Deep Generative Models

Energy-Based Models

Hamid Beigy

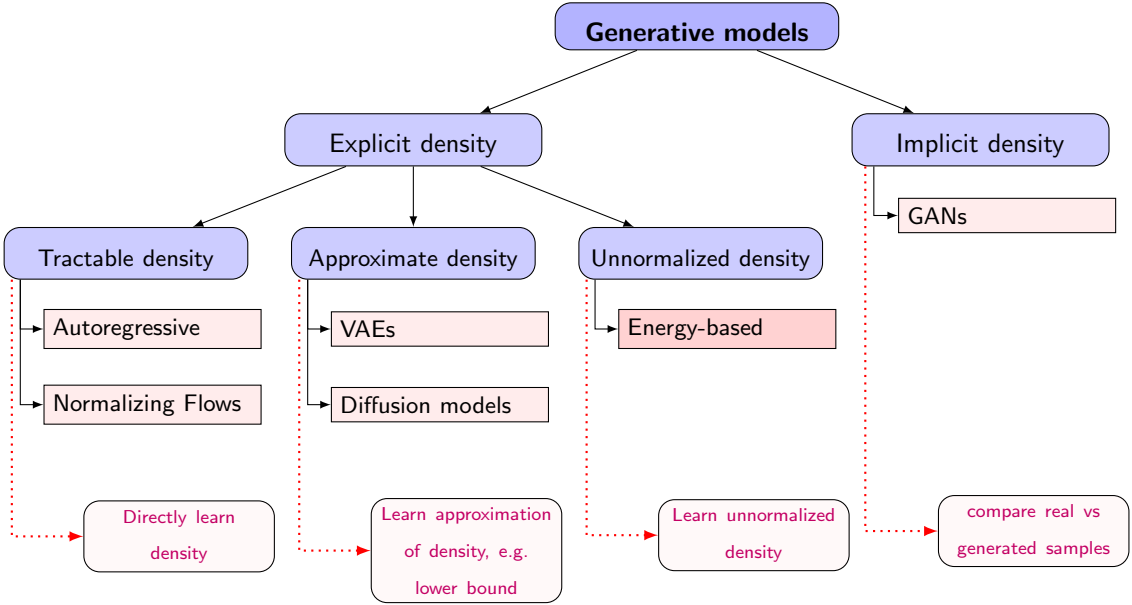
Sharif University of Technology

May 2, 2025

Table of contents

Introduction

Generative models categories



Generative models

1. Assume that the observed variable $\mathbf{x}$ is a random sample from an underlying process, whose true distribution $p_{\mathrm{d}}(\mathbf{x})$ is unknown.
2. We attempt to approximate this process with a chosen model, $p_\theta(\mathbf{x})$, with parameters θ such that $\mathbf{x} \sim p_\theta(\mathbf{x})$.
3. Learning is the process of searching for the parameter θ such that $p_\theta(\mathbf{x})$ well approximates $p_{\mathrm{d}}(\mathbf{x})$ for any observed $\mathbf{x}$, i.e.

$$p_\theta(\mathbf{x}) \approx p_{\mathrm{d}}(\mathbf{x})$$

4. We wish $p_\theta(\mathbf{x})$ to be sufficiently flexible to be able to adapt to the data for obtaining sufficiently accurate model and to be able to incorporate prior knowledge.

Deep generative models

Autoregressive models

1. Tractable density
2. Density is estimated as

$$p(\mathbf{x};\theta) = \prod_{j=1}^{m} p(\mathbf{x}_j \mid \mathbf{x}_{<j};\theta)$$

3. Tractable likelihood
4. No inferred latent factors

Normalizing flow models

1. Exact density
2. Density is estimated as

$$p_\theta(\mathbf{x}) = p_\mathbf{z}(\mathbf{z})|\det(\mathbf{J}_f)|$$
$$\text{where } \mathbf{z} = f(\mathbf{x})$$

3. Tractable likelihood
4. Latent feature representation

Latent variable models

1. Approximated density
2. Density is estimated as

$$p(\mathbf{x};\theta) = \int p(\mathbf{x},\mathbf{z};\theta)d\mathbf{z}$$

3. Intractable likelihood
4. Latent feature representation

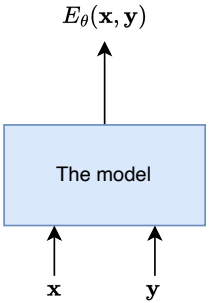
Generative adversarial networks

1. Implicit density
2. Can optimize f-divergences and Wasserstein distance

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_\mathrm{d}(\mathbf{x})}[\log D(x)] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]$$

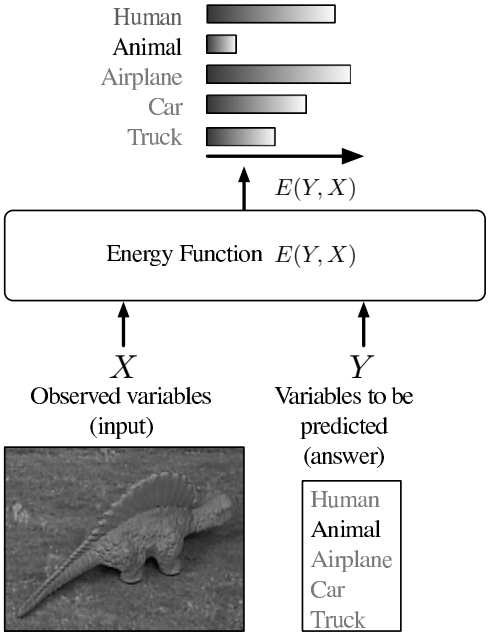
3. Latent feature representation
4. Very flexible model architectures, unstable training, hard evaluation, mode collapse

1. EBMs capture dependencies between variables by associating a scaler energy to each configuration.



2. **Learning** in EBMs consists in finding energy function in which observed configurations are given lower energies than unobserved ones.
3. **Inference** in EBMs consists of finding the observed variables and finding the remaining variables that minimize the energy.

When $\mathbf{x}_i$ is an image and $\mathbf{y}_i$ is its label from set $\{Human, Animal, Airplan, Car, Truck\}$, then the model can be represented as



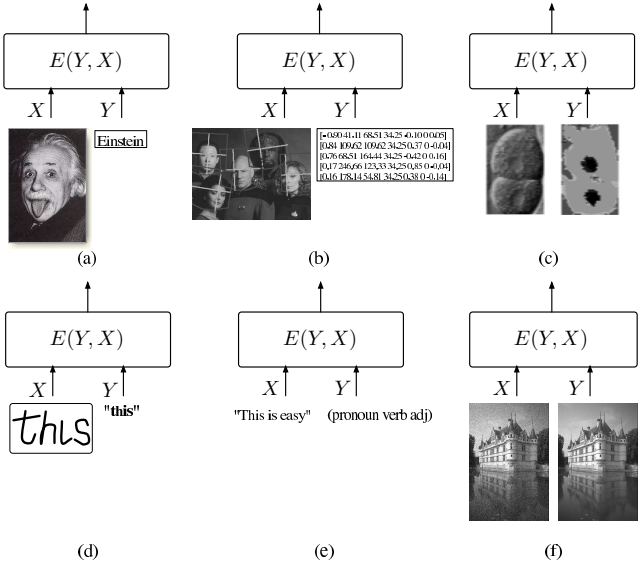


Figure 2: *Several applications of EBMs:* **(a) face recognition:** *Y is a high-cardinality discrete variable;* **(b) face detection and pose estimation:** *Y is a collection of vectors with location and pose of each possible face;* **(c) image segmentation:** *Y is an image in which each pixel is a discrete label;* **(d-e) handwriting recognition and sequence labeling:** *Y is a sequence of symbols from a highly structured but potentially infinite set (the set of English sentences). The situation is similar for many applications in natural language processing and computational biology;* **(f) image restoration:** *Y is a high-dimensional continuous variable (an image).*

What questions can an EBM answer?

An EBM may be used to answer questions of several types:

1. Prediction, classification, and decision-making:

   "Which value of $\mathbf{y}_i$ is most compatible with this $\mathbf{x}$?'

2. Ranking:

   "Is $\mathbf{y}_1$ or $\mathbf{y}_2$ more compatible with this $\mathbf{x}$?"

3. Detection:

   "Is Is this value of $\mathbf{y}$ compatible with $\mathbf{x}$?"

4. Conditional density estimation:

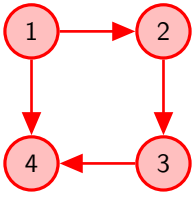
   "What is the conditional probability distribution over $\mathbf{y}$ given $\mathbf{x}$?"

