Bidirectional Helmholtz Machines

Samira Shabanian
Department of Computer Science and Operations Research
University of Montreal
Montreal, Quebec, Canada

31 août 2016

Résumé

L'entraînement sans surveillance efficace et inférence dans les modéles génératifs profonds reste un probléme difficile. Une approche assez simple, la machine de Helmholtz, consiste à entraîner du haut vers le bas un modéle génératif dirigé qui sera utilisé plus tard pour l'inférence approximative. Des résultats récents suggérent que de meilleurs modéles génératifs peuvent étre obtenus par de meilleures procédures d'inférence approximatives. Au lieu d'améliorer la procédure d'inférence, nous proposons ici un nouveau modéle, la machine de Helmholtz bidirectionnelle, qui garantit qu'on peut calculer efficacement les distributions de haut-vers-bas et de bas-vers-haut. Nous y parvenons en interprétant à les modéles haut-vers-bas et bas-vers-haut en tant que distributions d'inférence approximative, puis ensuite en définissant la distribution du modéle comme étant la moyenne géométrique de ces deux distributions. Nous dérivons une borne inférieure pour la vraisemblance de ce modéle, et nous démontrons que l'optimisation de cette borne se comporte en régulisateur. Ce régularisateur sera tel que la distance de Bhattacharyya sera minisée entre les distributions approximatives haut-vers-bas et bas-vers-haut. Cette approche produit des résultats de pointe en terme de modéles génératifs qui favorisent les réseaux significativement plus profonds. Elle permet aussi une inférence approximative amérliorée par plusieurs ordres de grandeur. De plus, nous introduisons un modéle génératif profond basé sur les modéles BiHM pour l'entraînement semi-supervisé.

Summary

Efficient unsupervised training and inference in deep generative models remains a challenging problem. One basic approach, called Helmholtz machine, involves training a top-down directed generative model together with a bottom-up auxiliary model used for approximate inference. Recent results indicate that better generative models can be obtained with better approximate inference procedures. Instead of improving the inference procedure, we here propose a new model, the bidirectional Helmholtz machine, which guarantees that the top-down and bottom-up distributions can efficiently invert each other. We achieve this by interpreting both the top-down and the bottom-up directed models as approximate inference distributions and by defining the model distribution to be the geometric mean of these two. We present a lower-bound for the likelihood of this model and we show that optimizing this bound regularizes the model so that the Bhattacharyya distance between the bottom-up and top-down approximate distributions is minimized. This approach results in state of the art generative models which prefer significantly deeper architectures while it allows for orders of magnitude more efficient approximate inference. Moreover, we introduce a deep generative model for semi-supervised learning problems based on BiHM models.

Acknowledgements

I would like to thank all people who supported me during my work on this thesis. Especially, I want to thank my supervisor Yoshua Bengio for his continuous support and advice and helping me to develop my background in computer science. Also, I would like to thank the rest of my thesis committee, Aaron Courville and Pascal Vincent, for their insightful comments. Thanks to my colleague Jörg Bornschein. It was a pleasure and a lot of fun to work with you!

Contents

List of Figures

List of Tables

List of Abbreviations

BiHM	bidirectional Helmholtz machine
CAE	contractive auto-encoders
DAG	directed acyclic graph
DBM	deep Boltzmann machine
DBN	deep belief network
$D_B(p, q)$	Bhattacharyya distance between p and q
$D_{KL}(p, q)$	Kullback-Leibler distance between p and q
ESS	effective sample size
$\mathcal{H}_p(X)$	entropy of X
HM	Helmholtz machine
IID	independence and identically distributed
IS	importance Sampling
IWAE	importance weighted auto-encoder
KL	Kullback-Leibler
LL	log-likelihood
MCMC	Markov Chain Monte Carlo
MLP	multilayer perceptron
MTC	manifold tangent classifier
NLL	negative log-likelihood
NVIL	neural variational inference and learning
PGM	probabilistic garaphical model
RBM	restricted Boltzmann machine
RWS	reweighted wake-sleep
SVM	support vector machine
TFD	Toronto face database
TSVM	transductive support vector machine
VAE	variational autoencoder
WS	wake-sleep

1 Introduction

Machine learning is a general framework for developing and analyzing the algorithms that can learn from data. It is widely used for applications including visual object recognition, speech perception, language comprehension, and more [23, 24, 35, 93].

1.1 Learning

Machine learning presents an interesting class of algorithms which is driven from given data and learning in the context of machine learning corresponds to extracting and modeling certain principles underlying the data. In many practical problems, the data set comprises instances of input vectors along with their corresponding target or label vectors, and a machine learning algorithm generates an output vector which is encoded in the same way as the target vector. In such problems, the learning task is to extract and model the relation between input and output values. These kind of problems are called supervised learning. Some common examples of supervised learning models are naive Bayes, support vector machines, random forests, and decision trees [15, 26, 41]. There are two main types of supervised learning problems that we can consider, classification, which aims to assign each input vector to one of a finite number of discrete categories; and regression, in which the desired output consists of one or more continuous variables.

Of course, not all the inputs can be labeled. Consider a case where there is no labeled input. For instance, it can distinguish that faces are very different from landscapes, which are very different from horses. We analyzed precisely the learning problems in which the data is unlabeled. Such problems are called unsupervised. Some of most widely used unsupervised learning problems are K-means, fuzzy clustering, hierarchical clustering [30, 77, 86, 98].

Unsupervised learning problems tend to fall into three main categories. The first category includes problems where the learning task is to discover groups of similar examples within the data. This is called clustering [30, 98]. The second category which is called density estimation, includes problems whose learning task is to determine the distribution of data within the input space [105]. Finally, in the third category there are problems whose learning goal is to explain the data as lying on a low-dimensional manifold embedded in the input space [84].

For many practical applications, however, we have to deal with cases where part of the labels is missing or obtaining the labels for the entire data set is expensive or impossible. Such problems are called called semi-supervised in which only a few inputs have targets [3, 22].

Semi-supervised learning is really important because of two main reasons besides the fact that it has many variant applications such as image search, genomics, natural language parsing, and speech analysis [29, 49, 65, 64, 82, 95]. First, it is often surprisingly effective using only a few labeled inputs and get better performance thanks to the unlabelled examples. The second reason for using this kind of learning is that in certain cases like the task of natural language parsing, it is really hard to get the labeled data or it might require experts.

As we see in all of the learning problems mentioned in this thesis, the data set has an important role and it is usually divided into two distinct sets. The first one, called the training set, is used to tune the parameters of the model and the second one, called the test set. Every machine learning algorithm can define a function whose precise form is determined during the training or learning phase. The learning phase can be viewed as an optimization problem that has an objective function (also known as cost or loss function) which is a numerical criterion for quantifying our preference for different models. In other words, the loss function can be optimized during training and one can provide the answer to the question of how good the model is in practice based on its values on the test set [8]. It is really critical that the model performs well on test set on which it is not trained. This performance is called generalization.

1.2 Generative Models

We often know in advance that we want the learned model to perform well on a particular task. It is common to consider the task of constructing a model that describes how data is generated. Such models are called generative models.

Generative models have many advantages. First, these models provide a natural way of modeling the problems. Importantly, the appropriate choice of these models are dependent on the applications. Second, generative models can be used for drawing samples from the learned distribution over the data set. If the training data consists of, for example, images, it can be used to generate textures from these images, or to solve inpainting tasks by sampling the missing or deteriorated parts of a given image from the distribution [16, 50, 62, 104].

Another advantage of generative models is that one can use then as feature extractors. For feature extraction, one makes use of the fact that many generative models comprise two types of variables. First type is called visible variables which correspond to the components of the inputs, and the second type is hidden or latent variables which correspond to the components of hidden layers. In fact,

many layers of hidden variables in which each layer captures correlations between the activities of hidden features in the layer below. If we consider a generative model with one layer of hidden variables, then after training, the expected states of the hidden variables given an input can be interpreted as the features extracted from this input. If the number of hidden units is small, this leads to low dimensional representations [112, 32, 91].

Besides the fact that generative models are able to deal well with unlabeled data, if the training set is labeled, then a generative model can learn the joint distribution over inputs and labels, and then it can be used as a learned classifier. In fact, one can sample the missing label for a represented image from the distribution or assign a new image to the class with the highest probability under the model [90, 58]. Importantly, it can also be used in semi-supervised learning problems [48].

Training good generative models and fitting them to complex and high dimensional training data is a major challenge. This is especially true for models with multiple layers of deterministic or stochastic variables, which is unfortunate because it has been argued previously that generative models with multiple hidden layers have the potential to capture higher-level abstractions [36, 2]. Such abstractions can lead us to better generalization. Although there has been progress in dealing with continous-valued latent variables, building a hierarchy of representations, especially with discrete-valued latent variables, remains a challenge [49].

One basic approach to this problem is called the Helmholtz machine [34]. In the Helmholtz machine, the generative model is a directed model that starts from some prior over latent variables at the top, down to a distribution over the data space at the bottom. Besides the generative model, Helmholtz machines contain also another model known as the approximate inference model, which runs in the opposite direction and is typically trained to efficiently infer high probability latent states given some observed data. Training a Helmholtz machine involves training an auxiliary model that helps to perform approximate inference for the generative model [34]. With the Helmholtz machine, a concept was introduced that proposed to not only fit a powerful but intractable generative model to the training data, but also to jointly train a parametric approximate inference model [34, 21]. Recent results indicate that significant improvements can be made when better approximate inference methods are used [92].

1.3 Outline of the thesis

There are many approaches that aimed at incorporating more powerful inference methods to gain better approximations of the true posterior [40, 92]. In contrast, we propose to regularize the top-down model such that the generative model stays close to the approximate inference model and vice versa. We achieve this by interpreting both generative and inference models as approximate inference models for

our actual generative model which is defined to be the geometric mean over the top-down and bottom-up approximate inference models. In this thesis, we introduce our new model and show that this model definition leads to an objective that can be interpreted as using a regularization term that encourages solutions where our generative model and approximate inference are close to each other in terms of the Bhattacharyya distance which is introduced in chapter 2.

In more detail, the thesis can be outlined as follows. Chapter 2 covers the necessary principles and concepts of machine learning that will be used in the later chapters including some needed concepts of probabilistic graphical models, neural networks, and probability theory.

In chapter 3, we propose our new model, referred to as bidirectional Helmholtz machine (BiHM), that is based on the idea that the generative model should be close to the class of distributions that can be modeled by our approximate inference distribution and that both the top-down and bottom-up distributions should contribute to the model. We achieve this by interpreting the top-down and the bottom-up directed models as approximate inference distributions and by defining the target distribution we fit to the training data to be the geometric mean of these two. We also present a lower-bound for the log-likelihood of this model and we show that maximizing this bound will pressure the model to stay close to the approximate inference distributions. Optimizing this bound maximizes the likelihood while it regularizes the model so that the Bhattacharyya distance between the bottom-up and top-down distributions is minimized. Experiments in this chapter demonstrate that we can use this approach to fit generative models with many layers of hidden binary stochastic variables to complex training distributions and that BiHMs prefer significantly deeper architectures than other approaches.

In chapter 4, we first describe a semi-supervised learning model using our generative model, BiHM. Then we analyze its properties theoretically and empirically. Using the properties of the model, a lower bound on the marginal likelihood of the model is derived. Then we show how to use an importance sampling based estimate for the gradient of this lower bound. Interestingly, this lower bound ensures that our generative model gets closer to the to the recognition model.

This thesis consists of a paper submitted to a conference and an under preparation paper. Each of them is assigned a separate chapter, and the content of them remained largely unchanged. Only minimal changes were applied to align the notation and achieve a consistent formatting throughout the thesis. Furthermore, all references were gathered in a joint bibliography in the end.

2 Background

Machine learning deals with the theoretic, algorithmic and applicative aspects of learning from a data set. Learning from a data set means that we would like to have a computer program that can learn to perform a task given this data set and build a model of the world.

In this chapter, we begin by providing an overview of some important background material regarding machine learning, probability theory, and graph theory that are required to understand most of the discussion in the remainder of this thesis. Most of the topics reviewed in this chapter are discussed in greater technical depth in [8, 51].

In section 2.1, we introduced some needed concepts of probabilistic graphical models. Then in section 2.2, neural networks including shallow and deep ones are introduced. Section 2.3 provides some basic concepts and tools in probability theory and section 2.4 describes two methods of sampling that will be used in the later chapters. In section 2.5, optimization is introduced. Finally, in section 2.6, we present a class of unsupervised artificial neural networks which is called the Helmholtz machine.

2.1 Probabilistic Graphical Models

In machine learning, it is useful to think that the data set is generated by a distribution, and probabilistic graphical models (PGMs) are widely used to represent this distribution as a graph. In fact, this graph is a factorized representation of a set of independences which hold in this distribution. Moreover, one can think of a probabilistic graphical models as a general framework for describing and applying probability theory and graph theory.

First, we will summarize some fundamental concepts from graph theory. A graph is an ordered pair $\mathcal{G} = (V, \mathcal{E})$, where V is a finite set of nodes and $\mathcal{E}$ is a set of directed edges. An edge consists of a pair of nodes uv from V that can be connected by a directed edge $u \to v$ or an undirected edge $u - v$. We say that a graph is directed if all edges are directed and undirected if all edges are undirected. In a directed graph whenever we have that $uv \in \mathcal{E}$, we say that v is the child of u in $\mathcal{G}$, and that u is the parent of v in $\mathcal{G}$. We use $Pa_{\mathcal{G}}(u)$ to denote the parents of u in $\mathcal{G}$. On the other hand, if there exists an edge between two nodes u and v in an undirected graph, i.e., $uv \in \mathcal{E}$, v belongs to the neighborhood of u and vice versa. The neighborhood

$N_u = \{v \in V : uv \in \mathcal{E}\}$ of u is defined by the set of nodes connected to u. A clique is a subset of V in which all nodes are pairwise connected. We call a sequence of nodes $v_1, v_2, \ldots, v_m \in V$, with $v_i \leftrightarrow v_{i+1} \in \mathcal{E}$ for $i = 1, \ldots, m$ a path from v_1 to v_m. A cycle in a directed graph $\mathcal{G}$ is a path from v_1 to v_m where $v_1 = v_m$. A directed acyclic graph (DAG) is a directed graph that contains no cycle.

Now, we are ready to describe the basic principles of PGMs. A general introduction to probabilistic graphical models for machine learning can, for example, be found in the book by Bishop [8]. The most comprehensive resource on graphical models is the textbook by Koller and Friedman [51]. In this section, we discuss two main classes of PGM based on directed and undirected graphs.

The first class is Bayesian networks introduced by Judea Pearl in 1985 and also known as directed graphical model whose graph is directed [79]. Formally a Bayesian network is defined by a directed acyclic graph $\mathcal{G} = (V, \mathcal{E})$ in which $V = \{X_1, \ldots, X_n\}$ where X_i is a random variable of the model for every $i = 1, \ldots, n$ and in addition, we have a set of local conditional probability distributions $p(X|Pa_{\mathcal{G}}(X))$ for every $X \in V$. Then the probability distribution over all the nodes in $\mathcal{G}$ is defined as

$$p(X) = \prod_{i=1}^{n} p(X_i|Pa_{\mathcal{G}}(X_i)),$$

where $X = (X_1, \ldots, X_n)$.

The second class is Markov networks also known as Markov random fields that go back to a paper of Kindermann in 1980 [46]. Markov networks use an undirected graph in which the edges do not carry arrows and have no directional significance. For each clique $\mathcal{C}$ in $\mathcal{G}$ a factor defined on $\mathcal{C}$ denoted by ϕ is a non-negative real function. Then, the set of all factors in $\mathcal{G}$ defines an unnormalized probability distribution as

$$\tilde{p}(X) = \prod_{\mathcal{C} \in \mathcal{G}} \phi(\mathcal{C}),$$

where $X = \cup_{\mathcal{C} \in \mathcal{G}} \mathcal{C}$. One can think of the factor ϕ on $\mathcal{C}$ representing the marginal probability distribution of the variables in $\mathcal{C}$. In such networks, we have no guarantees that $\tilde{p}(X)$ is a distribution. However, a joint distribution p over X can be defined by taking the product of the factors and then normalizing it to become a well-defined distribution. More formally,

$$p(X) = \frac{1}{Z}\tilde{p}(X),$$

where

$$Z = \sum_{X} \tilde{p}(X),$$

is a normalizing constant also known as the partition function. The bad news is that the partition function might be intractable.

It turns out that the learning tasks can be much more difficult for Markov networks than its corresponding problem for Bayesian networks. Moreover, Bayesian networks are useful for expressing probability relationships between random variables, however, Markov networks are better to express constraints between random variables.

2.2 Neural Networks

Neural networks are applicable in a wide variety of subjects like control systems, weather forecast, etc. The origin of the term "neural network" can be traced back to the work of trying to model the neuron in biological systems [28, 66, 111].

Perhaps one of the simplest neural networks in machine learning is the feedforward neural network, also known as multilayer perceptron (MLP) [88]. In such networks, the information moves through multiple layers, known as hidden or latent layers, in only forward direction to the output layer which is the last layer of the network. To formalize this intuition, assume a feedforward neural network with L layers. Then the output of this feedforward neural network is defined as

$$f_L(f_{L-1}(\dots(f_1(x)))),$$

where x is in the data set $\mathcal{D}$ and f_i is a transformation defined on $\mathbb{R}^{N_i}$, the ith hidden layer, as

$$f_i(u) = g(W_i u + b_i),$$

where W_i is an $N_{i+1} \times N_i$ matrix, b_i is a vector of size N_{i+1} and g is a real-valued function, known as activation function, defined on $\mathbb{R}^{N_{i+1}}$ for each $i = 1, \dots, L$. Moreover, W_i is called weight connection of the layer i and b_i is known as bias corresponding to the layer i for every $i = 1, \dots, L$.

Many existing machine learning algorithms fall into two main categories. The first category includes architectures for which $L = 1$. These architectures are known as shallow networks and besides a one hidden layer MLP, there are many other examples of shallow networks such as kernel regression and support vector machines [15, 71, 109]. It has been proved that any function can be approximated with any desired non-zero amount of error by a one hidden layer MLP with sufficient hidden units [17, 42, 99]. However, to extract meaningful and informative representations in such networks, we would need large amounts of data and too many hidden units [59]. In fact, these models have particularly simple analytical and computational properties. It has been shown that shallow networks are necessarily simple and are incapable of extracting informative representation and better models must then be used [2, 5].

The key to improving shallow networks is to use deep architectures which are composed of several layers of parameterized nonlinear modules. This would lead to introduce the second and much more powerful category known as deep networks. The earliest deep algorithms that had multiple layers of non-linear features can be traced back to Ivakhnenko and Lapa in 1965 [43]. As a few examples of deep models, Geoffrey Hinton and Terry Sejnowski in 1985 propose Boltzmann machines (BMs) which are undirected graphical model, in which all connections between layers are undirected [38, 31, 37]. Smolensky in 1986 proposes a particular type of Markov random field that has a two-layer architecture which is known as restricted Boltzmann machines (RBMs) [96, 2, 69]. Hinton et al. in 2006 propose deep belief networks (DBNs) which are probabilistic generative models that contain many layers of hidden variables with connections between the layers but not between units within each layer [36]. Convolutional neural networks (CNNs), DBN, stacked autoencoders, and stacked denoising autoencoders are the models that led to the current rise in popularity of deep neural networks [4, 36, 52, 107].

Unfortunately, the ability to use several layers of nonlinearity makes the objective function of a deep neural network almost always nonconvex. It often has multiple local optima or plateaus. These properties make the optimization hard in comparison with shallow models [102]. However, it has been shown that local minima are benign [18, 14, 23, 24, 35, 93].

2.3 Basic concepts in probability theory

In this thesis we focus on cases where all of the random variables are discrete, however, many of the definitions and the conclusions hold for continuous random variables as well.

Assume that p is an arbitrary probability distribution over a random vector $X \times Y$ where X and Y are also random vector. One of the important rules in probability theory is the Bayes sum rule which is:

$$p(X) = \sum_{y \in Y} p(X, y),$$

and another interesting and useful rule is known as Bayes product rule:

$$p(X, Y) = p(X|Y)p(Y),$$

where $p(X|Y)$ is called the conditional probability distribution of X given Y. We usually expect $p(X|Y)$ be different from $p(X)$, but in some situations we have $p(X|Y) = p(X)$. In this case, we say X is independent of Y. Clearly, if X is independent of Y, then Y is also independent of X.

