Three Deep Generative Models

4/8/16

Reviewed by: Jack Lanchantin
Department of Computer Science, University of Virginia

https://qdata.github.io/deep2Read/
Generalized Denoising Auto-Encoders as Generative Models
(Bengio et. al. - NIPS 2013)
**Question:** Do denoising auto-encoders completely characterize the input distribution or only some aspect of it?

- Clustering algorithms only capture the modes of the distribution, while manifold learning algorithms characterize the low-dimensional regions where the density concentrates.

**Propose:** Different probabilistic interpretation of DAEs, which is valid for any data type.
Overview:

The basic idea is that if we corrupt observed random variable $X$ into $\tilde{X}$ using conditional distribution $C(\tilde{X}|X)$, we are really training the DAE to estimate the reverse conditional $P(X|\tilde{X})$. Combining this estimator with the known $C(\tilde{X}|X)$, we show that we can recover a consistent estimator of $P(X)$ through a Markov chain that alternates between sampling from $P(X|\tilde{X})$ and sampling from $C(\tilde{X}|X)$, i.e., encode/decode, sample from the reconstruction distribution model $P(X|\tilde{X})$, apply the stochastic corruption procedure $C(\tilde{X}|X)$, and iterate.
Training:

Algorithm 1 The generalized denoising auto-encoder training algorithm requires a training set or training distribution $\mathcal{D}$ of examples $X$, a given corruption process $\mathcal{C}(\tilde{X}|X)$ from which one can sample, and with which one trains a conditional distribution $P_\theta(X|\tilde{X})$ from which one can sample.

repeat
  • sample training example $X \sim \mathcal{D}$
  • sample corrupted input $\tilde{X} \sim \mathcal{C}(\tilde{X}|X)$
  • use $(X, \tilde{X})$ as an additional training example towards minimizing the expected value of $-\log P_\theta(X|\tilde{X})$, e.g., by a gradient step with respect to $\theta$.
until convergence of training (e.g., as measured by early stopping on out-of-sample negative log-likelihood)
Sampling:

We define the following pseudo-Gibbs Markov chain associated with $P_\theta$:

$$X_t \sim P_\theta(X|\tilde{X}_{t-1})$$
$$\tilde{X}_t \sim \mathcal{C}(\tilde{X}|X_t)$$

(3)

which can be initialized from an arbitrary choice $X_0$. This is the process by which we are going to generate samples $X_t$ according to the model implicitly learned by choosing $\theta$. We define $T(X_t|X_{t-1})$ the transition operator that defines a conditional distribution for $X_t$ given $X_{t-1}$, independently of $t$, so that the sequence of $X_t$’s forms a homogeneous Markov chain. If the asymptotic marginal distribution of the $X_t$’s exists, we call this distribution $\pi(X)$, and we show below that it consistently estimates $\mathcal{P}(X)$. 
Figure 4: Successive samples generated by Markov chain associated with the trained DAEs according to the plain sampling scheme (left) and walkback sampling scheme (right). There are less “spurious” samples with the walkback algorithm.
Conclusion:

Training a model to denoise is a way to implicitly estimate the underlying data generating process, and that a simple Markov chain that alternates sampling from the denoising model and from the corruption process converges to that estimator. This provides a means for generating data from any DAE.
The Neural Autoregressive Distribution Estimator
(Larochelle and Murray - AISTATS 2011)
RBM Inference: If the observations can be decomposed into an input $x$ and a target $y$, then an RBM trained on such pairs can also be used to predict a new $x$. 

The Neural Autoregressive Distribution Estimator (Larochelle and Murray - AISTATS 2011)
Problem: RBM isn’t suited for estimating the joint probability of a given observation (due to partition function being intractable)

Solution: Convert RBM into Bayesian Network

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

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 Neural Autoregressive Distribution Estimator  
( Larochelle and Murray - AISTATS 2011)

\[ p(\mathbf{v}) = \prod_{i=1}^{D} p(v_i | \mathbf{v}_{<i}) \]

\[ = \prod_{i=1}^{D} \frac{p(v_i, \mathbf{v}_{<i})}{p(\mathbf{v}_{<i})} \]

\[ = \prod_{i=1}^{D} \frac{\sum_{\mathbf{v} > i} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))}{\sum_{\mathbf{v}_j \geq i} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))} \],

Intractable  →  Approximate
The Neural Autoregressive Distribution Estimator
(Larochelle and Murray - AISTATS 2011)

\[ p(v_i = 1|v_{<i}) = \text{sigm} \left( b_i + (W^\top)_i . h_i \right) \]

\[ h_i = \text{sigm} \left( c + W . v_{<i} \right), \]

**Training** is done by minimizing the average negative log-likelihood of the parameters given the training set:

\[
\frac{1}{T} \sum_{t=1}^{T} - \log p(v_t) = \frac{1}{T} \sum_{t=1}^{T} \sum_{i=1}^{D} - \log p(v_i|v_{<i}), \quad (11)
\]
Conclusion:

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.
Generative Adversarial Networks
(Goodfellow et. al. - NIPS 2014)
Generative Adversarial Networks
(Goodfellow et. al. - NIPS 2014)

Overview:

Framework for estimating generative models via an adversarial process

Simultaneously train two models:
(1) generative model $G$ that captures the data distribution,
(2) discriminative model $D$ that estimates the probability that a sample came from the training data rather than $G$
Method:

The adversarial modeling framework is most straightforward to apply when the models are both multilayer perceptrons. To learn the generator’s distribution $p_g$ over data $x$, we define a prior on input noise variables $p_z(z)$, then represent a mapping to data space as $G(z; \theta_g)$, where $G$ is a differentiable function represented by a multilayer perceptron with parameters $\theta_g$. We also define a second multilayer perceptron $D(x; \theta_d)$ that outputs a single scalar. $D(x)$ represents the probability that $x$ came from the data rather than $p_g$. We train $D$ to maximize the probability of assigning the correct label to both training examples and samples from $G$. We simultaneously train $G$ to minimize $\log(1 - D(G(z)))$: 
Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing.
## Summary of Prev 3 Models

<table>
<thead>
<tr>
<th></th>
<th>Deep directed graphical models</th>
<th>Deep undirected graphical models</th>
<th>Generative autoencoders</th>
<th>Adversarial models</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Training</strong></td>
<td>Inference needed during training.</td>
<td>Inference needed during training. MCMC needed to approximate partition function gradient.</td>
<td>Enforced tradeoff between mixing and power of reconstruction generation</td>
<td>Synchronizing the discriminator with the generator. Helvetica.</td>
</tr>
<tr>
<td><strong>Inference</strong></td>
<td>Learned approximate inference</td>
<td>Variational inference</td>
<td>MCMC-based inference</td>
<td>Learned approximate inference</td>
</tr>
<tr>
<td><strong>Sampling</strong></td>
<td>No difficulties</td>
<td>Requires Markov chain</td>
<td>Requires Markov chain</td>
<td>No difficulties</td>
</tr>
<tr>
<td><strong>Evaluating $p(x)$</strong></td>
<td>Intractable, may be approximated with AIS</td>
<td>Intractable, may be approximated with AIS</td>
<td>Not explicitly represented, may be approximated with Parzen density estimation</td>
<td>Not explicitly represented, may be approximated with Parzen density estimation</td>
</tr>
<tr>
<td><strong>Model design</strong></td>
<td>Nearly all models incur extreme difficulty</td>
<td>Careful design needed to ensure multiple properties</td>
<td>Any differentiable function is theoretically permitted</td>
<td>Any differentiable function is theoretically permitted</td>
</tr>
</tbody>
</table>