This version of the paper contains some typographical corrections made since publication, in Algorithm 1.

The Neural Autoregressive Distribution Estimator

Hugo Larochelle
Department of Computer Science
University of Toronto
Toronto, Canada

Iain Murray
School of Informatics
University of Edinburgh
Edinburgh, Scotland

Abstract

We describe a new approach for modeling the distribution of high-dimensional vectors of discrete variables. This model is inspired by the restricted Boltzmann machine (RBM), which has been shown to be a powerful model of such distributions. However, an RBM typically does not provide a tractable distribution estimator, since evaluating the probability it assigns to some given observation requires the computation of the so-called partition function, which itself is intractable for RBMs of even moderate size. Our model circumvents this difficulty by decomposing the joint distribution of observations into tractable conditional distributions and modeling each conditional using a non-linear function similar to a conditional of an RBM. Our model can also be interpreted as an autoencoder wired such that its output can be used to assign valid probabilities to observations. We show that this new model outperforms other multivariate binary distribution estimators on several datasets and performs similarly to a large (but intractable) RBM.

1 Introduction

The problem of estimating the distribution of multivariate data is perhaps the most general problem addressed by machine learning. Indeed, given the joint distribution of all variables we are interested in, we can potentially answer any question (though possibly only approximately) about the relationship between these variables, e.g. what is the most likely value of a subset of the observations given others, a question that covers any supervised learning problem. Alternatively, one could be interested in learning the distribution of observations to extract features derived from the statistical regularities exhibited by those observations.

In recent years, the restricted Boltzmann machine (RBM) (Smolensky, 1986; Freund & Haussler, 1992; Hinton, 2002) has frequently been used as a feature extractor. RBMs model the distribution of observations using binary hidden variables. By training an RBM to learn the distribution of observations, it is then possible to use the posterior over these hidden variables given some observation as learned features, which can be composed into several layers and fine-tuned for some given task (Hinton et al., 2006; Bengio et al., 2007). If the observations can be decomposed into an input $\mathbf{x}$ and a target $\mathbf{y}$, then an RBM trained on such pairs can also be used to predict the missing target for new inputs (Larochelle & Bengio, 2008; Tieleman, 2008).

One problem an RBM isn't suited for is, oddly enough, estimating the joint probability of a given observation. Evaluating joint probabilities under the model requires computing the so-called partition function or normalization constant of the RBM which, even for models of moderate size and observations of moderate dimensionality, is intractable. This is unfortunate, as it can make it difficult to use the RBM as a generic modeling tool for larger probabilistic systems. Even for a simple probabilistic clustering model or a Bayes classifier, some approximations need to be made.

In this paper, we describe the Neural Autoregressive Distribution Estimator (NADE), which is inspired by the RBM but is a tractable distribution estimator. Indeed, computing probabilities of observations or sampling new observations from the model can be done exactly and efficiently under NADE. Focusing on the problem of estimating the distribution of binary multivariate observations, we show on several datasets that NADE outperforms other tractable distribution estimators. We also show that its performance is very close to that of a large but intractable RBM whose partition

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

function has been approximated.

2 The Restricted Boltzmann Machine

An RBM is a Markov random field with bipartite structure, where a set of weights $\mathbf{W}$ connect observation variables $\mathbf{v}$ (with bias parameters $\mathbf{b}$) to hidden variables $\mathbf{h}$ (with bias $\mathbf{c}$). More specifically, from the energy function

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{v} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} \tag{1}$$

probabilities are assigned to any observation $\mathbf{v}$ as follows:

$$p(\mathbf{v}) = \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))/Z\,, \tag{2}$$

where Z is known as the partition function and ensures that $p(\mathbf{v})$ is a valid distribution and sums to 1. Unfortunately, for RBMs with even just a few hidden variables (i.e. more than 30), computing this partition function becomes intractable. For this reason, training under an RBM requires approximating the log-likelihood gradient on the parameters, with contrastive divergence (Hinton, 2002) being the most popular approach.

The RBM's intractable partition function reduces its use as a modeling tool that can be incorporated into some other larger probabilistic system. Even in a simple Bayes classifier where one RBM would need to be trained for each class, they cannot be used directly. However, in this particular case the relative partition functions of the RBMs have successfully been approximated to build a classifier (Hinton, 2002; Schmah et al., 2009).

Not knowing the exact value of Z also makes it hard to quantitatively evaluate how well the distribution estimated by the RBM fits the true distribution of our observations, e.g. by computing the average negative log-likelihood of test observations and comparing it to that of other methods. However, Salakhutdinov and Murray (2008) showed that using annealed importance sampling, it is possible to obtain a reasonable approximation to Z. By assuming this approximation corresponds to the true value of Z, we can then make such quantitative comparisons, with the caveat that there are no strong guarantees on the quality of the estimate of Z. Still, the experiments they report do suggest that the RBM is a very powerful model, outperforming a (tractable) mixture of multivariate Bernoullis by a large margin.

Hence the question: can we design a model which, much like an RBM, is a powerful model of multivariate discrete data but can also provide a tractable distribution estimator? In this paper, we describe such a model.

