})}\,\mathrm{d}\tilde{\mathbf{h}}}\,, && (23)\\
&= \frac{\mathrm{e}^{E^{BM}(\mathbf{x},\mathbf{h})}}{\int \mathrm{e}^{E^{BM}(\mathbf{x},\tilde{\mathbf{h}})}\,\mathrm{d}\tilde{\mathbf{h}}}\,. && (24)
\end{aligned}
$$

Due to the symmetry of a BM we get the conditional probability of the visible units given the hidden units in the same way. The partition function cancels out but the equations still contain a high dimensional integration over all possible hidden values. The exact calculation is usually intractable and even the approximation of high dimensional integrals is difficult, so that training algorithms become very slow and they tend to fail for models of moderate size.
An important subclass of BMs having a restricted communication structure allows an efficient calculation of the conditional probabilities. So that fast inference is possible, which made restricted BMs become very popular over the last decade.

3.1.3 Restricted Boltzmann Machines

A simplification where all lateral connections between visible units and all lateral connections between hidden units are removed, is a so called Restricted Boltzmann Machine (RBM). The RBMs structure is a bipartite graph where visible and hidden units are pairwise conditionally independent, shown in Figure 9.
Considering the general energy of a BM (21) we get a general definition for an RBM, if we remove the intra connection terms by setting v_{ij} and u_{ij} to zero, which leads to:

$$E^{RBM}(\mathbf{x},\mathbf{h}) = -\sum_{i,a}^{N,A} b_i^a\,\alpha_i^a(x_i) - \sum_{j,d}^{M,D} c_j^d\,\beta_j^d(h_j) - \sum_{i,j,a,d}^{N,M,A,D} \alpha_i^a(x_i)\,w_{ij}^{ad}\,\beta_j^d(h_j)\,. \quad (25)$$

The general definition for RBMs was given in [44] in a slightly different notation.

The major advantage of RBMs is that the units of the visible layer are conditional independent and so are the units of the hidden layer. This leads to a general factorization property of RBMs when marginalizing out the visible or hidden units. The integral over all possible states of the visible layer factorizes into a product of one

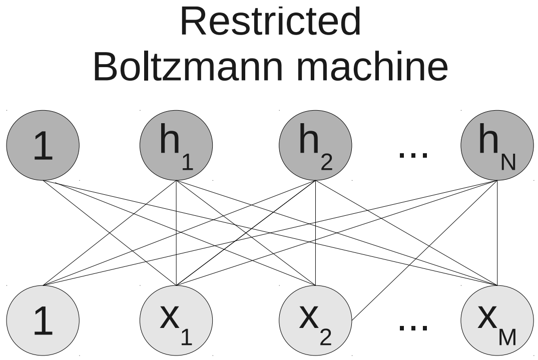


Figure 9: The graphical structure of a restricted Boltzmann machine, given as a bipartite undirected graph, with N visible units, M hidden units, visible and hidden bias.

dimensional integrals over all possible values for the corresponding unit. Therefore, the conditional probability can be calculated efficiently, which makes sampling methods used for inference work very well for RBMs.
The marginal probability distribution for the visible units is given by:

$$\begin{align}
P^{RBM}(\mathbf{x}) &= \int P^{RBM}(\mathbf{x},\tilde{\mathbf{h}})\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{26}\\
&\overset{(19),(20)}{=} \frac{1}{Z^{RBM}} \int \mathrm{e}^{E^{RBM}\left(\mathbf{x},\tilde{\mathbf{h}}\right)}\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{27}\\
&\overset{(25)}{=} \frac{1}{Z^{RBM}} \int \mathrm{e}^{\sum_{ia} b_i^a\,\alpha_i^a(x_i)+\sum_{jd} c_j^d\,\beta_j^d(\tilde{h}_j)+\sum_{ijad}\alpha_i^a(x_i)\,w_{ij}^{ad}\,\beta_j^d(\tilde{h}_j)}\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{28}\\
&= \frac{1}{Z^{RBM}}\,\mathrm{e}^{\sum_{ia} b_i^a\alpha_i^a(x_i)} \int \prod_j^M \mathrm{e}^{\sum_d c_j^d\beta_j^d(\tilde{h}_j)+\sum_{iad}\alpha_i^a(x_i)w_{ij}^{ad}\beta_j^d(\tilde{h}_j)}\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{29}\\
&= \frac{1}{Z^{RBM}}\,\mathrm{e}^{\sum_{ia} b_i^a\alpha_i^a(x_i)} \left(\int \mathrm{e}^{\sum_d c_1^d\beta_1^d(\tilde{h}_1)+\sum_{iad}\alpha_i^a(x_i)\,w_{i1}^{ad}\,\beta_1^d(\tilde{h}_1)}\,\mathrm{d}\tilde{h}_1\right. \nonumber\\
&\quad \times \int \mathrm{e}^{\sum_d c_2^d\beta_2^d(\tilde{h}_2)+\sum_{iad}\alpha_i^a(x_i)\,w_{i2}^{ad}\,\beta_2^d(\tilde{h}_2)}\,\mathrm{d}\tilde{h}_2 \times \ldots \nonumber\\
&\quad \left.\ldots \times \int \mathrm{e}^{\sum_d c_M^d\beta_M^d(\tilde{h}_M)+\sum_{iad}\alpha_i^a(x_i)\,w_{iM}^{ad}\,\beta_M^d(\tilde{h}_M)}\,\mathrm{d}\tilde{h}_M\right)\,, \tag{30}\\
&= \frac{1}{Z^{RBM}}\,\mathrm{e}^{\sum_{ia} b_i^a\alpha_i^a(x_i)} \prod_j^M \int \mathrm{e}^{\sum_d c_j^d\beta_j^d(\tilde{h}_j)+\sum_{iad}\alpha_i^a(x_i)w_{ij}^{ad}\beta_j^d(\tilde{h}_j)}\mathrm{d}\tilde{h}_j\,, \tag{31}
\end{align}$$

where × only highlights the multiplication.
It is straightforward to derive the marginal probability distribution for the hidden units by:

$$P^{RBM}(\mathbf{h}) \overset{(19),(20)}{=} \frac{1}{Z^{RBM}} \int \mathrm{e}^{E^{RBM}(\tilde{\mathbf{x}},\mathbf{h})} \mathrm{d}\tilde{\mathbf{x}} \,, \tag{32}$$

$$\overset{(25)}{=} \frac{1}{Z^{RBM}} \mathrm{e}^{\sum_{jd} c_j^d \beta_j^d(h_j)} \prod_i^N \int \mathrm{e}^{\sum_a b_i^a \alpha_i^a(\tilde{x}_i) + \sum_{jad} \alpha_i^a(\tilde{x}_i) w_{ij}^{ad} \beta_j^d(h_j)} \mathrm{d}\tilde{x}_i \,. \tag{33}$$

Using the Bayes theorem [5] it is then possible to formulate the conditional probability of the hidden units given the visible units by:

$$P^{RBM}(\mathbf{h}|\mathbf{x}) = \frac{P^{RBM}(\mathbf{x},\mathbf{h})}{P^{RBM}(\mathbf{x})} \,, \tag{34}$$

$$\overset{(19),(31)}{\underset{(25))}{=}} \frac{\frac{1}{Z^{RBM}} \mathrm{e}^{\sum_{ia} b_i^a \alpha_i^a(x_i) + \sum_d c_j^d \beta_j^d(h_j) + \sum_{iad} \alpha_i^a(x_i) w_{ij}^{ad} \beta_j^d(h_j)}}{\frac{1}{Z^{RBM}} \mathrm{e}^{\sum_{ia} b_i^a \alpha_i^a(x_i)} \prod_j^M \int \mathrm{e}^{\sum_d c_j^d \beta_j^d(\tilde{h}_j) + \sum_{iad} \alpha_i^a(x_i) w_{ij}^{ad} \beta_j^d(\tilde{h}_j)} \, \mathrm{d}\tilde{h}_j} \,, \tag{35}$$

$$= \frac{\mathrm{e}^{\sum_{ia} b_i^a \alpha_i^a(x_i)} \prod_j^M \mathrm{e}^{\sum_d c_j^d \beta_j^d(h_j) + \sum_{iad} \alpha_i^a(x_i) w_{ij}^{ad} \beta_j^d(h_j)}}{\mathrm{e}^{\sum_{ia} b_i^a \alpha_i^a(x_i)} \prod_j^M \int \mathrm{e}^{\sum_d c_j^d \beta_j^d(\tilde{h}_j) + \sum_{iad} \alpha_i^a(x_i) w_{ij}^{ad} \beta_j^d(\tilde{h}_j)} \, \mathrm{d}\tilde{h}_j} \,, \tag{36}$$

$$= \prod_j^M \frac{\mathrm{e}^{\sum_d c_j^d \beta_j^d(h_j) + \sum_{iad} \alpha_i^a(x_i) w_{ij}^{ad} \beta_j^d(h_j)}}{\int \mathrm{e}^{\sum_d c_j^d \beta_j^d(\tilde{h}_j) + \sum_{iad} \alpha_i^a(x_i) w_{ij}^{ad} \beta_j^d(\tilde{h}_j)} \, \mathrm{d}\tilde{h}_j} \,, \tag{37}$$

and straightforward the conditional probability of the visible units given the hidden units by:

$$P^{RBM}(\mathbf{x}|\mathbf{h}) = \frac{P^{RBM}(\mathbf{x},\mathbf{h})}{P^{RBM}(\mathbf{h})} \,, \tag{38}$$

$$\overset{(19),(33),(25)}{=} \prod_i^N \frac{\mathrm{e}^{\sum_a b_i^a \alpha_i^a(x_i) + \sum_{jad} \alpha_i^a(x_i) w_{ij}^{ad} \beta_j^d(h_j)}}{\int \mathrm{e}^{\sum_a b_i^a \alpha_i^a(\tilde{x}_i) + \sum_{jad} \alpha_i^a(\tilde{x}_i) w_{ij}^{ad} \beta_j^d(h_j)} \, \mathrm{d}\tilde{x}_i} \,. \tag{39}$$

Therefore, the conditional probabilities in RBMs are tractable as long as the one dimensional integrals in (37) and (39) are tractable. The factorization property is consequently very important and will be used frequently in the following chapters.

3.1.4 Maximum Likelihood Estimation

It remains the question how MRFs and therefore BMs and RBMs can be trained. A well studied and widely used technique concerning optimization of parametrized probabilistic models, is the method of Maximum-Likelihood Estimation (MLE) [5]. In MLE we consider a set of observed variables $\mathbf{X}_D = (\mathbf{x}_1, \ldots, \mathbf{x}_D)$, which are assumed to be i.i.d., which means that $\mathbf{X}_D$ is a representative, equally distributed set of independent chosen samples from the unknown underlying distribution $F(\mathbf{x})$. Furthermore, we have a parametrized model $P\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right)$ that defines a probability distribution over variable $\mathbf{x}$. As the name is suggesting, in MLE we want the probability of the data under the model to be maximal. We therefore want to find the optimal parametrization θ_{opt}, which maximizes the likelihood $P\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right)$.

Since we claimed the data $\mathbf{X}_D$ being i.i.d., the probability distribution simplifies to the product of the probabilities for each data point. Moreover it is common to use the logarithm of the likelihood, which has the advantage that products turn into a sum of logarithms. This is valid because the logarithm is a monotonically increasing function and therefore maximizing the likelihood is equivalent to maximizing the Log-Likelihood (LL). The LL is defined as:

$$\begin{aligned} \mathcal{L}\left(\mathbf{X}_D \,|\, \boldsymbol{\theta}\right) &= \ln P\left(\mathbf{x}_1, \ldots, \mathbf{x}_D \,|\, \boldsymbol{\theta}\right) \,, && (40) \\ &= \ln \prod_i^D P\left(\mathbf{x}_i \,|\, \boldsymbol{\theta}\right), && (41) \\ &= \sum_i^D \ln P\left(\mathbf{x}_i \,|\, \boldsymbol{\theta}\right). && (42) \end{aligned}$$

From a different perspective we want the model distribution $P\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right)$ as close as possible to the true distribution $F(\mathbf{x})$. The Kullback-Leibler-Divergence is a non symmetric measure for the difference of two probability density functions (PDF) P and Q defined by:

$$\mathrm{KL}\left(P \,||\, Q\right) = \int_{\infty}^{-\infty} P(\mathbf{x}) \ln \frac{P(\mathbf{x})}{Q(\mathbf{x})} \, \mathrm{d}\mathbf{x} \,. \qquad (43)$$

So that the Kullback-Leibler-Divergence between the true data distribution and the parametrized model distribution becomes:

$$\begin{aligned} \mathrm{KL}\left(F(\mathbf{x}) \,||\, P\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right)\right) &= \int_{\infty}^{-\infty} F(\mathbf{x}) \ln \frac{F(\mathbf{x})}{P\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right)} \,\mathrm{d}\mathbf{x}\,, & (44) \\ &= \int_{\infty}^{-\infty} F(\mathbf{x}) \ln F(\mathbf{x}) \,\mathrm{d}\mathbf{x} - \int_{\infty}^{-\infty} F(\mathbf{x}) \,\ln P\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right) \,\mathrm{d}\mathbf{x}. & (45) \\ &= \mathbb{E}_{F(x)} \left[\, \ln F(\mathbf{x}) \,\right] - \mathbb{E}_{F(x)} \left[\, \ln P\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right) \right]\,, & (46) \end{aligned}$$

where $\mathbb{E}_{F(x)}\left[\,\cdot\,\right]$ denotes the expectation value over $F(x)$. The first term is constant since $F(x)$ is constant and therefore the Kullback-Leibler Divergence is minimal when the second term is maximal. The second term is just the expected LL, which turns into $\frac{1}{D}\sum_{d=1}^{D} \ln P\left(\mathbf{x}_d \,|\, \boldsymbol{\theta}\right)$ for finite data and is equivalent to the average LL: $\frac{1}{D}\mathcal{L}\left(\mathbf{X}_D \,|\, \boldsymbol{\theta}\right) = \frac{1}{D}\sum_{d=1}^{D} \mathcal{L}\left(\mathbf{x}_d \,|\, \boldsymbol{\theta}\right)$. So that minimizing the Kullback-Leibler-Divergence is equivalent to maximizing the LL.

3.1.5 Maximum Likelihood Estimation in Markov Random Fields

For MRFs the parametrization of the LL function (42) is defined in the models energy (19). The LL of MRFs for a value $\mathbf{x}$ and a given parametrization of the energy $\boldsymbol{\theta}$ is then given by:

$$\begin{aligned} \mathcal{L}^{MRF}\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right) &\overset{(42)}{=} \ln P^{MRF}\left(\mathbf{x} \,|\, \boldsymbol{\theta}\right)\,, & (47) \\ &= \ln \int P^{MRF}\left(\mathbf{x}, \mathbf{h} \,|\, \boldsymbol{\theta}\right) \,\mathrm{d}\mathbf{h}\,, & (48) \\ &\overset{(19)}{=} \ln \int \frac{1}{Z^{MRF}} \,\mathrm{e}^{-\frac{1}{T}\,E(\mathbf{x},\mathbf{h})} \,\mathrm{d}\mathbf{h}\,, & (49) \\ &= \ln \int \mathrm{e}^{-\frac{1}{T}\,E(\mathbf{x},\mathbf{h})} \,\mathrm{d}\mathbf{h} - \ln Z^{BM}\,, & (50) \\ &\overset{(20)}{=} \ln \int \mathrm{e}^{-\frac{1}{T}\,E(\mathbf{x},\mathbf{h})} \,\mathrm{d}\mathbf{h} - \ln \int\int \mathrm{e}^{-\frac{1}{T}\,E\left(\tilde{\mathbf{x}},\tilde{\mathbf{h}}\right)} \,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}\,. & (51) \end{aligned}$$

Maximizing the LL-function is usually done using gradient based optimization methods. The gradient of the LL (51) with respect to the parameters $\boldsymbol{\theta}$ is given by:

$$\begin{align}
\frac{\partial \mathcal{L}^{MRF}(\mathbf{x}\,|\,\boldsymbol{\theta})}{\partial\theta} &\overset{(51)}{=} \frac{\partial}{\partial\theta}\left(\ln\int \mathrm{e}^{-\frac{1}{T}E(\mathbf{x},\mathbf{h})}\,\mathrm{d}\mathbf{h}\right) - \frac{\partial}{\partial\theta}\left(\ln\int\int \mathrm{e}^{-\frac{1}{T}E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}\,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}\right), \tag{52}\\
&= \frac{1}{\int \mathrm{e}^{-\frac{1}{T}E(\mathbf{x},\mathbf{h})}\,\mathrm{d}\mathbf{h}}\int \frac{\partial}{\partial\theta}\mathrm{e}^{-\frac{1}{T}E(\mathbf{x},\mathbf{h})}\,\mathrm{d}\mathbf{h} \nonumber\\
&\quad -\frac{1}{\int\int \mathrm{e}^{-\frac{1}{T}E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}\,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}}\int\int \frac{\partial}{\partial\theta}\mathrm{e}^{-\frac{1}{T}E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}\,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{53}\\
&= -\frac{1}{\int \mathrm{e}^{-\frac{1}{T}E(\mathbf{x},\mathbf{h})}\,\mathrm{d}\mathbf{h}}\int \mathrm{e}^{-\frac{1}{T}E(\mathbf{x},\mathbf{h})}\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\,\mathrm{d}\mathbf{h} \nonumber\\
&\quad +\frac{1}{\int\int \mathrm{e}^{-\frac{1}{T}E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}\,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}}\int\int \mathrm{e}^{-\frac{1}{T}E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}\frac{\frac{1}{T}\,\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}\,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{54}\\
&\overset{(19,20)}{=} -\frac{1}{Z^{MRF}\int P^{MRF}(\mathbf{x},\mathbf{h})\,\mathrm{d}\mathbf{h}}\int Z^{MRF}P^{MRF}(\mathbf{x},\mathbf{h})\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\,\mathrm{d}\mathbf{h} \nonumber\\
&\quad +\frac{1}{Z^{MRF}}\int\int Z^{MRF}P^{MRF}(\tilde{\mathbf{x}},\tilde{\mathbf{h}})\,\frac{\frac{1}{T}\,\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}\,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{55}\\
&= -\int \frac{P^{MRF}(\mathbf{x},\mathbf{h})}{P^{MRF}(\mathbf{x})}\,\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\,\mathrm{d}\mathbf{h} \nonumber\\
&\quad +\int\int P^{MRF}(\tilde{\mathbf{x}},\tilde{\mathbf{h}})\,\frac{\frac{1}{T}\,\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}\,\mathrm{d}\tilde{\mathbf{x}}\,\mathrm{d}\tilde{\mathbf{h}}\,, \tag{56}\\
&= -\int P^{MRF}(\mathbf{h}\,|\,\mathbf{x})\,\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\,\mathrm{d}\mathbf{h} \nonumber\\
&\quad +\int P^{MRF}(\tilde{\mathbf{x}})\int P^{MRF}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,\frac{\frac{1}{T}\,\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}\,\mathrm{d}\tilde{\mathbf{h}}\,\mathrm{d}\tilde{\mathbf{x}}\,, \tag{57}\\
&= -\mathbb{E}_{P^{MRF}(\mathbf{h}\,|\,\mathbf{x})}\left[\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right] + \mathbb{E}_{P^{MRF}(\tilde{\mathbf{h}}\,,\,\tilde{\mathbf{x}})}\left[\frac{\frac{1}{T}\,\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}\right]. \tag{58}
\end{align}$$

Where (58) highlights that the first term in (57) calculates the expectation of the hidden probabilities given the data under the current model and the second term in (57) calculates the expectation of the joint probability of visible and hidden units under the current model.

Some special MRFs allow exact inference, like MRFs, for example, where the energy defines a tree structure. In most cases nevertheless, the exact calculation of the gradient is intractable since it still involves an integration over all possible states.

If we are able to approximate these expectations efficiently and reliably, which is especially the case for RBMs, inference becomes possible.

3.2 Binary-Binary Restricted Boltzmann Machines

A Binary-Binary Restricted Boltzmann Machine (BB-RBM) is the original variant of RBMs, which was first proposed in [17]. It has binary units on the visible and the hidden layer. Therefore, the input data $\mathbf{x}$ needs to have a binary representation and the hidden representation $\mathbf{h}$ will also be binary, i.e. $x_i \in \{0,1\}$, $h_j \in \{0,1\}$.

3.2.1 Energy Function

In the original form of a BB-RBM each visible unit x_i has a bias scalar b_i and each hidden unit h_j a bias scalar c_j, respectively. The visible and hidden units are connected via a weight scalar w_{ij} and the transfer functions and the parametrization of the general RBM (25) become simply the identity:

$$\alpha_i^1(x_i) = x_i\,, \tag{59}$$

$$\beta_j^1(h_j) = h_j\,, \tag{60}$$

$$b_i^1 = b_i\,, \tag{61}$$

$$c_j^1 = b_j\,, \tag{62}$$

$$w_{ij}^{11} = w_{ij}\,, \tag{63}$$

where A and D in (25) was set to one. The corresponding energy function for a BB-RBM is then given by:

$$E^{BB}(\mathbf{x},\mathbf{h}) \overset{(25)}{=} -\sum_i^N b_i^1 \alpha_i^1(x_i) - \sum_j^M c_j^1\, \beta_j^1(h_j) - \sum_{ij}^{N,M} \alpha_i^1(x_i)\, w_{ij}^{11}\, \beta_j^1(h_j)\,, \tag{64}$$

$$\underset{(62),(63)}{\overset{(59),(60),(61)}{=}} -\sum_i^N b_i\, x_i - \sum_j^M c_j\, h_j - \sum_{ij}^{N,M} x_i\, w_{ij}\, h_j\,, \tag{65}$$

$$= -\mathbf{x}^T\mathbf{b} - \mathbf{c}^T\mathbf{h} - \mathbf{x}^T\mathbf{W}\mathbf{h}\,, \tag{66}$$

where the second equation is given in clearer matrix vector notation.

3.2.2 Joint Probability Density Function

Substituting the BB-RBM energy (66) into the general joint probability of MRFs (19) we obtain the corresponding PDF for a BB-RBM as:

$$
\begin{aligned}
P^{BB}\left(\mathbf{x},\mathbf{h}\right) &\overset{(19),(66)}{=} \frac{1}{Z^{BB}}\,\mathrm{e}^{\mathbf{x}^T\mathbf{b}+\mathbf{c}^T\mathbf{h}+\mathbf{x}^T\mathbf{W}\mathbf{h}}\,, && (67)\\
&\overset{(19),(65)}{=} \frac{1}{Z^{BB}}\,\mathrm{e}^{\sum_i^N b_i\,x_i+\sum_j^M c_j\,h_j+\sum_{ij}^{N,M} x_i\,w_{ij}\,h_j}\,, && (68)\\
&= \frac{1}{Z^{BB}}\prod_{ij}^{N,M}\mathrm{e}^{b_i\,x_i+c_j\,h_j+x_i\,w_{ij}\,h_j}\,, && (69)\\
&= \frac{1}{Z^{BB}}\prod_{ij}^{N,M}\phi_{ij}^{BB}\left(x_i,h_j\right), && (70)
\end{aligned}
$$

and the partition function becomes,

$$
\begin{aligned}
Z^{BB} &\overset{(20),(66)}{=} \sum_{\tilde{x}}^{\tilde{X}}\sum_{\tilde{h}}^{\tilde{H}}\mathrm{e}^{\tilde{\mathbf{x}}^T\mathbf{b}+\mathbf{c}^T\tilde{\mathbf{h}}+\tilde{\mathbf{x}}^T\mathbf{W}\tilde{\mathbf{h}}}\,, && (71)\\
&= \sum_{\tilde{x}}^{\tilde{X}}\sum_{\tilde{h}}^{\tilde{H}}\prod_{ij}^{N,M}\phi_{ij}^{BB}\left(\tilde{x}_i,\tilde{h}_j\right), && (72)
\end{aligned}
$$

where the temperature is assumed being one if not stated otherwise for reasons of readability. Equations (70) and (72) show that we are indeed having a PoE model (14) where ϕ_{ij}^{BB} represents the corresponding experts (16).

3.2.3 Marginal Probability Density Functions

Since we usually do not know the corresponding hidden representation, one would like to know the probability of a given input sample independently. Due to general factorization property (31), we could easily marginalize over all hidden states $\tilde{H}$. So that the marginal probability distribution of $\mathbf{x}$ is given by:

$$\begin{aligned}
P^{BB}(\mathbf{x}) &= \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{BB}(\mathbf{x}, \tilde{\mathbf{h}}), && (73)\\
&\overset{(67)}{=} \frac{1}{Z^{BB}} \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} \mathrm{e}^{\mathbf{x}^T\mathbf{b}+\mathbf{c}^T\tilde{\mathbf{h}}+\mathbf{x}^T\mathbf{W}\tilde{\mathbf{h}}}, && (74)\\
&= \frac{1}{Z^{BB}} \mathrm{e}^{\mathbf{x}^T\mathbf{b}} \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} \mathrm{e}^{\sum_j^M \left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)\tilde{h}_j}, && (75)\\
&= \frac{1}{Z^{BB}} \mathrm{e}^{\mathbf{x}^T\mathbf{b}} \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} \prod_j^M \mathrm{e}^{\left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)\tilde{h}_j}, && (76)\\
&= \frac{1}{Z^{BB}} \mathrm{e}^{\mathbf{x}^T\mathbf{b}} \left(\sum_{\tilde{h}_1} \mathrm{e}^{\left(c_1+\mathbf{x}^T\mathbf{w}_{*1}\right)\tilde{h}_1} \times \sum_{\tilde{h}_2} \mathrm{e}^{\left(c_2+\mathbf{x}^T\mathbf{w}_{*2}\right)\tilde{h}_2} \times \right.\\
&\qquad \left. \ldots \times \sum_{\tilde{h}_M} \mathrm{e}^{\left(c_M+\mathbf{x}^T\mathbf{w}_{*M}\right)\tilde{h}_M}\right), && (77)\\
&= \frac{1}{Z^{BB}} \mathrm{e}^{\mathbf{x}^T\mathbf{b}} \prod_j^M \sum_{\tilde{h}_j} \mathrm{e}^{\left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)\tilde{h}_j}, && (78)\\
&= \frac{1}{Z^{BB}} \mathrm{e}^{\mathbf{x}^T\mathbf{b}} \prod_j^M \left(1+\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}\right), && (79)\\
&= \frac{1}{Z^{BB}} \prod_j^M \tilde{\phi}_j^{BB}(\mathbf{x}), && (80)
\end{aligned}$$

where $\tilde{\phi}_i^{BB}$ are the individual statistical independent experts for the visible units and $\mathbf{w}_{*j}$ is a column vector containing the values of the jth column of the weight matrix. Due to the symmetry between visible and hidden layer of RBMs, it is straightforward to derive the marginal probability distribution of the hidden variables by:

$$P^{BB}\left(\mathbf{h}\right) \overset{(67)}{=} \frac{1}{Z^{BB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}}\prod_{i}^{N}\left(1+\mathrm{e}^{b_i+\mathbf{w}_{i*}^T\mathbf{h}}\right), \tag{81}$$

$$= \frac{1}{Z^{BB}}\prod_{i}^{N}\bar{\phi}_i^{BB}\left(\mathbf{h}\right), \tag{82}$$

where $\mathbf{w}_{i*}^T$ is a column vector, containing the values of the ith row of the weight matrix and $\bar{\phi}_i^{BB}$ are the individual statistical independent experts for the hidden units.

Furthermore, the RBMs factorization property allows also to simplify the calculation of the partition function in exactly the same way. We are able to marginalize out the visible or hidden units and the partition function becomes:

$$Z^{BB} \overset{(71)}{=} \sum_{\tilde{x}}^{\tilde{X}}\mathrm{e}^{\tilde{\mathbf{x}}^T\mathbf{b}}\prod_{j}^{M}\left(1+\mathrm{e}^{c_j+\tilde{\mathbf{x}}^T\mathbf{w}_{*j}}\right), \tag{83}$$

$$= \sum_{\tilde{h}}^{\tilde{H}}\mathrm{e}^{\mathbf{c}^T\tilde{\mathbf{h}}}\prod_{i}^{N}\left(1+\mathrm{e}^{b_i+\mathbf{w}_{i*}^T\tilde{\mathbf{h}}}\right). \tag{84}$$

This allows to calculate the partition function if the number of visible units or the number of hidden units is small enough, i.e. up to 25 units on normal computers, which allows a calculation without parallelization within minutes.

3.2.4 Conditional Probability Density Functions

Using the Bayes theorem it is now possible to formulate the conditional probability of the hidden units given the visibles:

$$
\begin{aligned}
P^{BB}(\mathbf{h}|\mathbf{x}) \quad &= \quad \frac{P^{BB}(\mathbf{x},\mathbf{h})}{P^{BB}(\mathbf{x})}\,, \qquad\qquad (85)\\
&\overset{(67),(79)}{=} \quad \frac{\frac{1}{Z^{BB}}\,\mathrm{e}^{\mathbf{x}^T\mathbf{b}+\mathbf{c}^T\mathbf{h}+\mathbf{x}^T\mathbf{W}\mathbf{h}}}{\frac{1}{Z^{BB}}\,\mathrm{e}^{\mathbf{x}^T\mathbf{b}}\prod_j^M\left(1+\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}\right)}\,, \qquad\qquad (86)\\
&= \quad \frac{\mathrm{e}^{\mathbf{x}^T\mathbf{b}}\,\mathrm{e}^{\sum_j^M\left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)h_j}}{\mathrm{e}^{\mathbf{x}^T\mathbf{b}}\prod_j^M\left(1+\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}\right)}\,, \qquad\qquad (87)\\
&= \quad \prod_j^M \frac{\mathrm{e}^{\left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)h_j}}{1+\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}}\,, \qquad\qquad (88)\\
&= \quad \prod_j^M P^{BB}(h_j\,|\,\mathbf{x})\,. \qquad\qquad (89)
\end{aligned}
$$

Further on, we get the probability of a particular hidden unit $h_j \in \{0,1\}$ being active given a visible state $\mathbf{x}$ by:

$$
\begin{aligned}
P^{BB}\left(h_j=1\,|\,\mathbf{x}\right) \quad &\overset{(88)}{=} \quad \frac{\mathrm{e}^{\left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)h_j}}{1+\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}}\,, \qquad\qquad (90)\\
&\overset{(\mathit{since}\,h_j=1)}{=} \quad \frac{\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}}{1+\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}}\,, \qquad\qquad (91)\\
&= \quad \frac{\frac{1}{\mathrm{e}^{\left(-c_j-\mathbf{x}^T\mathbf{w}_{*j}\right)}}}{1+\frac{1}{\mathrm{e}^{\left(-c_j-\mathbf{x}^T\mathbf{w}_{*j}\right)}}}\,, \qquad\qquad (92)\\
&= \quad \frac{1}{1+\mathrm{e}^{\left(-\left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)\right)}}\,. \qquad\qquad (93)
\end{aligned}
$$

And straightforward the probability of $h_j \in \{0,1\}$ being inactive is given by:

$$
\begin{aligned}
P^{BB}\left(h_j=0\,|\,\mathbf{x}\right) \quad &\overset{(88)}{=} \quad \frac{\mathrm{e}^{\left(c_j+\mathbf{x}^T\mathbf{w}_{*j}\right)h_j}}{1+\mathrm{e}^{c_j+\mathbf{x}^T\mathbf{w}_{*j}}}\,, \qquad\qquad (94)\\
&\overset{(\mathit{since}\,h_j=0)}{=} \quad \frac{1}{1+\mathrm{e}^{-\left(-c_j-\mathbf{x}^T\mathbf{w}_{*j}\right)}}\,. \qquad\qquad (95)
\end{aligned}
$$

Astonishingly the natural outcome is the Sigmoid function $\frac{1}{1+\mathrm{e}^{-x}}$, which is frequently used in artificial neural networks as non-linear activation function.