Consider θ defined as a set of the parameter variables of a neural network, data set $\mathcal{D}$ defined as a set of random variables, and p as a probability distribution defined over $\theta \times \mathcal{D}$. Then another useful rule in dealing with probability theory is known as Bayes' theorem, which takes the form

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}. \tag{2.1}$$

The quantity $p(\theta|\mathcal{D})$ on the left-hand side of (2.1) is called posterior probability distribution, which is difficult to approximate in many problems of interest. In this posterior, the first term in the numerator is the likelihood of the data given θ, which we should be able to evaluate using exact or approximate inference. The second is the prior over parameters of the model which is usually given. The posterior is proportional to the likelihood times the prior which means we have

$$\text{posterior} \propto \text{likelihood} * \text{prior}.$$

An unfortunate general property of maximum likelihood is the phenomenon of overfitting. This occurs if performing very well on training data leads to very poor performance on examples that were not in our training set. Adopting this Bayesian approach, the overfitting problem can be avoided. However, this approach needs to make assumptions about the form of the model, and if these are wrong then the results can mislead.

One of the most commonly used operations in probability theory is called expectation. Expectation of some real function $f(x)$ under p is denoted by $\mathbb{E}_p[f]$ and given by

$$\mathbb{E}_p[f] = \sum_x p(x)f(x), \tag{2.2}$$

where p is a probability over a discrete random variable X. Intuitively, the expectation of $f(x)$ is the average value of f under p. Moreover, if f is a concave function, i.e.,

$$f(\lambda x + (1-\lambda)y) \geq \lambda f(x) + (1-\lambda)f(y),$$

for any x and y in the domain of f and $\lambda \in [0,1]$, then

$$\mathbb{E}_p[f(x)] \leq f(\mathbb{E}_p[x]).$$

This inequality is known as Jensen's inequality.

Unfortunately, in some cases we get an exponential sum of terms in equation (2.2) which is computationally interactable. One way of avoiding this problem is to approximate the expectation of f as follows

$$\mathbb{E}[f] \simeq \hat{\mathbb{E}}[f] = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}), \tag{2.3}$$

where $x^{(1)}, \ldots, x^{(N)}$ are generated samples from p. Indeed, this is often applicable in practice to approximate this expectation by generating a set of N samples, estimating the value of the function or its expectation relative to each of the generated samples, and then aggregating the results. It is also useful to indicate how far f deviates from $\mathbb{E}_p[f]$. This is called the variance of f and is defined by

$$var(f) = \mathbb{E}_p[f^2] - \mathbb{E}_p[f]^2.$$

In practice, it is typical to use the estimator $\hat{\mathbb{E}}[f]$ in equation (2.3) to estimate the value of $\mathbb{E}[f]$. The mean and variance of this estimator are the two key quantities for evaluating it. If the mean of the estimator is $\mathbb{E}[f]$, it is called an unbiased estimator, otherwise it is called biased. Importantly, its variance measures how good this estimator is. In fact, the lower the variance, the higher the probability that the estimator is close to its mean.

Another important operation is called entropy and is denoted by $\mathcal{H}_p(X)$. The entropy of X is defined to be

$$\mathcal{H}_p(X) = \sum_{x \in X} p(x) \log(\frac{1}{p(x)}),$$

where we define $0 \log(\frac{1}{0}) = 0$ and X is a discrete random variable. This idea of using the expected value of $-\log p(X)$ was developed by Claude E. Shannon in 1948 [94].

It is often really important to measure the distance between two probability distributions. In probability theory, the Kullback-Leibler (KL) divergence and the Bhattacharyya distance are two measures of the difference between two discrete or continuous probability distributions. We discuss each in turn and our attention here is on discrete distributions.

Solomon Kullback and Richard Leibler in 1951 introduced KL divergence to probability theory as a measure of dissimilarity between two probability distributions [56, 54, 55]. Let p and q be two discrete arbitrary distributions over a set X. Then the KL divergence of q from p, denoted $D_{KL}(p\|q)$, is defined to be

$$D_{KL}(p\|q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)}.$$

Interestingly, KL divergence for any two probabilities is always positive unless the two distributions are equal in which case it is zero.

Another measure is the Bhattacharyya distance which is named after Anil K. Bhattacharyya [7]. For any probability distributions p and q over a set of variables X, the Bhattacharyya distance of p and q, denoted $D_B(p, q)$, is defined as

$$D_B(p, q) = -\ln(BC(p, q)),$$

where

$$BC(p, q) = \sum_{x \in X} \sqrt{p(x)q(x)}.$$

It is also interesting to mention that these two measures can define a topology on the space of probability distributions. However, these two defined concepts are not a true metric on the space of probability distributions. In fact, KL is not symmetric and neither obey the triangular inequality.

A set of variables is called independent and identically distributed (IID) if each random variable has the same probability distribution as the others and every event is independent of any intersection of the other events.

Negative log-likelihood denoted $\ell(\theta)$, is defined as

$$\ell(\theta) = -\log p_\theta(\mathcal{D}).$$

If the data set $\mathcal{D}$ is IID, then we have

$$\ell(\theta) = -\log \prod_{x \in \mathcal{D}} p_\theta(x) = -\sum_{x \in \mathcal{D}} \log p_\theta(x). \tag{2.4}$$

Using the Bayes' theorem, one can obtain

$$p(H|\mathcal{D}) = \frac{p(\mathcal{D}|H)p(H)}{p(\mathcal{D})}.$$

This procedure is called inference. Unfortunately, only a few interesting problems have an exact solution and the methods for finding these solutions are important for solving larger problems. Some of these common methods are variable elimination and the conditioning algorithms.

Of course, one can use approximate inference methods that trade off the accuracy of the results for the ability to scale up to much larger networks. The approximation inference techniques tend to fall into two categories. The first technique is called stochastic approximation inference also called Monte Carlo methods, which are based on numerical sampling methods. Some of the most successful stochastic approximation methods are importance sampling, the Metropolis sampling, Gibbs sampling and Slice sampling.

2.4 Sampling Methods

The development of sampling methods can lead to dramatic improvements in the speed and memory capacity of computers. In this section, we briefly describe two methods to sample from our model of interest.

First one is importance sampling (IS), which is a general approach in which we generate samples from a different distribution that overweights the important regions and then adjust weights so as to get an unbiased estimator. Indeed, importance sampling gives us a recipe on how to estimate the expectation of a function f over a set of variables X for a distribution p over X. This distribution p is called the target distribution. In (2.3), we mentioned that

$$\mathbb{E}_p[f] \simeq \hat{\mathbb{E}}_p[f] = \frac{1}{N}\sum_{n=1}^{N} f(x^{(n)}),$$

where $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ are generated samples from the target distribution p. With this estimation, it might appear that generating samples from p can be difficult or require an expensive computational procedure.

An alternative solution is to generate samples from a different distribution q which is called the proposal distribution and from which it is easy to generate samples. In fact, we can reformulate $\mathbb{E}_p[f]$ in terms of the proposal distribution q as follows:

$$\mathbb{E}_p[f] = \mathbb{E}_q[f\omega],$$

where ω, called the importance weight, is defined over X as

$$\omega(x) = \frac{p(x)}{q(x)},$$

for every $x \in X$. Then the unnormalized importance sampling estimator $\hat{\mathbb{E}}_{\mathcal{D}}[f]$ of $\mathbb{E}_p[f]$ is defined as

$$\hat{\mathbb{E}}_{\mathcal{D}}[f] = \frac{1}{N}\sum_{n=1}^{N} f(x^{(n)})\omega(x^{(n)}),$$

where $x^{(1)}, x^{(2)}, \ldots, x^{(N)}$ are generated by q. Note that $\hat{\mathbb{E}}_{\mathcal{D}}[f]$ is not well defined if the denominator of $\omega(x^{(i)})$ is zero for some $i = 1, \ldots, N$.

Fortunately, unnormalized importance sampling estimator is unbiased. Of course, increasing number of samples used for this estimator can improve the quality of the results since it helps to reduce its variance and thereby reduces the harm done by a poor set of samples. There is an important measure that can tell us whether we should continue generating additional samples or not. This measure is called the effective sampling size (ESS) and is defined by

$$\widehat{ess} = \frac{\left(\sum_{i=1}^{N} \omega(x^{(i)})\right)^2}{\sum_{i=1}^{N} \omega(x^{(i)})^2} \tag{2.5}$$

In fact, larger values of $\widehat{ess}$ indicate more information extracted per sample [85].

The second sampling method is Gibbs sampling which is specifically designed for graphical models to produce samples from the joint probability distribution of multiple random variables. This method is one of the Markov chain Monte Carlo (MCMC) techniques that defines a Markov chain, or a stochastic sampling process, and attempts to generate samples from the posterior distribution. In other words, the basic idea is to construct a Markov chain by updating each variable based on its conditional distribution given the state of the others. In the following, we will describe this procedure by explaining how Gibbs sampling can be used to produce samples.

Assume that $\mathcal{G}$ is an undirected graphical model whose set of all nodes is $\{X_1, \ldots, X_N\}$ and p is the corresponding joint probability distribution of $X = (X_1, \ldots, X_N)$. Moreover, assume that the random variable X_i take values which is shown usually by x_i, in a finite set Ω for every $i = 1, \ldots, N$. Furthermore, we can consider the sequence $\{X^{(k)}\}_k$ of joint random variables $X^{(k)}$ where $X^{(k)} = (X_1^{(k)}, \ldots, X_N^{(k)})$ is called state at time $k \geq 0$ taking values in Ω^N. This sequence is called Markov chain which is a time discrete stochastic process, where the next state of the system depends only on the current state and not on the sequence of events that preceded it. We first initialize all the variables in our model with $\{x_0^{(0)}, \ldots, x_N^{(0)}\}$. Between two successive points in time, the new state of the chain is produced by the following procedure. First one of the variables X_i is randomly picked with a probability q_i given by a strictly positive probability distribution q on X. Then, the new state for X_i is sampled based on its conditional probability distribution given the state $x_{-i} = (x_j)_{j \neq i}$ of other random variables $X_{-i} = (X_j)_{j \neq i}$. We continue this process until all the variables are updated. In this algorithm a specific number of steps is not discussed and in practice it is really sensitive to this number. Fortunately, samples drawn after many steps are effectively independent of the distribution from which the initial states are generated.

2.5 Optimization

One of the optimization problems is to find parameters that minimize the negative log-likelihood function on a data set $\mathcal{D}$. In fact, the log-likelihood function measures the probability p_θ of $\mathcal{D}$ and then this probability can be used for decision making such as classification and regression.

The optimizing problem is to find θ^* such that minimize $\ell(\theta)$ as define in 2.3. In other words,

$$\theta^* = -\arg\min_\theta \sum_{x \in \mathcal{D}} \log p_\theta(x), \tag{2.6}$$

Using the properties of the logarithm function, clearly θ^* is also an optimal solution for $\arg\max_\theta p_\theta(\mathcal{D})$. One explanation of why the logarithm function used is that it is more convenient to work with $\ell(\theta)$ than $p_\theta(\mathcal{D})$, and another most likely explanation is that a product of several relatively small factors might have a numerical underflow problem which it can be avoided working with $\ell(\theta)$.

The maximum likelihood estimation wishes to tune the parameters of the model such that $\mathcal{D}$ is generated from the corresponding probability distribution of this model. As for many functions, the gradient of the negative log-likelihood estimation must be zero at its minimum point θ^*. Unfortunately in many practical problems, there is no analytical form for the minimum of the negative log-likelihood estimation. Thus, we have to resort to iterative methods, such as gradient descent for optimizing the negative log-likelihood over the parameter space.

One approach for dealing with such problems is gradient descent, also known as sequential gradient descent. We begin with any arbitrary initial point θ_0. The gradient descent algorithm updates the parameter at iteration t by

$$\theta^{t+1} = \theta^t - \eta_t \nabla_\theta \ell(\theta),$$

where $\nabla_\theta \ell(\theta)$ is the gradient of ℓ with respect to θ and $\{\eta_t\}$ is a sequence of real positive numbers which are called learning rate [8].

In practice a more efficient approach is to consider a random subset of $\mathcal{D}$ at each iteration t. Let D be this random subset which is called minibatch. Then one can update the parameter at iteration t by

$$\theta^{t+1} = \theta^t + \eta_t \nabla_\theta \sum_{X \in D} \log p_\theta(X).$$

where $\{\eta_t\}$ is the learning rate. This approach is called minibatch stochastic gradient descent.

A particular case of interest arises when the sequence of the parameters of an estimator $\{\theta^t\}_t$ converges in probability to θ^* as N increases indefinitely where N is the cardinality of $\mathcal{D}$. An estimator having this property that is called a consistent estimator or asymptotically consistent estimator [76]. In other words, the distributions of the estimates become closer to the true value of the parameter being estimated, so that the probability of the estimator being arbitrarily close to θ^* converges to one. Fortunately, maximum likelihood estimators are consistent.

The second technique is deterministic approximation inference which includes variational methods and mean field approximation. Here we just explain the variational inference approach. The key point of variational inference is to approximate the posterior distribution with a simple family of probability distributions and hope to obtain a distribution from this family that is close enough to the true posterior. As we discussed in section 2.3, one of the common tools to measure the distance of two probability distributions is KL divergence. Clearly, we can rewrite log-likelihood function as

$$\log p(\mathcal{D}) = \mathcal{L}(q) + KL(q(H)||p(H|\mathcal{D}),$$

where

$$\mathcal{L}(q) = \sum_{H} q(H) \log(\frac{p(H,\mathcal{D})}{q(H)}).$$

Importantly, $\mathcal{L}(q)$ is a lower-bound of $\log p(\mathcal{D})$ and maximizing $\mathcal{L}(q)$ is equivalent to minimizing the KL divergence $KL(q(H)||p(H|\mathcal{D})$. This approach is called variational inference.

2.6 Helmholtz machine

We now present a class of unsupervised artificial neural networks proposed by Hinton and Dayan in 1995 that can combine stochastic neural computation, unsupervised learning, and the ability to perform probabilistic inference [34]. This class is known as Helmholtz machine (HM). A HM network consists of a top-down or feedback connections to build the probability distribution which is called generative model and in addition bottom-up connections form a feedforward network to define another probability distribution which is called the recognition model. Moreover, both of these models are Bayesian networks and units in one layer are conditionality independent of the others within the same layer given the previous layer.

As an example of HM, consider the network in figure 2.1. In this case, HM is composed of three layers: the input layer x which is given and observed; the first hidden layer h_1; and the second hidden layer h_2. One can define the generative model p with the parameters θ and the recognition model q with parameters ϕ.

To provide a more formal description of HM, assume p_θ and q_ϕ are the generative and recognition models over $\mathcal{D} \times h$ respectively where $\mathcal{D}$ is the data set and h is the set of all hidden layers. Moreover, assume that we wish to find θ^* such that

$$\theta^* = \arg\max_{\theta} \log p_\theta(x),$$

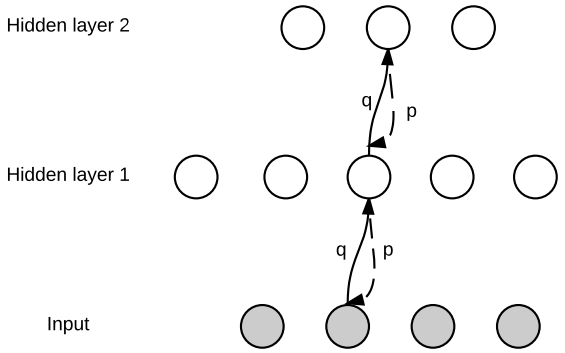

Figure 2.1 – It is an example of HM that has an input layer which is observed and two hidden layers on top of each other. The layers are connected through h both recognition model q with parameters ϕ shown with solid lines and generative model p with parameters θ shown as dashed lines.

where

$$\log p_\theta(x) = \log(\sum_h p_\theta(h)\ p_\theta(x|h)). \tag{2.7}$$

Now set

$$E_\theta(h, x) = -\log(p_\theta(h)\ p_\theta(x|h)). \tag{2.8}$$

Using Bayes' theorem, it follows that

$$p_\theta(h|x) = \frac{p_\theta(h)p_\theta(x|h)}{\sum_{h'} p_\theta(h')p_\theta(x|h')} = \frac{\exp(-E_\theta(h,x))}{\sum_{h'} \exp(-E_\theta(h',x))}. \tag{2.9}$$

Using equations (2.8) and (2.9), we can reformulate equation (2.10) as

$$
\begin{aligned}
\log p_\theta(x) &= \sum_h q_\phi(h|x) \log p(x) = \sum_h q_\phi(h|x) \log(\sum_{h'} \exp(-E_\theta(h', x))) \\
&= -\sum_h q_\phi(h|x)(E_\theta(h, x) - E_\theta(h, x)) + \sum_h q_\phi(h|x) \log(\sum_{h'} \exp(-E_\theta(h', x))) \\
&= -\sum_h q_\phi(h|x) E_\theta(h, x) + \sum_h q_\phi(h|x)[E_\theta(h, x) + \log(\sum_{h'} \exp(-E_\theta(h', x)))] \\
&= -\sum_h q_\phi(h|x) E_\theta(h, x) - \sum_h q_\phi(h|x) \log p_\theta(h|x).
\end{aligned}
\tag{2.10}
$$

So

$$
\log p_\theta(x) = -\sum_h q_\phi(h|x) E_\theta(h, x) - \sum_h q_\phi(h|x) \log p_\theta(h|x).
$$

Thus, we obtain the following

$$
\begin{aligned}
\log\ p_\theta(x) &= -\sum_h q_\phi(h|x) E_\theta(h, x) - \sum_h q_\phi(h|x) \log q_\phi(h|x) \\
&\quad + \sum_h q_\phi(h|x) \log(\frac{q_\phi(h|x)}{p_\theta(h|x)}) \\
&= -\mathcal{F}(x; \theta, \phi) + \sum_h q_\phi(h|x) \log(\frac{q_\phi(h|x)}{p_\theta(h|x)}),
\end{aligned}
$$

where the first term $\mathcal{F}(x; \theta, \phi)$ is called the Helmholtz energy function and defined as

$$
\mathcal{F}(x; \theta, \phi) = \sum_h q_\phi(h|x) E_\theta(h, x) + \sum_h q_\phi(h|x) \log q_\phi(h|x)
$$

relates to concepts from statistical physics, and the latter term is the KL divergence of $p_\theta(h|x)$ from $q_\phi(h|x)$. Since the KL divergence term can not be negative, it means that we can obtain a lower-bound of the log probability of the data given the model $\log\ p_\theta(x)$. In fact, we have

$$
\log\ p_\theta(x) \geq -\mathcal{F}(x; \theta, \phi).
$$

Therefore, to maximize the log-likelihood, we can minimize $\mathcal{F}(x; \theta, \phi)$. In fact, the recognition model is encouraged to be a good approximation to the true posterior distribution and moreover, the generative model is encouraged to be close to the recognition model. These models are usually trained using an unsupervised learning algorithm, such as the wake-sleep algorithm [34].

In the next chapter, we propose a new approach to generative modeling based on fitting the geometric mean of approximate inference and generative distributions.

Prologue to First Article

Bidirectional Helmholtz Machines, J. Bornschein, S. Shabanian, A. Fischer, Y. Bengio (2016). In *International Conference on Machine Learning (ICML)*.

Personal Contribution: J. Bornschein had the main idea. I ran all the UCI experiments and some explorative MNIST experiments. In particular, I found the best hyperparameters for UCI datasets. All authors contributed to the writing of the paper.

3 Bidirectional Helmholtz Machines

The key insight is whether we can achieve a better inference approximation by introducing a new generative model that is reasonably closer to the approximate inference model.

After a brief overview over training deep generative models in section 3.1, section 3.2 introduces the bidirectional Helmholtz machine (BiHM) model in detail and discusses important theoretical properties. In section 3.3, we will explain how to perform importance sampling-based training and inference. In section 3.4, we show how our model can be used to solve inpainting tasks by sampling the missing parts of a given image from the distribution. In section 3.5, we introduce a new way to estimate the partition function of BiHM model. The ability of BiHM to model complex distributions is demonstrated empirically in section 3.6. Finally in section 3.7, we empirically analyze our importance sampling-based estimates.

3.1 Introduction and background

Training good generative models and fitting them to complex and high dimensional training data with probability mass in multiple disjunct locations remains a major challenge. This is especially true for models with multiple layers of deterministic or stochastic variables, which is unfortunate because it has been argued previously that deeper generative models have the potential to capture higher-level abstractions and thus generalize better [36, 2]. Although there has been progress in dealing with continous-valued latent variables, building a hierarchy of representations, especially with discrete-valued latent variables, remains a challenge [49].

