Deep Generative Models

Aida Mostafazadeh Davani
mostafaz@usc.edu

A Generative Model is a powerful way of learning any kind of data distribution using unsupervised learning. All generative models represent probability distributions over multiple variables in someway and aim at learning the true data distribution of the training set so as to generate new data points with some variations.

1 Boltzmann Machines

A Boltzmann machine (Fahlman et al., 1983) is a network of units that are connected to each other. Let *N* be the number of units. Each unit takes a binary value (0 or 1). Let X_i be the random variable representing the value of the *i*-th unit for $i \in [1, N]$. We use a column vector *X* to denote the random values of the *N* units. The Boltzmann machine has two types of parameters: bias and weight.

The Boltzmann machine is an energy-based model meaning we define the joint probability distribution using an energy function:

$$P(x) = \frac{exp(-E(x))}{Z} \tag{1}$$

where *E(x)* is the energy function, and *Z* is the partition function that ensures that $\Sigma_x P(x) = 1$.

$$Z = \sum_{\bar{x}} exp(-E(\bar{x})) \tag{2}$$

The energy function of the Boltzmann machine is given by:

$$E(x) = \sum_{i=1}^{N} b_i x_i - \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w_{i,j} x_i x_j \tag{3}$$

$$= -x^\top W x - b^\top x \tag{4}$$

where *W* is the weight matrix of model parameters and *b* is the vector of bias parameters. In the general setting of the Boltzmann machine, we are given a set of training examples, each of which are *n*-dimensional. Equation 1 describes the joint probability distribution over the observed variables.

The Boltzmann machine becomes more powerful when not all the variables are observed. In this case, the latent variables can act similarly to hidden units in a multilayer perceptron and model higher-order interactions among the visible units. Instead, the Boltzmann machine becomes a universal approximator of probability mass functions over discrete variables. Formally, we decompose the units *x* into two subsets: the visible units *v* and the latent (or hidden) units *h*. The energy function becomes:

$$E(v, h) = -v^\top R v - v^\top W h - h^\top S h - b^\top v - c^\top h. \tag{5}$$

Learning algorithms for Boltzmann machines are usually based on maximum likelihood. All Boltzmann machines have an intractable partition function, so the maximum likelihood gradient must be approximated using the techniques in previous presentation.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

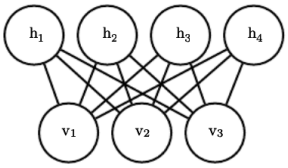

Figure 1: The restricted Boltzmann machine is an undirected graphical model based on a bipartite graph, with visible units in one part of the graph and hidden units in the other part. There are no connections among the visible units, nor any connections among the hidden units. Typically every visible unit is connected to every hidden unit, but it is possible to construct sparsely connected RBMs such as convolutional RBMs.

One interesting property of Boltzmann machines when trained with learning rules based on maximum likelihood is that the update for a particular weight connecting two units depends only on the statistics of those two units, collected under different distributions.

2 Restricted Boltzmann Machines

RBMs (Smolensky, 1986) are undirected probabilistic graphical models containing a layer of observable variables and a single layer of latent variables (figure 1). Like the general Boltzmann machine, the restricted Boltzmann machine is an energy-based model with the joint probability distribution specified by its energy function:

$$P(v = v, h = h) = \frac{1}{Z} exp(-E(v, h)) \tag{6}$$

The energy function for an RBM is given by

$$E(v, h) = -b^\top v - c^\top h - v^\top W h, \tag{7}$$

and *Z* is the normalizing constant known as the partition function:

$$Z = \sum_{\bar{v}} \sum_{\bar{h}} exp(-E(\bar{v}, \bar{h})) \tag{8}$$

It is apparent that the computing *Z* by exhaustively summing over all states could be computationally intractable. Long and Servedio (2010) formally proved that the partition function *Z* and consequently *P(v)* is intractable.

2.1 Conditional Distributions

However, due to the bipartite structure of the graph, $P(v \mid h)$ and $P(h \mid v)$ are factorial and relatively simple to compute and sample from.