Less surprisingly, due to the symmetry of RBMs we get the conditional probability of the visible units given the hiddens in a straightforward manner by:

$$P^{BB}(\mathbf{x}|\mathbf{h}) \overset{(69),(79)}{=} \prod_i^N \frac{\mathrm{e}^{\left(b_i+\mathbf{w}_{i*}^T\mathbf{h}\right)x_i}}{1+\mathrm{e}^{b_i+\mathbf{w}_{i*}^T\mathbf{h}}}\,, \tag{96}$$

$$= \prod_i^N P^{BB}(x_i\,|\,\mathbf{h})\,, \tag{97}$$

with a particular visible unit being active and inactive, respectively by:

$$P^{BB}(x=1|\mathbf{h}) \overset{(96)}{=} \frac{1}{1+\mathrm{e}^{b_i+\mathbf{w}_{i*}^T\mathbf{h}}}\,. \tag{98}$$

$$P^{BB}(x=0|\mathbf{h}) \overset{(96)}{=} \frac{1}{1+\mathrm{e}^{-b_i-\mathbf{w}_{i*}^T\mathbf{h}}}\,. \tag{99}$$

3.2.5 Log Likelihood Gradients

If we want to train a BB-RBM by maximizing the LL (51), which is the usual way, we need to be able to calculate the LL-Gradient (57). Since we have already derived the necessary probabilities $P^{BB}(\mathbf{x}|\mathbf{h})$ and $P^{BB}(\mathbf{x})$ it remains the derivative of the particular BB-RBM energy function. The partial derivative with respect to the parameters $\theta = \{\,\mathbf{W},\,\mathbf{b},\,\mathbf{c}\,\}$ is given by:

$$\frac{\partial E^{BB}\left(\mathbf{x},\mathbf{h}\right)}{\partial\mathbf{W}} = -\mathbf{x}\mathbf{h}^{\mathbf{T}}\,. \tag{100}$$

$$\frac{\partial E^{BB}\left(\mathbf{x},\mathbf{h}\right)}{\partial\mathbf{b}} = -\mathbf{x}\,. \tag{101}$$

$$\frac{\partial E^{BB}\left(\mathbf{x},\mathbf{h}\right)}{\partial\mathbf{c}} = -\mathbf{h}\,. \tag{102}$$

We get the partial derivative of the BB-RBM LL-Gradient (57) with respect to a weight parameter $w_{i,j}$ by:

$$\frac{\partial\,\mathcal{L}^{BB}\left(\,\mathbf{x}\,|\,\theta\,\right)}{\partial w_{ij}} \overset{(57),(100)}{=} \sum_{\mathbf{h}}^{\mathbf{H}} P^{BB}(\mathbf{h}\,|\,\mathbf{x})\,x_i h_j - \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} P^{BB}(\tilde{\mathbf{x}}) \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{BB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,\tilde{x}_i\tilde{h}_j\,, \tag{103}$$

$$\overset{(113)}{=} P^{BB}(h_j=1\,|\,\mathbf{x})\,x_i - \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} P^{BB}(\tilde{\mathbf{x}})\,P^{BB}(\tilde{h}_j=1\,|\,\tilde{\mathbf{x}})\,\tilde{x}_i\,, \tag{104}$$

$$= P^{BB}(h_j=1\,|\,\mathbf{x})\,x_i - \mathbb{E}_{P^{BB}(\tilde{\mathbf{x}})}\left[P^{BB}(\tilde{h}_j=1\,|\,\tilde{\mathbf{x}})\,\tilde{x}_i\right]\,, \tag{105}$$

where we used the general factorization property of the marginal probability distribution as follows:

$$\begin{aligned}
\sum_{\mathbf{h}}^{\mathbf{H}} P^{BB}(\mathbf{h}\,|\,\mathbf{x})\, x_i h_j &= \sum_{\mathbf{h}}^{\mathbf{H}} \left(\prod_{l=1}^{M} P^{BB}(h_l\,|\,\mathbf{x}) \right) x_i h_j && (106)\\
&= x_i \sum_{\mathbf{h}}^{\mathbf{H}} h_j \prod_{l=1}^{M} P^{BB}(h_l\,|\,\mathbf{x})\,, && (107)\\
&= x_i \sum_{h_1} \ldots \sum_{h_{j-1}} \sum_{h_j} \sum_{h_{j+1}} \ldots \sum_{h_M} h_j \prod_{l=1}^{M} P^{BB}(h_l\,|\,\mathbf{x})\,, && (108)\\
&= x_i \times \sum_{h_1} P^{BB}(h_1\,|\,\mathbf{x}) \times \ldots \times \sum_{h_{j-1}} P^{BB}(h_{j-1}\,|\,\mathbf{x}) \times \\
&\quad \sum_{h_{j=l}} P^{BB}(h_j\,|\,\mathbf{x})\, h_j \times \sum_{h_{j+1}} P^{BB}(h_{j+1}\,|\,\mathbf{x}) \times \ldots && (109)\\
&\quad \ldots \times \sum_{h_M} P^{BB}(h_M\,|\,\mathbf{x})\,, && (110)\\
&= x_i \times 1 \times \ldots \times 1 \times \sum_{h_{j=l}} P^{BB}(h_j\,|\,\mathbf{x})\, h_j \times 1 \times \ldots \times 1\,, && (111)\\
&= x_i \left(P^{BB}(h_j = 0\,|\,\mathbf{x})\, 0 + P^{BB}(h_j = 1\,|\,\mathbf{x})\, 1 \right)\,, && (112)\\
&= P^{BB}(h_j = 1\,|\,\mathbf{x})\, x_i\,. && (113)
\end{aligned}$$

The partial derivative of BB-RBM LL with respect to b_i then becomes,

$$\begin{aligned}
\frac{\partial\, \mathcal{L}^{BB}\left(\,\mathbf{x}\,|\,\theta\,\right)}{\partial b_i} &\overset{(57),(101)}{=} \sum_{\mathbf{h}}^{\mathbf{H}} P^{BB}(\mathbf{h}\,|\,\mathbf{x})\, x_i - \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} P^{BB}(\tilde{\mathbf{x}}) \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{BB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\, \tilde{x}_i\,, && (114)\\
&= x_i \sum_{\mathbf{h}}^{\mathbf{H}} P^{BB}(\mathbf{h}\,|\,\mathbf{x}) - \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} P^{BB}(\tilde{\mathbf{x}}) \tilde{x}_i \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{BB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,, && (115)\\
&= x_i - \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} P^{BB}(\tilde{\mathbf{x}})\, \tilde{x}_i\,, && (116)\\
&= x_i - \mathbb{E}_{P^{BB}(\tilde{\mathbf{x}})}\left[\tilde{x}_i\right]\,, && (117)
\end{aligned}$$

and the partial derivative of BB-RBM LL with respect to c_j,

$$\frac{\partial \mathcal{L}^{BB}\left(\mathbf{x}\,|\,\theta\right)}{\partial c_j} \overset{(57),(102)}{=} \sum_{\mathbf{h}}^{\mathbf{H}} P^{BB}(\mathbf{h}\,|\,\mathbf{x})\,h_j - \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} P^{BB}(\tilde{\mathbf{x}}) \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{BB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,\tilde{h}_j\,, \quad (118)$$

$$\overset{(113)}{=} P^{BB}(h_j = 1\,|\,\mathbf{x}) - \sum_{\tilde{\mathbf{x}}}^{\tilde{\mathbf{X}}} P^{BB}(\tilde{\mathbf{x}})\,P^{BB}(\tilde{h}_j = 1\,|\,\tilde{\mathbf{x}})\,, \quad (119)$$

$$= P^{BB}(h_j = 1\,|\,\mathbf{x}) - \mathbb{E}_{P^{BB}(\tilde{\mathbf{x}})}\left[P^{BB}(\tilde{h}_j = 1\,|\,\tilde{\mathbf{x}})\right]\,, \quad (120)$$

where we used the factorization property for x_i in an equivalent way as shown in (113).

3.2.6 Other Types of Units

In the original definition of BMs [2], the visible and hidden units have binary values. However, in most cases the input data is coming from a continuous rather than a binary domain. Therefore, it would be of most interest to have the opportunity to choose continuous units as well.

An easy way, making the original BM handle continuous data is simply to rescale the data into the interval $[0, 1]$ and considering it as the probability for the corresponding unit taking the value one. However, the model is still assuming an underlying binary representation, so that this variant usually works not very well.

If we assume the data coming truly from the interval $[0, \infty)$ the conditional probabilities (97) become exponential densities. This causes the normalization constant not to exist in each case so that truncated exponentials over the interval $[0, 1]$ are used instead, which leads to the so called Truncated Exponential RBMs [15]

A natural assumption when dealing with continuous variables is assuming them to be Gaussian distributed and therefore, a distribution over $\mathbb{R}$. This leads to the so called Gaussian-Binary RBM, which has been used successfully to model continuous domains and will be discussed in the next chapter.

So far we considered only the visible layer to have continuous values but one can also think of RBMs with continuous visible and hidden layer like a Gaussian-Gaussian RBM for example. But as we will see, training an RBM with continuous visible and binary hidden layer tends to be difficult already. Furthermore this training issue becomes crucial when having only continuous units since they get much more effected to sampling noise. This makes them uninteresting in practice although a completely continuous network seems to be the more powerful configuration.

3.3 Gaussian-Binary Restricted Boltzmann Machines

The original formulation of RBMs assumes the input data having a binary representation, but in many cases the input data is coming from a continuous domain. A popular variant of the BB-RBM is a so called Gaussian-Binary RBM (GB-RBM) [16], which assumes the input values $x_i \in [-\infty, +\infty]$ being normally distributed with mean b_i and variance σ_i^2. The hidden units are still binary distributed, $h_j \in \{0, 1\}$ so that a GB-RBM transfers the continuous input data to a binary representation.

3.3.1 Energy Function

The energy of the GB-RBM can be derived from the general RBM (25) by setting $A = 3$ and $D = 1$ with the corresponding transfer functions:

$$\begin{aligned}
\alpha_i^1(x_i) &= -x_i^2\,, && (121)\\
\alpha_i^2(x_i) &= x_i\,, && (122)\\
\alpha_i^3(x_i) &= 1\,, && (123)\\
\beta_j^1(h_j) &= h_j\,, && (124)\\
& && (125)
\end{aligned}$$

and the corresponding parameters are chosen as follows:

$$\begin{aligned}
b_i^1 &= \frac{1}{2\,\sigma_i^2}\,, && (126)\\
b_i^2 &= \frac{b_i}{\sigma_i^2}\,, && (127)\\
b_i^3 &= -\frac{b_i^2}{2\,\sigma_i^2}\,, && (128)\\
c_j^1 &= c_j\,, && (129)\\
w_{ij}^{11} &= 0\,, && (130)\\
w_{ij}^{21} &= \frac{w_{ij}}{\sigma_i^2}\,, && (131)\\
w_{ij}^{31} &= 0\,. && (132)
\end{aligned}$$

The corresponding energy function for a GB-RBM is then given by:

$$
\begin{aligned}
E^{GB}\left(\mathbf{x},\mathbf{h}\right) \quad &\overset{(25)}{=} \quad -\sum_{i}^{N} b_i^1 \alpha_i^1(x_i) - \sum_{i}^{N} b_i^2 \alpha_i^2(x_i) - \sum_{i}^{N} b_i^3 \alpha_i^3(x_i) \\
&\quad -\sum_{j}^{M} c_j^1\, \beta_j^1(h_j) - \sum_{ij}^{N,M} \alpha_i^1(x_i)\, w_{ij}^{11}\, \beta_j^1(h_j) \\
&\quad -\sum_{ij}^{N,M} \alpha_i^2(x_i)\, w_{ij}^{21}\, \beta_j^1(h_j) - \sum_{ij}^{N,M} \alpha_i^3(x_i)\, w_{ij}^{31}\, \beta_j^1(h_j)\,, \qquad (133) \\
&\overset{(121),\ldots,(132)}{=} \sum_{i}^{N} \frac{x_i^2}{2\,\sigma_i^2} - \sum_{i}^{N} \frac{x_i\, b_i}{\sigma_i^2} + \sum_{i}^{N} \frac{b_i^2}{2\,\sigma_i^2} - \sum_{j}^{M} c_j\, h_j \qquad (134) \\
&\quad -\sum_{ij}^{N,M} \frac{x_i\, w_{ij}\, h_j}{\sigma_i^2}\,, \\
&= \quad \sum_{i}^{N} \frac{\left(x_i - b_i\right)^2}{2\sigma_i^2} - \sum_{j}^{M} c_j\, h_j - \sum_{ij}^{N,M} \frac{x_i\, w_{ij}\, h_j}{\sigma_i^2}\,, \qquad (135) \\
&= \quad \left\| \frac{\mathbf{x} - \mathbf{b}}{2\boldsymbol{\sigma}} \right\|^2 - \mathbf{c}^T\mathbf{h} - \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{W}\mathbf{h}\,. \qquad (136)
\end{aligned}
$$

where the second equation is given in clearer matrix vector notation and the fraction bar denotes the component wise division.

Notice that there exists a slightly different formulation of the GB-RBM energy [24], where the quadratic term (123) uses σ_i instead of σ_i^2. But as stated in [6], this leads to a counter intuitive scaling of the conditional mean by σ_i^2, so that in this work a GB-RBM is always considered to be defined as (136).

3.3.2 Joint Probability Density Function

Equivalent as for the BB-RBM, we substitute the GB-RBM energy (136) into the general joint probability (19) and obtain the corresponding joint PDF as:

$$\begin{aligned}
P^{GB}\left(\mathbf{x},\mathbf{h}\right) &\overset{(19),(136)}{=} \frac{1}{Z^{GB}}\, \mathrm{e}^{-\left\|\frac{\mathbf{x}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^2+\mathbf{c}^T\mathbf{h}+\left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T\mathbf{W}\mathbf{h}}\,, && (137)\\
&\overset{(19),(135)}{=} \frac{1}{Z^{GB}}\, \mathrm{e}^{-\sum_i^N \frac{(x_i-b_i)^2}{2\sigma_i^2}+\sum_j^M c_j\,h_j+\sum_{ij}^{N,M}\frac{x_i\,w_{ij}\,h_j}{\sigma_i^2}}\,, && (138)\\
&= \frac{1}{Z^{GB}} \prod_{ij}^{N,M} \mathrm{e}^{-\frac{(x_i-b_i)^2}{2\sigma_i^2}+c_j\,h_j+\frac{x_i\,w_{ij}\,h_j}{\sigma_i^2}}\,, && (139)\\
&= \frac{1}{Z^{GB}} \prod_{ij}^{N,M} \phi_{ij}^{GB}(x_i,h_j) && (140)
\end{aligned}$$

with partition function,

$$\begin{aligned}
Z^{GB} &\overset{(20),(136)}{=} \int \sum_{\tilde{h}}^{\tilde{H}} \mathrm{e}^{-\left\|\frac{\tilde{\mathbf{x}}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^2+\mathbf{c}^T\tilde{\mathbf{h}}+\left(\frac{\tilde{\mathbf{x}}}{\boldsymbol{\sigma}^2}\right)^T\mathbf{W}\tilde{\mathbf{h}}}\, \mathrm{d}\tilde{\mathbf{x}}\,, && (141)\\
&= \int \sum_{\tilde{h}}^{\tilde{H}} \prod_{ij}^{N,M} \phi_{ij}^{GB}(\tilde{x}_i,\tilde{h}_j)\, \mathrm{d}\tilde{\mathbf{x}}\,, && (142)
\end{aligned}$$

where again, the temperature is assume to be one if not stated otherwise.

3.3.3 Marginal Probability Density Functions

In the same way as shown in (73), we get the probability for $\mathbf{x}$ by marginalization over the hidden values:

$$\begin{aligned}
P^{GB}(\mathbf{x}) &= \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{GB}(\mathbf{x}, \tilde{\mathbf{h}}), && (143)\\
&\overset{(137)}{=} \frac{1}{Z^{GB}} \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} \mathrm{e}^{-\left\|\frac{\mathbf{x}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^2 + \mathbf{c}^T\tilde{\mathbf{h}} + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{W}\tilde{\mathbf{h}}}, && (144)\\
&= \frac{1}{Z^{GB}} \mathrm{e}^{-\left\|\frac{\mathbf{x}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^2} \prod_{j}^{M} \left(1 + \mathrm{e}^{c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}\tilde{h}_j}\right), && (145)\\
&= \frac{1}{Z^{GB}} \prod_{j}^{M} \tilde{\phi}_j^{GB}(\mathbf{x}). && (146)
\end{aligned}$$

For the marginal probability density of **h** we integrate over all possible visible values. By using the general factorization property the marginal probability of **h** becomes:

$$P^{GB}(\mathbf{h}) = \int P^{GB}(\tilde{\mathbf{x}}, \mathbf{h})\,\mathrm{d}\tilde{\mathbf{x}}\,, \tag{147}$$

$$\overset{(137)}{=} \frac{1}{Z^{GB}} \int \mathrm{e}^{-\left\| \frac{\tilde{\mathbf{x}}-\mathbf{b}}{2\boldsymbol{\sigma}} \right\|^2 + \mathbf{c}^T\mathbf{h} + \left(\frac{\tilde{\mathbf{x}}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{W}\mathbf{h}}\,\mathrm{d}\tilde{\mathbf{x}} \tag{148}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \int \prod_i^N \mathrm{e}^{-\frac{(\tilde{x}_i-b_i)^2}{2\sigma_i^2} + \frac{\tilde{x}_i \mathbf{w}_{i*}^T\mathbf{h}}{\sigma_i^2}}\,\mathrm{d}\tilde{\mathbf{x}} \tag{149}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \Bigg(\int \mathrm{e}^{-\frac{(\tilde{x}_1-b_1)^2}{2\sigma_1^2} + \frac{\tilde{x}_1 \mathbf{w}_{1*}^T\mathbf{h}}{\sigma_1^2}}\,\mathrm{d}\tilde{x}_1 \times \int \mathrm{e}^{-\frac{(\tilde{x}_2-b_2)^2}{2\sigma_2^2} + \frac{\tilde{x}_2 \mathbf{w}_{2*}^T\mathbf{h}}{\sigma_2^2}}\,\mathrm{d}\tilde{x}_2 \times \tag{150}$$

$$\ldots \times \int \mathrm{e}^{-\frac{(\tilde{x}_N-b_N)^2}{2\sigma_N^2} + \frac{\tilde{x}_N \mathbf{w}_{N*}^T\mathbf{h}}{\sigma_N^2}}\,\mathrm{d}\tilde{x}_N \Bigg) \tag{151}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_i^N \int \mathrm{e}^{-\frac{(\tilde{x}_i-b_i)^2 - 2\tilde{x}_i \mathbf{w}_{i*}^T\mathbf{h}}{2\sigma_i^2}}\,\mathrm{d}\tilde{x}_i \tag{152}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_i^N \int \mathrm{e}^{-\frac{\tilde{x}_i^2 - 2\tilde{x}_i(b_i + \mathbf{w}_{i*}^T\mathbf{h}) + b_i^2}{2\sigma_i^2}}\,\mathrm{d}\tilde{x}_i \tag{153}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_i^N \int \mathrm{e}^{-\frac{\tilde{x}_i^2 - 2\tilde{x}_i(b_i + \mathbf{w}_{i*}^T\mathbf{h}) + (b_i + \mathbf{w}_{i*}^T\mathbf{h})^2 - (b_i + \mathbf{w}_{i*}^T\mathbf{h})^2 + b_i^2}{2\sigma_i^2}}\,\mathrm{d}\tilde{x}_i \tag{154}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_i^N \int \mathrm{e}^{-\frac{(\tilde{x}_i - (b_i + \mathbf{w}_{i*}^T\mathbf{h}))^2 - b_i^2 - 2b_i\mathbf{w}_{i*}^T\mathbf{h} - (\mathbf{w}_{i*}^T\mathbf{h})^2 + b_i^2}{2\sigma_i^2}}\,\mathrm{d}\tilde{x}_i \tag{155}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_i^N \mathrm{e}^{\frac{2b_i\mathbf{w}_{i*}^T\mathbf{h} + (\mathbf{w}_{i*}^T\mathbf{h})^2}{2\sigma_i^2}} \int \mathrm{e}^{-\frac{(\tilde{x}_i - b_i - \mathbf{w}_{i*}^T\mathbf{h})^2}{2\sigma_i^2}}\,\mathrm{d}\tilde{x}_i \tag{156}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_i^N \sqrt{2\pi\sigma_i^2}\,\mathrm{e}^{\frac{2b_i\mathbf{w}_{i*}^T\mathbf{h} + (\mathbf{w}_{i*}^T\mathbf{h})^2}{2\sigma_i^2}} \tag{157}$$

$$= \frac{1}{Z^{GB}}\,\mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_i^N \sqrt{2\pi\sigma_i^2}\,\mathrm{e}^{||\frac{b_i + \mathbf{w}_{i*}^T\mathbf{h}}{2\sigma_i^2}||^2 - ||\frac{b_i}{2\sigma_i^2}||^2} \tag{158}$$

$$= \frac{1}{Z^{GB}} \prod_i^N \bar{\phi}_i^{GB}\left(\tilde{\mathbf{h}}\right) \tag{159}$$

The calculation of the partition function is therefore simplified by the factorization via **h** or **x** given by:

$$
\begin{aligned}
Z^{GB} &\overset{(141)}{=} \int \mathrm{e}^{-\left\|\frac{\mathbf{x}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^2} \prod_{j}^{M} \left(1 + \mathrm{e}^{c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}}\right) \mathrm{d}\mathbf{x}\,, && (160)\\
&= \sum_{\tilde{h}}^{\tilde{M}} \mathrm{e}^{\mathbf{c}^T\mathbf{h}} \prod_{i}^{N} \sqrt{2\pi\sigma_i^2}\, \mathrm{e}^{||\frac{b_i+\mathbf{w}_{i*}^T\mathbf{h}}{2\sigma_i^2}||^2 - ||\frac{b_i}{2\sigma_i^2}||^2}\,, && (161)
\end{aligned}
$$

where (160) is still computational intractable also for very small RBMs due to the need for numerical integration in high dimensional spaces. But (161) allows to calculate the partition function via factorization over x for small hidden layers.

3.3.4 Conditional Probability Density Functions

Equivalent to the BB-RBM we get the conditional probability of the hidden units given the visibles:

$$
\begin{aligned}
P^{GB}(\mathbf{h}|\mathbf{x}) &= \frac{P^{GB}(\mathbf{x},\mathbf{h})}{P^{GB}(\mathbf{x})}\,, && (162)\\
&\overset{(137),(145)}{=} \frac{\frac{1}{Z^{GB}}\, \mathrm{e}^{-\left\|\frac{\mathbf{x}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^2 + \mathbf{c}^T\mathbf{h} + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{W}\mathbf{h}.}}{\frac{1}{Z^{GB}}\, \mathrm{e}^{-\left\|\frac{\mathbf{x}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^2} \prod_j^M \left(1 + \mathrm{e}^{c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}}\right)}\,, && (163)\\
&= \frac{\mathrm{e}^{\sum_j^M \left(c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}\right) h_j}}{\prod_j^M \left(1 + \mathrm{e}^{c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}}\right)}\,, && (164)\\
&= \prod_{j}^{M} \frac{\mathrm{e}^{\left(c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}\right) h_j}}{1 + \mathrm{e}^{c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}}}\,, && (165)\\
&= \prod_{j}^{M} P^{GB}(h_j\,|\,\mathbf{x})\,. && (166)
\end{aligned}
$$

Further on, the probability of a particular hidden unit h_j being active given a visible state $\mathbf{x}$ is:

$$\begin{aligned} P^{GB}\left(h_j = 1 \,|\, \mathbf{x}\right) &\overset{(88)}{=} \frac{\mathrm{e}^{\left(c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}\right) h_j}}{1 + \mathrm{e}^{c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}}}\,, && (167) \\ &= \frac{1}{1 + \mathrm{e}^{-(c_j + \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j})}}\,, && (168) \end{aligned}$$

and therefore the probability of h_j being inactive is given by:

$$\begin{aligned} P^{GB}\left(h_j = 0 \,|\, \mathbf{x}\right) &= 1 - P^{GB}\left(h_j = 1 \,|\, \mathbf{x}\right)\,, && (169) \\ &= \frac{1}{1 + \mathrm{e}^{-\left(-c_j - \left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\right)^T \mathbf{w}_{*j}\right)}}\,. && (170) \end{aligned}$$

When deriving the conditional probability for the visible units, the particular definition of the energy and the name Gaussian-Binary RBM finally becomes clear. Since it turns out that the probability for the visible units given the hidden units, is the product over N independent Gaussians distributed random variables given by:

$$\begin{aligned}
P^{GB}\left(\mathbf{x}|\mathbf{h}\right) &= \frac{P^{GB}\left(\mathbf{x},\mathbf{h}\right)}{P^{GB}\left(\mathbf{h}\right)}\,, && (171)\\
&\overset{(138),(157)}{=} \frac{\frac{1}{ZGB}\,\mathrm{e}^{-\left\|\frac{\mathbf{x}-\mathbf{b}}{2\boldsymbol{\sigma}}\right\|^{2}+\mathbf{c}^{T}\mathbf{h}+\left(\frac{\mathbf{x}}{\boldsymbol{\sigma}^{2}}\right)^{T}\mathbf{W}\mathbf{h}}}{\frac{1}{ZGB}\,\mathrm{e}^{\mathbf{c}^{T}\mathbf{h}}\prod_{i}^{N}\sqrt{2\pi\sigma_i^2}\,\mathrm{e}^{\frac{2b_i\mathbf{w}_{i*}^{T}\mathbf{h}+(\mathbf{w}_{i*}^{T}\mathbf{h})^{2}}{2\sigma_i^2}}}\,, && (172)\\
&= \frac{\prod_{i}^{N}\mathrm{e}^{-\frac{(x_i-b_i)^2}{2\sigma_i^2}+\frac{x_i\mathbf{w}_{i*}^{T}\mathbf{h}}{\sigma_i^2}}}{\prod_{i}^{N}\sqrt{2\pi\sigma_i^2}\,\mathrm{e}^{\frac{2b_i\mathbf{w}_{i*}^{T}\mathbf{h}+(\mathbf{w}_{i*}^{T}\mathbf{h})^{2}}{2\sigma_i^2}}}\,, && (173)\\
&= \prod_{i}^{N}\frac{1}{\sqrt{2\pi\sigma_i^2}}\frac{\mathrm{e}^{-\frac{x_i^2-2b_ix_i+b_i^2-2x_i\mathbf{w}_{i*}^{T}\mathbf{h}}{2\sigma_i^2}}}{\mathrm{e}^{\frac{2b_i\mathbf{w}_{i*}^{T}\mathbf{h}+(\mathbf{w}_{i*}^{T}\mathbf{h})^{2}}{2\sigma_i^2}}}\,, && (174)\\
&= \prod_{i}^{N}\frac{1}{\sqrt{2\pi\sigma_i^2}}\mathrm{e}^{\frac{-\left(x_i^2-2b_ix_i+b_i^2-2x_i\mathbf{w}_{i*}^{T}\mathbf{h}+2b_i\mathbf{w}_{i*}^{T}\mathbf{h}+(\mathbf{w}_{i*}^{T}\mathbf{h})^{2}\right)}{2\sigma_i^2}}\,, && (175)\\
&= \prod_{i}^{N}\frac{1}{\sqrt{2\pi\sigma_i^2}}\mathrm{e}^{-\frac{\left(x_i-b_i-\mathbf{w}_{i*}^{T}\mathbf{h}\right)^{2}}{2\sigma_i^2}}\,, && (176)\\
&= \prod_{i}^{N}\mathcal{N}\left(x_i|b_i+\mathbf{w}_{i*}^{T}\mathbf{h},\sigma_i^2\right). && (177)
\end{aligned}$$

So that the PDF for a single visible unit is a normal distribution with mean $b_i+\mathbf{w}_{i*}^{T}\mathbf{h}$ and variance σ_i^2.

3.3.5 Log Likelihood Gradients

For the calculation of LL-Gradient (57), we need the derivatives of the GB-RBM energy function with respect to the parameters $\theta = \{\, \mathbf{W},\, \mathbf{b},\, \mathbf{c},\, \boldsymbol{\sigma} \,\}$ given by:

$$\frac{\partial E^{GB}(\mathbf{x}, \mathbf{h})}{\partial \mathbf{W}} = -\frac{\mathbf{x}}{\boldsymbol{\sigma}^2}\mathbf{h}^{\mathbf{T}} . \tag{178}$$

$$\frac{\partial E^{GB}(\mathbf{x}, \mathbf{h})}{\partial \mathbf{b}} = -\frac{\mathbf{x} - \mathbf{b}}{\boldsymbol{\sigma}^2} . \tag{179}$$

$$\frac{\partial E^{GB}(\mathbf{x}, \mathbf{h})}{\partial \mathbf{c}} = -\mathbf{h} . \tag{180}$$

$$\frac{\partial E^{GB}(\mathbf{x}, \mathbf{h})}{\partial \boldsymbol{\sigma}} = -\left\| \frac{\mathbf{x} - \mathbf{b}}{\boldsymbol{\sigma}\sqrt{\boldsymbol{\sigma}}} \right\|^2 + \left(\frac{2\,\mathbf{x}}{\boldsymbol{\sigma}^3} \right)^T \mathbf{W}\mathbf{h} . \tag{181}$$

The derivative of the BB-RBM energy (100) and GB-RBM energy (178) with respect to the weights only differ in the scaling by σ^2. Therefore, the partial derivative of the LL-Gradient (57) with respect to the weight parameters $w_{i,j}$ only differs by the scaling of sigma σ^2, given by:

$$\frac{\partial \mathcal{L}^{GB}(\,\mathbf{x}\,|\,\theta\,)}{\partial w_{ij}} \underset{(178)}{\overset{(57)}{=}} \sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x})\,\frac{x_i h_j}{\sigma_i^2} - \int P^{GB}(\tilde{\mathbf{x}}) \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{GB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,\frac{\tilde{x}_i \tilde{h}_j}{\sigma_i^2}\,\mathrm{d}\tilde{\mathbf{x}} , \tag{182}$$

$$= \frac{\sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x})\,x_i h_j - \int P^{GB}(\tilde{\mathbf{x}}) \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{GB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,\tilde{x}_i \tilde{h}_j\,\mathrm{d}\tilde{\mathbf{x}}}{\sigma_i^2} , \tag{183}$$

$$\overset{(104)}{=} \frac{1}{\sigma_i^2}\Bigg(P^{GB}(h_j = 1|\,\mathbf{x})\,x_i - \int P^{GB}(\tilde{\mathbf{x}})\,P^{GB}(\tilde{h}_j = 1|\,\tilde{\mathbf{x}})\,\tilde{x}_i\,\mathrm{d}\tilde{\mathbf{x}} \Bigg) , \tag{184}$$

$$= \frac{1}{\sigma_i^2}\left(P^{GB}(h_j = 1|\,\mathbf{x})\,x_i - \mathbb{E}_{P^{GB}(\tilde{\mathbf{x}})}\left[P^{GB}(\tilde{h}_j = 1|\,\tilde{\mathbf{x}})\,\tilde{x}_i \right] \right) , \tag{185}$$

The partial derivative of GB-RBM LL with respect to b_i becomes:

$$\begin{aligned}
\frac{\partial \mathcal{L}^{GB}\left(\mathbf{x}\,|\,\theta\right)}{\partial b_i} \overset{(57),(179)}{=} & \sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x})\,\frac{x_i - b_i}{\sigma_i^2} \\
& - \int P^{GB}(\tilde{\mathbf{x}}) \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{GB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,\frac{\tilde{x}_i - b_i}{\sigma_i^2}\,\mathrm{d}\tilde{\mathbf{x}}\,, \qquad (186) \\
= & \frac{x_i - b_i}{\sigma_i^2} \sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x}) \\
& - \int P^{GB}(\tilde{\mathbf{x}}) \frac{\tilde{x}_i - b_i}{\sigma_i^2} \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{GB}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})\,\mathrm{d}\tilde{\mathbf{x}}\,, \qquad (187) \\
= & \frac{1}{\sigma_i^2}\left(x_i - b_i - \int P^{GB}(\tilde{\mathbf{x}})\,(\tilde{x}_i - b_i)\,\mathrm{d}\tilde{\mathbf{x}}\right)\,, \qquad (188) \\
= & \frac{1}{\sigma_i^2}\left(x_i - b_i - \mathbb{E}_{P^{GB}(\tilde{\mathbf{x}})}\left[\tilde{x}_i - b_i\right]\right)\,, \qquad (189)
\end{aligned}$$

where we used that $\sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x}) = 1$.
The derivative of the BB-RBM energy (100) and GB-RBM energy (178) with respect to the hidden bias are equivalent, so that also the partial derivative of the LL-Gradient (57) with respect to the hidden bias parameters c_j are equivalent, given by:

$$\frac{\partial \mathcal{L}^{GB}\left(\mathbf{x}\,|\,\theta\right)}{\partial c_j} \overset{(57),(102)}{=} P^{GB}(h_j = 1\,|\,\mathbf{x}) - \mathbb{E}_{P^{GB}(\tilde{\mathbf{x}})}\,P^{GB}(\tilde{h}_j = 1\,|\,\tilde{\mathbf{x}})\,. \qquad (190)$$

Finally, the partial derivative of GB-RBM LL with respect to σ_i is given by:

$$\frac{\partial \mathcal{L}^{GB}(\mathbf{x} \mid \theta)}{\partial \sigma_i} \overset{(57),(181)}{=} \sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h} \mid \mathbf{x}) \frac{(x_i - b_i)^2 - 2x_i \mathbf{w}_{i*}^T \mathbf{h}}{\sigma_i^3} - \int P^{GB}(\tilde{\mathbf{x}}) \sum_{\tilde{\mathbf{h}}}^{\tilde{\mathbf{H}}} P^{GB}(\tilde{\mathbf{h}} \mid \tilde{\mathbf{x}}) \frac{(\tilde{x}_i - b_i)^2 - 2\tilde{x}_i \mathbf{w}_{i*}^T \tilde{\mathbf{h}}}{\sigma_i^3} \,\mathrm{d}\tilde{\mathbf{x}}\,, \quad (191)$$

$$= \frac{(x_i - b_i)^2}{\sigma_i^3} - \frac{2x_i}{\sigma_i^3} \sum_{j}^{M} P^{GB}(h_j = 1 \mid \mathbf{x})\, w_{ij} - \int P^{GB}(\tilde{\mathbf{x}}) \left(\frac{(\tilde{x}_i - b_i)^2}{\sigma_i^3} - \frac{2\tilde{x}_i}{\sigma_i^3} \sum_{j}^{M} P^{GB}(\tilde{h}_j = 1 \mid \tilde{\mathbf{x}}) w_{ij} \right) \mathrm{d}\tilde{\mathbf{x}}\,, \quad (192)$$

$$= \frac{(x_i - b_i)^2}{\sigma_i^3} - \frac{2x_i}{\sigma_i^3} \sum_{j}^{M} P^{GB}(h_j = 1 \mid \mathbf{x})\, w_{ij} - \mathbb{E}_{P^{GB}(\tilde{\mathbf{x}})} \left[\frac{(\tilde{x}_i - b_i)^2}{\sigma_i^3} - \frac{2\tilde{x}_i}{\sigma_i^3} \sum_{j}^{M} P^{GB}(\tilde{h}_j = 1 \mid \tilde{\mathbf{x}}) w_{ij} \right] . \quad (193)$$

where we used the factorization property again, which is not as obvious in this case as for the weight parameters. Therefore, the detailed derivation is given by:

$$\sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x})\,\frac{(x_i-b_i)^2-2x_i\mathbf{w}_{i*}^T\mathbf{h}}{\sigma_i^3} \tag{194}$$

$$= \frac{(x_i-b_i)^2}{\sigma_i^3}\sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x}) - \frac{2x_i}{\sigma_i^3}\sum_{\mathbf{h}}^{\mathbf{H}} P^{GB}(\mathbf{h}\,|\,\mathbf{x})\,\mathbf{w}_{i*}^T\mathbf{h}\,, \tag{195}$$

$$= \frac{(x_i-b_i)^2}{\sigma_i^3}\sum_{\mathbf{h}}^{\mathbf{H}}\left(\prod_{l=1}^{M} P^{GB}(h_l\,|\,\mathbf{x})\right) - \frac{2x_i}{\sigma_i^3}\sum_{\mathbf{h}}^{\mathbf{H}}\left(\prod_{l=1}^{M} P^{GB}(h_l\,|\,\mathbf{x})\right)\sum_{j}^{M} w_{ij}h_j\,, \tag{196}$$

$$= \frac{(x_i-b_i)^2}{\sigma_i^3}\left(\sum_{h_1}\dots\sum_{h_M}\prod_{l=1}^{M} P^{GB}(h_l\,|\,\mathbf{x})\right)$$

$$-\frac{2x_i}{\sigma_i^3}\sum_{j}^{M}\left(\sum_{h_1}\dots\sum_{h_j}\dots\sum_{h_M} w_{ij}h_j\prod_{l=1}^{M} P^{GB}(h_l\,|\,\mathbf{x})\right)\,, \tag{197}$$

$$= \frac{(x_i-b_i)^2}{\sigma_i^3}\left(\sum_{h_1} P^{GB}(h_1\,|\,\mathbf{x})\times\ldots\times\sum_{h_M} P^{GB}(h_M\,|\,\mathbf{x})\right)$$

$$-\frac{2x_i}{\sigma_i^3}\sum_{j}^{M}\left(\sum_{h_1} P^{GB}(h_1\,|\,\mathbf{x})\times\ldots\times\sum_{h_{j-1}} P^{GB}(h_{j-1}\,|\,\mathbf{x})\times\right. \tag{198}$$

$$\left.\sum_{h_{l=j}} P^{GB}(h_j\,|\,\mathbf{x})\,w_{ij}h_j\times\sum_{h_{j+1}} P^{GB}(h_{j+1}\,|\,\mathbf{x})\times\ldots\times\sum_{h_M} P^{GB}(h_M\,|\,\mathbf{x})\right)\,, \tag{199}$$

$$= \frac{(x_i-b_i)^2}{\sigma_i^3}(1\times\cdots\times 1)$$

$$-\frac{2x_i}{\sigma_i^3}\sum_{j}^{M}\left(1\times\ldots\times 1\times\sum_{h_{l=j}} P^{GB}(h_j\,|\,\mathbf{x})\,w_{ij}h_j\times 1\ldots\times 1\right)\,, \tag{200}$$

$$= \frac{(x_i-b_i)^2}{\sigma_i^3} - \frac{2x_i}{\sigma_i^3}\sum_{j}^{M} P^{GB}(h_j=0\,|\,\mathbf{x})w_{ij}\,0 + P^{GB}(h_j=1\,|\,\mathbf{x})w_{ij}\,1\,, \tag{201}$$

$$= \frac{(x_i-b_i)^2}{\sigma_i^3} - \frac{2x_i}{\sigma_i^3}\sum_{j}^{M} P^{GB}(h_j=1\,|\,\mathbf{x})w_{ij}\,. \tag{202}$$

3.4 Training Boltzmann Machines

The gradient for MRFs (58), defined as the difference of two expectation values is not tractable in the case of BMs, so that we need to use approximation methods instead.