With the Helmholtz machine, a concept was introduced that proposed to not only fit a powerful but intractable generative model $p(\mathbf{x}, \mathbf{h})$ to the training data, but also to jointly train a parametric approximate inference model $q(\mathbf{h}|\mathbf{x})$ [34, 21]. The q model would be used to efficiently perform approximate inference over the latent variables $\mathbf{h}$ of the generative model given an observed example $\mathbf{x}$. This basic idea has been applied and enhanced many times; initially with the wake-sleep (WS) algorithm [34] and more recently with the variational autoencoder (VAE) [49], stochastic backpropagation and approximate inference in deep generative models [82], neural variational inference and learning (NVIL) [68] and reweighted wake-sleep (RWS) [10]. Most of these approaches rely on the variational bound to perform

appropriate inference and to obtain an objective function that contains the parameters of both the generative model p and the approximate inference model q in one joint expression (e.g. WS, VAE and NVIL).

Recent results indicate that significant improvements can be made when better approximate inference methods are used: Saliman et. al. [92] for example presented an iterative inference procedure that improves the samples from q by employing a learned MCMC transition operator. Burda et al. [11] present the importance weighted auto encoder (IWAE), an improved VAE that, similarly to RWS, uses multiple samples from q to calculate gradients. And RWS already reported that autore-gressive q distributions lead to noticeable improvements. In contrast to these previous approaches, we here propose to interpret both p and q as approximate inference models for our actual generative model $p^*(\mathbf{x}, \mathbf{h})$. We define the target distribution p^* to be the geometric mean over the top-down and bottom-up approximate inference models, i.e.

$$p^*(\mathbf{x}, \mathbf{h}) = \frac{1}{Z}\sqrt{p(\mathbf{x}, \mathbf{h})q(\mathbf{x}, \mathbf{h})},$$

where Z is the normalization constant.

The motivation behind this definition is to ensure that the intractable generative model p^* stays close to the approximate inference models we have at our disposal. In fact, we show that the proposed objective can be interpreted as adding a regularization term to the log-likelihood objective towards solutions where p and q are close to each other in terms of the Bhattacharyya distance.

3.2 Model definition and properties

We introduce the concept by defining a joint probability distribution over three variables, an observed random variable vector $\mathbf{x}$ and two latent random variable vectors $\mathbf{h}_1$ and $\mathbf{h}_2$. Analogous to a Deep Boltzmann Machine, we think of these as layers in a neural network with links between $\mathbf{x}$ and $\mathbf{h}_1$ on the one side, and $\mathbf{h}_1$ and $\mathbf{h}_2$ on the other side. We will present the approach for the specific case of an architecture with two hidden layers, but it can be applied to arbitrary graphs of variables without loops. It can especially be used to train architectures with more than two stacked layers of latent variables.

From now on, we use $\mathbf{x}$ to denote an observed random variable vector and $\mathbf{h}_1, \mathbf{h}_2$ to denote a pair of latent random variable vectors. Also assume that p and q are directed graphical models from $\mathbf{h}_2$ to $\mathbf{x}$ and vice versa that can be factorized as

$$p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = p(\mathbf{h}_2)\ p(\mathbf{h}_1|\mathbf{h}_2)\ p(\mathbf{x}|\mathbf{h}_1),$$

and

$$q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = q(\mathbf{x})\ q(\mathbf{h}_1|\mathbf{x})\ q(\mathbf{h}_2|\mathbf{h}_1),$$

where the prior distribution $p(\mathbf{h}_2)$ and all conditional distributions belong to parametrized families of distributions which can be evaluated and sampled from efficiently. We usually do not assume an explicit form for $q(\mathbf{x})$.

The associated joint probability p^* to p and q is given by

$$p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = \frac{1}{Z}\sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}\;\;, \tag{3.1}$$

where Z is a normalization constant. Moreover, we assume that the marginal distributions $p^*(\mathbf{x})$ and $q(\mathbf{x})$ are the same, and then define

$$\begin{aligned} q(\mathbf{x}) = p^*(\mathbf{x}) &= \sum_{\mathbf{h}_1, \mathbf{h}_2} p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \\ &= \frac{\sqrt{q(\mathbf{x})}}{Z} \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{h}_1|\mathbf{x}) q(\mathbf{h}_2|\mathbf{h}_1)} \\ &= \left(\frac{1}{Z} \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{h}_1|\mathbf{x}) q(\mathbf{h}_2|\mathbf{h}_1)} \right)^2 . \end{aligned}$$

Based on the definition of the normalization constant Z, it is guaranteed that

$$\sum_{\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2} p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = 1$$

In general, for any two arbitrary vectors a and b of a real inner product space, the Cauchy-Schwarz inequality states that

$$|\langle a, b \rangle|^2 \leq \langle a, a \rangle \langle b, b \rangle,$$

where $\langle \cdot, \cdot \rangle$ is the inner product, and the equality holds if and only if a and b are linearly dependent. Consider the case where $a = \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}$ and $b = \sqrt{q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}$. In this case, we can prove that the following property

$$\begin{aligned} Z &= \sum_{\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}, \\ &\leq \sum_{\mathbf{x}, \mathbf{h}_1} \sqrt{\sum_{\mathbf{h}_2} p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \sum_{\mathbf{h}_2} q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)} = \sum_{\mathbf{x}, \mathbf{h}_1} \sqrt{p(\mathbf{x}, \mathbf{h}_1) q(\mathbf{x}, \mathbf{h}_1)}, \\ &\leq \sum_{\mathbf{x}} \sqrt{\sum_{\mathbf{h}_1} p(\mathbf{x}, \mathbf{h}_1) \sum_{\mathbf{h}_1} q(\mathbf{x}, \mathbf{h}_1)} = \sum_{\mathbf{x}} \sqrt{p(\mathbf{x}) q(\mathbf{x})} \leq 1, \end{aligned}$$

holds on BiHM models. Furthermore, we see that $Z = 1$ only if

$$p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2),$$

for every $\mathbf{x}, \mathbf{h}_1$, and $\mathbf{h}_2$.

This upper bound for Z which is the denominator in equation (3.2), provides a lower-bound $\tilde{p}^*(\mathbf{x})$ for the ratio as follows:

$$\begin{aligned}\tilde{p}^*(\mathbf{x}) &= \left(\sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{h}_1|\mathbf{x})\ q(\mathbf{h}_2|\mathbf{h}_1)} \right)^2 \\ &= Z^2\ p^*(\mathbf{x}) \leq p^*(\mathbf{x}). \end{aligned} \tag{3.2}$$

This suggests that the model distribution $p^*(\mathbf{x})$ can be fitted to some training data by maximizing the bound of the log-likelihood $\log \tilde{p}^*(\mathbf{x})$ instead of $\log p^*(\mathbf{x})$, as we elaborate in the following section. Since $\log \tilde{p}^*(\mathbf{x})$ can reach the maximum only when $Z \to 1$, the model is implicitly pressured to find a maximum likelihood solution that yields

$$p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \approx q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \approx p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2),$$

for every $\mathbf{x}, \mathbf{h}_1$, and $\mathbf{h}_2$.

3.2.1 Alternative view

We now turn our attention to the model log-likelihood $\log p^*(\mathbf{x})$. It can be decomposed into two terms

$$\begin{aligned}\log p^*(\mathbf{x}) &= 2 \log \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{h}_1|\mathbf{x}) q(\mathbf{h}_2|\mathbf{h}_1)} - 2 \log Z \\ &= \log \tilde{p}^*(\mathbf{x}) - 2 \log Z = \log \tilde{p}^*(\mathbf{x}) + 2\, D_B(p, q) \\ &\geq \log \tilde{p}^*(\mathbf{x}), \end{aligned} \tag{3.3}$$

where Z is given by

$$Z = \sum_{\mathbf{x}', \mathbf{h}'_1, \mathbf{h}'_2} \sqrt{p(\mathbf{x}', \mathbf{h}'_1, \mathbf{h}'_2) q(\mathbf{x}', \mathbf{h}'_1, \mathbf{h}'_2)}.$$

We clearly see that the proposed training objective $\log \tilde{p}^*(\mathbf{x})$ corresponds precisely to the correct (but intractable) log-likelihood $\log p^*(\mathbf{x})$ minus 2 times the Bhattacharyya distance $D_B(p, q)$ defined in section 2.3. Of course, maximizing $\log \tilde{p}^*(\mathbf{x})$ is maximizing the true log-likelihood and minimizing the distance between p and q.

It is interesting to compare this approach to variational approach which is a commonly used method for optimizing the log-likelihood. In variational approach the marginal probability $\log p(\mathbf{x})$ of some model containing latent variables $\mathbf{h}$ is rewritten in terms of the KL-divergence $D_{KL}(q(\mathbf{h}|\mathbf{x}) \,||\, p(\mathbf{h}|\mathbf{x}))$ to obtain a lower-bound as follows:

$$\begin{aligned} \log p(\mathbf{x}) &= \mathbb{E}_{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x})}[\log p(\mathbf{x},\mathbf{h}) - \log q(\mathbf{h}|\mathbf{x})] + D_{KL}(q(\mathbf{h}|\mathbf{x}) \,||\, p(\mathbf{h}|\mathbf{x})) \\ &\geq \mathbb{E}_{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x})}[\log p(\mathbf{x},\mathbf{h}) - \log q(\mathbf{h}|\mathbf{x})]. \end{aligned} \tag{3.4}$$

Analogous to variational methods that maximize the lower bound (3.4), we can thus maximize $\log \tilde{p}^*(\mathbf{x})$, and it will tighten the bound as $D_B(p,q)$ approaches zero.

At first glance, this seems very similar to the variational lower bound. There are, however, some important conceptual differences. First, the KL-divergence in variational methods measures the distance between distributions given some training data. The Bhattacharyya distance here in contrast quantifies a property of the model $p^*(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)$ independently of any training data. In fact, we saw that $D_B(p,q) = -\log Z$.

Second, the variational lower bound is typically used to construct approximate inference algorithms. We here use our bound $\tilde{p}^*(\mathbf{x})$ just to remove the normalization constant Z from our target distribution $p^*(\mathbf{x},\mathbf{h}_1,\mathbf{h}_2)$. Even after applying the lower-bound, we still have to tackle the inference problem which manifests itself in form of the full combinatorial sum over $\mathbf{h}_1$ and $\mathbf{h}_2$ in equation (3.2). Although it seems intuitively reasonable to use a variational approximation on top of the bound $\tilde{p}^*(\mathbf{x})$ we will here not follow this direction but rather use importance sampling to perform approximate inference and learning (see section 3.3). Combining a variational method with the bound $\tilde{p}^*(\mathbf{x})$ is therefore subject to future work.

We can also argue that optimizing $\log \tilde{p}^*(\mathbf{x})$ instead of $\log p^*(\mathbf{x})$ is beneficial in the light of the original goal we formulated in section 3.1: To learn a generative model $p^*(\mathbf{x})$ that is regularized to be close to the model q which we use to perform approximate inference for p^*. Let us assume we have two equally well trained models $p^*_{\boldsymbol{\theta}_1}$ and $p^*_{\boldsymbol{\theta}_2}$, i.e., in expectation over the empirical distribution

$$\mathbb{E}[\log p^*_{\boldsymbol{\theta}_1}(\mathbf{x})] = \mathbb{E}[\log p^*_{\boldsymbol{\theta}_2}(\mathbf{x})],$$

but the expected bound $\tilde{p}^*(\mathbf{x})$ for the first model is closer to the log-likelihood than the expected bound for the second model

$$\mathbb{E}[\log \tilde{p}^*_{\boldsymbol{\theta}_1}(\mathbf{x})] > \mathbb{E}[\log \tilde{p}^*_{\boldsymbol{\theta}2}(\mathbf{x})].$$

Using equation (3.3) we see that $D_B(p_{\boldsymbol{\theta}_1}, q_{\boldsymbol{\theta}_1}) < D_B(p_{\boldsymbol{\theta}_2}, q_{\boldsymbol{\theta}_2})$ which indicates that $q_{\boldsymbol{\theta}_1}$ is closer to $p^*_{\boldsymbol{\theta}_1}$ than $q_{\boldsymbol{\theta}_2}$ is to $p^*_{\boldsymbol{\theta}_2}$ in the sense of Bhattacharyya measure. According to our original goal, we thus prefer solution $p^*_{\boldsymbol{\theta}_1}$ where the bound $\tilde{p}^*(\mathbf{x})$ is maximized and the distance $D_B(p,q)$ minimized.

Algorithm 1 Learning $p^*(\mathbf{x})$ using importance sampling with q as proposal

for number of training iterations **do**

- Sample $\mathbf{x}$ from the training distribution (i.e. $\mathbf{x} \sim \mathcal{D}$)

for $k = 1, 2, \ldots, K$ **do**

- Sample $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1^{(k)}|\mathbf{x})$; for each layer $l = 2$ to L sample

$$\mathbf{h}_l^{(k)} \sim q(\mathbf{h}_l|\mathbf{h}_{l-1}^{(k)})$$

- Compute $q(\mathbf{h}^{(k)}|\mathbf{x})$ and $p(\mathbf{x}, \mathbf{h}^{(k)})$ for $\mathbf{h}^{(k)} = (\mathbf{h}_1^{(k)}, \ldots, \mathbf{h}_L^{(k)})$

end for

- Compute unnormalized importance weights

$$\omega_k = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}^{(k)})}{q(\mathbf{h}^{(k)}|\mathbf{x})}}$$

- Normalize the weights $\tilde{\omega}_k = {}^{\omega_k}/_{\sum_{k'} \omega_{k'}}$
- Update parameters of p and q : Gradient decent with gradient estimator

$$2 \sum_k \tilde{\omega}_k \frac{\partial \log p^*(\mathbf{x}, \mathbf{h}^{(k)})}{\partial \boldsymbol{\theta}}$$

end for

In this algorithm, $\mathcal{D}$ is training data set, L is the number of layers, K is the number of generated samples, and $\boldsymbol{\theta}$ denotes the set of the parameters of p and q.

Note that the decomposition (3.3) also emphasizes why our recursive definition

$$q(\mathbf{x}) = \sum_{\mathbf{h}_1, \mathbf{h}_2} p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$$

is a consistent and reasonable one. Clearly, minimizing $D_B(p, q)$ during learning means that the joint distributions $p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$ and $q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)$ approach each other. This implies that the marginals $p(\mathbf{h}_l)$ and $q(\mathbf{h}_l)$ become more similar for every $l = 1, 2$. This also implies $p(\mathbf{x}) \approx q(\mathbf{x})$ in the limit of $D_B(p, q) \to 0$; a requirement that most simple parametrized distributions $q(\mathbf{x})$ could never fulfill.

Although we have focused our formal definitions on discrete variables, the same analysis applies to the continuous variables.

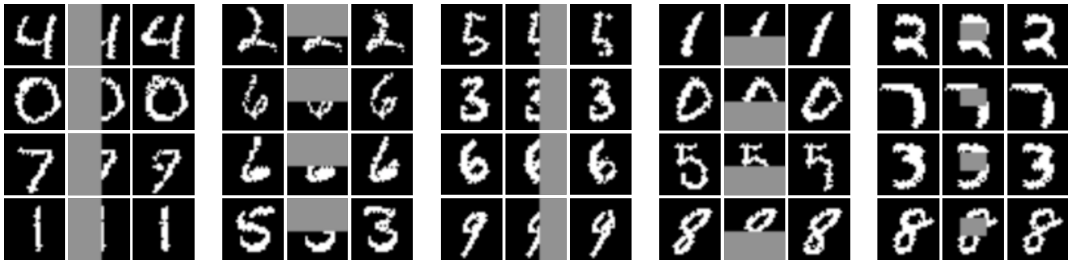

Figure 3.1 – Inpainting of binarized MNIST digits. The left column in each block shows the original digit randomly sampled from the MNIST test set; the first column shows the masked version presented to the algorithm, and the next column show different, independent reconstructions after 100 Gibbs iterations (see section 3.4). All images were selected randomly.

3.3 Inference and training with IS

Based on the construction of $p^*(\mathbf{x})$ outlined in the previous section, we can define a wide range of possible models. Furthermore, we have a wide range of potential training and appropriate inference methods we could employ to maximize $\log \tilde{p}^*(\mathbf{x})$.

Until now, we did not specify the actual choice of probability distributions that we use. We now discuss one possible choice of our conditional distributions and top-level prior. In this section, we concentrate on a binary observed random variable vector $\mathbf{x}$ and binary latent random variable vector $\mathbf{h}_l$ for $l = 1, \ldots, L$ where L is the number of layers. It is often a notational convenience to set $\mathbf{h}_0 = \mathbf{x}$. We model all our conditional distributions by simple sigmoid belief network layers, e.g.,

$$p(\mathbf{h}_{l-1} \,|\, \mathbf{h}_l) = \prod_i \mathcal{B}(h_{l-1,i} \,|\, \sigma(W_{l,i}\, \mathbf{h}_l + b_{l,i}))$$

where $\mathcal{B}(\cdot \,|\, c)$ refers to the Bernoulli distribution with parameter c, $W_{l,i}$ are the connection weights between layer $\mathbf{h}_l$ and latent variable $h_{l-1,i}$ and $b_{l,i}$ is the bias of $h_{l-1,i}$ for every $l = 1, \ldots, L$ and $\sigma(\cdot)$ which is called sigmoid function, is a real valued function that maps every $a \in \mathbb{R}$ to

$$\frac{1}{1 + \exp(-a)}.$$

For our top-level prior $p(\mathbf{h}_L)$, we use a factorized Bernoulli distribution:

$$p(\mathbf{h}_L) = \prod_i \mathcal{B}(h_{L,i} \,|\, \sigma(b_{L,i})).$$

Similarly, we consider

$$q(\mathbf{h}_l \,|\, \mathbf{h}_{l-1}) = \prod_i \mathcal{B}(h_{l,i} \,|\, \sigma(W'_{l,i}\, \mathbf{h}_{l-1} + b'_{l,i}))$$

where $W'_{l,i}$ are the connection weights between layer $\mathbf{h}_{l-1}$ and latent variable $h_{l,i}$, and $b'_{l,i}$ is the bias of $h_{l,i}$ for every $l = 1, \ldots, L$.

We form an estimate of $\tilde{p}^*(\mathbf{x})$ by using importance sampling instead of the exhaustive sum over $\mathbf{h}_1$ and $\mathbf{h}_2$ in equation (3.2). We explain how to estimate $\tilde{p}^*(\mathbf{x})$ for the specific BiHM model with $L = 2$, but the ideas can be generalized to deeper network structures. We use $q(\mathbf{h}_1|\mathbf{x})q(\mathbf{h}_2|\mathbf{h}_1)$ as the proposal distribution which is by construction easy to evaluate and to sample from. Using basic convergence bound, we can estimate $\tilde{p}^*(\mathbf{x})$ as

$$\begin{aligned}
\tilde{p}^*(\mathbf{x}) &= \left(\sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)\; q(\mathbf{h}_1|\mathbf{x})\; q(\mathbf{h}_2|\mathbf{h}_1)} \right)^2 \\
&= \left(\mathbb{E}^{\mathbf{h}_2 \sim q(\mathbf{h}_2|\mathbf{h}_1)}_{\mathbf{h}_1 \sim q(\mathbf{h}_1|\mathbf{x})} \left[\sqrt{\frac{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}{q(\mathbf{h}_1|\mathbf{x})\; q(\mathbf{h}_2|\mathbf{h}_1)}} \right] \right)^2 \\
&\simeq \left(\frac{1}{K} \sum_{k=1}^{K} \sqrt{\frac{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)})}{q(\mathbf{h}_1^{(k)}|\mathbf{x})\; q(\mathbf{h}_2^{(k)}|\mathbf{h}_1^{(k)})}} \right)^2 ,
\end{aligned} \tag{3.5}$$

with $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1|\mathbf{x})$ and $\mathbf{h}_2^{(k)} \sim q(\mathbf{h}_2|\mathbf{h}_1^{(k)})$, for any $k = 1, \ldots, K$ where K is the number of samples.

