Review on Representation Learning Using Directed Graphical Models with Multiple Hidden Layers

Samud Ahmadov **Navie Chan** **Amir Hossein Heidari Zadi**

Abstract

With the resurgence of interest in training deep directed graphical models (DGM) with multiple hidden layers, we present an in-depth review of four influential papers in this area. We review the methods, assumptions and shortcomings of each paper, and discuss the unifying theme underlying them. We also discuss the results of our own experiments on the more complex Toronto Face data set (compared to MNIST which the original papers rely on), and compare the results of each of the algorithms.

1 Introduction

The fundamental aim of learning is to discover the structures present in the data. This could mean learning features or latent variables that are useful to capture the dependencies between the variables. One approach is to build parametrized generative models, that contain multiple layers of latent variables, that have been argued to have the potential to generalize better and capture more high-level abstractions (Bengio et al. [2009]). One major difficulty in learning such models is to be able to perform efficient inference over the models that are often intractable using naive methods. We present and discuss papers that have focused on solving this issue through various approximation techniques. As our starting point we start with "Auto-Encoding Variational Bayes" by Kingma and Welling [2014] and its generalization "Importance Weighted Autoencoders" presented by Burda et al. [2015a]. We then show how these models relate to a much older algorithm presented by Hinton et al. [1995] in "The wake-sleep algorithm for unsupervised neural networks", and then we also discuss a generalization over the wake-sleep algorithm in "Reweighted Wake-Sleep" presented by Bornschein and Bengio [2014a].

2 Description of the problem

The general problem of unsupervised learning is to find a good hidden representation of data.To do so, we assume that the observed data x has been generated by some random process, involving some latent variable z. We think of the latent space as an economical representation of data (since it is lower-dimensional than the original distribution). So the process has 2 steps 1) sample z from prior $p(z)$ and 2) generate x by using conditional distribution $p(x|z)$.
Learning the posterior distribution $p(z|x)$ involves an intractable normalization constant $p(x) = \int_z p(x|z)p(z)$.
We therefore need to find methods that can get past this intractable integral by some form of efficient approximation, and that they can scale to a large data set. e.g. cannot perform expensive operations such as Markov Chain Monte Carlo (MCMC).

3 Auto-encoding variational bayes

Variational auto-encoders (VAE) as proposed in Kingma and Welling [2014] is a type of directed graphical model, with continuous latent variables that has an intractable posterior distribution

$p_\theta(z \mid x)$.
Using the techniques of variational inference, this paper tries to model the true posterior distribution $p_\theta(z \mid x)$ with an approximate distribution $q_\phi(z \mid x)$. However, the significance of the work of Kingma and Welling [2014] comes from coming up with a Monte Carlo estimate of the variational lower bound that can be maximized efficiently using gradient ascent.
This enables us to learn more complex approximate posterior q that would not suffer from assumptions made in methods such as the mean field, which requires the approximate posterior to be fully factorizable.
From a coding theory perspective, the latent variables z can be thought of as a code with an economical representation for the observation x. In this regard we can think of $q_\phi(z \mid x)$ as a probabilistic encoder, and $p_\theta(x \mid z)$ as the probabilistic decoder.

3.1 The variational bound

Given dataset $\mathbf{X} = \{x^{(i)}\}_{i=1}^N$ with N number of i.i.d samples. We can write marginal likelihood as follow (the derivation can be found in the Appendix B):

$$\log p_\theta(x^{(i)}) = D_{KL}(q_\phi(z \mid x^{(i)} \mid\mid p_\theta(z \mid x^{(i)}) + \mathcal{L}(\theta, \phi; x^{(i)}) \tag{1}$$

Since the Kullback–Leibler (KL) divergence is a positive term, we can treat the term $\mathcal{L}$ of eq. (1) as a lower bound.

$$\log p_\theta(x^{(i)}) \geq \mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x^{(i)})}[-\log q_\phi(z \mid x^{(i)}) + \log p_\theta(x^{(i)}, z)] \tag{2}$$

which we can reformulate $\mathcal{L}$ as follows:

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x^{(i)})}[-\log q_\phi(z \mid x^{(i)}) + \log p_\theta(x^{(i)} \mid z) + \log p_\theta(z)] \tag{3}$$

$$\mathcal{L}(\theta, \phi; x^{(i)}) = \mathbb{E}_{q_\phi(z|x^{(i)})}\left[\log \frac{p_\theta(z)}{q_\phi(z \mid x^{(i)})} + \log p_\theta(x^{(i)} \mid z)\right] \tag{4}$$

$$\mathcal{L}(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z \mid x^{(i)} \mid\mid p_\theta(x^{(i)}) + \mathbb{E}_{q_\phi(z|x^{(i)})}[\log p_\theta(x^{(i)} \mid z)] \tag{5}$$

