EECE 571F: Deep Learning with Structures

Lecture 9: Energy-based Models (EBMs)

Renjie Liao

University of British Columbia

Winter, Term 1, 2023

Outline

- Classic EBMs
  - EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs
  - Inference: Gibbs Sampling
  - Learning: Contrastive Divergence
  - EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs
- Modern EBMs
  - EBMs with Learnable Energy Functions
  - Inference: Langevin Monte Carlo (LMC)
  - Learning: Contrastive Divergence

Energy-based Models (EBMs)

EBMs are a class of generative models, originated from statistical physics (e.g., statistical mechanics and thermodynamics).

Image Credit: [1]

Energy-based Models (EBMs)

EBMs are a class of generative models, originated from statistical physics (e.g., statistical mechanics and thermodynamics).

They are fundamental models in deep learning, e.g., Boltzmann Machines (BMs) [2, 3] and their variants.

Image Credit: [1]

Energy-based Models (EBMs)

EBMs are a class of generative models, originated from statistical physics (e.g., statistical mechanics and thermodynamics).

They are fundamental models in deep learning, e.g., Boltzmann Machines (BMs) [2, 3] and their variants.

It originates from Ludwig Boltzmann and his famous distribution (a.k.a., Boltzmann distribution):

$$p(X) \propto \exp\left(-\frac{E(X)}{kT}\right)$$

Image Credit: [1]

Energy-based Models (EBMs)

EBMs are a class of generative models, originated from statistical physics (e.g., statistical mechanics and thermodynamics).

They are fundamental models in deep learning, e.g., Boltzmann Machines (BMs) [2, 3] and their variants.

It originates from Ludwig Boltzmann and his famous distribution (a.k.a., Boltzmann distribution):

Energy of the system

$$p(X) \propto \exp\left(-\frac{E(X)}{kT}\right)$$

Boltzmann Constant

Temperature

Image Credit: [1]

Outline

- Classic EBMs
  - **EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs**
  - Inference: Gibbs Sampling
  - Learning: Contrastive Divergence
  - EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs
- Modern EBMs
  - EBMs with Learnable Energy Functions
  - Inference: Langevin Monte Carlo (LMC)
  - Learning: Contrastive Divergence

Discrete Observable and Latent Variables

EBMs with both discrete observable and latent variables are extensively studied in the literature, e.g., Boltzmann Machines (BMs) [2,3] and Restricted Boltzmann Machines (RBMs) [4,5].

Discrete Observable and Latent Variables

EBMs with both discrete observable and latent variables are extensively studied in the literature, e.g., Boltzmann Machines (BMs) [2,3] and Restricted Boltzmann Machines (RBMs) [4,5].

Specifically, the success of deep stacked RBMs [5] was the starting point of deep learning back in 2006!

Therefore, let us start with RBMs!

Restricted Boltzmann Machines

EBMs with both discrete observable and latent variables are extensively studied in the literature, e.g., Boltzmann Machines (BMs) [2,3] and Restricted Boltzmann Machines (RBMs) [4,5].

Specifically, the success of deep stacked RBMs [5] was the starting point of deep learning back in 2006!

Therefore, let us start with RBMs!

Suppose we have binary visible units (observable variables) x, binary hidden units (latent variables) h

- Energy function $E_\theta(x, h) = -a^\top x - b^\top h - x^\top W h$

Restricted Boltzmann Machines

EBMs with both discrete observable and latent variables are extensively studied in the literature, e.g., Boltzmann Machines (BMs) [2,3] and Restricted Boltzmann Machines (RBMs) [4,5].

Specifically, the success of deep stacked RBMs [5] was the starting point of deep learning back in 2006!

Therefore, let us start with RBMs!

Suppose we have binary visible units (observable variables) x, binary hidden units (latent variables) h

- Energy function $E_\theta(x, h) = -a^\top x - b^\top h - x^\top W h$
- Probability distribution $p_\theta(x, h) = \frac{1}{Z} \exp\left(-E_\theta(x, h)\right)$

Restricted Boltzmann Machines

EBMs with both discrete observable and latent variables are extensively studied in the literature, e.g., Boltzmann Machines (BMs) [2,3] and Restricted Boltzmann Machines (RBMs) [4,5].

Specifically, the success of deep stacked RBMs [5] was the starting point of deep learning back in 2006!

Therefore, let us start with RBMs!

Suppose we have binary visible units (observable variables) x, binary hidden units (latent variables) h

- Energy function $$E_\theta(x,h) = -a^\top x - b^\top h - x^\top W h$$
- Probability distribution $$p_\theta(x,h) = \frac{1}{Z}\exp\left(-E_\theta(x,h)\right) \qquad Z = \int\int \exp\left(-E_\theta(x,h)\right) dxdh$$

Partition function / Normalization constant

Restricted Boltzmann Machines

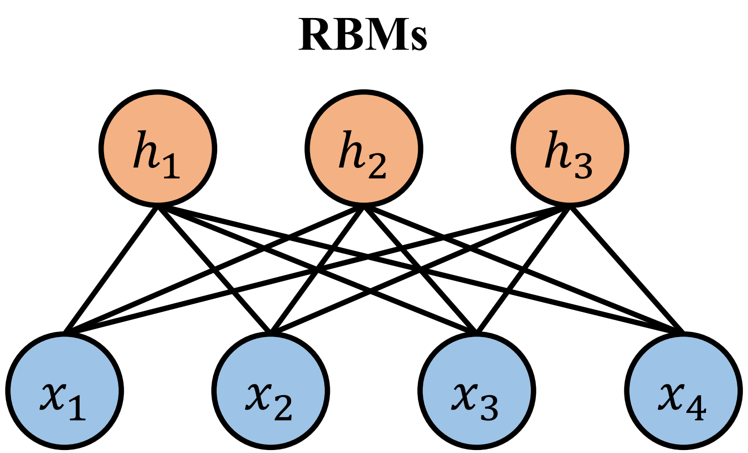
Binary visible units (observable variables) x, binary hidden units (latent variables) h