In general it is possible to estimate an expectation sufficiently well by a finite sum of samples drawn independently from the corresponding distribution [5]. We could therefore approximate the gradient for MRFs (58) by:

$$\begin{aligned}
\frac{\partial\,\mathcal{L}^{MRF}\left(\mathbf{x}\,|\,\boldsymbol{\theta}\right)}{\partial\theta} &= -\mathbb{E}_{P^{MRF}(\mathbf{h}\,|\,\mathbf{x})}\left[\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right] \\
&\quad +\mathbb{E}_{P^{MRF}(\tilde{\mathbf{x}})}\left[E_{P^{MRF}(\tilde{\mathbf{h}}\,|\,\tilde{\mathbf{x}})}\left[\frac{\frac{1}{T}\,\partial E(\tilde{\mathbf{x}},\tilde{\mathbf{h}})}{\partial\theta}\right]\right], \qquad (203) \\
&\approx -\left\langle\frac{\frac{1}{T}\,\partial E(\mathbf{x}_d,\mathbf{h}_d)}{\partial\theta}\right\rangle_{P^{MRF}(\mathbf{h}_d\,|\,\mathbf{x}_d)} + \left\langle\frac{\frac{1}{T}\,\partial E(\mathbf{x}_m,\mathbf{h}_m)}{\partial\theta}\right\rangle_{P^{MRF}(\mathbf{h}_m\,|\,\mathbf{x}_m)}, \qquad (204) \\
&= -\left\langle\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right\rangle_{data} + \left\langle\frac{\frac{1}{T}\,\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right\rangle_{model}, \qquad (205)
\end{aligned}$$

where in general $\langle\,\cdot\,\rangle_{P(\,\cdot\,)}$ denotes the average over the samples coming from the probability distribution $P(\,\cdot\,)$. The approximation (204) is only reliable if the samples $\mathbf{x}_d$ are drawn independently form the data distribution and the samples $\mathbf{x}_m$ are drawn independently from the current model distribution. We are then able to choose the most likely hidden states using the corresponding conditional probability $P^{MRF}(\,\mathbf{h}_d\,|\,\mathbf{x}_d\,)$ and $P^{MRF}(\,\mathbf{h}_m\,|\,\mathbf{x}_m\,)$, respectively.

Since we assumed the data being i.i.d. the first term in (205) can be calculated directly using the data and the sampled hidden states. But we encounter a problem when we want to estimate the second term, since we do not have any independently drawn samples from the current model distribution. Consequently, we need a method to generate samples from the model distribution.

3.4.1 Markov Chain Monte Carlo Methods

Markov Chain Monte Carol methods (MCMC) [31] are widely used techniques for numerical sampling. They allow sampling from a large class of distributions including Boltzmann distributions and therefore MRFs. Furthermore MCMC scale well with the dimensionality of the data, which made them become very popular especially in the context of probabilistic machine learning models. An advisably general introduction to sampling methods that goes beyond the brief introduction of this work is given in [5].

Suppose we have a distribution $P(\mathbf{x}) = \frac{1}{Z}\tilde{P}(\mathbf{x})$, we call desired distribution. We cannot sample easily from this distribution, but we are able to evaluate the unnormalized probability $\tilde{P}(\mathbf{x})$ efficiently, as it is the case for BMs for example. The fundamental idea of most sampling algorithms is, to choose a so called proposal distribution $Q(\mathbf{x})$ from which we are able generate samples of. Samples from the proposal distribution are then accepted as samples from the desired distribution if they fulfil an appropriate acceptance criterion. Consequently, $Q(\mathbf{x})$ should be chosen as similar as possible to $P(\mathbf{x})$ but as simple as necessary to be able to sample from it easily.
For MCMC, the proposal distribution is conditioned on the previous sample by $Q(\mathbf{x}^{(N+1)}\,|\,\mathbf{x}^{(N)})$, so that the samples form a Markov chain. A Markov chain is a sequence of random variables $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \cdots, \mathbf{x}^{(N)}$ that fulfil the Markov property. The Markov property, which strictly speaking means the first-order Markov property, expresses that the next variable in the sequence only depends on the current variable, defined by:

$$P(\mathbf{x}^{(N+1)}\,|\,\mathbf{x}^{(N)}\,,\,\cdots,\mathbf{x}^{(1)}) = P(\mathbf{x}^{(N+1)}\,|\,\mathbf{x}^{(N)})\,. \tag{206}$$

This definition could also be extended to the N^{th}-order Markov property where the current variable only depends on the N^{th} previous variables. For convenience the first order Markov property is assumed if no order is mentioned explicitly.

The first MCMC algorithm was the basic Metropolis algorithm, which assumes the proposal distribution to be symmetric, $Q(\mathbf{x}^{(N+1)}\,|\,\mathbf{x}^{(N)}) = Q(\,\mathbf{x}^{(N)}|\,\mathbf{x}^{(N+1)})$. It defines an acceptance probability for a new sample $\mathbf{x}^{(N+1)}$ from the proposal distribution by the ratio of current sample's and the new sample's probability under the desired distribution. The Metropolis acceptance ratio is therefore defined as:

$$A^{MR}(\mathbf{x}^{(N+1)}\,,\,\mathbf{x}^{(N)}) = min\left(1\,,\,\frac{\tilde{P}(\mathbf{x}^{(N+1)})}{\tilde{P}(\mathbf{x}^{(N)})}\right)\,, \tag{207}$$

where the partition function has cancelled out. But usually, as it is the case in BMs, the proposal distribution is not symmetric. The Metropolis-Hastings acceptance ratio, generalizes the Metropolis ratio to non symmetric proposal distributions by taking into account how likely the samples are under the proposal distribution. It therefore multiplies the Metropolis ratio by the ratio between the probability that the current sample generates the next sample $Q(\,\mathbf{x}^{(N+1)}|\,\mathbf{x}^{(N)})$ and the probability that the next sample generates the current sample $Q(\,\mathbf{x}^{(N)}|\,\mathbf{x}^{(N+1)})$, under the proposal distribution. The Metropolis-Hastings ratio is therefore defined as:

$$A^{MHR}(\mathbf{x}^{(N+1)}\,,\,\mathbf{x}^{(N)}) = min\left(1\,,\,\frac{\tilde{P}(\mathbf{x}^{(N+1)})\,Q(\,\mathbf{x}^{(N)}|\,\mathbf{x}^{(N+1)})}{\tilde{P}(\mathbf{x}^{(N)})\,Q(\mathbf{x}^{(N+1)}\,|\,\mathbf{x}^{(N)})}\right)\,. \tag{208}$$

Obviously symmetric proposal functions cancel out so that the Metropolis Hastings ratio reduces to the standard Metropolis ratio in that case.

Algorithm 1 Metropolis Hastings Algorithm

```
Require: $N\,,\,k\,,\,\mathbf{x}^{(init)}\,,\,\tilde{P}(\,\cdot\,)\,,\,Q(\,\cdot\,|\,\cdot\,)$
  $\mathrm{S} \leftarrow \{\ \}$
  $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(init)}$
  $i \leftarrow 0$
  while $i < N$ do
      $u \leftarrow 0$
      while $u < k$ do
          $\mathbf{x^{(1)}} \sim Q(\mathbf{x}\,|\,\mathbf{x^{(0)}})$
          if $A^{MHR}(\mathbf{x^{1}}\,,\,\mathbf{x^{(0)}}) \geq random(0,1)$ then
              $\mathbf{x^{(0)}} \leftarrow \mathbf{x^{(1)}}$
          end if
          $u \leftarrow u + 1$
      end while
      $\mathrm{S} \leftarrow \mathrm{S} \cup \mathbf{x^{(0)}}$
      $i \leftarrow i + 1$
  end while
  return S
```

The Metropolis Hastings Algorithm shown in Algorithm 1 uses the ratio (208) to generate a set of samples from the desired distribution. It therefore takes the proposal distribution $Q(\,\cdot\,|\,\cdot\,)$, the unnormalized desired distribution $\tilde{P}(\,\cdot\,)$, the number of samples we want to generate N, an acceptance step size k and the initial state of the variables $\mathbf{x}^{(init)}$. We assume the $\mathbf{x}^{(init)}$ as random if not stated otherwise. Let's assume $k = 1$ for the moment, then in each iteration the algorithm samples a new state $\mathbf{x^{(1)}}$ from the previous state $\mathbf{x^{(0)}}$ and calculates the ratio between them. If the ratio is bigger than a uniform random value between zero and one, the sample is accepted, added to the sample set S and assigned to $\mathbf{x^{(0)}}$ for the next iteration.

Obviously if the sample is not accepted we keep the same $\mathbf{x^{(0)}}$ for next iteration. Therefore, it is most likely that we add the same sample multiple times to our set and that the samples are not independent of each other. That is the reason why we choose a step size k, which regularizes that only every k^{th} sample is added to the set of samples. In the limit case $k \to \infty$, this guarantees that the Markov chain converges to the stationary desired distribution, so that the samples are drawn independently. Consequently, if k is big enough we get samples that are almost independent, which is sufficient for most applications.

However, it is unknown how big k needs to be in order to generate a reliable set of samples and this choice highly depends on the complexity of our model PDF. In a BM where each variable depends on all others, k usually needs to be very big, which makes the sampling become intractable due to the computational cost. But for simpler models like RBMs for example, a small k is often sufficient for generating a reliable set of samples. But the convergence speed of the Markov chain to the stationary distribution also depends on the choice of the proposal distribution, which should be as close as possible to the desired distribution.

3.4.2 Gibbs Sampling

Gibbs sampling is a very popular MCMC algorithm, which is a special case of the Metropolis-Hastings algorithm. It offers a smart way to choose the proposal distribution depending on the desired distribution.

Given the desired distribution $P(\mathbf{x}) = P(x_0, \cdots, x_D)$ for Gibbs sampling we need to be able to formulate the proposal distribution as the conditional probability of a variable x_i given all other variables $\mathbf{x}_{\setminus i} = \{x_0, \cdots, x_D\} \setminus \{x_i\}$. The proposal distribution is then given by $P(x_i \,|\mathbf{x}_{\setminus i})$ and allows to reformulate the desired distribution to $P(\mathbf{x}) = P(x_i \,|\, \mathbf{x}_{\setminus i})\, P(\mathbf{x}_{\setminus i})$. By inserting the functions into the Metropolis-Hastings ratio (208), it turns out that the ratio for Gibbs sampling becomes constantly one and therefore all samples are accepted.

$$
\begin{aligned}
A^{GS}(x^{(N+1)}\,,\,x^{(N)}) &= min\left(1\,,\,\frac{\frac{1}{Z}\tilde{P}(\mathbf{x}^{(N)})\,Q(\,\mathbf{x}^{(N)}|\,\mathbf{x}^{(N+1)})}{\frac{1}{Z}\tilde{P}(\mathbf{x}^{(N)})\,Q(\mathbf{x}^{(N+1)}\,|\,\mathbf{x}^{(N)})}\right)\,, && (209)\\
&= min\left(1\,,\,\frac{P(\mathbf{x}^{(N+1)})\,P(\,x_i^{(N)}|\,\mathbf{x}_{\backslash i}^{(N+1)})}{P(\mathbf{x}^{(N)})\,P(x_i^{(N+1)}\,|\,\mathbf{x}_{\backslash i}^{(N)})}\right)\,, && (210)\\
&= min\left(1,\frac{P(x_i^{(N+1)}|\mathbf{x}_{\backslash i}^{(N+1)})P(\mathbf{x}_{\backslash i}^{(N+1)})P(x_i^{(N)}|\mathbf{x}_{\backslash i}^{(N+1)})}{P(x_i^{(N)}|\mathbf{x}_{\backslash i}^{(N)})P(\mathbf{x}_{\backslash i}^{(N)})P(x_i^{(N+1)}|\mathbf{x}_{\backslash i}^{(N)})}\right), && (211)\\
&= min\left(1,\frac{P(x_i^{(N+1)}|\,\mathbf{x}_{\backslash i}^{(N)})\,P(\mathbf{x}_{\backslash i}^{(N)})\,P(x_i^{(N)}|\,\mathbf{x}_{\backslash i}^{(N)})}{P(x_i^{(N)}|\,\mathbf{x}_{\backslash i}^{(N)})\,P(\mathbf{x}_{\backslash i}^{(N)})\,P(x_i^{(N+1)}\,|\,\mathbf{x}_{\backslash i}^{(N)})}\right)\,, && (212)\\
&= 1\,, && (213)
\end{aligned}
$$

where we used in (211) that we only change $\mathbf{x}_i^{(N)}$ to $\mathbf{x}_i^{(N+1)}$ when sampling and therefore $\mathbf{x}_{\backslash i}^{(N+1)} = \mathbf{x}_{\backslash i}^{(N)}$.

It is worth mentioning that $\mathbf{x}^{(N)}$ and $\mathbf{x}^{(N+1)}$ are highly dependent after one step of Gibbs sampling and that we therefore will only get independent samples in the limit case, when updating all variables randomly and equally often. But due to the computational cost one usually wants to sample only a few times, which increases the probability that variables are updated differently often. Therefore, it is better to consider only the samples after all variables have been updated, in a fixed or random order equally often.

Taking this into account we can formulate the Gibbs sampling algorithm as a variant of the Metropolis-Hastings algorithm as shown in Algorithm 2 .

For BMs, the proposal function for sampling a visible or hidden state is therefore defined as:

$$
\begin{aligned}
Q^{BM}(x_i\,|\,\mathbf{x}_{\backslash i}^{(N)}\,,\,\mathbf{h}) &= P^{BM}(x_i\,|\,\mathbf{x}_{\backslash i}^{(N)}\,,\,\mathbf{h})\,, && (214)\\
Q^{BM}(h_j\,|\,\mathbf{x}\,,\,\mathbf{h}_{\backslash i}^{(M)}) &= P^{BM}(h_j\,|\,\mathbf{x}\,,\,\mathbf{h}_{\backslash i}^{(M)})\,. && (215)
\end{aligned}
$$

In RBMs the visible units are conditionally independent as well as the hidden units, so that the proposal distribution becomes:

$$
\begin{aligned}
Q^{RBM}(x_i\,|\,\mathbf{x}_{\backslash i}^{(N)}\,,\,\mathbf{h}) &= P^{RBM}(x_i\,|\,\mathbf{h})\,, && (216)\\
Q^{RBM}(h_j\,|\,\mathbf{x}\,,\,\mathbf{h}_{\backslash i}^{(M)}) &= P^{RBM}(h_j\,|\,\mathbf{x})\,. && (217)\\
& && (218)
\end{aligned}
$$

Algorithm 2 Gibbs Sampling

```
Require: $N$ , $k$ , $\mathbf{x}^{(init)}$ , $Q(\,\cdot\,|\,\cdot\,)$
  $\mathrm{S} \leftarrow \{\ \}$
  $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^{(init)}$
  $i \leftarrow 0$
  while $i < N$ do
      $u \leftarrow 0$
      while $u < k$ do
          for $d = 1$ to $D$ do
              $x_d^{(0)} \sim Q(x_d \,|\, x_0^{(0)}, \cdots, x_{d-1}^{(0)}, x_{d+1}^{(0)}, \cdots, x_D^{(0)})$
          end for
          $u \leftarrow u + 1$
      end while
      $\mathrm{S} \leftarrow \mathrm{S} \cup \mathbf{x}^{\mathbf{(0)}}$
      $i \leftarrow i + 1$
  end while
  return S
```

Therefore, Gibbs sampling in RBMs has the advantage that we are able to sample the visible or hidden states in parallel. If we first sample all visible states and then all hidden states the Gibbs sampling in RBMs can be parallelized efficiently. The Gibbs sampling schema is shown in Figure 10.

$$\begin{tikzcd}
 & \mathbf{h}^{(0)} \sim P^{RBM}(\mathbf{h}\,|\,\mathbf{x}^{(0)}) \arrow[dr] & & \mathbf{h}^{(1)} \sim P^{RBM}(\mathbf{h}\,|\,\mathbf{x}^{(1)}) \arrow[dr] & & \cdots & {} \arrow[dr] & & \mathbf{h}^{(k)} \sim P^{RBM}(\mathbf{h}\,|\,\mathbf{x}^{(k)}) \\
\mathbf{x}^{(0)} \arrow[ur] & & \mathbf{x}^{(1)} \sim P^{RBM}(\mathbf{x}\,|\,\mathbf{h}^{(0)}) \arrow[ur] & & {} & & & \mathbf{x}^{(k)} \sim P^{RBM}(\mathbf{x}\,|\,\mathbf{h}^{(k-1)}) \arrow[ur] &
\end{tikzcd}$$

Figure 10: Visualization of the Markov chain in Gibbs sampling for an RBM.

3.4.3 Contrastive Divergence

Approximating the MRF gradient (205) by a finite set of samples generated by k steps of Gibbs sampling is named Contrastive Divergence (CD) [18]. Algorithm 3 shows the pseudo code for the offline learning version of CD-k, which could be easily transformed into online and batch learning. The algorithm requires the training data

$\mathbf{X}_D$, the model's conditional distribution $P(\,\cdot\,|\,\cdot\,)$, the parametrization $\boldsymbol{\theta}$, the number of Gibbs sampling steps k and the learning rate η. In each iteration of the inner loop we sample the hidden state $\mathbf{h}^{(0)}$ given the data $\mathbf{x}_d^{(0)}$ and use k step Gibbs-sampling to generate the model representative samples $\mathbf{x}^{(k)}, \mathbf{h}^{(k)}$. The samples are used to compute the gradient approximation $\tilde{\nabla}\boldsymbol{\theta}$. Finally the parameters $\boldsymbol{\theta}$ are update in each outer loop by the average gradient approximation scaled by the learning rate η.

Algorithm 3 Constrastive Divergence

```
Require: $\mathbf{X}_D\,,\ P(\,\cdot\,|\,\cdot\,)\,,\ \boldsymbol{\theta}\,,\ k\,,\ \eta\,)$
  while Stopping criterion is not met do
    $\tilde{\nabla}\boldsymbol{\theta} \leftarrow 0$
    for all $\mathbf{x}_d \in \mathbf{X}_D$ do
      $\mathbf{x}_d^{(0)}, \mathbf{h}_d^{(0)}, \mathbf{x}_d^{(k)}, \mathbf{h}_d^{(k)} \leftarrow \mathit{GibbsSampling}\,(1\,,\ k\,,\ \mathbf{x}_d\,,\ P(\,\cdot\,|\,\cdot\,))$    ▷ Alg.(2)
      $\tilde{\nabla}\boldsymbol{\theta} \leftarrow \tilde{\nabla}\boldsymbol{\theta} + \left\langle \frac{\partial E(\mathbf{x}^{(0)},\mathbf{h}^{(0)})}{\partial \theta} \right\rangle - \left\langle \frac{\partial E(\mathbf{x}^{(k)},\mathbf{h}^{(k)})}{\partial \theta} \right\rangle$    ▷ Eq. (205)
    end for
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \frac{\eta}{D}\tilde{\nabla}\boldsymbol{\theta}$
  end while
```

As shown in (46) maximizing the LL is equivalent to minimize $\mathrm{KL}\,(F(\mathbf{x})\,||\,P(\mathbf{x}\,|\,\theta))$. When using k step Gibbs sampling we only approximate the model distribution by $P_k(\mathbf{x}\,|\,\theta)$, which causes an error given by the difference of the true model distribution and the approximation by: $\mathrm{KL}\,(P_k(\mathbf{x}\,|\,\theta)\,||\,P(\mathbf{x}\,|\,\theta))$. So that CD actually does not minimize the Kullback-Leibler divergence between data and model exactly, instead it minimizes the so called contrastive divergence given by:

$$\mathrm{KL}\,(F(\mathbf{x})\,||\,P(\mathbf{x}\,|\,\theta)) - \mathrm{KL}\,(P_k(\mathbf{x}\,|\,\theta)\,||\,P(\mathbf{x}\,|\,\theta))\,. \tag{219}$$

Gibbs sampling is guaranteed to produce true samples from the model distribution when $k \leftarrow \infty$ so that $P_{k\to\infty}(\mathbf{x}\,|\,\theta) = P(\mathbf{x}\,|\,\theta)$, which causes the second term in (219) to be zero. Consequently, CD is equivalent to maximizing the LL when $k \to \infty$ or k is big enough so that it can be guaranteed that the Markov chain converged to the station distribution. Surprisingly even for $k = 1$ CD performs already quite well so that the produced error does not affect the gradient very much.

In the original CD algorithm Gibbs sampling is initialized by the current data point, which cause the approximation of the gradient to be highly depended on the data. A variant of CD called Persistent Contrastive Divergence [41] (PCD), initializes the

Gibbs sampling with the last sample $\mathbf{x}_{d-1}^{(k)}$ instead of the current data point $\mathbf{x}_d$. This makes the approximation of the model distribution more independent of the data and therefore closer to the true LL. In CD, sampling noise influences only the current approximation since we reset the Markov chain for the next approximation. In PCD however the noise can possibly effect the following approximations. Therefore, the learning rate for PCD usually needs to be effectively smaller to compensated this effect.

Fast Persistent Contrastive Divergence (FPCD) tries to speed up PCD by combining two sets of parameters θ_{fast}, which are updated using a big learning rate and $\theta_{regular}$, which are updated using a smaller learning rate. The samples for the second term of the gradient are then generated using an overlay of the parameters by $\mathbf{x}^{(k)} \sim P(\mathbf{x} \,|\, \mathbf{h}, \theta_{fast} + \theta_{regular})$ and $\mathbf{h}^{(k)} \sim P(\mathbf{h} \,|\, \mathbf{x}, \theta_{fast} + \theta_{regular})$. As mentioned already both parameters are then updated using the same gradient but with different learning rates. This has the effect that the Markov chain mixes faster so that the convergence to the stationary distribution is fasten. However, the algorithm introduces additional hyper-parameters, which leads the learning to fail if they are not chosen correctly.

3.4.4 Parallel Tempering

The samples of a Markov chain are only guaranteed to be drawn independent from the desired distribution in the limit case. When using Gibbs sampling with a small k, we most likely generate dependent samples. This effect is illustrated in Figure (11), where the samples tend to stay close to the previous samples. Consequently, if the initial samples do not cover all modes of the model distribution, the generated samples will also most likely not cover these modes. This leads to an estimation of the model distribution that is biased on the initial samples.

The question is therefore how we could generate samples that are distributed over all modes. For MCMC sampling methods this means that we want the samples, after one step of sampling to be as independent of the previous samples as possible but still be a representative sample of our distribution. This describes what is known in the literature as a ”fast mixing” Markov chain.

Parallel Tempering [11] (PT) is an algorithm that provides a fast mixing rate and surprisingly, we already know all concepts this algorithm is working with. First of all let us reconsider the PDF of MRFs (19) where we defined the temperature parameter $T \in [1, \infty)$, which we discarded up to now. It scales the energy down, which leads

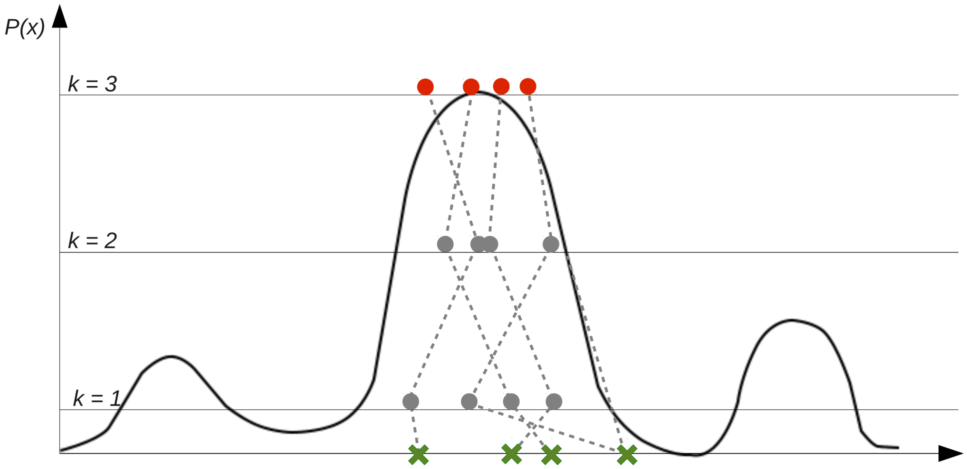


Figure 11: Illustration of generating samples for a distribution (black line) using Gibbs sampling. The final samples (red dots) and intermediate samples (grey dots) tend to stay close to the initial samples (green crosses), indicated by the dashed pathways. The generated sampling missed the two smaller modes so that they are not a representative set of samples for this distribution.

to a regularization of the PDF's manifold. This becomes clear if we think of that the energy is applied to an exponential function to calculate the probability. If we choose a big temperature the energy is scaled down, which leads to more equally distributed probabilities, due to nature of the exponential function.
Therefore, we can use the temperature to generate samples, which are distributed more homogeneously.

The idea of PT is to run several Markov chains on different temperatures. We start Gibbs sampling from the highest temperature where all samples have the same probability. While continuing the sampling procedure, the temperature is lowered, which has the effect that regions of higher density are coming up. If the decreasing of the temperatures is smooth enough, the samples will move to all regions of higher density. This generates samples that are likely from all modes of the distribution which is illustrated in Figure 12.

Instead of running the described procedure for each gradient update, PT runs a couple of Markov chains persistently and exchanges samples between the chains after k sampling step. The exchange of samples from a particular chain, is performed with the chain with the next higher temperature and the next lower temperature. Whether

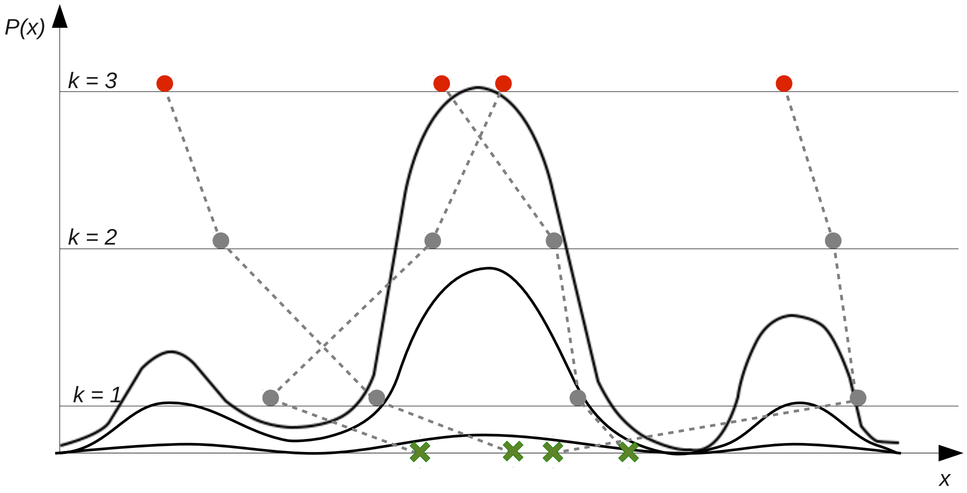


Figure 12: Illustration of generating samples for a distribution (black line) using Parallel Tempering sampling. The model distribution for the first sampling step $k = 1$ is scaled down so that it is nearly uniform. The samples (grey dots) spread randomly over the whole input space. For the second sampling step $k = 2$ the model distribution is scaled down less so that the three modes appear, which attract the samples of the previous step. The final samples (red dots) are distributed over all modes so that they represent a good set of samples for the final model distribution.

two samples are exchanged, will be determined using the Metropolis-Hastings ratio (208) as already explained. Since PT is just an advanced way of sampling it can just be used for CD instead of normal Gibbs sampling.

The pseudo code for one iteration of PT is given in Algorithm 4, which requires the number of Gibbs sampling steps k, the conditional probability distribution $P(\cdot\,|\,\cdot)$, the unnormalized probability distribution $\tilde{P}(\cdot)$, the temperatures ordered from big to small values $(T_1 = \infty\,, \cdots, T_L = 1)$ and the initial samples $\left(\mathbf{x}_1^{(init)}, \cdots, \mathbf{x}_T^{(init)}\right)$. To implement the persistent chain, the samples of the last iteration are used as initial samples.

The first step of the algorithm, is to sample from models on different temperatures using Gibbs sampling, where $P_{T_l}(\cdot\,|\,\cdot)$ denotes that we are sampling on temperature T_l. Afterwards the samples are exchanged using the Metropolis-Hastings ratio and the exchange order is determined using a deterministic even odd algorithm [10].