3 Converting an RBM into a Bayesian Network

Another type of model which has also been shown to provide a powerful framework for deriving tractable distribution estimators is the family of fully visible Bayesian networks (e.g. Frey et al., 1996; Bengio and Bengio, 2000), which decompose the observation's probability distribution as follows:

$$p(\mathbf{v}) = \prod_{i=1}^{D} p(v_i|\mathbf{v}_{\text{parents}(i)})\,, \tag{3}$$

where all observation variables v_i are arranged into a directed acyclic graph and $\mathbf{v}_{\text{parents}(i)}$ corresponds to all the variables in $\mathbf{v}$ that are parents of v_i into that graph. The main advantage of such a model is that if all conditional distributions $p(v_i|\mathbf{v}_{\text{parents}(i)})$ are tractable then so is the whole distribution $p(\mathbf{v})$.

One such model which has been shown to be a good model of multivariate discrete distributions is the fully visible sigmoid belief network (FVSBN) (Neal, 1992; Frey et al., 1996). In the FVSBN, the acyclic graph is obtained by defining the parents of v_i as all variables that are to its left[1], or $\mathbf{v}_{\text{parents}(i)} = \mathbf{v}_{<i}$, where $\mathbf{v}_{<i}$ refers to the subvector containing all variables v_j such that $j < i$ (see Figure 1 for an illustration). As for the conditionals $p(v_i|\mathbf{v}_{\text{parents}(i)})$, they each correspond to a log-linear model or logistic regressor:

$$p(v_i|\mathbf{v}_{\text{parents}(i)}) = \text{sigm}\Big(b_i + \sum_{j<i} W_{ij} v_j\Big)\,, \tag{4}$$

where $\text{sigm}(x) = 1/\left(1 + \exp(-x)\right)$.

Of course, the distribution of an RBM could also be written in the form of Equation 3,

$$\begin{aligned} p(\mathbf{v}) &= \prod_{i=1}^{D} p(v_i|\mathbf{v}_{<i}) \\ &= \prod_{i=1}^{D} p(v_i, \mathbf{v}_{<i})/p(\mathbf{v}_{<i}) \\ &= \prod_{i=1}^{D} \frac{\sum_{\mathbf{v}_{>i}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v}_{j\geq i}} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))}\,, \end{aligned} \tag{5}$$

but unfortunately most of the conditionals are themselves intractable. However, if they could each be approximated, perhaps the associated distribution estimator would still be powerful enough to yield good estimations.

[1] Before building the graph, the variables are usually randomly reordered. In our notation, we'll assume that this operation has already been performed on $\mathbf{v}$.

To achieve this, one could consider the following approach. To approximate the conditional $p(v_i|\mathbf{v}_{<i})$ under an RBM, we first find an approximation $q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})$ for $p(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})$, such that $q(v_i|\mathbf{v}_{<i})$ can be easily obtained. Such a choice for $q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})$ and a popular approach for RBMs in general is the mean-field distribution, where a factorial decomposition is assumed:

$$q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}) = \mu_i(i)^{v_i}(1-\mu_i(i))^{1-v_i} \prod_{j>i} \mu_j(i)^{v_j}(1-\mu_j(i))^{1-v_j} \prod_k \tau_k(i)^{h_k}(1-\tau_k(i))^{1-h_k}, \quad (6)$$

where $\mu_j(i)$ is the marginal probability of observation v_j being equal to 1, given $\mathbf{v}_{<i}$. Similarly, $\tau_k(i)$ is the marginal probability of hidden variable h_k being equal to 1. The dependence on i comes from conditioning on $\mathbf{v}_{<i}$, i.e. for each value of i. The mean-field approximation then proceeds by finding the parameters $\mu_j(i)$ for $j \geq i$ and $\tau_k(i)$ which minimize the KL divergence between $q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})$ and $p(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})$. The most frequently used approach for doing this consists in setting the derivatives of the KL to 0, yielding the following equations (see Appendix for a derivation):

$$\tau_k(i) = \text{sigm}\left(c_k + \sum_{j \geq i} W_{kj}\mu_j(i) + \sum_{j<i} W_{kj}v_j\right) \quad (7)$$

$$\mu_j(i) = \text{sigm}\left(b_j + \sum_k W_{kj}\tau_k(i)\right) \quad \forall j \geq i\,. \quad (8)$$

The fixed point satisfying these equations is found by initializing $\mu_j(i)$ and $\tau_k(i)$ to 0 and alternating between applying Equations 7 and 8 from right to left. This procedure is guaranteed to converge to a fixed point, which might not be a global optimum. Still, the general principle of mean-field has been shown to work well in practice for RBMs (Welling & Hinton, 2002; Salakhutdinov & Hinton, 2009). In the setting of converting an RBM into a Bayesian network, the value of $\mu_j(i)$ would be used as an estimate of $p(v_i = 1|\mathbf{v}_{<i})$.

However, this mean-field procedure can be quite slow, with convergence often taking around 20 iterations. Each iteration can be quite costly for large dimensionalities of observations $\mathbf{v}$ or hidden vectors $\mathbf{h}$. Moreover, the same procedure would need to be followed for each observation v_i, making it impractical.