Algorithm 2 Sampling from $p^*(\mathbf{x}, \mathbf{h})$

• Draw $(\mathbf{x}, \mathbf{h}_1, \ldots, \mathbf{h}_L) \sim p(\mathbf{x}, \mathbf{h}_1, \ldots, \mathbf{h}_L)$ from the top-down model to initialize state

for number of iterations **do**

  for $l = \underbrace{1, 3, \ldots, L-1}_{odd\,layers}, \underbrace{2, \ldots, L}_{even\,layers}$ **do**

    for $k = 1, 2, \ldots, K$ **do**

      • Draw proposal sample $\mathbf{h}_l^{(k)} \sim \frac{1}{2}p(\mathbf{h}_l|\mathbf{h}_{l+1}) + \frac{1}{2}q(\mathbf{h}_l|\mathbf{h}_{l-1})$

      • Compute

$$\omega^{(k)} = \frac{\sqrt{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1})p(\mathbf{h}_{l-1}|\mathbf{h}_l^{(k)})q(\mathbf{h}_{l+1}|\mathbf{h}_l^{(k)})q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})}}{(p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) + q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}))}$$

    end for

    • Randomly pick $\mathbf{h}_l \in \{\mathbf{h}_l^{(1)}, \ldots \mathbf{h}_l^{(K)}\}$ with probability proportional to $\{\omega^{(1)}, \ldots, \omega^{(K)}\}$

  end for

  for $k = 1, 2, \ldots, K$ **do**

    • Draw proposal sample $\mathbf{x}^{(k)} \sim p(\mathbf{x}|\mathbf{h}_1)$

    • Compute estimated marginals $\tilde{p}^*(\mathbf{x}^{(k)})$ using equation (3.5)

    • Compute importance weights

$$\omega^{(k)} = \sqrt{\tilde{p}^*(\mathbf{x}^{(k)})q(\mathbf{h}_1|\mathbf{x}^{(k)})/p(\mathbf{x}^{(k)}|\mathbf{h}_1)}$$

  end for

  • Randomly pick $\mathbf{x} \in \{\mathbf{x}^{(1)}, \ldots \mathbf{x}^{(K)}\}$ with probability proportional to $\{\omega^{(1)}, \ldots, \omega^{(K)}\}$

end for

To simplify the notation, we here assume $p(\mathbf{h}_L|\mathbf{h}_L + 1) = p(\mathbf{h}_L)$. An algorithm for inpainting is obtained by drawing the initial state from $q(\mathbf{h}|\mathbf{x})$ instead from $p(\mathbf{x}, \mathbf{h})$ and by keeping the known elements of $\mathbf{x}$ fixed when sampling from $p(\mathbf{x}|\mathbf{h}_1)$. Moreover, $\mathcal{D}$ is training data set, L is the number of layers, K is the number of samples, and $\boldsymbol{\theta}$ denotes the set of the parameters of p and q.

Using the same approach, we can also derive the well known estimator for the marginal probability of a datapoint under the top-down generative model p :

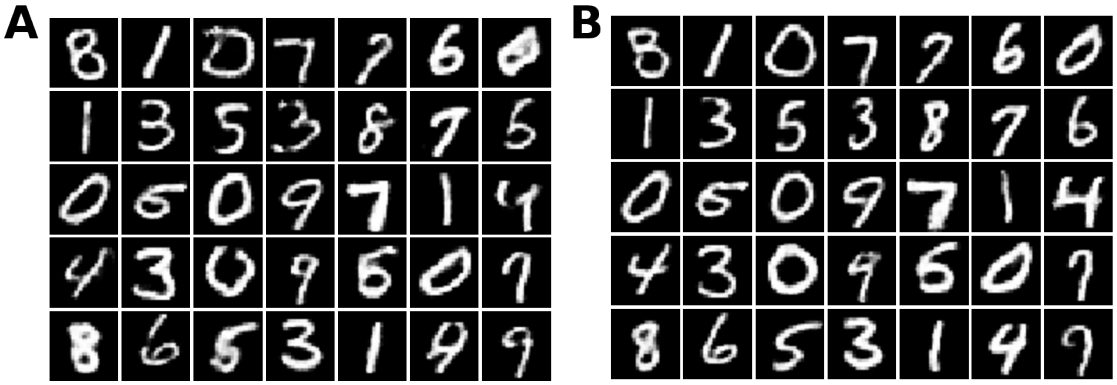

Figure 3.2 – MNIST experiments: **A**) Random samples from top-down model $p(\mathbf{x})$. **B**) Generally improved samples after running 10 iterations of algorithm 2 starting from the samples in **A**, i.e., approximately from the joint model $p^*(\mathbf{x})$. In **A** and **B** we show expected samples instead of sampling from the bottom-most Bernoulli distribution.

$$
\begin{aligned}
p(\mathbf{x}) &= \mathbb{E}^{\mathbf{h}_2 \sim q(\mathbf{h}_2|\mathbf{h}_1)}_{\mathbf{h}_1 \sim q(\mathbf{h}_1|\mathbf{x})} \left[\frac{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}{q(\mathbf{h}_1|\mathbf{x})\ q(\mathbf{h}_2|\mathbf{h}_1)} \right] \\
&\simeq \frac{1}{K} \sum_{k=1}^{K} \frac{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)})}{q(\mathbf{h}_1^{(k)}|\mathbf{x})\ q(\mathbf{h}_2^{(k)}|\mathbf{h}_1^{(k)})} = \tilde{p}(\mathbf{x}) \ , \qquad (3.6)
\end{aligned}
$$

where K is the number of samples.

Comparing (3.5) and (3.6) and making use of Jensen's inequality it becomes clear that $p(\mathbf{x}) \geq \tilde{p}^*(\mathbf{x})$. Analogous to the parameter updates in RWS [10], we can derive an importance sampling based estimate for the log-likelihood gradient with respect to the parameters of p and q which is jointly denoted by $\boldsymbol{\theta}$. Moreover, it can be used to optimize our proposed regularized objective. To provide a formal derivation, consider $\mathbf{h}$ jointly denotes the binary hidden variables of all hidden layers that we introduced earlier. Then we can write

$$
\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}} \log \tilde{p}^*(\mathbf{x}) &= \frac{\partial}{\partial \boldsymbol{\theta}} \log \left(\sum_{\mathbf{h}} \sqrt{p(\mathbf{x}, \mathbf{h}) q(\mathbf{h}|\mathbf{x})} \right)^2 \\
&= \frac{2}{\sum_{\mathbf{h}} \sqrt{p(\mathbf{x}, \mathbf{h}) q(\mathbf{h}|\mathbf{x})}} \sum_{\mathbf{h}'} \sqrt{p(\mathbf{x}, \mathbf{h}') q(\mathbf{h}'|\mathbf{x})} \frac{\partial}{\partial \boldsymbol{\theta}} \log \sqrt{p(\mathbf{x}, \mathbf{h}') q(\mathbf{h}'|\mathbf{x})} \\
&\simeq 2 \sum_{k=1}^{K} \tilde{\omega}_k \frac{\partial}{\partial \boldsymbol{\theta}} \log \sqrt{p(\mathbf{x}, \mathbf{h}^{(k)}) q(\mathbf{h}^{(k)}|\mathbf{x})} \ ,
\end{aligned}
$$

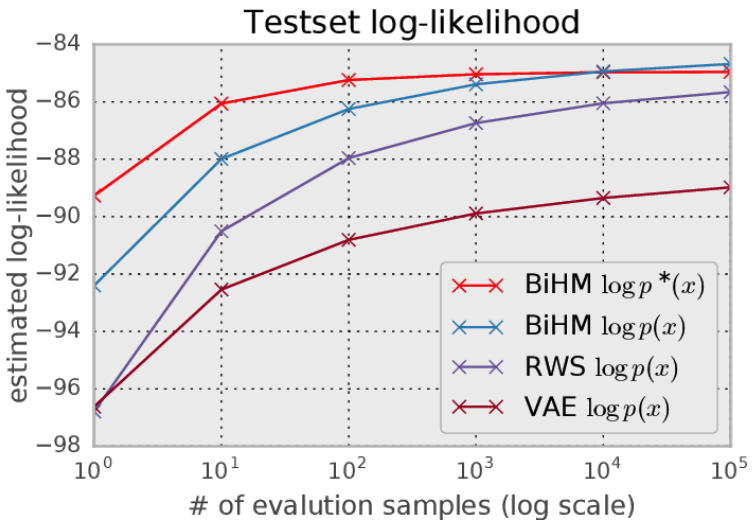

Figure 3.3 – Sensitivity of the test set log-likelihood estimates to the number of samples K. We plot the $\log p(\mathbf{x})$ and $\log p^*(\mathbf{x})$ estimates of our best BiHM model together with the $\log p(\mathbf{x})$ estimates of our best RWS and VAE models.

where $\mathbf{h}^{(k)} \sim q(\mathbf{h} \,|\, \mathbf{x})$ and the importance weight $\tilde{\omega}_k$ is defined

$$\tilde{\omega}_k = \frac{\omega_k}{\sum_{k'} \omega_{k'}},$$

where $\omega_k = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}^{(k)})}{q(\mathbf{h}^{(k)} \,|\, \mathbf{x})}}$ for $k = 1, \ldots, K$ and K is the number of samples. Going back to our specific BiHM model with $L = 2$, we see that

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log \tilde{p}^*(\mathbf{x}) \simeq 2 \sum_{k=1}^{K} \tilde{\omega}_k \frac{\partial}{\partial \boldsymbol{\theta}} \log \sqrt{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}) q(\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)} | \mathbf{x})} \tag{3.7}$$

where $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1 \,|\, \mathbf{x})$, $\mathbf{h}_2^{(k)} \sim q(\mathbf{h}_2 \,|\, \mathbf{h}_1^{(k)})$, and importance weights

$$\tilde{\omega}_k = \frac{\omega_k}{\sum_{k'} \omega_{k'}} \tag{3.8}$$

where $\omega_k = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)})}{q(\mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)} | \mathbf{x})}}$ for any k in $\{1, \ldots, K\}$, and K is the number of samples.

In contrast to VAEs, the updates do not require any form of backpropagation through more than one layer because, as far as the gradient computation

$$\frac{\partial}{\partial \boldsymbol{\theta}} \log p^*(\mathbf{x}, \mathbf{h}_1^{(k)}, \mathbf{h}_2^{(k)}),$$

is concerned, these samples are considered fully observed. The gradient approxi-

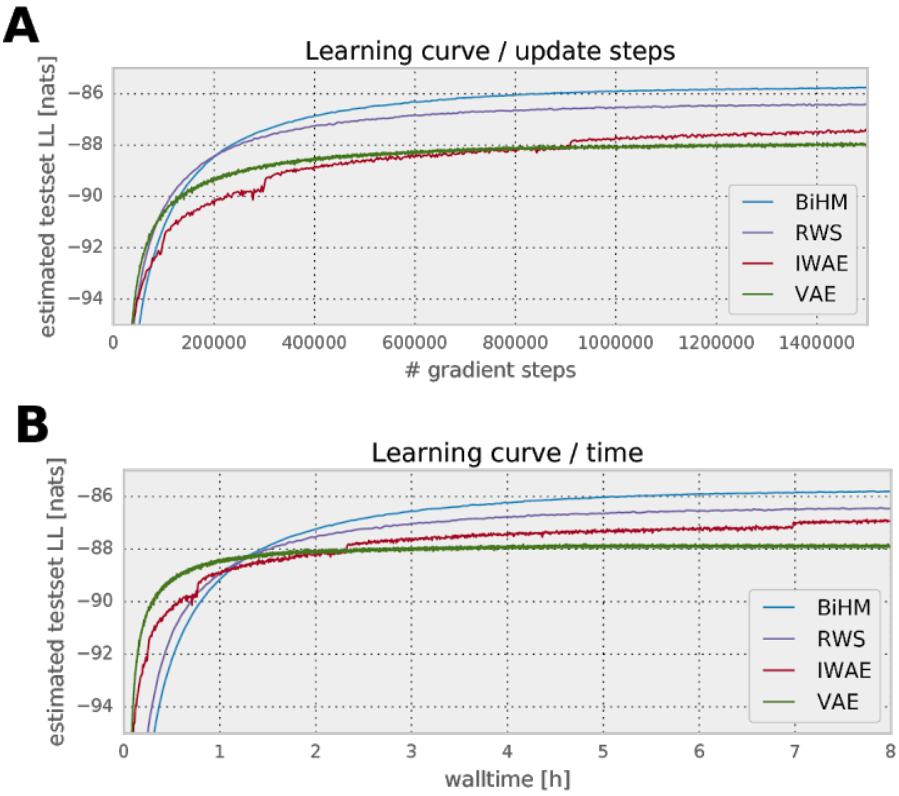

Figure 3.4 – Learning curves for MNIST experiments: For BiHM, RWS and VAE we chose the learning rate such that we get optimal results after 10^6 update steps; for IWAE we use the original learning rate schedule published in [11]. BiHM and RWS use K=10 samples per datapoint; IWAE uses K=5 and a batch size of 20 (according to the original publication). We generally observe that BiHM show very good convergence in terms of progress per update step and in terms of total training time.

mation (3.7) computes the weighted average over the individual gradients. These properties are basically inherited from the RWS training algorithm. But in contrast to RWS, and in contrast to most other algorithms which employ a generative model p and an approximate inference model q, we here automatically obtain parameter updates for both p and q because we optimize p^* which contains both. The resulting training method is summarized in algorithm 1.

Although we have focused our formal definitions on discrete variables, the same analysis applies to the continuous variables. We now discuss two general approaches for approximate sampling from a BiHM. One can either easily and efficiently sample from the directed model p, or one can use Gibbs sampling to draw higher-quality samples from the undirected model p^*. For the latter, importance resampling is used to approximately draw samples from the conditional distributions.

In the following, we show how the importance resampling weights for approxi-

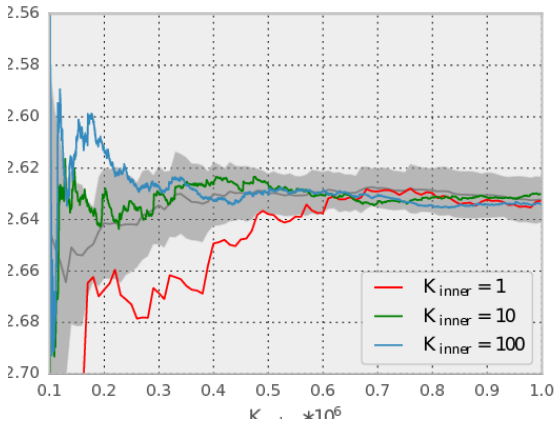

Figure 3.5 – $\log Z^2$ estimates for different values of K_{inner} as a function of the number of samples K_{outer}

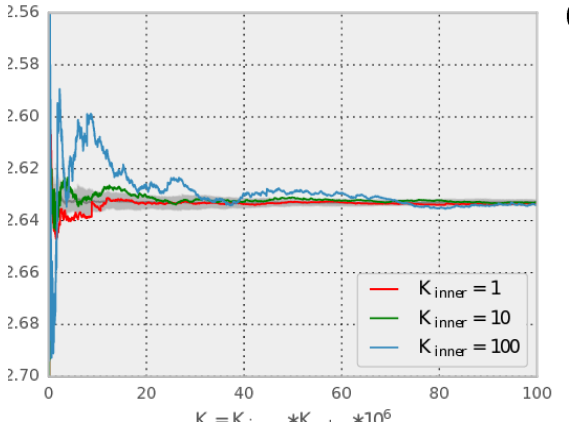

Figure 3.6 – $\log Z^2$ estimates for different values of K_{inner} as a function of the total number of samples $K_{\text{inner}} \cdot K_{\text{outer}}$ for the BiHM trained on MNIST; the gray region shows the mean and the standard deviation for 10 runs with K_{inner}=1. This shows that, from the point of view of total computation, convergence is fastest with K_{inner}=1; and that we obtain a high quality estimate of the partition function with only a few million samples. .

mate sampling from the conditional distributions $p^*(\mathbf{h}_l|\mathbf{h}_{l-1}, \mathbf{h}_{l+1})$ can be derived for any l in $\{1, \ldots, L\}$ where L is the number of hidden layers in our model. If the proposal distribution is given by

$$\frac{1}{2}p(\mathbf{h}_l|\mathbf{h}_{l+1}) + \frac{1}{2}q(\mathbf{h}_l|\mathbf{h}_{l-1}),$$

the unnormalized importance weight for a sample $\mathbf{h}_l^{(k)}$ from this proposal is

$$\omega_l^k = \frac{p^*(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}, \mathbf{h}_{l+1})}{\frac{1}{2}p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) + \frac{1}{2}q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})} \quad . \tag{3.9}$$

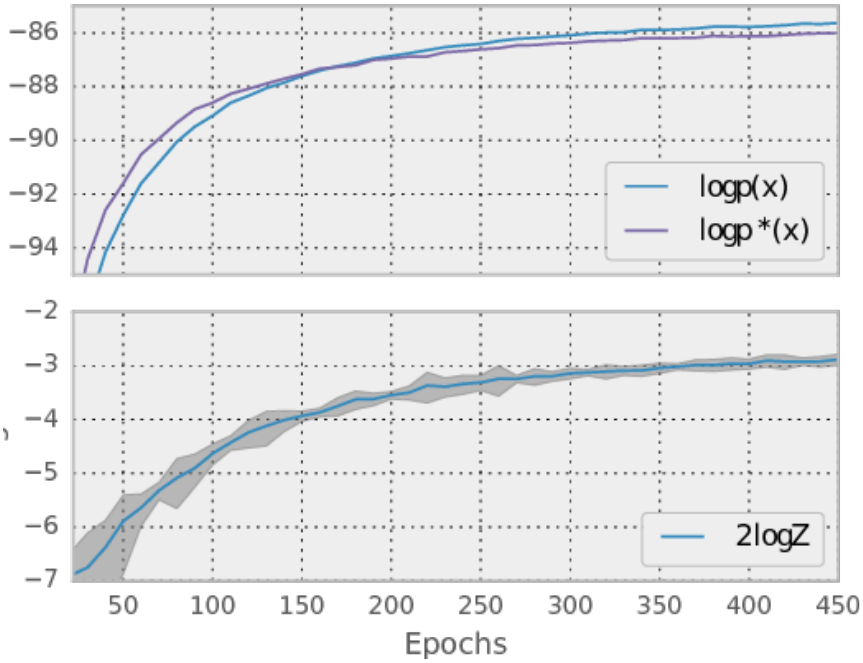

Figure 3.7 – $\log Z^2$ estimates for different values of K_{inner} as a function of evolution of the estimates of $\log p(\mathbf{x})$, $\log p^*(\mathbf{x})$, and $2 \log Z$ during training on MNIST.

Using the product form of p^* and properties of conditional distributions, we have that

$$p^*(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1}, \mathbf{h}_{l+1}) = \frac{\sqrt{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1})p(\mathbf{h}_{l-1}|\mathbf{h}_l^{(k)})q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})q(\mathbf{h}_{l+1}|\mathbf{h}_l^{(k)})}}{\sum_{\mathbf{h}_l} \sqrt{p(\mathbf{h}_l|\mathbf{h}_{l+1})p(\mathbf{h}_{l-1}|\mathbf{h}_l)q(\mathbf{h}_l|h_{l-1})q(\mathbf{h}_{l+1}|\mathbf{h}_l)}}.$$

Plugging this equality into equation (3.9), we get

$$\omega_l^k = C_l^k \frac{\sqrt{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1})p(\mathbf{h}_{l-1}|\mathbf{h}_l^{(k)})q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})q(\mathbf{h}_{l+1}|\mathbf{h}_l^{(k)})}}{p(\mathbf{h}_l^{(k)}|\mathbf{h}_{l+1}) + q(\mathbf{h}_l^{(k)}|\mathbf{h}_{l-1})},$$

where the term

$$C_l^k = \frac{2}{\sum_{\mathbf{h}_l} \sqrt{p(\mathbf{h}_l|\mathbf{h}_{l+1})p(\mathbf{h}_{l-1}|\mathbf{h}_l)q(\mathbf{h}_l|\mathbf{x})q(\mathbf{h}_{l+1}|\mathbf{h}_l)}},$$

does not depend on the sample $\mathbf{h}_l^{(k)}$ and thus can be ignored when drawing samples from $\{\mathbf{h}_l^{(1)}, \mathbf{h}_l^{(2)}, \ldots, \mathbf{h}_l^{(K)}\}$ with a probability proportional to $\{\omega_1, \omega_2, \ldots, \omega_K\}$ for any $l = 1, \ldots, L$ where K is the number of samples.