Algorithm 4 Parallel Tempering Sampling

```
Require: $k\,,\, P(\,\cdot\,|\,\cdot\,)\,,\, \tilde{P}(\,\cdot\,)\,,\, (T_1 = \infty\,, \cdots, T_L = 1)\ ,\ \left(\mathbf{x}_1^{(init)}, \cdots, \mathbf{x}_L^{(init)}\right)$
  $\left(\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_L^{(k)}\right) \leftarrow \left(\mathbf{x}_1^{(init)}, \cdots, \mathbf{x}_L^{(init)}\right)$
  for $l = 1$ to $L$ do
      $\mathbf{x}_l^{(k)} \leftarrow GibbsSampling\,(1\,,\, k\,,\, \mathbf{x}_l\,,\, P_{T_l}(\,\cdot\,|\,\cdot\,))$                ▷ Alg.(2)
  end for
  $l \leftarrow 2$
  while $l < L$ do
      if $\left(\frac{\tilde{P}_{T_l}(\mathbf{x}_{l+1}^{(k)})\,\tilde{P}_{T_{l+1}}(\mathbf{x}_l^{(k)})}{\tilde{P}_{T_l}(\mathbf{x}_l^{(k)})\,\tilde{P}_{T_{l+1}}(\mathbf{x}_{l+1}^{(k)})}\right) \geq random(0,1)$ then                ▷ Eq.(208)
          $\mathbf{x} \leftarrow \mathbf{x}_l^{(k)}$
          $\mathbf{x}_l^{(k)} \leftarrow \mathbf{x}_{l+1}^{(k)}$
          $\mathbf{x}_{l+1}^{(k)} \leftarrow \mathbf{x}$
      end if
      $l \leftarrow l + 2$
  end while
  $l \leftarrow 1$
  while $l < L$ do
      if $\left(\frac{\tilde{P}_{T_l}(\mathbf{x}_{l+1}^{(k)})\,\tilde{P}_{T_{l+1}}(\mathbf{x}_l^{(k)})}{\tilde{P}_{T_l}(\mathbf{x}_l^{(k)})\,\tilde{P}_{T_{l+1}}(\mathbf{x}_{l+1}^{(k)})}\right) \geq random(0,1)$ then                ▷ Eq.(208)
          $\mathbf{x} \leftarrow \mathbf{x}_l^{(k)}$
          $\mathbf{x}_l^{(k)} \leftarrow \mathbf{x}_{l+1}^{(k)}$
          $\mathbf{x}_{l+1}^{(k)} \leftarrow \mathbf{x}$
      end if
      $l \leftarrow l + 2$
  end while
  return $\left(\mathbf{x}_1^{(k)}, \cdots, \mathbf{x}_L^{(k)}\right)$
```

3.4.5 Regularizing the Gradient

There are several modifications, which can be applied on RBMs and its training procedures. A practical tutorial about those modification and the choice of the corresponding hyperparameters is give in [19].
The learning rate η controls the speed of learning and needs to be chosen model and problem depended. If it is chosen to big, the gradient could be prevented from converging or could even diverge. If the learning rate is to small the training is usually very slow and can get stuck in a local optima with an unsatisfying LL. A common choice of the learning rate, which have been reported by many authors is 0.1 for BB-RBMs and 0.01 for GB-RBMs.
The authors in [8] proposed an adaptive learning that works on a local approximation of the LL. Surprisingly, their empirical analysis showed that the learning rate converges to a value between 0.1 and 0.01 for BB-RBMs, after approximately 400 gradient updates, which corresponds to a full loop through the data.

Since the stochastic gradient approximation of CD is affected to noise, training is usually performed in batch mode. Additionally, to reduce the effect, we could add a momentum term δ that helps to stabilize the gradient. This is achieved by adding a percentage of the gradient of the previous update to the current gradient before updating the parameter. Accordingly, the gradient becomes a weighted sum of the current and the previous gradients. Therefore, local influence of noise gets compensated by the averaging process, which leads to sluggish change of the gradient direction.
To prevent the weights of growing incomprehensible big, we could regularize the gradient using an L2-norm of the weights. The derivative, which is simply the weight norm, is then scaled by a weight decay parameter λ and subtracted from the current gradient.
The update rule for the BM parameters θ_{BM} with gradient $\nabla_{\theta_{BM}}(t)$ at time step t with learning rate η, momentum term δ and weight decay term λ is then given by:

$$\theta_{BM} \leftarrow \theta_{BM} + \eta \left[\nabla_{\theta_{BM}}(t) + \delta \, \nabla_{\theta_{BM}}(t-1) - \lambda \, \theta_{BM} \right] \, . \tag{220}$$

A sparse representation of the data is often a desirable property since it structures the data more clearly, which supports discrimination tasks. In [35] the authors proposed an alternative or additional sparseness penalty term, which forces the number of active hidden units $\mathbf{h}^{(0)}$ to stay smaller than a chosen threshold ε given by:

$$\lambda_s \, | \varepsilon - \left\langle \mathbf{h}^{(0)} \right\rangle_{h_j} |^2 \, , \tag{221}$$

where λ_s regularizes this penalty.

The effect of noise and a big learning rate can easily lead to divergence of the gradient especially in the case of GB-RBM. Therefore, it can be of interested to restrict the norm of the gradient not to become incomprehensible big. Especially for GB-RBM we are able to choose a plausible upper bound of the gradient norm. In the experiments we will see that this prevents divergence and allows to increase the learning rate.

The gradient is sensible to the representation of the data, so that learning a data set where each bit is flipped is harder than learning the original data set. It seems that BB-RBMs assumes the important part of the data to be coded as ones so that if training a BB-RBM fails, one should try to learn the flipped version of the data. The authors in [8] recently proposed an enhanced version of the gradient, which seems to be more robust to the representation of the data.

3.4.6 Performance Measures in Training

Since we are usually not able to calculate the LL during training, we have no direct measurement for the convergence of the training process. A simple and most natural idea is to visualize the BMs weights during training, which should obviously contain some data related structure. Figure (13) shows the weights of a BB-RBM trained on the MNIST [27] dataset, which consist of 60,000 binary images showing handwritten digits of the size 28x28. The weights have some stroke like structures, which are the learned features of the handwritten digits.

However, structured weights do not allow to interpret how well these perform compared to other sets of weights. But we could perform k steps of Gibbs sampling and visualize the samples, to see whether the BM reconstructs images similar to the training data. Figure 14 shows the binary training data and the corresponding reconstructions from the RBM trained on the MNIST datset. Between each row one step of Gibbs sampling was performed. The digits vary during sampling but stay relatively close to the original sample.

Instead or in addition to the visualization of samples, we can calculate the reconstruction error defined as the average squared distance of the training data and its samples generated by k steps of Gibbs-sampling. The k^{th} order reconstruction error, where we assume the first order error if not stated otherwise is then given by:

$$\mathcal{RE}^k\left(\mathbf{x}\,,\,\mathbf{x}^{(k)}\right) = \frac{1}{DN}\sum_{d}^{D}\sum_{i}^{N}\left(x_i - x_i^{(k)}\right)^2 . \tag{222}$$

If the RBM is used in a classification task we could also choose the classification

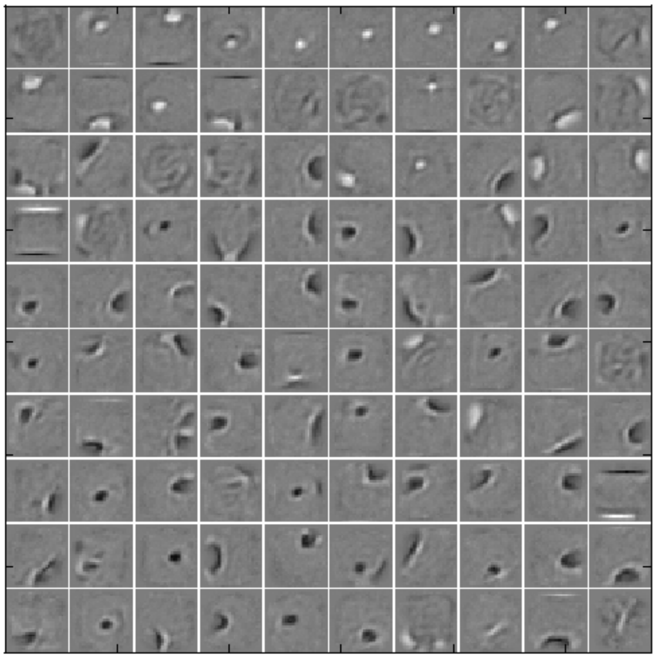


Figure 13: Weights of a BB-RBM with 100 hidden units trained on the MNIST dataset showing stroke like features.

error as an indirect convergence measurement.

Furthermore in [12] the authors analysed empirically that CD learning can diverge after converging to a local optimum. While the true LL diverged the reconstruction error did not, so that it is not a reliable proxy to the LL.

We could also calculate the pseudo log likelihood (238) as an approximation to the LL, but it differs usually quite a lot from the true LL. The behaviour of LL and PLL could also be contrastive especially in later stages of training.

Therefore, the only reliable performance measure is the true LL, which is traceable for RBMs if at least one layer is binary and small enough. Since then the partition function can be calculate using (84) or (83) for BB-RBMs or (161) for GB-RBMs. For bigger models there exists the opportunity to approximate the partition function by MCMC methods.

3.4.7 Annealed Importance Sampling

Annealed Importance Sampling (AIS) [32] is a variant of Importance Sampling (IS) [5], which is a general method to approximate expectation values for distributions

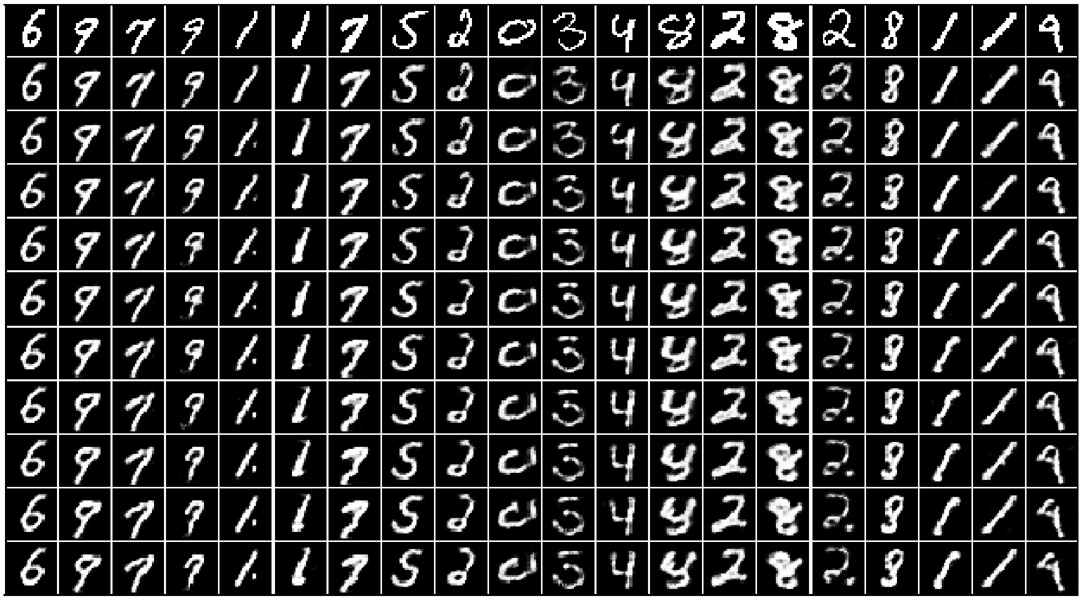


Figure 14: (first row) Training data of the MNIST dataset [27] and (second to tenth row) the corresponding reconstructions. From one row to the next, ten steps of Gibbs sampling were performed. For the reconstruction, the probabilities are displayed instead of the binary states.

from which we cannot sample directly. Similar to MCMC sampling IS introduces a proposal distribution $Q(\mathbf{x})$, which should be as close as possible to the desired distribution $P(\mathbf{x})$, but we need to be able to sample from it easily.
Let us consider the expectation value of x under the desired distribution $P(\mathbf{x})$, which is given in the following form:

$$\begin{aligned} \mathbb{E}_{P(\mathbf{x})}\left[\mathbf{x}\right] &= \int \mathbf{x}\, P(\mathbf{x})\, \mathrm{d}\mathbf{x}\,, && (223)\\ &= \int \mathbf{x}\, \frac{P(\mathbf{x})}{Q(\mathbf{x})} Q(\mathbf{x})\, \mathrm{d}\mathbf{x}\,, && (224)\\ &= \frac{Z_Q}{Z_P} \int \mathbf{x}\, \frac{\tilde{P}(\mathbf{x})}{\tilde{Q}(\mathbf{x})} Q(\mathbf{x})\, \mathrm{d}\mathbf{x}\,, && (225)\\ &\approx \frac{Z_Q}{Z_P} \sum_{l=1}^{L} \mathbf{x}_Q^{(l)}\, \frac{\tilde{P}(\mathbf{x}_Q^{(l)})}{\tilde{Q}(\mathbf{x}_Q^{(l)})}\,, && (226) \end{aligned}$$

where $\tilde{P}(\mathbf{x})$ and $\tilde{Q}(\mathbf{x})$ are the unnormalized PDFs of $P(\mathbf{x})$ and $Q(\mathbf{x})$, respectively and Z_P and Z_Q the corresponding normalization constants. Finally the expectation is approximated over a finite set of i.i.d. samples of $Q(\mathbf{x})$.
In an equivalent way we can evaluate the inverse ratio of $\frac{Z_Q}{Z_P}$ by,

$$\begin{aligned}
\frac{Z_P}{Z_Q} &= \frac{1}{Z_Q}\int \tilde{P}(\tilde{\mathbf{x}})\,\mathrm{d}\tilde{\mathbf{x}}\,, && (227)\\
&= \frac{1}{Z_Q}\int \frac{\tilde{P}(\tilde{\mathbf{x}})}{Q(\tilde{\mathbf{x}})}Q(\tilde{\mathbf{x}})\,\mathrm{d}\tilde{\mathbf{x}}\,, && (228)\\
&= \int \frac{\tilde{P}(\tilde{\mathbf{x}})}{\tilde{Q}(\tilde{\mathbf{x}})}Q(\tilde{\mathbf{x}})\,\mathrm{d}\tilde{\mathbf{x}}\,, && (229)\\
&\approx \sum_{m=1}^{M} \frac{\tilde{P}(\mathbf{x}_Q^{(m)})}{\tilde{Q}(\mathbf{x}_Q^{(m)})}\,, && (230)
\end{aligned}$$

Substituting (229) into (225) and approximating the expectations by a finite set of i.i.d. samples we obtain,

$$\begin{aligned}
\mathbb{E}_{P(\mathbf{x})}\left[\mathbf{x}\right] &\overset{(229),(225)}{=} \frac{\int \mathbf{x}\,\frac{\tilde{P}(\mathbf{x})}{\tilde{Q}(\mathbf{x})}Q(\mathbf{x})\,\mathrm{d}\mathbf{x}}{\int \frac{\tilde{P}(\tilde{\mathbf{x}})}{\tilde{Q}(\tilde{\mathbf{x}})}Q(\tilde{\mathbf{x}})\,\mathrm{d}\tilde{\mathbf{x}}}\,, && (231)\\
&\overset{(230),(226)}{\approx} \frac{\sum_{l=1}^{L}\mathbf{x}_Q^{(l)}\,\frac{\tilde{P}(\mathbf{x}_Q^{(l)})}{\tilde{Q}(\mathbf{x}_Q^{(l)})}}{\sum_{m=1}^{M}\frac{\tilde{P}(\mathbf{x}_Q^{(m)})}{\tilde{Q}(\mathbf{x}_Q^{(m)})}}\,, && (232)\\
&= \sum_{l=1}^{L}\mathbf{x}_Q^{(l)}\,\frac{\frac{\tilde{P}(\mathbf{x}_Q^{(l)})}{\tilde{Q}(\mathbf{x}_Q^{(l)})}}{\sum_{m=1}^{M}\frac{\tilde{P}(\mathbf{x}_Q^{(m)})}{\tilde{Q}(\mathbf{x}_Q^{(m)})}}\,, && (233)\\
&= \sum_{l=1}^{L}\mathbf{x}_Q^{(l)}\,w_l\,. && (234)
\end{aligned}$$

Therefore, IS approximates the expectation using a weighted sum of the samples from $Q(x)$. The weights w_l, known as importances weights, judge the importance of a sample being a representative of $P(x)$ by the ratio of the two probabilities. The two set of samples $\left\{\mathbf{x}_Q^{(1)}, \cdots, \mathbf{x}_Q^{(L)}\right\}$ and $\left\{\mathbf{x}_Q^{(1)}, \cdots, \mathbf{x}_Q^{(M)}\right\}$ are usually chosen to be the same, due to computational cost.

It is clear that the success of IS highly depends on the choice of the proposal function. If it is not sufficiently close to the desired distribution the estimation will be very bad. AIS tries to compensate this effect by combining the idea of IS with the annealing of the temperature in energy based models. Given the temperatures $T_1, \cdots, T_K$ the ratio of the partition functions can be decomposed by:

$$\frac{Z_P}{Z_Q} = \frac{Z_1}{Z_0}\frac{Z_2}{Z_1}\cdots\frac{Z_K}{Z_{K-1}} \,. \tag{235}$$

If the temperatures between two intermediate distribution $P_{T_k}(\mathbf{x})$ and $P_{T_{k+1}}(\mathbf{x})$ are close enough the estimation of the partition function ratio will be sufficiently good.

$$\frac{Z_{k+1}}{Z_k} \overset{(229)}{\approx} \frac{1}{L}\sum_{l}^{L} \frac{\tilde{P}_{T_{k+1}}(\mathbf{x}^{(l)})}{\tilde{P}_{T_k}(\mathbf{x}^{(l)})} \,. \tag{236}$$

So that we can estimate the partition ratio between the PDFs with highest and lowest temperature, with $L = 1$ by:

$$\frac{Z_K}{Z_0} \approx \prod_{k=1}^{K-1} \frac{Z_{k+1}}{Z_k} \tag{237}$$

Bringing it all together Algorithm 5 shows AIS for estimating the partition function of RBMs. Note that the Algorithm already takes care of an important implementation detail, it computes the logarithm probabilities to avoid underflow problems of the product.

Algorithm 5 Annealed importance sampling for estimating the partition function

```
Require: $P(\cdot\,|\,\cdot)\,,\, \tilde{P}(\cdot)\,,\, \mathbf{x}^{(init)}\,,\, Z_{T_\infty}\,,\, (T_1 = \infty\,, \cdots, T_L = 1)$
  $\mathbf{x} = \mathbf{x}^{(init)}$
  $u = 0$
  for $l = 1$ to $L - 1$ do
      $\mathbf{x} \leftarrow GibbsSampling\,(1\,,\, 1\,,\, \mathbf{x}\,,\, P_{T_l}(\cdot\,|\,\cdot))$        ▷ Alg.(2)
      $u \leftarrow u + \ln\left(\tilde{P}_{T_l}(\mathbf{x})\right) - \ln\left(\tilde{P}_{T_{l+1}}(\mathbf{x})\right)$
  end for
  return $\exp\left(\ln(u) - \ln(Z_{T_\infty})\right)$
```

3.4.8 Other Approaches for Training Restricted Boltzmann Machines

Apart from approximating the LL gradient by MCMC methods, there are other learning algorithms, which have been proposed for training BM. This chapter only gives a brief introduction so that the reader gets an idea of the individual approaches.

The pseudo likelihood approximates the joint PDF of a model by the product of one dimensional PDFs, one for each variable. So that we are able to formulate the Pseudo Log Likelihood (PLL) for BMs by:

$$\begin{aligned}\mathcal{PL}^{BM}\left(\mathbf{x}\,,\,\mathbf{h}\,|\,\boldsymbol{\theta}\right) &= \frac{1}{N}\sum_{i}^{N}\ln P^{BM}\left(x_i|\,\mathbf{x}_{\backslash i}\,,\,\mathbf{h}\,,\,\boldsymbol{\theta}\right)+\\ &\quad \frac{1}{M}\sum_{j}^{M}\ln P^{BM}\left(h_j|\,\mathbf{x}\,,\,\mathbf{h}_{\backslash j}\,,\,\boldsymbol{\theta}\right)\ . \end{aligned} \tag{238}$$

The normalization constants of the individual PDFs are tractable one dimensional integrals over all possible values of the corresponding variable. Therefore, we are able to calculate the exact gradient to perform exact inference. However, PLL obviously assumes the data distribution to be separable into one dimensional distributions. Since this is usually not the case, PLL will perform relatively bad compared to CD.

Ratio Matching [30] is an algorithm only for binary models. Its idea is, that we get a feeling for how we should change the model parameters if we compare the probability of the data with the probability of the data where one bit is flipped. The ratio is then computed for all possible flipped versions of the data. The BM Ratio Matching score for a single data point is defined as:

$$\mathcal{RM}^{BM}\left(\mathbf{x}\,|\,\boldsymbol{\theta}\right)=\sum_{d}^{D}\left(\frac{1}{1+\frac{\tilde{P}^{BM}(\mathbf{x})}{\tilde{P}^{BM}(\mathbf{x}^{\neg d})}}\right)^2\ , \tag{239}$$

where $\mathbf{x}^{\neg d}$ denotes that bit d is flipped and the partition function cancelled out.

The idea of Score Matching [20] is close to the idea of Ratio Matching. In Score matching we define a particular score function $\Psi(\cdot)$ and minimize the squared distance between the score of the data distribution and the score of the model distribution. This again cancels out the partition function and the Score Matching of a single data point for BMs, is given in the simplified tractable form as proposed in

[20] by:

$$\mathcal{SM}^{BM}\left(\mathbf{x}\,|\,\boldsymbol{\theta}\right) = \sum_{d}^{D} \frac{1}{2}\left(\Psi_d(\tilde{P}^{BM}(x))\right)^2 + \frac{\partial \Psi_d(\tilde{P}^{BM}(x))}{\partial x_d}\,. \qquad (240)$$

All methods have been analysed and compared to each other and CD in [30] and [38]. PLL performs worst compared to the other methods. Ratio matching performs worst than score matching and CD, but shows nice denoising properties. Score matching does not show a better performance than CD but it has, as well as Ratio Matching a much higher computational cost. The algorithms have not been compared with advance MCMC sampling method like PT or even CD-k with a bigger k yet.
This thesis considers only CD, PCD and PT for training RBMs.

4 Analysis of Gaussian-Binary Restricted Boltzmann Machines

In general, a profound understanding of a model, its capabilities and limitations, requires a clear understanding of how it models data. For probabilistic models like BMs, accordingly, we need to understand how the marginal probability distribution of the input data is structured.
Figure 15 shows the marginal probability density $P^{BB}(\mathbf{x})$ of a BB-RBM with two visible units x_1, x_2 and two hidden units h_1, h_2. The two visible units can take the four possible states $\mathbf{x} \in \{0,1\}^2$, which correspond to the four positions on the plain. The probability for each state, illustrated as cylinders depend on the product of the visible experts e_{x1}, e_{x2}. The experts themselves, referring to (39) are sigmoid functions, which depend on the hidden units and the corresponding weights. The steepness of the experts' sigmoid, controlled by the weights, defines how likely it is to switch from an active to an inactive state and vice versa.

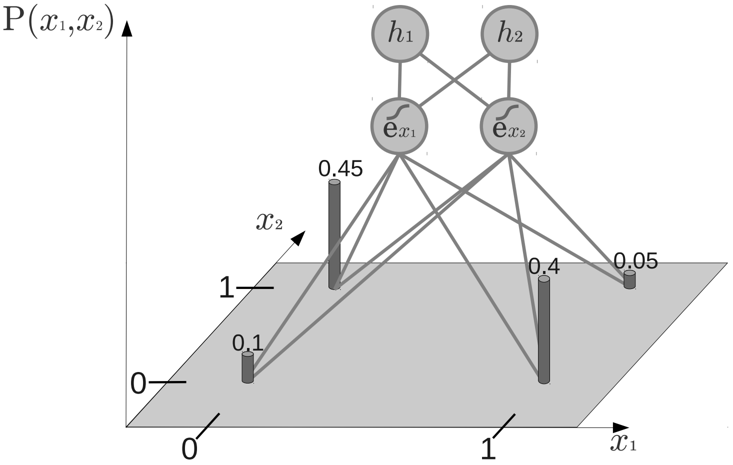


Figure 15: Exemplary illustration for the visible marginal PDF of an RBM with two binary visible units and two arbitrary hidden units. The probabilities denoted as cylinders for the four possible visible states depend on the two experts.

Figure 15 also implies that RBMs can be universal approximators [25]. Let N be the number of visible units and $K \leq \{0,1\}^N$ be the total number of states of the PDF we want to learn. We are able to model the distribution exactly if we have one hidden unit per visible state plus a bias unit, hence $M = 2^N + 1$ hidden units.

4.1 Conceptual Understanding of Gaussian-Binary RBMs

Similar to the illustration for a BB-RBM we are able to illustrate the marginal PDF for a GB-RBM. Referring to (145), the experts marginal PDF has a rather unintuitive form where one expert is an unnormalized Gaussian with mean $\mathbf{b}$ and the other M experts are the sum of the value one and an exponential function.
But we are able to derive a more intuitive formulation of the marginal PDF using the Bayes'theorem and the polynomial expansion as proposed in [43].

$$
\begin{aligned}
P\left(\mathbf{x}\right) \quad &= \quad \sum_{\mathbf{h}} P\left(\mathbf{x}|\mathbf{h}\right) P(\mathbf{h}) && (241)\\
&\overset{(158,177)}{=} \sum_{\mathbf{h}} \mathcal{N}\left(\mathbf{x}; \mathbf{b} + \mathbf{W}\mathbf{h}, \boldsymbol{\sigma}^2\right) \frac{\prod_i^N \sqrt{2\pi\sigma_i}^2}{Z} \mathrm{e}^{\mathbf{c}^T\mathbf{h} + \|\frac{\mathbf{b}+\mathbf{W}\mathbf{h}}{2\boldsymbol{\sigma}^2}\|^2 - \|\frac{\mathbf{b}}{2\boldsymbol{\sigma}^2}\|^2} && (242)\\
&\overset{h_j \in \{0,1\}}{=} \underbrace{\frac{\prod_i^N \sqrt{2\pi\sigma_i}^2}{Z}}_{P(\mathbf{h}:\mathbf{h}\in\mathbf{H}_0)} \mathcal{N}\left(\mathbf{x}; \mathbf{b}, \boldsymbol{\sigma}^2\right)\\
&\quad + \sum_{j=1}^{M} \underbrace{\frac{\prod_i^N \sqrt{2\pi\sigma_i}^2}{Z} \mathrm{e}^{\|\frac{\mathbf{b}+\mathbf{w}_{*j}}{2\boldsymbol{\sigma}^2}\|^2 - \|\frac{\mathbf{b}}{2\boldsymbol{\sigma}^2}\|^2 + c_j}}_{P(\mathbf{h}_j:\mathbf{h}_j\in\mathbf{H}_1)} \mathcal{N}\left(\mathbf{x}; \mathbf{b} + \mathbf{w}_{*j}, \boldsymbol{\sigma}^2\right)\\
&\quad + \sum_{j=1}^{M-1} \sum_{k>j}^{M} \underbrace{\frac{\prod_i^N \sqrt{2\pi\sigma_i}^2}{Z} \mathrm{e}^{\|\frac{\mathbf{b}+\mathbf{w}_{*j}+\mathbf{w}_{*k}}{2\boldsymbol{\sigma}^2}\|^2 - \|\frac{\mathbf{b}}{2\boldsymbol{\sigma}^2}\|^2 + c_j + c_k}}_{P(\mathbf{h}_{jk}:\mathbf{h}_{jk}\in\mathbf{H}_2)}\\
&\quad \mathcal{N}\left(\mathbf{x}; \mathbf{b} + \mathbf{w}_{*j} + \mathbf{w}_{*k}, \boldsymbol{\sigma}^2\right)\\
&\quad + \ldots, && (243)
\end{aligned}
$$

where $\mathbf{H}_u$ denotes the set of all possible binary vectors with exactly u ones and $M-u$ zeros respectively. Accordingly, the binary vector $\mathbf{h}_{jk} \in \mathbf{H}_2$ for example denotes the vector which has only entry j and k set to one and $P(\mathbf{h}_{jk} : \mathbf{h}_{jk} \in \mathbf{H}_2)$ its corresponding marginal probability. $\mathcal{N}\left(\mathbf{x} \,|\, \boldsymbol{\mu}\,, \boldsymbol{\sigma}\right)$ denotes a multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and diagonal covariance matrix, which has the variances σ_i^2 as diagonal elements.
The polynomial expansion in (243) leads to a weighted sum of 2^M Gaussian distributions, which share the variances $\boldsymbol{\sigma}^2$. Following the naming of mixture models [5], each Gaussian distribution is called a component of the model distribution and is scaled by mixing coefficient that is the marginal probability of the corresponding hidden state. Although all components have their own means, they depend on each

other in a well defined structure. The first component is shifted from the origin by the visible bias $\mathbf{b}$ and we name it the anchor component. It corresponds to the case where all hidden units take the value zero. Further, there are N components shifted from the anchor component by a single weight vector, $\mathbf{w}_{*j}$. We name them first order components. The N first order components correspond to the N cases where only one hidden unit takes the value one. Following this formulation the i^{th} order components represents all possible combinations of hidden states where exactly i units take the value one. The components are therefore shifted by the sum over the weight vectors of the active units.

This formulation allows us to give a clear illustration of the marginal PDF for GB-RBM. Figure 16 (a) and (b) show the experts of a GB-RBM with two visible and two hidden units as a sum of two Gaussians each. Figure 16 (c) shows the marginal PDF of a the model as the product of the experts, which leads to a total number of $2^2 = 4$ components. Regarding the previous discussion, the anchor component is only shifted by the visible bias $\mathbf{b}$. The first order components are shifted by $\mathbf{b} + \mathbf{w}_{*1}$ and $\mathbf{b} + \mathbf{w}_{*2}$, respectively. The second order component, which is the highest order component for a GB-RBM with two hidden units, is shifted by $\mathbf{b} + \mathbf{w}_{*1} + \mathbf{w}_{*2}$.

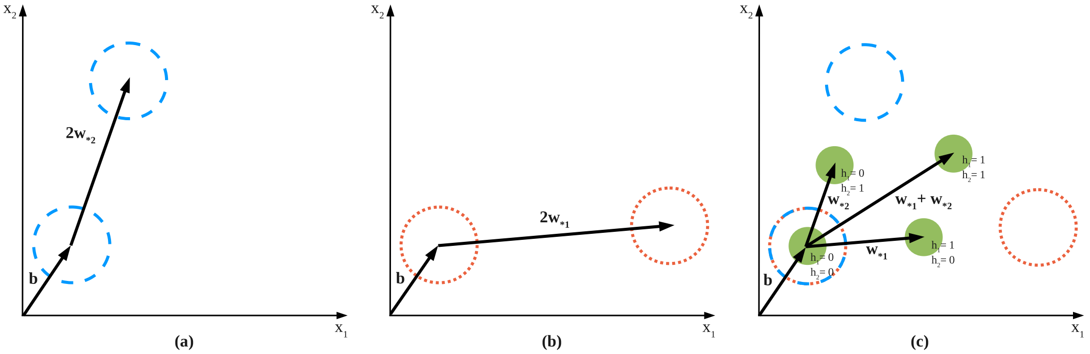


Figure 16: Illustration of a GB-RBM (with two visible and two hidden units) as a PoE and a MoG model. The arrows indicate the visible bias vector and the weight vectors, the circles denote Gaussian distributions. (a) and (b) visualize the two experts of the model. (c) visualizes the components in the GB-RBM denoted by the filled green circles. The four components are the results of the product of the two experts, which leads to the components placed right between two dotted circles.

A major disadvantage of GB-RBM is obviously that only the anchor and the first order components are independent, i.e. they can be placed freely in data space. The positions of the i^{th} order components are just the combination of the i first order components. This forces the 2^N components to lie on the vertices of a parallelepiped, which is a projected N-dimensional hypercube.

Furthermore, the scaling depends on the components position, except for the anchor component, which is fixed relative to the other components. Only the first order components can be scaled freely by choosing the corresponding hidden bias $\mathbf{c}$. The scaling of the higher order components are determined by their position and on the hidden biases of the first order components.

These restrictions limit the kind of distributions GB-RBMs can model. If the data as usual, is not distributed so that data clusters are located on the vertices of a projected hypercube, components will be placed in regions where no or less data is present. Figure 17 shows a two dimensional example where the data is distributed like a parallelepiped on the left and not distributed like a parallelepiped on the right.

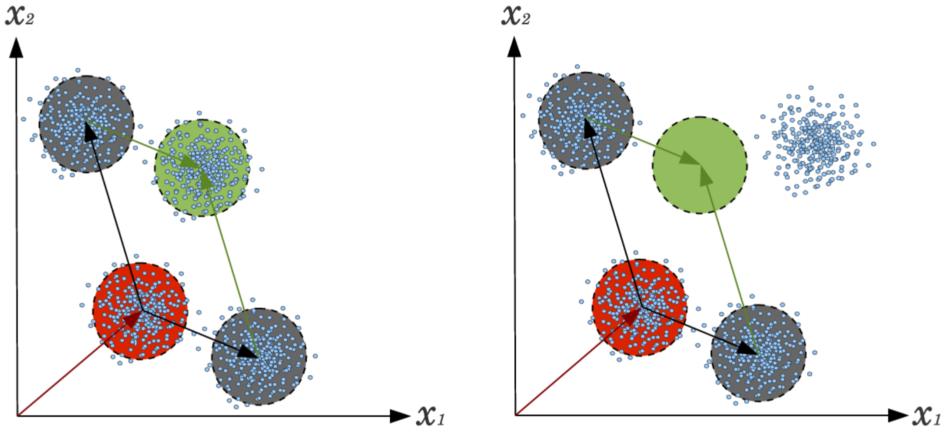


Figure 17: 2D example where the data is distributed (left) like a parallelepiped and (right) not like a parallelepiped where one component is position in an area without data.