4 The Neural Autoregressive Distribution Estimator

While not directly applicable, the mean-field procedure of the last section can serve as an inspiration for coming up with powerful functions to use in a Bayesian network and model the conditionals $p(v_i = 1|\mathbf{v}_{<i})$.

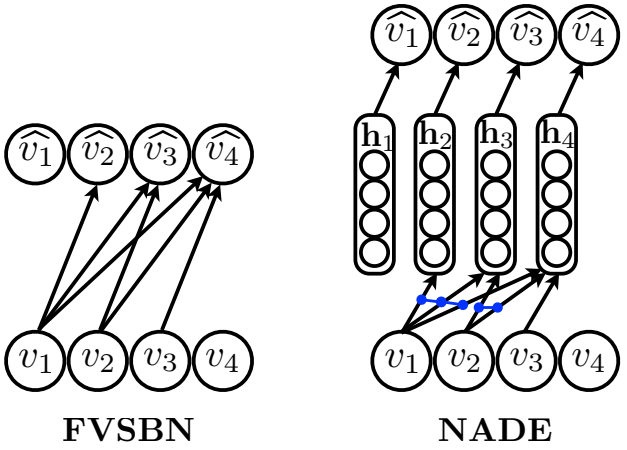

Figure 1: **(Left)** Illustration of a fully visible sigmoid belief network. **(Right)** Illustration of a neural autoregressive distribution estimator. $\widehat{v_i}$ is used as a shorthand for $p(v_i = 1|\mathbf{v}_{<i})$. Arrows connected by a blue line correspond to connections with shared or tied parameters.

For instance, consider the application of the aforementioned mean-field procedure for only one iteration. With $\mu_j(i)$ initialized to 0 for $j \geq i$, we can rewrite this procedure as follows:

$$p(v_i = 1|\mathbf{v}_{<i}) = \text{sigm}\left(b_i + (\mathbf{W}^\top)_{i,\cdot}\mathbf{h}_i\right) \quad (9)$$

$$\mathbf{h}_i = \text{sigm}\left(\mathbf{c} + \mathbf{W}_{\cdot,<i}\mathbf{v}_{<i}\right), \quad (10)$$

which corresponds to a feed-forward neural network with a single hidden layer, and tied weighted connections going in and out of the hidden layer. Moreover, since there is one neural network for each conditional $p(v_i = 1|\mathbf{v}_{<i})$, connections are also tied *across* these neural networks.

The tied connections can be leveraged to speed up the computations of each conditional by sharing calculations across neural networks. Indeed, the i^{th} and $(i+1)^{\text{th}}$ hidden layer activations passed into the sigmoid in Equation (10) are almost exactly the same. The difference between the two is simply

$$(\mathbf{c} + \mathbf{W}_{\cdot,<i+1}\mathbf{v}_{<i+1}) - (\mathbf{c} + \mathbf{W}_{\cdot,<i}\mathbf{v}_{<i}) = \mathbf{W}_{\cdot,i+1}v_{i+1}$$

which can be computed in $O(H)$, where H is the number of hidden units. Hence, the complete cost of computing $p(\mathbf{v})$ is $O(HD)$, instead of the $O(HD^2)$ cost of a naive procedure that doesn't take advantage of the weight sharing across conditionals.

We call the proposed new Bayesian network for distribution estimation the Neural Autoregressive Distribution

Algorithm 1 Computation of $p(\mathbf{v})$ and learning gradients for NADE

Input: training observation vector $\mathbf{v}$
Output: $p(\mathbf{v})$ and gradients of $-\log p(\mathbf{v})$ on parameters

Computing $p(\mathbf{v})$
$\mathbf{a} \leftarrow \mathbf{c}$
$p(\mathbf{v}) \leftarrow 1$
for i from 1 to D **do**
  $\mathbf{h}_i \leftarrow \text{sigm}(\mathbf{a})$
  $p(v_i = 1|\mathbf{v}_{<i}) \leftarrow \text{sigm}(b_i + \mathbf{V}_{i,.}\mathbf{h}_i)$
  $p(\mathbf{v}) \leftarrow p(\mathbf{v}) \left(p(v_i = 1|\mathbf{v}_{<i})^{v_i} \right.$
    $\left. (1 - p(v_i = 1|\mathbf{v}_{<i}))^{1-v_i}\right)$
  $\mathbf{a} \leftarrow \mathbf{a} + \mathbf{W}_{.,i} v_i$
end for