Model	ADULT	CONNE	DNA	MUSH	NIPS	OCR	RCV1	WEB
auto regressive								
FVSBN	13.17	12.39	83.64	10.27	276.88	39.30	49.84	29.35
NADE	13.19	11.99	84.81	9.81	273.08	27.22	46.66	28.39
EoNADE	13.19	12.58	82.31	9.68	272.38	27.31	46.12	27.87
DARN	13.19	11.91	81.04	9.55	274.68	28.17	46.10	28.83
RWS - NADE	13.16	11.68	84.26	9.71	271.11	26.43	46.09	27.92
non AR								
RBM	16.26	22.66	96.74	15.15	277.37	43.05	48.88	29.38
RWS - SBN	13.65	12.68	90.63	9.90	272.54	29.99	46.16	28.18
BiHM								
$-\log \tilde{p}(\mathbf{x})$	13.58	11.98	86.33	9.34	270.33	27.91	45.67	28.02
$-\log \tilde{p^*}(\mathbf{x})$	13.79	12.12	86.39	9.4	272.62	29.46	45.78	28.78
$-\log p(\mathbf{x})$	13.78	12.43	86.49	9.40	272.66	27.10	46.12	28.14
$-\log p^*(\mathbf{x})$	13.82	12.31	86.92	9.40	272.71	27.30	46.98	28.22
$-2\log Z$	0.20	0.27	0.56	0.09	1.97	1.87	0.41	0.54

Table 3.1 – Negative log-likelihood (NLL) on various binary data sets from the UCI repository: The top rows quote results from shallow models with autoregressive weights between their units within one layer. The second block shows results from non-autoregressive models (quoted from [10]). In the third block we show the results obtained by training a BiHMs. We report the estimated test set NLL when evaluating just the top-down model, $\log p(\mathbf{x})$, and when evaluating $\log p^*(\mathbf{x})$. We also reported $\log \tilde{p}(x)$, $\log \tilde{p}^*(x)$ and -2 $\log(Z)$ on test set. Our BiHM models consistently obtain similar (or better) results than while they prefer deeper architectures.

In a particular situation that $L = 2$, importance resampling is used to approximately draw samples from $p^*(\mathbf{h}_1 \,|\, \mathbf{x}, \mathbf{h}_2)$. We here choose to draw the proposal samples from the mixture distribution

$$1/2\, p(\mathbf{h}_1|\mathbf{h}_2) + 1/2\, q(\mathbf{h}_1|\mathbf{x}),$$

which ensures that we have a symmetric chance of covering the high probability configurations of $p^*(\mathbf{h}_1|\mathbf{x}, \mathbf{h}_2)$ induced by p and q. The importance weights we use to resample a final sample from $p^*(\mathbf{h}_1|\mathbf{x}, \mathbf{h}_2)$ are thus given by

$$\omega_1^k = \frac{\sqrt{p(\mathbf{h}_1^{(k)}|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1^{(k)})q(\mathbf{h}_1^{(k)}|\mathbf{x})q(\mathbf{h}_2|\mathbf{h}_1^{(k)})}}{p(\mathbf{h}_1^{(k)}|\mathbf{h}_2) + q(\mathbf{h}_1^{(k)}|\mathbf{x})} \;, \tag{3.10}$$

where $\mathbf{h}_1^{(k)}$ is randomly drawn from $p(\mathbf{h}_1|\mathbf{h}_2)$ or $q(\mathbf{h}_1|\mathbf{x})$.

Dataset	BiHM layer sizes	RWS layer sizes	ess
ADULT	100, 70, 50, 25	100, 20, 5	81.50%
CONNECT4	300, 110, 50, 20	150, 50, 10	89.17%
DNA	200,150,130,100,70,50,30,20,10	150, 10	11.28%
MUSHROOM	150, 100, 90, 60, 40, 20	150, 50, 10	92.56%
NIPS-0-12	200, 100, 50, 25	150, 50, 10	16.83%
OCR	600, 500, 100, 50, 30, 10	300, 100, 10	22.58%
RCV1	500, 120, 70, 30	200, 50, 10	70.60%
WEB	650, 580, 70, 30, 10	300, 50, 10	55.95%

Table 3.2 – Architectures for our best UCI BiHM models compared to our best RWS models. We observe that BiHMs prefer significantly deeper architectures than RWS.

3.4 Sampling and inpainting

In this section, we show how our model can be used to solve inpainting tasks by sampling the missing parts of a given image from the distribution.

For $p^*(\mathbf{x}|\mathbf{h}_1)$ we choose to approximate the sample by drawing the proposal samples from $p(\mathbf{x}|\mathbf{h}_1)$. For Gibbs sampling, we iteratively update all odd layers followed by all even layers until we consider the chain to be in equilibrium. The pseudo code can be found in algorithm 2.

Equipped with approximate sampling procedures for the conditional distributions, it is straightforward to construct an algorithm for inpainting: Given a corrupted input datapoint $\tilde{\mathbf{x}}$, we first initialize a Markov chain by drawing $\mathbf{h}_1, \mathbf{h}_2 \sim q(\mathbf{h}_1, \mathbf{h}_2|\mathbf{x})$ and then run the Gibbs sampling procedure. Whenever we sample the bottom layer $\mathbf{x} \sim p^*(\mathbf{x}|\mathbf{h}_1)$ (approximately), we keep the non-corrupted elements of $\tilde{\mathbf{x}}$ fixed. Note that this method approximately samples reconstructions $\mathbf{x} \sim p^*(\mathbf{x})$ that are consistent with $\tilde{\mathbf{x}}$; it does not provide a MAP reconstruction which would maximize $\log p^*(\mathbf{x})$ given $\tilde{\mathbf{x}}$.

3.5 Estimating the partition function

To compute $p^*(\mathbf{x}) = \frac{1}{Z^2}\tilde{p}^*(\mathbf{x})$ and to monitor the training progress it is desirable to estimate the normalization constant Z. In stark contrast to undirected models like RBMs or DBMs, we can here derive an unbiased importance sampling estimator

for Z^2 :

$$Z^2 = \mathbb{E}_{\mathbf{x},\mathbf{h}\sim p(\mathbf{x},\mathbf{h})}\left[\sqrt{\frac{q(\mathbf{h}|\mathbf{x})}{p(\mathbf{x},\mathbf{h})}}\mathbb{E}_{\mathbf{h}'\sim q(\mathbf{h}|\mathbf{x})}\left[\sqrt{\frac{p(\mathbf{h}',\mathbf{x})}{q(\mathbf{h}'|\mathbf{x})}}\right]\right]$$
$$= \mathbb{E}^{\mathbf{x},\mathbf{h}\sim p(\mathbf{x},\mathbf{h})}_{\mathbf{h}'\sim q(\mathbf{h}|\mathbf{x})}\left[\sqrt{\frac{p(\mathbf{x},\mathbf{h}')q(\mathbf{h}|\mathbf{x})}{p(\mathbf{x},\mathbf{h})q(\mathbf{h}'|\mathbf{x})}}\right]. \tag{3.11}$$

We denote the number of samples used to approximate the outer expectation and the inner expectation with K_{outer} and K_{inner} respectively.

In section 3.6, we show that we obtain high quality estimates for Z^2 with K_{inner}=1 and a relatively small number of samples K_{outer}. By taking the logarithm, we obtain a biased estimator for $2\log Z$, which will, unfortunately, underestimate $2\log Z$ on average due to the concavity of the logarithm and the variance of the Z^2 estimate. This can lead to overestimates for $\log p^*(\mathbf{x})$ using equation (3.5) if we are not careful. Fortunately, the bias on the estimated $\log Z$ is induced only by the concavity of the logarithm; the underlying estimator for Z^2 is unbiased. We can thus effectively minimize the bias by minimizing the variance of the Z^2 estimate for example by taking more samples. This is a much better situation than for Z-estimating methods that rely on Markov chains in high dimensional spaces, which might miss entire modes because of mixing issues.

3.6 Experimental results

In this section we present experimental results obtained when applying the algorithm to various binary data sets. Our main goal is to ensure that the theoretical properties discussed in previous sections translate into a robust algorithm that yields competitive results even when used with simple sigmoid belief network layers as conditional distributions. We train all models using Adam [47] with a mini-batch size of 100. We initialize the weights according to [33], set the biases to -1, and use L_1 regularization λ=10^{-3} on all the weights. Our implementation is available at https://github.com/jbornschein/bihm.

3.6.1 UCI Binary Datasets Expriments

To ascertain that a BiHM with importance sampling as training method works in general, we applied it to the 8 binary data sets from the UCI data set repository that were evaluated for example in [60]. We use a learning rate of 10^{-2} or 10^{-3} for all the experiments. The final log-likelihood estimates, the corresponding lower bounds and the estimation of $\log(Z)$ are given in table 3.1. The architectures and layer sizes can be found in tables and 3.2.

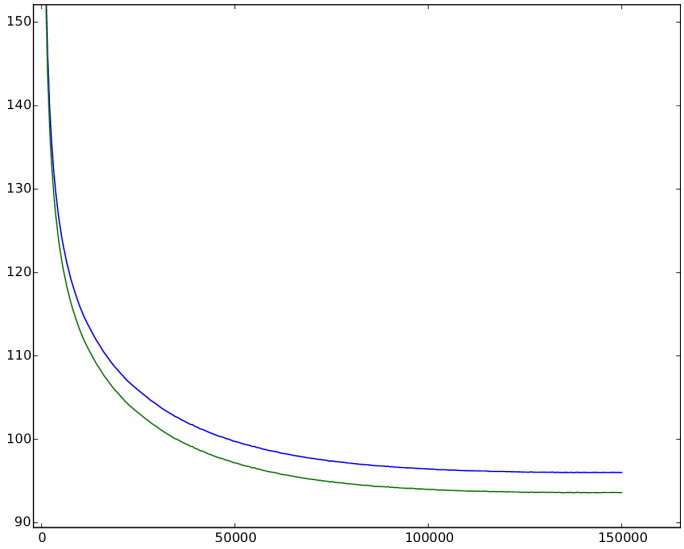

Figure 3.8 – Training convergence for binarized MNIST. Convergence of $\log \tilde{p}^*(\mathbf{x})$ shown by blue and $\log p(\mathbf{x})$ shown by green over SGD updates on the validation set using 10 samples.

We generally observe that BiHM prefers deeper architectures than RWS to obtain the best results. The results are summarized in table 3.1.

3.6.2 Binarized MNIST Experiments

We use the MNIST data set that was binarized according to [70] and downloaded in binarized form [57]. Compared to RWS, we again observe that BiHM prefers significantly deeper and narrower models. Our best model consists of 12 layers with (300-200-100-75-50-35-30-25-20-15-10-10) latent variables. We follow the same experimental procedure as in the RWS paper: First train the model with K=10 samples and a learning rate of 10^{-3} until convergence and then fine-tune the parameters with K=100 samples and a learning rate of 3×10^{-4}. All layers are actually used to model the empirical distribution; we confirmed that training shallower models (obtained by leaving out individual layers) decreases the performance. We obtain test set log-loglikelihoods of $\log p^*(\mathbf{x}) \simeq$ **-84.8** $\pm$ **0.23** and $\log p(\mathbf{x}) \simeq$ **-84.5** $\pm$ **0.22**.

The next section presents a more detailed analysis of these estimates and their dependency on the number of samples from the proposal distribution $q(\mathbf{h}|\mathbf{x})$. Note that even though this model is relatively deep, it is not particularly large, with about $700,000$ parameters in total. The DBMs in [89] contain about $900,000$ and 1.1 million parameters; a variational autoencoder with two deterministic, 500 units wide encoder and decoder layers, and with 100 top level latent units contains more than 1.4 million parameters. To highlight the models ability to generate crisp (non-blurry) digits we use algorithm 2 to draw samples. The results are visualized in

Figure 3.9 – Results after training on TFD: **A**) Random selection of 12 samples drawn from $p^*(\mathbf{x})$ (10 iterations of Gibbs sampling). **B**) The left column in each block shows the input; the right column shows a random output sample generated by the inpaiting algorithm (see section 3.4).

figure 3.2 shows samples obtained when drawing from the top-down generative model $p(\mathbf{x})$ before running any Gibbs iterations.

To obtain the samples in figure 3.2 we used $10,000$ samples. However, using only 100 samples lead to visually indistinguishable results. Using MCMC to sample from p^* results in digits that are visually indistinguishable from 3.2, but without bias towards 1. We initialized the chains by sampling from p and then running 20 up-downward sweeps using equation (3.10) with $10,000$ samples each. Again, using 100 samples is enough to obtain visually indistinguishable results. Figure 3.1 visualizes the results when running the inpainting algorihm to reconstruct partially occluded images.

3.6.3 Toronto Face Database Experiments

We also trained models on the 98,058 examples from the unlabeled section of the Toronto face database (TFD) [101]. Each training example is of size 48×48 pixels and we interpret the gray-level as Bernoulli probability for the bottommost layer. We observe that training proceeds rapidly during the first few epochs but mostly only learns the mean-face. During the next few hundred epochs training proceeds much slower but the estimated log-likelihood $\log p^*(\mathbf{x})$ increases steadily. Figure 3.9 **A** shows random samples from a model with (1000-700-700-300) latent variables in 4 hidden layers. It was trained with a learning rate of $3 * 10^{-5}$; all other hyperparameters were set to the same values as before. Figure 3.9 **B** shows the results from inpainting experiments with this model. Figure 3.10 shows 100

Moldel	$\leq$ - $\log p(x)$	$\approx$ - $\log p(\mathbf{x})$
NADE	–	88.9
continuous latent variables		
VAE	96.2	88.7
VAE + HMC (8 iterations)	88.3	85.5
IWAE (2 latent layers)	–	85.3
binary latent variables		
NVIL (2 latent layers)	99.6	–
RWS (5 latent layers)	96.3	85.4
BiHM (12 latent layers)	**89.2**	**84.5**

Table 3.3 – Comparison of BiHMs to other recent methods in the literature. We report the lower bounds and estimates for the marginal log probability on the binarized MNIST test set.

samples from the BiHM trained on the TFD data set.

3.7 Analysis of IS-based estimates

As we explained earlier, we deal with the fact that inference, learning, and sampling are all intractable in our model and they can be estimated using importance sampling. In this section, we discuss the analysis of our estimations using this approach.

3.7.1 Estimating the partition function

As we discussed, we can perform efficient approximate sampling from p to estimate the partition function Z, as in equation (3.11). We note, however, that some models , such as DBNs, require to run a Markov chain to estimate Z.

There are many choices of K_{inner} and K_{outer} to approximate Z. In figure 3.5 we plot $-2 \log Z$ estimates over the number of outer samples K_{outer} for our best MNIST model and for 3 different choices of K_{inner}, for example $K_{\text{inner}} \in \{1, 10, 100\}$. The gray area additionally shows the mean and standard deviation for 10 runs with $K_{\text{inner}} = 1$.

In figure 3.6 we plot the estimates over the total number of samples $K_{\text{outer}} \cdot K_{\text{inner}}$. We observe that choosing $K_{\text{inner}} = 1$ and using only about 10 million samples results in high quality estimates for $2 \log Z$ with an standard error far below 0.1 nats. Estimating based on 10 million samples takes less than 2 minutes on a GTX980 GPU. Figure 3.7 shows that the development of the $2 \log Z$ estimate during learning and in relation to the log-likelihood estimates.

Figure 3.10 – The figure shows 100 samples from the BiHM trained on the TFD data set.

3.7.2 Importance sampling efficiency

As we discussed in section 2.4, one of the most commonly used measures to estimate the quality of an importance sampling estimator is the ESS, and larger values of ESS imply more information extracted per sample.

We compute the ESS over the MNIST test set for K=100,000 proposal samples from $q(\mathbf{h}|\mathbf{x})$. For our best RWS model which is a model with (400-300-200-100-10) latent variables in 5 stochastic layers, we obtain $\widehat{ess} \simeq 0.10\% \pm 0.06$; for the BiHM model we obtain $\widehat{ess} \simeq 11.9\% \pm 1.1$. When we estimate the ESS for using $q(\mathbf{h}|\mathbf{x})$ from the BiHM as a proposal distribution for $p(\mathbf{h}|\mathbf{x})$, we obtain $\widehat{ess}$=1.2% $\pm$ 0.2. The estimated ESS values indicate that training BiHM models indeed results in distributions whose intractable posterior $p^*(\mathbf{h}|\mathbf{x})$ as well as top-down model $p(\mathbf{h}|\mathbf{x})$ are much better modeled by the learned $q(\mathbf{h}|\mathbf{x})$.

We also estimated the ESS for a VAE with two deterministic, 500 units wide ReLU layers in the encoder and decoder. This model has a single stochastic layer with 100 continuous variables at the top; it reaches a final estimated test set log-

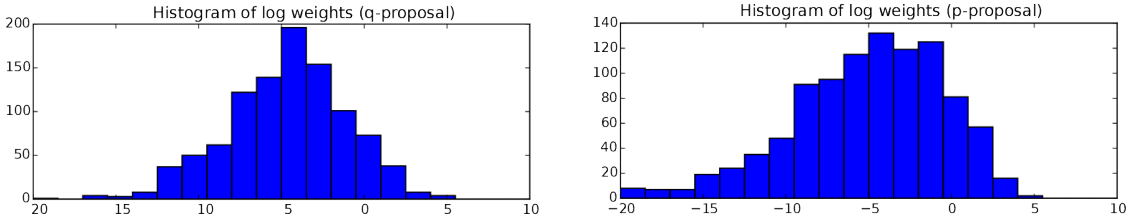

Figure 3.11 – Hiistograms of the importance weights when we use $q(\mathbf{h}|\mathbf{x})$ as a proposal for $p^*(\mathbf{h}|\mathbf{x})$ (left), and when we use $p(\mathbf{x}, \mathbf{h})$ as a proposal distribution for $p^*(\mathbf{x}, \mathbf{h})$ (right).

likelihood of $\log p(\mathbf{x}) \simeq$ -88.9 $\pm$0.28. The final variational lower bound, which corresponds exactly to the importance sampling estimate of $\log p(\mathbf{x})$ with K=1 sample, is -95.8. For this model we obtain an ESS of $0.07\% \pm 0.02$. These results indicate that we need thousands of samples to obtain reliable log-likelihood estimates with low approximation error.

In figure 3.3 we plot the estimated test set log-likelihood over the number of samples K used to estimate $\log p^*(\mathbf{x})$ and $\log p(\mathbf{x})$. For all the models and for small a number of samples K we significantly underestimate the log-likelihood; but, in comparison to RWS, the estimates for the BiHM model are much higher and less sensitive to K. For example, using K=10 samples to evaluate the BiHM model results in a higher log-likelihood estimate than using K=10,000 samples to evaluate the RWS model.

3.7.3 Symmetry of p and q

In figure 3.11, we show the histogram of importance weights when using $q(\mathbf{h}|\mathbf{x})$ as proposal for p^* when performing inference for a given $\mathbf{x}$ from the test set, and when using p as proposal for p^* when drawing samples according to algorithm 2. According to our goal formulated in section 3.1, both p and q should stay close to p^*. The weights of the former occur whenever we perform approximate inference for example during learning; the weights of the latter occur when we sample from the model. Unsurprisingly, we observe that the quality of both proposal distributions is roughly symmetric relative to the target distribution p^*. This indicates that drawing a sample from p^* and performing approximate inference $\mathbf{h} \sim p^*(\mathbf{h}|\mathbf{x})$ are now similarly hard problems. This is different from other Helmholtz machines, where sampling from the model is straight forward and exact, but inference is even harder by comparing ESS in section 3.7.2.

3.7.4 Computational cost

As we discussed, we can also efficiently and accurately evaluate the log-likelihood at test time because we obtain unbiased low-variance estimates for the partition

function Z with rather small sample sizes and without, e.g., annealed importance sampling.

To demonstrate the computational efficiency of our approach we show typical MNIST learning curves in figure 3.4. For BiHM, RWS and VAE the learning rate was chosen within a factor of 2 to obtain optimal results after 10^6 update steps ($5 \cdot 10^{-4}$ for BiHM and RWS, $3 \cdot 10^{-3}$ for VAE; K=10 for BiHM and RWS). For the IWAE experiment we use the original code, hyperparameters and learning rate schedule from [11]: This experiment thus uses a mini-batch size of 20 instead of 100, K=5 training samples and and 8 different learning rates over the course of ≈ 3300 epochs. In all cases we used K=1000 samples to evaluate the test set log-likelihoods.

We generally observe that BiHM show very good convergence in terms of progress per update step and competitive performance in terms of total training time. Note that BiHMs and RWS allow for an efficient distributed implementation in the future: per sample, only the binary activations and a single floating point number (the importance weight) need to be communicated between layers. VAEs and IWAEs need to communicate continuous activations during the forward pass and continuous partial gradients during the backward pass. At test time BiHMs are typically much more effective than the other methods: BiHMs obtain good LL estimates with K=10 or 100 samples per datapoint while VAE, RWS and IWAE models require $\approx$ 10,000 samples to obtain competitive results (compare figure 3.3).