In order to minimize the KL divergence, we want to calculate the gradient of $\mathcal{L}$ w.r.t global parameters θ and ϕ. We could use naive Monte Carlo gradient estimator: $\nabla_\phi \mathbb{E}_{q_\phi(z)}[f(z)] = \mathbb{E}_{q_\phi(z)}[f(z)\nabla_{q_\phi} \log q_\phi(z)] \simeq \frac{1}{L}\sum_{l=1}^{L} f(z)\nabla_{q_\phi(z^{(l)})} \log q_\phi(z^{(l)})$ such that $z^{(l)} \sim q_\phi(z \mid x^{(i)})$. But this naive Monte Carlo gradient estimator has high variance Paisley et al. [2012].

3.2 The SGVB estimator and AEVB algorithm

The naive Monte Carlo gradient estimator has high variance, making it impractical. But Kingma and Welling [2014] proposes a reparametrization trick to create Monte Carlo estimate of the lower bound that can be used with simple optimizers like SGD, while not suffering from high variance. The idea behind reparametrizing latent variable $z \sim q_\phi(z|x)$ using some differentiable function of input x and an auxiliary variable :

$$z = g_\phi(\epsilon, x) \quad \text{where} \quad \epsilon \sim p(\epsilon) \tag{6}$$

If we apply the reparametrization trick to lower bound eq. (2), we can get Monte Carlo gradient estimates as follows:

$$\mathcal{L}^A(\theta, \phi; x^{(i)}) \simeq \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)}, z^{(i,l)}) - \log q_\phi(z^{(i,l)} \mid x^{(i)}) \tag{7}$$

$$\text{where} z^{(i,l)} = g_\phi(\epsilon^{(i,l)}, x^{(i)}) \text{and} \epsilon^{(l)} \simeq p(\epsilon)$$

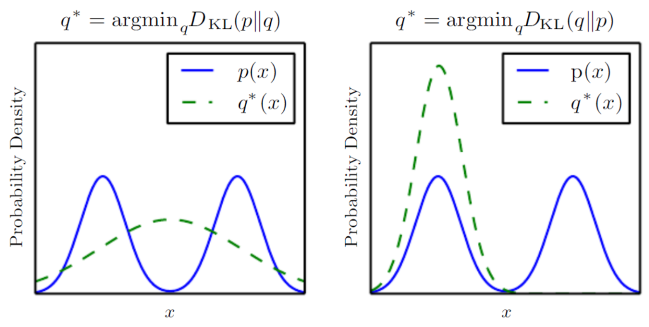

Figure 1: Mode-averaging (left), and mode-collapsing (right) caused by minimizing D_{KL}. Figure from Goodfellow et al. [2016]

While this estimator is still good, if we take a close look at the eq. (5) we can see that if the $D_{KL}(q_\phi(z \mid x^{(i)} \mid\mid p_\theta(x^{(i)})$ part of the equation can be calculated analytically (equation can be found in Appendix B of Kingma and Welling [2014]), we can come up with an even better estimator with a tighter bound than eq. (7).

$$\mathcal{L}^B(\theta, \phi; x^{(i)}) = -D_{KL}(q_\phi(z \mid x^{(i)} \mid\mid p_\theta(x^{(i)}) + \frac{1}{L}\sum_{l=1}^{L} \log p_\theta(x^{(i)} \mid z^{(i,l)}) \tag{8}$$
$$\text{where} \quad z^{(i,l)} = g_\phi(\epsilon^{(i,l)}, x^{(i)}) \quad \text{and} \quad \epsilon^{(l)} \simeq p(\epsilon)$$

After obtaining the lower bound, the derivative of $\nabla_{\phi,\theta}\mathcal{L}^B(\theta, \phi; x^{(i)})$ can be easily taken and simple optimizers like SGD can be used to update the global parameters.

3.3 Issues with VAE

Our goal in both variational inference, and variational auto-encoders is to minimize the Kullback–Leibler probability divergence measure D_{KL} between distributions p and q. However, we note that this measure is not symmetric, minimizing $D_{KL}(q||p)$ will give different behaviour than minimizing $D_{KL}(p||q)$.
Minimizing $D_{KL}(q||p)$, causes q to under-estimate the support of p. This is an issue when the true distribution p has multiple modes that are sufficiently separated, and minimizing $D_{KL}(q||p)$ forces q to choose a single mode. This would visually manifest itself as blurring of generated samples from the generative model $p_\theta(x, z)$. This is of course much more desirable than the mode-averaging (if you assume that the posterior factorizes over the hidden states) caused by minimizing $D_{KL}(p||q)$, where q over-estimates the support of p. This is also called explain-away effect, since two independent variables that share a child node in a DGM, become conditionally dependent when the child is observed.