Computing gradients of $-\log p(\mathbf{v})$
$\delta\mathbf{a} \leftarrow 0$
$\delta\mathbf{c} \leftarrow 0$
for i from D to 1 **do**
  $\delta b_i \leftarrow (p(v_i = 1|\mathbf{v}_{<i}) - v_i)$
  $\delta\mathbf{V}_{i,.} \leftarrow (p(v_i = 1|\mathbf{v}_{<i}) - v_i)\, \mathbf{h}_i^\top$
  $\delta\mathbf{h}_i \leftarrow (p(v_i = 1|\mathbf{v}_{<i}) - v_i)\, \mathbf{V}_{i,.}^\top$
  $\delta\mathbf{c} \leftarrow \delta\mathbf{c} + (\delta\mathbf{h}_i)\mathbf{h}_i(1 - \mathbf{h}_i)$
  $\delta\mathbf{W}_{.,i} \leftarrow (\delta\mathbf{a})v_i$
  $\delta\mathbf{a} \leftarrow \delta\mathbf{a} + (\delta\mathbf{h}_i)\mathbf{h}_i(1 - \mathbf{h}_i)$
end for

return $p(\mathbf{v}), \delta\mathbf{b}, \delta\mathbf{V}, \delta\mathbf{c}, \delta\mathbf{W}$

Estimator (NADE). Training under NADE is done by minimizing the average negative log-likelihood of the parameters given the training set:

$$\frac{1}{T}\sum_{t=1}^{T} -\log p(\mathbf{v}_t) = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{D} -\log p(v_i|\mathbf{v}_{<i})\,, \quad (11)$$

where the dependence on the model parameters is given in (9) and (10). This minimization can be done using any non-linear optimization method, such as stochastic gradient descent.

In practice, we have found that untying the connections going in and out of the hidden units gives better distribution estimation performance. In other words, we learn a separate set of weights $\mathbf{V}$ (a $D \times H$ matrix) for the connections from the hidden units to the outputs, replacing $\mathbf{W}^\top$ in (9). Notice that NADE was not gaining computationally from sharing these connections, hence untying them does not make it slower. Also, using a separate set of weights means that we can invoke the universal approximation theorems for neural networks for each conditional, since $\mathbf{V}$ could be such that each hidden unit only has non-zero connections to one of the conditional outputs. This is of course unlikely to be learned in practice, but conceptually speaking it is good to know that by increasing the number of hidden units, we can cover a family of distributions whose best distribution is increasingly close to the true distribution.

Figure 1 illustrates this new model and Algorithm 1 gives the pseudocode for computing $p(\mathbf{v})$ and the parameter derivatives of $-\log p(\mathbf{v})$ under NADE.

5 Related Work

Several models for tractably estimating the distribution of multivariate discrete or binary data have previously been proposed. The most common is probably a mixture model with multivariate Bernoullis as components for binary data. Lowd and Domingos (2005) argued that it improves over a Bayesian network with probabilistic decision trees for each conditional.

Better Bayesian networks can be obtained by using log-linear logistic regressors for the conditionals, also known as fully visible sigmoid belief networks or logistic autoregressive Bayesian networks (Frey, 1998). While the choice of log-linear conditionals probably yields a misspecified model in most cases, these networks do tend to perform better than mixture models (Frey et al., 1996).

While RBMs of moderate size are intractable as distribution estimators, they can be made small enough to be tractable. Even small RBMs are equivalent to very large mixture models with constrained parameters. Indeed, Larochelle et al. (2010) have shown that small, tractable RBMs can outperform standard mixture models.

In this paper, we show that NADE outperforms all of the aforementioned models. Though this comparison excludes intractable models such as factor models in general or models based on orthogonal expansions of Walsh bases (Ott & Kronmal, 1976; Chen et al., 1989), we also provide a comparison with a large and intractable RBM which suggests that NADE has comparable modeling power.

Bengio and Bengio (2000) also proposed using neural networks for the conditionals of a Bayesian network. While there was some amount of parameter sharing across conditionals, computing $p(\mathbf{v})$ under their model costs $O(HD^2)$ instead of the $O(HD)$ cost of NADE, though it should be noted that the optimal H (in terms of generalization) might be smaller for their model than in NADE. The lack of parameter sharing in their proposal also seems to have made the model more susceptible to overfitting: pruning heuristics based on pairwise statistical tests had to be used on some datasets. In

contrast, we did not have to use any pruning for NADE, presumably because of the particular choice of parameter sharing that it implements.

Part of the motivation behind this work is that the mean-field approximation in an RBM is too slow to be practical in estimating the RBM conditionals. Recent work by Salakhutdinov and Larochelle (2010) proposed a neural net training procedure that found weights whereby one mean-field iteration would give an output close to the result of running the mean-field procedure to convergence. In the setting of estimating conditionals $p(v_i|\mathbf{v}_{<i})$, their procedure would be similar to training NADE on the output of mean-field instead of on the training observations, which is much less direct and would also require that an RBM be trained first.

NADE is also closely related to autoencoders, i.e. neural networks trained to reproduce their inputs at their outputs. In particular, the denoising autoencoder (Vincent et al., 2008) is a variation where a proportion of the inputs are "destroyed" (i.e. set to 0) but must be reconstructed at the output based on the other inputs. An alternative view of NADE is as an autoencoder that has been wired such that its output can be used to assign probabilities to observations in a valid way. It also has all of the flexibility of normal autoencoders in the choice of hidden layer activation function, which need not be the sigmoid function in order to be a valid distribution estimator.

Finally, the general approach of decomposing the probability of an intractable model and approximating the conditionals was previously explored by Li and Stephens (2003) for a model of linkage disequilibrium in genetic data.