- Energy function $E_\theta(x, h) = -a^\top x - b^\top h - x^\top W h$

- Probability distribution

$$p_\theta(x, h) = \frac{1}{Z} \exp\left(-E_\theta(x, h)\right)$$

Bipartite Graphical Model



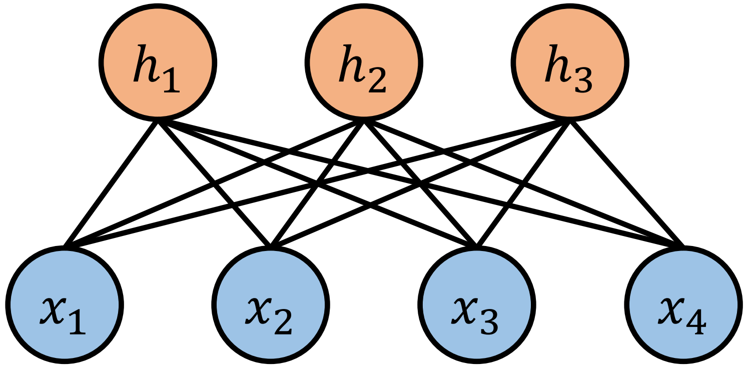
Restricted Boltzmann Machines

Binary visible units (observable variables) x, binary hidden units (latent variables) h

- Energy function $E_\theta(x, h) = -a^\top x - b^\top h - x^\top W h$

Bipartite Graphical Model

RBMs



- Probability distribution

$$p_\theta(x, h) = \frac{1}{Z} \exp\left(-E_\theta(x, h)\right)$$

- Bipartite graph structure implies conditional independence

$$p(h|x) = \prod\nolimits_j p(h_j|x)$$

$$p(x|h) = \prod\nolimits_i p(x_i|h)$$

Independent
Bernoulli distributions

Restricted Boltzmann Machines

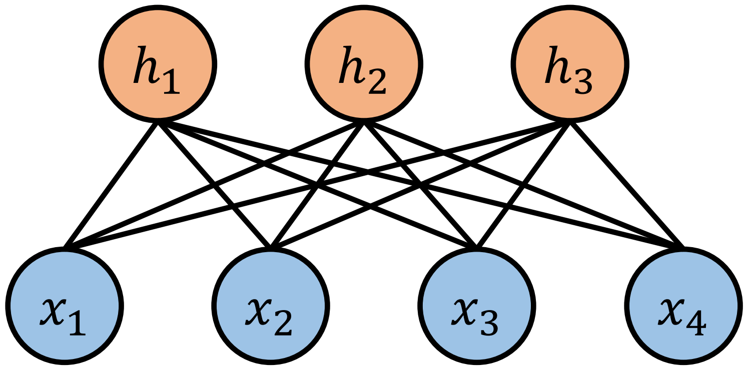
- Bipartite graph structure implies conditional independence

Why?

$$p(h|x) = \prod_j p(h_j|x)$$

$$p(x|h) = \prod_i p(x_i|h)$$

RBMs



Restricted Boltzmann Machines

- Bipartite graph structure implies conditional independence

$$p(h|x) = \prod_j p(h_j|x)$$

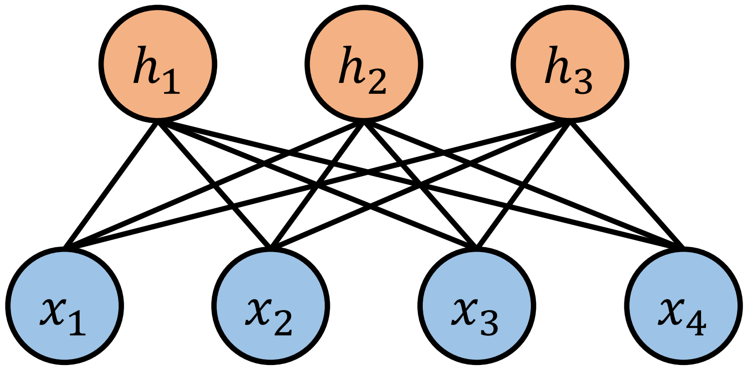
Why?

$$p(x|h) = \prod_i p(x_i|h)$$

Intuition:

- Observed visible units block the paths among hidden units
- Change of one hidden unit would not affect others

RBMs



Restricted Boltzmann Machines

- Bipartite graph structure implies conditional independence

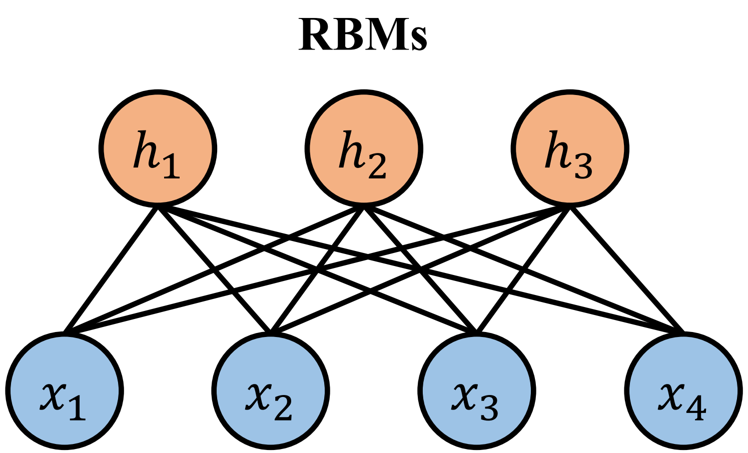
Why?

$$p(h|x) = \prod_j p(h_j|x)$$

$$p(x|h) = \prod_i p(x_i|h)$$

Intuition:

- Observed visible units block the paths among hidden units
- Change of one hidden unit would not affect others



Formally:

$$E_\theta(x, h) = -a^\top x - b^\top h - x^\top W h$$

$$p(x|h = \tilde{h}) \propto \exp\left(-E_\theta(x, h = \tilde{h})\right) \propto \exp\left(-\tilde{a}^\top x\right) = \prod_i \exp\left(-\tilde{a}_i x_i\right)$$

Outline

- Classic EBMs
  - EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs
  - **Inference: Gibbs Sampling**
  - Learning: Contrastive Divergence
  - EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs
- Modern EBMs
  - EBMs with Learnable Energy Functions
  - Inference: Langevin Monte Carlo (LMC)
  - Learning: Contrastive Divergence

Restricted Boltzmann Machines

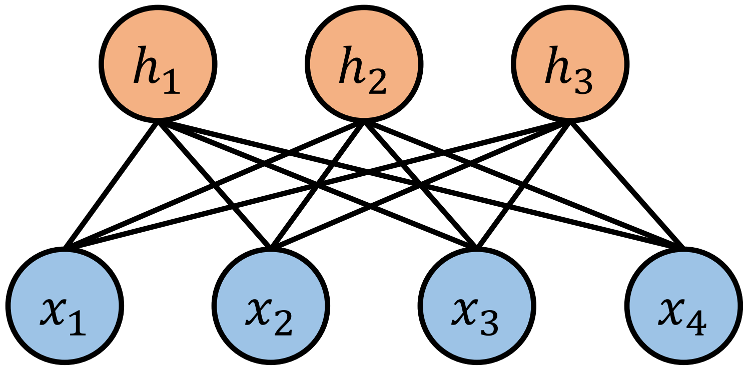
Inference: Computing Marginals $p(x)$ & Maximum A Posterior (MAP) $\arg\max_h p(h|x)$

- MAP is simple for RBMs due to the conditional independence.
- For computing marginals,

$$p(x) = \int \frac{1}{Z} \exp\left(-E(x,h)\right) dh$$

Intractable!

