

Variational Noise-contrastive Estimation

Ben Rhodes

May 29, 2018

1 Abstract

This thesis presents a method for estimating the parameters of unnormalised probability densities with unobserved (or ‘latent’) variables. The method is an extension of noise-contrastive estimation, which applies only to unnormalised, fully observed models.

2 Introduction

An unnormalised model, $\phi(\mathbf{u}; \boldsymbol{\theta})$, is a parametrised family of non-negative functions that you can think of as scaled probability density ¹ functions that do not integrate to 1 for all values of $\boldsymbol{\theta}$. That is:

$$\int_{\mathbf{u}} \phi(\mathbf{u}; \boldsymbol{\theta}) \, \mathrm{d}\mathbf{u} = Z(\boldsymbol{\theta}) \neq 1 \qquad \phi(\mathbf{u}; \boldsymbol{\theta}) \geq 0 \qquad (1)$$

where $Z(\boldsymbol{\theta})$ is called the *partition function*. The partition function is important because it allows us to normalise $\phi(\mathbf{u}; \boldsymbol{\theta})$, obtaining a model $p(\mathbf{u}; \boldsymbol{\theta})$ that, regardless of the value of $\boldsymbol{\theta}$, integrates to 1.

$$p(\mathbf{u}; \boldsymbol{\theta}) = \frac{\phi(\mathbf{u}; \boldsymbol{\theta})}{Z(\boldsymbol{\theta})}, \qquad \int_{\mathbf{u}} p(\mathbf{u}; \boldsymbol{\theta}) \, \mathrm{d}\mathbf{u} = 1. \qquad (2)$$

Unfortunately, the partition function is rarely computable analytically, and the cost of approximating it generally scales exponentially with the dimension of \mathbf{u} . The intractability of the partition function becomes problematic when we want to estimate good values of $\boldsymbol{\theta}$ given some data $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, where ‘good’ — in practical terms — means that $p(\mathbf{x}_i; \boldsymbol{\theta})$ is large for all i (whilst still satisfying the constraint of integrating to 1). It is problematic because the standard approach of maximum likelihood estimation requires us to maximise an objective that depends on the partition function; namely, the log-likelihood:

$$\ell_n(\boldsymbol{\theta}) = \sum_{i=0}^n \log p(\mathbf{x}_i; \boldsymbol{\theta}) = \sum_{i=0}^n \log \phi(\mathbf{x}_i; \boldsymbol{\theta}) - n \log(Z(\boldsymbol{\theta})), \qquad (3)$$

¹unless stated otherwise, assume that all results apply equally to probability mass functions defined over discrete random variables, replacing integrals with sums as necessary.

To handle this intractability, multiple methods for estimating the parameters θ without directly computing the partition function, such as pseudolikelihood [?], score matching [?], ratio matching [?], contrastive divergence [?], persistent contrastive divergence [??] and noise-contrastive estimation (NCE) [?]. The final three methods — contrastive divergence, its persistent variant, and NCE — are perhaps the most versatile, being simple to implement and applicable to a large class of models. In some respects, NCE is preferable to contrastive divergence since it has a well-defined objective function (a point we shall return to later) and, under some quite general assumptions, is guaranteed to converge to the true value of θ . However, NCE cannot currently be applied to models with hidden, or *latent*, variables, whereas contrastive divergence can. It is this deficiency of NCE that this thesis aims to remedy.

A **latent variable model** is a family of probability densities $p(\mathbf{u}; \theta)$ for which we do not have a direct expression; instead we only have a joint $p(\mathbf{u}, \mathbf{z}; \theta)$ density, where \mathbf{z} are hidden variables, not present in our data set. We thus obtain $p(\mathbf{u}; \theta)$ only after marginalisation:

$$p(\mathbf{u}; \theta) = \int p(\mathbf{u}, \mathbf{z}; \theta) d\mathbf{z} . \quad (4)$$

Learning the parameters θ is problematic here for essentially the same reason as in unnormalised models: the presence of an intractable integral. In section 3.3 we discuss a general strategy for overcoming this intractability using variational inference.

To see why latent variable models are important, it helps to consider two conceptually distinct (but mathematically equivalent) types of latent variables. The first type represent ‘missing data’; perhaps certain pixels in an image were corrupted, generating NaNs. The second type represent compressions of the visible data, which often (but not always) capture low-dimensional causes of the visible data. Such compressions are at the core of unsupervised learning, since the ability to build parsimonious models of the hidden causes of your observations appears to be a fundamental aspect of intelligence. Thus, it is primarily the second view of latent variables that motivates the widespread use of such models.

Let us now consider the combined case of an unnormalised, latent variable model. To write down an expression for $p(\mathbf{u}; \theta)$, we combine equations 2 and 4 to get:

$$p(\mathbf{u}; \theta) = \frac{\int \phi(\mathbf{u}, \mathbf{z}; \theta) d\mathbf{z}}{Z(\theta)} , \quad (5)$$

where $Z(\theta)$ is as we defined it in equation 1. This expression contains two troublesome integrals, and it is this double intractability that is the fundamental technical barrier we must overcome to perform parameter estimation.

3 Background

3.1 Maximum Likelihood and Contrastive Divergence

As stated in the introduction, maximum likelihood learning consists of maximising the log-likelihood given in equation 3, which is a summation over n data points. However, for much of the theory presented in this thesis, it will be convenient to remove the dependency on a finite set of data, and just refer to the underlying distributions that generate the data. To do this, note that the average log-likelihood is the sample version of:

$$\ell(\theta) = \mathbb{E}_{\mathbf{x} \sim p^*} [\log p(\mathbf{x}; \theta)] \quad (6)$$

$$= \mathbb{E}_{\mathbf{x} \sim p^*} [\log \phi(\mathbf{x}; \theta)] - \log Z(\theta) \quad (7)$$

$$= \mathbb{E}_{\mathbf{x} \sim p^*} \left[\log \int \phi(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{z} \right] - \log \int \phi(\mathbf{x}, \mathbf{z}; \theta) d\mathbf{x} d\mathbf{z} , \quad (8)$$

where p^* is the data generating distribution. One can show that the gradient of $\ell(\theta)$ is given by:

$$\nabla_{\theta} \ell(\theta) = \mathbb{E}_{\mathbf{x} \sim p^*} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})} [\nabla_{\theta} \log \phi(\mathbf{x}, \mathbf{z}; \theta)] - \mathbb{E}_{\mathbf{x}, \mathbf{z} \sim p(\mathbf{x}, \mathbf{z})} [\nabla_{\theta} \log (\phi(\mathbf{x}, \mathbf{z}; \theta))] , \quad (9)$$

Intuitively, the first gradient ‘pushes up’ on ϕ wherever the *data* is likely, whilst the second term ‘pushes down’ on ϕ wherever the *model* is likely. The net effect is to reallocate probability mass from regions where the data is unlikely to regions where it is likely. It is critical that reallocate mass, and not simply increase the total amount, since the model must integrate to one. At equilibrium, these two forces directly cancel, and the model equals the data generating distribution.

In general, the expectations in equation 9 will not be computable analytically. Then the question becomes, can we efficiently sample from the model’s posterior over latents $p(\mathbf{z}|\mathbf{x})$ and the from the joint $p(\mathbf{x}, \mathbf{z})$? Sampling must be efficient since we need to re-sample after *every* gradient update. For many models of interest, such as the Restricted Boltzmann Machine discussed in 5.3, sampling from $p(\mathbf{z}|\mathbf{x})$ is straightforward, whilst sampling from the joint $p(\mathbf{x}, \mathbf{z})$ is problematic. The reason is that many unnormalised, latent variable models are defined via undirected graphs, which do not permit ancestral sampling and generally require us to use a Monte-Carlo Markov Chain (MCMC) method that can be very slow to converge.

Contrastive divergence [?] bypasses slow MCMC sampling by initialising the Markov chain at a data point and only running it for k -steps (rather than waiting for it to converge), where k is a tuning hyperparameter. Thus, the gradient updates are given by:

$$\nabla_{\theta} \ell(\theta) = \mathbb{E}_{\mathbf{x} \sim p^*} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})} [\nabla_{\theta} \log \phi(\mathbf{x}, \mathbf{z}; \theta)] - \mathbb{E}_{\mathbf{x}, \mathbf{z} \sim p_k(\mathbf{x}, \mathbf{z})} [\nabla_{\theta} \log (\phi(\mathbf{x}, \mathbf{z}; \theta))] , \quad (10)$$

where p_k is the distribution obtained after running the Markov chain for k steps, starting at a datapoint. ? showed that the difference between the MLE gradient update and the

contrastive divergence update is given by

$$\mathbb{E}_{\mathbf{x} \sim p_k(\mathbf{x})} [\nabla_{\boldsymbol{\theta}} \log p_k(\mathbf{x})] . \quad (11)$$

In addition, they showed that this term tends to zero as k tends to infinity and in the specific case of the RBM, provided a bound on this term. Non model-specific bounds seem hard to derive, and little is known in general about the convergence properties of contrastive divergence [?]. One interesting result we do have is that the contrastive divergence update is *not* the update of any objective function [?], which prevents us applying non-linear optimisation methods. Despite the weak theory supporting contrastive divergence, it has proven empirically to be an efficient learning technique in many situations.