If components are placed in non data regions, they need to be scaled down so that their probability gets very small. Due to the definition of the mixing coefficients we are only able to scale them down by reducing both hidden biases c_1 and c_2, which will also affect the scaling of the first order components. Accordingly, the model usually uses mainly the anchor and lower order components to model the distribution if the data and component variances have a comparable size. Since the lower order

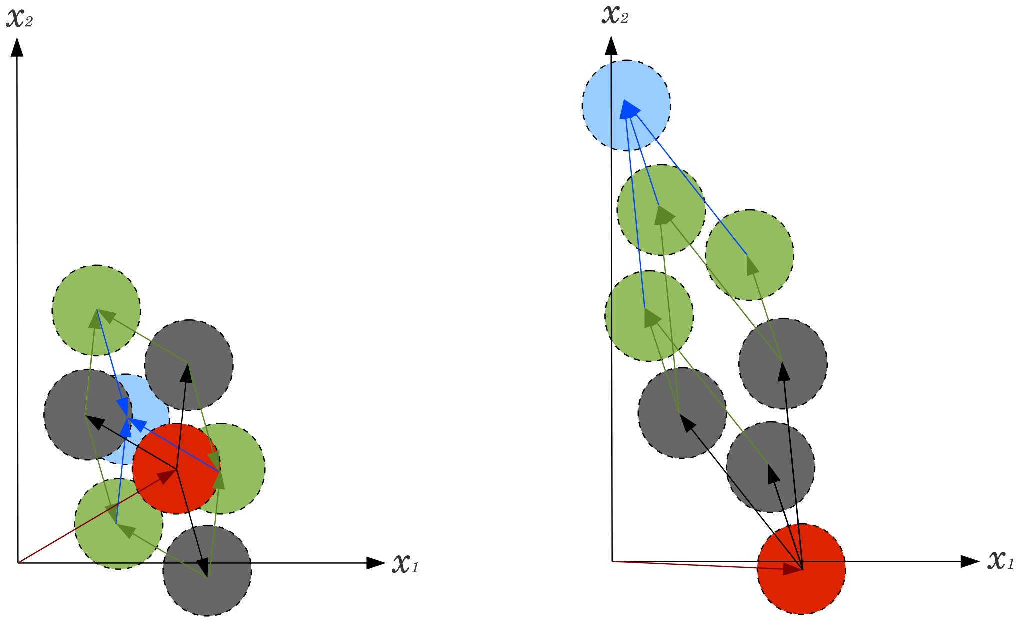


Figure 18: 2D example where (left) the visible bias is positioned centrally and (right) positioned peripheral, which causes the higher order components to be positioned far outside. The anchor component is given in red, the first order components in green and second order component in blue.

components correspond to just a few hidden units being active, this leads naturally to a sparse representation of the data.

In the previous discussion we implied the variance of the Gaussians having a meaningful size. It is worth mentioning that if the variance is to big, the best GB-RBMs can do, is to place all Gaussians in the mean of the data, which is equivalent to having one single Gaussian. If the variance is too small we need a lot of free components to model the PDF. Therefore, the variance plays an important role.

The position of the anchor component, relative to the first order components plays also an important role since it defines the projection direction of the hypercube. If the bias is located in the center of the other components the hypercube is projected from a top view. If the visible bias is located peripheral, then the projection will be stretched in the direction towards the mean. This has the effect that the higher order components will be placed far away from the other components. Figure 18 shows a two dimensional example where the visible bias has a central position on the left and the same model where the visible bias is switched with one of the first order component on the right. For uni model distributions this forces the visible bias to move to the data's mean. Summarizing, a GB-RBM is extremely limited in the class of distributions it can represent, compared to a mixture model.

4.2 Connection to Mixtures of Gaussians

A mixture model [5] defines a PDF over input space $\mathbf{x}$ using M components. In contrast to PoE where the components, named experts are combined by multiplication, a mixture model combines the components additively by a weighted sum. Each component itself needs to be a normalized PDF, so that the model becomes a valid PDF if we guarantee that the sum of the weights is one. A mixture model is defined as:

$$P^{MM}\left(\mathbf{x}\right) = \sum_{j}^{M} \eta_j \, \phi_j\left(\mathbf{x}\right) \,, \tag{244}$$

with the necessary normalization conditions,

$$\sum_{j=1}^{N} \eta_j = 1 \,, \tag{245}$$

$$\int \phi_j\left(\tilde{\mathbf{x}}\right) \, \mathrm{d}\tilde{\mathbf{x}} = 1 \,, \, \forall \phi_j \,. \tag{246}$$

The most popular choice for the mixture functions are normal distributions with mean $\boldsymbol{\mu}_j$ and covariance matrix $\mathbf{\Sigma}_j$. The mixture model is then called a Mixture of Gaussians (MoG) given by:

$$P^{MG}\left(\mathbf{x}\right) = \sum_{j}^{M} \eta_j \, \mathcal{N}\left(\mathbf{x}|\boldsymbol{\mu}_j \,, \, \mathbf{\Sigma}_j\right) \,, \tag{247}$$

with the necessary normalization condition,

$$\sum_{j=1}^{N} \eta_j = 1 \,. \tag{248}$$

From (243) we see that a GB-RBM is a restricted MoGs with 2^M components that share the same diagonal covariance matrix. The restrictions are given by the depending means and the scaling factors, which force the components to be located on

the corners of a parallelepiped as shown in Figure 16.
Mixture models are usually trained using the expectation maximization algorithm [5]. The algorithm is divided into two steps the "Expectation", where responsibilities of components for the data are evaluated and "Maximization", where the responsibilities are used to adapt the parameters, accordingly.
Due to the curse of dimensionality, it is impossible to use EM for training an RBM with 2^M components except for trivial cases. Even in the trivial cases we would need to adapt the algorithm to ensure the complex constraints, which makes it unattractive compared to CD where the constraints are ensured automatically.

The major advantage of an MoG compared to the GB-RBM is that we are able to adapted the covariance matrices freely, while in GB-RBM they all have the same diagonal covariance matrix. Consequently, GB-RBM are quite limited in modelling covariances.

4.3 Principal Component Analysis for Whitening Data

Whitened data has zero mean and unit variance in all directions. Accordingly, the whitening procedure removes the first and second order statistics from the data, which helps algorithms like ICA to learn higher order statistics of the data.
Since the components of GB-RBMs share the same diagonal covariance matrix, a single component is not able to learn the covariances in the data. The only opportunity would be an approximation using several mostly first order components to compensate this effect, which is inefficient. Therefore, whitened data seems to be more suitable for GB-RBM if we want to concentrate on learning the higher order statistics.

Whitening is usually performed using Principal Component Analysis (PCA). PCA aims to find an orthogonal transformation, which transfers the data variables into a set of linearly uncorrelated variables named Principal Components (PC). Since two variables are uncorrelated if their covariance is zero, the transformed data needs to have a diagonal covariance matrix. Consequently, the problem of PCA reduces to the diagonalization of the covariance matrix, which is always possible for symmetric matrices like covariance matrices $\mathbf{\Sigma}$ given by:

$$\begin{aligned} & \mathbf{V}^{-1}\mathbf{\Sigma}\mathbf{V} && = \boldsymbol{\lambda}\mathbf{I}\,, && (249) \\ \Leftrightarrow\quad & \mathbf{\Sigma}\mathbf{V} && = \boldsymbol{\lambda}\mathbf{I}\mathbf{V}\,, && (250) \end{aligned}$$

which is the characteristic polynomial of $\mathbf{\Sigma}$ with eigenvectors $\mathbf{V}$, eigenvalues $\boldsymbol{\lambda}$ and $\mathbf{I}$ denotes the identity matrix. Consequently, we can multiply the mean free data with $\mathbf{V}$, so that it gets a diagonal covariance matrix.

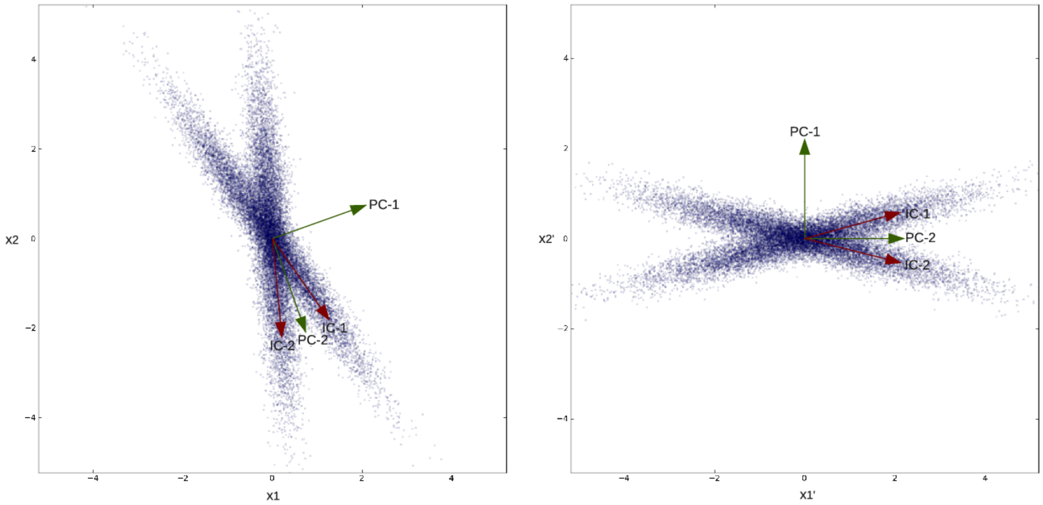


Figure 19: (left) Scatter plot of an example distribution before the PCA transformation is applied. The PCs are shown in green and for comparison the ICs are shown in red. (right) The same data after the PCA transformation has been applied, which rotates the PCs on the coordinate axis.

A much more intuitive motivation comes from the fact that one of the eigenvectors point in the directions of maximum variance. Another eigenvector points in the direction of the maximum remaining variance, under the restriction to be orthogonal to the first one. The third eigenvector points in the direction of the maximum remaining variance, under the restriction to be orthogonal to the first and second one, and so on. This allows to select only the first M components for dimensionality reduction, which preserve as much variance of the data as possible.
Figure 19 shows an example data distribution with zero mean before PCA on the left and after PCA on the right. The green arrows show the PCs, which are parallel to the coordinate axis after PCA transformation. Additionally, the ICs, which point in the most independent directions, as described in Chapter 2, are shown in red to highlight the difference between both methods. ICA is able to recognize the two statistically independent sources while PCA is limited to the variance of the data.
It is easy to see that we achieve unit variance if our already diagonal covariance matrix becomes the identity matrix. This is done by dividing the data by its standard

deviation, which is given by the square root of the eigenvalues.

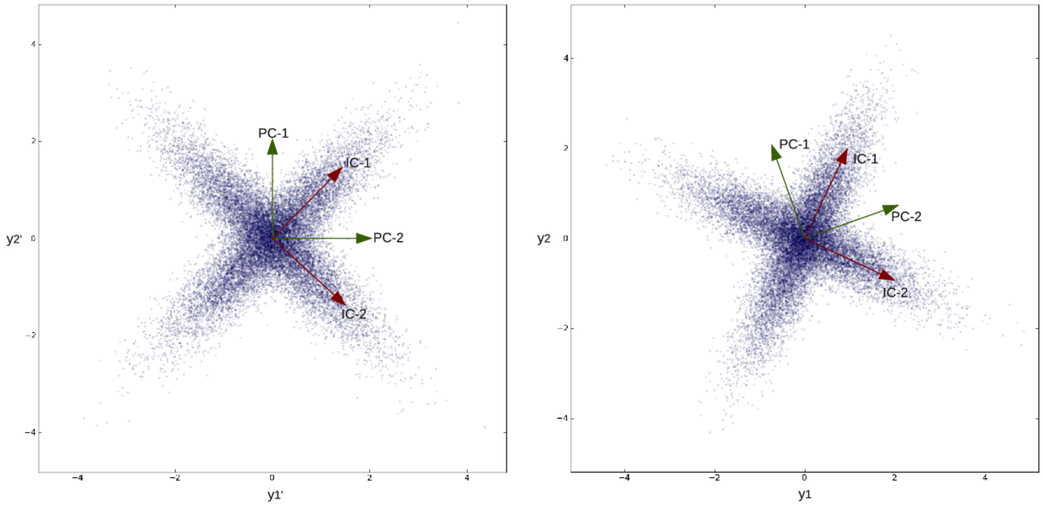


Figure 20: (left) Scatter plot of an example distribution after the PCA transformation and whitening. The PCs are shown in green and for comparison the ICs are shown in red. (right) The same data after applying the inverse PCA transformation, which leads to ZCA whitened data. Note that the shown PCs belong to the original space, since in whitened space all directions have unit variance and therefore no direction of highest variance exist.

Therefore, the whitening procedure becomes:

$$\mathbf{y} = \left(\frac{1}{\sqrt{\boldsymbol{\lambda}}} \mathbf{I} \right) \mathbf{V}^T \mathbf{x} \,. \tag{251}$$

Figure 20 shows the whitened version of the data shown in Figure 19 on the left and the result when rotating the whitened data back to the original space on the right, which is known as Zero Phase Component Analysis (ZCA). It shows that the IC become orthogonal in the whitened space, so that the problem of ICA reduces to finding a rotation matrix.

4.4 Connection to Independent Component Analysis

In Chapter 2 we introduced the concept of statistical independence and independent components (IC). In Independent Components Analysis (ICA) we assume the data

$\mathbf{x}$ being a linear combination of M independent sources $\mathbf{s}$. The joint probability of $\mathbf{s}$ can then be expressed as:

$$P(\mathbf{s}) = \prod_{j}^{M} p_j(s_j), \tag{252}$$

and their linear combinations is given by,

$$\mathbf{x} = \mathbf{A}\mathbf{s}. \tag{253}$$

In the complete case, where the number of input dimensions equals the number of output dimensions, $M = N$ we can derive the probability distribution of the inputs $\mathbf{x}$ as:

$$\begin{aligned} P(\mathbf{x}) &= \prod_{j}^{N} p_j(s_j), & (254) \\ &= |\det \mathbf{W}| \prod_{j}^{M} p_j\left(\mathbf{w}_{*j}^T \mathbf{x}\right), & (255) \end{aligned}$$

where $\mathbf{W} = \mathbf{A}^{-1}$, and the p_j denote the unknown densities of the independent components. The aim is to find $\mathbf{W}$ to recover the statistically independent sources $\mathbf{s}$ from the input data.

Obviously ICA and GB-RBM belong to the PoE [18] model, which have been addressed by [39] and they will become equivalent if we choose $p_j(s_j)$ to be the sum of two Gaussians as given in (243).
But the success of ICA highly depends on the choice of the prior distribution. Since in ICA we are looking for the directions of most Non-Gaussianity, the prior distributions for the experts will be chosen as super-Gaussian or sub-Gaussian. While in GB-RBMs we have a weighted sum of two Gaussians with the same variance, which are Gaussians or sub-Gaussians.
Furthermore, the posterior distribution in ICA over the sources, are assumed to be marginally independent denoted by $s_i \perp\!\!\!\perp s_j \ \forall i \neq j$. This is not the case in GB-RBMs, which only assumes the visible variables to be conditionally independent of the hidden variables and vice versa, denoted by $x_i \perp\!\!\!\perp x_k \,|\, \mathbf{h}\,,\ \forall i \neq k$ and $h_j \perp\!\!\!\perp h_k \,|\, \mathbf{x}\,,\ \forall j \neq k$, respectively.

ICA can be trained by maximizing the LL defined for a single data point by:

$$\begin{aligned}\mathcal{L}^{ICA}\left(\mathbf{x}\,|\,\boldsymbol{\theta}\right) &= \mathcal{L}^{ICA}\left(\mathbf{x}\,|\,\mathbf{W}\right)\,, &(256)\\ &= \ln|\det\mathbf{W}| + \sum_{j}^{M}\ln p_j\left(\mathbf{w}_{*j}^T\mathbf{x}\right)\,, &(257)\end{aligned}$$

There exist various ICA algorithm based on different principles. Mainly all of them work on the whitened data so that the problem, as mentioned, simplifies to the search of a rotation matrix W, that makes the variables $\mathbf{x}$ most statistical independent. This work uses the popular Fast-ICA algorithm, presented in [1].
An advisable literature for ICA and its applications is given by [22].

5 Experiments

We have seen that GB-RBMs are quite limited in their representational power, so that it is rather unsure if they are a good model for natural images. In addition to the limitations, different authors [6, 24, 43] reported that GB-RBMs are difficult to train.

This chapter describes the experiments that were made in order to analyse how GB-RBMs model natural images and why the successful training highly depends on the choice of the hyperparameters. Initially, the dataset is described and it is shown that the preprocessing of the data is very important. According to the relation of GB-RBM to ICA and MoG, described in the previous chapter, the results of GB-RBMs are compared to the results of both models. Moreover, it will be shown that the variance plays an important role and how the number of hidden units affect the model. Finally, the training methods and the choice of the hyperparameters are compared.

5.1 The Natural Image Dataset

The Van Hateren's Natural Image Database[1] is a common choice when working with natural images. The experiments were done using this dataset although the results were similar when using photographs of arbitrary scenes. An example image is shown in Figure 21.

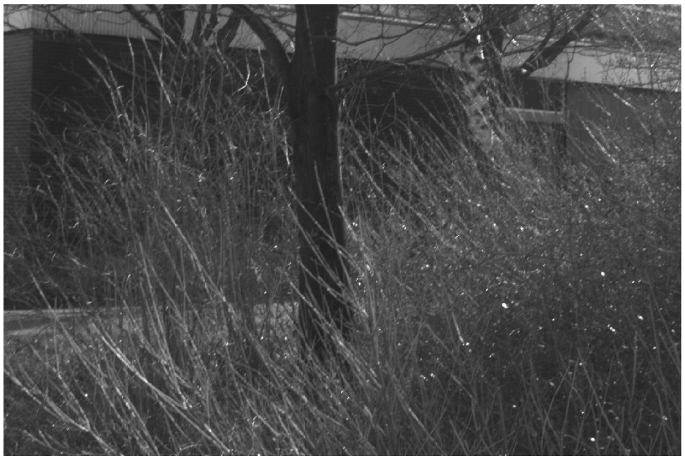


Figure 21: An image from the Van Hateren's Natural Image database.

[1] http://www.kyb.tuebingen.mpg.de/?id=227

As described in Chapter 2, we want to model simple cells, which are focused on the same subregion of the input signal. The input for those cells can in principle be any small patch from any natural image. We therefore sampled randomly 70,000 image patches of size 14 times 14 pixel from the images of the database and divided it into 50,000 samples for training and 20,000 samples for testing. Each image was normalized separately to have zero mean in order to compensate different lighting conditions. Due to the random sampling, the variances of the pixel intensities per dimension were approximately the same, with an average variance of 383.86 and a standard deviation of 43.76. To avoid numerical problems the dataset was rescaled by dividing each pixel intensities by a factor of 10.
Figure 22 shows 80 image patches on the left, their zero mean version in the middle and the ZCA whitened version on the right.

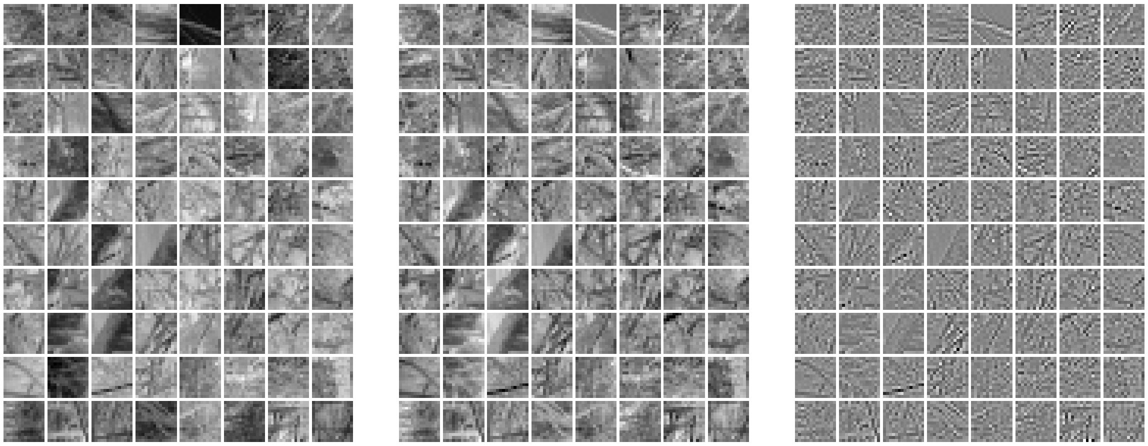


Figure 22: (left) Some images patches of size 14x14 pixels sampled from the Van Hateren's Natural Image Database, (middle) the corresponding zero mean version and (right) the corresponding whitened version.

Since it is not possible to visualize a 196 dimensional PDF, we are mainly limited to inspecting the weights of the GB-RBM to get an idea of how the model's PDF is structured. In Chapter 2 we mentioned that ICs of natural image patches are sparsely distributed. Accordingly, a linear mixture of two sparse distributions, like Laplacians should function as a two dimensional representative distribution. This is of course a very rough approximation, but it should help to understand how a GB-RBM models natural image patches.

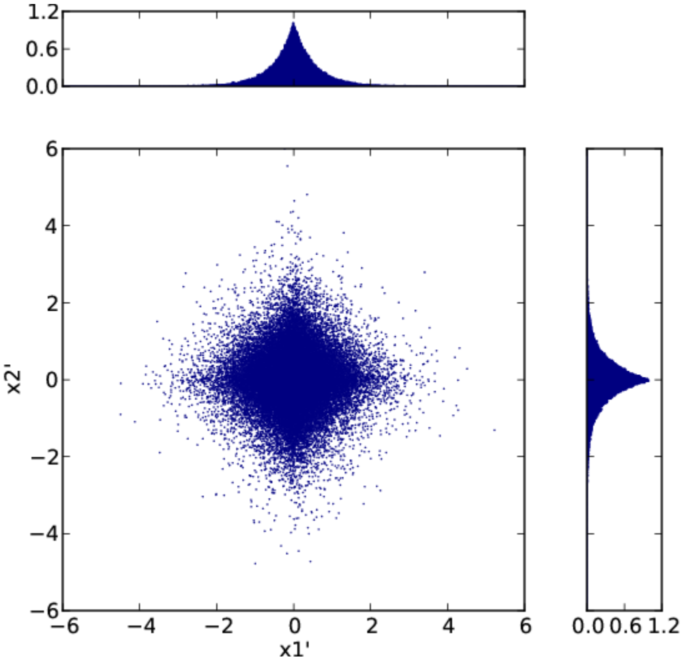


Figure 23: Showing data from two independent Laplacian distributions.

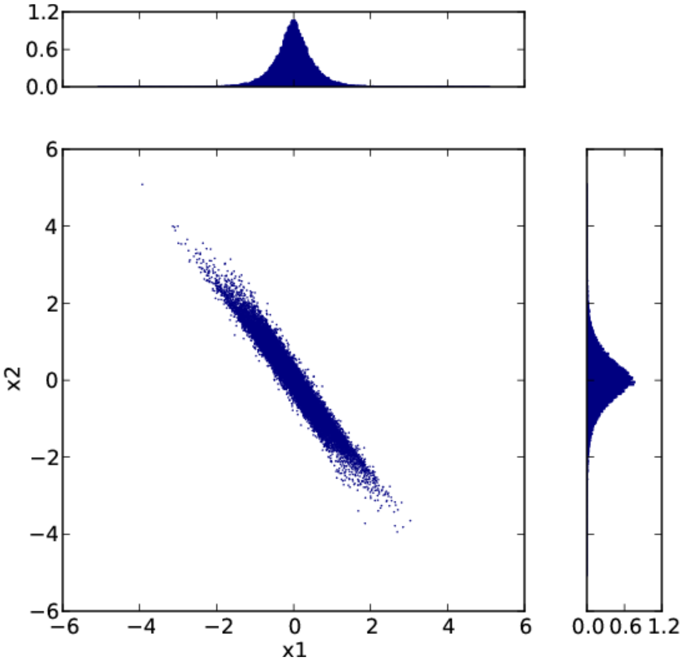


Figure 24: Showing data from a random mixture of two independent Laplacian distributions.

We sampled 70,000 two dimensional data points, 50,000 for training and 20,000 for testing, from two independent Laplacian distributions, shown in Figure 23. The two Laplacian density distributions are shown as histograms beside the axis. Figure 24 shows the same data after mixing it with a random matrix. Now one of the marginal distributions looks much more Gaussian than Laplacian. Figure 25 shows the same data after whitening and both marginal distributions look much like Gaussians now.

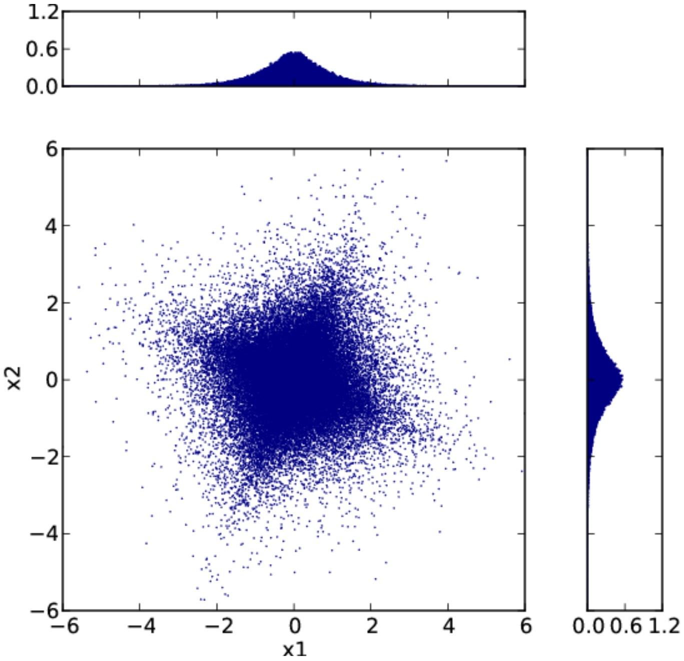


Figure 25: Showing whitened data from a random mixture of two independent Laplacian distributions.

This two dimensional dataset was used in addition to the natural image dataset to illustrate how GB-RBMs model a mixture of sparse distributions.
The LL for the data given the model can also be used as a measurement for the model's performance. But it is important to note that it only measures how well the model fits the data PDF, in terms of the Kullback-Leibler Divergence and this does not deduce that the model learned any structured filters.

5.2 Independent Component Analysis on Natural Images

As a well studied and plausible model for natural images, ICA represents a reference model [21] for natural image statistics. This means, if a model is trained on natural images, but does not learn localized, orientated and frequency selective (LOFS) filters like ICA, we assume that the model learned a worse representation.
Figure 26 shows the ICs learned from the natural image patches, which are the reshaped columns of the ICA mixing matrix and will also be denoted as filters. The LL for the training data was -259.0859 and for the test data -259.4393.

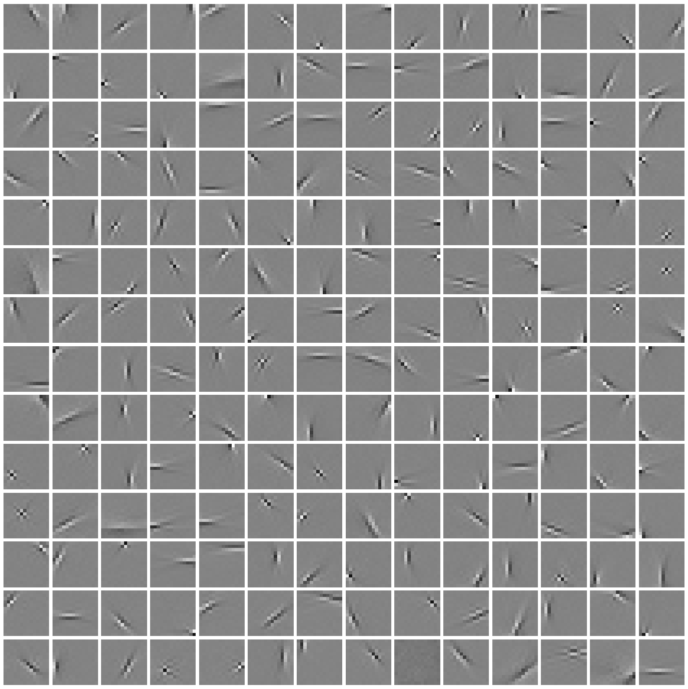


Figure 26: The 196 ICs of the natural image dataset learned by FastICA. Each patch is a reshaped column of the ICA mixing matrix. The LL for the training data was -259.0859 and for the test data set -259.4393

The ICA result for the 2D data is shown in Figure 27 on the right and the randomly initialized configuration before training, on the left. The red lines indicate the two ICs and the blue dots represent the training data points. The LL for the randomly initialized model was -2.8015 for the training and -2.8028 for the test data. After training the LL was slightly better, -2.7428 for the training and -2.7423 for the test data.

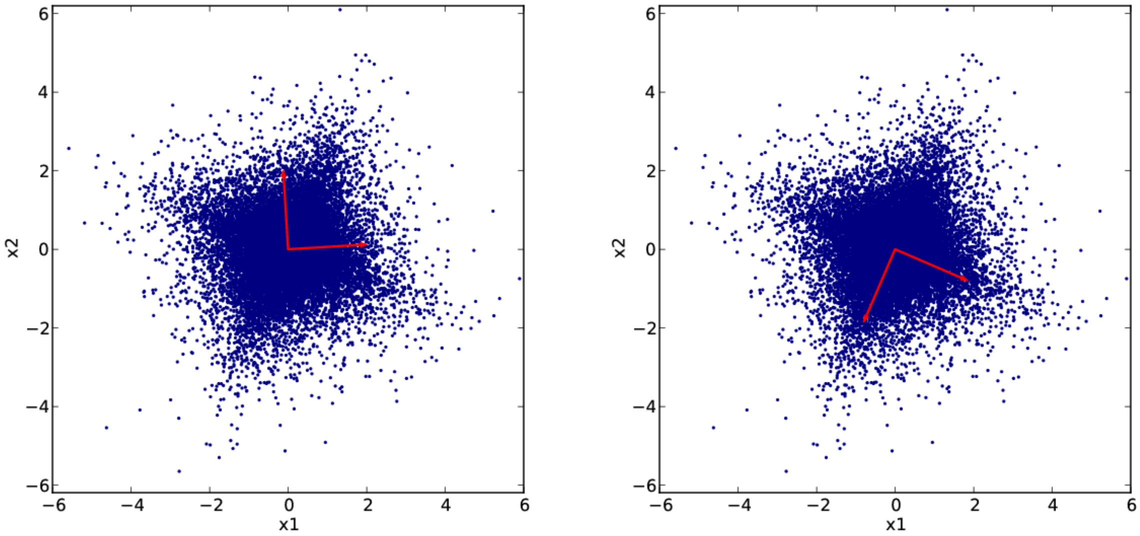


Figure 27: Scatter plot of the 2D dataset, (left) before training and (right) after training, where the red lines are the columns of the ICA mixing matrix. The LL before training was -2.8015 for the training data and -2.8028 for the test data set and after training -2.7428 and -2.7423, respectively.

These results were used as a baseline for interpreting the results of GB-RBM in the following experiments.

5.3 Training Gaussian-Binary RBMs on differently Preprocessed Natural Images

From the theoretical analysis we know that GB-RBM are quite limited in the way they can represent data.

The experiments described in the following, compared GB-RBMs trained on the natural images with differently preprocessed data. For all experiments, GB-RBMs with 196 visible and 196 hidden units were trained using the same setup[2].

In the first experiment, we trained a GB-RBM on the natural image dataset without any preprocessing. The learned filters, which are the columns of the RBM weight matrix are shown in Figure 28. They were ordered by their probability of being active under the training data, in descending columnwise order, from the top left to

[2] Training setup: LL average of 5 trials, 50,000 image patches, 300 epochs, CD-1, batch size 100, learning rate 0.01 Momentum 0.9, Weight decay 0.0, variance fixed to the variance of training data, but comparable results were achieved when training the variance.

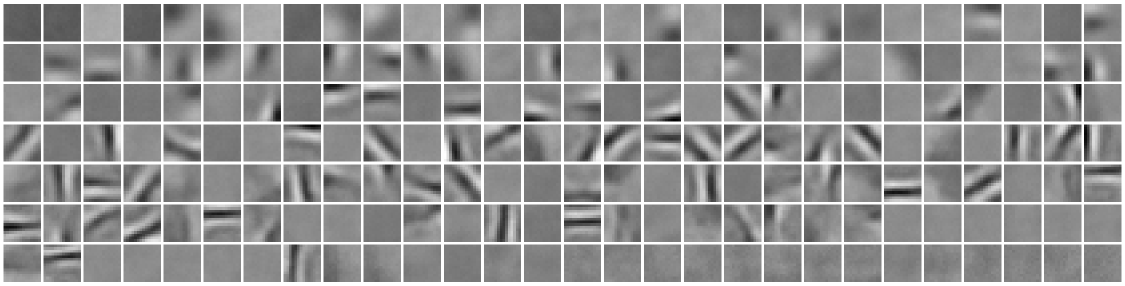


Figure 28: Filters of a GB-RBM trained on the natural image dataset without any preprocessing. The filters were sorted descending from the left to the right, from the top to the bottom, by their average activation probability.

the bottom right. The filters of the first two rows show low frequency filters, they are almost uniform or show a smooth change from light to dark. The following rows have still uniform filters, but a lot more show LOFS structures. Like in ICA we got dot-like and bar-like LOFS filters, but with a lower frequency. Figure 31 (a) shows that the activation was decreasing exponentially from the first to the last filter, so that the filters of the first three rows represented more than 90% of the total activation.