4 Importance weighted auto-encoder

Importance weighted auto-encoder (IWAE) is another learning algorithm for deep generative models, which uses the same architecture as variational auto-encoder proposed by Kingma and Welling [2014]. IWAE also uses recognition model to approximate intractable true posterior along with generative model. They combine the idea of multiple sampling (Bornschein and Bengio [2014a]) and a clever reformulation of the variational lower bound with the VAE algorithm, which yields tighter lower bound.
While VAE algorithm has been very successful to come up with a differentiable lower bound which can be used with simple optimizers to update global parameters of recognition and generative models,

it has its flaws too. VAE makes the assumption that the approximate posterior is approximately factorial. This assumption constraints the complexity of the posterior that the model can learn. Different than VAE, IWAE proposes to generate multiple approximations from recognition model and average their weights. This gives a tighter lower bound. As more approximate posterior has been generated by recognition model, lower bound gets closer to true log-likelihood.

4.1 Method

As we can see from lower bound equations of VAE eq(5,8), taking some of log-likelihoods constraints the flexibility of the model. If the generative model does not make a good data generation, then VAE estimator heavily penalizes the approximate posterior. However, IWAE uses tighter lower bound, that gives more flexibility to the model. By Jensen's inequality and concavity of logarithm function, the following inequality holds:

$$\mathcal{L}_k(x) = \mathbb{E}_{(z_1,..z_k)\sim q(z|x)}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p(x,z_i)}{q(z_i \mid x)}\right] \leq \log \mathbb{E}_{(z_1,..z_k)\sim q(z|x)}\left[\frac{1}{k}\sum_{i=1}^{k}\frac{p(x,z_i)}{q(z_i \mid x)}\right] \tag{9}$$

All of the latent representation z sampled independently from approximate posterior. The lower bound gets closer into true marginal likelihood as number of samples increased. The proof can be found in Appendix A of Burda et al. [2015a].
IWAE uses similar training procedure as VAE. But there is a small difference between their gradient estimators. VAE uses seperates D_{KL} and computes it analytically, which gives a tighter bound and lower variance (eq 3-5). The same trick cannot be applied here if k > 1, because of the sum inside the log. Note that, VAE is a special case of IWAE, when k = 1. Let $w_i(x,z) = \frac{p(x,z_i)}{q(z_i|x)}$, $\tilde{w} = w_i/\sum_{i=1}^{k} w_i$normalized importance weight and $\epsilon \sim p(\epsilon)$ be the an auxiliary variable used in reparametrization trick in VAE. IWAE uses the same reparametrization trick too.

$$\nabla_\theta \mathcal{L}_k(x) = \nabla_\theta \mathbb{E}_{h_1,...h_k}\left[\log \frac{1}{k}\sum_{i=1}^{k} w_i(x,z)\right] = \nabla_\theta \mathbb{E}_{\epsilon_1,...\epsilon_k}\left[\log \frac{1}{k}\sum_{i=1}^{k} w_i(x, g(\epsilon_i, x))\right] \tag{10}$$

$$= \mathbb{E}_{\epsilon_1,...\epsilon_k}\left[\nabla_\theta \log \frac{1}{k}\sum_{i=1}^{k} w_i(x, g(\epsilon_i, x))\right] \tag{11}$$

$$= \mathbb{E}_{\epsilon_1,...\epsilon_k}\left[\sum_{i=1}^{k} \tilde{w}\nabla_\theta \log w_i(x, g(\epsilon_i, x))\right] \tag{12}$$

Now we can expand the $\nabla_\theta \log w_i(x, g(\epsilon_i, x))$ as follows:

$$\nabla_\theta \log w_i(x, g(\epsilon_i, x)) = \nabla_\theta \log p(x, g(\epsilon_i, x)) - \nabla_\theta \log q(g(\epsilon_i, x) \mid x) \tag{13}$$

5 The wake-sleep algorithm for unsupervised neural networks

Wake-sleep algorithm is an learning algorithm for Helmholtz machines, which are directed graphical models with multiple layers of latent variables z, which can also be thought of as a neural net with stochastic units that has two sets of connections (one using top-down generative weights, and the other bottom-up recognitions weights) as shown in fig. (2) (Appendix A).