A related learning method to contrastive divergence is *persistent* contrastive divergence [??]. Instead of initialising the Markov chains at a datapoint after every gradient update, we ‘persist’ the state of the chain and reuse it across gradient steps.

3.2 Noise-contrastive estimation

NCE converts an unsupervised density estimation problem into a supervised classification problem, by training a (non-linear) logistic classifier to distinguish between the true data and samples from some reference distribution, which we refer to as noise samples.

Concretely, we first generate m samples $\{\mathbf{y}_1, \dots, \mathbf{y}_m\}$ from a noise distribution $p_{\mathbf{y}}(\mathbf{y})$ and mix these samples with our data $(\mathbf{x}_1, \dots, \mathbf{x}_n)$. Our goal is now to classify whether a random sample \mathbf{u} from this mixture came from the data-generating distribution p or the noise $p_{\mathbf{y}}$. We do so by calculating posterior probabilities, using our unnormalised model $\phi(\mathbf{u}; \boldsymbol{\theta})$ as a surrogate for the true probability of the data $p(\mathbf{u})$. If $\nu = \frac{m}{n}$, then:

$$\mathbb{P}(\mathbf{u} \sim p_{\mathbf{x}}; \boldsymbol{\theta}) = \frac{\phi(\mathbf{u}; \boldsymbol{\theta})}{\phi(\mathbf{u}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{u})} \quad (12)$$

$$\mathbb{P}(\mathbf{u} \sim p_{\mathbf{y}}; \boldsymbol{\theta}) = \frac{\nu p_{\mathbf{y}}(\mathbf{u})}{\phi(\mathbf{u}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{u})} . \quad (13)$$

Note that $\mathbb{P}(\mathbf{u} \sim p_{\mathbf{y}}; \boldsymbol{\theta}) = 1 - \mathbb{P}(\mathbf{u} \sim p_{\mathbf{x}}; \boldsymbol{\theta})$, which makes sense because this is a two-class classification problem, and so we have a Bernoulli distribution over the class label. Using the average log-likelihood for a Bernoulli, we derive the following objective function:

$$J_n(\boldsymbol{\theta}) = \frac{1}{n} \left\{ \sum_{i=1}^n \log \mathbb{P}(\mathbf{x}_i \sim p_{\mathbf{x}}; \boldsymbol{\theta}) + \sum_{i=1}^m \log [\mathbb{P}(\mathbf{y}_i \sim p_{\mathbf{y}}; \boldsymbol{\theta})] \right\} \quad (14)$$

$$= \frac{1}{n} \sum_{i=1}^n \log \left(\frac{\phi(\mathbf{x}_i; \boldsymbol{\theta})}{\phi(\mathbf{x}_i; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{x}_i)} \right) + \sum_{i=1}^m \log \left(\frac{\nu p_{\mathbf{y}}(\mathbf{y}_i)}{\phi(\mathbf{y}_i; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{y}_i)} \right) , \quad (15)$$

which is the sample version of $J(\boldsymbol{\theta})$:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}} \log \left(\frac{\phi(\mathbf{x}; \boldsymbol{\theta})}{\phi(\mathbf{x}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{x})} \right) + \nu \mathbb{E}_{\mathbf{y}} \log \left(\frac{\nu p_{\mathbf{y}}(\mathbf{y})}{\phi(\mathbf{y}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{y})} \right) . \quad (16)$$

If we denote the true data generating distribution by $p(\mathbf{u}; \theta^*)$, and assume that this distribution is contained within our unnormalised family $\phi(\mathbf{u}; \boldsymbol{\theta})$, then, under some rather general technical conditions described by ?, optimising $J(\boldsymbol{\theta})$ guarantees that $\boldsymbol{\theta}$ converges (in probability) to θ^* in the limit of infinite data.

Why should the *normalised* data generating distribution live in an *unnormalised* family? This assumption may seem odd at first, but note that it is simple to give the model family $\phi(\mathbf{u}; \boldsymbol{\theta})$ an extra degree of freedom by multiplying it with a scaling parameter c . This new parameter is estimated in conjunction with $\boldsymbol{\theta}$, so that we automatically obtain a normalised solution.²

It turns out that for a large number of noise samples, ν , the choice of noise distribution q is not important. However, increasing ν uses more computational resource, so we still need to select an appropriate distribution. It remains an area of active research to select a noise distribution $p_{\mathbf{y}}$ based on theoretically justified properties, with the main constraints currently being that it should be cheap to sample from, non-zero wherever ϕ is non-zero, and similar to the underlying the data generating distribution.

In the case of latent variable models, the extended version of NCE that we present in this thesis is able to partially automate the construction of a suitable noise distribution, which is a significant advance over hand-crafted choices.

²note, this is *not* the same as normalising the model i.e. computing the partition function, which provides the normalisation constant for *every* value of $\boldsymbol{\theta}$

4 Variational noise-contrastive estimation

Guttmann [2018, unpublished] derived a *variational* lower bound to the NCE objective function, which is the mathematical keystone to this thesis. Before presenting the bound, we first give a more standard introduction to the variational approach, which should make the subsequent material clearer.

4.1 Variational inference and EM

As discussed in section 3.1, the expected average log-likelihood for a latent variable model is given by:

$$\ell(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}} \log \left(\int p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) d\mathbf{z} \right) \quad (17)$$

$$= \mathbb{E}_{\mathbf{x}} \log \left(\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left(\frac{p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x})} \right) \right) \quad (18)$$

where, in the second line, we applied a simple identity known as importance sampling, where the ‘auxiliary’ density q can be any valid density that is non-zero whenever p is non-zero. We then apply Jensen’s inequality, using the fact that the log function is concave, giving:

$$\ell(\boldsymbol{\theta}) \geq \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left(\log \left(\frac{p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x})} \right) \right). \quad (19)$$

It is straightforward to show that this inequality is tight when $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$. This inequality directly leads to an iterative algorithm, known as Expectation Maximisation (EM) for estimating the parameters $\boldsymbol{\theta}$. Whilst the idea of EM is quite old, originally by ?, the view presented here is similar to that given by ?. The algorithm simply involves alternating between optimising the right-hand side of equation 17 with respect to $\boldsymbol{\theta}$ (which we can do since we have an expression for $p(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})$) and then, with our new $\boldsymbol{\theta}$, setting: $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$. One can prove that this iterative algorithm always increases the log-likelihood.

EM, as stated above, assumes we have an expression for $p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$. That is, we assume *inference* of the latent variables is tractable. What if this is not the case? Well, we still want q to approximate the posterior over latent variables, and so we could parametrise it with parameters $\boldsymbol{\alpha}$. If this parametric family is sufficiently broad, we might hope that one of its members is close to the true posterior, and that we can find such a member by optimising $\boldsymbol{\alpha}$. Thus, instead of resetting: $q(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta})$, we optimise the right-hand side of 17 with respect to the *variational parameters* $\boldsymbol{\alpha}$.

How to perform this optimisation with respect to $\boldsymbol{\alpha}$ is a matter of ongoing research. It is non-trivial because the right-hand side of 17 involves an expectation with respect to $q(\mathbf{z}|\mathbf{x}; \boldsymbol{\alpha})$, so standard gradient-based methods are hard to straightforwardly apply. There are methods to cope with this obstacle, such as the reparameterisation trick [?], which we discuss in later chapters.

4.2 Variational lower bound for NCE

Inspired by the variational lower bound for the log-likelihood, it's natural to wonder how we might apply a similar strategy to the NCE objective function in equation 14. There are two fundamental steps:

- Use importance sampling to re-express the intractable integral over \mathbf{z} as an expectation; call this E .
- Apply Jensen's inequality to a concave function $g(E)$.

