Deep Generative Stochastic Networks Trainable by Backprop

Yoshua Bengio∗
Éric Thibodeau-Laufer
Guillaume Alain
Département d’informatique et recherche opérationnelle, Université de Montréal,* & Canadian Inst. for Advanced Research
Jason Yosinski
Department of Computer Science, Cornell University

Abstract
We introduce a novel training principle for probabilistic models that is an alternative to maximum likelihood. The proposed Generative Stochastic Networks (GSN) framework is based on learning the transition operator of a Markov chain whose stationary distribution estimates the data distribution. The transition distribution of the Markov chain is conditional on the previous state, generally involving a small move, so this conditional distribution has fewer dominant modes, being unimodal in the limit of small moves. Thus, it is easier to learn because it is easier to approximate its partition function, more like learning to perform supervised function approximation, with gradients that can be obtained by backprop. We provide theorems that generalize recent work on the probabilistic interpretation of denoising autoencoders and obtain along the way an interesting justification for dependency networks and generalized pseudolikelihood, along with a definition of an appropriate joint distribution and sampling mechanism even when the conditionals are not consistent. GSNs can be used with missing inputs and can be used to sample subsets of variables given the rest. We validate these theoretical results with experiments on two image datasets using an architecture that mimics the Deep Boltzmann Machine Gibbs sampler but allows training to proceed with simple backprop, without the need for layerwise pretraining.

1. Introduction
Research in deep learning (see Bengio (2009) and Bengio et al. (2013a) for reviews) grew from breakthroughs in unsupervised learning of representations, based mostly on the Restricted Boltzmann Machine (RBM) (Hinton et al., 2006), auto-encoder variants (Bengio et al., 2007; Proceedings of the 31st International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).)

Figure 1. Top: A denoising auto-encoder defines an estimated Markov chain where the transition operator first samples a corrupted $\tilde{X}$ from $C(\tilde{X}|X)$ and then samples a reconstruction from $P_\theta(X|\tilde{X})$, which is trained to estimate the ground truth $P(X|\tilde{X})$. Note how for any given $\tilde{X}$, $P(X|\tilde{X})$ is a much simpler (roughly unimodal) distribution than the ground truth $P(X)$ and its partition function is thus easier to approximate.

Bottom: More generally, a GSN allows the use of arbitrary latent variables $H$ in addition to $X$, with the Markov chain state (and mixing) involving both $X$ and $H$. Here $H$ is the angle about the origin. The GSN inherits the benefit of a simpler conditional and adds latent variables, which allow far more powerful deep representations in which mixing is easier (Bengio et al., 2013b).
Intuition: Domain of the visible variables (Section 3.1). May be purely local, not requiring support over the whole.

Strengthen the consistency theorems introduced in Bengio et al. (2013c) by introducing latent variables in the framework to define Generative Stochastic Networks (GSNs) (Section 3). GSNs aim to estimate the data generating distribution indirectly, by parametrizing the transition operator of a Markov chain rather than directly parametrizing \( P(X) \). Most critically, this framework transforms the unsupervised density estimation problem into one which is more similar to supervised function approximation and difficult to approximate because of the large number of latent variables (Section 3.4).

In all of these cases, what is intractable is the computation or approximation of a sum (often weighted by probabilities), such as a marginalization or the estimation of the gradient of the normalization constant. If only a few terms in this sum dominate (corresponding to the dominant modes of the distribution), then many good approximate methods can be found, such as Monte-Carlo Markov chains (MCMC) methods.

Similarly difficult tasks arise with structured output problems where one wants to sample from \( P(y,h|x) \) and both \( y \) and \( h \) are high-dimensional and have a complex highly multimodal joint distribution (given \( x \)).