The wake-sleep algorithm as proposed by Hinton et al. [1995] views the objective as minimizing the "description length", which is number of bits required to communicate the input to an imaginary recipient. The goal is to come up with an economical representation of the input. This encoded version of the input is communicated by sending state of the hidden variables in the neural net, and the difference between the input and the reconstruction of the input created by the generative network.

In Hinton et al. [1995] a sigmoid belief network (SBN) is used for simplicity, however different types of activation functions could be used too. Note that since this is really a graphical model, the units should be stochastic units, and not deterministic. SBN is just a neural net with stochastic binary units, calculated by the sigmoid function.

$$p(s_v = 1) = \sigma(b_v + W \cdot \mathbf{s}) \tag{14}$$

where s_v is the state of unit v, with bias b_v and parents $\mathbf{s}$. The activation function is the same whether using recognition weights for W, or generative weights.

The algorithm is divided into a "wake" phase and a "sleep" phase. In the wake phase, an input $x^{(i)}$ is sampled from the data set $\mathbf{X}$ (note that this algorithm can be used in an on-line setting), and using the recognition weights the state of each hidden layer is calculated bottom-up starting from first hidden layer z^1, layer-by-layer up to the last layer z^L, which a has factorized unconditional distribution (since it has no parents).

Note that in the wake phase, the recognition weights determine a conditional distribution $q(z \mid x^{(i)})$ over the total hidden state (called "total representation" in Hinton et al. [1995]). In this phase we are therefore sampling from the recognition distribution $q(z \mid x^{(i)})$ and setting the state of the hidden units. Note that we are only making one sample.
This paper proposes a simple weight update rule during this phase that would view recognition weights as fixed and update the generative weights, by minimizing the expected cost $C = -\int_z q(z|x^{(i)}) \log p(x^{(i)}, z)$ using maximum likelihood. Note that this expected cost is the same expected energy term in the following reformulation of the variational lower bound from eq. (3).

$$\mathcal{L}(x^{(i)}) = -\mathbb{H}(q(z \mid x^{(i)})) + \mathbb{E}_{q(x^{(i)}|z)}[E(x^{(i)})] \tag{15}$$

where $E(x^{(i)}) = -\log p(x^{(i)}, z)$, and $p(x, z)$ is the unnormalized posterior distribution given the network's generative weights.
Therefore, we see that even though the units are driven using the recognition weights in the wake phase, only the generative weights get updated. Note that minimizing the expected energy is the same as minimizing the variational lower bound, if the recognition weights are fixed. And this is same as minimizing $D_{KL}(q(z \mid x) \mid\mid p(z \mid x))$.
Note that during the wake phase, in order to calculate the maximum likelihood of the expected cost, we will only differentiate w.r.t. to the generative weights.

We can now how wake-sleep algorithm is trying to minimize the same variational lower bound (or Helmholtz free energy), as VAE with making different assumptions.

During the sleep phase, the units are driven top-down using the generative weights. This is done by performing ancestral sampling starting with the top layer z^L (and only using the biases of the topmost layer, since it contains no parents in the sleep phase), and continuing all the way down to the first layer I as show in fig. (2). Note that this generates an unbiased sample of the network's generative model.

The weight update rule during the sleep phase is exact opposite of the wake phase, with the generative weights being fixed and recognition weights being updated. The weight update formulas are the same, but with p and q interchanged. (Note that the state of the input units are generated during the sleep phase, and unlike the wake phase, we need to include them in them sleep phase update rule).
Note that there is a major drawback here in that during the sleep phase, we are actually minimizing $D_{KL}(p(z \mid x) \mid\mid q(z \mid x))$ (since p and q are interchanged), which causes mode-averaging as discussed in section 3.3.
This issue is the explaining-away effect, where the recognition weights are unable to recover true causes given observed data, since the independent hidden causes become dependent when conditioning on their shared child.

6 Reweighted wake-sleep

Reweighted wake-sleep is another deep generative learning algorithm. It is a new interpretation of wake-sleep algorithm originally proposed by Hinton et al. [1995]. This paper has two major contributions. 1) Generating unbiased estimator of likelihood by using weighted sum of multiple samples from recognition network (and thus generalizing on the original wake-sleep algorithm, with wake-sleep being reweighted wake-sleep with a single sample). 2) It empirically shows that Sigmoid Belief Networks are not powerful enough to estimate the true posterior. The authors have showed using more powerful layer model, like NADE Larochelle and Murray [2011] yields better results.