6 Experiments

We conducted experiments on eight datasets of multivariate binary observations from several application domains, including biological, image and text data. All datasets were separated into training, validation and test portions. The same datasets were used in Larochelle et al. (2010), which also provided results for some of the baselines. The performance of NADE was compared to that of the following models:

- **MoB**: a mixture of multivariate Bernoullis, trained using the EM algorithm;
- **RBM**: a restricted Boltzmann machine made tractable by using only 23 hidden units, trained by contrastive divergence with up to 25 steps of Gibbs sampling;
- **RBM mult.**: an RBM with groups of multinomial hidden units, i.e. with a hidden layer where units have been segmented into groups within which only one unit can be active (i.e. equal to 1). It was proposed by Larochelle et al. (2010) to allow the number of parameters of the RBM to grow while maintaining tractability;
- **RBForest**: an RBM where the activation of hidden units within a group obey tree constraints (see Larochelle et al. (2010) for more details);
- **FVSBN**: a fully visible sigmoid belief network.

Both FVSBN and NADE were trained by stochastic gradient descent. The initial learning rate was set to a value in $\{0.05, 0.005, 0.0005\}$ and the decrease constant[2] chosen from $\{0, 0.001, 0.000001\}$ based on performance on a validation set. Early stopping with a look ahead of 10 iterations was used to control for overfitting, with a maximum number of iterations of 500. No other regularization technique (e.g. priors on the parameters) were used for either FVSBN or NADE. A single random ordering of the observation variables was used to decompose $p(\mathbf{v})$ into conditionals, and the same ordering was used for both FVSBN and NADE. In all experiments, we used 500 hidden units for NADE. Code for NADE and for collecting the datasets is available here: `http://www.cs.toronto.edu/~larocheh/code/nade.tgz`. The results for MoB and the RBM models where taken from Larochelle et al. (2010). For MoB, the number of mixture components was chosen among $\{32, 64, 128, 256, 512, 1024\}$ based on the validation set performance and early stopping was used to determine the number of EM iterations (no other regularization technique or prior were used). See Larochelle et al. (2010) for more details.

The test average log-likelihood for the different datasets are given in Table 1. NADE is the sole best performing model on 6 of the datasets, with FVSBN outperforming NADE on only 1 dataset. On the remaining dataset, both NADE and FVSBN are the best performing models with statistically indistinguishable results. NADE also beats the other models on all datasets by a large margin.

6.1 Sensitivity to ordering of observations

As mentioned earlier, NADE requires that the observation variables be put in some order to decompose the joint likelihood into conditionals. In our experiments, the ordering was determined by simply randomly shuffling the observations, but one could wonder whether this has a significant impact on the performance of NADE.

[2]The learning rate λ_t for the t^{th} parameter update is obtained from the decrease constant γ and initial learning rate λ by combining them as follows: $\lambda_t = \frac{\lambda}{1+\gamma t}$.

Table 1: Distribution estimation results. To normalize the results, the average test log-likelihood (ALL) for each model on a given dataset was subtracted by the ALL of MoB (which is given in the last row under "Normalization"). 95% confidence intervals are also given. The best result as well as any other result with an overlapping confidence interval is shown in bold.

Model	ADULT	CONNECT-4	DNA	MUSHROOMS	NIPS-0-12	OCR-LETTERS	RCV1	WEB
MoB	0.00 ± 0.10	0.00 ± 0.04	0.00 ± 0.53	0.00 ± 0.10	0.00 ± 1.12	0.00 ± 0.32	0.00 ± 0.11	0.00 ± 0.23
RBM	4.18 ± 0.06	0.75 ± 0.02	1.29 ± 0.48	-0.69 ± 0.09	12.65 ± 1.07	-2.49 ± 0.30	-1.29 ± 0.11	0.78 ± 0.20
RBM mult.	4.15 ± 0.06	-1.72 ± 0.03	1.45 ± 0.40	-0.69 ± 0.05	11.25 ± 1.06	0.99 ± 0.29	-0.04 ± 0.11	0.02 ± 0.21
RBForest	4.12 ± 0.06	0.59 ± 0.02	1.39 ± 0.49	0.04 ± 0.07	12.61 ± 1.07	3.78 ± 0.28	0.56 ± 0.11	-0.15 ± 0.21
FVSBN	**7.27 ± 0.04**	11.02 ± 0.01	**14.55 ± 0.50**	4.19 ± 0.05	13.14 ± 0.98	1.26 ± 0.23	-2.24 ± 0.11	0.81 ± 0.20
NADE	**7.25 ± 0.05**	**11.42 ± 0.01**	13.38 ± 0.57	**4.65 ± 0.04**	**16.94 ± 1.11**	**13.34 ± 0.21**	**0.93 ± 0.11**	**1.77 ± 0.20**
Normalization	-20.44	-23.41	-98.19	-14.46	-290.02	-40.56	-47.59	-30.16

To measure the sensitivity of NADE to the ordering of the observations we trained a dozen separate models for the MUSHROOMS, DNA and NIPS-0-12 datasets using different random shufflings. We then computed the standard deviation of the twelve associated test log-likelihood averages, for each of the datasets. Standard deviations of 0.045, 0.050 and 0.150 were observed on MUSHROOMS, DNA and NIPS-0-12 respectively, which is quite reasonable when compared to the intrinsic uncertainty associated with using a finite test set (see the confidence intervals of Table 1). Hence, it does not seem necessary to optimize the ordering of the observation variables.