One complication in applying such a strategy is that the intractable integral $\int \phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta}) d\mathbf{z}$ appears twice: inside the expectation with respect to the data:

$$\mathbb{E}_{\mathbf{x}} \log \left(\frac{\phi(\mathbf{x}; \boldsymbol{\theta})}{\phi(\mathbf{x}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{x})} \right) \quad (20)$$

and it also appears in the second term, which is an expectation with respect to the noise distribution:

$$\nu \mathbb{E}_{\mathbf{y}} \log \left(\frac{\nu p_{\mathbf{y}}(\mathbf{y})}{\phi(\mathbf{y}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{y})} \right) . \quad (21)$$

Let us focus only on the first term (18). Introducing the notation:

$$r(\mathbf{x}; \boldsymbol{\theta}) = \frac{\int \phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) d\mathbf{z}}{p_{\mathbf{y}}(\mathbf{x})} = \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \left(\frac{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})} \right) \quad (22)$$

and

$$g(r) = -\log \left(1 + \nu \frac{1}{r} \right), \quad (23)$$

we see that:

$$\log \left(\frac{\phi(\mathbf{x}; \boldsymbol{\theta})}{\phi(\mathbf{x}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{x})} \right) = g(r(\mathbf{x}; \boldsymbol{\theta})) . \quad (24)$$

It can be shown that g is a concave function of r . Since r is really just an expectation, as is clear from 20, we can apply Jensen's inequality:

$$\log \left(\frac{\phi(\mathbf{x}; \boldsymbol{\theta})}{\phi(\mathbf{x}; \boldsymbol{\theta}) + \nu p_{\mathbf{y}}(\mathbf{x})} \right) \geq \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} g \left(\frac{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})} \right) \quad (25)$$

$$\geq -\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log \left(1 + \nu \frac{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})} \right) , \quad (26)$$

where, to get the last line, we simply substitute in the definition of g .

It does not appear possible to apply the same combination of importance sampling and Jensen's inequality to the second term of the NCE objective (19). However, we don't

necessarily need to. The intractable integral in the second term can be approximated just with importance sampling, *re-using* the variational distribution q that we got from the first term. Thus, we obtain a variational lower bound to the NCE objective function given by:

$$\mathcal{J}_1(\boldsymbol{\theta}, q) = -\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log \left(1 + \nu \frac{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})} \right) - \nu \mathbb{E}_{\mathbf{y}} \log \left(1 + \frac{1}{\nu} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{y})} \left[\frac{\phi(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{y}) p_{\mathbf{y}}(\mathbf{y})} \right] \right). \quad (27)$$

We have that $J(\boldsymbol{\theta}) \geq \mathcal{J}_1(\boldsymbol{\theta}, q)$ for all q , and so we can apply variational EM, maximising $J(\boldsymbol{\theta})$ by iterating between maximising $\mathcal{J}_1(\boldsymbol{\theta}, q)$ with respect to $\boldsymbol{\theta}$, and the tightening of the bound by maximising $\mathcal{J}_1(\boldsymbol{\theta}, q)$ with respect to the variational distribution q .

4.3 Necessary ingredients to apply VNCE

Having established the objective function 25 for VNCE, it is helpful to explicitly lay out the ingredients required for optimising it. We require:

- A noise distribution over the visible variables $p_{\mathbf{y}}$. This density should, ideally, be cheap to evaluate and sample from, and similar to the data distribution.
- A variational distribution over latent variables q . This will typically have its own parameters $\boldsymbol{\alpha}$, apart from the case where we set q to equal the model's posterior over latents. This is only possible for models with tractable posteriors.
- If the variational distribution q has parameters $\boldsymbol{\alpha}$, then we need compute $\nabla_{\boldsymbol{\alpha}} J_1(\boldsymbol{\alpha})$. This is difficult in general, but depending on the parameterisation of q , it may be possible to do analytically, perhaps using the reparameterisation trick [?].
- The gradient $\nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta})$. A derivation of this gradient is given in the appendix, section 7.2. The key term required to compute it is the gradient of the logarithm of the unnormalised model: $\nabla_{\boldsymbol{\theta}}(\log \phi(\mathbf{u}; \boldsymbol{\theta}))$.

5 Research Objectives

In the previous section, we established a method of parameter estimation for unnormalised, latent variable models involving the maximisation of a variational lower bound to the NCE objective function, as given in equation 25. We will refer to this method as VNCE. The over-arching goal in the remainder of the thesis is to appraise the usefulness of VNCE across different models, data sets, and evaluation metrics. In addition, we aim to contribute a better theoretical understanding of the new estimation method, which may involve studying its asymptotic properties or its mathematical similarities with other methods such as contrastive divergence.

5.1 Models and data sets

The first steps, which are ongoing, involve applying the estimator to models of different levels of complexity with synthetic data. To this end, we have investigated a simple unnormalised mixture of two gaussians, discussed in section 5.2, and a more sophisticated model: the Restricted Boltzmann Machine (RBM), discussed in section 5.3. The RBM has received a significant amount of attention over the last 15 years in the machine learning community, and there is thus a wealth of published material on estimating its parameters. A natural first step is to therefore build on this work.

After experimenting on synthetic data, we will then apply an RBM to real data. Some of the most thorough empirical work to date on different methods for learning RBMs [?] considered a range of data sets such as MNIST³, the small 20-Newsgroups data set⁴ and an image dataset from CalTech101⁵. It makes sense to evaluate the new estimation method on the same data sets, to enable straightforward comparisons of my results with the published literature. That said, one important finding by ? was that comparisons of estimation methods does vary significantly depending on the data set, and so I will consider experimenting with an even greater variety of data later in the thesis, assuming the preliminary results warrant further investigation.

5.2 Evaluation metrics

There are many possible metrics to consider. Of particular interest are the quality of the probabilistic model (*does it assign high probability to the test data?*), the asymptotic properties of the estimator (*does it use data efficiently?*) and the computational cost of the estimator (*does it converge quickly?*).

With regards to the quality of the probabilistic model, we want to know: is it close to the true data generating distribution? One way of measuring closeness between distributions is the KL divergence. We can estimate the KL divergence between the true distribution and our model by evaluating the model's log-likelihood on the test data. For relatively simple, low-dimensional models, where we can analytically solve for the

³<http://yann.lecun.com/exdb/mnist>

⁴<http://www.cs.toronto.edu/roweis/data.html>

⁵http://www.vision.caltech.edu/Image_Datasets

partition function and marginal distribution over observed variables, we can indeed use the log-likelihood as an evaluation metric. For more complex models, we must resort to approximations of the log-likelihood.

With regards to asymptotic properties, it helps to consider the case of ordinary NCE. There it was proven that the estimator converges (in distribution) to a normal distribution centred on the true parameter value, in the limit of infinite data. It is natural to wonder if a similar result might hold in our case. That is, do we have:

$$\sqrt{n}(\hat{\theta}_n - \theta) \rightarrow \mathcal{N}(0, V) \quad (28)$$

In the 1-dimensional case, V/n is a scalar called the asymptotic variance. We would like this value to be small, as this implies that the estimator uses data efficiently when approximating the true parameter value.

Another means of evaluation is through comparison with the ordinary NCE objective function, which is an upper bound to our new objective function. This is only possible for models where we can analytically integrate out the latent variables, but this restriction isn't as great as it first seems, because RBMs with only a small number of hidden units - in the single digits - can still be useful generative models of real data.

A weaker, but still interesting measure of the usefulness of a density model is to apply it to discriminative tasks. For instance, having trained an RBM to capture the statistics of patches of natural images, you can then use that generative model as part of image-based classifier. This form of measurement is quite indirect, but still potentially useful.

Finally, for all the above metrics, we can ask: what is the computational cost of achieving a particular score under that metric? This is an important practical consideration, and given that NCE is relatively fast compared to other estimators of unnormalised models, we might hope the same is true of VNCE.

5.3 Baselines

Now that we have a sense of the models, data sets and evaluation metrics available, we have one final important task: specifying a baseline estimation method. As mentioned, for simple models, we can actually evaluate the intractable integrals that normally prevent maximum likelihood and NCE from being applied, and so we can compare the new method to these. For more complex methods, such as RBMs with many hidden units, the dominant approach to learning is to use contrastive divergence (or some variant, such as persistent contrastive divergence). Hence, the primary research questions underlying this thesis are:

How does VNCE compare to contrastive divergence under the metrics specified above? Is one method more theoretically grounded than the other, and in what sense?

5.4 Additional Research objectives

You can think of the research objectives above as a road-map for performing a thorough empirical and theoretical comparison of VNCE to others, notably contrastive divergence.

But the new method is actually more general than contrastive divergence, in two senses.

Firstly, we are able not only to estimate the parameters of a model, but also an approximation of the associated normalising constant ⁶. Previously, this had to be approximated separately with a method such as annealed importance sampling [?], and so it would be interesting to compare these approximations.

Secondly, and more significantly, contrastive divergence requires you to be able to sample from the model’s posterior over latent variables. VNCE does not require this; it only requires an approximation of the posterior, which is learned automatically. Hence, there are interesting models, such as an RBM with connections between the hidden variables, or stacked RBMs, where contrastive divergence is much harder to apply than ours. This potentially opens a new class of models for which learning was previously intractable.

⁶Note, we only learn the normalising constant for the final parameters learned after convergence, not intermediate values.

6 Results

In this section, we present a mixture of theoretical and empirical results, including:

- A proof that the EM algorithm for VNCE never decreases the NCE objective function.
- Application of VNCE to toy Mixture of Gaussians models.
- Application of VNCE to a Restricted Boltzmann Machine.
- An adaptive variant of VNCE.