Comparison to RWS, number of samples used, dependence on the number of layers, and updated results: By choosing different hyperparameters, using Adam instead of RMSProp and following the experimental protocol from the RWS paper (5 samples to estimate the gradient for 1000 epochs; fine-tuning with 100 samples for gradient estimation for 500 epochs) we obtain a final $-\log p \simeq 86.7 \pm 0.23$ and $-\log \tilde{p}^* \simeq 89.8 \pm 0.24$ on MNIST with (400-300-200-100-50-25-10) units in 7 hidden layers. With RWS we obtain $-\log p \simeq 86.1 \pm 0.23$. The error bounds indicate the standard error of the mean over the test set. We repeated the analysis 10 times (10 times 10k samples per test set example) and observe that the variance introduced by the IS sampling estimators is less than 0.01. The uncertainty introduced by IS is thus an order of magnitude smaller than the uncertainty introduced by the finite MNIST test set. Removing any of the layers decreases performance. For example, for 400-300-200-50-25-10 hidden units we obtain $-\log p \simeq 86.9 \pm 23$.

The goal of this article was to introduce a new scheme to construct probabilistic generative models which are automatically regularized to be close to approximate inference distributions we have at our disposal. Using the Bhattacharyya distance we derived a lower-bound on the log-likelihood, and we demonstrated that the bound can be used to fit deep generative models with multiple layers of latent variables to complex training distributions. Note that our definition for p^* forced us to choose a prior distribution $q(\mathbf{x})$ which will be part of our generative model $p^*(\mathbf{x}, \mathbf{h})$. This is different from the typical variational approaches to train Helmholtz machines where we would think of $q(\mathbf{h}|\mathbf{x})$ solely as an approximate inference method

given a training example $\mathbf{x}$, and where $q(\mathbf{x})$ would be the (empirical) training distribution – something we cannot assume because $q(\mathbf{x})$ is part of our model p^*.

Prologue to Second Article

Semi-supervised Bidirectional Helmholtz Machines, J. Bornschein, S. Shabanian, A. Fischer, Y. Bengio. (in preparation)

Personal Contribution: J. Bornschein and I had the main idea. I wrote 70 percent of the code. All authors suggested experiments for understanding the model and I ran all the experiments for MNIST.

4 Semi-supervised BiHM

Semi-supervised learning is very useful in a variety of applications and it is needed in a broad range of problems that are of interest such as image search, genomics, natural language parsing, and speech analysis [29, 64, 65, 95]. As we discussed earlier, it is designed to solve the classification problem when only a few observations are labeled. Importantly, algorithms for semi-supervised learning are typically cheaper than the ones for supervised learning since in such algorithms, each observation does not require a label.

4.1 Introduction

There are many approaches for semi-supervised learning which are surprisingly successful at solving classification problems. One of the earliest approach is transductive support vector machines (TVSMs) which was proposed by Joachims in 1999 [44]. This builds on the SVM model and the approach is to label the unlabeled data such that the decision boundary has maximal margin over all of the data. A Gaussian based approach similar to TSVMs was proposed by Lawrence and Jordan in 2005 which makes use of a null category noise model [61]. Perhaps one of the simplest approaches in semi-supervised learning is a self-training scheme in which the unlabeled samples are expanded into the labeled ones [87]. However, these models may not be appropriate with large amounts of unlabeled data, since obtaining the optimal parameters is difficult in such a setting. In 2003, Zhu et al. introduced a new semi-supervised model which is based on the idea of constructing a graph connecting similar data points [114]. Intuitively, such models assume that similar data points have similar labels and information propagates from labeled data points. However, computing the probabilities in such models is expensive and it is highly dependent on the structure of the graph [9, 29]. In all of these models, the supervised and unsupervised parts are jointly trained at each step. There are some networks in which the unsupervised portion is trained on unlabeled data separately as a first step, and then its features are the input of a supervised classifier [12, 13]. We note that all of these networks are shallow.

Recently, many deep semi-supervised models have been proposed. For example, the manifold tangent classifier (MTC) is an approach that proposes to learn the manifold on which the data lies using contractive auto-encoders (CAEs), and then train a classifier [83]. Moreover, several papers have proposed the use of supervised

and unsupervised tasks in the same neural network [81, 110]. Such networks attempt to optimize the combined loss function of an autoencoder and a classifier. There has been significant work on extending and improving these networks [1, 58, 80, 82].

Many of these improvements are based on using the power of generative models. In [45], the authors attempt to use non-parametric density models, or more recently, in [113], a model based on a Gaussian mixture was proposed. Unfortunately, in many cases, they do not perform very well for reasons such as a large number of mixture components or a poor inference model. Some of the earliest extensions of variational approximations for semi-supervised models were developed in [63, 108]. Most recently, a deep generative model for semi-supervised learning was introduced based on the auto-encoding variational Bayes model with labeled units [49, 48].

Here, we introduce a theoretic framework of semi-supervised learning model using BiHM.

4.2 Model Definition and Properties

We define a joint probability distribution over all variables – an observed vector $\mathbf{x}$, a latent variable vector $\mathbf{h}$, and an output layer vector $\mathbf{y}$. We will explain the model considering one hidden layer, but the discussion is extendable to any arbitrary graph without loops. It can be specially used to train architectures with more than two stacked layers of hidden variables.

Assume that $p^*(\mathbf{x}, \mathbf{h}, \mathbf{y})$ is a joint distribution which is defined as

$$p^*(\mathbf{x}, \mathbf{h}, \mathbf{y}) = \frac{1}{Z}\sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})q(\mathbf{x}, \mathbf{h}, \mathbf{y})}$$

where p and q are the two joint distributions that can be factorized as

$$p(\mathbf{x}, \mathbf{h}, \mathbf{y}) = p(\mathbf{y})\ p(\mathbf{h}|\mathbf{y})\ p(\mathbf{x}|\mathbf{h}),$$

and

$$q(\mathbf{x}, \mathbf{h}, \mathbf{y}) = q(\mathbf{x})\ q(\mathbf{h}|\mathbf{x})\ q(\mathbf{y}|\mathbf{h}),$$

and Z is the normalization constant. We will typically think of the top level prior $p(\mathbf{y})$ as a simple multinomial distribution over Y different label classes $\mathbf{y} \in \{1, \cdots, Y\}$ with $p(\mathbf{y} = y) = \pi_y$. In this case we will use a conditioned multinomial distribution for the bottom-up layer $q(\mathbf{y}|\mathbf{h})$ and we implement it with an MLP with a single Softmax layer defined as

$$q(\mathbf{y} = y|\mathbf{h}) = \frac{\exp(\mathbf{h}^T W_y + b_y)}{\sum_{k=1}^{Y} \exp(\mathbf{h}^T W_k + b_k)},$$

where W_y and b_y is the y-th column of weight matrix and bias vector of the last layer for every $y = 1, \ldots, Y$. We do not assume an explicit form for $q(\mathbf{x})$. In fact,

from now on we assume that the marginal distributions $p^*(\mathbf{x})$ and $q(\mathbf{x})$ are the same.

Given this model definition, there are at least three quantities we are regularly interested in when we deal with semi-supervised learning tasks. The first quantity is $p^*(\mathbf{x})$ which is the marginal probability distribution for data points that can be obtained as

$$\begin{aligned} p^*(\mathbf{x}) = q(\mathbf{x}) &= \sum_{\mathbf{h},\mathbf{y}} p^*(\mathbf{x},\mathbf{h},\mathbf{y}) \\ &= \frac{\sqrt{q(\mathbf{x})}}{Z} \sum_{\mathbf{h},\mathbf{y}} \sqrt{p(\mathbf{x},\mathbf{h},\mathbf{y})\; q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})} = \frac{1}{Z^2} A^2(\mathbf{x}), \end{aligned} \tag{4.1}$$

where

$$A(x) = \sum_{\mathbf{h},\mathbf{y}} \sqrt{p(\mathbf{x},\mathbf{h},\mathbf{y})\; q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})}.$$

The second quantity is $p^*(\mathbf{x},\mathbf{y})$ which is

$$\begin{aligned} p^*(\mathbf{x},\mathbf{y}) &= \sum_{\mathbf{h}} p^*(\mathbf{x},\mathbf{h},\mathbf{y}) \\ &= \frac{\sqrt{q(\mathbf{x})}}{Z} \sum_{\mathbf{h}} \sqrt{p(\mathbf{x},\mathbf{h},\mathbf{y})\; q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})} \\ &= \frac{1}{Z^2} A(\mathbf{x}) \left(\sum_{\mathbf{h}} \sqrt{p(\mathbf{x},\mathbf{h},\mathbf{y})\; q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})} \right) \\ &= \frac{1}{Z^2} A(\mathbf{x}) B(\mathbf{x},\mathbf{y}), \end{aligned} \tag{4.2}$$

where

$$B(x,y) = \sum_{\mathbf{h}} \sqrt{p(\mathbf{x},\mathbf{h},\mathbf{y})\; q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})}.$$

Finally, the last quantity that we are interested in, is the conditional probability $p^*(\mathbf{y}|\mathbf{x})$ which can be obtained as

$$p^*(\mathbf{y}|\mathbf{x}) = \frac{p^*(\mathbf{x},\mathbf{y})}{p^*(\mathbf{x})} = \frac{B(\mathbf{x},\mathbf{y})}{A(\mathbf{x})} \tag{4.3}$$

Interestingly, $p^*(\mathbf{y}|\mathbf{x})$ does not depend on the partition function Z.

4.2.1 Importance sampling estimators

As we can see, $A(\mathbf{x})$ and $B(\mathbf{x},\mathbf{y})$ are intractable for most problems of interest. In order to evaluate these terms, we use importance sampling. We use $q(\mathbf{h}|\mathbf{x})$ and

$q(\mathbf{y}|\mathbf{h})$ as proposal distributions to estimate $A(\mathbf{x})$ as follows:

$$\begin{aligned} A(\mathbf{x}) &= \sum_{\mathbf{h},\mathbf{y}} \sqrt{p(\mathbf{x},\mathbf{h},\mathbf{y})\; q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})} \\ &= \mathbb{E}_{\substack{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x}) \\ \mathbf{y}\sim q(\mathbf{y}|\mathbf{h})}} \sqrt{\frac{p(\mathbf{x},\mathbf{h},\mathbf{y})}{q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})}} \\ &\simeq \sum_{k=1}^{K} \sqrt{\frac{p(\mathbf{x},\mathbf{h}^{(k)},\mathbf{y}^{(k)})}{q(\mathbf{h}^{(k)}|\mathbf{x})q(\mathbf{y}^{(k)}|\mathbf{h}^{(k)})}} = \tilde{A}(\mathbf{x}) \end{aligned} \tag{4.4}$$

where $\mathbf{h}^{(k)}$ is generated by $q(\mathbf{h}|\mathbf{x})$ and $y^{(k)}$ is generated by $q(\mathbf{y}|\mathbf{h}^{(k)})$ for every $k = 1,\ldots,K$ where K is the number of generated samples.

To evaluate $B(\mathbf{x},\mathbf{y})$, we use only $q(\mathbf{h}|\mathbf{x})$ as a proposal distribution and then we have

$$\begin{aligned} B(\mathbf{x},\mathbf{y}) &= \sum_{\mathbf{h}} \sqrt{p(\mathbf{x},\mathbf{h},\mathbf{y})\; q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})} \\ &= \mathbb{E}_{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x})} \sqrt{\frac{p(\mathbf{x},\mathbf{h},\mathbf{y})q(\mathbf{y}|\mathbf{h})}{q(\mathbf{h}|\mathbf{x})}} \\ &\simeq \sum_{k=1}^{K} \sqrt{\frac{p(\mathbf{x},\mathbf{h}^{(k)},\mathbf{y})q(\mathbf{y}|\mathbf{h})}{q(\mathbf{h}^{(k)}|\mathbf{x})}} = \tilde{B}(\mathbf{x},\mathbf{y}), \end{aligned} \tag{4.5}$$

where $\mathbf{h}^{(k)}$ is generated by $q(\mathbf{h}|\mathbf{x})$ for every $k = 1,\ldots,K$, where K is the number of generated samples.

To provide a generative semi-supervised BiHM objective funtion, we need to consider that inputs fall into two main categories. The first category includes inputs whose labels are observed. For this class of inputs, we have

$$\begin{aligned} \log p^*(\mathbf{y}|\mathbf{x}) &= \log B(\mathbf{x},\mathbf{y}) - \log A(\mathbf{x}) \\ &\simeq \log \tilde{B}(\mathbf{x},\mathbf{y}) - \log \tilde{A}(\mathbf{x}). \end{aligned} \tag{4.6}$$

and we use the following objective function:

$$\log \tilde{B}(\mathbf{x},\mathbf{y}) - \gamma \log \tilde{A}(\mathbf{x}). \tag{4.7}$$

where γ is a real number in $[-1,+1]$. The second category where the labels for inputs are missing, we use $\log p^*(\mathbf{x})$ which can be estimated as:

$$\begin{aligned} \log p^*(\mathbf{x}) &= 2 * \log A(\mathbf{x}) - 2 * \log(Z) \\ &\simeq 2 * \log \tilde{A}(\mathbf{x}) - 2 * \log(Z) \geq 2 * \log \tilde{A}(\mathbf{x}), \end{aligned} \tag{4.8}$$

using the fact that $Z \leq 1$.

Now assume that θ is the set of all parameters in our model. In order to update θ using stochastic gradient descent, we need to derive estimators for the gradients of $\log A(\mathbf{x})$ and $\log B(\mathbf{x}, \mathbf{y})$ and use these estimations to compute estimated gradients for $\log p^*(\mathbf{x}), \log p^*(\mathbf{x}, \mathbf{y})$ and $\log p^*(\mathbf{y}|\mathbf{x})$. We start with the first term $\log A(\mathbf{x})$:

$$\begin{aligned}
\frac{\partial}{\partial \theta} \log A(\mathbf{x}) &= \frac{1}{A(x)} \sum_{\mathbf{h},\mathbf{y}} \frac{\partial}{\partial \theta} \Big(\sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})} \Big) \\
&= \sum_{\mathbf{h},\mathbf{y}} \frac{\sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})}}{A(x)} \frac{\partial}{\partial \theta} \Big(\log \sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})} \Big) \\
&= \mathop{\mathbb{E}}_{\mathbf{y},\mathbf{h} \sim q(\mathbf{y},\mathbf{h}|\mathbf{x})} \frac{1}{A(x)} \sqrt{\frac{p(\mathbf{x}, \mathbf{h}, \mathbf{y})}{q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})}} \frac{\partial}{\partial \theta} \Big(\log \sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})} \Big) \\
&\simeq \sum_{k=1}^{K} \tilde{\omega}_A^k \frac{\partial}{\partial \theta} \Big(\log \sqrt{p(\mathbf{x}, \mathbf{h}^{(k)}, \mathbf{y}^{(k)})\ q(\mathbf{h}^{(k)}|\mathbf{x}) q(\mathbf{y}^{(k)}|\mathbf{h}^{(k)})} \Big) \qquad (4.9)
\end{aligned}$$

with

$$\tilde{\omega}_A^k = \frac{\omega_A^k}{\sum_{k'} \omega_A^{k'}},$$

where

$$\omega_A^k = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}^{(k)}, \mathbf{y}^{(k)})}{q(\mathbf{h}^{(k)}|\mathbf{x}) q(\mathbf{y}^{(k)}|\mathbf{h})}},$$

with $\mathbf{h}^{(k)} \sim q(\mathbf{h}^{(k)}|\mathbf{x})$ and $\mathbf{y}^{(k)} \sim q(\mathbf{y}|\mathbf{h}^{(k)})$ where K is the number of samples. Now, the estimators for the gradients of the second term $\log B(\mathbf{x}, \mathbf{y})$ is:

$$\begin{aligned}
\frac{\partial}{\partial \theta} \log B(\mathbf{x}, \mathbf{y}) &= \frac{1}{B(\mathbf{x}, \mathbf{y})} \frac{\partial}{\partial \theta} \sum_{\mathbf{h}} \sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})} \\
&= \sum_{\mathbf{h}} \frac{\sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})}}{B(\mathbf{x}, \mathbf{y})} \frac{\partial}{\partial \theta} \log \sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})} \\
&= \mathop{\mathbb{E}}_{\mathbf{h} \sim q(\mathbf{y}|\mathbf{h})} \frac{\sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x})}}{B(\mathbf{x}, \mathbf{y}) \sqrt{q(\mathbf{y}|\mathbf{h})}} \frac{\partial}{\partial \theta} \log \sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x}) q(\mathbf{y}|\mathbf{h})} \\
&\simeq \sum_{k=1}^{K} \tilde{\omega}_B^k \frac{\partial}{\partial \theta} \log \sqrt{p(\mathbf{x}, \mathbf{h}^{(k)}, \mathbf{y})\ q(\mathbf{h}^{(k)}|\mathbf{x}) q(\mathbf{y}|\mathbf{h}^{(k)})} \qquad (4.10)
\end{aligned}$$

with $\tilde{\omega}_B^k = \frac{\omega_B^k}{\sum_{k'} \omega_B^{k'}}$ where

hidden units	γ	$\leq$ - $\log \tilde{p}^*(x)$	- $\log \tilde{p}^*(y\|x)$	error
230,200,170,130,80,50,25	-0.001	110.61	0.54	0.15
200,170,140,110,80,50,25	0	110.03	0.47	0.13
250,200,150,120,80,50,25	0.001	110.32	0.57	0.16
300,200,130,80,50,25	-0.01	109.87	0.50	0.16
250,200,150,100,80,50,25,15	0.1	111.07	0.78	0.25
300,200,130,80,50,25,15	0.01	110.2	0.61	0.15

Table 4.1 – Performance on Binarized MNIST : best achieved results by semi-supervsied BiHM.

$$\omega_B^k = \sqrt{\frac{p(\mathbf{x}, \mathbf{h}^{(k)}, \mathbf{y}^{(k)}) q(\mathbf{y}^{(k)}|\mathbf{h})}{q(\mathbf{h}^{(k)}|\mathbf{x})}},$$

with $\mathbf{h}^{(k)} \sim q(\mathbf{h}^{(k)}|\mathbf{x})$ for every $k = 1, \ldots, K$ where K is the number of samples.

4.3 Experiments

We initialize the weights according to [33], set the biases to -1. The objectives were optimized using minibatch gradient ascent until convergence, using Adam and learning rate of 0.001. The model is trained using 10 samples and Minibatch of size 100 and a varied range of γ.

We test performance on the Binarized MNIST dataset that was binarized according to [70] and downloaded in binarized form [57]. The dataset for semi-supervised learning is created by splitting the 50,000 training points between a labelled and unlabelled set, with the labelled set of size 1000. We ensure that each class has the same number of labelled points. Table 4.1 shows the best achieved results.

4.4 Results and Discussion

Although careful training of a BiHM model achieves high quality results, it does not necessarily imply that the hierarchical latent representations are informative enough to be used for semi-supervised learning problems. If we compare our results in 4.1 with some existing results in the literature such as in [48], we conclude that our model does not perform very well.