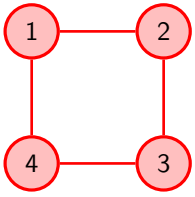
Energy-based models

1. The EBM approach provides a common theoretical framework for many learning models including:
   - Traditional discriminative approaches
   - Traditional generative approaches
   - Graph-Transformer networks
   - Conditional random fields
   - Maximum margin Markov networks
   - Several manifold learning methods
2. Energy-based models have
   - Very flexible model architectures
   - Stable training
   - Relatively high sample quality
   - Flexible composition

Probabilistic Graphical Models

1. PGMs provide a general framework for describing and applying probabilistic models in the probabilistic approach.
2. Consider a graph $G = (V, E)$ with vertex set V and edge set $E \subseteq V \times V$.
3. A graphical model is a family of probabilistic distributions defined in terms of a directed or undirected graph.





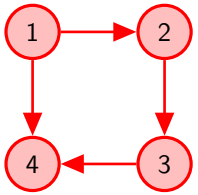
4. The nodes are random variables and joint probability distributions are defined by taking products over functions defined on connected subsets of nodes.
5. Graphical models can be defined over different types of graphs:
   - Directed graphical models
   - Undirected graphical models
   - Mixed graphical models

1. Let $G = (V, E)$ be a directed acyclic graph and $\mathbf{x} = (x_1, \ldots, x_n)$ be a collection of random variables indexed by nodes of G.
2. For each node $v \in V$, let $pa(v)$ be the parents of node v and $\mathbf{x}_{pa(v)}$ be a vector of random variables indexed by parents of v.

   A directed graphical model consists of a family of distributions that factorize as.

$$p(\mathbf{x}) = \prod_{v \in V} p(x_v \mid \mathbf{x}_{pa(v)})$$

3. Consider the following directed graphical model:

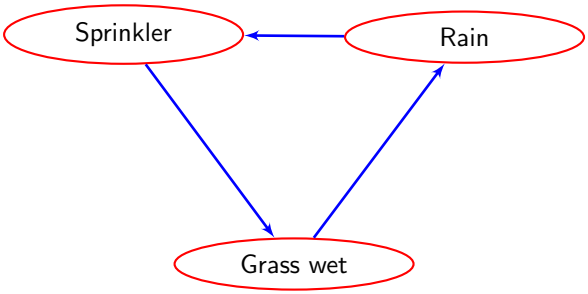


4. The joint distribution that this directed graphical model describes is

$$p(x_1, x_2, x_3, x_4) = p(x_1)\, p(x_2 \mid x_1)\, p(x_3 \mid x_2)\, p(x_4 \mid x_1, x_3)$$

1. Traditionally, each conditional probability distribution $p(x_v \mid \mathbf{x}_{pa(v)})$ is parameterized by a lookup table or a linear model.

Rain	Sprinkler T	Sprinkler F
F	0.4	0.6
T	0.01	0.99



Sprinkler T	Sprinkler F
0.2	0.8

Sprinkler rain		Grass wet T	Grass wet F
F	F	0.4	0.6
F	T	0.01	0.99
T	F	0.01	0.99
T	T	0.01	0.99

Neural Network Based Classifier

1. A more flexible way to parametrize such conditional distribution is to use neural networks.
2. This neural networks takes $\mathbf{x}_{pa(v)}$ as input and produces the distributional parameters over the variables:

$$\theta = \text{NN}(\mathbf{x}_{pa(v)})$$
$$p(x_v \mid \mathbf{x}_{pa(v)}) = p_\theta(x_v)$$

3. As an example, if x_v is a continuous variable, θ could denote the mean and variance parameters.
4. Consider a multi-class classifier, whose input is $\mathbf{x} = (x_1, \ldots, x_n)$ and the class label is $y \in \{1, \ldots, K\}$.
5. The graphical model representation of this classifier is

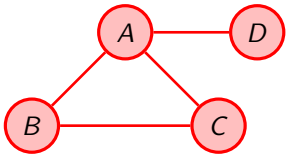
1. The classic multi-class logistic regression is to use a single layer to obtain the logits z_k.
2. Then, feed the logits z_k into a softmax layer to calculate the class posterior:

$$
\begin{aligned}
p(y = k \mid \mathbf{x}) &= \frac{\exp(z_k)}{\sum_{j=1}^{K} \exp(z_j)} \\
z_k &= \mathbf{w}_k^{\mathsf{T}} \mathbf{x} + b_k \qquad \text{for } k = 1, \ldots, K \\
&= \mathit{Linear}(\mathbf{x}; \mathbf{w}_k, b_k)
\end{aligned}
$$

3. Instead of a linear layer, we can use a deep network.
4. In this way, the multi-layer neural network could be viewed as a non-linear feature extractor.

Undirected Graphical Model

1. Let $G = (V, E)$ be an undirected graph and $\mathbf{x} = (x_1, \ldots, x_n)$ be the random variables.
2. Let $\mathcal{C}$ denote the set of cliques of graph.
3. Let $\phi_c(\mathbf{x}_c) \geq 0$ be the potential function associated with each clique $c \in \mathcal{C}$.
4. Consider the following undirected graphical model



5. The total potential function of the graph is defined as

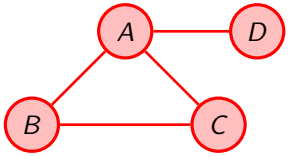
$$\Phi(\mathbf{x}) = \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c) \qquad\qquad \phi_c(\mathbf{x}_c) \geq 0.$$

   where $\mathbf{x}_c$ is an arbitrary instantiation of of the set of random variables denoted by clique c.

6. For the above graph, the potential function can be factorized as

$$\Phi(\mathbf{x}) = \phi_{A,B,C}(a, b, c) \times \phi_{A,D}(a, d).$$

1. Consider the following undirected graphical model



2. Let $\{A, D\}$ be binary random variables, the potential function corresponding to this clique could be represented by a table

$$\phi_{\{A,D\}}(a = 0, d = 0) = +4.00$$
$$\phi_{\{A,D\}}(a = 0, d = 1) = +0.23$$
$$\phi_{\{A,D\}}(a = 1, d = 0) = +5.00$$
$$\phi_{\{A,D\}}(a = 1, d = 1) = +9.45$$

3. Like other models in machine learning, the potential function can be parametrized as

$$\Phi(\mathbf{x}; \theta) = \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c; \theta) \qquad \phi_c(\mathbf{x}_c; \theta) \geq 0.$$

4. Potential functions show **how likely a given state is.** So, **the higher the potential, more likely that state is**.

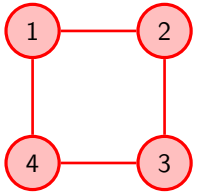
1. An undirected graphical model consists of a family of distributions that factorizes as

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c; \theta).$$

   where Z is a normalizing constant(partition function) given by

$$Z = \sum_{\mathbf{x}} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c; \theta).$$

2. Consider the following UGM



3. The joint distribution that this model describes is

$$p(x_1, x_2, x_3, x_4) = \frac{1}{Z} \phi(x_1, x_2) \phi(x_2, x_3) \phi(x_3, x_4) \phi(x_1, x_4)$$

4. In contrast with joint distributions, **the potential functions do not need to be self-normalized.**

Energy-Based Models

Representing probability distributions

1. Probability distributions $p_\theta(\mathbf{x})$ are a key building block in generative modeling.
2. They have the following properties
   - Non-negative: $p_\theta(\mathbf{x}) \geq 0$
   - Sum to one: $\sum_{\mathbf{x}} p_\theta(\mathbf{x}) = 1$ or $\int_{\mathbf{x}} p_\theta(\mathbf{x}) d\mathbf{x} = 1$
3. Making non-negativeness is easy and we can choose any of the following function:

$$
\begin{aligned}
g_\theta(\mathbf{x}) &= f_\theta(\mathbf{x})^2 \\
g_\theta(\mathbf{x}) &= \exp(f_\theta(\mathbf{x})) \\
g_\theta(\mathbf{x}) &= |f_\theta(\mathbf{x})| \\
g_\theta(\mathbf{x}) &= \log(1 + \exp(f_\theta(\mathbf{x})))
\end{aligned}
$$

4. In general $Z_\theta = \sum_{\mathbf{x}} g_\theta(\mathbf{x}) \neq 1$.
5. Hence, $g_\theta(\mathbf{x})$ is not a valid probability mass function or density.