Deep Boltzmann machines (Salakhutdinov & Hinton, 2009) combine the difficulty of inference (for the positive phase where one tries to push the energies associated with the observed \( x \) down) and also that of sampling (for the negative phase where one tries to push up the energies associated with \( x \)'s sampled from \( P(x) \)). Unfortunately, using an MCMC method to sample from \( P(x,h) \) in order to estimate the gradient of the partition function may be seriously hurt by the presence of a large number of

Example application: In Section 4 we show an example application of the GSN theory to create a deep GSN whose computational graph resembles the one followed by Gibbs sampling in deep Boltzmann machines (with continuous latent variables), but that can be trained efficiently with back-propagated gradients and without layerwise pre-training. Because the Markov Chain is defined over a state \( (X, h) \) that includes latent variables, we reap the dual advantage of more powerful models for a given number of parameters and better mixing in the chain as we add noise to variables representing higher-level information, first suggested by the results obtained by Bengio et al. (2013b) and Luo et al. (2013). The experimental results show that such a model with latent states indeed mixes better than shallower models without them (Table 1).

Dependency networks: Finally, an unexpected result falls out of the GSN theory: it allows us to provide a novel justification for dependency networks (Heckerman et al., 2000) and for the first time define a proper joint distribution between all the visible variables that is learned by such models (Section 3.4).
important modes, as argued below.

To evade the problem of highly multimodal joint or posterior distributions, the currently known approaches to dealing with the above intractable sums make very strong explicit assumptions (in the parametrization) or implicit assumptions (by the choice of approximation methods) on the form of the distribution of interest. In particular, MCMC methods are more likely to produce a good estimator if the number of non-negligible modes is small: otherwise the chains would require at least as many MCMC steps as the number of such important modes, times a factor that accounts for the mixing time between modes. Mixing time itself can be very problematic as a trained model becomes sharper, as it approaches a data generating distribution that may have well-separated and sharp modes (i.e., manifolds).

We propose to make another assumption that might suffice to bypass this multimodality problem: the effectiveness of function approximation.

In particular, the GSN approach presented in the next section relies on estimating the transition operator of a Markov chain, e.g. \( P(x_t|x_{t-1}) \) or \( P(x_t,h_t|x_{t-1},h_{t-1}) \). Because each step of the Markov chain is generally local, these transition distributions will often include only a very small number of important modes (those in the neighbourhood of the previous state). Hence the gradient of their partition function will be easy to approximate. For example consider the denoising transitions studied by Bengio et al. (2013c) and illustrated in Figure 1, where \( \tilde{x}_{t-1} \) is a stochastically corrupted version of \( x_{t-1} \) and we learn the denoising distribution \( P(x|\tilde{x}) \). In the extreme case (studied empirically here) where \( P(x|\tilde{x}) \) is approximated by a unimodal distribution, the only form of training that is required involves function approximation (predicting the clean \( x \) from the corrupted \( \tilde{x} \)).

Although having the true \( P(x|\tilde{x}) \) turn out to be unimodal makes it easier to find an appropriate family of models for it, unimodality is by no means required by the GSN framework itself. One may construct a GSN using any multimodal model for output (e.g. mixture of Gaussians, RBMs, NADE, etc.), provided that gradients for the parameters of the model in question can be estimated (e.g. log-likelihood gradients).

The approach proposed here thus avoids the need for a poor approximation of the gradient of the partition function in the inner loop of training, but still has the potential of capturing very rich distributions by relying mostly on “function approximation”.

Besides the approach discussed here, there may well be other very different ways of evading this problem of intractable marginalization, including approaches such as sum-product networks (Poon & Domingos, 2011), which are based on learning a probability function that has a tractable form by construction and yet is from a flexible enough family of distributions.

3. Generative Stochastic Networks

Assume the problem we face is to construct a model for some unknown data-generating distribution \( P(X) \) given only examples of \( X \) drawn from that distribution. In many cases, the unknown distribution \( P(X) \) is complicated, and modeling it directly can be difficult.

A recently proposed approach using denoising auto-encoders transforms the difficult task of modeling \( P(X) \) into a supervised learning problem that may be much easier to solve. The basic approach is as follows: given a clean example data point \( X \) from \( P(X) \), we obtain a corrupted version \( \tilde{X} \) by sampling from some corruption distribution \( C(\tilde{X}|X) \). For example, we might take a clean image, \( X \), and add random white noise to produce \( \tilde{X} \). We then use supervised learning methods to train a function to reconstruct, as accurately as possible, any \( X \) from the data set given only a noisy version \( \tilde{X} \). As shown in Figure 1, the reconstruction distribution \( P(X|\tilde{X}) \) may often be much easier to learn than the data distribution \( P(X) \), because \( P(X|\tilde{X}) \) tends to be dominated by a single or few major modes (such as the roughly Gaussian shaped density in the figure).