It is tricky to pinpoint failings of a model, particularly for complex latent variable models. One thing we can try to do is to visually inspect the informativeness of latent layers – consider our best performing BiHM model which consists of 1

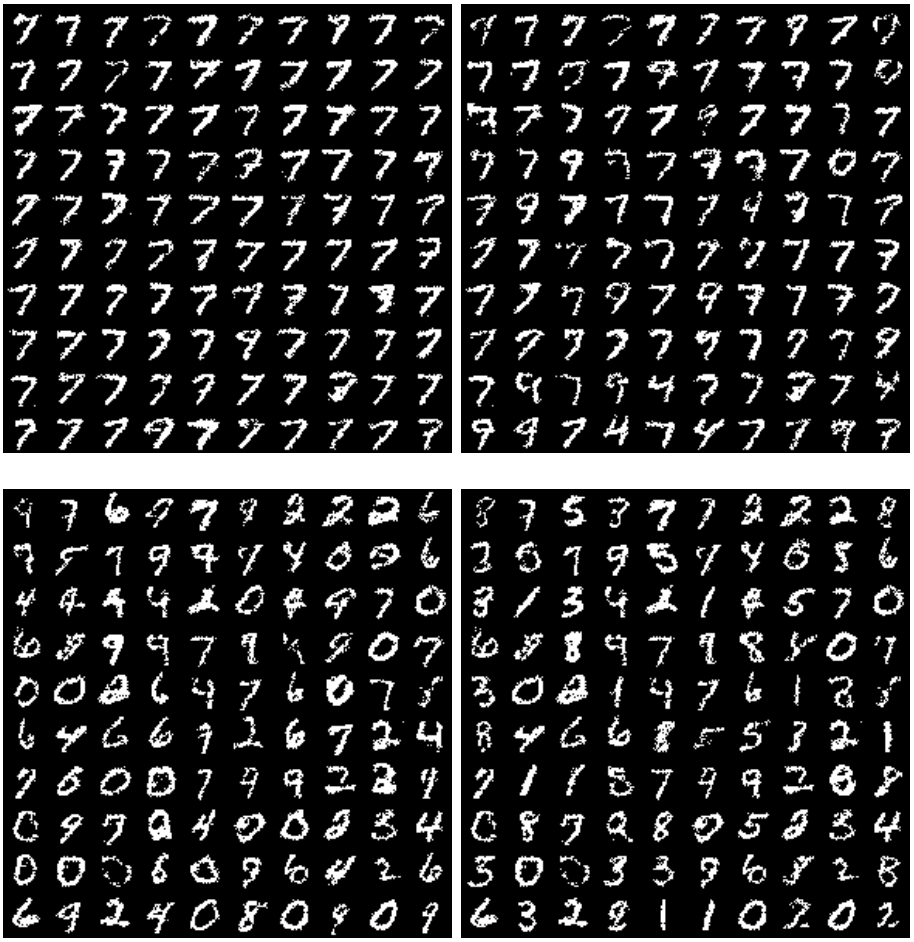

Figure 4.1 – Random generated sample from top-down model p starting from a given layer 5 (top left), 6 (top right), 7 (bottom left), and 8 (bottom right).

visible and 12 latent layers with 300, 200, 100, 75, 50, 35, 30, 25, 20, 15, 10, 10 latent variables. Assume that random samples $\mathbf{h}_{12}^{(0)}, \ldots, \mathbf{h}_{1}^{(0)}$ are generated top-down from our trained BiHM model where $\mathbf{h}_{l}^{(0)}$ corresponds to the generated sample for layer $l = 1, \ldots, 12$. Now we generate random samples from $p(x, \mathbf{h}_1, \ldots, \mathbf{h}_{l_0-1} | \mathbf{h}_{l_0}^{(0)})$ for any arbitrary layer $l_0 = 1, \ldots, 12$. Figure 4.3 shows the results for $l_0 = 1, \ldots, 4$. Not only do the generated samples get noisier as we go deeper from one layer to another, but also figures 4.1 and 4.2 show how we generate different digits as we go back from a deeper layer.

This experiment reveals the likely reason why our proposed semi-supervised model does not perform well. The problem appears to be that the higher layers of the BiHM are not informative enough to be well-applicable to a semi-supervised task. In Sec. 4.6 we shall investigate the usefulness of latent representations of lower layers in the BiHM.

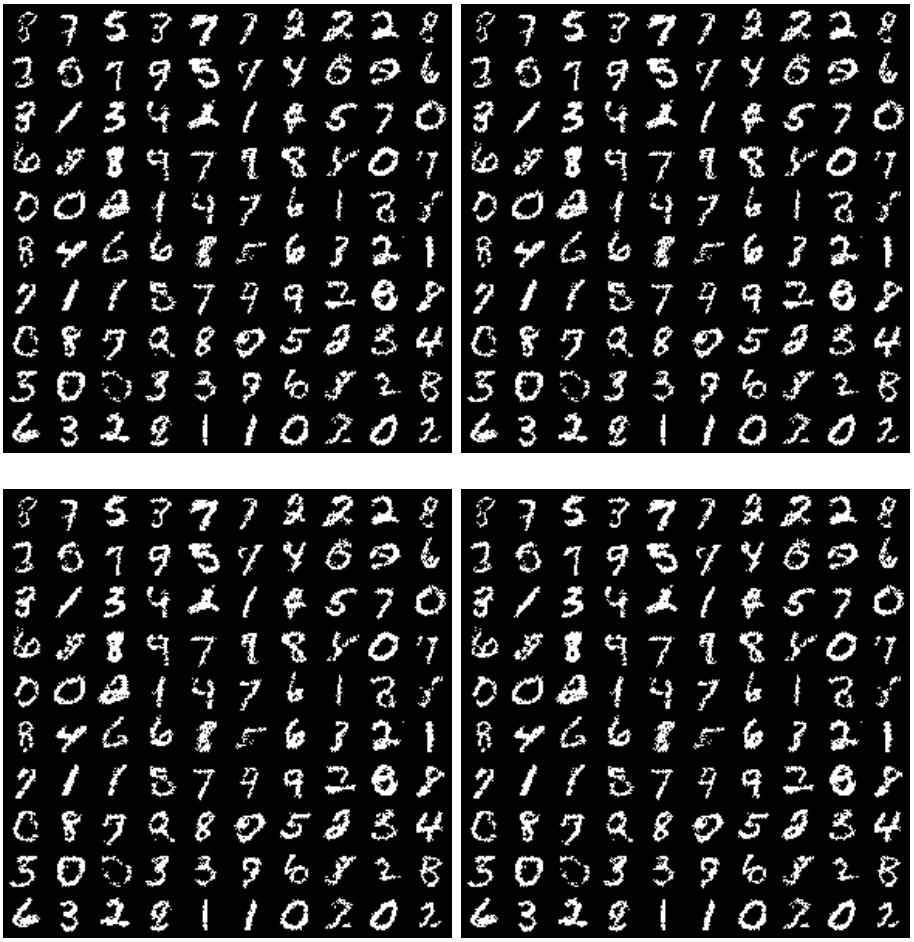

Figure 4.2 – Random generated sample from top-down model p starting from layer 9 (top left), 10 (top right), 11 (bottom left), and 12 (bottom right).

4.5 Some open questions

The experiments raise several important and difficult questions regarding the semi-supervised BiHM model that can greatly impact performance. The first question is how optimizing $p^*(\mathbf{x}, \mathbf{y})$ vs. optimizing $p^*(\mathbf{y}|\mathbf{x})$ can influence the results for different fractions of labeled training data. The second question is whether we can reuse the same $\mathbf{h}$ samples to estimate both $A(\mathbf{x})$ and $B(\mathbf{x}, \mathbf{h})$. The third question is whether we can somehow factor out some common terms from $A(\mathbf{x})$ and $B(\mathbf{x}, \mathbf{y})$. The last question is how we can guarantee that $\log p^*(\mathbf{y}|\mathbf{x})$ is negative.

We start by briefly discussing a basic theoretical solution that one can use for the last question regarding the positiveness of $-\log p^*(\mathbf{y}|\mathbf{x})$. Intuitively, if we assume that we use the same $\mathbf{h}$ samples to estimate $A(\mathbf{x})$ and $B(\mathbf{x}, \mathbf{y})$, and consider the situation that $\mathbf{y}$ is not one of the samples generated by our proposal distribution

$q(\mathbf{y}|\mathbf{h})$, then it is seems likely that

$$\tilde{B}(\mathbf{x}, \mathbf{y}) \geq \tilde{A}(\mathbf{x}).$$

Indeed, in practice, situations like this often occur, perhaps because $q(\mathbf{y}|\mathbf{h})$ is not good yet. Our solution to this problem is to use a better proposal distribution $q'(\mathbf{y}|\mathbf{h})$ which is defined as

$$q'(\mathbf{y}'|\mathbf{h}) = \frac{1}{2}q(\mathbf{y}|\mathbf{h}) + \frac{1}{2}\delta_{\mathbf{y}}(\mathbf{y}'),$$

where $\mathbf{y}$ is the corresponding label for $\mathbf{x}$ and $\delta_{\mathbf{y}}$ is a Dirac measure defined for a given $\mathbf{y}'$ by 1 if $\mathbf{y} = \mathbf{y}'$ and 0 otherwise.

Now if we use $q(\mathbf{h}|\mathbf{x})$ and $q'(\mathbf{y}|\mathbf{h})$ as proposal distributions to estimate $A(\mathbf{x})$, we obtain

$$\begin{aligned} A(\mathbf{x}) &= \sum_{\mathbf{h},\mathbf{y}} \sqrt{p(\mathbf{x}, \mathbf{h}, \mathbf{y})\ q(\mathbf{h}|\mathbf{x})q(\mathbf{y}|\mathbf{h})} \\ &= \mathbb{E}_{\substack{\mathbf{h}\sim q(\mathbf{h}|\mathbf{x}) \\ \mathbf{y}\sim q(\mathbf{y}|\mathbf{h})}} \frac{1}{q'(\mathbf{y}|\mathbf{h})} \sqrt{\frac{p(\mathbf{x}, \mathbf{h}, \mathbf{y})q(\mathbf{y}|\mathbf{h})}{q(\mathbf{h}|\mathbf{x})}} \\ &\simeq \sum_{i=1}^{K} \frac{1}{q'(\mathbf{y}^{(i)}|\mathbf{h}^{(i)})} \sqrt{\frac{p(\mathbf{x}, \mathbf{h}^{(i)}, \mathbf{y}^{(i)})q(\mathbf{y}^{(i)}|\mathbf{h}^{(i)})}{q(\mathbf{h}^{(i)}|\mathbf{x})}} = \tilde{A}(\mathbf{x}) \end{aligned} \tag{4.11}$$

where $\mathbf{h}^{(i)}$ is generated by $q(\mathbf{h}|\mathbf{x})$ and $y^{(i)}$ is generated by $q'(\mathbf{y}|\mathbf{h}^{(i)})$ for every $i = 1, \ldots, K$ where K is the number of generated samples. Of course if we use more than two samples, the problem of optimizing $-\log p^*(\mathbf{y}|\mathbf{x})$ can be solved using this proposal distribution.

4.6 A new direction

As previously discussed (Sec. 4.4), deeper layers in a BiHM model might not be appropriate for use in semi-supervised problems and it might be more helpful to use the lower latent layer representations. More formally, assume that $\mathbf{x}$ denotes an observed random variable vector and $\mathbf{h}_1, \mathbf{h}_2$ denote a pair of latent random variable vectors. Also assume that p and q are directed graphical models from $\mathbf{h}_2$ to $\mathbf{x}$ and vice versa that can be factorized as

$$p(\mathbf{x}, \mathbf{y}, \mathbf{h}_1, \mathbf{h}_2) = p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)p(\mathbf{y}|\mathbf{x}, \mathbf{h}_1),$$

and

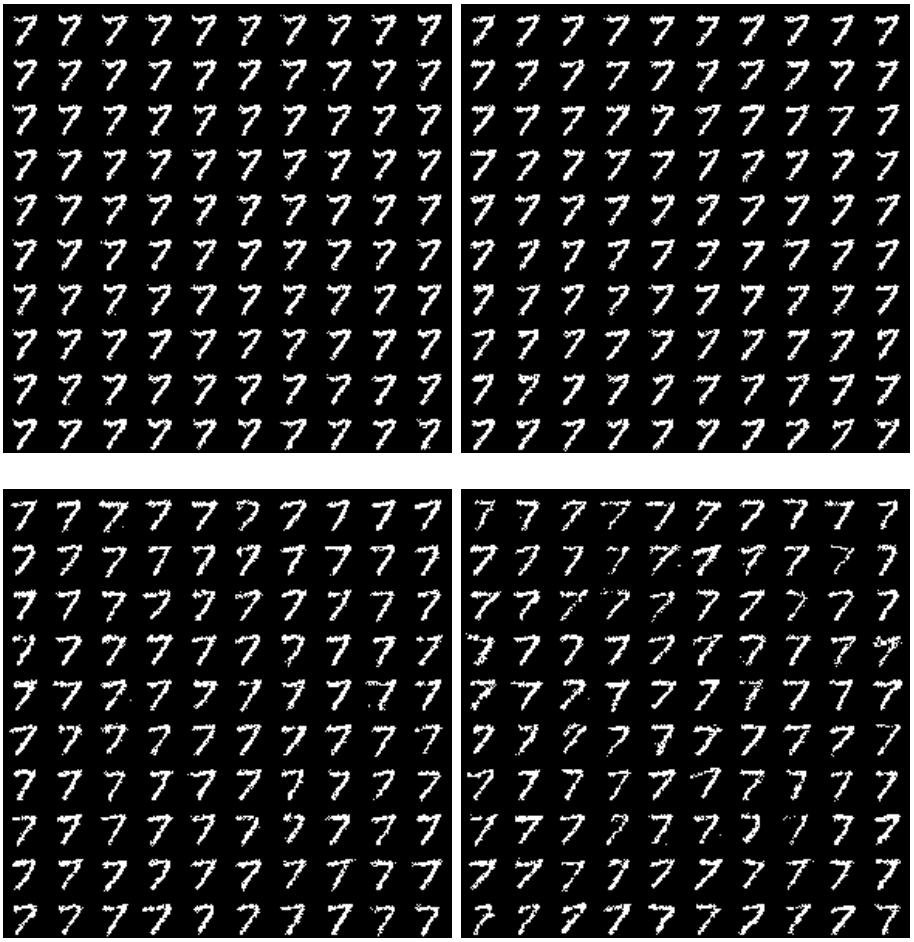

Figure 4.3 – Random generated sample from top-down model p starting from layer 1 (top left), 2 (top right), 3 (bottom left), and 4 (bottom right).

$$q(\mathbf{x}, \mathbf{y}, \mathbf{h}_1, \mathbf{h}_2) = q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})q(\mathbf{x})q(\mathbf{y}|\mathbf{x}, \mathbf{h}_1),$$

where $p(\mathbf{h}_2)$, $p(\mathbf{h}_1|\mathbf{h}_2)$, $p(\mathbf{x}|\mathbf{h}_1)$, $q(\mathbf{h}_2|\mathbf{h}_1)$, $q(\mathbf{h}_1|\mathbf{x})$, and $q(\mathbf{y}|\mathbf{x}, \mathbf{h}_1)$ belong to parametrized families of distributions which can be evaluated and sampled from efficiently. They difference with our previous formulation is that the conditoning on the label occurs only in the lower layer. Moreover, we assume $p(\mathbf{y}|\mathbf{x}, \mathbf{h}_1) = q(\mathbf{y}|\mathbf{x}, \mathbf{h}_1)$ and $p^*(\mathbf{x}) = q(\mathbf{x})$. The associated joint probability p^* to p and q is given by

$$p^*(\mathbf{x}, \mathbf{y}, \mathbf{h}_1, \mathbf{h}_2) = \frac{1}{Z}\sqrt{p(\mathbf{x}, \mathbf{y}, \mathbf{h}_1, \mathbf{h}_2)q(\mathbf{x}, \mathbf{y}, \mathbf{h}_1, \mathbf{h}_2)},$$

where Z is the normalization constant. As before, we are interested in evaluating three terms. The first one is:

$$
\begin{aligned}
p^*(\mathbf{x}) &= \sum_{\mathbf{y},\mathbf{h}_1,\mathbf{h}_2} p^*(\mathbf{x},\mathbf{y},\mathbf{h}_1,\mathbf{h}_2) \\
&= \frac{1}{Z^2}\left[\sum_{\mathbf{y},\mathbf{h}_1,\mathbf{h}_2} q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)\sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})}\right]^2 \\
&= \frac{1}{Z^2}\left[\sum_{\mathbf{h}_1,\mathbf{h}_2} \sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})}\sum_{\mathbf{y}} q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)\right]^2 \\
&= \frac{1}{Z^2}\left[\sum_{\mathbf{h}_1,\mathbf{h}_2} \sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})}\right]^2 \\
&\geq \left[\sum_{\mathbf{h}_1,\mathbf{h}_2} \sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})}\right]^2.
\end{aligned}
$$

Therefore, $p^*(\mathbf{x}) \geq A(\mathbf{x})^2$ where

$$
A(\mathbf{x}) = \sum_{\mathbf{h}_1,\mathbf{h}_2} \sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})},
$$

which can be estimated as

$$
\begin{aligned}
A(\mathbf{x}) &= \sum_{\mathbf{h}_1,\mathbf{h}_2} \sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})} \\
&= \mathbb{E}_{\mathbf{h}_2\sim q(\mathbf{h}_2|\mathbf{h}_1);\mathbf{h}_1\sim q(\mathbf{h}_1|\mathbf{x})}\sqrt{\frac{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)}{q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})}} \\
&\simeq \frac{1}{K}\sum_{k=1}^{K}\sqrt{\frac{p(\mathbf{h}_2^{(k)})p(\mathbf{h}_1^{(k)}|\mathbf{h}_2^{(k)})p(\mathbf{x}|\mathbf{h}_1^{(k)})}{q(\mathbf{h}_2^{(k)}|\mathbf{h}_1^{(k)})q(\mathbf{h}_1^{(k)}|\mathbf{x})}},
\end{aligned}
$$

with $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1|\mathbf{x})$ and $\mathbf{h}_2^{(k)} \sim q(\mathbf{h}_2|\mathbf{h}_1^{(k)})$ for every $k = 1,\ldots,K$ where K is the number of generated samples.

The second term is:

$$
\begin{aligned}
p^*(\mathbf{x},\mathbf{y}) &= \sum_{\mathbf{h}_1,\mathbf{h}_2} p^*(\mathbf{x},\mathbf{y},\mathbf{h}_1,\mathbf{h}_2) \\
&= \frac{1}{Z}\sum_{\mathbf{h}_1,\mathbf{h}_2} q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)\sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})} \\
&= \frac{1}{Z}\sum_{\mathbf{h}_1,\mathbf{h}_2} \sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})} \\
&\geq \sum_{\mathbf{h}_1,\mathbf{h}_2} q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)\sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})}.
\end{aligned}
$$

In this case, we have $p^*(\mathbf{x},\mathbf{y}) \geq B(\mathbf{x},\mathbf{y})$ where

$$
B(\mathbf{x},\mathbf{y}) = \sum_{\mathbf{h}_1,\mathbf{h}_2} q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)\sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})}.
$$

Then $B(\mathbf{x},\mathbf{y})$ can be evaluated as follows:

$$
\begin{aligned}
B(\mathbf{x},\mathbf{y}) &= \sum_{\mathbf{h}_1,\mathbf{h}_2} q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)\sqrt{p(\mathbf{h}_2)p(\mathbf{h}_1|\mathbf{h}_2)p(\mathbf{x}|\mathbf{h}_1)q(\mathbf{h}_2|\mathbf{h}_1)q(\mathbf{h}_1|\mathbf{x})} \\
&\simeq \frac{1}{K}\sum_{k=1}^{K} q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)\sqrt{\frac{p(\mathbf{h}_2^{(k)})p(\mathbf{h}_1^{(k)}|\mathbf{h}_2^{(k)})p(\mathbf{x}|\mathbf{h}_1^{(k)})}{q(\mathbf{h}_2^{(k)}|\mathbf{h}_1^{(k)})q(\mathbf{h}_1^{(k)}|\mathbf{x})}},
\end{aligned}
$$

with $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1|\mathbf{x})$ and $\mathbf{h}_2^{(k)} \sim q(\mathbf{h}_2|\mathbf{h}_1^{(k)})$ for every $k = 1,\ldots,K$ where K is the number of samples. The last term is $p^*(\mathbf{y}|\mathbf{x})$:

$$
p^*(\mathbf{y}|\mathbf{x}) = \frac{p^*(\mathbf{x},\mathbf{y})}{p^*(\mathbf{x})} = \frac{B(\mathbf{x},\mathbf{y})}{A(\mathbf{x})}
$$

One of the important properties of this model is that the gradient of $p^*(\mathbf{x})$ with respect to the parameters of $q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)$ is zero. It means that we do not update the parameters of $q(\mathbf{y}|\mathbf{x},\mathbf{h}_1)$ when the label is not given.

Experimental evaluation of this model is left as future work.

5 Conclusion and Future Work

This thesis presented two research articles addressing challenges in the context of the training of deep generative models for unsupervised and semi-supervised learning problems. The first article introduced a new approach for defining and training deep generative models; and the second one presented two different approaches to use BiHM for a semi-supervised learning task. In the following the main results will be summarized and discussed.

5.1 Summary and Conclusion

It has been argued previously that generative models with multiple hidden layers have the potential to capture higher-level abstractions [36, 2]. Such abstractions can lead to a better generalization. However, training such generative models is a major challenge. Although there has been progress in dealing with continous-valued latent variables building a hierarchy of representations, especially with discrete-valued latent variables, remains a challenge [49]. In this thesis, we introduced a new approach to this problem referred as bidirectional Helmholtz machine. To present this model in details, it was required to provide some necessary concepts from machine learning, probability theory, and probabilistic graph theory in chapter 2.