The first result is an analog of the theorem for ordinary EM, described in section 3.3, which states that it never decreases the log-likelihood. By deriving this analog result, we shall see that the optimal choice of q at each E step is given by the model's posterior over latents $p(\mathbf{z}|\mathbf{u})$.

Theorem 1. *Given the VNCE objective function:*

$$\mathcal{J}_1(\boldsymbol{\theta}, q) = -\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log \left(1 + \nu \frac{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})} \right) - \nu \mathbb{E}_{\mathbf{y}} \log \left(1 + \frac{1}{\nu} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{y})} \left[\frac{\phi(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{y}) p_{\mathbf{y}}(\mathbf{y})} \right] \right), \quad (29)$$

a random initial starting point $\boldsymbol{\theta}_0$, and the following optimisation procedure:

1. (E-step) $q_k(\mathbf{z}|\mathbf{u}) = p(\mathbf{z}|\mathbf{u}; \boldsymbol{\theta}_k)$
2. (M-step) $\boldsymbol{\theta}_{k+1} = \arg \max_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta}, q_k)$
3. Unless converged, repeat steps 1 and 2

Then it follows that, for all $k \in \mathbb{N}$, we have: $J(\boldsymbol{\theta}_{k+1}) \geq J(\boldsymbol{\theta}_k)$, where J is the NCE objective function.

Proof. Let $k \in \mathbb{N}$. In section 3.3, we showed that the NCE objective is lower bounded by the VNCE objective, so that: $J(\boldsymbol{\theta}_k) \geq J_1(\boldsymbol{\theta}_k, q)$ for all q . We derived this by applying Jensen's inequality to the first term of the NCE objective function. Specifically, we showed that:

$$\log \left(\frac{\phi(\mathbf{x}; \boldsymbol{\theta}_k)}{\phi(\mathbf{x}; \boldsymbol{\theta}_k) + \nu p_{\mathbf{y}}(\mathbf{x})} \right) \geq -\mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log \left(1 + \nu \frac{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}_k)} \right). \quad (30)$$

We now want to show that the above inequality becomes an *equality* after the E-step of optimisation. To see this, first note that we can rewrite ϕ as:

$$\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}_k) = p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}_k) \phi(\mathbf{x}; \boldsymbol{\theta}_k), \quad (31)$$

since if we divide both sides by $Z(\boldsymbol{\theta}_k)$, then the equation just becomes a straightforward factorisation of the joint density. Therefore, during the E-step of optimisation when we set q_k such that $q_k(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}_k)$, it follows that:

$$-\mathbb{E}_{\mathbf{z} \sim q_k(\mathbf{z}|\mathbf{x})} \log \left(1 + \nu \frac{q_k(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}_k)} \right) = -\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{x}; \boldsymbol{\theta}_k)} \log \left(1 + \nu \frac{p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}; \boldsymbol{\theta}_k)} \right) \quad (32)$$

$$= \log \left(\frac{\phi(\mathbf{x}; \boldsymbol{\theta}_k)}{\phi(\mathbf{x}; \boldsymbol{\theta}_k) + \nu p_{\mathbf{y}}(\mathbf{x})} \right), \quad (33)$$

where, to reach the final line, we simply dropped the expectation since the expression inside the expectation was constant with respect to \mathbf{z} , and rearranged the fraction inside the log. This proves that we have equality in equation ??, and hence $J_1(\boldsymbol{\theta}_k, q_k) = J(\boldsymbol{\theta}_k)$.

Now, in the M-step of optimisation, we have

$$\boldsymbol{\theta}_{k+1} = \arg \max_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta}, q_k) \implies J_1(\boldsymbol{\theta}_{k+1}, q_k) \geq J_1(\boldsymbol{\theta}_k, q_k), \quad (34)$$

We therefore obtain:

$$J(\boldsymbol{\theta}_{k+1}) \geq J_1(\boldsymbol{\theta}_{k+1}, q_k) \geq J_1(\boldsymbol{\theta}_k, q_k) = J(\boldsymbol{\theta}_k). \quad (35)$$

□

Observe that, just as in the ordinary EM algorithm, the above result does not hold in general if we only take a ‘partial’ E-step, by making q close, but not exactly equal, to $p_{\mathbf{z}|\mathbf{x}}$. However, we can always take a partial M-step, increasing the value of $J_1(\boldsymbol{\theta}, q_k)$ through a few gradient steps without computing the true argmax.

6.1 Mixture of Gaussian model

As a simple illustration of the correctness of VNCE, we apply it to a toy mixture of two Gaussians model, where the only unknown parameter is the standard deviation of one of the Gaussians. We consider two cases: firstly a normalised MoG, and then an unnormalised version. Finally, for the unnormalised version, we run a population analysis to compare the performance of VNCE against both NCE and maximum likelihood estimation across multiple sample sizes.

6.1.1 The normalised case

A common latent variable model is the Mixture of Gaussian (MoG) model. For a mixture with two components, a single binary latent variable $z \sim \text{Ber}(\pi)$ determines which of two Gaussians the data was generated from. if $\pi = 1/2$, we have the following joint:

$$p(u, z; \theta) = \frac{(1-z)}{2} \mathcal{N}(u; 0, \theta) + \frac{z}{2} \mathcal{N}(u; 0, \sigma_1). \quad (36)$$

We assume that the variance of the second component, σ_1^2 , is known, and our task is to estimate the value of θ . Whilst this estimation problem does not involve an unnormalised

model, we can still apply VNCE to it. This provides a useful sanity check of the method's validity in a simple setting where we do not have a scaling parameter to estimate in conjunction with θ .

For a simple experiment, we set $\sigma_1 = 1$ and let $\theta^* = 4$ be the true value of θ . We then make the following choices of noise and variational distribution:

$$p_{\mathbf{y}}(u) = \mathcal{N}(u; 0, \theta^*) \quad (37)$$

$$q(z|u) = p(z = 0 \mid u; \theta) = \left(1 + \frac{\theta_k}{\sigma_1} \exp \left(\frac{-u^2}{2} \left(\frac{1}{\sigma_1^2} - \frac{1}{\theta^2} \right) \right) \right)^{-1}. \quad (38)$$

The choice of noise distribution is not too important for this simple model, so long as it very approximately matches the data generating distribution. With this setup, we can then apply the EM type algorithm discussed in section 5.1. We can visualise this algorithm by iteratively plotting the variational bound, for all values of θ , each time we update the variational distribution q in figure 2. It is clear from the figure that the algorithm converges to the correct value of θ .

6.1.2 The unnormalised case

We would like to modify the MoG given in equation 27 to obtain an unnormalised family. An obvious candidate is:

$$\phi(u, z; \theta, c) = e^{-c} \left((1 - z) e^{-\frac{u^2}{2\theta^2}} + z e^{-\frac{u^2}{2\sigma_1^2}} \right) \quad (39)$$

where e^{-c} is a scaling parameter needed for NCE, as discussed at the end of section 3.2. It is important to note that the usual MoG family in equation 27 is *not* nested within unnormalised family in equation 30. We can see this more easily by normalising 30, giving us:

$$p(u, z, \theta) = (1 - z) \frac{\theta}{\theta + \sigma_1} \mathcal{N}(u; 0, \theta) + z \frac{\sigma_1}{\theta + \sigma_1} \mathcal{N}(u; 0, \sigma_1) \quad (40)$$

We see that the parameter θ of the unnormalised MoG controls both the width of the first component and its scale relative to the second component.