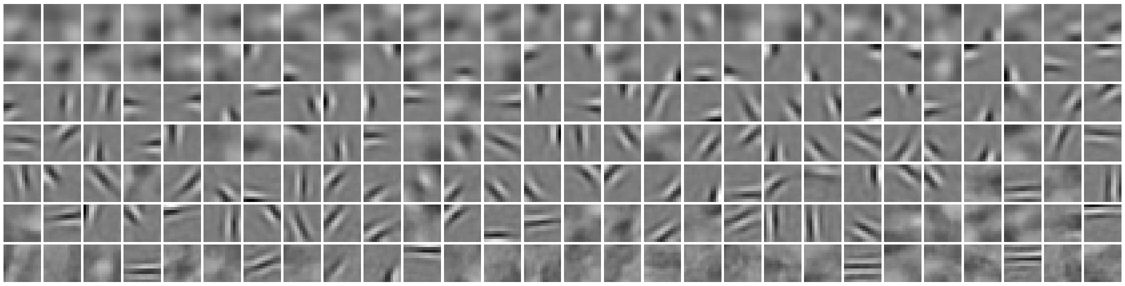


Figure 29: Filters of a GB-RBM trained on the natural image dataset, where the mean has been removed for each image patch separately. The filters were sorted descending from the left to the right, from the top to the bottom by their average activation probability.

We assume that the use of zero mean images helps to focus on the structures if the lighting conditions vary a lot. This becomes clearer if we think of images that show similar structures under different illumination conditions, which mainly shifts the mean. Assuming that we have filters to model the structure, we would need

additional filters to model the different means. So if the image mean is removed, we would not expect uniform filters anymore.
Figure 29 shows the filters learned form the natural image dataset with zero mean images, again ordered descendingly by their activation. Comparing these filters to the filters for the non zero mean images, the first filters are still very smooth, but show patterns of dark and light spots. Most obviously the total amount of LOFS filters and their frequency increased. Figure 31 (b) shows that the activation for these filters were still exponentially decreasing, but the activation of the first filter was distributed a bit more equally.

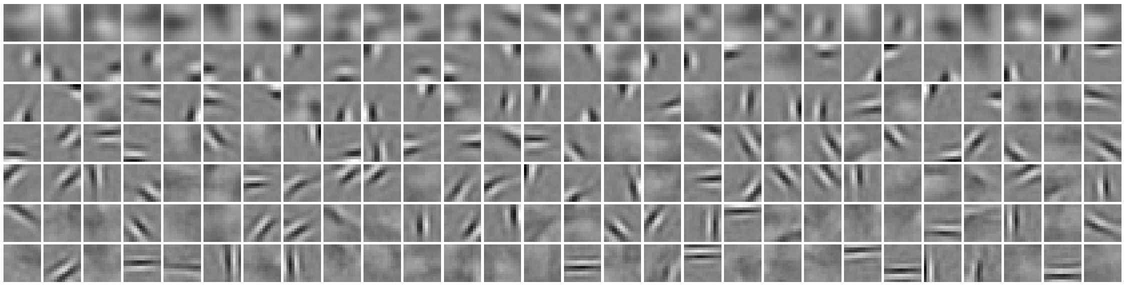
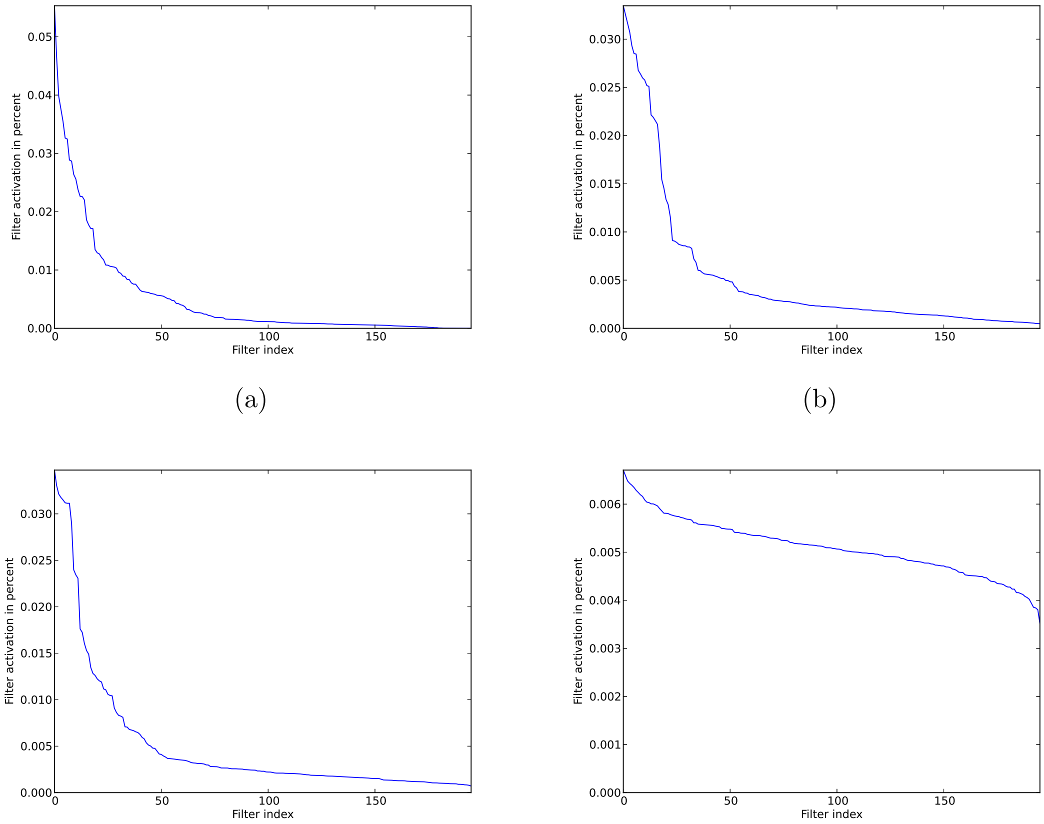


Figure 30: Filters of a GB-RBM trained on the natural image dataset, where the mean has been removed for each image patch separately and the dataset has been normalized such that each pixel dimension has zero mean and unit variance. The filters were sorted descending from the left to the right, from the top to the bottom by their average activation probability.

Before we consider the whitened data, we have a look to the results when the data was normalized so that each pixel dimension had zero mean and unit variance. Figure 30 shows the filters learned form the normalized natural image dataset with zero mean images, ordered descendingly by their activation. Comparing the filters with the filters of the unnormalized version, shown in Figure 29, they look almost the same given in a different order. Figure 31 (c) shows the activation distribution of the filters, which is comparable to the unnormalized version, shown in Figure 31 (b). Since the variances along the pixel dimensions of the original data where already quite similar, normalizing the data had basically not effect on the resulting filters.

Finally, we have a look to filters of a GB-RBM trained on the whitened natural image dataset shown in Figure 32. The filters had the same LOFS structure as the filters of ICA, shown in Figure 26. Therefore, ICA and GB-RBM learned a similar structure, but GB-RBMs use scaled Gaussian distributions and ICA uses Laplacian



(a)

(b)

(c)

(d)

Figure 31: Showing the four filter activation distributions with the filter index on the x-axis and the percentage activation over the whole training data on the y-axis. (a) Unmodified dataset, (b) zero mean image dataset, (c) normalized zero mean image dataset, (d) whitened zero mean image dataset.

distributions to model the PDF. Figure 31 (d) shows that the activation of the filters were more homogeneous than exponentially distributed.

To see how GB-RBMs model data, we can reconstruct an image patch by one step of Gibbs sampling and compare how similar it looks to the originally presented image. Figure 33 consists of four images, each showing 28 natural images in the first row and the corresponding one step Gibbs sampling reconstruction in the second row. The first image (a) belongs to the unmodified dataset and the learned filters

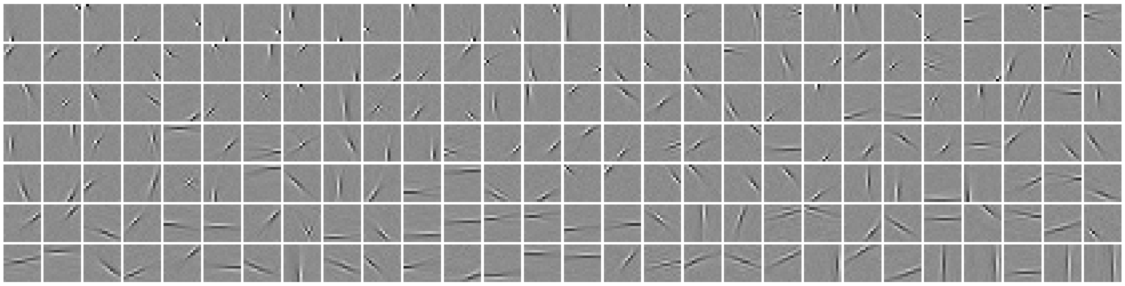
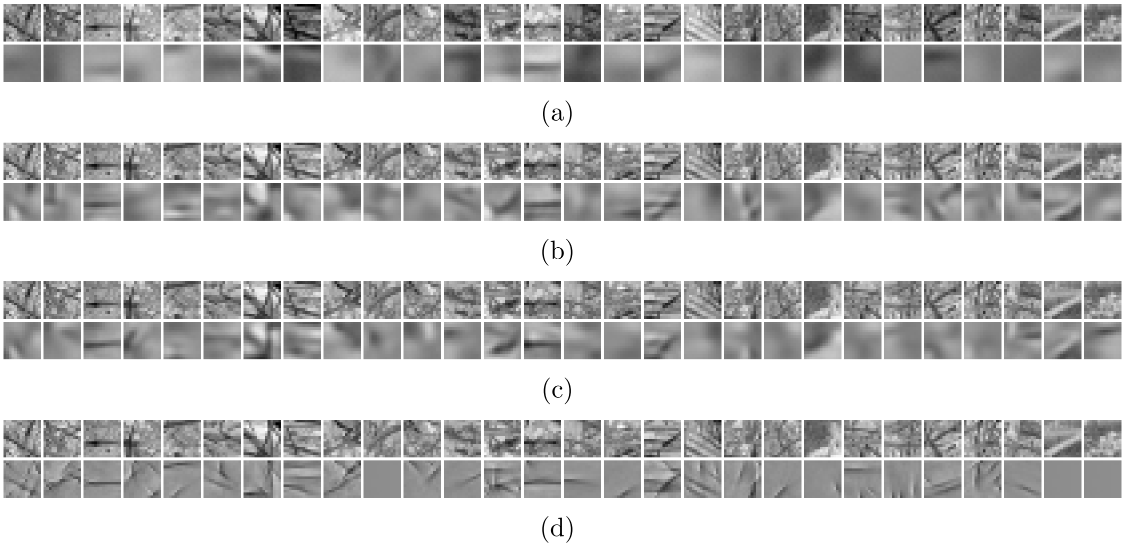


Figure 32: Filters of a GB-RBM trained on the natural image dataset where the mean has been removed for each image patch separately and the dataset has been whitened to have zero mean and unit variance in all directions. The filters were sorted descenting from the left to the right, from the top to the bottom by their average activation probability.

shown in Figure 28. The GB-RBM reconstructed mainly the contrast of the image and only little of the structure, which corresponds to the uniform and low frequency filters. Image (b) belongs to the GB-RBM trained on the zero mean images and the reconstructions showed more structures of the original images, but as a blurred version. The blurring removed most of the detail structure of the images. Image (c) corresponds to the GB-RBM trained on the normalized zero mean images. Since the filters of the unnormalized and normalized version were very similar, the reconstructions were also very similar. Image (d) belongs to the GB-RBM trained on the whitened zero mean images. It shows the de-whitened images and de-whitened reconstructions, which were not blurred and showed more detailed structures. It seemed that this GB-RBM reconstructed only the most important edges and failed to reconstruct large, mostly homogeneous regions.

Comparing the LL of the models is not straightforward since modifying the data space leads also to a change of the probability distribution. But we can transform the probabilities for the transformed datasets back, by multiplying the likelihood with the determinant of the transformation matrix. For the LL we therefore added the logarithm of the determinant, which was -135.7842 for the normalization matrix and 152.7298 for the whitening matrix. Notice, that the normalization has been applied also before the whitening process. Table 1 shows the average LL and the transformed average LL for the differently preprocessed data. The LL of the whitened version was the best compared to the other GB-RBMs, but worse than ICA.
The following experiments were done using the whitened data, since we were interested in learning high frequency filters, which are comparable to the ICA results.



(a)

(b)

(c)

(d)

Figure 33: Each image shows 28 randomly selected images in the first row and the reconstruction after one step of Gibbs sampling of the corresponding GB-RBM in the second row. (a) Unmodified dataset, (b) zero mean image dataset, (c) normalized zero mean image dataset, (d) whitened zero mean image dataset, showing the dewhitened images and reconstructions.

Dataset	Model	LL Train	LL Test	Transformed LL Train	Transformed LL Test
unmodified	GB-RBM	-453,9998	-454,0010	-453,9998	-454,0010
zero mean images	GB-RBM	-358,8723	-358,9146	-358,8723	-358,9146
normalized	GB-RBM	-226.5642	-225.1643	-362.3484	-360.9485
whitened	GB-RBM	-270,5140	-270,0225	-253,5684	-253,0769
whitened	ICA	-259,0859	-259,4393	-242,1403	-242,4937

Table 1: Showing the average LL and the LL transformed back to the zero mean image space for different datasets and models.

5.4 Learning the Variance of Gaussian-Binary RBMs

In the previous experiments the variance parameters of the model were fixed to the data variances. The following experiments were done to illustrate the effect of training the variance parameters. Figure 34 shows the filters learned by a GB-RBM[3] with 196 visible and 196 hidden units. The average model's variance per dimension was 0.76129, with a standard deviation of 0.1687. The LL improved slightly from -270.5140 to -266.8235 in training and overfitted from -270.0225 to -272.1207 for the test data. The filters were more dot-like compared to the filters without training the variance, shown in Figure 32.

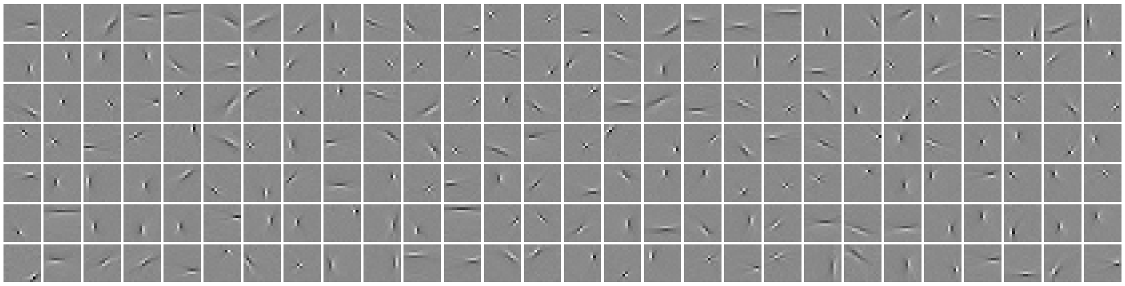
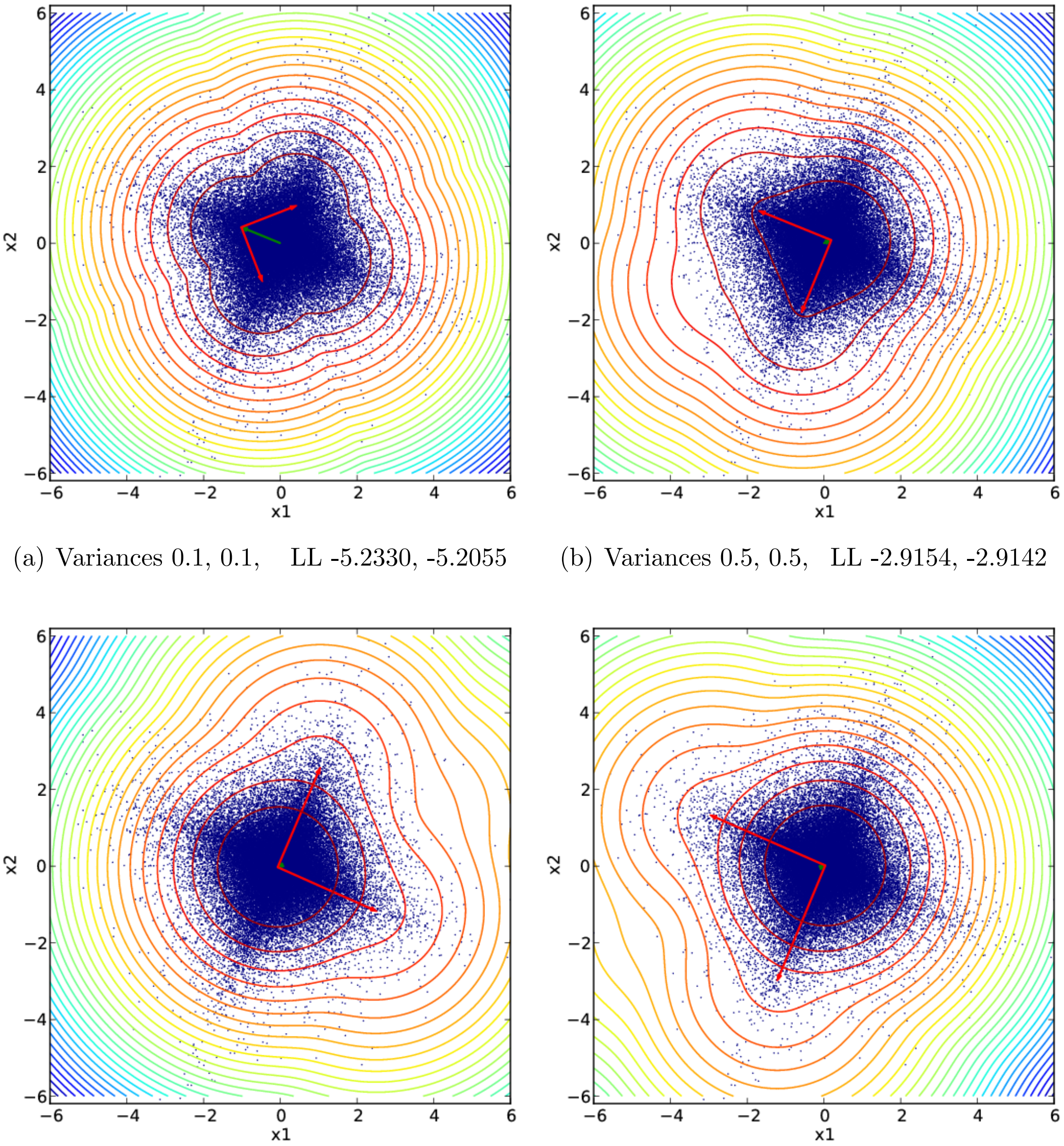


Figure 34: Filters of a GB-RBM with 196 hidden units and trained variances. The average variance per dimension was 0.7487, with a standard deviation of 0.2212. The average LL estimated by AIS was -266.8235 for the training data and -272.1207 for the test data.

To illustrate how different values for the variances affect the PDF, we trained[3] GB-RBMs with two hidden units and differently set variances on the 2D dataset. Since the higher order components are usually damped, their contributing density is not visible in a normal PDF plot, so that it is appropriate to show the logarithm of the PDF instead. Figure 35 and 36 show the log-PDF of the GB-RBMs for different variances. The green arrow represents the visible bias, the red arrows represent the weights and the blue dots are the data points. For very small variances like 0.1, all four components were equally scaled and arranged in a square on the data as shown in Figure 35 (a).

[3] Training setup: LL average of 5 trials,, 50,000 image patches, 300 epochs, CD-1, batch size 100, learning rate 0.01 (0.001 for the variance parameter) Momentum 0.9, Weight decay 0.0.



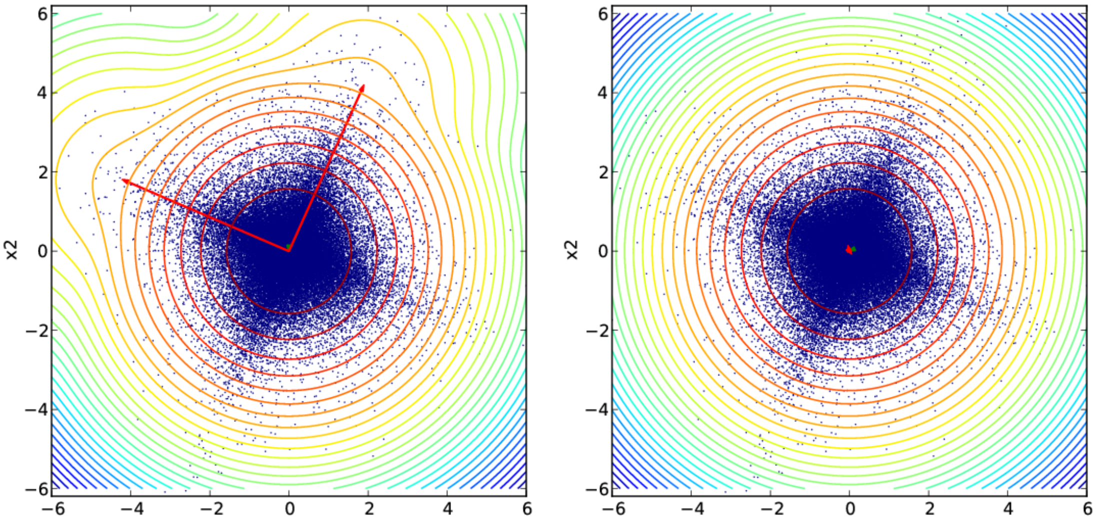
(a) Variances 0.1, 0.1, LL -5.2330, -5.2055

(b) Variances 0.5, 0.5, LL -2.9154, -2.9142

(c) Variances 0.86, 0.85, (Trained) LL -2.8065 , -2.8039

(d) Variances 1,0, 1.0, LL -2.8142, -2.8145

Figure 35: Contour plots for different variances of the GB-RBM's log-probability distributions. The GB-RBMs had two visible and two hidden units trained on the 2D dataset (blue dots). The green arrow represents the visible bias and the red arrows represent the weights.



(a) Variances 1.5, 1.5, LL -2.9070 , -2.9077 (b) Variances 2.0, 2.0, LL -3.0306 , -3.0297

Figure 36: Contour plots for different variances of the GB-RBM's log-probability distributions. The GB-RBMs had two visible and two hidden units trained on the 2D dataset (blue dots). The green arrow represents the visible bias and the red arrows represent the weights.

When the variance was increased, the configuration of the components changed. The anchor component was then placed in the mean of the data and the first order components were placed in the directions of the ICs. The first order components were extremely scaled down compared to the anchor component (The figures show the log-PDF). Consequently, the second order component was scaled down even more and placed in a region of less density between the two first order components.

While the variance was further increased, the norms of the weights increased and the components were further scaled down. This effect continued until the variance was much bigger than the data variance. Then the anchor component covered already most of the data and the best solution was to place all components in the mean, as shown in Figure 36 (b). Figure 35 (c) shows the resulting log-PDF when the variance was trained. The variance was approximately 0.85 for both dimensions with the best LL of -2.8065 for the training data and -2.8039 for the test data. Figure 37 shows the LL for different variances, which has a flat maximum at a value slightly smaller than one.

When sampling from the model using Eq. (177), it is worth mentioning that smaller variances reduces the effect of the sampling noise. Figure 38 shows the average

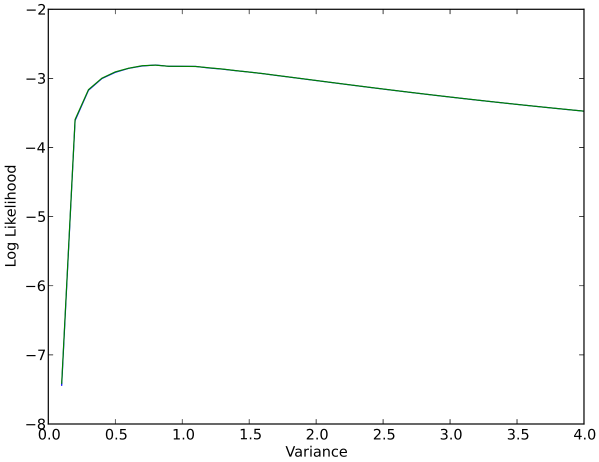


Figure 37: Average LL for GB-RBMs with two visible and two hidden units, trained on 2D data with different, fixed variance values. LL Training data (green), LL Test data (blue).

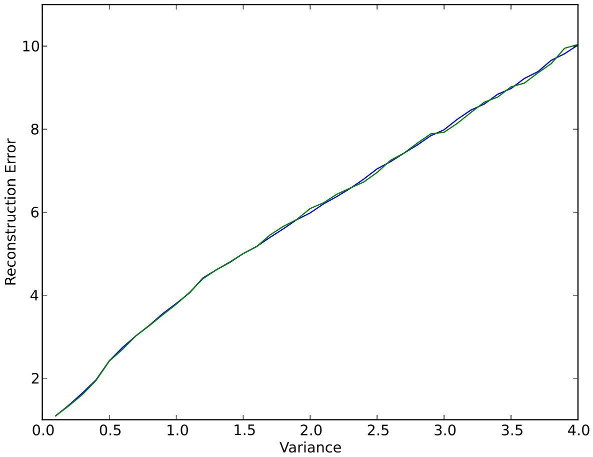
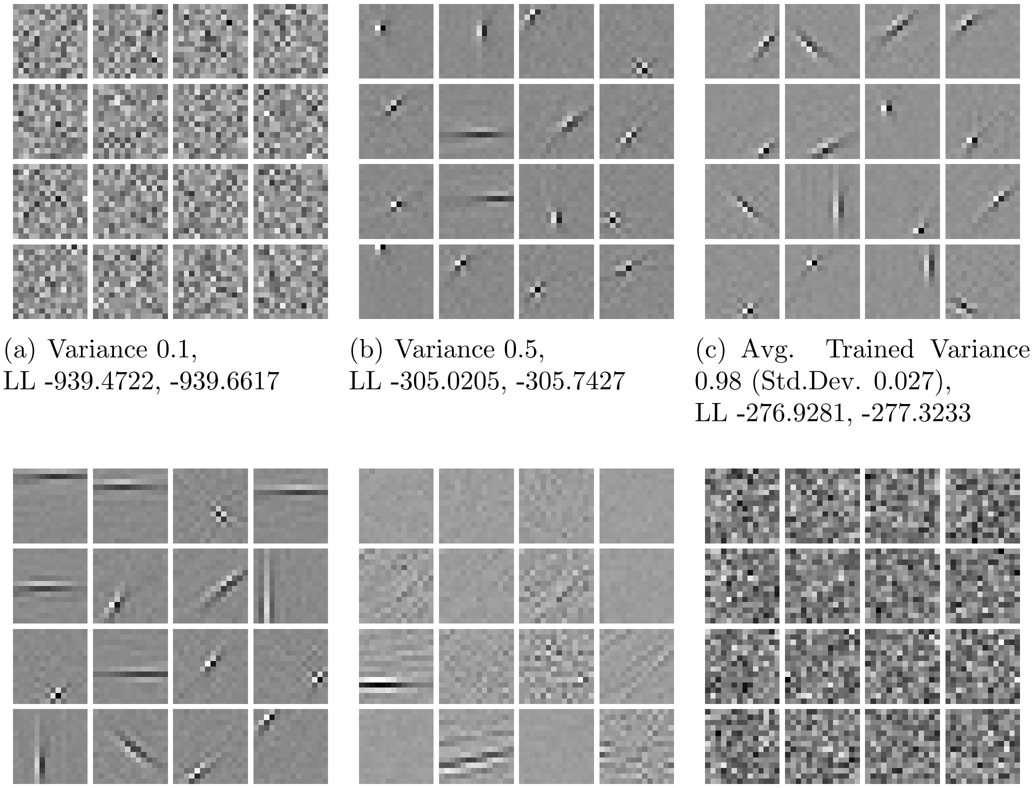


Figure 38: Average RE for GB-RBMs with two visible and two hidden units, trained on 2D data with different, fixed variance values. LL Training data (green), LL Test data (blue).

reconstruction error (RE), which increases linearly with the variance parameter as expected .

To verify whether the results for the 2D data are transferable to the natural images, we trained[4] GB-RBMs with 16 hidden units and differently set variances on the natural image data. The small number of hidden units allowed us to calculate the LL exactly. Figure 39 shows the filters learned for different variances. Comparable to the 2D data we did not learn filters for small and big variances.



(a) Variance 0.1,
LL -939.4722, -939.6617

(b) Variance 0.5,
LL -305.0205, -305.7427

(c) Avg. Trained Variance 0.98 (Std.Dev. 0.027),
LL -276.9281, -277.3233

(d) Variance 1,0,
LL -277.0113, -277.3918

(e) Variance 1.5,
LL -284.9037, -284.9027

(f) Variance 2.0,
LL -296.8294 -296.8279

Figure 39: Filters of GB-RBMs with 16 hidden units with different variances, trained on the natural image dataset. Note that all six images were normalized separately to highlight the filter's structure. The norm of the filters in (e) and (f) was small compared to (a)-(d), see Table 2.

Avg. Variance	Avg. Weight Norm	Std. Dev. Weight Norm	Visible Bias Norm	Avg. 1^{st} order scaling	Std. Dev. 1^{st} order scaling	Anchor Scaling
0.1 (a)	2.3800	0.0418	1.3539	4998.51	13851.62	3.61e-19
0.5 (b)	2.9282	0.1008	0.5901	0.05032	0.01828	0.50592
0.98 (c)	3.9979	0.1496	0.1417	0.01461	0.00160	0.79905
1.0 (d)	4.0041	0.1979	0.4527	0.01413	0.00249	0.80280
1.5 (e)	0.7474	0.1085	0.4046	0.00402	0.00048	0.93787
2.0 (f)	0.5409	0.0333	0.3829	0.00581	0.00097	0.91149

Table 2: Showing the average weight norms, visible bias norm, the anchor and first order scaling factors for GB-RBMs with 16 hidden units and different variances.

Considering additionally Table 2, it can be observed that the results are comparable to the 2D experiments. Except for very small variances, the visible bias was placed roughly in the mean and scaled down slightly more, as bigger the variance got. Comparably, the weights grew while the variance increased and converged to zero when a critical threshold was reached, although the convergence began already for a variance of 1.5. The first order components were scaled down more, as bigger the variance got. For variances of 0.1 the visible bias was scaled down extremely, but the first order components were scaled up. Consequently, the second order components were scaled up even more. It seems that it changed the role with the visible bias.

Figure 40 shows the relation of the LL and the variance. The graph is comparable to the 2D data graph shown in Figure 37, although the maximum is closer to one. Figure 39 (c) shows the learned filters when training the variance, with an average variance of 0.98. The optimal variance decreased with an increasing number of hidden units, as will be shown in the next experiment. Also similar to the 2D results, the reconstruction error decreased while the variance decreased, as shown in Figure 41.

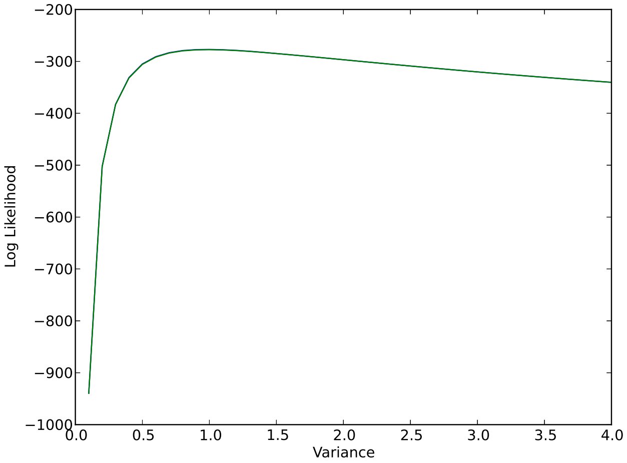


Figure 40: Average LL for GB-RBMs with 196 visible and 16 hidden units, trained on natural image data with different, fixed variance values. LL Training data (green), LL Test data (blue).

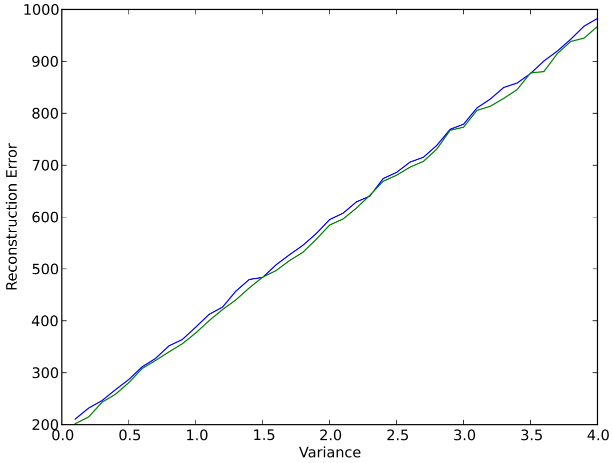
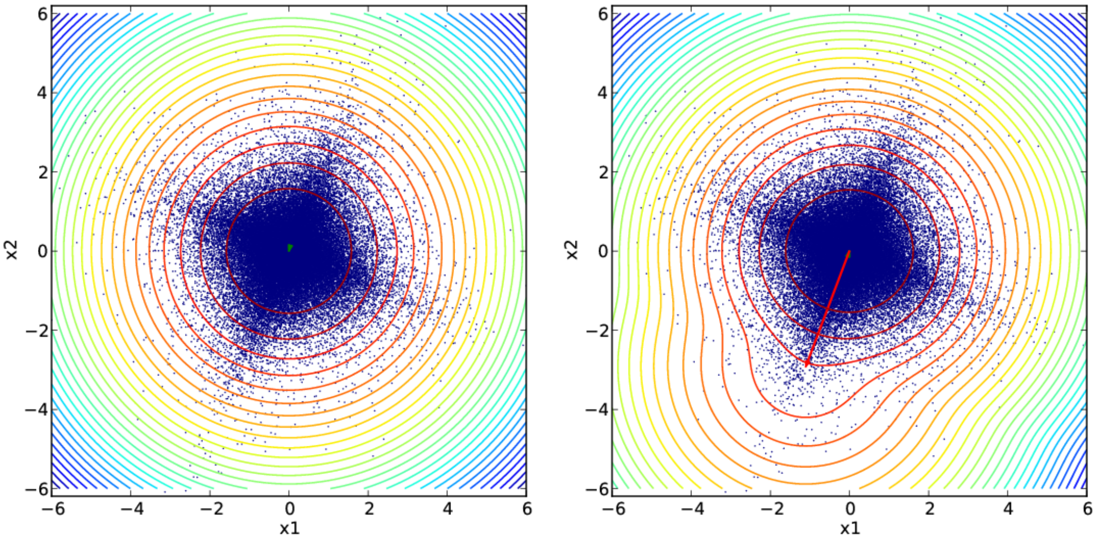


Figure 41: Average RE for GB-RBMs with 196 visible and 16 hidden units, trained on natural image data with different, fixed variance values. LL Training data (green), LL Test data (blue).