RBMs



Restricted Boltzmann Machines

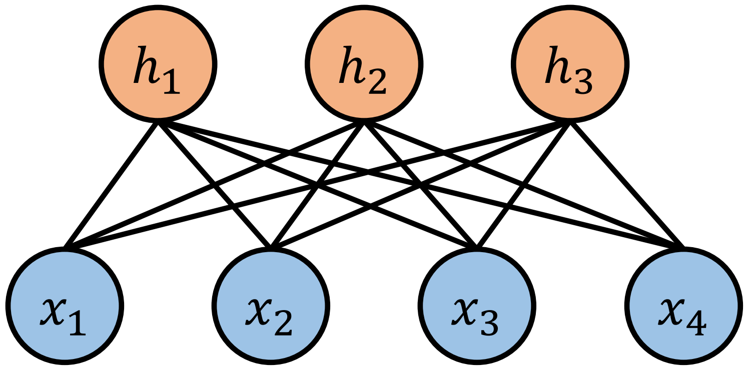
Inference: Computing Marginals $p(x)$ & Maximum A Posterior (MAP) $\arg\max_{h} p(h|x)$

- MAP is simple for RBMs due to the conditional independence.
- For computing marginals, we need Markov chain Monte Carlo, e.g., Gibbs sampling.

$$p(x) = \int \frac{1}{Z} \exp\left(-E(x,h)\right) dh$$

Intractable!

RBMs



Restricted Boltzmann Machines

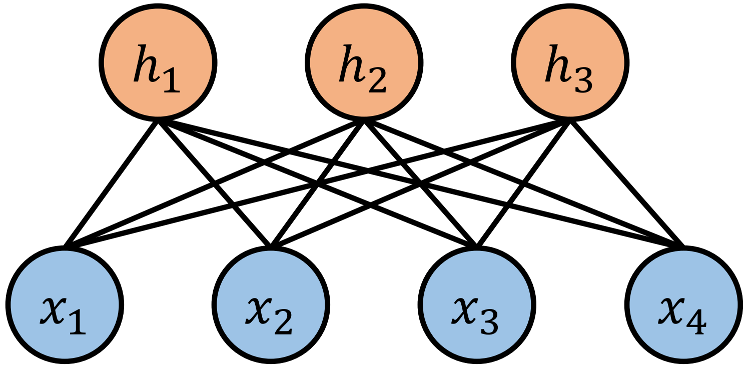
Inference: Computing Marginals $p(x)$ & Maximum A Posterior (MAP) $\arg\max_h p(h|x)$

- MAP is simple for RBMs due to the conditional independence.
- For computing marginals, we need Markov chain Monte Carlo, e.g., Gibbs sampling.

In general, Gibbs sampler draw samples from $p(x_1, x_2, \ldots, x_n)$ by iteratively sampling from the conditional distributions.

Given initial sample $(x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)})$
for $t = 1, \ldots, T$ **do**
  for $i = 1, \ldots, n$ **do**
    $x_i^{(t)} \sim p\left(x_i | x_1 = x_1^{(t)}, \ldots, x_{i-1} = x_{i-1}^{(t)}, x_{i+1} = x_{i+1}^{(t-1)}, \ldots, x_n = x_n^{(t-1)}\right)$
  end
end
Return $(x_1^{(T)}, x_2^{(T)}, \ldots, x_n^{(T)})$

RBMs



Restricted Boltzmann Machines

Inference: Computing Marginals $p(x)$ & Maximum A Posterior (MAP) $\arg\max_h p(h|x)$

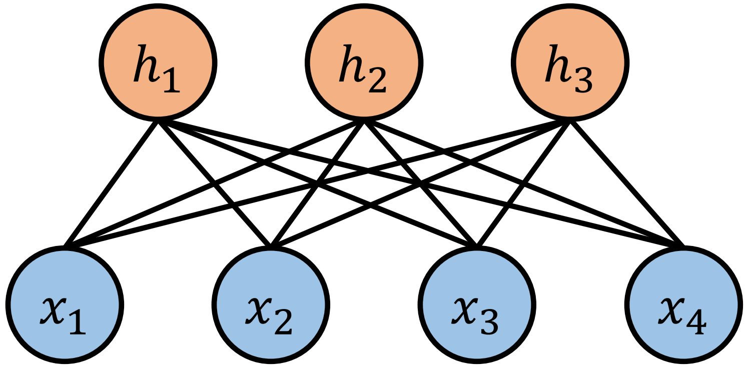
- MAP is simple for RBMs due to the conditional independence.
- For computing marginals, we need Markov chain Monte Carlo, e.g., Gibbs sampling

In general, Gibbs sampler draw samples from $p(x_1, x_2, \ldots, x_n)$ by iteratively sampling from the conditional distributions.

In RBMs, we do not iterative over individual variables. Instead, we do block-Gibbs sampling, i.e., sampling a block of variables conditioned on the other block.