As before, fix $\sigma_1 = 1$ and $\theta^* = 4$. Using equation 31, we can easily sample synthetic data from our unnormalised model ϕ (still under the assumption that $z \sim \text{Ber}(\frac{1}{2})$). To do so, we simply sample $w \sim \text{Ber}(\frac{\sigma_1}{\theta + \sigma_1})$ and then sample $\{x_1, \dots, x_i\}$ data points:

$$x_i \sim \begin{cases} \mathcal{N}(u; 0, \theta), & \text{if } w = 0 \\ \mathcal{N}(u; 0, \sigma_1), & \text{if } w = 1 \end{cases}$$

Using this synthetic data, we can proceed as before, using an EM algorithm to learn the parameter θ . To do this, we would need to use the posterior over latents, given by:

$$p(z = 0 \mid u) = \frac{1}{1 + \exp(\frac{-u^2}{2} (\frac{1}{\sigma_1^2} + \frac{1}{\sigma_0^2}))}. \quad (41)$$

Figure 1: EM type algorithm applied to a normalised Mixture of Gaussians model. The notation $J_{\theta_k}^1$ in the legends refers to the variational lower bound of the NCE objective function where the variational distribution $q(z|u)$ is set to $p(z|u; \theta_k)$



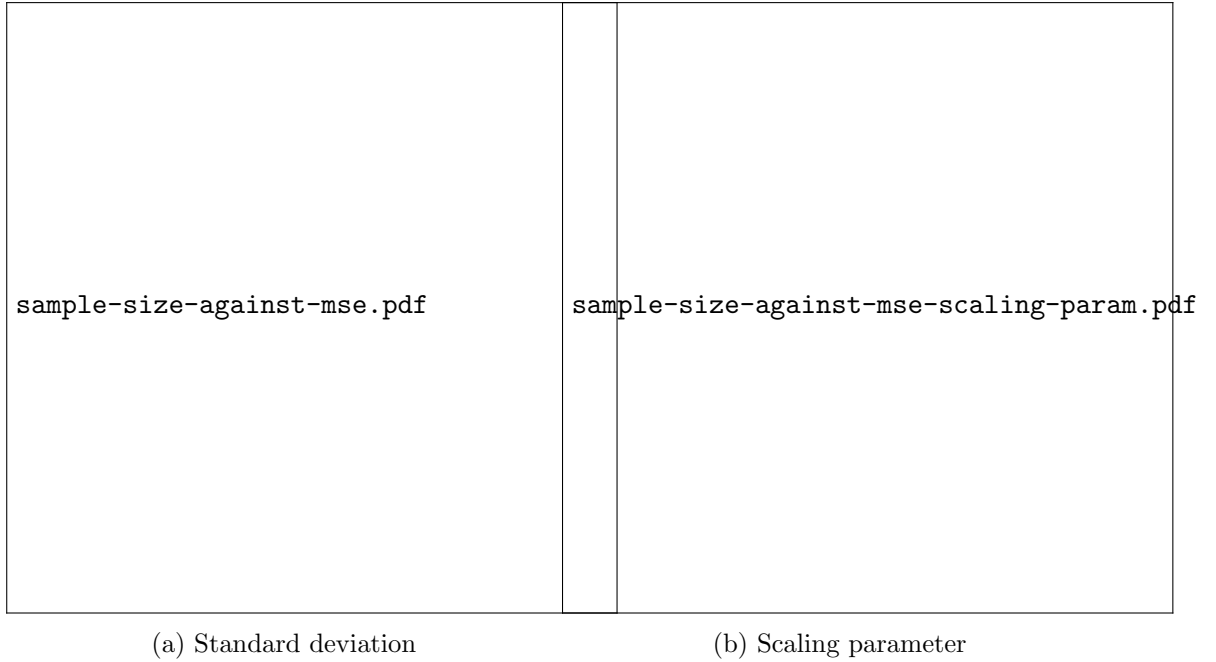


Figure 2: log sample size against log mean-squared error (MSE) produced when estimating the standard deviation and scaling parameter for 500 different unnormalised Mixture of Gaussian models. The thick central lines show the median MSE of the 500 runs, whilst the dashed lines mark the 1st and 9th deciles. The constant negative slope of the red line is evidence of the consistency of VNCE.

However, for more complex models, we do not have access to such a posterior, and so it is important that VNCE still works when we approximate it with a parametrised variational distribution q . To test this, we use the following q :

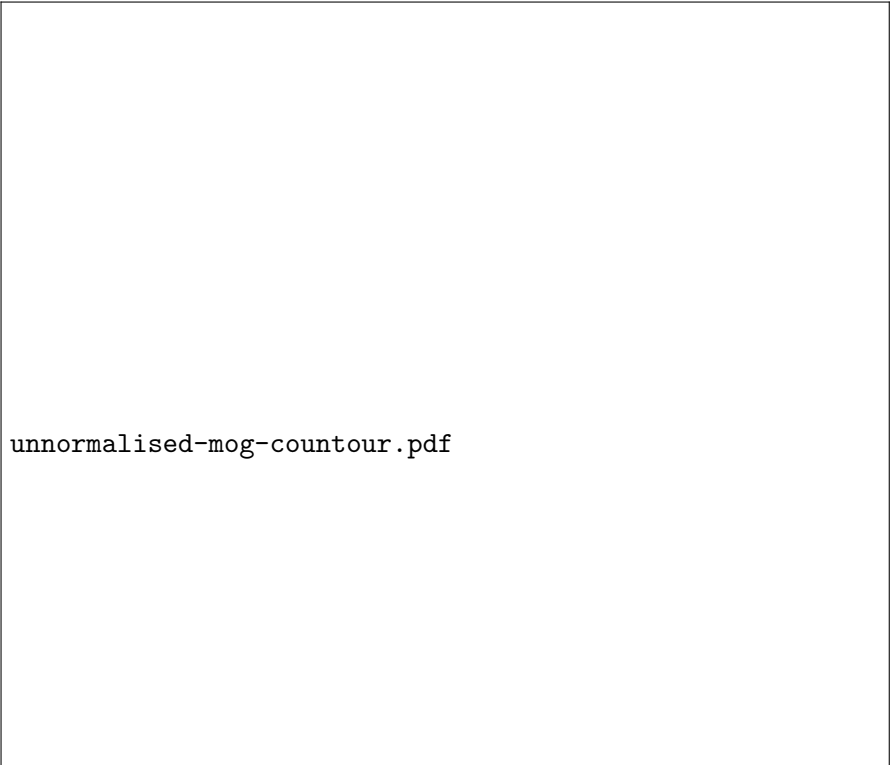
$$q(z = 0 \mid u; \mathbf{w}) = \frac{1}{1 + \exp(w_0 + w_1 u + w_2 u^2)}, \quad (42)$$

which clearly contains the true posterior.

Visualising learning is less straightforward now that our parameter set is 2-dimensional, but we can compare the contour plot of the true NCE objective function against the lower bound obtained at the end of learning. This is given in figure 5. We see that the algorithm converges to the correct value of θ , and correct normalising constant.

6.1.3 Population Analysis of unnormalised Mixture of Gaussian model

Figure 3 shows the MSE $\mathbb{E}||\theta - \theta^*||^2$ for VNCE, NCE and maximum likelihood across multiple runs and with different sample sizes. To produce it, we generated 500 distinct ground-truth values for standard deviation parameter in the unnormalised MoG, sampling uniformly from the interval $[2, 6]$. For each of the 500 θ^* s, we estimate it using all three estimation methods and with a range of sample sizes. Every run was initialised from five random values and the best result out of the five was kept in order to avoid local optima, which exist since both the likelihood and NCE objective functions are bi-modal.



unnormalised-mog-countour.pdf

Figure 3: Contour plot of true NCE objective function compared to the variational lower bound for an unnormalised Mixture of Gaussians. For the lower bound, we have set the variational distribution $q(z|u)$ to be the posterior $p(z|u; \tilde{\theta})$, where $\tilde{\theta}$ is our final estimate obtained after optimisation. The vertical axis corresponds to c , the scaling parameter.

Figure 3 (a) demonstrates that the estimation accuracy of VNCE increases with sample size, and is comparable to that of NCE. This gives some evidence of the consistency of VNCE. We note that the comparison is complicated by the fact that NCE was particularly prone to falling into local optima (despite our use of multiple random initialisations). This is why NCE’s worst-case accuracy is significantly worse than that of the other two methods, as shown by the upper dashed blue line.

6.2 Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) [?] is an unnormalised, latent variable model over binary vectors given by:

$$p(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta}) \propto \exp(\mathbf{u}^T W \mathbf{z} + \mathbf{a}^T \mathbf{u} + \mathbf{b}^T \mathbf{z}) , \quad (43)$$

where $W \in \mathbb{R}^{d \times m}$; $\mathbf{a}, \mathbf{u} \in \mathbb{R}^d$; $\mathbf{b}, \mathbf{z} \in \mathbb{R}^m$, and $\boldsymbol{\theta} \equiv (W, \mathbf{a}, \mathbf{b})$. We refer to W as the weight matrix and \mathbf{a} and \mathbf{b} as the biases. \mathbf{u} contains the visible variables/units and \mathbf{z} the hidden units, which are typically of lower dimension since we would like to find a smaller set of random variables that capture the correlations between the higher-dimensional observations.

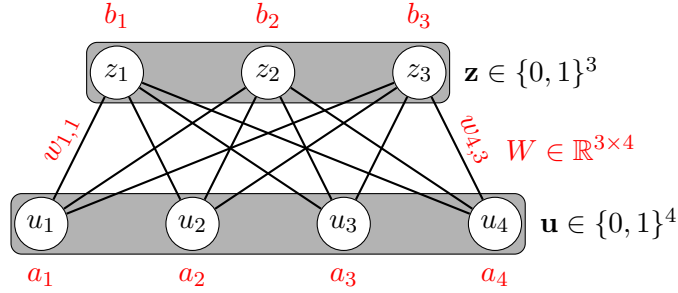


Figure 4: The undirected graph for a Restricted Boltzmann Machine.