1. The maintaining $g_\theta(\mathbf{x}) \geq 0$ is easy but making $\sum_{\mathbf{x}} p_\theta(\mathbf{x}) = 1$ is a hard problem.
2. A solution is to normalize $g_\theta(\mathbf{x})$ by its volume as

$$p_\theta(\mathbf{x}) = \frac{g_\theta(\mathbf{x})}{Volume(g_\theta)} = \frac{g_\theta(\mathbf{x})}{\int_{\mathbf{x}} g_\theta \, d\mathbf{x}}$$

3. Then, by definition we have $\int_{\mathbf{x}} p_\theta(\mathbf{x}) d\mathbf{x} = 1$.
4. We can calculate the volume analytically if we choose some analytical functions such as

Density function	**Volume**
$g_{(\mu,\sigma)}(x) = e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\int_x e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = \sqrt{2\pi\sigma^2}$
$g_\lambda(x) = e^{-\lambda x}$	$\int_0^\infty e^{-\lambda x} dx = \frac{1}{\lambda}$
$g_\theta(x) = h(x)\exp(\theta T(x))$	$\exp\big(\log \int h(x)\exp(\theta T(x)) dx\big)$

5. The above functional forms seem to be restrictive but they are very useful as building blocks for more complex distributions.

1. **Problem:** The maintaining $g_\theta(\mathbf{x}) \geq 0$ is easy but it might not be normalized.
2. **Solution:** A solution is to normalize $g_\theta(\mathbf{x})$ by its volume as

$$p_\theta(\mathbf{x}) = \frac{1}{Volume(g_\theta)} g_\theta(\mathbf{x}) = \frac{1}{\int_{\mathbf{x}} g_\theta \, d\mathbf{x}} g_\theta(\mathbf{x})$$

3. Typically, we choose $g_\theta(\mathbf{x})$ in such a way that the volume is known analytically.
4. By combining some building blocks, we obtain more complex models.
5. **Autoregressive:** Products of normalized functions $p_{\theta_1}(\mathbf{x})\, p_{\theta_2(\mathbf{x})}(\mathbf{y})$

$$\int_{\mathbf{x}} \int_{\mathbf{y}} p_{\theta_1}(\mathbf{x})\, p_{\theta_2(\mathbf{x})}(\mathbf{y}) \mathrm{d}\mathbf{x} \mathrm{d}\mathbf{y} = \int_{\mathbf{x}} p_{\theta_1}(\mathbf{x}) \underbrace{\int_{\mathbf{y}} p_{\theta_2(\mathbf{x})}(\mathbf{y}) \mathrm{d}\mathbf{y}}_{=1} \mathrm{d}\mathbf{x} = \int_{\mathbf{x}} p_{\theta_1}(\mathbf{x}) \mathrm{d}\mathbf{x} = 1$$

6. **Latent variables:** Convex combination of normalized functions $\alpha\, p_{\theta_1}(\mathbf{x}) + (1-\alpha)\, p_{\theta_2}(\mathbf{y})$

$$\int_{\mathbf{x}} \alpha\, p_{\theta_1}(\mathbf{x}) + (1-\alpha)\, p_{\theta_2}(\mathbf{y}) \mathrm{d}\mathbf{x} = \alpha + (1-\alpha) = 1$$

7. How about using models where the volume of $g_\theta(\mathbf{x})$ is not easy to compute analytically?

1. By restricting potential functions to be strictly positive, it is convenient to express them as

$$\phi_c(\mathbf{x}_c) = \exp[-E_c(\mathbf{x}_c)]$$

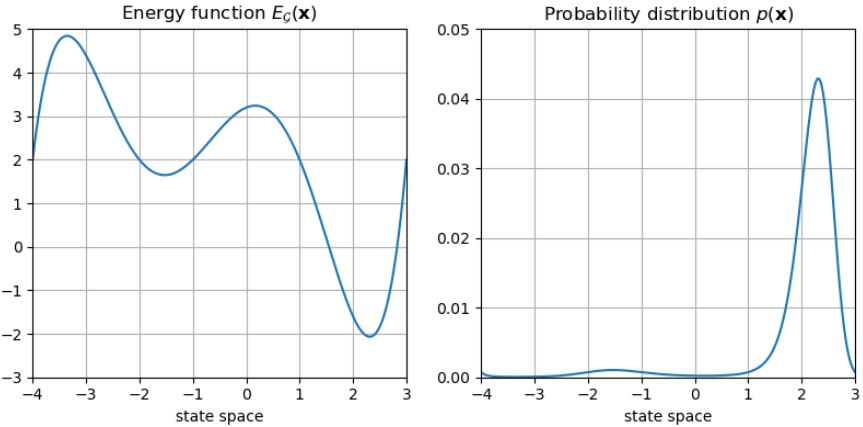
where $E_c(\mathbf{x}_c)$ is called an **energy function** (Alternatively, happiness, $H(\mathbf{x}) = -E(\mathbf{x})$, is used).

2. Hence, the negative log-potential is called energy.
3. The high-probability states correspond to low-energy configuration
4. The parametrized versions of the probability density functions

$$p_\theta(\mathbf{x}) = \frac{1}{Z_\theta} \exp\left(-\sum_{c \in \mathcal{C}} E_c(\mathbf{x}_c; \theta)\right)$$
$$= \frac{1}{Z_\theta} \exp(-E_\theta(\mathbf{x}))$$

5. Distributions of this exponential forms are called energy-based models, also known as Gibbs (Boltzmann) distributions.
6. A benefit of EBM is that
   - energy functions are not constrained to be non-negative.
   - energy functions can be very flexible parametrized.

An energy function and its corresponding probability distributions



1. The density function given by an EBM is

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

   - $E_\theta(\mathbf{x})$ (the energy) is a nonlinear regression function with parameters θ.
   - Z_θ denotes the normalizing constant (partition function):

$$Z_\theta = \int \exp(-E_\theta(\mathbf{x}))d\mathbf{x}$$

   - Z_θ is constant w.r.t $\mathbf{x}$ and is a function of θ.

2. Why exponential form?
   - Want to capture very large variations in probability. we want to work with log-probability.
   - Exponential families. Many common distributions can be written in this form.
   - These distributions arise under fairly general assumptions in statistical physics.

1. The density function given by an EBM is

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

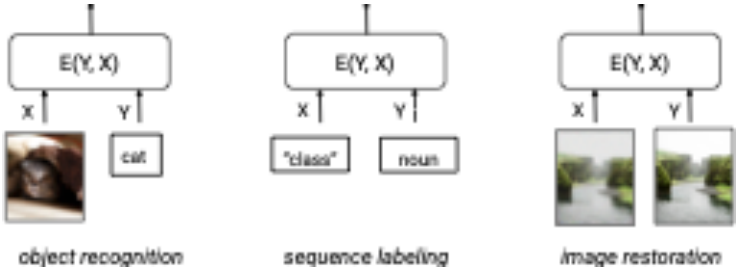
2. **Pros:**
   - **Extreme flexibility:** can use any function $E_\theta(\mathbf{x})$ you want
3. **Cons:**
   - Sampling from $p_\theta(\mathbf{x})$ is hard.
   - Evaluating and optimizing likelihood $p_\theta(\mathbf{x})$ is hard (learning is hard).
   - No feature learning (but we can add latent variables)
4. **Problem:** A fundamental problem is that computing Z_θ numerically scales exponentially in the number of dimensions of $\mathbf{x}$.

Representing probability distributions

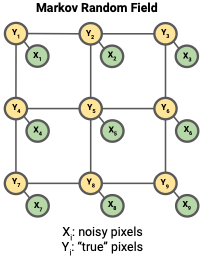
1. For two vectors $\mathbf{x}$ and $\mathbf{z}$, evaluating $p_\theta(\mathbf{x})$ and $p_\theta(\mathbf{z})$ requires Z_θ
2. However, in some tasks such as ranking and anomaly detection, we do not require knowing Z_θ.
3. In these tasks, we need require relative comparisons of $p_\theta(\mathbf{x})$ and $p_\theta(\mathbf{z})$ such as their ratio:

$$\frac{p_\theta(\mathbf{x})}{p_\theta(\mathbf{z})} = \exp(E_\theta(\mathbf{z}) - E_\theta(\mathbf{x}))$$

does not involve computing Z_θ.



1. Let $\mathbf{y} \in \{0,1\}^{3\times 3}$ be the true image and $\mathbf{x} \in \{0,1\}^{3\times 3}$ be a corrupted image.
2. Corrupted image $\mathbf{x} \in \{0,1\}^{3\times 3}$ is given and we must recover the true image $\mathbf{y} \in \{0,1\}^{3\times 3}$.