One of the major problem with original wake-sleep algorithm is generating biased estimator of likelihood. The authors try to solve the problem by using multiple samples from inference network. As the number of samples increase, the estimator becomes less biased.

Let $p(z \mid x)$ be our true posterior and $q(z \mid x)$ approximation of it. Then we can re-write the marginal likelihood as follows:

$$\log p(x) = \sum_z q(z \mid x) \frac{p(x, z)}{q(z \mid x} = \mathbb{E}_{z \sim q(z|x)} \left[\frac{p(x, z)}{q(z \mid x} \right] \simeq \frac{1}{K} \sum_{k=1}^{K} \frac{p(x, z^{(k)})}{q(z^{(k)} \mid x)} \tag{16}$$

This is unbiased estimate of marginal likelihood. But if we take the log of the eq (16), this will give a biased estimate, because logarithm if concave function and it will underestimate the true log-likelihood. Increasing the number of samples will reduce the bias and variance.

6.1 Training procedure

We assume that p and q parametrized by θ and ϕ. As we discussed in Section 5, wake-sleep algorithm has two phases. In the wake phase ϕ is fixed and we update generative model $p_\theta(x, z)$. Then our $\mathcal{L}_p(\theta, x) = logp_\theta(x)$. We take the derivative w.r.t θ

$$\nabla \mathcal{L}_p(\theta, x) = \frac{1}{p(x)} \mathbb{E}_{z \sim q(z|x)} \left[\frac{p(x, z}{q(z \mid x) \nabla \log p(x, z)} \right] \tag{17}$$

$$\simeq \sum_{k=1}^{K} \tilde{w_k} \nabla \log p(x, z^{(k)}) \;\; \text{where} \;\; z^{(k)} \sim q(z \mid x) \tag{18}$$

Note that $\tilde{w_k} = \frac{w_k}{\sum_{k=1}^{K} w_k}$ and $w_k = \frac{p(x,z^k)}{q(z^{(k)}|x}$

In the sleep phase, θ is fixed and we do maximum likelihood learning for q. $\mathcal{L}(\phi, x, z) = logq_\phi(x \mid z)$ and we take the gradient w.r.t ϕ

$$\nabla \mathcal{L}_q(\phi, x) \simeq \sum_{k=1}^{K} \tilde{w_k} \nabla \log q(z^{(k)} \mid x) \;\; where \;\; x, z \sim p(x, z) \tag{19}$$

7 Comparison of models

All the models presented in this paper make certain assumptions that makes calculating the approximate posterior tractable. These assumptions come with their downsides, and the strength and weakness of each method should be understood so that they can be used properly given the right conditions.

In the following we are going to point out the strength and weaknesses of each of the models, and compare their assumptions and results.

VAE - In Kingma and Welling [2014], the approximate posterior $q_\phi(z \mid x)$ is assumed to be approximately factorial which limits the complexity of the model it can learn, with the parameters of the distributions $p_\theta(x \mid z)$ and $q_\phi(z \mid x)$ computed with a feed-forward neural network. Therefore the parameters are subject to learning capacity of the embedded neural network.
The algorithm presented can be easily scaled to large data sets, given its efficient inference scheme, however the model is limited to performing only one sample from the recognition distribution $q_\phi(z \mid z)$, which also limits the flexibility of the model.

IWAE - The assumption about approximate posterior being approximately factorial, puts constraints on the model. But different than original VAE Kingma and Welling [2014], this algorithm uses reweighted sum of the multiple samplings from recognition distribution. IWAE gives tighter lower bound than VAE, which helps model to learn richer representation. The lower bound gets closer to true log-likelihood as we increase the number of samples.
VAE uses the strict criterion, when approximate posterior fails finding good representation of data, VAE heavily penalizes it. Placing a small porition of data in the high posterior region might be still enough to make accurate inference. IWAE lower the standards, by taking multiple samples from recognition network, which gives more flexibility to model and helps it train models whose posterior distribution do not fit VAE assumptions.