Given initial sample $(x^{(0)}, h^{(0)})$
for $t = 1, \ldots, T$ **do**
  $h^{(t)} \sim p\left(h|x = x^{(t-1)}\right)$
  $x^{(t)} \sim p\left(x|h = h^{(t)}\right)$
end
Return $(x^{(T)}, h^{(T)})$

RBMs



Restricted Boltzmann Machines

Inference: Computing Marginals $p(x)$ & Maximum A Posterior (MAP) $\arg\max_h p(h|x)$

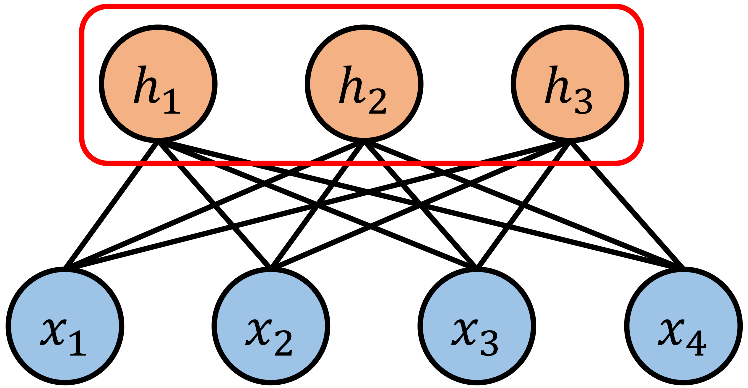
- MAP is simple for RBMs due to the conditional independence.
- For computing marginals, we need Markov chain Monte Carlo, e.g., Gibbs sampling

In general, Gibbs sampler draw samples from $p(x_1, x_2, \ldots, x_n)$ by iteratively sampling from the conditional distributions.

In RBMs, we do not iterative over individual variables. Instead, we do block-Gibbs sampling, i.e., sampling a block of variables conditioned on the other block.

Given initial sample $(x^{(0)}, h^{(0)})$
for $t = 1, \ldots, T$ **do**
  $h^{(t)} \sim p\left(h | x = x^{(t-1)}\right)$
  $x^{(t)} \sim p\left(x | h = h^{(t)}\right)$
end
Return $(x^{(T)}, h^{(T)})$

RBMs



Restricted Boltzmann Machines

Inference: Computing Marginals $p(x)$ & Maximum A Posterior (MAP) $\arg\max_h p(h|x)$

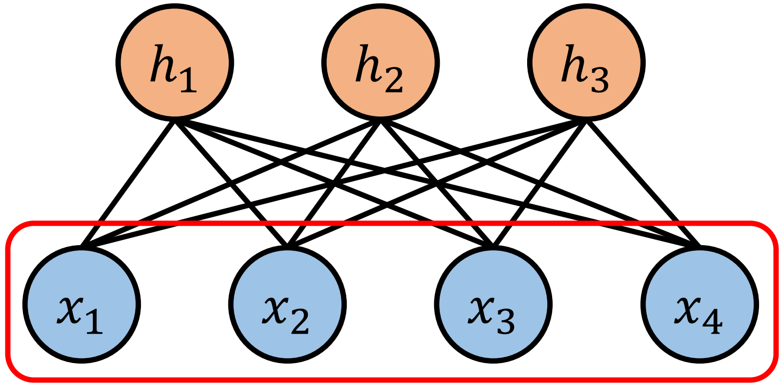
- MAP is simple for RBMs due to the conditional independence.
- For computing marginals, we need Markov chain Monte Carlo, e.g., Gibbs sampling

In general, Gibbs sampler draw samples from $p(x_1, x_2, \ldots, x_n)$ by iteratively sampling from the conditional distributions.

In RBMs, we do not iterative over individual variables. Instead, we do block-Gibbs sampling, i.e., sampling a block of variables conditioned on the other block.

Given initial sample $(x^{(0)}, h^{(0)})$
for $t = 1, \ldots, T$ **do**
  $h^{(t)} \sim p\left(h|x = x^{(t-1)}\right)$
  $x^{(t)} \sim p\left(x|h = h^{(t)}\right)$
end
Return $(x^{(T)}, h^{(T)})$

RBMs



Restricted Boltzmann Machines

Inference: Computing Marginals $p(x)$ & Maximum A Posterior (MAP) $\arg\max_h p(h|x)$

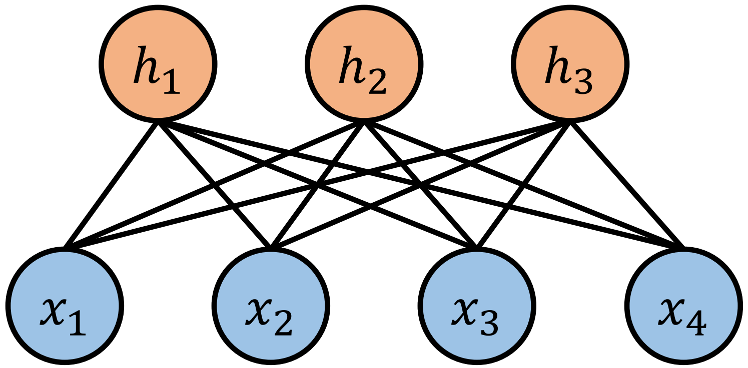
- MAP is simple for RBMs due to the conditional independence.
- For computing marginals, we need Markov chain Monte Carlo, e.g., Gibbs sampling

In general, Gibbs sampler draw samples from $p(x_1, x_2, \ldots, x_n)$ by iteratively sampling from the conditional distributions.

In RBMs, we do not iterative over individual variables. Instead, we do block-Gibbs sampling, i.e., sampling a block of variables conditioned on the other block.

Given initial sample $(x^{(0)}, h^{(0)})$
for $t = 1, \ldots, T$ **do**
  $h^{(t)} \sim p\left(h|x = x^{(t-1)}\right)$
  $x^{(t)} \sim p\left(x|h = h^{(t)}\right)$
end
Return $(x^{(T)}, h^{(T)})$

RBMs



The block-Gibbs shares the same convergence guarantee as Gibbs (due to conditional independence) but is more efficient due to parallel sampling!

Outline

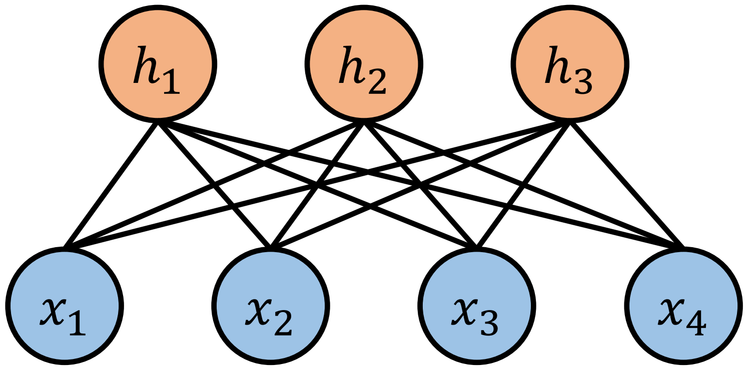
- Classic EBMs
  - EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs
  - Inference: Gibbs Sampling
  - **Learning: Contrastive Divergence**
  - EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs
- Modern EBMs
  - EBMs with Learnable Energy Functions
  - Inference: Langevin Monte Carlo (LMC)
  - Learning: Contrastive Divergence