3. We model the joint probability distribution $p(\mathbf{x}, \mathbf{y})$ as

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\left(\sum_i \psi_i(x_i, y_i) + \sum_{(i,j)\in E} \psi_{i,j}(y_i, y_j) \right)$$

where

- $\psi_i(x_i, y_i)$: the i-th corrupted pixel depends on the i-th original pixel
- $\psi_{i,j}(y_i, y_j)$: neighboring pixels tend to have the same value

4. How did the original image $\mathbf{y}$ look like? **Solution:** maximize $p(\mathbf{y} \mid \mathbf{x})$.

Product of Experts

1. Let n expert models $p_{\theta_1}(\mathbf{x}), \ldots, p_{\theta_m}(\mathbf{x})$, each parameterized by $\theta_1, \ldots, \theta_m$, respectively.
2. The probability distribution of the product of experts (PoE) can be expressed as:

$$p_\theta(\mathbf{x}) = \frac{\prod_k p_{\theta_k}(\mathbf{x})}{Z_\theta} = \frac{\prod_k p_{\theta_k}(\mathbf{x})}{\sum_{\mathbf{z}} \prod_k p_{\theta_k}(\mathbf{z})} \qquad \text{where} \quad \theta = \{\theta_1, \ldots, \theta_m\}.$$

Image Source: (Du, Li, and Mordatch 2020)

Product of Experts

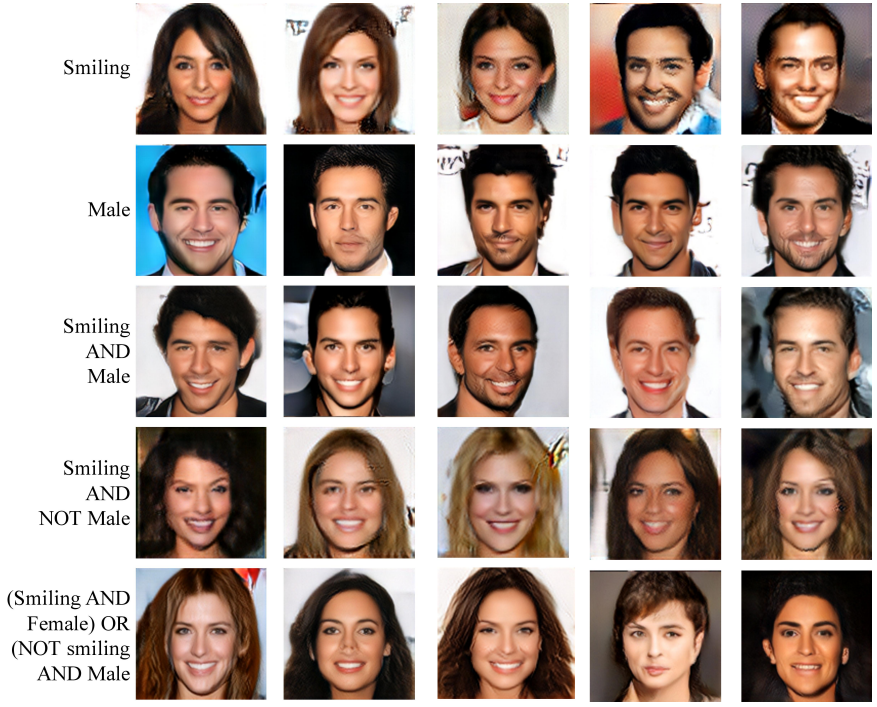
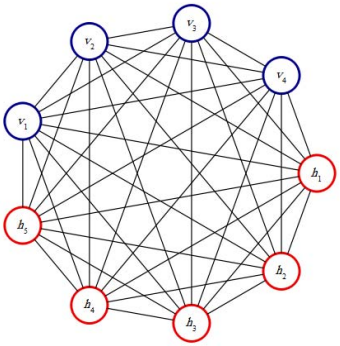


Image Source: (Du, Li, and Mordatch 2020)

Boltzmann Machine (BM)

1. BMs are fully connected networks of binary units.
2. BM is an undirected symmetric network of binary units divided into visible and hidden.



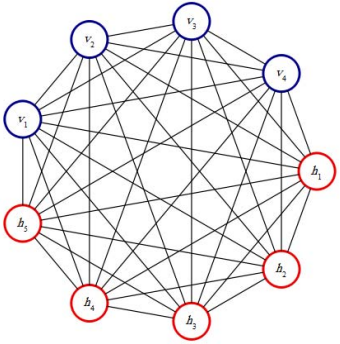
3. BMs are theoretically capable of learning any given distribution.
4. BMs set strengths of connections between units to capture their correlations to build a generative network capable of producing new examples of the same distribution.
5. Since all variables in a BM are not directly observed, it gives us a handle to control the sampling of new examples.
6. The model can take in an incomplete example and use it to output the complete version.

Boltzmann Machine (BM)

1. BM is a network with an energy defined for the overall network.
2. For a BM with only observed units, the energy is defined as
$$E_\theta(\mathbf{x}) = -\mathbf{x}^\top \mathbf{W}\mathbf{x} - \mathbf{b}^\top \mathbf{x}$$
3. Learning algorithms for BMs are usually based on maximum likelihood.
4. All BMs have an intractable partition function, so the maximum likelihood gradient must be approximated.
5. A BM admits the following likelihood for points $\mathbf{x}_1, \ldots, \mathbf{x}_m$.
$$\mathcal{L}(\mathbf{x}) = \prod_{i=1}^{n} p(\mathbf{x}_i)$$
6. We will work with the log-likelihood instead of the true likelihood.
7. The gradient w.r.t weights becomes
$$\nabla_{w_{i,j}} \log \mathcal{L}(\mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[x_i x_j] - \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[x_i x_j]$$
8. The gradient w.r.t biases becomes
$$\nabla_{b_i} \log \mathcal{L}(\mathbf{x}) = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[x_i] - \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[x_i]$$

Generating a sample by Boltzmann Machine

1. In BM, we generate in two steps:
   - Pick the hidden states from $p(\mathbf{h})$.
   - Pick the visible states from $p(\mathbf{v}|\mathbf{h})$.



2. The probability of generating a visible vector, $\mathbf{v}$, is computed by summing over all possible hidden states.

$$p(\mathbf{v}) = \sum_{\mathbf{h}} p(\mathbf{h})p(\mathbf{v}|\mathbf{h})$$

Gibbs sampling

1. Given an ordered set of variable, $x_1, \ldots, x_d$, and a starting configuration $x^0 = (x_1^0, \ldots, x_d^0)$,

Gibbs sampling uses the following procedure

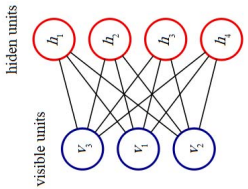
- Repeat until convergence for $t = 1, 2, \ldots,$
  - Set $\mathbf{x} \leftarrow \mathbf{x}^{t-1}$.
  - For each variable x_i in the order we fixed:
    1) Sample $x_i' \sim p(x_i \mid \mathbf{x}_{-i})$.
    2) Update $\mathbf{x} \leftarrow (x_1, \ldots, x_i', \ldots, x_d)$.
  - Set $\mathbf{x}^t \leftarrow \mathbf{x}$.

We use $\mathbf{x}_{-i}$ to denote all variables in $\mathbf{x}$ except x_i.

2. It is often very easy to performing each sampling step, since we only need to condition x_i on other variables.
3. Note that when we update x_i, we immediately use its new value for sampling other variables x_j.

Restricted Boltzmann Machine (RBM)

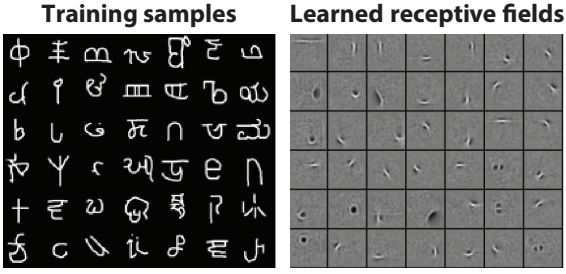
1. The intractability of the joint distribution is one of the biggest drawbacks of BMs.
2. RBMs are a special type of BMs with two layers: One visible and one hidden layer.