But how does learning the reconstruction distribution help us solve our original problem of modeling \( P(X) \)? The two problems are clearly related, because if we knew everything about \( P(X) \), then our knowledge of the \( C(\tilde{X}|X) \) that we chose would allow us to precisely specify the optimal reconstruction function via Bayes rule: \( P(X|\tilde{X}) = \frac{1}{z} C(\tilde{X}|X) P(X) \), where \( z \) is a normalizing constant that does not depend on \( X \). As one might hope, the relation is also true in the opposite direction: once we pick a method of adding noise, \( C(\tilde{X}|X) \), knowledge of the corresponding reconstruction distribution \( P(X|\tilde{X}) \) is sufficient to recover the density of the data \( P(X) \).

This intuition was borne out by proofs in two recent papers. Alain & Bengio (2013) showed that denoising auto-encoders with small Gaussian corruption and squared error loss estimated the score (derivative of the log-density with respect to the input) of continuous observed random variables. More recently, Bengio et al. (2013c) generalized this to arbitrary variables (discrete, continuous or both), arbitrary corruption (not necessarily asymptotically small), and arbitrary loss function (so long as they can be seen as a log-likelihood).

Beyond proving that \( P(X|\tilde{X}) \) is sufficient to reconstruct the data density, Bengio et al. (2013c) also demonstrated a method of sampling from a learned, parametrized model of the density, \( P_\theta(\tilde{X}) \), by running a Markov chain that alternately adds noise using \( C(\tilde{X}|X) \) and denoises by sampling from the learned \( P_\theta(X|\tilde{X}) \), which is trained to approximate the true \( P(X|\tilde{X}) \). The most important contribution of that paper was demonstrating that if a learned, parametrized reconstruction function \( P_\theta(X|\tilde{X}) \) converges to the true \( P(X|\tilde{X}) \), then under some relatively benign conditions the stationary distribution \( \pi(X) \) of the resulting Markov chain will exist and will indeed converge.
to the data distribution $P(X)$.

Before moving on, we should pause to make an important point clear. Alert readers may have noticed that $P(X|X)$ and $P(X)$ can each be used to reconstruct the other given knowledge of $C(X|X)$. Further, if we assume that we have chosen a simple $C(X|X)$ (say, a uniform Gaussian with a single width parameter), then $P(X|X)$ and $P(X)$ must both be of approximately the same complexity. Put another way, we can never hope to combine a simple $C(X|X)$ and a simple $P(X|X)$ to model a complex $P(X)$. Nonetheless, it may still be the case that $P(X|X)$ is easier to model than $P(X)$ due to reduced computational complexity in computing or approximating the partition functions of the conditional distribution mapping corrupted input $\tilde{X}$ to the distribution of corresponding clean input $X$. Indeed, because that conditional is going to be mostly assigning probability to $X$ locally around $\tilde{X}$, $P(X|X)$ has only one or a few modes, while $P(X)$ can have a very large number.

So where did the complexity go? $P(X|\tilde{X})$ has fewer modes than $P(X)$, but the location of these modes depends on the value of $\tilde{X}$. It is precisely this mapping from $\tilde{X} \rightarrow \text{mode location}$ that allows us to trade a difficult density modeling problem for a supervised function approximation problem that admits application of many of the usual supervised learning tricks.

In the next four sections, we extend previous results in several directions.

### 3.1. Generative denoising autoencoders with local noise

The main theorem in Bengio et al. (2013c) (stated in supplemental as Theorem S1) requires that the Markov chain be ergodic. A set of conditions guaranteeing ergodicity is given in the aforementioned paper, but these conditions are restrictive in requiring that $C(X|X) > 0$ everywhere that $P(X) > 0$. The effect of these restrictions is that $P_0(X|X)$ must have the capacity to model every mode of $P(X)$, exactly the difficulty we were trying to avoid. We show in this paper’s supplemental material how we may also achieve the required ergodicity through other means, allowing us to choose a $C(X|X)$ that only makes small jumps, which in turn only requires $P_0(X|\tilde{X})$ to model a small part of the space around each $\tilde{X}$.