Learning RBMs

Learning: Maximum Likelihood

$$\max_{\theta} \quad \log p_{\theta}(x)$$

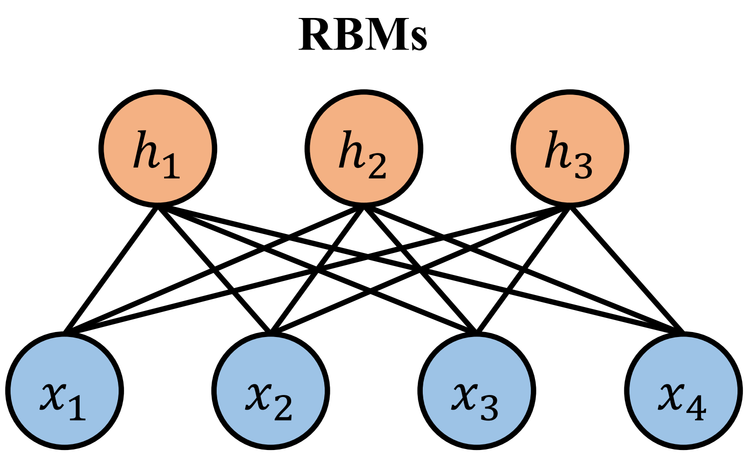
RBMs



Learning RBMs

Learning: Maximum Likelihood

$$\max_{\theta} \quad \log p_\theta(x) = \log \int p_\theta(x,h)dh$$
$$= \log \int \exp \log p_\theta(x,h)dh$$
$$= \log \int \exp\left(-E_\theta(x,h) - \log Z\right) dh$$



Learning RBMs

Learning: Maximum Likelihood

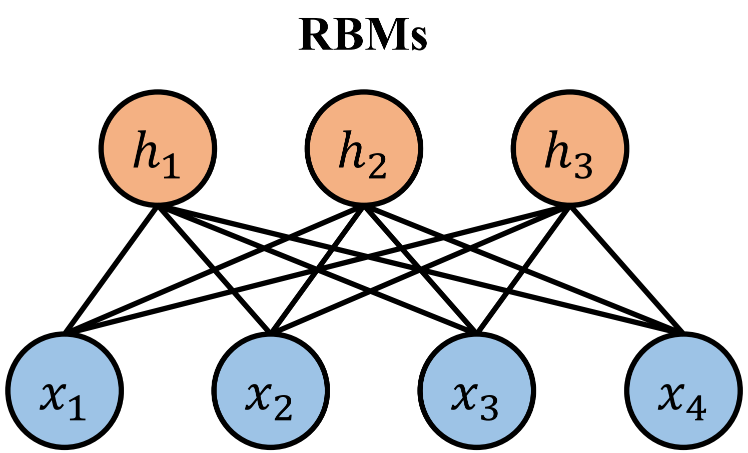
$$\max_{\theta} \quad \log p_\theta(x) = \log \int p_\theta(x,h)dh$$

$$= \log \int \exp \log p_\theta(x,h)dh$$

$$= \log \int \exp\left(-E_\theta(x,h) - \log \boxed{Z}\right) dh$$

Intractable!

$$Z = \int \int \exp\left(-E_\theta(x,h)\right) dx dh$$



Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$
\begin{aligned}
\frac{\partial \log p_\theta(x)}{\partial \theta} &= \frac{1}{p_\theta(x)} \frac{\partial p_\theta(x)}{\partial \theta} \\
&= \frac{1}{p_\theta(x)} \frac{\partial \int p_\theta(x,h) dh}{\partial \theta} \\
&= \frac{1}{p_\theta(x)} \int \frac{\partial p_\theta(x,h)}{\partial \theta} dh \\
&= \frac{1}{p_\theta(x)} \int \frac{\partial \frac{1}{Z} \exp\left(-E_\theta(x,h)\right)}{\partial \theta} dh \\
&= \frac{1}{p_\theta(x)} \int \frac{\left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) \exp\left(-E_\theta(x,h)\right) Z - \frac{\partial Z}{\partial \theta} \exp\left(-E_\theta(x,h)\right)}{Z^2} dh
\end{aligned}
$$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$
\begin{aligned}
\frac{\partial \log p_\theta(x)}{\partial \theta} &= \frac{1}{p_\theta(x)} \int \frac{\left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) \exp\left(-E_\theta(x,h)\right) Z - \frac{\partial Z}{\partial \theta} \exp\left(-E_\theta(x,h)\right)}{Z^2} dh \\
&= \frac{1}{p_\theta(x)} \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h) dh - \frac{1}{p_\theta(x)} \int \frac{1}{Z} \frac{\partial Z}{\partial \theta} p_\theta(x,h) dh \\
&= \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x) dh - \int \frac{1}{Z} \frac{\partial Z}{\partial \theta} p_\theta(h|x) dh \\
&= \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x) dh - \frac{1}{Z} \frac{\partial Z}{\partial \theta} \\
&= \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x) dh - \frac{1}{Z} \frac{\partial \int \int \exp\left(-E_\theta(x,h)\right) dx dh}{\partial \theta} \\
&= \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x) dh - \int \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h) dx dh
\end{aligned}
$$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$\begin{aligned}\frac{\partial \log p_\theta(x)}{\partial \theta} &= \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x)dh - \int\int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h)dxdh \\ &= \mathbb{E}_{p_\theta(h|x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]\end{aligned}$$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$\frac{\partial \log p_\theta(x)}{\partial \theta} = \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x)dh - \int\int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h)dxdh$$
$$= \mathbb{E}_{p_\theta(h|x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Recall we sample multiple training data and maximize the summed log likelihood of them, which in expectation amounts to:

Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$\frac{\partial \log p_\theta(x)}{\partial \theta} = \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x)dh - \int\int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h)dxdh$$
$$= \mathbb{E}_{p_\theta(h|x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Recall we sample multiple training data and maximize the summed log likelihood of them, which in expectation amounts to:

$$\min_\theta \quad \mathrm{KL}\left(p_{\text{data}}(x) \| p_\theta(x)\right) = \int p_{\text{data}}(x)\log p_{\text{data}}(x)dx - \int p_{\text{data}}(x)\log p_\theta(x)dx$$
$$= -\mathcal{H}_{p_{\text{data}}(x)} + \mathrm{CrossEntropy}(p_{\text{data}}(x), p_\theta(x))$$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$\frac{\partial \log p_\theta(x)}{\partial \theta} = \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x)dh - \int\int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h)dxdh$$
$$= \mathbb{E}_{p_\theta(h|x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Recall we sample multiple training data and maximize the summed log likelihood of them, which in expectation amounts to:

$$\min_\theta \quad \mathrm{KL}\left(p_{\text{data}}(x) \| p_\theta(x)\right) = \int p_{\text{data}}(x) \log p_{\text{data}}(x)dx - \int p_{\text{data}}(x) \log p_\theta(x)dx$$
$$= -\mathcal{H}_{p_{\text{data}}(x)} + \mathrm{CrossEntropy}(p_{\text{data}}(x), p_\theta(x))$$

$$\min_\theta \quad \mathrm{CrossEntropy}(p_{\text{data}}(x), p_\theta(x)) \quad \Leftrightarrow \quad \max_\theta \int p_{\text{data}}(x) \log p_\theta(x)dx$$

Maximum Likelihood

Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$
\begin{aligned}
\frac{\partial \log p_\theta(x)}{\partial \theta} &= \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x) dh - \int\int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h) dx dh \\
&= \mathbb{E}_{p_\theta(h|x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]
\end{aligned}
$$

Since we care about $\max_\theta \int p_{\text{data}}(x) \log p_\theta(x) dx$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

$$\frac{\partial \log p_\theta(x)}{\partial \theta} = \int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(h|x)dh - \int\int \left(-\frac{\partial E_\theta(x,h)}{\partial \theta}\right) p_\theta(x,h)dxdh$$
$$= \mathbb{E}_{p_\theta(h|x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Since we care about $\max_\theta \int p_{\text{data}}(x) \log p_\theta(x) dx$

we have the gradient

$$\int p_{\text{data}}(x) \frac{\partial \log p_\theta(x)}{\partial \theta} dx = \mathbb{E}_{p_\theta(h|x)p_{\text{data}}(x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

Stochastic Approximated Gradient

$$\int p_{\mathrm{data}}(x)\frac{\partial \log p_\theta(x)}{\partial \theta}dx = \mathbb{E}_{p_\theta(h|x)p_{\mathrm{data}}(x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Monte Carlo Estimation!

Stochastic Approximated Gradient

Learning: Maximum Likelihood

Stochastic Approximated Gradient

$$\int p_{\text{data}}(x) \frac{\partial \log p_\theta(x)}{\partial \theta} dx = \boxed{\mathbb{E}_{p_\theta(h|x) p_{\text{data}}(x)} \left[-\frac{\partial E_\theta(x,h)}{\partial \theta} \right]} - \mathbb{E}_{p_\theta(h,x)} \left[-\frac{\partial E_\theta(x,h)}{\partial \theta} \right]$$

Monte Carlo Estimation!

Positive Gradient: sample from the data distribution $p_\theta(h|x) p_{\text{data}}(x)$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

Stochastic Approximated Gradient

$$\int p_{\text{data}}(x) \frac{\partial \log p_\theta(x)}{\partial \theta} dx = \mathbb{E}_{p_\theta(h|x)p_{\text{data}}(x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Monte Carlo Estimation!

Positive Gradient: sample from the data distribution $p_\theta(h|x)p_{\text{data}}(x)$

Negative Gradient: sample from the model distribution $p_\theta(h,x)$

Stochastic Approximated Gradient

Learning: Maximum Likelihood

Stochastic Approximated Gradient

$$\int p_{\text{data}}(x) \frac{\partial \log p_\theta(x)}{\partial \theta} dx = \mathbb{E}_{p_\theta(h|x)p_{\text{data}}(x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right] - \mathbb{E}_{p_\theta(h,x)}\left[-\frac{\partial E_\theta(x,h)}{\partial \theta}\right]$$

Monte Carlo Estimation!

Positive Gradient: sample from the data distribution $p_\theta(h|x)p_{\text{data}}(x)$

Negative Gradient: sample from the model distribution $p_\theta(h,x)$

If we use finite-step Gibbs sampler, this method is called *Contrastive Divergence* (CD) [6]!

Outline

- Classic EBMs
  - EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs
  - Inference: Gibbs Sampling
  - Learning: Contrastive Divergence
  - **EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs**
- Modern EBMs
  - EBMs with Learnable Energy Functions
  - Inference: Langevin Monte Carlo (LMC)
  - Learning: Contrastive Divergence

Continuous Observable and Discrete Latent Variables

GRBMs: Gaussian-Bernoulli (a.k.a. Gaussian-Binary) Restricted Boltzmann Machines [7]

Continuous visible units (observable variables) $\mathbf{v}$, binary hidden units (latent variables) $\mathbf{h}$

Energy function: $E_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{2}\left(\frac{\mathbf{v}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^\top \left(\frac{\mathbf{v}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right) - \left(\frac{\mathbf{v}}{\boldsymbol{\sigma}^2}\right)^\top W\mathbf{h} - \mathbf{b}^\top\mathbf{h}$

Gaussian-Bernoulli Restricted Boltzmann Machines

GRBMs: Gaussian-Bernoulli (a.k.a. Gaussian-Binary) Restricted Boltzmann Machines [7]

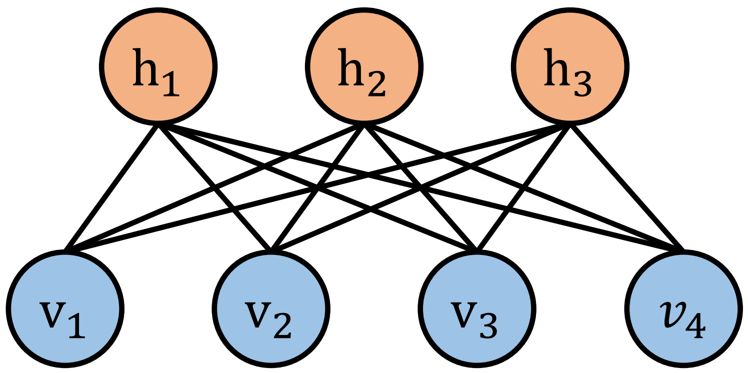
Continuous visible units (observable variables) $\mathbf{v}$, binary hidden units (latent variables) $\mathbf{h}$

Energy function: $$E_\theta(\mathbf{v},\mathbf{h}) = \frac{1}{2}\left(\frac{\mathbf{v}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^\top \left(\frac{\mathbf{v}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right) - \left(\frac{\mathbf{v}}{\boldsymbol{\sigma}^2}\right)^\top W\mathbf{h} - \mathbf{b}^\top\mathbf{h}$$