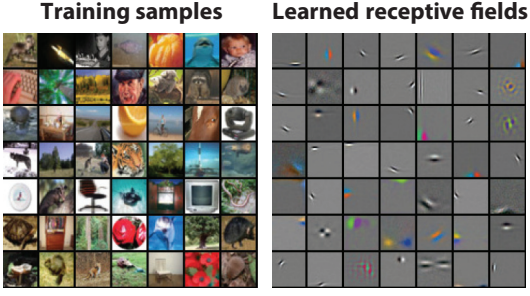
3. The connections in an RBM are undirected and the graph is a bipartite graph.
4. The probability density is calculated by

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z_\theta} \exp(-E_\theta(\mathbf{v}, \mathbf{h}))$$
$$E_\theta(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^\top \mathbf{W}\mathbf{h} - \mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h}$$
$$Z_\theta = \sum_{\mathbf{v} \in \{0,1\}^D} \sum_{\mathbf{h} \in \{0,1\}^F} \exp(-E_\theta(\mathbf{v}, \mathbf{h}))$$

1. The learned receptive fields of Bernoulli–Bernoulli RBM

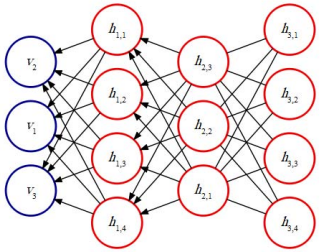


2. The learned receptive fields of Gaussian–Bernoulli RBM



Deep Belief Networks (DBN)

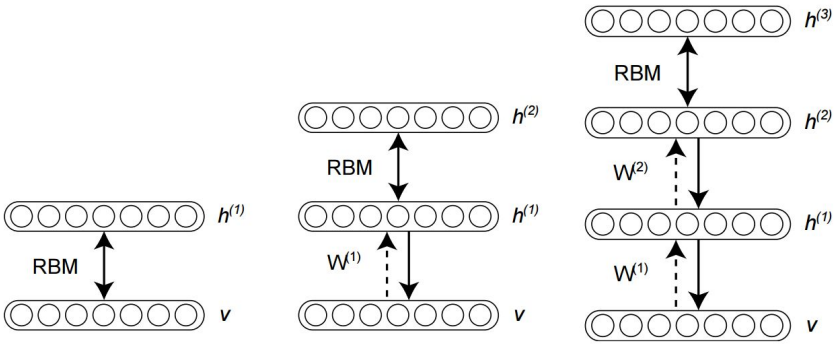
1. A single layer of binary features is not the best way to capture the structure in high-dimensional input data.
2. DBN is a hybrid PGM involving both directed and undirected connections.
3. Deep belief networks consisting of many hidden layers.
   - Connections between top two layers are undirected
   - Connections between all other layers is directed, pointing towards data.



$$p(\mathbf{v}, \mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(k)}) = p(\mathbf{v}|\mathbf{h}^{(1)})p(\mathbf{h}^{(1)}|\mathbf{h}^{(2)}) \ldots p(\mathbf{h}^{(k-2)}|\mathbf{h}^{(k-1)})p(\mathbf{h}^{(k-1)}, \mathbf{h}^{(k)})$$

4. $p(\mathbf{h}^{(k-1)}, \mathbf{h}^{(k)})$ (the marginal distribution over the top two layers) is an RBM.

1. Deep belief networks training
   - 1.1 We first train the bottom RBM with parameters $\mathbf{W}^{(1)}$.
   - 1.2 We then initialize the second layer weights to $\mathbf{W}^{(2)} = \mathbf{W}^{(1)}$, ensuring that the two-hidden-layer DBN is at least as good as our original RBM.
   - 1.3 Improve the fit of the DBN to the training data by untying and refitting parameters $\mathbf{W}^{(2)}$.

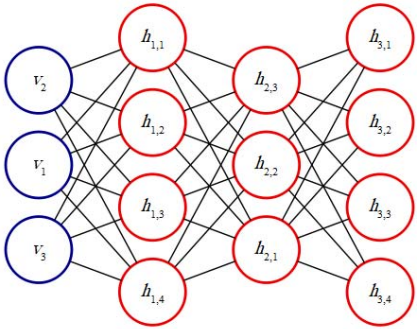


(a) Stage 1 (b) Stage 2 (c) Stage 3

2. **Find the variational lower bound of the log-likelihood of the two-hidden-layer DBN.**

Deep Boltzmann Machine (DBM)

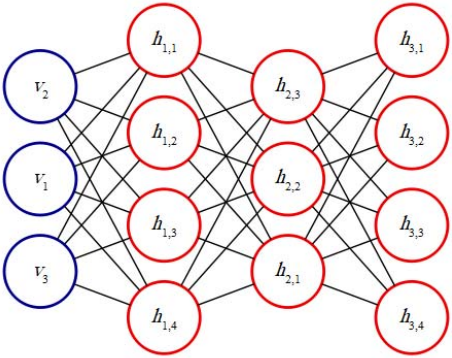
1. DBM is an undirected deep network of several hidden layers (Salakhutdinov and Larochelle 2010).
2. Every unit is connected to every unit from the adjacent layers.
3. There are no connections between units of the same layer.

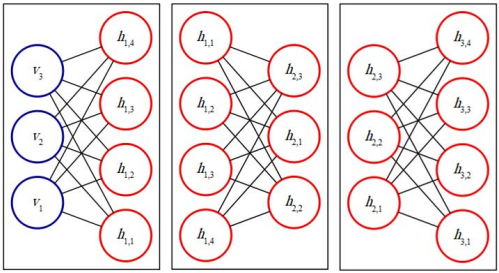


4. **Derive the conditional probability of each layer given its above layer.**
5. **Derive derivative of the log-likelihood with respect to the model parameters.**

Deep Boltzmann Machine (DBM)

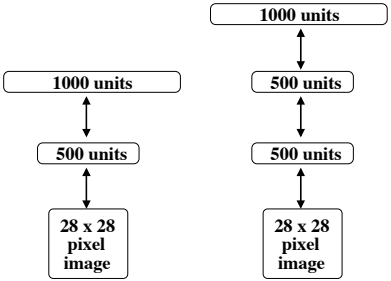
1. DBMs can also be viewed as a group of RBMs stacked together.



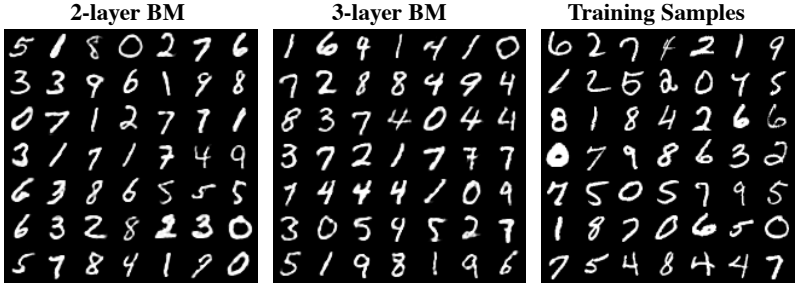


2. Training of DBMs is often done in two stages:
   - A pre-training stage where every RBM is trained independently.
   - a fine tuning stage where the network is trained at once using backpropagation.

1. Considering two architectures for MNIST dataset.



2. The results using Gibbs sampling.



Training Energy-Based Models

Maximum Likelihood Training of EBMs

1. We use the maximum likelihood estimation to train EBMs:
2. Let

$$\mathcal{L}_{NLL}(\theta) = -\log \prod_{k=1}^{m} p_\theta(\mathbf{x}_k) = -\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log\ p_\theta(\mathbf{x})]$$

3. MLE is equivalent to minimizing $\mathrm{D}_{KL}(\,p_{\mathrm{d}}(\mathbf{x})\ ||\ p_\theta(\mathbf{x}))$:

$$\begin{aligned}\mathcal{L}_{KL}(\theta) = \mathrm{D}_{KL}(\,p_{\mathrm{d}}(\mathbf{x})\ ||\ p_\theta(\mathbf{x})) &= \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}\left[\log \frac{p_{\mathrm{d}}(\mathbf{x})}{p_\theta(\mathbf{x})}\right] \\ &= \underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log\ p_{\mathrm{d}}(\mathbf{x})]}_{\text{Independent of } \theta} - \mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\log\ p_\theta(\mathbf{x})] \\ &= \text{Constant} + \mathcal{L}_{NLL}(\theta)\end{aligned}$$

4. We cannot directly compute the likelihood of an EBM as in the maximum likelihood approach due to the intractable normalizing constant Z_θ.
5. Nevertheless, we can still estimate the gradient of the log-likelihood with MCMC approaches, allowing for likelihood maximization with gradient ascent.