Wake-sleep - The fundamental difference in the learning algorithm of wake-sleep compared to the previous models is that the weights of the inference network and generative network are not updated jointly. Thus the main problem of this algorithm is getting an unbiased sample from the posterior distribution over hidden states given the observed data. However, with the simplification made in Hinton et al. [1995] by training the weights of the two networks separately the estimator of the marginal likelihood $p(x)$ becomes biased.
This is an issue that is solved by Bornschein and Bengio [2014a] by performing multiple sampling of recognition and generative networks, which they have shown it empirically lowers the bias of the marginal likelihood estimator.
Note that this algorithm minimizes the reverse KL $D_{KL}(p(z \mid x) \mid\mid q(z \mid x))$, which causes over-estimation of the support of p, and thus mode-averaging, which is undesirable.
Another limitation of the method presented is that the distribution q determined by the recognition weights is a factorial distribution in each hidden layer, since they become conditionally independent given the states of the layer below, thus limiting the complexity of the distribution it can learn.
VAE and IWAE uses lower bound optimization method to update global parameters jointly, while RWS and wake-sleep optimizes one parameter in each phase.
VAE and IWAE works only with continuous latent variables, while RWS and wake-sleep algorithm works with both continuous and discrete latent variables. Another difference between RWS, wake-sleep algorithm and VAE, IWAE algorithm is that one sample from approximate distribution is enough to get an unbiased gradient estimator.

Reweighted wake-sleep - As discussed in previous paragraph, wake-sleep algorithm optimizes biased estimator. Reweighted wake-sleep algorithm uses the similar approach on IWAE by using multiple samples from recognition network, which makes the estimator less biased. As similar to IWAE, the more samples leads less biased estimator. But when we take the log of estimator eq. 16, it creates biased estimator, because it underestimates the true log-likelihood.
Reweighted-sleep algorithm uses more powerful layer model, such as NADE and DARN, while wake-sleep algorithm uses simple Sigmoid Belief Network.
VAE and IWAE uses lower bound optimization method to update global parameters jointly, while RWS and wake-sleep optimizes one parameter in each phase.
VAE and IWAE works only with continuous latent variables, while RWS and wake-sleep algorithm works with both continuous and discrete latent variables. Another difference between RWS, wake-sleep algorithm and VAE, IWAE algorithm is that one sample from approximate distribution is enough to get an unbiased gradient estimator.

Experiment results - The authors of the papers have kindly open-sourced the implementation of their respective algorithms, and we were able to run experiments on the more complex Toronto Face data set Susskind et al. [2010] than the ones used in the papers, which would enable a direct comparison between the models. This is important, since the only data set that all the papers have in common is the MNIST data set, which still is not ideal for direct comparisons given they might have

Table 1: Log-likelihood of experiments

Model	Test NLL	Train NLL
VAE (2 layers)	512.26	511.20
IWAE (2 layers, $k = 50$)	494.06	490.18
Wake-sleep (SBM, 2 layers)	490.28	491.12
Reweighted wake-sleep	-	-

used different subsets of the data set and used different binarization schemes, which would affect the results of negative log-likelihood (NLL).
Our results for NLL are presented in table (1). Unfortunately due to time constraints we were unable to get results from the reweighted wake-sleep algorithm.
The recognition parameters, generative parameters and samples for each model can be found in Appendix C.

8 Related works

Making an inference over intractable posterior by approximating it has been very popular topic in machine learning. There are many different approaches to this particular problem. Paisley et al. [2012] is using similar gradient estimator as Kingma and Welling [2014]. They have introduced control variates to reduce the high variance of the stochastic gradient estimator. Another similar approach has been proposed in Salimans et al. [2013]. They are also trying to minimize the KL divergence of approximate distribution to the intractable true posterior and using similar reparametrization trick to learn parameters of exponential family. Rezende et al. [2014] is another algorithm that combines the Variational Bayesian inference and neural networks. They are also trying to update the global parameters of recognition and generative models jointly by optimizing lower bound.
Gregor et al. [2013] has used similar approach, but they have equipped the stochastic hidden layers with auto-regressive connections. They are optimizing approximate parameters based on minimum description length method, which is the same as maximizing lower bound.Makhzani et al. [2015] is combining the Generative Adversarial Networks with Variational Auto-encoders, that turns auto-encoder generative model. They are using similar training procedure as discussed in Section 3.2. They have a discriminative model as which tries to distinguish real data and fake data which generated by auto-encoder's hidden representation.

9 Conclusion

In this paper, we presented four different unsupervised learning algorithms for graphical models with multiple hidden layers and showed the unifying theme of trying to find efficient approximation schemes of the estimated lower bound, while trying to make the bound even tighter.
We discussed the assumptions made by each of the algorithms, and how that affects their performance. We also showed how these papers have borrowed ideas from each other, and built on top of the previous work done in this field.

Acknowledgments

We thank professor David Duvenaud, and David Madras who gave scientific guidance and their time.

References

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

Jorg Bornschein and Yoshua Bengio. Reweighted wake-sleep. *arXiv preprint arXiv:1406.2751*, 2014a.