Conditional distributions (conditional independence holds again):

$$p(\mathbf{v}|\mathbf{h}) = \mathcal{N}\left(\mathbf{v}|W\mathbf{h}+\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)\right)$$
$$p(\mathbf{h}_j = 1|\mathbf{v}) = \left[\mathrm{Sigmoid}\left(W^\top \frac{\mathbf{v}}{\boldsymbol{\sigma}^2} + \mathbf{b}\right)\right]_j$$

GRBMs



Gaussian-Bernoulli Restricted Boltzmann Machines

GRBMs: Gaussian-Bernoulli (a.k.a. Gaussian-Binary) Restricted Boltzmann Machines [7]

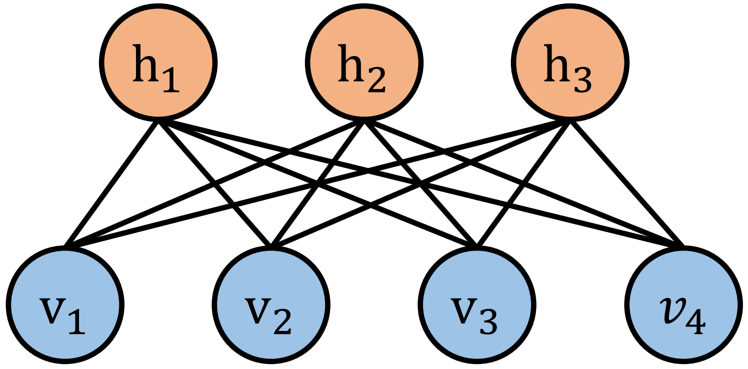
Continuous visible units (observable variables) $\mathbf{v}$, binary hidden units (latent variables) $\mathbf{h}$

Energy function: $E_\theta(\mathbf{v}, \mathbf{h}) = \frac{1}{2}\left(\frac{\mathbf{v}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right)^\top \left(\frac{\mathbf{v}-\boldsymbol{\mu}}{\boldsymbol{\sigma}}\right) - \left(\frac{\mathbf{v}}{\boldsymbol{\sigma}^2}\right)^\top W\mathbf{h} - \mathbf{b}^\top \mathbf{h}$

Conditional distributions (conditional independence holds again):

$$p(\mathbf{v}|\mathbf{h}) = \mathcal{N}\left(\mathbf{v}|W\mathbf{h} + \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)\right)$$
$$p(\mathbf{h}_j = 1|\mathbf{v}) = \left[\mathrm{Sigmoid}\left(W^\top \frac{\mathbf{v}}{\boldsymbol{\sigma}^2} + \mathbf{b}\right)\right]_j$$

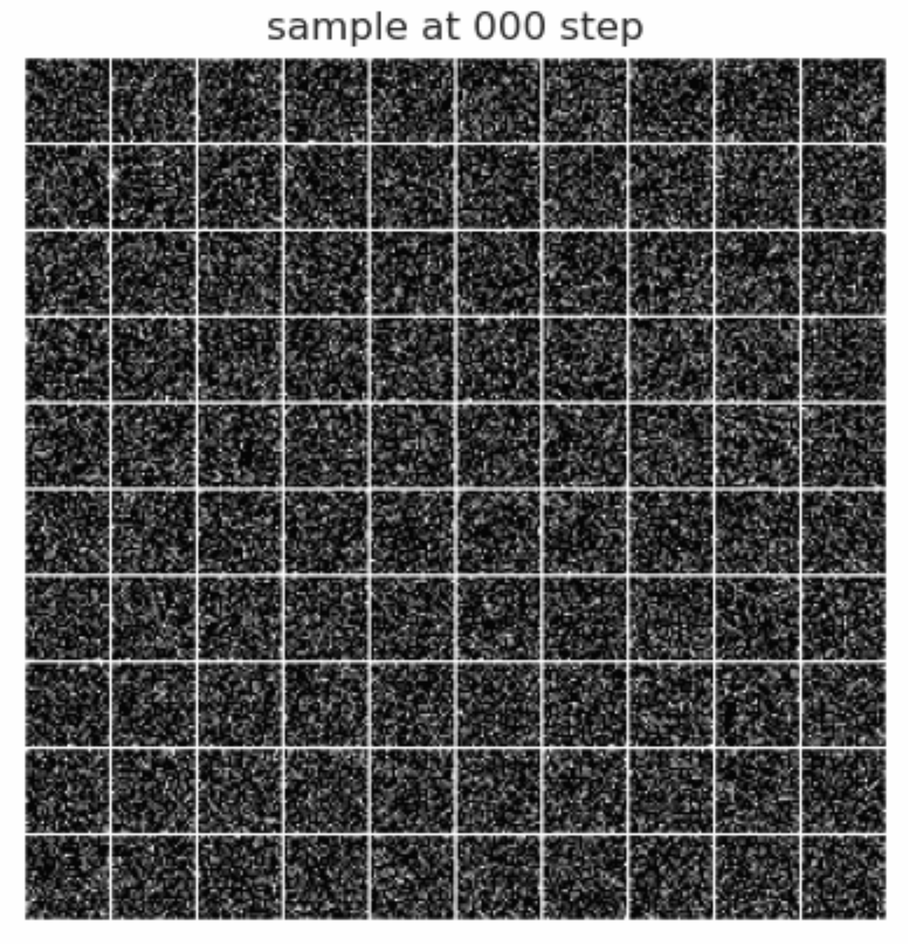
GRBMs



Recent work [8] introduces Gibbs-Langevin sampling, which makes CD-based learning work much better than before!