Chapter 3 theoretically and empirically analyzed the bidirectional Helmholtz machine. While it is common knowledge that more powerful inference methods can gain better approximations of true posterior [40, 92], it is shown that this new model regularize the top-down model such that the generative model stays close to the approximate inference model and vice versa. In fact, this is achieved by interpreting both generative and inference models as approximate inference models for our actual generative model which is defined to be the geometric mean over the top-down and bottom-up approximate inference models.

The work in chapter 4 propose a BiHM model for semi-supervised problems. Since the computation of the exact posterior distribution is intractable, a lower bound on the marginal likelihood of the model is driven. This lower bound ensures that our generative model is as close as possible to the recognition model. Moreover, an importance sampling based estimate for the gradient of this lower bound with respect to the parameters of the model is proposed. Then the properties of such models theoretically and empirically are analyzed.

5.2 Future Work

Most of the work presented in this thesis focused on a new generative model BiHM in which a wide range of potential choices for our parametrized conditional distributions could be used. However, in this thesis, BiHMs with binary latent variables are analyzed theoretically and empirically. It would be important to transfer some of the concepts and ideas to BiHMs with real valued hidden variables. Importantly, it could be used for constructing models with real valued hidden and visible variables with a Gaussian conditional distribution.

Furthermore, BiHMs consist of the bottom-up and top-down directed models which are parameterized with a fully connected neural network. It is interesting to investigate if these models can be parameterized with a a convolutional network. It could be a key factor in the success of BiHMs for problems with complex data distributions such as images and sound.

Another approach is to use modeling sequences based on the BiHMs which could be a sequence of BiHMs. This lead to an extension of the BiHM that can be used to model time-series data.

Bibliography

[1] Adams, R. P. and Z. Ghahramani (2009). Archipelago: nonparametric Bayesian semi-supervised learning. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[2] Bengio, Y. (2009). Learning deep architectures for AI. In *Foundations and Trends in Machine Learning 2* (1), pp. 1-127, Also published as a book. Now Publishers, 2009.

[3] Bengio, Y., O. Delalleau, and N. L. Roux (2006). Label propagation and quadratic criterion. In *Semi-Supervised Learning*, pp. 193-216.

[4] Bengio, Y., P. Lamblin, D. Popovici, and H. Larochelle (2007). Greedy layer-wise training of deep networks. In B. Scholkopf, J. Platt, and T. Homan (Eds.), In *Advances in Neural Information Processing Systems* 19 (NIPS'06), pp. 153-160, MIT Press.

[5] Bengio, Y. and Y. LeCun (2007). Scaling learning algorithms towards AI. In *Large-Scale Kernel Machines*, ed. L. Bottou, O. Chapelle, D. DeCoste, J Weston, pp. 321-360, Cambridge, MA: MIT Press.

[6] Bergstra, J., O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Des-jardins, J. Turian, D. Warde-Farley, and Y. Bengio (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the 12th Python in Science Conference*.

[7] Bhattacharyya, A. (1943). On a measure of divergence between two statistical populations defined by their probability distribution. In *Bulletin of the Calcutta Mathematical Society*, 35, pp. 99-110.

[8] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[9] Blum, A., J. Lafferty, M. R. Rwebangira, and R. Reddy (2004). Semi-supervised learning using randomized mincuts. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[10] Bornschein, J. and Y. Bengio (2015). Reweighted wake-sleep. In *International Conference on Learning Representations (ICLR)*.

[11] Burda, Y., R. Grosse, and R. Salakhutdinov (2015). Importance weighted autoencoders. arXiv preprint arXiv : 1509.00519.

[12] Chapelle, O., J. Weston, and B. Scholkopf (2003). Cluster kernels for semi-supervised learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 585-592. Cambridge, MA, USA: MIT Press.

[13] Chapelle, O. and A. Zien (2005). Semi-supervised classification by low density separation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 57-64.

[14] Choromanska, A., M. Henaff, M. Mathieu, G. B. Arous, Y. LeCun (2015). The loss surfaces of multilayer networks, *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[15] Cortes, C. and V. Vapnik (1995). Support vector networks. In *Machine Learning 20*, pp. 273-297.

[16] Courville, A., J. Bergstra, and Y. Bengio (2011). Unsupervised models of images by spikeand-slab RBMs. In *L. Getoor and T. Scheffer, editors, Proceedings of the 28th International Conference on Machine Learning (ICML)*, pp. 1145-1152. ACM.

[17] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. In *Mathematics of Control, Signals, and Systems 2*, pp. 303-314.

[18] Dauphin, Y., R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y.Bengio (2014). Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Neural Information Processing Systems (NIPS)*, 27, pp. 2933 ?2941.

[19] Dayan, P. (2000). *Helmholtz machines and wake-sleep learning. Handbook of Brain Theory and Neural Network*. MIT Press, Cambridge, MA, 44(0).

[20] Dayan, P. and G. E. Hinton (1996). Varieties of Helmholtz machine. In *Neural Networks*, 9(8), pp. 1385-1403.

[21] Dayan, P., G. E. Hinton, M. R. Neal, and R. S. Zemel (1995). The Helmholtz machine. In *Neural computation*, 7(5), pp. 889-904.

[22] Delalleau, O., Y. Bengio, N.Le Roux (2005). Efficient non-parametric function induction in semi-supervised learning. In *International Conference on Artificial Intelligence and Statistics (AISTAT)*.

[23] Deng, L., G. E. Hinton, and B. Kingsbury (2013). New types of deep neural network learning for speech recognition and related applications: An overview. In *Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP)*.

[24] Deng, L., J. Li, K. Huang, Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero (2013). Recent advances in deep learning for speech research at Microsoft. In *Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP)*.

[25] Dietterich, T., G. and G. Bakiri (1995). Solving multiclass learning problems via error-correcting output codes. arXiv preprint cs/9501101.

[26] Drucker, H., C. J. Burges, L. Kaufman, C. J. C, B. L. Kaufman, A. Smola, and V. Vapnik (1996). *Support vector regression machines*.

[27] Duchi, J., E. Hazan, and Y. Singer (2010). Adaptive subgradient methods for online learning and stochastic optimization. In *Journal of Machine Learning Research*, 12, pp. 2121-2159.

[28] Elfadel, I. (1995). Convex potentials and their conjugates in analog mean-field optimization. In *Neural Computation 7*, pp. 1079-1104.

[29] Fergus, R., Y. Weiss, and A. Torralba (2009). Semi-supervised learning in gigantic image collections. In *Advances in Neural Information Processing Systems (NIPS)*.

[30] Fukunaga, K. and L. Hostetler (1975, January). The estimation of the gradient of a density function, with applications in pattern recognition. In *Information Theory, IEEE Transactions on 21* (1), pp. 32-40.

[31] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. In *Biological Cybernetics 36*, pp. 193-202.

[32] Gehler, P. V., A. D. Holub, and M. Welling (2006). The rate adapting poisson model for information retrieval and object recognition. In *W. Cohen and A. Moore, editors, Proceedings of 23rd International Conference on Machine Learning (ICML)*, pp. 337-344, ACM.

[33] Glorot, X. and Y. Bengio (2010). Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[34] Hinton, G. E., P. Dayan, B. J. Frey, and R. M. Neal (1995). The wake-sleep algorithm for unsupervised neural networks. In *Science*, 268, pp. 1558-1161.

[35] Hinton, G. E., L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury (2012). Deep neural networks for acoustic modeling in speech recognition. In *IEEE Signal Processing Magazine, 29(6)*, pp. 82-97.

[36] Hinton, G. E., S. Osindero, and Y. Teh (2006). A fast learning algorithm for deep belief nets. In *Neural Computation 18*, pp. 1527-1554.

[37] Hinton, G. E., R. Salakhutdinov (2006). Reducing the dimensionality of data with neural networks. In *Science*, 313(5786), pp. 504-507.

[38] Hinton, G. E. and T. Sejnowski (1983). Optimal perceptual inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[39] Hinton, G. E. and R. S. Zemel (1994). Autoencoders, minimum description length, and Helmholtz free energy. In *Advances in Neural Information Processing Systems 6*. J. D. Cowan, G. Tesauro and J. Alspector (Eds.), Morgan Kaufmann: San Mateo, CA.

[40] Hjelm, R. D., K. Cho, J. Chung, R. Salakhutdinov, V. Calhoun, N. Jojic (2016). Iterative refinement of approximate posterior for training directed belief networks, In *Proceedings of the International Conference on Learning Representations (ICLR)*, arXiv preprint arXiv :1511.06382.

[41] Ho, T. K. (1995). Random decision forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, Montreal, QC, 14-16, pp. 278-282.

[42] Hornik, K., M. Stinchcombe, and H. White (1989). Multilayer feedforward networks are universal approximators. In *Neural Networks 2*, pp. 359-366.

[43] Ivakhnenko, A. G. and V. G. Lapa (1965). Cybernetic predicting devices. In *CCM Information Corporation*.

[44] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceeding of the International Conference on Machine Learning (ICML)*, V. 99, pp. 200-209.

[45] Kemp, C., T. L. Griffiths, S. Stromsten, and J. B. Tenenbaum (2003). Semi-supervised learning with trees. In *Advances in Neural Information Processing Systems (NIPS)*.

[46] Kindermann, R. (1980). *Markov Random Fields and Their Applications (Contemporary Mathematics*; *V. 1)*. American Mathematical Society.

[47] Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv :1412.6980.

[48] Kingma, D. P., D. J. Rezende, S. Mohamed, and M. Welling (2014). Semi-supervised learning with deep generative models. arXiv preprint arXiv :1406.5298.

[49] Kingma, D. P. and M. Welling (2014). Auto-encoding variational bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[50] Kivinen J. and C. Williams (2012). Multiple texture Boltzmann machines. In *N. Lawrence and M. Girolami, editors, Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, V. 22 of JMLR W&CP, pp. 638-646. .

[51] Koller, D. and N. Friedman (2009). *Probabilistic graphical models*: *principles and techniques*. MIT Press.

[52] Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 1*, pp. 1097-1105.

[53] Kschischang, F., B. Frey, and H. A. Loeliger (2001). Factor graphs and the sum-product algorithm. In *IEEE Transactions on Information Theory 47*, pp. 498-519.

[54] Kullback S. (1959). *Information Theory and Statistics*, John Wiley and Sons.

[55] Kullback, S. (1987). Letter to the Editor: The Kullback-Leibler distance. In *The American Statistician*, 41(4), pp. 40-341.doi101080/00031305.1987.10475510. JSTOR 2684769.

[56] Kullback, S. and R. A. Leibler (1951). On information and sufficiency. In *Annals of Mathematical Statistics 22*, pp. 49-86. .

[57] Larochelle, H. (2011). Binarized mnist dataset. URL http://www.cs.toronto.edu/~larocheh/public/datasets/binarized_mnist/binarized_mnist_[train|valid|test].amat.

[58] Larochelle H. and Y. Bengio (2008). Classification using discriminative restricted Boltzmann machines. In W. W. Cohen, A. McCallum, and S. T. Roweis, editors, In *Proceedings of the 25th International Conference on Machine learning (ICML)*, pp. 536-543, ACM.

[59] Larochelle H., Y. Bengio, J. Louradour , P. Lamblin, (2008). Exploring strategies for training deep neural networks. In *Journal of Machine Learning Research*, pp. 1-40

[60] Larochelle, H. and I. Murray (2011). The neural autoregressive distribution estimator. In *Journal of Machine Learning Research*, V. 15, pp. 29-37.

[61] Lawrence, N. D. and M. I. Jordan (2005). Semi-supervised learning via Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS)*.

[62] Le Roux, N., N. Heess, J. Shotton, and J. M. Winn (2011). Learning a generative model of images by factoring appearance and shape. In *Neural Computation*, 23(3), pp. 593-650.

[63] Li, P., Y. Ying, and C. Campbell (2009). A variational approach to semi-supervised clustering. In *Proceedings of the European Symposium on Artificial Neural Networks (ESANN)*, pp. 11-16.

[64] Liang, P. (2005). *Semi-supervised learning for natural language*. PhD thesis, Massachusetts Institute of Technology.

[65] Liu, Y. and K. Kirchhoff (2013). Graph-based semi-supervised learning for phone and segment classification. In *Proceedings of Interspeech*.

[66] McCulloch, W. S., and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. In *Bulletin of Mathematical Biophysics*, V. 5, pp. 115-133.

[67] Mitchell, T. M. (1997). *Machine learning*. New York: McGraw-Hill.

[68] Mnih, A. and K. Gregor (2014). Neural variational inference and learning in belief networks. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*.

[69] Mohamed, A., G. E. Dahl, G. E. Hinton (2012). Acoustic modeling using deep belief networks. In *IEEE Transactions on Audio, Speech, and Language Processing (20)*, pp. 14-22.

[70] Murray, I. and R. Salakhutdinov (2009). Evaluating probabilities under high-dimensional latent variable models. In *Advances in Neural Information Processing Systems (NIPS)*, V. 21, pp. 1137-1144.

[71] Nadaraya, E. A. (1964). On estimating regression. In *Theory of Probability and its Applications* 9(1), pp. 141-142.

[72] Neal, R. M. (1992). Connectionist learning of belief networks, In *Artificial Intelligence*, V. 56, pp. 71-113.

[73] Neal, R. M. (2001, April). Annealed importance sampling. In *Statistics and Computing 11* (2), pp. 125-139.

[74] Neal, R. and G. Hinton (1999). A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan (Ed.), *Learning in Graphical Models*. Cambridge, MA: MIT Press.

[75] Netzer, Y., T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng (2011). Reading digits in natural images with unsupervised feature learning. Deep Learning and Unsupervised Feature Learning Workshop, In *Advances in Neural Information Processing Systems (NIPS)*.

[76] Newey, W. and D. McFadden (1994). Large sample estimation and hypothesis testing. *Handbook of Econometrics 4*, pp. 2111-2245.

[77] Nock, R. and F. Nielsen (2006). On weighting clustering. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8), pp. 1-13.

[78] Pal, C., C. Sutton, and A. McCallum (2005). Fast inference and learning with sparse belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*.

[79] Pearl, J. (1985, August). Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine*, pp. 329-334.

[80] Pitelis, N., C. Russell, and L. Agapito (2014). Semi-supervised learning using an unsupervised atlas. In *Proceddings of the European Conference on Machine Learning (ECML)*, V. LNCS 8725, pp. 565-580.

[81] Ranzato, M. and M. Szummer (2008). Semi-supervised learning of compact document representations with deep networks. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pp. 792-799.

[82] Rezende, D. J., S. Mohamed, and D. Wierstra (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*.

[83] Rifai, S., Y. Dauphin, P. Vincent, Y. Bengio, and X. Muller (2011). The manifold tangent classifier. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2294-2302.

[84] Rifai, S., G. Mesnil, P. Vincent, X. Muller, Y. Bengio, Y. Dauphin, and X. Glorot (2011). Higher order contractive auto-encoder. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*.

[85] Robert, P. C. and G. Casella (2009). *Introducing Monte Carlo methods with R*. Springer Science and Business Media.

[86] Rokach, L. and O. Maimon (2005). *Clustering methods. Data mining and knowledge discovery handbook*. Springer US, pp. 321-352.

[87] Rosenberg, C., M. Hebert, and H. Schneiderman (2005). Semi-supervised self-training of object detection models. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05)*.

[88] Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. In *D. E. Rumelhart and J. L. McClelland (Eds.), Parallel Distributed Processing*, V. 1, Chapter 8, pp. 318-362. Cambridge, MIT Press.

[89] Salakhutdinov, R. and G. E. Hinton (2009). Deep Boltzmann machines. In *International Conference on Artificial Intelligence and Statistics*, pp. 448-455.

[90] Salakhutdinov, R. and G. E. Hinton (2009). Replicated softmax: an undirected topic model. In *Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, Advances in Neural Information Processing Systems (NIPS 22)*, pp. 1607-1614, MIT Press.

[91] Salakhutdinov, R., A. Mnih, and G. E. Hinton (2007). Restricted Boltzmann machines for collaborative filtering. In *Z. Ghahramani, editor, Proceedings of the 24th International Conference on Machine Learning (ICML)*, pp. 791-798, ACM.

[92] Salimans, T. and D. Knowles (2013). Fixed-form variational posterior approximation through stochastic linear regression. In *Bayesian Analysis*, V. 8, pp. 837-882.

[93] Seide, F., G. Li, X. Chen, and D. Yu (2011). Feature engineering in context-dependent deep neural networks for conversational speech transcription. In *Proceedings of the Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 24-29.

[94] Shannon, C. E. (1948). A mathematical theory of communication. In *The Bell System Technical Journal*, 27(3), pp. 379-423 and 623-656.

[95] Shi, M. and B. Zhang (2011). Semi-supervised learning improves gene expression-based prediction of cancer recurrence. In *Bioinformatics*, 27(21), pp. 3017-3023.

[96] Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland (Eds.), In *Parallel Distributed Processing*, V. 1, Chapter 6, pp. 194-281, Cambridge: MIT Press.

[97] Sohl-Dickstein, J., E. A. Weiss, N. Maheswaranathan, and S. Ganguli (2015). Deep unsupervised learning using nonequilibrium thermodynamics. CoRR, abs/1503.03585. URL http://arxiv.org/abs/1503.03585.

[98] Steinhaus, H. (1957). Sur la division des corps materiel en parties. In *Bull. Acad. Polon. Sci.*, pp. 801-804.

[99] Stinchcombe, M. and H. White (1989). Universal approximation using feedforward networks with non-sigmoid hidden layer activation function. In *International Joint Conference on Neural Networks (IJCNN)*, Washington DC, pp. 613-617, IEEE.

[100] Stuhlmuller, A., J. Taylor, and N. Goodman (2013). Learning stochastic inverses. In *Advances in Neural Information Processing Systems*, pp. 3048-3056.

[101] Susskind, J. M., A. K. Anderson, and G. E. Hinton (2010). The Toronto face dataset. Technical Report UTML TR 2010-001, University of Toronto.

[102] Sutskever, I., J. Martens, G. Dahl, and G. E. Hinton (2013). On the importance of initialization and momentum in deep learning. In *International Conference on Machine Learning (ICML)*.

[103] Tang, Y. and R. Salakhutdinov (2013). Learning stochastic feedforward neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 530-538.

[104] Tang, Y., R. Salakhutdinov, and G. E. Hinton (2012). Robust Boltzmann machines for recognition and denoising. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2264-2271.

[105] Titterington, D., A. Smith, and U. Makov (1985). *Statistical analysis of finite mixture distributions*. Wiley, New York.

[106] Van Merriënboer, B., D. Bahdanau, V. Dumoulin, D. Serdyuk, D. Warde-Farley, J. Chorowski, and Y. Bengio (2015). Blocks and fuel: Frameworks for deep learning. ArXiv e-prints, (1506.00619), URL http://adsabs.harvard.edu/abs/2015arXiv150600619V.

[107] Vincent, P., H. Larochelle, Y. Bengio, and P. A. Manzagol (2008). Extracting and composing robust features with denoising autoencoders. In *International Conference on Machine Learning (ICML)*.

[108] Wang, Y., G. Haffari, S. Wang, and G. Mori (2009). A rate distortion approach for semi-supervised conditional random fields. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2008-2016.

[109] Watson, G. S. (1964). Smooth regression analysis. In *Sankhyā: The Indian Journal of Statistics*, Series A, 26(4), pp. 359-372.

[110] Weston, J., F. Ratle, H. Mobahi, and R. Collobert (2012). Deep learning via semi-supervised embedding. In *Neural Networks*: *Tricks of the Trade*, Springer, pp. 639-655, Springer.

[111] Widrow, B. and M. E. Hoff (1960). Adaptive switching circuits. In *IRE WESCON Convention Record*, V. 4, pp. 96-104, Reprinted in *Anderson and Rosenfeld* (1988).

[112] Xing, E. P., R. Yan, and A. G. Hauptmann (2005). Mining associated text and images with dual-wing harmoniums. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI)*. AUAI Press.

[113] Zhu, X. (2006). Semi-supervised learning literature survey. Technical report, Computer Science, University of Wisconsin-Madison.

[114] Zhu, X., Z. Ghahramani, and J. Lafferty (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceddings of the International Conference on Machine Learning (ICML)*, V. 3, pp. 912-919.

Index