Maximum Likelihood Training of EBMs

1. The density function given by an EBM is

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

2. The gradient of negative log-likelihood (NLL) is decomposed to:

$$\nabla_\theta \log\, p_\theta(\mathbf{x}) = -\nabla_\theta E_\theta(\mathbf{x}) - \nabla_\theta \log Z_\theta$$

3. The first gradient term, $-\nabla_\theta E_\theta(\mathbf{x})$, is straightforward to evaluate with automatic differentiation.
4. The challenge is in approximating the second gradient term, $\nabla_\theta \log Z_\theta$, which is intractable to compute exactly.

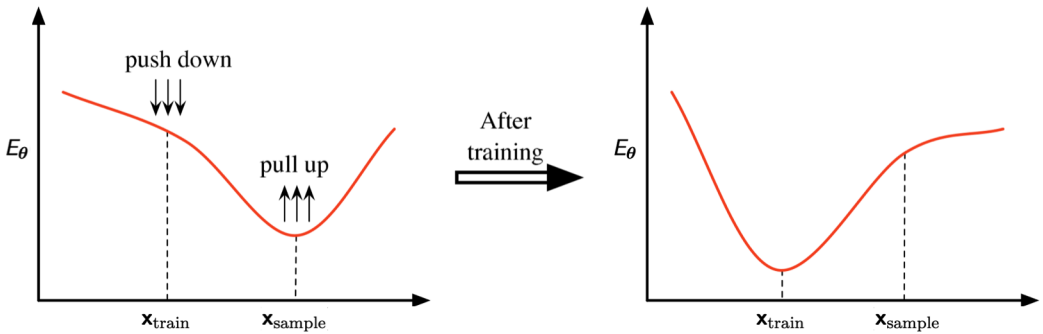
$\nabla_\theta \log Z_\theta$ an be rewritten as follows:

$$
\begin{aligned}
\nabla_\theta \log Z_\theta &= \nabla_\theta \log \int \exp(-\mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x} \\
&= \left(\int \exp(-\mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x}\right)^{-1} \nabla_\theta \int \exp(-\mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x} \\
&= \left(\int \exp(-\mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x}\right)^{-1} \int \nabla_\theta \exp(-\mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x} \\
&= \left(\int \exp(-\mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x}\right)^{-1} \int \exp(-\mathrm{E}_\theta(\mathbf{x}))(-\nabla_\theta \mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x} \\
&= \int \left(\int \exp(-\mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x}\right)^{-1} \exp(-\mathrm{E}_\theta(\mathbf{x}))(-\nabla_\theta \mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x} \\
&= \int \frac{\exp(-\mathrm{E}_\theta(\mathbf{x}))}{Z_\theta}(-\nabla_\theta \mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x} \\
&= \int p_\theta(\mathbf{x})(-\nabla_\theta \mathrm{E}_\theta(\mathbf{x}))\mathrm{d}\mathbf{x} = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[-\nabla_\theta \mathrm{E}_\theta(\mathbf{x})]
\end{aligned}
$$

1. Then, $\nabla_\theta \mathcal{L}_{NLL}(\theta)$ equals to

$$\nabla_\theta \mathcal{L}_{NLL}(\theta) = \underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\mathsf{d}}(\mathbf{x})}[\nabla_\theta \mathsf{E}_\theta(\mathbf{x})]}_{\text{Positive phase}} - \underbrace{\mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[\nabla_\theta \mathsf{E}_\theta(\mathbf{x})]}_{\text{Negative phase}}$$

2. Positive phase tries to change the parameters to minimize the energy at points coming from training set.
3. Negative phase tries to change the parameters to maximize the energy at points coming from model.



1. Then, $\nabla_\theta \mathcal{L}_{NLL}(\theta)$ equals to

$$\nabla_\theta \mathcal{L}_{NLL}(\theta) = \underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\mathrm{d}}(\mathbf{x})}[\nabla_\theta \mathrm{E}_\theta(\mathbf{x})]}_{\text{Positive phase}} - \underbrace{\mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[\nabla_\theta \mathrm{E}_\theta(\mathbf{x})]}_{\text{Negative phase}}$$

2. For computing the negative phase, usually collect samples from $p_{\mathrm{d}}(\mathbf{x})$ using Markov chains and make a Monte Carlo estimate of the expectation.
3. The problem is that in practice MCMC chains mix very poorly on complex data.
4. An unbiased one-sample Monte Carlo estimate of log-likelihood gradient is

$$\nabla_\theta \log Z_\theta \approx -\nabla_\theta \mathrm{E}_\theta(\tilde{\mathbf{x}})$$

   where $\tilde{\mathbf{x}} \sim p_\theta(\mathbf{x})$ is a random sample from the distribution over $\mathbf{x}$ given by the EBM.
5. This algorithm is called **contrastive divergence training**.

1. The loss function equals to

$$\mathcal{L}_{CD}(\theta) = \underbrace{\mathrm{D}_{KL}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x}))}_{\mathcal{L}_{NLL}(\theta)} - \mathrm{D}_{KL}(\, p^t_\theta(\mathbf{x}) \,||\, p_\theta(\mathbf{x}))$$

   where $p^t_\theta(\mathbf{x})$ is the distribution resulting from running t steps of the MC chain starting from $p_{\mathrm{d}}(\mathbf{x})$.

2. The second term is another negative likelihood loss.
3. The second term tries to minimize $\mathbb{E}_{\mathbf{x}\sim\, p^t_\theta(\mathbf{x})}[\, p_\theta(\mathbf{x})]$.
4. It can be shown that the above loss function and negative log likelihood loss functions have the same solution:

$$\mathcal{L}_{CD}(\theta) = 0 \qquad \Longrightarrow \qquad \mathcal{L}_{NLL}(\theta) = 0$$

   But they get there in different ways.

5. **Homework:** Drive the updating rule of the parameters (Gagnon and Lajoie 2022).

1. As long as we can draw random samples from the model, we have access to an unbiased Monte Carlo estimate of the log-likelihood gradient, allowing us to optimize the parameters with stochastic gradient ascent.
2. Since drawing random samples is far from being trivial, much of the literature has focused on methods for efficient MCMC sampling from EBMs.
3. Some efficient MCMC methods make use of the fact that the gradient of the log-probability w.r.t. $\mathbf{x}$ (score) is equal to the (negative) gradient of the energy, therefore easy to calculate:

$$\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x}) - \underbrace{\nabla_{\mathbf{x}} \log Z_\theta}_{=0} = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$$

4. For example, Langevin MCMC initially draws a sample $\mathbf{x}_0$ from a simple prior distribution, and then uses a process for K steps with step size $\epsilon > 0$:

$$\mathbf{x}^{k+1} \leftarrow \mathbf{x}^k + \frac{\epsilon^2}{2} \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}) + \epsilon \mathbf{z}^k$$

where $\mathbf{z}^k \sim \mathcal{N}(0, \mathbf{I})$ is a Gaussian noise term.

Maximum Likelihood Training

1. The density function given by an EBM is

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

2. The gradient of negative log-likelihood (NLL) is decomposed to:

$$\nabla_\theta \log\, p_\theta(\mathbf{x}) = -\nabla_\theta E_\theta(\mathbf{x}) - \nabla_\theta \log Z_\theta$$

3. The first gradient term, $-\nabla_\theta E_\theta(\mathbf{x})$, is straightforward to evaluate with automatic differentiation, but the exact computation of the second term is interactable.

$$\nabla_\theta \log Z_\theta = \mathbb{E}_{\mathbf{x}\sim p_\theta(\mathbf{x})}[-\nabla_\theta E_\theta(\mathbf{x})]$$

4. Sampling converges slowly in high dimensional spaces and is thus very expensive, yet we need sampling for each training iteration in contrastive divergence.
5. **The goal is training without sampling**
   - Score Matching
   - Noise Contrastive Estimation
   - Adversarial training

Training Energy-Based Models

Score-based methods

Score Matching (SM)

1. Let $f(\mathbf{x})$ and $g(\mathbf{x})$ be two continuously differentiable real-valued functions.
2. If $f(\mathbf{x})$ and $g(\mathbf{x})$ have equal first derivatives everywhere, then $f(\mathbf{x}) = g(\mathbf{x}) + \mathit{Constant}$.
3. When $f(\mathbf{x})$ and $g(\mathbf{x})$ are log-pdfs with equal first derivatives, the normalization requirement implies that $\int \exp(f(\mathbf{x}))d\mathbf{x} = \int \exp(g(\mathbf{x}))d\mathbf{x} = 1$ and $f(\mathbf{x}) \equiv g(\mathbf{x})$.
4. We can approximately learn an EBM by matching the first derivatives of its log-pdf to the first derivatives of the log-pdf of the data distribution.
5. If they match, then the EBM captures the data distribution exactly.
6. The **first-order gradient function of a log-pdf** is also called **the score of that distribution**.
7. For training EBMs, it is useful to transform the equivalence of distributions to the equivalence of scores, because the score of an EBM can be easily obtained.