### 3.2. Generalizing the denoising autoencoder to GSNs

The denoising auto-encoder Markov chain is defined by $X_t \sim C(X|X_t)$ and $X_{t+1} \sim P_0(X|\tilde{X}_t)$, where $\tilde{X}_t$ alone can serve as the state of the chain. The GSN framework generalizes this by defining a Markov chain with both a visible $X_t$ and a latent variable $H_t$ as state variables, of the form

\[
\begin{align*}
H_{t+1} &\sim P_0(H|H_t, X_t) \\
X_{t+1} &\sim P_0(X|H_{t+1}).
\end{align*}
\]

Denoising auto-encoders are thus a special case of GSNs. Note that, given that the distribution of $H_{t+1}$ depends on a previous value of $H_t$, we find ourselves with an extra $H_0$ variable added at the beginning of the chain. This $H_0$ complicates things when it comes to training, but when we are in a sampling regime we can simply wait a sufficient number of steps to burn in.

The next theoretical results give conditions for making the stationary distributions of the above Markov chain match a target data generating distribution.

**Theorem 2.** Let $(H_t, X_t)_{t=0}^{\infty}$ be the Markov chain defined by the following graphical model.

If we assume that the chain has a stationary distribution $\pi_{H,X}$, and that for every value of $(x, h)$ we have that

- all the $P(X_t = x|H_t = h) = g(x, h)$ share the same density for $t \geq 1$
- all the $P(H_{t+1} = h|H_t = h', X_t = x) = f(h, h', x)$ share the same density for $t \geq 0$
- $P(H_0 = h|X_0 = x) = P(H_1 = h|X_0 = x)$
- $P(X_1 = x|H_1 = h) = P(X_0 = x|H_1 = h)$

then for every value of $(x, h)$ we get that

- $P(X_0 = x|H_0 = h) = g(x, h)$ holds, which is something that was assumed only for $t \geq 1$
- $P(X_t = x, H_t = h) = P(X_0 = x, H_0 = h)$ for all $t \geq 0$
- the stationary distribution $\pi_{H,X}$ has a marginal distribution $\pi_X$ such that $\pi(x) = P(X_0 = x)$.

Those conclusions show that our Markov chain has the property that its samples in $X$ are drawn from the same distribution as $X_0$.

The proof is given in this paper’s supplemental material. We also address there the issue of the consistency of the stationary distribution that we obtain when using an increasing, but finite, number of training samples.

We avoid discussing the training criterion for a GSN. Various alternatives exist, but this analysis is for future work. Right now Theorem 2 suggests the following rules:

- Pick the transition distribution $f(h, h', x)$ to be useful (e.g. through training that maximizes reconstruction likelihood).
- Make sure that during training $P(H_0 = h|X_0 = x) \rightarrow P(H_1 = h|X_0 = x)$. One interesting way to achieve this is, for each $X_0$ in the training set, iteratively sample
which would visually look like a fuzzy combination of actual modes. In the experiments performed here, we have only considered unimodal reconstruction distributions (with factorized outputs), because we expect that even if $P(X|H)$ is not unimodal, it would be dominated by a single mode when the noise level is small. However, future work should investigate multimodal alternatives.

A related element to keep in mind is that one should pick the family of conditional distributions $P_{b_2}(X|H)$ so that one can sample from them and one can easily train them when given $(X, H)$ pairs, e.g., by maximum likelihood.

3.3. Handling missing inputs or structured output

In general, a simple way to deal with missing inputs is to clamp the observed inputs and then apply the Markov chain with the constraint that the observed inputs are fixed and not resampled at each time step, whereas the unobserved inputs are resampled each time, conditioned on the clamped inputs. As proved in this paper’s supplementary material, this procedure gives rise to sampling from the appropriate conditional distribution:

**Proposition 1.** If a subset $x^{(s)}$ of the elements of $X$ is kept fixed (not resampled) while the remainder $X^{(-s)}$ is updated stochastically during the Markov chain of Theorem 2, but using $P(X_t|H_t, X^{(s)}_t = x^{(s)})$, then the asymptotic distribution $\pi_n$ of the Markov chain produces samples of $X^{(-s)}$ from the conditional distribution $\pi_n(X^{(-s)}|X^{(s)} = x^{(s)})$.