Jorg Bornschein and Yoshua Bengio. *reweighted-ws*. 2014b. `https://github.com/jbornschein/reweighted-ws`.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*, 2015a.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. *iwae*. 2015b. `https://github.com/yburda/iwae`.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Karol Gregor, Ivo Danihelka, Andriy Mnih, Charles Blundell, and Daan Wierstra. Deep autoregressive networks. *arXiv preprint arXiv:1310.8499*, 2013.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. Thewake-sleepälgorithm for unsupervised neural networks. *Science*, 268(5214):1158, 1995.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2014.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. In *AISTATS*, volume 1, page 2, 2011.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

J. Paisley, D. Blei, and M. Jordan. Variational Bayesian Inference with Stochastic Search. *ArXiv e-prints*, June 2012.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Tim Salimans, David A Knowles, et al. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

J. Susskind, A. Anderson, and Hinton. G.E.: The Toronto face dataset. Technical report, U. Toronto, 2010.

Appendix A

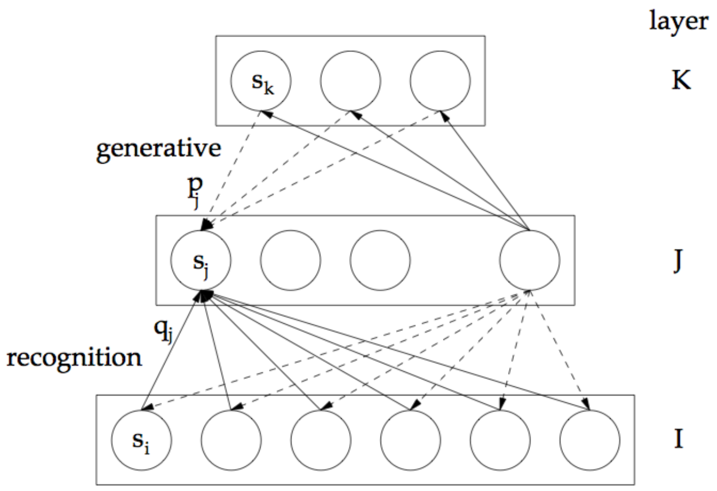

Figure 2: Helmholtz machine architecture. Layer I in this figure represents the input during recognition phase, and the From Hinton et al. [1995]

Appendix B

Marginal likelihood derivation

$p(z \mid x)$ is our true posterior and $q(z \mid x)$ is our approximate posterior.

$$\begin{aligned}
D_{KL}(q(z \mid x) \mid\mid p(z \mid x)) &= \mathbb{E}_{q(z|x)}\left[\log \frac{q(z \mid x)}{p(z \mid x)}\right] \\
&= \mathbb{E}_{q(z|x)}[\log(q(z \mid x)] - \mathbb{E}_{q(z|x)}[(z \mid x)] \\
&= \mathbb{E}_{q(z|x)}[\log(q(z \mid x)] - \mathbb{E}_{q(z|x)}\left[\log \frac{p(z,x)}{p(x)}\right] \\
&= \mathbb{E}_{q(z|x)}[\log q(z \mid x)] - \mathbb{E}_{q(z|x)}[\log p(z,x)] + \log p(x)
\end{aligned}$$

$$\log p(x) = D_{KL}(q(z \mid x) \mid\mid p(z \mid x)) + \mathbb{E}_{q(z|x)}[-\log q(z \mid x) + \log p(z,x)]$$

Appendix C

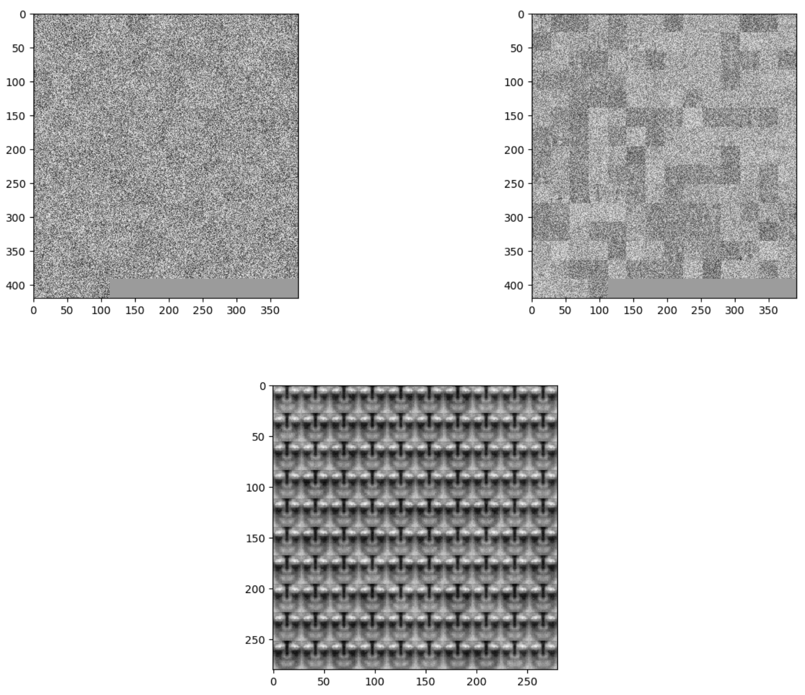

Figure 3: Weights of p (top left), weights of q (top right), samples (bottom) from VAE