Since we are conditioning on the visible units *v*, we can treat these as constant with respect to the distribution $P(h \mid v)$.

$$P(h \mid v) = \frac{P(h, v)}{P(v)} \tag{9}$$

$$= \frac{1}{P(v)} \frac{1}{Z} exp\{b^\top v + c^\top h + v^\top W h\} \tag{10}$$

$$= \frac{1}{Z'} exp\{c^\top h + v^\top W h\} \tag{11}$$

$$= \frac{1}{Z'} exp\{\sum_{j=1}^{n_h} c_j h_j + \sum_{j=1}^{n_h} v^\top W_{:,j} h_j\} \tag{12}$$

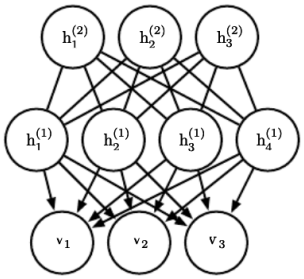

Figure 2: In a deep belief network there are no intralayer connections. Usually, every unit in each layer is connected to every unit in each neighboring layer, though it is possible to construct more sparsely connected DBNs. The connections between the top two layers are undirected. The connections between all other layers are directed, with the arrows pointed toward the layer that is closest to the data

$$= \frac{1}{Z'} \prod_{j=1}^{n_h} exp\{c_j h_j + v^\top W_{:,j} h_j\} \tag{13}$$

It is now a simple matter of normalizing the distributions over the individual binary h_j:

$$P(h_j = 1 \mid \boldsymbol{v}) = \frac{\tilde{P}(h_j = 1 \mid \boldsymbol{v})}{\tilde{P}(h_j = 0 \mid \boldsymbol{v}) + \tilde{P}(h_j = 1 \mid \boldsymbol{v})} \tag{14}$$

$$= \frac{exp\{c_j + \boldsymbol{v}^\top W_{:,j}\}}{exp\{0\} + exp\{c_j + \boldsymbol{v}^\top W_{:,j}\}} \tag{15}$$

$$= \sigma\left(c_j + \boldsymbol{v}^\top W_{:,j}\right) \tag{16}$$

We can now express the full conditional over the hidden layer as the factorial distribution:

$$P(h \mid v) = \prod_{j=1}^{n_h} \sigma\left((2h - 1) \cdot (c + W^\top v)\right)_j \tag{17}$$

and accordingly:

$$P(v \mid h) = \prod_{i=1}^{n_v} \sigma\left((2v - 1) \cdot (b + Wh)\right)_i \tag{18}$$

3 Deep Belief Networks

The introduction of deep belief networks in 2006 (Hinton and Salakhutdinov, 2006) began the current deep learning renaissance. Prior to the introduction of deep belief networks, deep models were considered too difficult to optimize. DBNs are generative models with several layers of latent variables (figure 2). A DBN with *l* hidden layers contains *l* weight matrices: $W^{(1)}, ..., W^{(l)}$. It also contains *l* + 1 bias vectors $b^{(0)}, ..., b^{(l)}$, with $b^{(0)}$ providing the biases for the visible layer. The probability distribution represented by the DBN is given by:

$$P(h^{(l)}, h^{(l-1)}) \propto exp\left(b^{(l)} h^{(l)} + b^{(l-1)\top} h^{(l-1)} + h^{(l-1)} W^{(l)} h^{(l)}\right) \tag{19}$$

$$P(h_i^{(k)} = 1 \mid h^{(k+1)}) = \sigma\left(b_i^{(k)} + W_{:,i}^{(k+1)\top} h^{(k+1)}\right) \forall i, \forall k \in 1, ..., l-1 \tag{20}$$

$$P(v_i = 1 \mid h^{(1)}) = \sigma\left(b_i^{(0)} + W_{:,i}^{(1)\top} h^{(1)}\right) \forall i \tag{21}$$

To generate a sample from a DBN, we first run several steps of Gibbs sampling on the top two hidden layers. We can then use a single pass of ancestral sampling through the rest of the model to draw a sample from the visible units.