Score Matching (SM)

1. The score of an EBM can be easily obtained by $\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x}) = -\nabla_{\mathbf{x}} E_\theta(\mathbf{x})$, which does not involve the typically intractable normalizing constant Z_θ.
2. The basic score matching objective minimizes a discrepancy between two distributions called the **Fisher divergence**:

$$\mathrm{D}_{FS}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x})) = \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}\left[\frac{1}{2}\|\nabla_{\mathbf{x}} \log\, p_{\mathrm{d}}(\mathbf{x}) - \nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})\|^2\right]$$

3. The first term admits a trivial unbiased Monte Carlo estimator using the empirical mean of samples $\mathbf{x} \sim p_{\mathrm{d}}(\mathbf{x})$.
4. The second term is generally impractical to calculate since it requires knowing $\nabla_{\mathbf{x}} \log\, p_\theta(\mathbf{x})$.

Training Energy-Based Models

Noise Contrastive Estimation based Methods

Noise Contrastive Estimation (NCE)

1. The core idea behind NCE is to distinguish data samples from a dataset (signal) from artificially generated noise samples.
2. This is achieved by training a binary classifier that learns to classify whether a given sample comes from the actual data distribution or from a noise distribution.
3. The classifier implicitly learns the parameters of the data distribution.
4. NCE involves the following steps:
   - 4.1 A noise distribution is chosen, which should ideally be simple enough to sample from and calculate probabilities.
   - 4.2 Noise samples are generated from this noise distribution.
   - 4.3 A logistic regression model is trained to discriminate between samples from the true data distribution and the noise samples.
   - 4.4 The parameters learned by the logistic regression model are then used as estimates for the parameters of the true data distribution.

Noise Contrastive Estimation

1. Advantages of NCE over MLE methods
   - 1.1 **Computational Efficiency:** NCE avoids the computation of the partition function, which can be intractable for large models.
   - 1.2 **Scalability:** NCE scales well with the size of the dataset and the complexity of the model.
   - 1.3 **Flexibility:** NCE can be applied to a wide range of models, including those where MLE is not feasible.
2. Challenges and considerations of using NCE
   - 2.1 **Choice of Noise Distribution:** The performance of NCE is sensitive to the choice of noise distribution. A poor choice can lead to suboptimal parameter estimation.
   - 2.2 **Hyperparameter Tuning:** NCE requires careful tuning of hyperparameters, including the number of noise samples and the learning rate for the classifier.
   - 2.3 **Convergence:** Ensuring convergence of the estimation process can be challenging, especially for complex models with many parameters.

Noise Contrastive Estimation

1. NCE is based on the idea that we can learn an Energy-Based Model by contrasting it with another distribution with known density.
2. Let $p_{\text{d}}(\mathbf{x})$ be the data distribution, and let $p_n(\mathbf{x})$ be a chosen distribution with known density, called a noise distribution.
3. This noise distribution is usually simple and has a tractable pdf, like $\mathcal{N}(0, \mathbf{I})$, such that we can compute the pdf and generate samples from it efficiently.
4. Let y be a binary variable with Bernoulli distribution, which we use to define a mixture distribution of noise and data:

$$p_{n,data}(\mathbf{x}) = p(y=0)\, p_n(\mathbf{x}) + p(y=1)\, p_{\text{d}}(\mathbf{x})$$

5. Based on the Bayes' rule, given a sample $\mathbf{x}$ from this mixture, the posterior probability of $y=0$ is

$$p_{n,data}(y=0 \mid \mathbf{x}) = \frac{p_{n,data}(\mathbf{x} \mid y=0)\, p(y)}{p_{n,data}(\mathbf{x})} = \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \alpha\, p_{\text{d}}(\mathbf{x})}$$

Drive this equation.

where $\alpha = \frac{p(y=1)}{p(y=0)}$.

Noise Contrastive Estimation

1. Let our energy-based model has the following form:

$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z_\theta}$$

2. Unlike other EBMs, Z_θ is treated as a learnable (scalar) parameter in NCE.
3. Given this model, we can define a mixture of noise and the model distribution:

$$p_{n,\theta}(\mathbf{x}) = p(y=0)\, p_n(\mathbf{x}) + p(y=1)\, p_\theta(\mathbf{x})$$

4. The posterior probability of $y = 0$ given this noise/model mixture is

$$p_{n,\theta}(y=0 \mid \mathbf{x}) = \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \alpha\, p_\theta(\mathbf{x})}$$

1. In NCE, we indirectly fit $p_\theta(\mathbf{x})$ to $p_{\text{d}}(\mathbf{x})$ by fitting $p_{n,\theta}(y \mid \mathbf{x})$ to $p_{n,data}(y \mid \mathbf{x})$ through a standard conditional maximum likelihood objective:

$$\begin{aligned}\theta^* &= \arg\min_\theta \Big\{ \mathbb{E}_{p_{n,data}(\mathbf{x},y)}[\mathrm{D}_{KL}(\, p_{n,data}(y \mid \mathbf{x}) \;||\; p_{n,\theta}(y \mid \mathbf{x}))] \Big\} \\ &= \arg\max_\theta \Big\{ \mathbb{E}_{p_{n,data}(\mathbf{x},y)}[\log\, p_{n,\theta}(y \mid \mathbf{x})] \Big\} \end{aligned}$$

Derive this function

2. This optimization problem can be solved using stochastic gradient ascent.
3. Like any other deep classifier, when the model is sufficiently powerful, $p_{n,\theta^*}(y \mid \mathbf{x})$ will match $p_{n,data}(y \mid \mathbf{x})$ at the optimum.

$$\begin{aligned} & p_{n,\theta^*}(y=0 \mid \mathbf{x}) = p_{n,data}(y=0 \mid \mathbf{x}) \\ \iff \quad & \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \alpha\, p_{n,\theta^*}(\mathbf{x})} = \frac{p_n(\mathbf{x})}{p_n(\mathbf{x}) + \alpha\, p_{n,data}(\mathbf{x})} \\ \iff \quad & p_{\theta^*}(\mathbf{x}) = p_{\text{d}}(\mathbf{x}) \end{aligned}$$

4. Consequently, $E_{\theta^*}(\mathbf{x})$ is an unnormalized energy function that matches the data distribution $p_{\text{d}}(\mathbf{x})$, and Z_{θ^*} is the corresponding normalizing constant.

Training Energy-Based Models

Adversarial training

Adversarial training

1. When training EBMs with MLE, we need to sample from the EBM per training iteration.
2. Sampling using multiple MCMC steps is expensive and requires careful tuning of the Markov chain.
3. One way to avoid this difficulty is to use non-MLE methods that do not need sampling, such as Score Matching and Noise Contrastive Estimation.
4. We can sidestep costly MCMC sampling by learning an auxiliary model through adversarial training, which allows fast sampling.
5. From the definition of EBMs, we can rewrite the maximum likelihood objective by introducing a variational distribution $q_\phi(\mathbf{x})$ parameterized by ϕ:

$$
\begin{aligned}
\mathbb{E}_{p_d(\mathbf{x})}[\log\, p_\theta(\mathbf{x})] &= \mathbb{E}_{p_d(\mathbf{x})}[-E_\theta(\mathbf{x})] - \log Z_\theta \\
&= \mathbb{E}_{p_d(\mathbf{x})}[-E_\theta(\mathbf{x})] - \log \int \exp(-E_\theta(\mathbf{x})) \\
&= \mathbb{E}_{p_d(\mathbf{x})}[-E_\theta(\mathbf{x})] - \log \int \exp(-E_\theta(\mathbf{x})) \frac{q_\phi(\mathbf{x})}{q_\phi(\mathbf{x})} \\
&\leq \mathbb{E}_{p_d(\mathbf{x})}[-E_\theta(\mathbf{x})] - \int q_\phi(\mathbf{x}) \log \frac{\exp(-E_\theta(\mathbf{x}))}{q_\phi(\mathbf{x})} \qquad \text{Using Jensen inequality} \\
&= \mathbb{E}_{p_d(\mathbf{x})}[-E_\theta(\mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{x})}[-E_\theta(\mathbf{x})] - H(\, q_\phi(\mathbf{x}))
\end{aligned}
$$

Adversarial training

1. The upperbound of $\mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}[\log\, p_\theta(\mathbf{x})]$ is

$$\mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}[\log\, p_\theta(\mathbf{x})] \leq \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}[-E_\theta(\mathbf{x})] - \mathbb{E}_{q_\phi(\mathbf{x})}[-E_\theta(\mathbf{x})] - H(\, q_\phi(\mathbf{x}))$$

2. For EBM training,
   - First minimize the upper bound with respect to $q_\phi(\mathbf{x})$ so that it is closer to the likelihood objective.
   - Then maximize with respect to $E_\theta(\mathbf{x})$ as a surrogate for maximizing likelihood.
3. This amounts to using the following maximin objective

$$\max_\theta \min_\phi \mathbb{E}_{q_\phi(\mathbf{x})}[E_\theta(\mathbf{x})] - \mathbb{E}_{p_{\mathrm{d}}(\mathbf{x})}[E_\theta(\mathbf{x})] - H(\, q_\phi(\mathbf{x}))$$

4. Optimizing the above objective is similar to training GANs and can be achieved by adversarial training.
5. The variational distribution $q_\phi(\mathbf{x})$ should allow both fast sampling and efficient entropy evaluation to make the maximin objective function tractable.