5.5 Gaussian-Binary RBMs with a different Number of Hidden Units

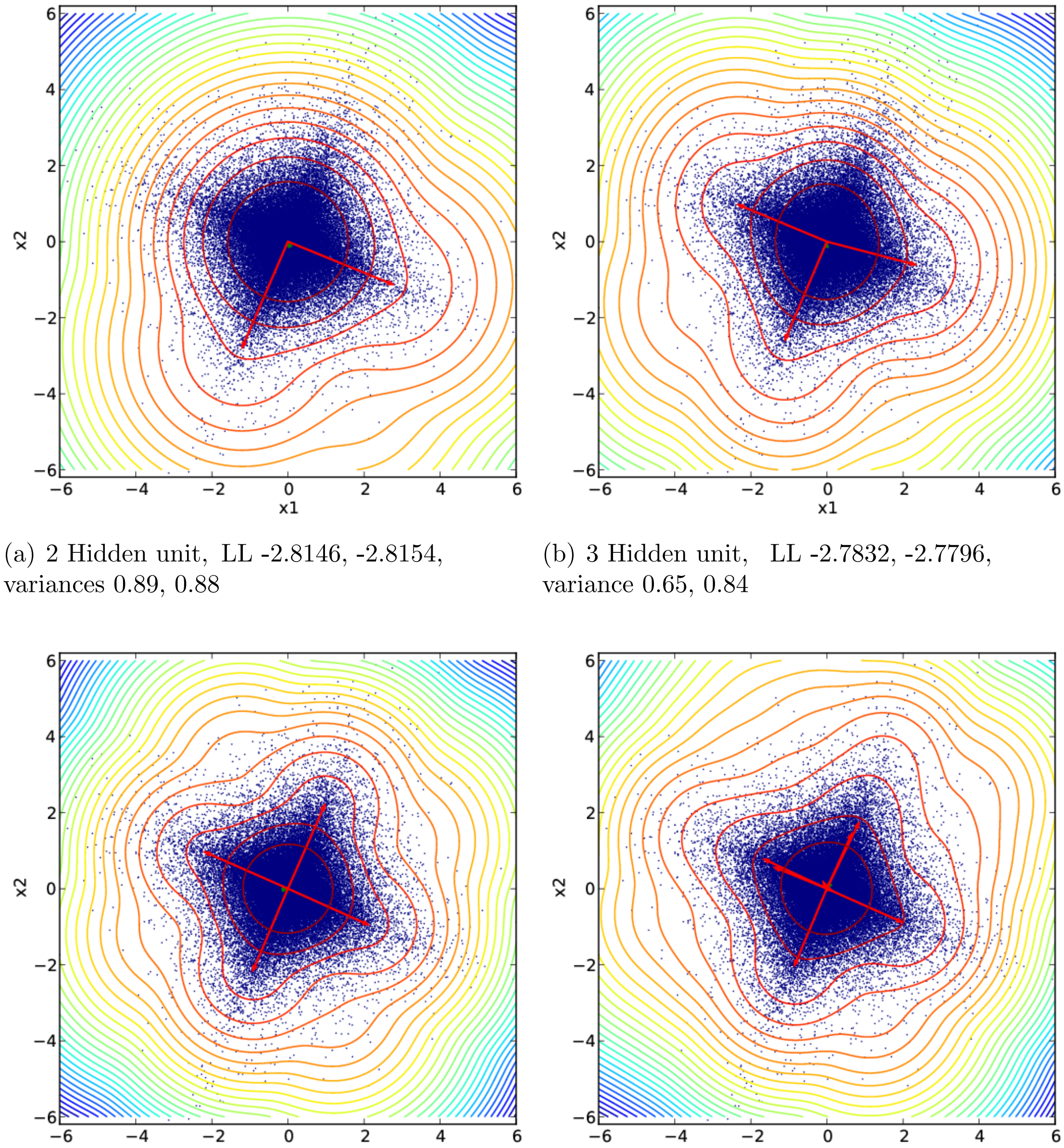
So far, we only considered GB-RBM with two hidden units for the 2D data and 16 and 196 hidden units for the natural image data. The results of the following experiments show how the number of hidden units affect the structure of the model's PDF and the LL. We therefore trained[4] GB-RBMs on the whitened, two dimensional dataset with various numbers of hidden units. The corresponding log-PDFs are shown in Figure 42 and 43. The model without hidden units had only the anchor component, so that the PDF was given by a single unscaled Gaussian with zero mean and variance one, shown in Figure 42 (a). For one hidden unit (b), the visible bias was still positioned in the data's mean and the first order component was placed in the direction of one of the ICs. The variance was slightly decreased so that the first order component could cover a small amount of the density. The model was compensating the asymmetrical shape of the PDF by setting slightly different variances for the dimensions.



(a) 0 Hidden unit, LL -2.8379, -2.8361, variances 1.01, 1.00

(b) 1 Hidden unit, LL -2.8249, -2.8234, variances 0.98 0.90

Figure 42: Contour plots of the GB-RBM's log-PDFs for zero and one hidden unit. The green arrow represents the visible bias, the red arrows represent the weights and the blue dots are the 2D data points.



(a) 2 Hidden unit, LL -2.8146, -2.8154, variances 0.89, 0.88

(b) 3 Hidden unit, LL -2.7832, -2.7796, variance 0.65, 0.84

(c) Variances 1.5, 1.5, LL -2.7404, -2.7365, variances 0.58, 0.56

(d) Variances 2.0, 2.0, LL -2.7330, -2.7316, variances 0.50, 0.52

Figure 43: Contour plots of the GB-RBM's log-PDFs for different numbers of hidden units. The green arrow represents the visible bias, the red arrows represent the weights and the blue dots are the 2D data points.

Figure 43 shows the log-PDF for two, three, four and eight hidden units. In each case the visible bias was positioned in the data's mean and the first order components were placed in the directions of the ICs. If the number of first order components was bigger than the number of directions that needed to be modelled, the additional components were placed in the data's mean or along an already covered IC, as shown in Figure 43 (d). With an increasing number of components the variance was scaled down, so that the first order components could cover more of the density. Note that the two variances had different values to compensate the PDFs asymmetrical shape, in the case of one and three hidden units. The LL improved with an increasing number of hidden units.

For the natural image dataset, we trained [4] several GB-RBMs with different numbers of hidden units. Table 3 shows the LL for the trained models with zero to 784 hidden units. Equivalent to the 2D case the LL as well as the RE improved, although the model overfitted to the training data, while the number of hidden units increased. For zero hidden units the whole density was covered by the anchor Gaussian, but the more hidden units we added the more of the density was modelled by the first order components, indicated by the decreasing variance. As expected, the average variance per input dimension decreased. All GB-RBMs learned LOFS filters, even in the highly overcomplete case, as shown for 784 hidden units in Figure 44.

Number Hidden Units	LL Train	LL Test	RE Train	RE Test	Average Variance	Std. Dev. of Variance
0	-277.62	-277.62	195.02	195.01	0.9888	0.0047
16	-276.91	-277.32	191.05	192.05	0.9727	0.0333
49	-275.37	-276.65	182.18	184.89	0.9369	0.0795
98	-272.94	-275.49	168.53	173.30	0.8805	0.1255
196	-266.82	-272.12	137.71	143.91	0.7487	0.2212
392	-255.53	-266.29	84.13	93.64	0.5191	0.1341
784	-232.75	-253.29	48.98	58.83	0.3501	0.0318

Table 3: LL, RE and variance for GB-RBMs with different numbers of hidden units trained on the natural image dataset.

Moreover, the more hidden units we use, the more likely it gets that higher order components are placed accidentally in regions of higher density. If the GB-RBM with 196 hidden units would only use the first order components, the samples shown in Figure 33 would only be made out of one filter, which is obviously not the case.

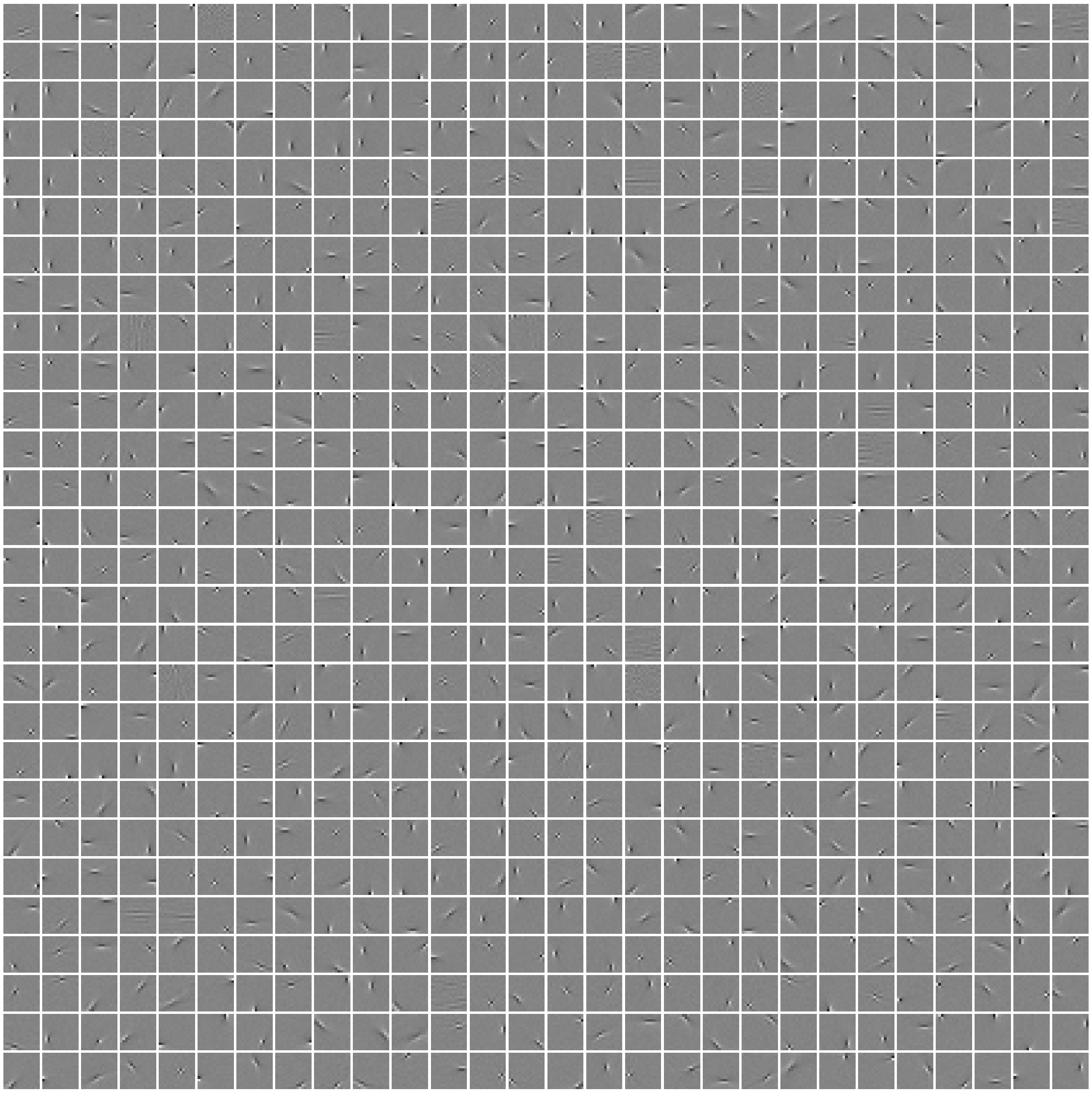


Figure 44: Filters of a GB-RBM with 784 hidden units trained on natural images. The average variance per dimension was 0.35006, with a standard deviation of 0.03178. The average LL estimated by AIS was -232.75348 for the training data and -253.2924 for the test data.

This can also be shown by calculating the average number of active hidden units for the training data, shown in Table 4.

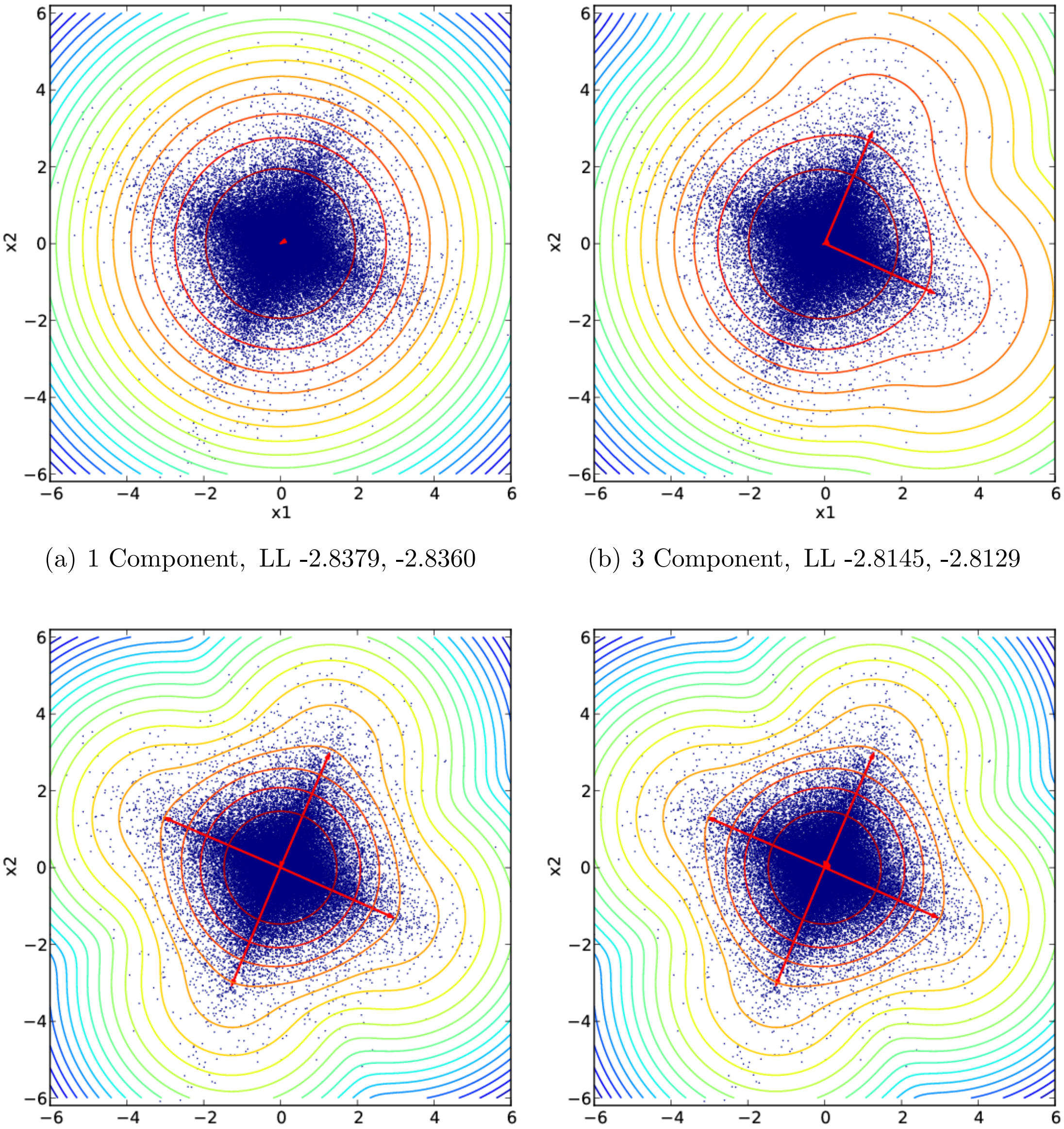
Number of	Average number of	Standard deviation of
hidden units	active hidden units	active hidden units
16	0.2067	0.5068
49	0.7692	1.2075
98	1.7938	2.3916
196	5.4631	5.8260
392	11.8821	11.2857
784	25.0901	23.1206

Table 4: Showing mean and standard deviation of the number of active hidden units for the trained GB-RBMs, for the natural image training data.

5.6 Comparing GB-RBM with Mixture of Gaussians

GB-RBMs are constrained MoGs where all components share a diagonal covariance matrix and as we have seen the model uses mainly the anchor and lower order components to model natural images. Consequently, an unconstrained MoG, having $N+1$ components should in principle be able to learn the same filters as a GB-RBM with N hidden units. But it is not clear if the constraints of GB-RBMs are the major reason why the model learns these filters.
The following experiments were done to check whether MoG are capable of learning LOFS filters. We therefore trained MoGs with one, three, five and eight components on the whitened 2D dataset. All components had an identity matrix as covariance matrix and we only trained the mean and the scaling factors, which were initialized randomly. Figure 45 shows the log-PDFs for the trained MoGs, which obviously show the same structure as the log-PDFs for GB-RBMs, shown in Figure 35 and 36. The red arrows point from the origin to the mean of the components. For all four MoGs, similar to the anchor component in GB-RBMs, one component was placed in the data's mean that had a much bigger scaling factor than the other components. If the variance of the GB-RBMs were also fixed to one, even the LL was almost the same.

For the natural image dataset, we trained an MoG with 196 components and fixed identity covariance matrices. Equivalent to the 2D experiments, we trained only the means and the scaling factors. The learned filters are shown in Figure 46, which shows similar filters as the once learned by GB-RBMs and ICA. But also a lot of uniform filters that correspond to weights that converged to zero.



(a) 1 Component, LL -2.8379, -2.8360

(b) 3 Component, LL -2.8145, -2.8129

(c) 5 Component, LL -2.7927, -2.7931

(d) 8 Component, LL -2.7927, -2.7931

Figure 45: Contour plots of the MoGs log-PDFs for different numbers of components. The covariance matrix has been fixed to the identity matrix. The red arrows point to the components means. In each case one component is placed in the data's mean.

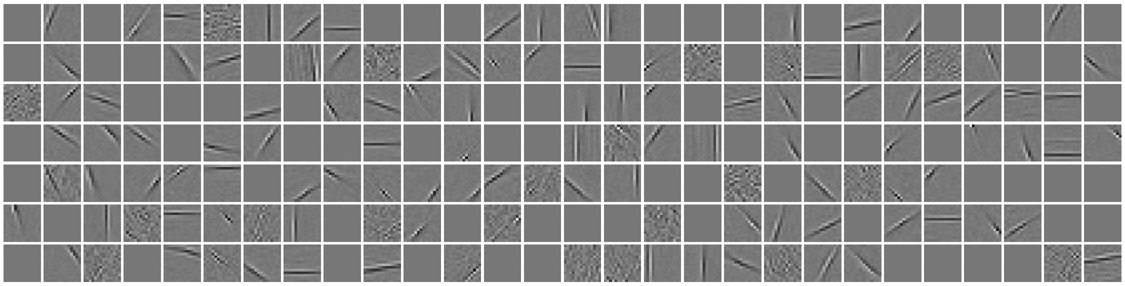


Figure 46: Means of an MoG with 196 components and a fixed identity covariance matrices. The LL was -274.8760 and -271.11095 for the test data.

We assumed, that this happened due to the fact that the EM algorithm is not able to escape local minima with all components. Therefore, we trained an MoG with the same setup, but allowed only one of the components to be positioned in the data's mean. The filters are shown in Figure 46 showing more structured filters and only one uniform filter that corresponds to the anchor component. But we also got a lot of noisy filters, which did not disappear while continuing training. We assumed that this happened due to the training algorithm, since the MoG fixed to the weights learned by GB-RBM but free scaling factors had a slightly better LL of -273.67 in training. The LL of the MoGs is worse compared to the LL of GB-RBM, which is caused by the missing higher order components.

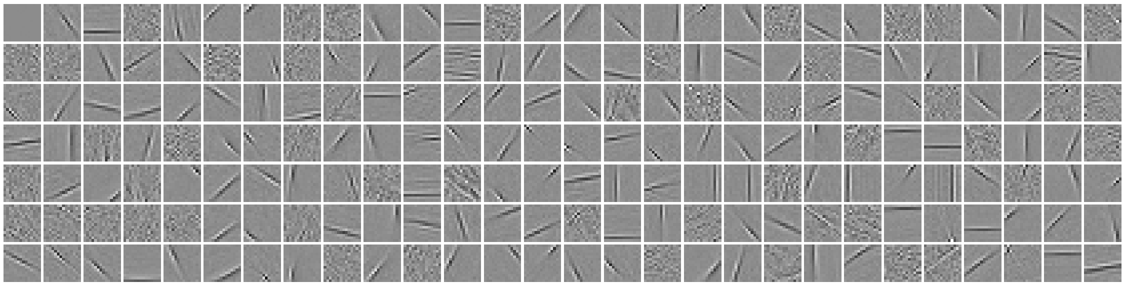
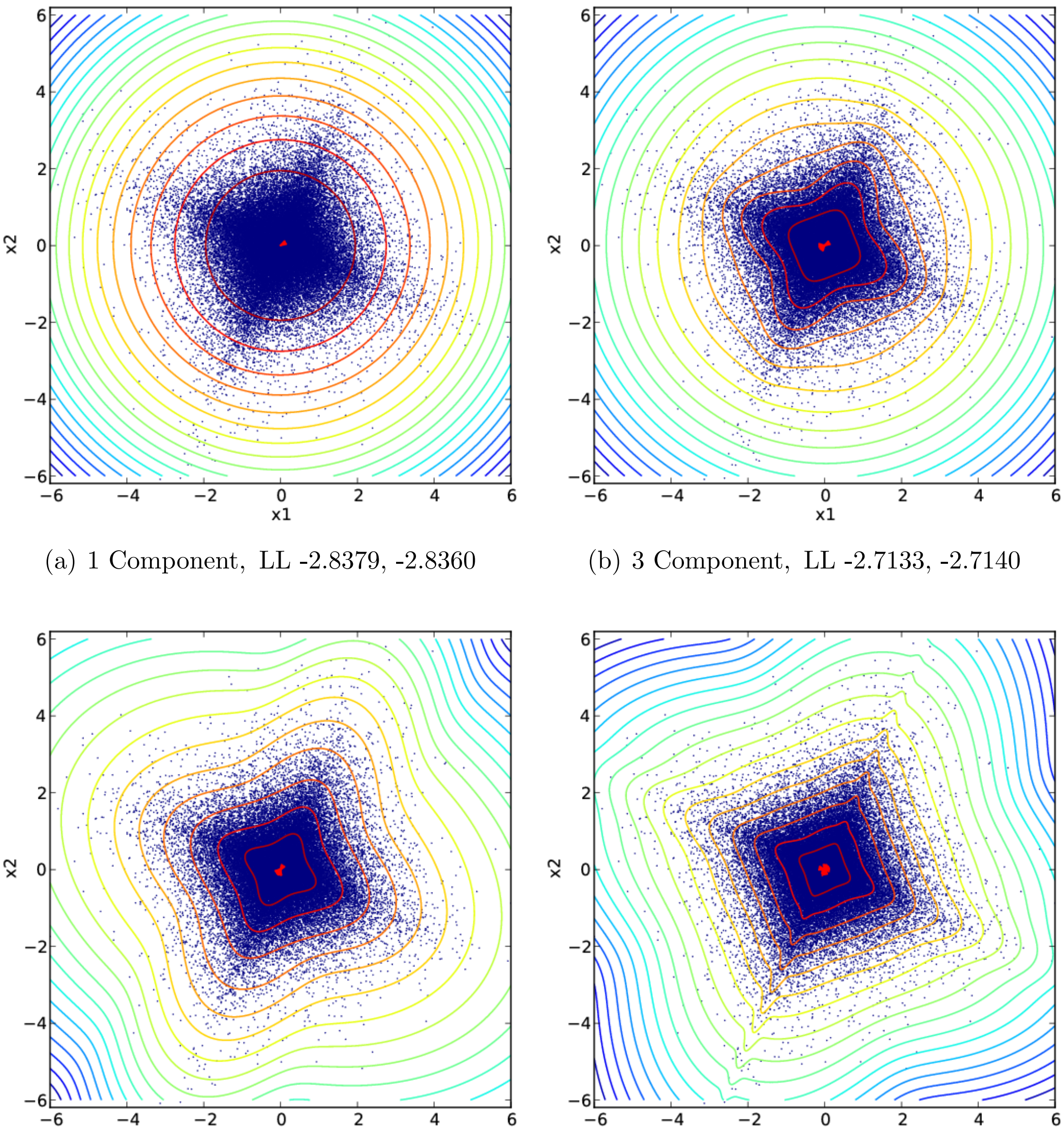


Figure 47: Means of an MoG with 196 components and a fixed identity covariance matrices. Only the first weight was allowed to have a value close to zero. The LL was -274.2224 and -270.5980 for the test data.

We also trained MoGs with one, three, five and eight components on the whitened 2D dataset, with full, trained covariance matrices. The plots of the 2D log-PDFs



(a) 1 Component, LL -2.8379, -2.8360

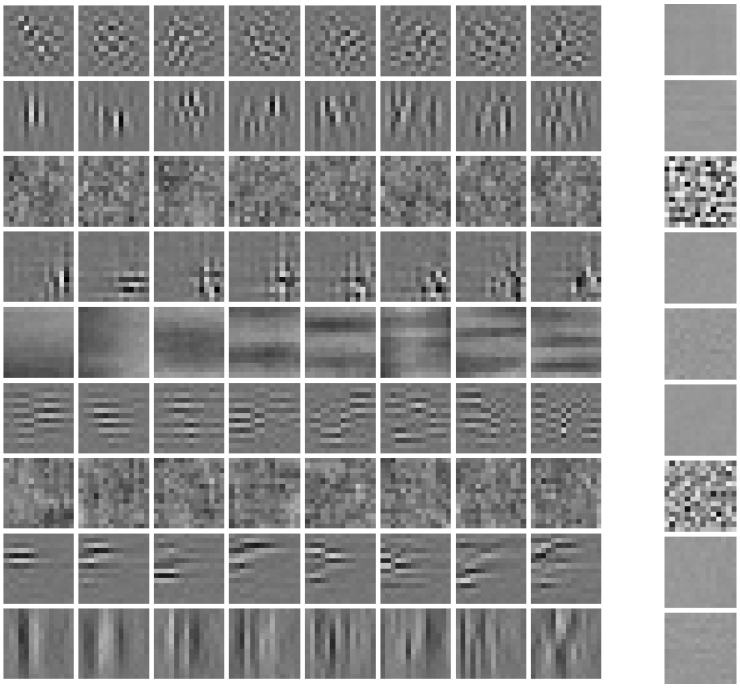
(b) 3 Component, LL -2.7133, -2.7140

(c) 5 Component, LL -2.7012, -2.7015

(d) 8 Component, LL -2.6974, -2.69791

Figure 48: Contour plots of the MoGs' log-PDFs for different numbers of components with full covariance matrices. The red arrows point to the components means which are placed in the data's mean in each case.

are shown in Figure 48. For one component, the result was equivalent to the MoG with diagonal covariance matrix. For all experiments the components were placed in the data's mean and the shape of the PDF was modelled only by the covariance matrices and the scaling factors. The LL was better than for the experiments with spherical covariance matrices and the model fitted the PDF shape, already with three components quite well.



(a) Eigenvectors (b) Means

Figure 49: (a) Each row shows eight eigenvectors of the covariance matrix of a multivariate Gaussian distribution. (b) The corresponding mean of the components. The Components had free covariance matrices and the MoG was trained on the natural image data.

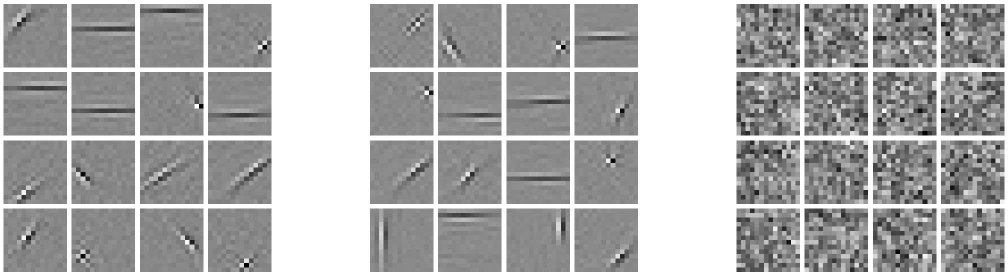
In the 2D experiments, at least some eigenvectors of the covariance matrices pointed into the directions of the ICs. For the natural images, we therefore trained an MoG with full covariance matrices and nine components. Figure 49 shows eight eigenvectors for each component and their mean. The components that had almost uniform looking mean filters were placed in the data's mean. The eigenvectors for these components show localized, orientated structures with different frequencies.

Although these filters are different from the results of ICA, GB-RBM and MoG with spherical covariance matrix, they look like a combination of those filters for different frequencies. The LL for the model was -229.3514 for the training data and -241.8714 for the test data.

5.7 Training GB-RBM Successfully

From the analysis we know, how a GB-RBM models data and especially, how it models the natural image data. This knowledge can be used to choose a better training setup, which allows faster and more successful training. The following experiments were done to analyse and explain the effect of different values for the hyperparameters.

We trained[4] GB-RBMs with 16 hidden units on the natural images, using different learning rates. The small number of hidden units allowed us to calculate the LL exactly. Figure 51 shows the LL evolution for learning rates of 0.1, 0.01 and 0.001 over 1,000 epochs, which correspond to 500,000 gradient updates. Accordingly, the learning rate needs to be sufficiently big for successful training, in an acceptable number of gradient updates. Figure 50 shows the corresponding filters after 1,000 epochs. We did not learn any meaningful filters for a learning rate of 0.001.



(a) Learning rate 0.1 (b) Learning rate 0.01 (c) Learning rate 0.001

Figure 50: Learned filters of a GB-RBM with 16 hidden units trained on the natural images for different learning rates. Note that the images have been normalized, (c) had values close to zero.

[4] Training setup: LL average of 10 trials, 50,000 image patches, 1,000 epochs, CD-1, batch size 100 Momentum 0.0, Weight decay 0.0

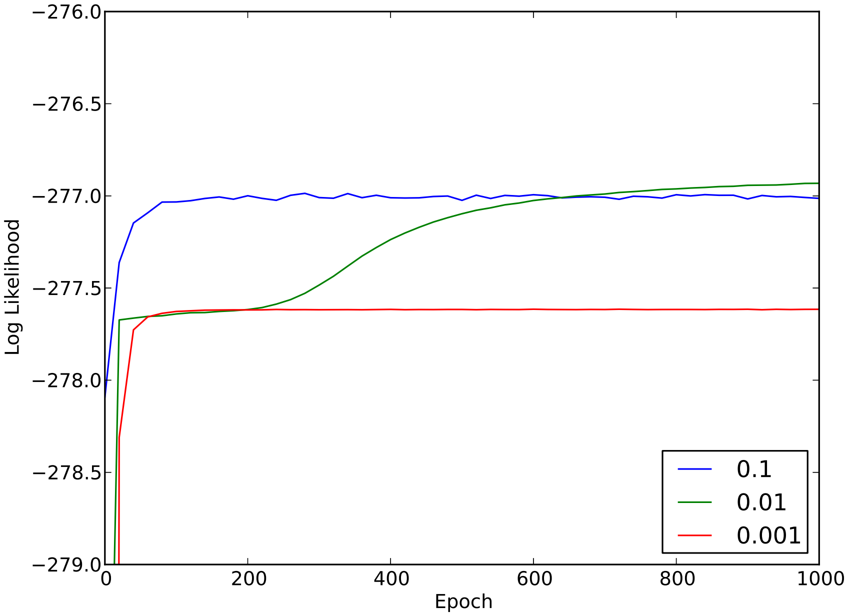


Figure 51: LL evolution of GB-RBMs with 16 hidden units, trained on the natural images using different learning rates.

Figure 52 shows the evolution of the average weight norm of the GB-RBMs. At the beginning, the weights for all three learning rates converged quickly to a value close to zero. This happened because, in the early states of training, the randomly initialized components will move towards regions of high density and in the case of natural images most of the data is located close to the mean. For a learning rate of 0.001 the weights continued converging to zero, which led to the noisy filters, while the weights for the bigger learning rates began to grow after some gradient updates, which led to the LOFS filters. The average weight norm and the LL have a comparable evolution, so that the weight norms are a strong indicator for the learning process of GB-RBMs on natural images.