In order to see how the hidden units in an RBM explain the couplings between the visible units, it helps to picture an undirected graph over which the model factorizes as in figure 4. Note that the Markov blanket of each u_i is \mathbf{z} , and, conversely, the Markov blanket of each z_i is \mathbf{u} . Therefore, we obtain the factorisations:

$$p(\mathbf{u}|\mathbf{z}) = \prod_{i=1}^d p(u_i|\mathbf{z}) \quad p(\mathbf{z}|\mathbf{u}) = \prod_{i=1}^m p(z_i|\mathbf{u}) ; \quad (44)$$

the first factorisation says that the visible units are independent given the latents and thus the latents capture correlations in the visibles.

We gain further insight into the model’s properties by not only considering it’s conditional independencies, but its specific parametric form, as given in equation 32. If we

marginalise out the latent variables, we obtain:

$$p(\mathbf{u}) = \sum_{z_1, \dots, z_m} p(\mathbf{u}, \mathbf{z}) \quad (45)$$

$$\propto \exp(\mathbf{a}^T \mathbf{u}) \prod_{i=1}^m \sum_{z_i \in \{0,1\}} \exp(b_i z_i + z_i \mathbf{u}^T W_{:,i}) \quad (46)$$

$$\propto \exp(\mathbf{a}^T \mathbf{u}) \prod_{i=1}^m (1 + \exp(b_i + \mathbf{u}^T W_{:,i})) , \quad (47)$$

where $W_{:,i}$ is the i^{th} column of the weight matrix W . Since $p(\mathbf{u})$ is a product over $m+1$ terms, we can think of the RBM as a ‘product of experts’ [?]:

$$p(\mathbf{u}) \propto \prod_{i=1}^{m+1} f_i(\mathbf{u}) \quad (48)$$

where each expert ‘votes’ on the probability of a vector \mathbf{u} . Since the experts act multiplicatively, the resulting distribution can be quite sharp, since each expert can vote for a specific sub-region of the input space. In the RBM, each expert takes the inner product of a fixed ‘template’ vector — $W_{:,i}$ or \mathbf{a} — with the input \mathbf{u} , and so inputs that match multiple templates are likely under the model.

6.2.1 Training Restricted Boltzmann Machines

The factorisation of $p(\mathbf{u})$ in equation 36 shows that marginalising out the latents is cheap for an RBM, costing only $O(m)$ time, where m is the dimension of the latent vector \mathbf{z} . However, computing the partition function *is* intractable, since we would have to compute:

$$Z(\boldsymbol{\theta}) = \sum_{\mathbf{u} \in \{0,1\}^d} \exp(\mathbf{a}^T \mathbf{u}) \prod_{i=1}^m (1 + \exp(b_i + \mathbf{u}^T W_{:,i})) , \quad (49)$$

which cannot be further factorised due to the presence of \mathbf{u} in every term of the product. Naively, computing $Z(\boldsymbol{\theta})$ costs $O(m2^d)$. We can rearrange the sums to reduce this to $O(d2^m)$, but this is still exponential in the number of hidden units. Further optimisations exist to reduce the cost, but it is still infeasible to compute for much more than around 30 hidden units.

Hence, we are not dealing with the general case that VNCE was designed for: when *both* marginalising out the hiddens and computing the partition function are intractable. Since only the latter is intractable, we can apply NCE to the problem. As with the mixture of Gaussians, this is useful since we can compare the performance of NCE to VNCE.

Another important comparison to make is with contrastive divergence since it is perhaps the most common method for training an RBM. As discussed in section 3.1,

contrastive divergence roughly approximates the gradient ascent rule for maximum likelihood learning, using updates of the form:

$$\nabla_W \ell(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x} \sim p^*} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})} [\nabla_W \log \phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})] - \mathbb{E}_{\mathbf{x}, \mathbf{z} \sim p_k(\mathbf{x}, \mathbf{z})} [\nabla_W \log (\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}))] \quad (50)$$

$$= \mathbb{E}_{\mathbf{x} \sim p^*} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})} [u_i z_j] - \mathbb{E}_{\mathbf{x}, \mathbf{z} \sim p_k(\mathbf{x}, \mathbf{z})} [u_i z_j] \quad , \quad (51)$$

for the weight matrix W (similar updates are used for the biases). The notation p_k refers to the distribution obtained after running a Markov Chain Monte Carlo method for k steps, starting at the data generating distribution p^* . If we took $k \rightarrow \infty$, then we would obtain p (the current model). Whilst we are free to choose any MCMC method, in the case of an RBM, the conditional independence statements in equation ?? allow us to implement efficient block-Gibbs sampling, where we alternate between sampling $\mathbf{u}|\mathbf{z}$ and $\mathbf{z}|\mathbf{u}$.

To our knowledge, there is no prior work comparing NCE to contrastive divergence in the case of the RBM, so the following experiments consider all three pairwise comparisons of VNCE, NCE and contrastive divergence.

6.2.2 Experimental Setup

Following ?, we use a toy dataset derived from all mutually exclusive 3×3 patches of the 11,000 16×16 images of handwritten digits from the USPS dataset.⁷ After thresholding the 256 intensity levels at 150, we obtained 275,000 9-dimensional binary vectors. We use this relatively low-dimensional toy dataset in order to make computing the partition function, and hence the log-likelihood, cheap.

For all three estimation methods, we train RBMs with varying number of hidden units: $m \in \{2, 4, 6, 8\}$. For each value of m , we use the same initial weights and biases across all three methods, sampling the weights and hidden biases with: $W_{i,j} \sim \mathcal{U}(-4\sqrt{\frac{6}{d+m}}, 4\sqrt{\frac{6}{d+m}})$ and setting the visible biases to $\frac{\log(p_i)}{1-\log(p_i)}$ where p_i is the empirical frequency of $u_i = 1$. Setting the visible biases in this way ensures that the initial model approximately matches the empirical marginals, and hence most of training consists in discovering the correlations between the visibles. For VNCE and NCE we must also initialise the scaling parameter; we set it to $-m \log(2) + \sum_{i=0}^d (\log(1 - p_i))$, which approximately normalises the initial model (it would exactly normalise it if the weight matrix and hidden biases were initialised to zero).

For contrastive divergence, we use the largest stable learning rate, which is 0.1, a batch size of 100 and train for 100 epochs. For each data point in a batch, we initialise a Markov chain at that point and run it for 1 step to produce 100 ‘fantasy’ samples. Many implementations of CD [?] do not actually take samples of the latent variables on the *final* step of Gibbs sampling. Instead they directly use the probability of the latents to compute the gradient update. Whilst this is a useful property of CD in the case of the RBM, for many other models we will be forced to take samples. Hence, to make our comparison of VNCE and CD fair for the general case, we always take samples.

⁷<https://cs.nyu.edu/~roweis/data.html>

For both VNCE and NCE, we set the noise distribution over visible units to be a product of marginals. Each marginal is a Bernoulli distribution, with parameter p_i equal to the empirical marginal probability. We set the ratio of noise to data samples, ν , equal to 1. We optimise their objective functions using stochastic gradient ascent (SGD) with the same settings as for contrastive divergence. We also investigated using a non-linear optimiser such as L-BFGS-B, however they tend not to scale well to large datasets. In addition, when using VNCE with L-BFGS-B inside the inner loop of the EM-algorithm, we have found that waiting for the built-in convergence criterion causes slow overall convergence. This can be remedied however by limiting the number of iterations to a small number rather than waiting for convergence.

For VNCE, we update the variational distribution q to equal the model’s current posterior over latents at every E-step of the EM algorithm. In the general case q will have its own parameters but for the RBM this is not necessary since we have an analytic expression for $p(\mathbf{z}|\mathbf{u})$ given in equation ?? that is efficient to sample from.

6.2.3 Experimental Results

Figure 6 shows the learning curves for training RBMs with SGD using a range of latent dimensions: $m \in \{2, 4, 6, 8\}$. In all cases, we see contrastive divergence converges to a higher log-likelihood than NCE or VNCE, although the gap in final log-likelihoods is very small for $m = 2$ hidden units, and grows with m . NCE and VNCE converge to the same final log-likelihood, showing that the variational approximation has not compromised the quality of the final model. The estimation methods differ significantly in terms of speed: The log-likelihood reached by VNCE at convergence is obtained 2-5x more quickly by NCE, and 10-50x more quickly by contrastive divergence. We note that the final gap between NCE/VNCE and contrastive divergence could be narrowed with a more sophisticated choice of noise distribution or by using a larger ratio of noise to data samples, at the cost of speed.