Practically, it means that we must choose an output (reconstruction) distribution from which it is not only easy to sample from, but also from which it is easy to sample a subset of the variables in the vector $X$ conditioned on the rest being known. In the experiments below, we used a factorial distribution for the reconstruction, from which it is trivial to sample conditionally a subset of the input variables. In general (with non-factorial output distributions) one must use the proper conditional for the theorem to apply, i.e., it is not sufficient to clamp the inputs, one must also sample the reconstructions from the appropriate conditional distribution (conditioning on the clamped values).

This method of dealing with missing inputs can be immediately applied to structured outputs. If $X^{(s)}$ is viewed as an “input” and $X^{(-s)}$ as an “output”, then sampling from $X^{(-s)}_{t+1} \sim P(X^{(-s)}|f((X^{(s)}_t, X^{(-s)}_t), Z_t, H_t), X^{(s)}_t)$ will converge to estimators of $P(X^{(-s)}|X^{(s)}_t)$. This still requires good choices of the parameterization (for $f$ as well as for the conditional probability $P$), but the advantages of this approach are that there is no approximate inference of latent variables and the learner is trained with respect to simpler conditional probabilities: in the limit of small noise, we conjecture that these conditional probabilities can be well approximated by unimodal distributions.

Theoretical evidence comes from Alain & Bengio (2013): when the amount of corruption noise converges to 0 and the input variables have a smooth continuous density, then a unimodal Gaussian reconstruction density suffices to fully capture the joint distribution.
3.4. Dependency Networks as GSNs

Dependency networks (Heckerman et al., 2000) are models in which one estimates conditionals $P_t(x_t|x_{-t})$, where $x_{-i}$ denotes $x \setminus x_i$, i.e., the set of variables other than the $i$-th one, $x_i$. Note that each $P_t$ may be parametrized separately, thus not guaranteeing that there exists a joint of which they are the conditionals. Instead of the ordered pseudo-Gibbs sampler defined in Heckerman et al. (2000), which resamples each variable $x_i$ in the order $x_1, x_2, \ldots$, we can view dependency networks in the GSN framework by defining a proper Markov chain in which at each step one randomly chooses which variable to resample. The corruption process therefore just consists of $H = f(X, Z) = X_{-s}$ where $X_{-s}$ is the complement of $X_s$, with $s$ a randomly chosen subset of elements of $X$ (possibly constrained to be of size 1). Furthermore, we parametrize the reconstruction distribution as $P_{y_t}(X = x|H) = \delta_{x_{-s}=X_{-s}}P_{y_{s,t}}(X_s = x_s|x_{-s})$ where the estimated conditionals $P_{y_{s,t}}(X_s = x_s|x_{-s})$ are not constrained to be consistent conditionals of some joint distribution over all of $X$.

**Proposition 2.** If the above GSN Markov chain has a stationary distribution, then the dependency network defines a joint distribution (which is that stationary distribution), which does not have to be known in closed form. Furthermore, if the conditionals are consistent estimators of the ground truth conditionals, then that stationary distribution is a consistent estimator of the ground truth joint distribution.