Hybrid Modeling

1. Consider using deep generative modeling in the context of finding the joint distribution over observables and decision variables that is factorized as

$$p(\mathbf{x}, y) = p(y \mid \mathbf{x})\, p(\mathbf{x})$$

where $\mathbf{x} \in \mathbb{R}^D$ and $y \in \{0, 1, \ldots, K-1\}$.

2. By taking the logarithm of the joint we obtain two additive components:

$$\log\, p(\mathbf{x}, y) = \log\, p(y \mid \mathbf{x}) + \log\, p(\mathbf{x})$$

3. How can we model the above problem using EBMs?

4. Let $E_\theta(\mathbf{x}, y)$ be parameterized by a neural network $NN_\theta(\mathbf{x})$ where its input is $\mathbf{x}$ and returns K values: $NN_\theta : \mathbb{R}^D \mapsto \mathbb{R}^K$.

5. This means that we can define energy function as

$$E_\theta(\mathbf{x}, y) = -NN_\theta(\mathbf{x})[y]$$

where $[y]$ denotes the specific output of the neural networks $NN_\theta(\mathbf{x})$.

1. Then, the joint probability distribution is defined as

$$p(\mathbf{x}, y) = \frac{\exp(-E_\theta(\mathbf{x}, y))}{Z_\theta} = \frac{\exp(NN_\theta(\mathbf{x})[y])}{Z_\theta}$$

2. The marginal distribution $p(\mathbf{x})$ is

$$p_\theta(\mathbf{x}) = \sum_y p(\mathbf{x}, y) = \frac{\sum_y \exp(NN_\theta(\mathbf{x})[y])}{Z_\theta}$$

3. We can re-write the numerator in the following manner:

$$\sum_y \exp(NN_\theta(\mathbf{x})[y]) = \exp\left(\log\left\{\sum_y \exp(NN_\theta(\mathbf{x})[y])\right\}\right) = \exp(LogSumExp_y(NN_\theta(\mathbf{x})[y]))$$

4. We can say that the energy function of the marginal distribution is expressed as $-LogSumExp_y(NN_\theta(\mathbf{x})[y])$.

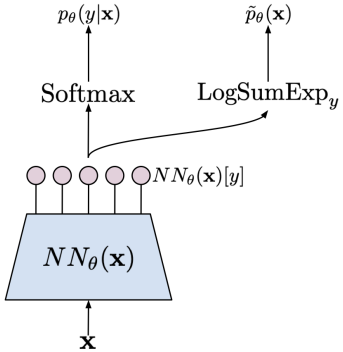
1. The conditional distribution $p_\theta(y \mid \mathbf{x})$ is

$$\begin{aligned} p_\theta(y \mid \mathbf{x}) &= \frac{p_\theta(\mathbf{x}, y)}{p_\theta(\mathbf{x})} \\ &= \frac{\frac{\exp(NN_\theta(\mathbf{x})[y])}{Z_\theta}}{\frac{\sum_y \exp(NN_\theta(\mathbf{x})[y])}{Z_\theta}} \\ &= \frac{\exp(NN_\theta(\mathbf{x})[y])}{\sum_y \exp(NN_\theta(\mathbf{x})[y])}. \end{aligned}$$

2. This means that the energy-based model could be used either as a classifier or a marginal distribution.
3. Any any classifier could be seen as an energy-based model (Grathwohl et al. 2020).
4. The logarithm of the joint distribution is

$$\begin{aligned} \log\ p_\theta(\mathbf{x}, y) &= \log \frac{\exp(f_\theta(\mathbf{x})[y])}{\sum_y \exp(NN_\theta(\mathbf{x})[y])} + \log \frac{\sum_y \exp(NN_\theta(\mathbf{x})[y])}{Z_\theta} \\ &= \log Softmax(NN_\theta(\mathbf{x})[y]) + (LogSumExp_y(NN_\theta(\mathbf{x})[y]) - \log Z_\theta) \end{aligned}$$

1. The model requires a shared neural network that is used for calculating both distributions.



2. We have a single neural network to train and the training objective is the logarithm of the joint distribution.
3. The training objective is a sum of the logarithm of the conditional $p_\theta(y \mid \mathbf{x})$ and the logarithm of the marginal $p_\theta(\mathbf{x})$.
4. Calculating the gradient with respect to the parameters θ requires taking the gradient of each of the component separately (**Derive the weight update equations**).

Summary

Summary

1. Both Variational Autoencoders and EBM learn the parameters by maximizing the (marginal) log-likelihood, which can be interpreted also as the minimization of $\mathrm{D}_{KL}(\, p_{\mathrm{d}}(\mathbf{x}) \,||\, p_\theta(\mathbf{x}))$.
2. VAEs are intrinsically latent variable models imposing an information bottleneck and approximating the posterior on the latent variables $p_{\mathrm{d}}(\mathbf{z} \mid \mathbf{x})$ through variational inference, whereas EBMs generally are not.
3. EBMs can easily extended to latent variable models (Xiao, Yan, and Amit 2020).
4. Che et. al. showed that GANs can be better understood through the lens of EBM (Che et al. 2020).
5. They showed that GAN generators and discriminators collaboratively learn an implicit energy-based model.

References

Reading

1. Paper Learning Deep Generative Models (Salakhutdinov 2015).
2. Chapter A Tutorial on Energy-Based Learning (Lecun et al. 2006).
3. Paper How to Train Your Energy-Based Models (Song and Kingma 2021).
4. Chapter 24 of Probabilistic Machine Learning: Advanced Topics (Murphy 2023).
5. Chapter 7 of Deep Generative Modeling (Tomczak 2024).

References i

- Che, Tong et al. (2020). "Your GAN is Secretly an Energy-based Model and You Should Use Discriminator Driven Latent Sampling". In: *Advances in Neural Information Processing Systems*.
- Du, Yilun, Shuang Li, and Igor Mordatch (2020). "Compositional Visual Generation with Energy Based Models". In: *Advances in Neural Information Processing Systems*.
- Gagnon, Léo and Guillaume Lajoie (2022). *Clarifying MCMC-based training of modern EBMs : Contrastive Divergence versus Maximum Likelihood*. arXiv: 2202.12176.
- Grathwohl, Will et al. (2020). "Your classifier is secretly an energy based model and you should treat it like one". In: *International Conference on Learning Representations*.
- Lecun, Yann et al. (2006). "A tutorial on energy-based learning". In: *Predicting structured data*. Ed. by G. Bakir et al. MIT Press.
- Murphy, Kevin P. (2023). *Probabilistic Machine Learning: Advanced Topics*. The MIT Press.
- Salakhutdinov, Ruslan (2015). "Learning Deep Generative Models". In: *Annual Review of Statistics and Its Application* 2, pp. 361–385.
- Salakhutdinov, Ruslan and Hugo Larochelle (2010). "Efficient Learning of Deep Boltzmann Machines". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 9, pp. 693–700.
- Song, Yang and Diederik P. Kingma (2021). "How to Train Your Energy-Based Models". In: *CoRR* abs/2101.03288.

References ii

- Tomczak, Jakub M. (2024). *Deep Generative Modeling*. Springer.
- Xiao, Zhisheng, Qing Yan, and Yali Amit (2020). "Exponential Tilting of Generative Models: Improving Sample Quality by Training and Sampling from Latent Energy". In: *CoRR* abs/2006.08100.

Questions?