As mentioned, for all learning rates the components converged towards the data's mean, in an early stage of training. But scaled Gaussians, placed at the same location, having the same covariance matrix are equivalent to a single Gaussian, with the same mean and covariance matrix, scaled by the sum of the single scaling factors. Consequently, placing the components all in the data's mean is unnecessary since

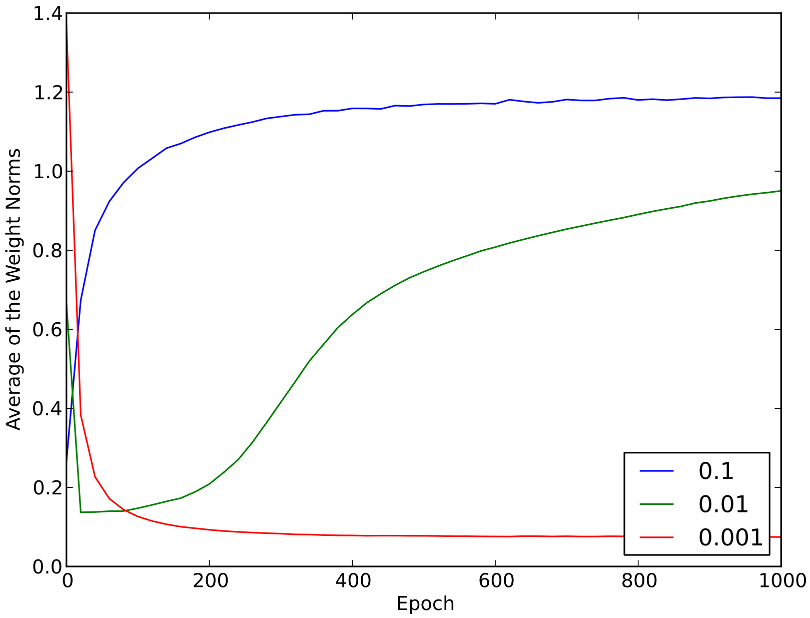


Figure 52: Evolution of the average weight norm of GB-RBMs with 16 hidden units, trained on the natural images using different learning rates.

the anchor component can model this region already. Accordingly, fixing the visible bias to zero without training, led to the same solutions for the natural image data. The other components could be used to model the density in the directions of the independent sources, which is much sparser than the density around the data's mean. And in general, the density of the natural image dataset decrease exponentially while the distance to the data's mean increase.

Consequently, the model needed to learn that the components have to be scaled down in order to model these regions. Figure 53 shows the evolution of the average first order scaling factors for the three different learning rates. The graph shows a similar evolution as the average weights and the LL evolution. The GB-RBMs with a learning rate of 0.1 and 0.01 learned that the components need to be scaled down. The GB-RBM with learning rate 0.001 did not learn small scaling factors. What makes learning difficult, is that moving a component will also change the scaling, see Eq. (243). We assume that the learning rate of 0.001 is simply to small to learn the right weight - hidden bias combination in an acceptable number of gradient updates.

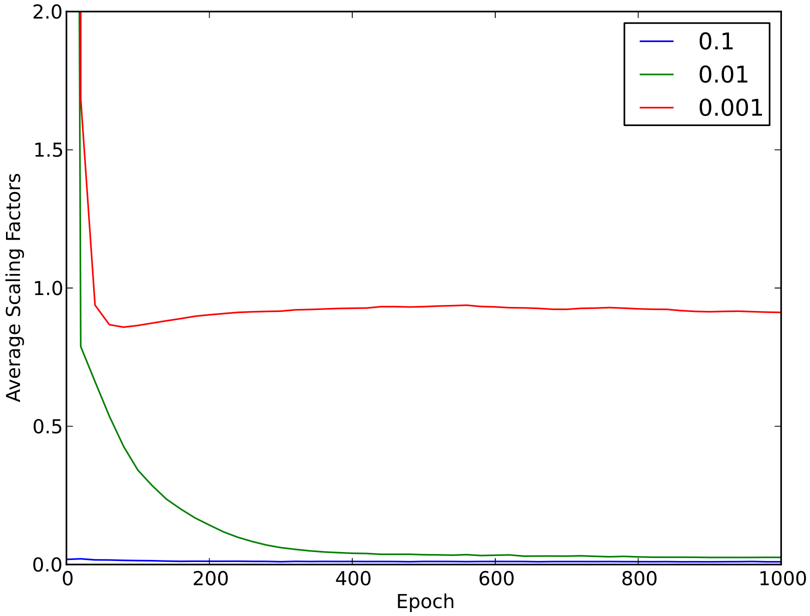


Figure 53: Evolution of the average first order scaling factors of GB-RBMs with 16 hidden units, trained on the natural images using different learning rates.

Consequently, if we use a small learning rate it is advisable to initialize the GB-RBM so that the scaling factors are relatively small, i.e around 0.01. Considering Eq. (243), this can be achieved by choosing the initial hidden by $c_j = -\|\frac{\mathbf{b}+\mathbf{w}_{*j}}{2\sigma}\|^2 + \|\frac{\mathbf{b}}{2\sigma}\|^2 + \tau_j$, where $\tau_j = 0.01$ determines the relative scaling to the anchor component.

The simple solution where all components are placed in the data's mean is a strong local optimum. Adding noise in the early stage of training helps to escape this local optimum. Another opportunity is to use the hidden states rather than the probabilities for the gradient calculation. This adds noise and has the advantage that the influence will automatically be reduced when the probabilities are getting closer to zero and one while continuing training.

A momentum term adds a percentage of the past gradient to the current gradient, which leads to a smoother gradient trajectory and makes the gradient more robust to noise. But it makes the gradient less flexible, which can slow down or even prevent convergence. Especially if the step size of the gradient is very big, the gradient can easily oscillate around the optimum.

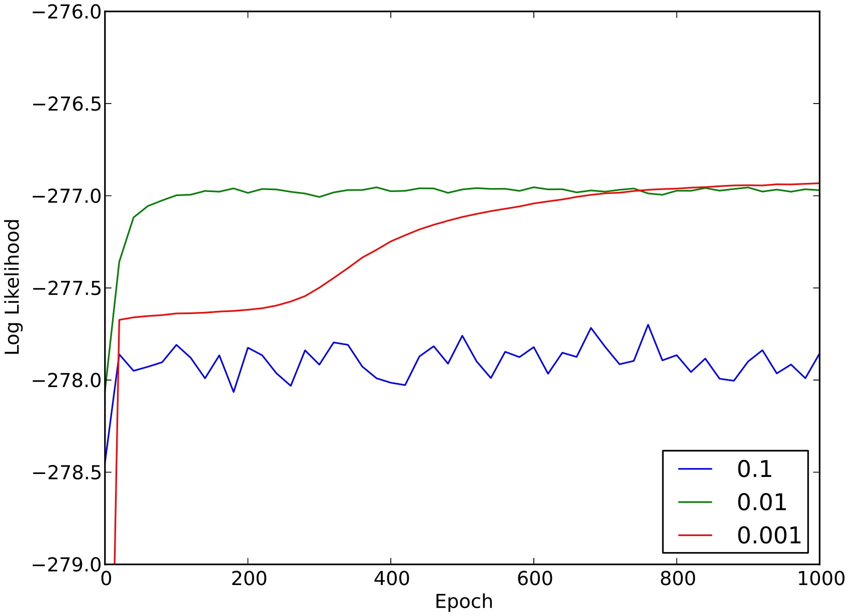


Figure 54: LL evolution of GB-RBMs with 16 hidden units, trained on the natural images using different learning rates and a momentum of 0.9.

Figure 54 shows the LL evolution of GB-RBMs with 16 hidden units trained[4] for different learning rates, using a momentum of 0.9. The LL evolution for a learning rate of 0.01 with momentum is comparable with the LL evolution of 0.1 without a momentum. Equivalent, a learning rate of 0.001 with momentum and 0.01 without momentum are comparable. The LL of the learning rate 0.1 is worse and oscillating around a value of -278.0. But Figure 56 shows that all versions learn LOFS filters, so that for a learning rate of 0.1, the momentum prevented the convergence, but the weights point in the right directions. If the learning rate was reduced or the momentum removed after 100 epochs, the LL for the learning rate of 0.1 converged very quickly to a value around -277.0. The average weight norm was growing in all three cases, as shown in Figure 55. Accordingly, a momentum term can be used to keep the gradient in an exploratory state that prevents it from converging to zero, especially in the beginning of training and for small learning rates.

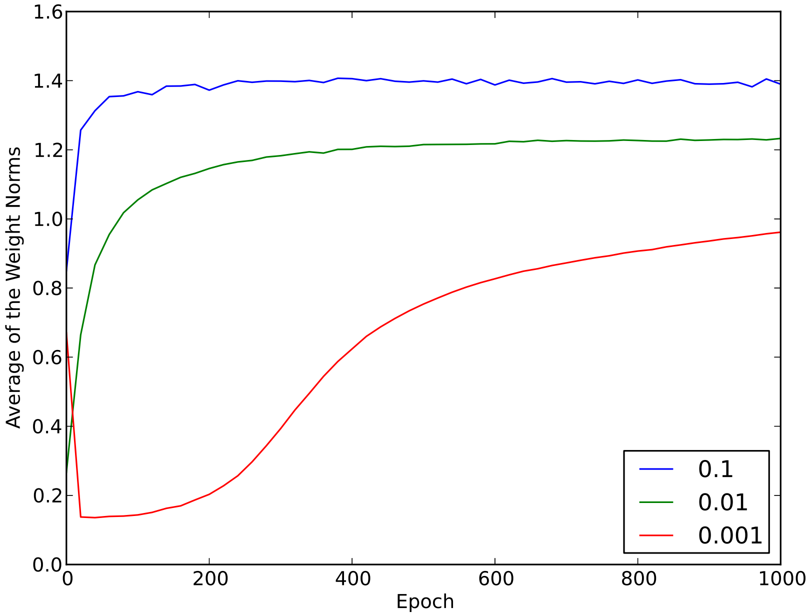


Figure 55: Evolution of the average weight norm of GB-RBMs with 16 hidden units, trained on the natural images using different learning rates and a momentum of 0.9.

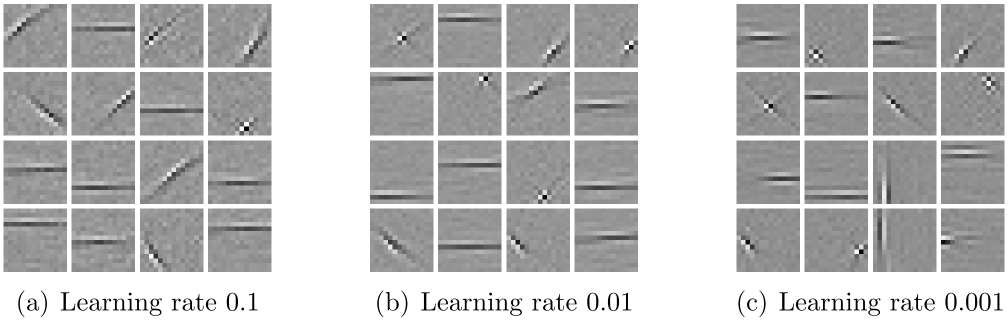


Figure 56: Learned filters of a GB-RBM with 16 hidden units trained on the natural images for different learning rates and a momentum of 0.9.

If the learning rate is too big, the gradient can easily diverge, which causes numerical overflows. This happened already for the GB-RBMs with 16 hidden units for a learning rate of 0.1 in some trials and became worse for bigger models. But, since we know that the components are placed on the data, there is no need for a gradient with a norm twice as big as the maximal data norm. Then the gradient is still able to place a component within one gradient update everywhere on the data. We therefore restricted the gradient's norm for the experiments done in this thesis, which allowed us to train also big models with a learning rate of 0.1.

Using a weight decay when training GB-RBMs is not necessary. We know that the components will be placed on the data and therefore the weight's norm will naturally stay smaller than the maximal data's norm. Furthermore, a weight decay term prevents the weights from growing, but according to the previous discussion this is exactly what is needed. So that a weight decay will worsen the training process.

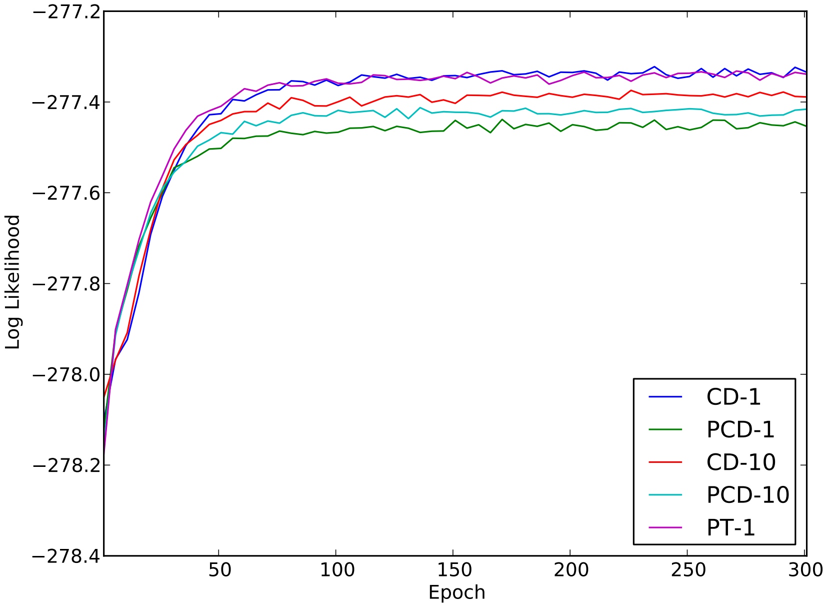


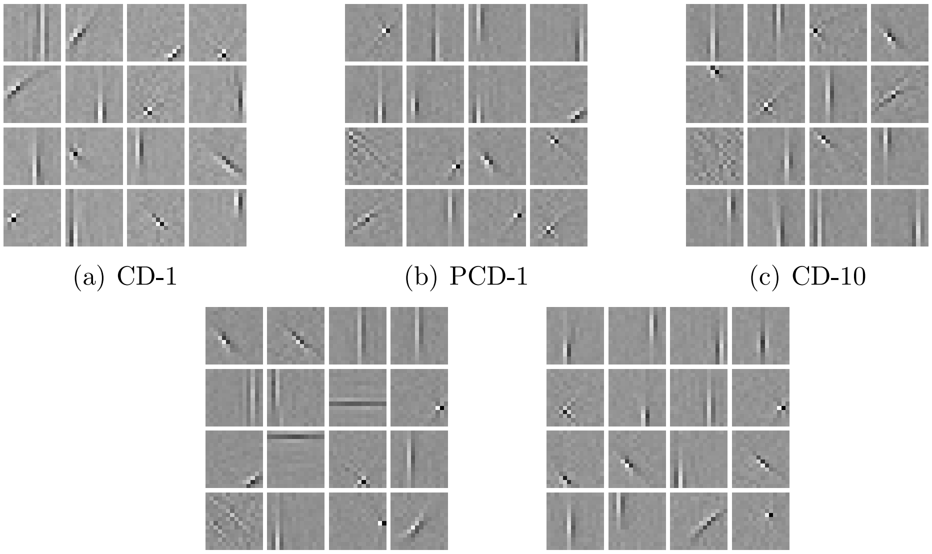
Figure 57: LL evolution of GB-RBMs with 16 hidden units, trained with different training methods on the natural images data.

For the last experiment we trained[5] GB-RBMs using different training methods. Figure 58 shows the LL evolution for CD with one step of Gibbs sampling (CD-1), CD

[5] Training setup: Average of 5 trials, 50,000 image patches, 200 epochs, batch size 100, Learning rate 0.01, Momentum 0.9, Weight decay 0.0

with ten steps of Gibbs sampling (CD-10), PCD with one step of Gibbs sampling (PCD-1), PCD with ten steps of Gibbs sampling (PCD-10) and PT with one step of Gibbs sampling using 20 temperatures (PT-20). PCD-1, PCD-10 and CD-10 show slightly worst performance than for CD-1 and PT-20, but they will reach the same value as CD-1 and PT-20 if we reduce the learning rate in the later stage of training. We never, not only for this experiment, observed any benefit of using a bigger k or PT for training GB-RBMs on whitened natural images. This comes from the unimodality of the data distribution which causes PCD and PT to perform basically like CD since we never miss a mode as described in Chapter 4.

Since the use of a bigger k or PT lead to a much higher computational cost, it is advisable to use CD-1. The following list summarizes how a GB-RBM can be trained efficiently on whitened natural images. A GB-RBM with 784 hidden units has been trained using the following setup, in just 100 epochs. The filters are shown in Figure 44 which show LOFS structures.



(a) CD-1

(b) PCD-1

(c) CD-10

(d) PCD-10

(e) PT-20

Figure 58: Filters of GB-RBMs with 16 hidden units, trained with different training methods on the natural images data.

In our experience the following tips will help training GB-RBMs successfully.

- Using CD-1 is sufficient
- Initialize the weights to small random values, $w_{ij} = \mathcal{N}(0, 0.01)$
- Initialize the hidden values so that the first order scaling factors are small, $c_j = -\|\frac{\mathbf{b}+\mathbf{w}_{*j}}{2\sigma}\|^2 + \|\frac{\mathbf{b}}{2\sigma}\|^2 + \tau_j$, i.e. $\tau_j = 0.01$
- Fix the visible bias to zero, $\mathbf{b} = 0$
- Choose a learning rate between 0.1 or 0.01, which should be reduced in final training stage
- Use a momentum between 0.5 and 0.95
- Use no weight decay
- Restricted the gradient norm, $\|\bigtriangledown\mathbf{w}_{*j}\| \leq max_d\|\mathbf{x}_d\|$
- Use hidden states for the gradient calculation at least for the first epochs
- Track the weight norms as a measurement for the training progress

6 Conclusions

This thesis discussed Gaussian-Binary RBMs for learning natural image statistics. We motivated this work with a brief introduction to natural images and how they are processed by the simple cells in the primary visual cortex. A detailed introduction to BMs and the related concepts was given, which can be used as a reference work on this topic.

We analysed the model and showed that GB-RBM with N hidden units can be reformulated as a constrained MoG with 2^N components that share a diagonal covariance matrix. This formulation allows a much better understanding of how GB-RBMs actually model data than the common PoE formulation does. It turned out that the constraints of the MoG representation forces the components to lie on a parallelepiped, which is a projected hypercube and limits the representational power of the model. We showed that GB-RBMs use mainly the lower order components to model the data, unless the data is structured according to a parallelepiped. This leads naturally to a sparse hidden representation, since the order of the components represents the number of simultaneously active hidden units. We argued that ICA and GB-RBMs are related, since both models belong to the PoE, although they use different prior distributions.

We have shown that GB-RBMs are capable of learning natural image statistics and that the learned filters show a location, orientation and frequency selective structure, comparable to the receptive fields of simple cells, found in the primary visual cortex. These filter were very similar to the filters learned by ICA, which we assumed to be a good model for natural images. In contrast to normal ICA methods, like FastICA, GB-RBMs are not limited to a complete representation and we showed that it is possible to learn highly overcomplete representations.

Due to the restriction of the components to share a diagonal covariance matrix, we assumed that whitened data, which is more symmetrical, should be more suitable for GB-RBMs. We showed that only whitened data leads to high frequency LOFS filters similar to the ICA result. The GB-RBMs modelled the natural image data by placing the anchor component in the data's mean and the higher order components in the directions of the independent sources. With an increasing number of hidden units the LL improved since more sources were covered and the filters showed still LOFS structures.
When training the variances of the model the LL improved and the filter became

more dot-like. The optimal variance for the whitened data depended on the number of hidden units and was always smaller than one. But training the variance also increased the effect of over fitting.
We showed that an MoG, where all components share a diagonal covariance matrix is able to learn the same filters than GB-RBMs do. And in the 2D case, the learned PDF had exactly the same structure.
We further showed that the knowledge about the natural image PDF and how GB-RBMs model data can be used to choose a better training setup. To train a GB-RBM on natural image data, the learning rate needs to be big, which can easily lead to divergence of the gradient especially for big models. Since we know that the components are placed on the data, we proposed to restrict the gradient to twice the maximal data norm. This prevented divergence and allowed us to train big models successfully. We argued that a weight decay is counterproductive since the weights should reach a certain norm. A momentum term keeps the gradient in an exploratory mode, which helps to avoid local optima. Empirical results showed that CD-1 is sufficient and that using more Gibbs sampling steps or Parallel tempering will not improve the training process.

All three models are capable of learning LOFS filters but only ICA and GB-RBMs allow fast and reliable training on natural images. An advantage of GB-RBMs and MoGs compared to ICA is that they are not restricted to the complete case so that they are able to learn highly over-complete representations. In contrast to ICA, both models are generative models and allow stacking for building deep networks. ICA has a better LL than GB-RBMs and MoGs but the LL does not provide any further information apart from how likely the data is generated by the model.

In order to compare the models further, future work could focus on the discriminative properties of the filters of ICA and GB-RBMs. One could also focus on how the proposed training improvements affect the solutions learned by deep believe networks and deep Boltzmann machines. A promising research direction is also to find a preprocessing of the data, which supports the structure of a GB-RBM better. But in general it is not clear how good the binary hidden representation is that GB-RBMs learn and if the conditional independence is not a too strong assumption for modelling natural image statistics. It would be interesting to train other models, which provide a binary representation on the natural images and compare the results. Moreover, the learned representation could then be used to train a binary RBM or DBN to see whether it improves the performance of the whole network.

References

[1] Hyvärinen A. and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9(7):1483–1492, 1997.

[2] David H. Ackley, Geoffrey E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. 1985.

[3] Yoshua Bengio. Learning deep architectures for AI. *FOUND TRENDS MACH LEARN*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the Conference on Neural Information Processing Systems*, pages 153–160, 2006.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*, chapter 8, pages 359–422. Springer, Secaucus, NJ, USA, 2006.

[6] Cho, Ilin, and Raiko. Improved learning of gaussian-bernoulli restricted boltzmann machines. 2011.

[7] KyungHyun Cho, Alexander Ilin, and Tapani Raiko. Improved learning of gaussian-bernoulli restricted boltzmann machines. In *Proceedings of the*, pages 10–17, 2011.

[8] KyungHyun Cho, Tapani Raiko, and Alexander Ilin. Enhanced gradient and adaptive learning rate for training restricted boltzmann machines. In *Proceedings of the Annual International Conference on Machine Learning*, Washington, 6 2011.

[9] Aaron C. Courville, James Bergstra, and Yoshua Bengio. A spike and slab restricted boltzmann machine. *J MACH LEAN RES*, 15:233–241, 2011.

[10] G. Desjardins, A. Courville, and Y. Bengio. Adaptive parallel tempering for stochasticmaximum likelihood learning of rbms. 2011.

[11] Guillaume Desjardins, Aaron Courville, Yoshua Bengio, Pascal Vincent, and Olivier Delalleau. Parallel tempering for training of restricted boltzmann machines. In *Proceedings of the International conference on Artificial Intelligence and Statistics*, 2010.

[12] A. Fischer and C. Igel. Empirical analysis of the divergence of gibbs sampling based learning algorithms for restricted boltzmann machines. 2009.

[13] Asja Fischer and Christian Igel. Markov-random-fields und boltzmann maschinen. Vorlesungsnotizen Version 0.1.5, 2010.

[14] Manuel Gnther, Dennis Haufe, and Rolf P. Wrtz. Face recognition with disparity corrected gabor phase differences. In *Proc. ICANN*, 2012.

[15] Larochelle H., Bengio Y., Louradour J., and Lamblin P. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 2009.

[16] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *SCIENCE*, 313(5786):504–507, 7 2006.

[17] G. E. Hinton and T. J. Sejnowski. Learning and relearning in boltzmann machines. In *Parallel distributed processing: explorations in the microstructure of cognition*, volume 1, pages 282–317. MIT Press, Cambridge, MA, USA, 1986.

[18] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *NEURAL COMPUT*, 14:1771–1800, 8 2002.

[19] Geoffrey E. Hinton. A practical guide to training restricted boltzmann machines. Technical report, Department of Computer Science, University of Toronto, 8 2010.

[20] Aapo Hyvärinen. Connections between score matching, contrastive divergence, and pseudolikelihood for continuous-valued variables. *NEURAL NETWORKS*, 18(5):1529 –1531, sept. 2007.

[21] Aapo Hyvärinen, Patrik O. Hoyer, and Jarmo Hurri. *Natural Image Statistics - A probabilistic approach to early computational vision.* 2009.

[22] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. *Independent Component Analysis.* 2001.

[23] M. Inki. Extensions of independent component analysis for natural image data. 2005.

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, Toronto, 4 2009.

[25] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *NEURAL COMPUT*, 20(6):1631–1649, 2008.

[26] Nicolas Le Roux, Nicolas Heess, Jamie Shotton, and John Winn. Learning a generative model of images by factoring appearance and shape. *NEURAL COMPUT*, 23(3):593–650, 12 2011.

[27] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Honglak Lee, Chaitanya Ekanadham, and Andrew Y.Ng. Sparse deep belief net model for visual area v2. In *Proceedings of the 20th Conference on Neural Information Processing Systems*. MIT Press, 2007.

[29] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 609–616, New York, NY, USA, 2009. ACM.

[30] Benjamin M. Marlin, Kevin Swersky, Bo Chen, and Nando de Freitas. Inductive principles for learning restricted boltzmann machines. In *Proceedings of the International conference on Artificial Intelligence and Statistics*, 2010.

[31] Radford M. Neal. Probabilistic inference using markov chain monte carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

[32] Radford M. Neal. Annealed importance sampling. *STAT COMPUT*, 11:125–139, 2001. 10.1023/A:1008923215028.

[33] Bruno A. Olshausen and David J. Field. Natural image statistics and efficient coding. *NETWORKS*, 7(2):333–339, 5 1996.

[34] Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *VISION RES*, 37(23):3311 – 3325, 1997.

[35] Marc'Aurelio Ranzato, Y-Lan Boureau, and Yann LeCun. Sparse feature learning for deep belief networks. In *Proceedings of the Conference on Neural Information Processing Systems*, pages –1–1, 2007.

[36] Marc'Aurelio Ranzato and Geoffrey E. Hinton. Modeling pixel means and covariances using factorized third-order boltzmann machines. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2551–2558, 2010.

[37] Marc'Aurelio Ranzato, Alex Krizhevsky, and Geoffrey E. Hinton. Factored 3-way restricted boltzmann machines for modeling natural images. *J MACH LEAN RES*, 9:621–628, 2010.

[38] Kevin Swersky, MarcAurelio Ranzato, David Buchman, Benjamin M. Marlin, and Nando de Freitas. On autoencoders and score matching for energy based models. *ICML*, 2011.

[39] Yee Whye Teh, Max Welling, Simon Osindero, and Geoffrey E. Hinton. Energy-based models for sparse overcomplete representations. *J MACH LEARN RES*, 4:1235–1260, 2003.

[40] Lucas Theis, Sebastian Gerwinn, Fabian Sinz, and Matthias Bethge. In all likelihood, deep belief is not enough. *J MACH LEAN RES*, 12:3071–3096, 11 2011.

[41] Tijmen Tieleman. Training restricted boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th Annual International Conference on Machine Learning*, pages 1064–1071, New York, NY, USA, 2008. ACM.

[42] C. S. Wallace and D. L. Dowe. Minimum message length and kolmogorov complexity. 1999.

[43] Nan Wang, Jan Melchior, and Laurenz Wiskott. An analysis of gaussian-binary restricted boltzmann machines for natural images. *ESANN*, 2012.

[44] Max Welling, Michal Rosen-Zvi, and Geoffrey E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Proceedings of the 17th Conference on Neural Information Processing Systems*. MIT Press, December 2004.

[45] Laurenz Wiskott, Jean-Marc Fellous, Norbert Fellous, and Christoph von der Malsburg. Face recognition by elastic bunch graph matching. 1999.

Appendix

More than half of the time spent on this thesis was used for developing an RBM toolkit in Python named pyrbm. Although there exist already various libraries for RBMs, in my opinion these libraries are focused mainly on usage rather than on modifiability and comprehensibility of the code and the mathematics of RBMs.
That is why I decided to write my own RBM-module, which has a simple modular and well documented structure. The implementation is very close to the mathematical notation used in this thesis which supports the comprehensibility and allows to modify the code easily to your own needs.

Beside I came across GPU coding and implemented the whole code also for the GPU using CUDAMAT[6]. The structure for the CPU code and the GPU code are equivalent. This means that the functions perform the same calculations, so that it is easy to learn GPU coding with CUDAMAT beside.
Table 5 shows the time in seconds needed for one gradient update of a GB-RBM. The CPU was a intel i5-750 with 4 cores and 8 GB RAM and the GPU was a Geforce GTX-570 with 3GB RAM. We were running the code on the GPU, the CPU with and without MKL[7]. Figure 59 shows the structure of the toolkit. The latest version will be available as an open source project[8] soon and is planned to be integrated into MDP[9].

Number of Visibles	Number of Hiddens	CPU time in s	CPU+MKL time in s	GPU time in s	Speed up CPU MKL to GPU
50	50	0.002322	0.001790	0.003375	-0.469624
100	100	0.006465	0.004346	0.003594	0.209154
400	400	0.072758	0.029232	0.004627	5.317720
800	800	0.265682	0.086769	0.005640	14.384469
1600	1600	1.139502	0.274237	0.010303	25.617161
6400	6400	19.40761	4.574618	0.116391	38.303899

Table 5: Time needed for one gradient update on CPU (intel i5-750 with 8 GB RAM) with and without MKL and on GPU (Geforce GTX-570 with 3GB RAM).

[6] http://code.google.com/p/cudamat/
[7] http://software.intel.com/en-us/articles/intel-mkl/
[8] http://sourceforge.net/projects/pyrbm/
[9] http://mdp-toolkit.sourceforge.net/

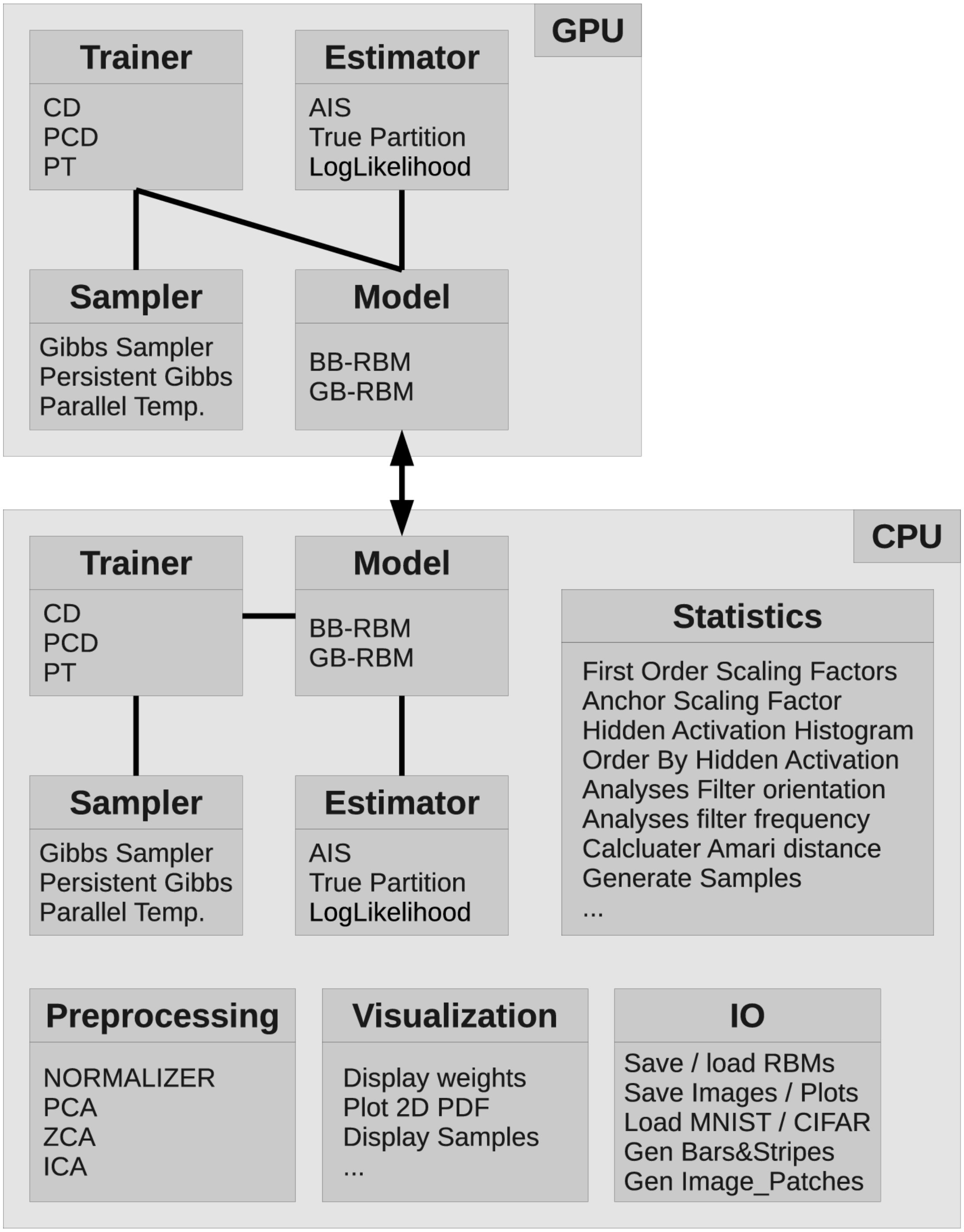


Figure 59: Structure of the RBM toolkit pyrbm