The proposition can be proven by immediate application of Theorem 1 from Bengio et al. (2013c) with the above definitions of the GSN. This joint stationary distribution can exist even if the conditionals are not consistent. To show that, assume that some choice of (possibly inconsistent) conditionals gives rise to a stationary distribution $\pi$. Now let us consider the set of all conditionals (not necessarily consistent) that could have given rise to that $\pi$. Clearly, the conditionals derived from $\pi$ is part of that set, but there are infinitely many others (a simple counting argument shows that the fixed point equation of $\pi$ introduces fewer constraints than the number of degrees of freedom that define the conditionals). To better understand why the ordered pseudo-Gibbs chain does not benefit from the same properties, we can consider an extended case by adding an extra component of the state $X$, being the index of the next variable to resample. In that case, the Markov chain associated with the ordered pseudo-Gibbs procedure would be periodic, thus violating the ergodicity assumption of the theorem. However, by introducing randomness in the choice of which variable(s) to resample next, we obtain aperiodicity and ergodicity, yielding as stationary distribution a mixture over all possible resampling orders. These results also show in a novel way (see e.g. Hyvärinen (2006) for earlier results) that training by pseudolikelihood or generalized pseudolikelihood provides a consistent estimator of the associated joint, so long as the GSN Markov chain defined above is ergodic. This result can be applied to show that the multi-prediction deep Boltzmann machine (MP-DBM) training procedure introduced by Goodfellow et al. (2013) also corresponds to a GSN. This has been exploited in order to obtain much better samples using the associated GSN Markov chain than by sampling from the corresponding DBM (Goodfellow et al., 2013). Another interesting conclusion that one can draw from this paper and its GSN interpretation is that state-of-the-art classification error can thereby be obtained: 0.91% on MNIST without fine-tuning (best comparable previous DBM results was well above 1%) and 10.6% on permutation-invariant NORB (best previous DBM results was 10.8%).

4. Experimental Example of GSN

The theoretical results on Generative Stochastic Networks (GSNs) open for exploration a large class of possible parametrizations which will share the property that they can capture the underlying data distribution through the GSN Markov chain. What parametrizations will work well? Where and how should one inject noise? We present results of preliminary experiments with specific selections for each of these choices, but the reader should keep in mind that the space of possibilities is vast.
As a conservative starting point, we propose to explore families of parametrizations which are similar to existing deep stochastic architectures such as the Deep Boltzmann Machine (DBM) (Salakhutdinov & Hinton, 2009). Basically, the idea is to construct a computational graph that is similar to the computational graph for Gibbs sampling or variational inference in Deep Boltzmann Machines. However, we have to diverge a bit from these architectures in order to accommodate the desirable property that it will be possible to back-propagate the gradient of reconstruction log-likelihood with respect to the parameters $\theta_1$ and $\theta_2$. Since the gradient of a binary stochastic unit is 0 almost everywhere, we have to consider related alternatives. An interesting source of inspiration regarding this question is a recent paper on estimating or propagating gradients through stochastic neurons (Bengio, 2013).

Here we consider the following stochastic non-linearities:

$$ h_i = \eta_{\text{out}} + \tanh(\eta_{\text{in}} + a_i) $$

where $a_i$ is the linear activation for unit $i$ (an affine transformation applied to the input of the unit, coming from the layer below, the layer above, or both) and $\eta_{\text{in}}$ and $\eta_{\text{out}}$ are zero-mean Gaussian noises.

To emulate a sampling procedure similar to Boltzmann machines in which the filled-in missing values can depend on the representations at the top level, the computational graph allows information to propagate both upwards (from input to higher levels) and downwards, giving rise to the computational graph structure illustrated in Figure 2, which is similar to that explored for deterministic recurrent auto-encoders (Seung, 1998; Behnke, 2001; Savard, 2011). Downward weight matrices have been fixed to the transpose of corresponding upward weight matrices.

The walkback algorithm was proposed in Bengio et al. (2013c) to make training of generalized denoising auto-encoders (a special case of the models studied here) more efficient. The basic idea is that the reconstruction is obtained after not one but several steps of the sampling Markov chain. In this context it simply means that the computational graph from $X$ to a reconstruction probability actually involves generating intermediate samples as if we were running the Markov chain starting at $X$. In the experiments, the graph was unfolded so that $2D$ sampled reconstructions would be produced, where $D$ is the depth (number of hidden layers). The training loss is the sum of the reconstruction negative log-likelihoods (of target $X$) over all those reconstruction steps.