These results are strongly in favour of contrastive divergence as a method for training RBMs, particularly as we increase the number of hidden units. This observation is consistent with the limited theory that exists for contrastive divergence. As discussed in section 3.1, the difference between the maximum likelihood gradient updates and contrastive divergence updates as stated in equation ??, can be analytically bounded in the case of the RBM. ? provided a bound, later improved by ?, which was demonstrated to be both tight and small on a toy problem. More generally the authors show that the bound will be small when the absolute values of the RBM parameters are small. This theory gives us some assurance that contrastive divergence will work well for the RBM, and the theory appears to borne out in practice.

Nevertheless, the amount of bias in contrastive divergence learning can be much larger for particular data generating distributions [?], and may be larger for other classes of models. In particular, one might expect that for models where MCMC sampling is less efficient (because we cannot do block sampling, say), that contrastive divergence will be less effective. We investigate these possibilities in subsequent experiments.

usps-different_num_hiddens.pdf

Figure 5: Learning curves for CD, NCE and VNCE when training Restricted Boltzmann Machines on 3x3 binarised image patches using varying numbers of hidden units. All models here were trained with SGD. The green dashed line is the log-likelihood of the noise distribution used in NCE, which in this case is a product of the empirical marginals.

7 Adaptive VNCE

? proposed a variation of NCE where, after each update of the parameters θ , we reset the noise distribution to be equal to the current model. He showed that this scheme, termed ‘self-contrastive estimation’, yields the same gradient updates as maximum likelihood learning. We prove an analog of this result for the latent-variable case, and investigate how to efficiently implement it.

In the the appendix, section 7.2, we derive the gradient of the VNCE objective function, which is:

$$\nabla_{\theta} J_1(\theta) = \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q} \left[\frac{\nu}{r(\mathbf{x}, \mathbf{z}; \theta) + \nu} \nabla_{\theta} \log(\phi(\mathbf{x}, \mathbf{z}; \theta)) \right] \quad (52)$$

$$- \mathbb{E}_{\mathbf{y}} \frac{\nu}{\mathbb{E}_{\mathbf{z} \sim q} [r(\mathbf{y}, \mathbf{z}; \theta)] + \nu} \mathbb{E}_{\mathbf{z} \sim q} [r(\mathbf{y}, \mathbf{z}; \theta) \nabla_{\theta} \log(\phi(\mathbf{y}, \mathbf{z}; \theta))] \quad (53)$$

where,

$$r(\mathbf{u}, \mathbf{z}; \theta) = \frac{\phi(\mathbf{x}, \mathbf{z}; \theta)}{q(\mathbf{z}|\mathbf{u})p_{\mathbf{y}}(\mathbf{u})} \quad (54)$$

Now let us suppose that during each E step, not only do we set $q_k(\mathbf{z}|\mathbf{u}) = p(\mathbf{z}|\mathbf{u}; \boldsymbol{\theta}_k)$, but we also update the noise to be equal to the current model over visibles: $p_{\mathbf{y}}(\mathbf{u}) = \phi(\mathbf{u}; \boldsymbol{\theta}_k)$. Then,

$$r(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta}) = \frac{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})}{p(\mathbf{z}|\mathbf{u}; \boldsymbol{\theta}_k)\phi(\mathbf{u}; \boldsymbol{\theta}_k)} = 1 , \quad (55)$$

which implies that the gradient is:

$$\nabla_{\boldsymbol{\theta}} J_1(\boldsymbol{\theta}) = \frac{\nu}{1 + \nu} \left(\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} [\nabla_{\boldsymbol{\theta}} \log(\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}))] - \mathbb{E}_{\mathbf{y}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{y})} [\nabla_{\boldsymbol{\theta}} \log(\phi(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta}))] \right) , \quad (56)$$

which is proportional to the maximum likelihood gradient update, as presented in equation 9 in section 3.1.

Now, if during the M-step we only take a single gradient step using equation ??, and then repeat the E-step as stated above, then we have precisely recovered maximum likelihood estimation. If however, we take multiple gradient steps during the M-step, we have a new adaptive version of VNCE that approximates maximum likelihood learning, but is potentially faster. The main problem with this adaptive scheme is that at each E-step, we must acquire noise samples from $\phi(\mathbf{u}; \boldsymbol{\theta}_k)$, which could be costly if we have to run an Markov chain Monte Carlo method for a long time. This is the same problem faced by maximum likelihood learning, and one solution there is to only run the Markov chain for k steps, initialising it at a data point. In other words, we can use *contrastive divergence*.

Hence, we have developed an adaptive version of VNCE that includes maximum likelihood and contrastive divergence as special cases. The main question of interest is whether deviating from these special cases, by using multiple gradient updates during the M-step of learning, leads to a faster or more accurate learning method.

8 Present conclusions and remaining work

Our goal in this preliminary version of the thesis was twofold. Firstly, to give an exposition of an extension to noise-contrastive estimation which applies to unnormalised models with latent variables. Secondly, to evaluate the performance of this new method in the context of two models: a simple unnormalised mixture of gaussians, and a Restricted Boltzmann Machine.

Both evaluations are ongoing, so it is not yet clear how the new method compares to others, such as contrastive divergence. Nevertheless, current results demonstrate that the method can be successfully used for density estimation, at least on simple problems. The remainder of the thesis will need to critically assess how the method scales with model complexity, sample size and diversity of data generating distributions.

In terms of theory, there is also plenty of work remaining. In particular, we have discussed how it would be interesting to explore the convergence properties of this new estimator, as well as performing a mathematical comparison with contrastive divergence. Finally, once the investigation of RBMs is complete, the next logical step is to investigate

models with intractable posteriors that are hard to sample from, since our estimation method can, in theory, work in this setting, whilst contrastive divergence cannot.

9 Appendix

9.1 Variational lower bound $J_1(\theta)$

$$-\log[1 + \nu \exp(-h(\mathbf{u}; \boldsymbol{\theta}))] = -\log \left[1 + \nu \exp \left(-\log \frac{\phi(\mathbf{u}; \boldsymbol{\theta})}{p_{\mathbf{y}}(\mathbf{u})} \right) \right] \quad (57)$$

$$= -\log \left[1 + \nu \exp \left(\log \frac{p_{\mathbf{y}}(\mathbf{u})}{\phi(\mathbf{u}; \boldsymbol{\theta})} \right) \right] \quad (58)$$

$$= -\log \left[1 + \nu \frac{p_{\mathbf{y}}(\mathbf{u})}{\phi(\mathbf{u}; \boldsymbol{\theta})} \right] \quad (59)$$

Let further

$$r(\mathbf{u}; \boldsymbol{\theta}) = \exp(h(\mathbf{u}; \boldsymbol{\theta})) = \frac{\phi(\mathbf{u}; \boldsymbol{\theta})}{p_{\mathbf{y}}(\mathbf{u})} \quad (60)$$

and

$$g(r) = -\log \left(1 + \nu \frac{1}{r} \right), \quad (61)$$

we then obtain

$$-\log[1 + \nu \exp(-h(\mathbf{u}; \boldsymbol{\theta}))] = g(r(\mathbf{u}; \boldsymbol{\theta})). \quad (62)$$

The first derivative of g with respect to r is

$$\frac{dg}{dr} = \frac{-1}{1 + \nu \frac{1}{r}} \frac{-\nu}{r^2} \quad (63)$$

$$= \frac{\nu}{r^2 + \nu r} \quad (64)$$

The second derivative of g with respect to r is thus

$$\frac{d^2g}{dr^2} = -\nu \frac{2r + \nu}{(r^2 + \nu r)^2}, \quad (65)$$

which is always negative for $r > 0$. The function $g(r)$ is thus concave.

We now introduce an auxiliary distribution $q_u(\mathbf{z})$ over \mathbf{z} and write

$$\phi(\mathbf{u}; \boldsymbol{\theta}) = \int \phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta}) d\mathbf{z} \quad (66)$$

$$= \int q_u(\mathbf{z}) \frac{\phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta})}{q_u(\mathbf{z})} d\mathbf{z} \quad (67)$$

$$= \mathbb{E}_{\mathbf{z} \sim q_u} \left[\frac{\phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta})}{q_u(\mathbf{z})} \right] \quad (68)$$

This corresponds to estimating $\phi(\mathbf{u}, \boldsymbol{\theta})$ by importance sampling. The subscript for $q_u(\mathbf{z})$ is meant to indicate that we can use different auxiliary distributions for different values of \mathbf{u} .