One could try to reduce the variance of the learned solution by training an ensemble of several NADE models on different observation orderings, while sharing the weight matrix $\mathbf{W}$ across those models but using different output matrices $\mathbf{V}$. While we haven't extensively experimented with this variant, we have found such sharing to produce better filters when used on the binarized MNIST dataset (see next section).

6.2 NADE vs. an intractable RBM

While NADE was inspired by the RBM, does its performance come close to that of the RBM in its most typical regime, i.e. with hundreds of hidden units? In other words, was tractability gained with a loss in performance?

To answer these questions, we trained NADE on a binarized version of the MNIST dataset. This version was used by Salakhutdinov and Murray (2008) to train RBMs with different versions of contrastive divergence and evaluate them as distribution estimators. Since the partition function cannot be computed exactly, it was approximated using annealed importance sampling. This method estimates the mean of some unbounded positive weights by an empirical mean of samples. It isn't possible to meaningfully upper-bound the partition function from these results: the true test log-likelihood averages could be much smaller than the values and error bars reported by Salakhutdinov and Murray (2008), although their approximations were shown to be accurate in a tractable case.

RBMs with 500 hidden units were reported to obtain -125.53, -105.50 and -86.34 in average test log-likelihood when trained using contrastive divergence with 1, 3 and 25 steps of Gibbs sampling, respectively. In comparison, NADE with 500 hidden units, a learning rate of 0.0005 and a decrease constant of 0 obtained -88.86. This is almost as good as the best RBM claim and much better than RBMs trained with just a few steps of Gibbs sampling. Again, it also improves over mixtures of Bernoullis which, with 10, 100 and 500 components obtain -168.95, -142.63 and -137.64 average test log-likelihoods respectively (taken from Salakhutdinov and Murray (2008)). Finally, FVSBN trained by stochastic gradient descent achieves -97.45 and improves on the mixture models but not on NADE.

It then appears that tractability was gained at almost no cost in terms of performance. We are also confident that even better performance could have been achieved with a better optimization method than stochastic gradient descent. Indeed, the log-likelihood on the training

set for NADE was −84.07, which is quite close to its test log-likelihood. Moreover NADE will be able to take advantage of future non-linear optimization methods, currently an active area research (e.g. Martens, 2010). In contrast, training an RBM, for which gradients must be approximated with sampling, is largely restricted to simple optimization methods.

We also generated samples from NADE after training on binarized MNIST. This is done by sequentially sampling each v_i according to $p(v_i|\mathbf{v}_i)$ as estimated by the model. The samples are displayed in Figure 2, along with the probabilities $p(v_i|\mathbf{v}_i)$ used to obtain these samples. While some are noisy, several of them are clear images of a distinguishable digit. Notice that those are **exact** samples under NADE. In contrast, samples from an RBM result from a Markov chain which can mix slowly and whose convergence is hard to establish.

7 Conclusion and Future Work

We have proposed NADE, a new model for estimating the distribution of high-dimensional discrete observations. NADE can be seen as a method for converting an RBM into a tractable distribution estimator. It can also be understood as a special kind of autoencoder whose output assigns valid probabilities to observations and hence is a proper generative model. On several datasets, NADE outperforms other common baselines for tractable distribution estimation and performs similarly to a large intractable RBM.

While this work has focused on binary distributions, one could model other distributions by adjusting the output non-linearity accordingly: a linear output for Gaussian distributions, a so-called softmax output for multinomial distributions or an exponentiated output to yield Exponential or Poisson distributions (Ranzato & Szummer, 2008).

In future work, we would like to investigate the use of NADE on problems other than distribution estimation, in particular on problems for which RBMs and autoencoders are often considered. We give preliminary results in two application areas: initialization of deep neural networks and the unsupervised learning of features.

In Figure 2, we give a visualization of the weights learned by NADE on the binarized version of MNIST. Many of the filters seem to act as edge detectors, which is a sensible feature to extract from images of digits. While the filters seem a bit noisy compared to those learned by an RBM, they are a clear improvement on those learned by a regular autoencoder (Larochelle et al., 2009). We conjecture that the noisiness of the filters is directly linked to having a fixed ordering of the inputs when factoring $p(\mathbf{v})$ into conditionals, which implies that the weights on different pixels receive gradients from a different number of sources (i.e. a different number of conditionals). By using a better optimization procedure taking into account second order information of the optimization objective or by normalizing the scale of the activations going into each $\mathbf{h}_i$ separately, we think it should be possible to improve the quality of the features learned.

The weight matrix $\mathbf{W}$ learned by NADE can also be used as an initial value for the hidden layer weights of a neural network. In preliminary experiments where a variant of NADE (in which a class label is included in the observation vector) was used to initialize a two hidden layer neural network gave around 1.3% classification error on MNIST. We are confident that this result could be further improved in future work.