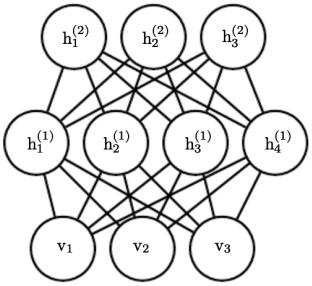

Figure 3: A deep Boltzmann machine is an undirected graphical model with several layers of latent variables. Like RBMs and DBNs, DBMs lack intralayer connections.DBMs are less closely tied to RBMs than DBNs are. Deep Boltzman nmachines have been applied to a variety of tasks, including document modeling

To train a deep belief network, one begins by training an RBM to maximize $\mathbb{E}_{v \sim P_{data}} \log p(v)$ using contrastive divergence or stochastic maximum likelihood. The parameters of the RBM then define the parameters of the first layer of the DBN. Next, a second RBM is trained to approximately maximize

$$\mathbb{E}_{v \sim p_{data}} \mathbb{E}_{h^{(1)} \sim p^{(1)}(h^{(1)}|v)} log p^{(2)}(h^{(1)}) \tag{22}$$

where $p^{(1)}$ is the probability distribution represented by the first RBM, and $p^{(2)}$ is the probability distribution represented by the second RBM.

The trained DBN may be used directly as a generative model, but most of the interest in DBNs arose from their ability to improve classification models. We can take the weights from the DBN and use them to define an MLP:

$$h^{(1)} = \sigma\left(b^{(1)} + v^\top W^{(1)}\right) \tag{23}$$

$$h^{(l)} = \sigma\left(b^{(l)} + h^{(l-1)\top} W^{(l)}\right) \forall l \in 2, ..., m \tag{24}$$

The MLP is initialized with the learned weights and biases from the generative training of the DBN and propagates information upward from the visible units to the deepest hidden units, but it does not propagate any information downward or sideways.

4 Deep Boltzmann Machines

A DBM (Salakhutdinov and Hinton, 2009), an entirely undirected model (figure 3), is an energy-based model and the joint probability distribution over the model variables is parametrized by the energy function *E* like other deep generative models.

$$P\left(v, h^{(1)}, h^{(2)}, h^{(3)}\right) = \frac{1}{Z(\theta)} exp\left(-E(v, h^{(1)}, h^{(2)}, h^{(3)}; \theta)\right) \tag{25}$$

To simplify our presentation, we omit the bias parameters below. The DBM energy function is then defined as follows:

$$E(v, h^{(1)}, h^{(2)}, h^{(3)}; \theta) = -v^\top W^{(1)} h^{(1)} - h^{(1)\top} W^{(2)} h^{(2)} - h^{(2)\top} - W^{(3)} h^{(3)} \tag{26}$$

DBM energy function includes connections between the hidden units (latent variables) in the form of the weight matrices.

The DBM layers can be organized into a bipartite graph, with odd layers on one side and even layers on the other (figure 4). Thus we can apply the same equations we have previously used for the conditional distributions of an RBM to determine the conditional distributions in a DBM. The units within a layer are conditionally independent from each other given the values of the neighboring

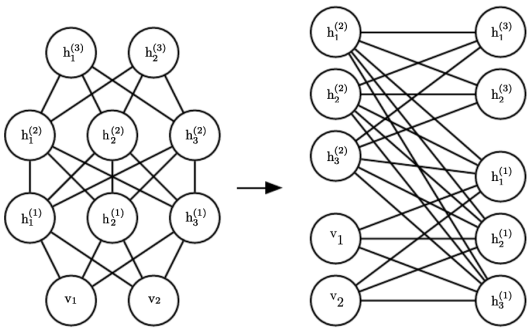

Figure 4: Converting the DBM to its bipartite form

layers, so the distributions over binary variables can be fully described by the Bernoulli parameters, giving the probability of each unit being active.

$$P(v_i = 1 \mid h^{(1)}) = \sigma(W^{(1)}_{i,:} h^{(1)}) \tag{27}$$

$$P(h^{(1)}_i \mid v, h^{(2)}) = \sigma\left(v^\top W^{(1)}_{:,i} + W^{(2)}_{i,:} h^{(2)}\right) \tag{28}$$

And the same probability distribution can be modeled for upper layers.

Due to the bipartite structure, Gibbs sampling can be performed the same as in RBMs by updating one block in the same time (only two iterations).

5 Boltzmann Machines for Real-Valued Data

While Boltzmann machines were originally developed to be used with binary data, many applications such as image and audio modeling seem to require the ability to represent probability distributions over real values.