We can thus write $r(\mathbf{u}; \boldsymbol{\theta})$ as

$$r(\mathbf{u}; \boldsymbol{\theta}) = \frac{1}{p_{\mathbf{y}}(\mathbf{u})} \mathbb{E}_{\mathbf{z} \sim q_u} \left[\frac{\phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta})}{q_u(\mathbf{z})} \right] \quad (69)$$

$$= \mathbb{E}_{\mathbf{z} \sim q_u} \left[\frac{\phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta})}{q_u(\mathbf{z}) p_{\mathbf{y}}(\mathbf{u})} \right] \quad (70)$$

and since g is concave, we have

$$-\log[1 + \nu \exp(-h(\mathbf{u}; \boldsymbol{\theta}))] = g(r(\mathbf{u}; \boldsymbol{\theta})) \quad (71)$$

$$= g \left(\mathbb{E}_{\mathbf{z} \sim q_u} \left[\frac{\phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta})}{q_u(\mathbf{z}) p_{\mathbf{y}}(\mathbf{u})} \right] \right) \quad (72)$$

$$\geq \mathbb{E}_{\mathbf{z} \sim q_u} g \left(\frac{\phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta})}{q_u(\mathbf{z}) p_{\mathbf{y}}(\mathbf{u})} \right). \quad (73)$$

With the definition of g , we thus obtain

$$-\log[1 + \nu \exp(-h(\mathbf{u}; \boldsymbol{\theta}))] \geq -\mathbb{E}_{\mathbf{z} \sim q_u} \log \left[1 + \nu \frac{q_u(\mathbf{z}) p_{\mathbf{y}}(\mathbf{u})}{\phi(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta})} \right]. \quad (74)$$

We now plug this relation into the definition of $J(\boldsymbol{\theta})$:

$$J(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x}} \log [1 + \nu \exp(-h(\mathbf{x}; \boldsymbol{\theta}))] - \nu \mathbb{E}_{\mathbf{y}} \log [1 + 1/\nu \exp(h(\mathbf{y}; \boldsymbol{\theta}))] \quad (75)$$

$$\geq -\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log \left[1 + \nu \frac{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})} \right] - \nu \mathbb{E}_{\mathbf{y}} \log [1 + 1/\nu \exp(h(\mathbf{y}; \boldsymbol{\theta}))] \quad (76)$$

$$\geq -\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{x})} \log \left[1 + \nu \frac{q(\mathbf{z}|\mathbf{x}) p_{\mathbf{y}}(\mathbf{x})}{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})} \right] - \nu \mathbb{E}_{\mathbf{y}} \log \left[1 + \frac{1}{\nu} \mathbb{E}_{\mathbf{z} \sim q(\mathbf{z}|\mathbf{y})} \left[\frac{\phi(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{y}) p_{\mathbf{y}}(\mathbf{y})} \right] \right]. \quad (77)$$

9.2 Gradient of $J_1(\boldsymbol{\theta})$

To find the argmax of $J_1(\boldsymbol{\theta})$, we need to differentiate it. To do so, let us first further reduce the notational burden, by writing:

$$J_1(\boldsymbol{\theta}) = -\mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q_k} [\log(\psi_1(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}))] - \nu \mathbb{E}_{\mathbf{y}} [\log(\psi_2(\mathbf{y}; \boldsymbol{\theta}))] \quad (78)$$

where

$$\psi_1(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta}) = 1 + \frac{\nu}{r(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})} \quad \psi_2(\mathbf{y}; \boldsymbol{\theta}) = 1 + \frac{1}{\nu} \mathbb{E}_{\mathbf{z} \sim q_k} [r(\mathbf{y}, \mathbf{z}; \boldsymbol{\theta})] \quad (79)$$

$$\text{and} \quad r(\mathbf{u}, \mathbf{z}; \boldsymbol{\theta}) = \frac{\phi(\mathbf{x}, \mathbf{z}; \boldsymbol{\theta})}{q(\mathbf{z}|\mathbf{u}) p_{\mathbf{y}}(\mathbf{u})} \quad (80)$$

Now, let us take derivative with respect to θ :

$$\nabla_{\theta} \log(\psi_1(\mathbf{x}, \mathbf{z}; \theta)) = \frac{1}{\psi_1(\mathbf{x}, \mathbf{z}; \theta)} \nabla_{\theta} \psi_1(\mathbf{x}, \mathbf{z}; \theta) \quad (81)$$

$$= \frac{1}{\psi_1(\mathbf{x}, \mathbf{z}; \theta)} \frac{-\nu}{r(\mathbf{x}, \mathbf{z}; \theta)^2} \nabla_{\theta} r(\mathbf{x}, \mathbf{z}; \theta) \quad (82)$$

$$= \frac{1}{\psi_1(\mathbf{x}, \mathbf{z}; \theta)} \frac{-\nu}{r(\mathbf{x}, \mathbf{z}; \theta)^2} \frac{\nabla_{\theta} \phi(\mathbf{x}, \mathbf{z}; \theta)}{q_k(\mathbf{z} | \mathbf{x}) p_{\mathbf{y}}(\mathbf{x})} \quad (83)$$

Hence, we need:

$$\nabla_{\theta} \phi(\mathbf{x}, \mathbf{z}; \theta) = \nabla_{\theta} \exp(\log(\phi(\mathbf{x}, \mathbf{z}; \theta))) \quad (84)$$

$$= [\nabla_{\theta} \log(\phi(\mathbf{x}, \mathbf{z}; \theta))] \exp(\log(\phi(\mathbf{x}, \mathbf{z}; \theta))) \quad (85)$$

$$= [\nabla_{\theta} \log(\phi(\mathbf{x}, \mathbf{z}; \theta))] \phi(\mathbf{x}, \mathbf{z}; \theta) \quad (86)$$

Plugging this back in, we get:

$$\nabla_{\theta} \log(\psi_1(\mathbf{x}, \mathbf{z}; \theta)) = \frac{1}{\psi_1(\mathbf{x}, \mathbf{z}; \theta)} \frac{-\nu}{r(\mathbf{x}, \mathbf{z}; \theta)^2} \frac{[\nabla_{\theta} \log(\phi(\mathbf{x}, \mathbf{z}; \theta))] \phi(\mathbf{x}, \mathbf{z}; \theta)}{q_k(\mathbf{z} | \mathbf{x}) p_{\mathbf{y}}(\mathbf{x})} \quad (87)$$

$$= \frac{1}{\psi_1(\mathbf{x}, \mathbf{z}; \theta)} \frac{-\nu}{r(\mathbf{x}, \mathbf{z}; \theta)} [\nabla_{\theta} \log(\phi(\mathbf{x}, \mathbf{z}; \theta))] \quad (88)$$

$$= \frac{-\nu}{r(\mathbf{x}, \mathbf{z}; \theta) + \nu} [\nabla_{\theta} \log(\phi(\mathbf{x}, \mathbf{z}; \theta))] \quad (89)$$

$$(90)$$

Where in the final line we used the fact that $\psi_1(\mathbf{x}, \mathbf{z}; \theta) - 1 = \frac{\nu}{r(\mathbf{x}, \mathbf{z}; \theta)}$.

Now we find the derivative of the second (blue) term of $J_1^K(\theta)$. Recalling that $\psi_2(\mathbf{y}; \theta) = 1 + \frac{1}{\nu} \mathbb{E}_{\mathbf{z} \sim q_k}[r(\mathbf{y}, \mathbf{z}; \theta)]$, we have:

$$\nabla_{\theta} \log(\psi_2(\mathbf{y}; \theta)) = \frac{1}{\psi_2(\mathbf{y}; \theta)} \nabla_{\theta} \psi_2(\mathbf{y}; \theta) \quad (91)$$

$$= \frac{1}{\psi_2(\mathbf{y}; \theta)} \frac{1}{\nu} \mathbb{E}_{\mathbf{z} \sim q_k} \left[\frac{\nabla_{\theta} \phi(\mathbf{y}, \mathbf{z}; \theta)}{q_k(\mathbf{z} | \mathbf{y}) p_{\mathbf{y}}(\mathbf{y})} \right] \quad (92)$$

$$= \frac{1}{\nu} \frac{1}{\psi_2(\mathbf{y}; \theta)} \mathbb{E}_{\mathbf{z} \sim q_k} \left[r(\mathbf{y}, \mathbf{z}; \theta) [\nabla_{\theta} \log(\phi(\mathbf{y}, \mathbf{z}; \theta))] \right] \quad (93)$$

$$(94)$$

Putting this all together, we arrive at:

$$\nabla_{\theta} (J_1^k(\theta)) = \mathbb{E}_{\mathbf{x}} \mathbb{E}_{\mathbf{z} \sim q_k} \frac{\nu}{r(\mathbf{x}, \mathbf{z}; \theta) + \nu} [\nabla_{\theta} \log(\phi(\mathbf{x}, \mathbf{z}; \theta))] \quad (95)$$

$$- \mathbb{E}_{\mathbf{y}} \frac{1}{1 + \frac{1}{\nu} \mathbb{E}_{\mathbf{z} \sim q_k}[r(\mathbf{y}, \mathbf{z}; \theta)]} \mathbb{E}_{\mathbf{z} \sim q_k} \left[r(\mathbf{y}, \mathbf{z}; \theta) [\nabla_{\theta} \log(\phi(\mathbf{y}, \mathbf{z}; \theta))] \right] \quad (96)$$

$$(97)$$