Acknowledgments

Hugo acknowledges the financial support of NSERC.

Appendix

We give here a short derivation for the mean-field updates of Equations 7 and 8. Equation 6 defines the factorial distribution we wish to use as an approximation for the true marginals. To simplify the derivation, we will extend the definition of $\mu_j(i)$ for $j < i$ by defining that $\mu_j(i) = v_j$ in that special case.

Hence, we wish to minimize the KL-divergence between the approximating distribution $q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})$ of Equation 6 and the true conditional

$$p(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}) = \exp(-E(\mathbf{v}, \mathbf{h}))/Z(\mathbf{v}_{<i}) \qquad (12)$$

where $Z(\mathbf{v}_{<i}) = \sum_{v_i, \mathbf{v}_{>i}, \mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$. First, we can develop the KL-divergence as follows:

$$\begin{aligned}
& \mathrm{KL}(q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})||p(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})) \\
= & -\sum_{v_i, \mathbf{v}_{>i}, \mathbf{h}} q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}) \log p(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}) \\
& + \sum_{v_i, \mathbf{v}_{>i}, \mathbf{h}} q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}) \log q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}) \\
= & -\tau(i)^\top \mathbf{W}\mu(i) - \mathbf{b}^\top \mu(i) - \mathbf{c}^\top \tau(i) + \log Z(\mathbf{v}_{<i}) \\
& + \sum_{j \geq i} (\mu_j(i) \log \mu_j(i) + (1 - \mu_j(i)) \log(1 - \mu_j(i))) \\
& + \sum_k (\tau_k(i) \log \tau_k(i) + (1 - \tau_k(i)) \log(1 - \tau_k(i)))
\end{aligned}$$

Then, we take the derivative with respect to $\tau_k(i)$ and

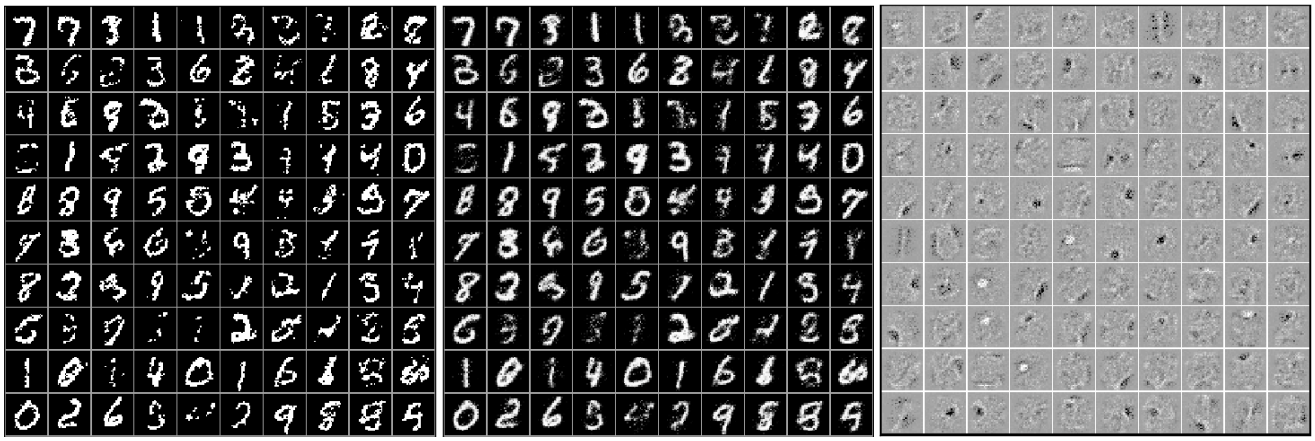

Figure 2: **(Left)**: samples from NADE trained on a binary version of MNIST. **(Middle)**: probabilities from which each pixel was sampled. **(Right)**: visualization of some of the rows of $\mathbf{W}$. This figure is better seen on a computer screen.

set it to 0, to obtain:

$$\begin{aligned}
0 &= \frac{\partial \mathrm{KL}(q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})||p(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}))}{\partial \tau_k(i)} \\
0 &= -c_k - \mathbf{W}_{k,\cdot}\mu(i) + \log\left(\frac{\tau_k(i)}{1-\tau_k(i)}\right) \\
\frac{\tau_k(i)}{1-\tau_k(i)} &= \exp(c_k + \mathbf{W}_{k,\cdot}\mu(i)) \\
\tau_k(i) &= \frac{\exp(c_k + \mathbf{W}_{k,\cdot}\mu(i))}{1+\exp(c_k + \mathbf{W}_{k,\cdot}\mu(i))} \\
\tau_k(i) &= \mathrm{sigm}\left(c_k + \sum_{j\geq i} W_{kj}\mu_j(i) + \sum_{j<i} W_{kj}v_j\right)
\end{aligned}$$

where in the last step we have replaced the matrix/vector multiplication $\mathbf{W}_{k,\cdot}\mu(i)$ by its explicit summation form and have used the fact that $\mu_j(i) = v_j$ for $j < i$.