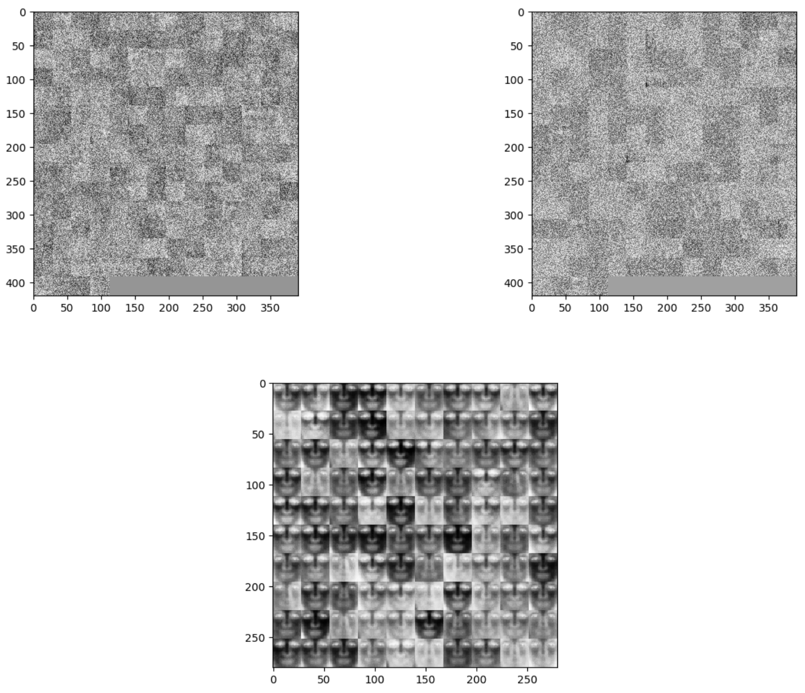

Figure 4: Weights of p (top left), weights of q (top right), samples (bottom) from IWAE

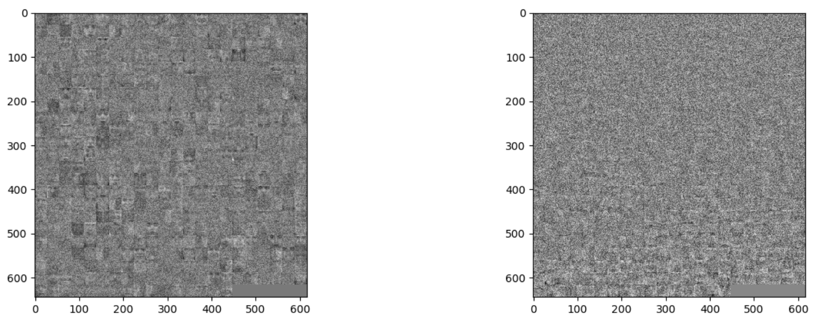

Figure 5: Generative weights (left), recognition weights (top right) from wake-sleep algorithm

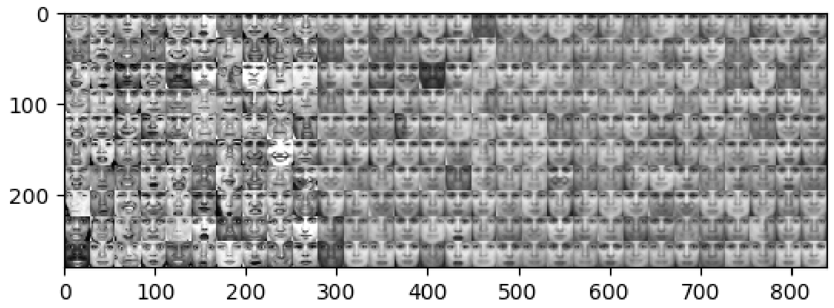

Figure 6: Left 1/3 are original images, middle 1/3 are reconstructed images, right 1/3 are samples from wake-sleep algorithm