5.1 Gaussian-Bernoulli RBMs

The most common RBM designed for exponential family conditional distributions is the RBM with binary hidden units and real-valued visible units, with the conditional distribution over the visible units being a Gaussian distribution whose mean is a function of the hidden units.

For example by using a precision matrix for the Gaussian distribution we will have

$$p(v \mid h) = \mathcal{N}(v; Wh, \mathcal{B}^{-1}) \tag{29}$$

Based on the Gaussian function:

$$log\mathcal{N}(v; Wh, \mathcal{B}^{-1}) = -\frac{1}{2}(v - Wh)^\top \mathcal{B}(v - Wh) + f(\mathcal{B}) \tag{30}$$

where $f(\mathcal{B})$ is only a function of parameters and can be discarded from the energy function. $p(v \mid h)$ can be represented by all the terms from equation 30 involving v, but we have more freedom for defining $p(h|v)$.

5.2 Undirected Models of Conditional Covariance

In some types of real valued data, such as images, much of the information is embedded in the covariance between units (pixels). Therefore, the Gaussian RBM cannot capture the conditional covariance.

In **Mean and Covariance RBM** hidden unit is divided into mean units (Gaussian RBM - $h^{(m)}$) and covariance units (cRBM - $h^{(c)}$). The energy function

$$E_{mc}(x, h^{(m)} = h^{(c)}) = E_m(x, h^{(m)}) + E_c(x, h^{(c)}) \tag{31}$$

where

$$E_m(x, h^{(m)}) = \frac{1}{2} x^\top x - \sum_j x^\top W_{:,j} h_j^{(m)} - \sum_j b_j^{(m)} h_j^{(m)} \tag{32}$$

and

$$E_c(x, h^{(c)}) = \frac{1}{2} \sum_j h_j^{(c)} \left(x^\top r^{(j)} \right)^2 - \sum_j b_j^{(c)} h_j^{(c)} \tag{33}$$

6 Convolutional Boltzmann Machines

Replacing matrix multiplication by discrete convolution with a small kernel is the standard way of solving memory and computation problems for high-dimensional inputs. Deep convolutional networks usually require a pooling operation so that the spatial size of each successive layer decreases (Lee et al., 2009).

In order to generalize this to the energy based models, we could introduce a binary pooling unit p over *n* binary detector units **d** and enforce $p = max_i d_i$ by setting the energy function to be ∞ whenever that constraint is violated which requires evaluating 2^n different energy configurations.

Instead, **probabilistic max pooling** constrains the detector units so at most one may be active at a time. The resulting model only has *n* + 1 states.

7 Boltzmann Machines for Structured or Sequential Outputs

Let's assume that the model is trained to map from inputs *x* to outputs *y*, and different entries of *y* are related to each other. Boltzmann machines can supply the probability distribution $p(y \mid x)$ that represents the relationships. The same is true for sequence modeling ($p(x^{(t)} \mid x^{(1)}, ..., x^{(t-1)})$).

An important sequence modeling task for the video game and film industry is modeling sequences of joint angles of skeletons used to render 3-D characters. Conditional RBM over preceding *m* values of *x* is introduced to model $p(x^{(t)} \mid x^{(t-1)}, ..., x^{(t-m)})$ for this problem. The weights in the RBM over *x* never change, but by conditioning on different past values, we can change the probability of different hidden units in the RBM being active.

Another sequence modeling task is to model the distribution over sequences of musical notes used to compose songs. **RNN-RBM** is designed to emit the RBM parameters (both bias and weights) for each time step.

References

Scott E Fahlman, Geoffrey E Hinton, and Terrence J Sejnowski. 1983. Massively parallel architectures for al: Netl, thistle, and boltzmann machines. In *National Conference on Artificial Intelligence, AAAI*.

Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507.

Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM.

Philip M Long and Rocco A Servedio. 2010. Restricted boltzmann machines are hard to approximately evaluate or simulate.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Deep boltzmann machines. In *Artificial intelligence and statistics*, pages 448–455.

Paul Smolensky. 1986. Information processing in dynamical systems: Foundations of harmony theory. Technical report, Colorado Univ at Boulder Dept of Computer Science.