Similarly, we set the derivative with respect to $\mu_j(i)$ for $j \geq i$ to 0 and obtain:

$$\begin{aligned}
0 &= \frac{\partial \mathrm{KL}(q(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i})||p(v_i, \mathbf{v}_{>i}, \mathbf{h}|\mathbf{v}_{<i}))}{\partial \mu_j(i)} \\
0 &= -b_j - \tau(i)^\top \mathbf{W}_{\cdot,j} + \log\left(\frac{\mu_j(i)}{1-\mu_j(i)}\right) \\
\frac{\mu_j(i)}{1-\mu_j(i)} &= \exp(b_j + \tau(i)^\top \mathbf{W}_{\cdot,j}) \\
\mu_j(i) &= \frac{\exp(b_j + \tau(i)^\top \mathbf{W}_{\cdot,j})}{1+\exp(b_j + \tau(i)^\top \mathbf{W}_{\cdot,j})} \\
\mu_j(i) &= \mathrm{sigm}\left(b_j + \sum_k W_{kj}\tau_k(i)\right)
\end{aligned}$$

We then recover the mean-field updates of Equations 7 and 8.

References

Bengio, Y., & Bengio, S. (2000). Modeling high-dimensional discrete data with multi-layer neural networks. *Advances in Neural Information Processing Systems 12 (NIPS'99)* (pp. 400–406). MIT Press.

Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems 19 (NIPS'06)* (pp. 153–160). MIT Press.

Chen, X. R., Krishnaiah, P. R., & Liang, W. W. (1989). Estimation of multivariate binary density using orthogonal functions. *Journal of Multivariate Analysis*, *31*, 178–186.

Freund, Y., & Haussler, D. (1992). A fast and exact learning rule for a restricted class of Boltzmann machines. *Advances in Neural Information Processing Systems 4 (NIPS'91)* (pp. 912–919). Denver, CO: Morgan Kaufmann, San Mateo.

Frey, B. J. (1998). *Graphical models for machine learning and digital communication*. MIT Press.

Frey, B. J., Hinton, G. E., & Dayan, P. (1996). Does the wake-sleep algorithm learn good density estimators? *Advances in Neural Information Processing Systems 8 (NIPS'95)* (pp. 661–670). MIT Press, Cambridge, MA.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, *14*, 1771–1800.

Hinton, G. E., Osindero, S., & Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, *18*, 1527–1554.

Larochelle, H., & Bengio, Y. (2008). Classification using discriminative restricted Boltzmann machines. *Proceedings of the 25th Annual International Conference*

on Machine Learning (ICML 2008) (pp. 536–543). Helsinki, Finland: Omnipress.

Larochelle, H., Bengio, Y., Louradour, J., & Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, *10*, 1–40.

Larochelle, H., Bengio, Y., & Turian, J. (2010). Tractable multivariate binary density estimation and the restricted Boltzmann forest. *Neural Computation*, *22*, 2285–2307.

Li, N., & Stephens, M. (2003). Modeling linkage disequilibrium and identifying recombination hotspots using single-nucleotide polymorphism data. *Genetics*, 2213–2233.

Lowd, D., & Domingos, P. (2005). Naive Bayes models for probability estimation. *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)* (pp. 529–536). New York, USA: ACM.

Martens, J. (2010). Deep learning via Hessian-free optimization. *Proceedings of the 27th International Conference on Machine Learning (ICML 2010)* (pp. 735–742). Haifa, Israel: Omnipress.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

Ott, J., & Kronmal, R. A. (1976). Some classification procedures for multivariate binary data using orthogonal functions. *Journal of the American Statistical Association*, *71*, 391–399.

Ranzato, M., & Szummer, M. (2008). Semi-supervised learning of compact document representations with deep networks. *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)* (pp. 792–799). Helsinki, Finland: ACM.

Salakhutdinov, R., & Hinton, G. E. (2009). Deep Boltzmann machines. *Proceedings of The Twelfth International Conference on Artificial Intelligence and Statistics (AISTATS'09)* (pp. 448–455). Clearwater (Florida), USA.

Salakhutdinov, R., & Larochelle, H. (2010). Efficient learning of deep Boltzmann machines. *Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS'10)* (pp. 693–700). Sardinia, Italy.

Salakhutdinov, R., & Murray, I. (2008). On the quantitative analysis of deep belief networks. *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 208)* (pp. 872–879). Helsinki, Finland: Omnipress.

Schmah, T., Hinton, G. E., Zemel, R. S., Small, S. L., & Strother, S. C. (2009). Generative versus discriminative training of RBMs for classification of fMRI images. *Advances in Neural Information Processing Systems 21* (pp. 1409–1416).

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland (Eds.), *Parallel distributed processing*, vol. 1, chapter 6, 194–281. Cambridge: MIT Press.

Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)* (pp. 1064–1071). Helsinki, Finland: Omnipress.

Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)* (pp. 1096–1103). ACM.

Welling, M., & Hinton, G. E. (2002). A new learning algorithm for mean field Boltzmann machines. *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks* (pp. 351–357). London, UK: Springer-Verlag.