The supplemental material provides full details on the experiments and more detailed figures of generated samples. We summarize the results here. The experiments were performed on the MNIST and Toronto Face Database (TFD) (Susskind et al., 2010) datasets, following the setup in Bengio et al. (2013b), where the model generates quantized (binary) pixels. A lower bound on the log-likelihood, based only on the generated samples (or rather the conditional expectations $E[X|H]$ for the sampled $H$’s) and introduced in Breuleux et al. (2011) was used to compare various models in Table 1. The results show that a GSN with latent state performed better than a pure denoising auto-encoder (or equivalently, that a deeper GSN yields both better samples and a better likelihood bound). As can be seen, the samples are of quality comparable to those obtained by Deep Boltzmann Machines (DBMs) and Deep Belief Nets (DBNs). Figures 3 and 4 illustrate generated samples and show the fast mixing. Figure 3 (bottom) also shows successful conditional sampling of the left hand side of the image given the right hand side.

5. Conclusion

We have introduced a new approach to training generative models, called Generative Stochastic Networks (GSN), that is an alternative to maximum Stochastic likelihood, with the objective of avoiding the intractable marginalizations and
Deep Generative Stochastic Networks Trainable by Backprop

Figure 4. GSN samples from a 3-layer model trained on the TFD dataset. Every second sample is shown; see supplemental material for every sample. At the end of each row, we show the nearest example from the training set to the last sample on that row, to illustrate that the distribution is not merely copying the training set.

Table 1. Test set log-likelihood lower bound (LL) obtained by a Parzen density estimator constructed using 10000 generated samples, for different generative models trained on MNIST. The LL is not directly comparable to AIS likelihood estimates because we use a Gaussian mixture rather than a Bernoulli mixture to compute the likelihood, but we can compare with Rifai et al. (2012); Bengio et al. (2013b; c) (from which we took the last three columns). A DBN-2 has 2 hidden layers, a CAE-1 has 1 hidden layer, and a CAE-2 has 2. The DAE is basically a GSN-1, with no injection of noise inside the network.

<table>
<thead>
<tr>
<th></th>
<th>GSN-2</th>
<th>DAE</th>
<th>DBN-2</th>
<th>CAE-1</th>
<th>CAE-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>LL</td>
<td>214</td>
<td>144</td>
<td>138</td>
<td>68</td>
<td>121</td>
</tr>
<tr>
<td>STD. ERR.</td>
<td>1.1</td>
<td>1.6</td>
<td>2.0</td>
<td>2.9</td>
<td>1.6</td>
</tr>
</tbody>
</table>

the danger of poor approximations of these marginalizations. The training procedure is more similar to function approximation than to unsupervised learning because the reconstruction distribution is simpler than the data distribution, often unimodal (provably so in the limit of very small noise). This makes it possible to train unsupervised models that capture the data-generating distribution simply using back-prop and gradient descent (in a computational graph that includes noise injection). The proposed theoretical results state that under mild conditions (in particular that the noise injected in the networks prevents perfect reconstruction), training the model to denoise and reconstruct its observations (through a powerful family of reconstruction distributions) suffices to capture the data-generating distribution through a simple Markov chain. Another way to put it is that we are training the transition operator of a Markov chain whose stationary distribution estimates the data distribution, and it turns out that this is a much easier learning problem because the normalization constant for this conditional distribution is generally dominated by fewer modes. These theoretical results are extended to the case where the corruption is local but still allows the chain to mix and to the case where some inputs are missing or constrained (thus allowing to sample from a conditional distribution on a subset of the observed variables or to learned structured output models). The GSN framework is shown to lend to dependency networks a valid estimator of the joint distribution of the observed variables even when the learned conditionals are not consistent, also allowing to prove consistency of generalized pseudolikelihood training, associated with the stationary distribution of the corresponding GSN (that randomly chooses a subset of variables and then resamples it). Experiments have been conducted to validate the theory, in the case where the GSN architecture emulates the Gibbs sampling process of a Deep Boltzmann Machine, on two datasets. A quantitative evaluation of the samples confirms that the training procedure works very well (in this case allowing us to train a deep generative model without layerwise pretraining) and can be used to perform conditional sampling of a subset of variables given the rest.

Acknowledgements

The authors would like to acknowledge the stimulating discussions and help from Vincent Dumoulin, Pascal Vincent, Yao Li, Aaron Courville, Ian Goodfellow, and Hod Lipson, as well as funding from NSERC, CIFAR (YB is a CIFAR Senior Fellow), NASA (JY is a Space Technology Research Fellow), and the Canada Research Chairs.

References