Gaussian-Bernoulli Restricted Boltzmann Machines

Results of training GRBMs for modelling MNIST Images [8]



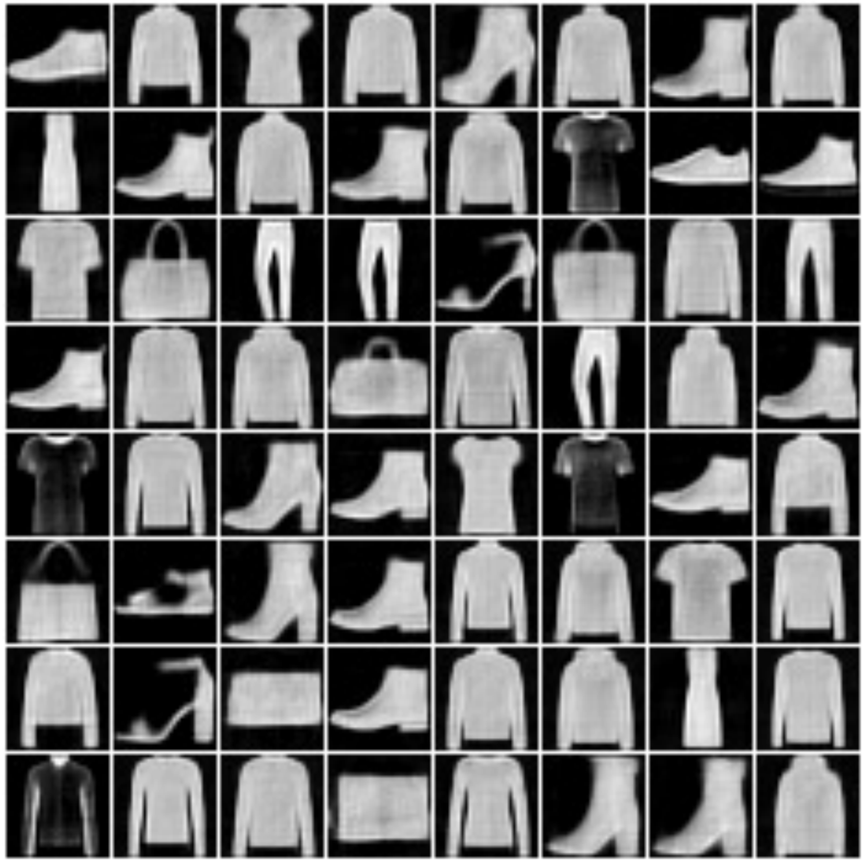
Methods	FID
VAE	16.13
2sVAE (Dai & Wipf, 2019)	12.60
PixelCNN++ (Salimans et al.)	11.38
WGAN (Arjovsky et al., 2017)	10.28
NVAE (Vahdat & Kautz, 2020)	7.93
GRBMs	
Gibbs	47.53
Langevin wo. Adjust	43.80
Langevin w. Adjust	41.24
Gibbs-Langevin wo. Adjust	17.49
Gibbs-Langevin w. Adjust	19.27

Table 1: Results on MNIST dataset.

Image Credit: [8]

Gaussian-Bernoulli Restricted Boltzmann Machines

Results of training GRBMs for modelling Fashion-MNIST and CelebA-32 Images [8]



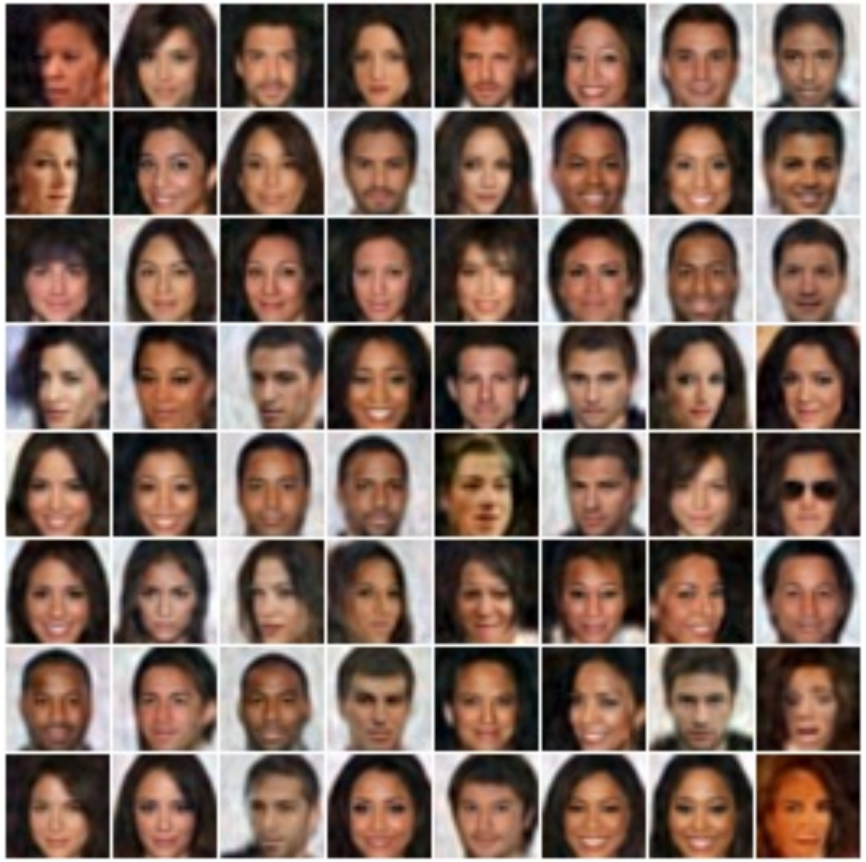


Image Credit: [8]

Outline

- Classic EBMs
  - EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs
  - Inference: Gibbs Sampling
  - Learning: Contrastive Divergence
  - EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs
- Modern EBMs
  - **EBMs with Learnable Energy Functions**
  - Inference: Langevin Monte Carlo (LMC)
  - Learning: Contrastive Divergence

Deep EBMs

Recall EBMs without latent variables are: $$p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$$

Deep EBMs

Recall EBMs without latent variables are: $$p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$$

We know energy function design is key to EBMs, e.g., RBM's energy function implies conditional independence.

However, it is typically hard to know how to design energy function in advance.

Deep EBMs

Recall EBMs without latent variables are: $p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$

We know energy function design is key to EBMs, e.g., RBM's energy function implies conditional independence.

However, it is typically hard to know how to design energy function in advance.

Why not learn the energy function from data?

Deep EBMs

Recall EBMs without latent variables are: $p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$

We know energy function design is key to EBMs, e.g., RBM's energy function implies conditional independence.

However, it is typically hard to know how to design energy function in advance.

Why not learn the energy function from data?

Yes! But there are at least two requirements on the parameterization of the energy function:

- It should be expressive enough to capture the complicated unnormalized probability density of data.
- It should be differentiable to enable CD-based learning.

Deep EBMs

Recall EBMs without latent variables are: $p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$

We know energy function design is key to EBMs, e.g., RBM's energy function implies conditional independence.

However, it is typically hard to know how to design energy function in advance.

Why not learn the energy function from data?

Yes! But there are at least two requirements on the parameterization of the energy function:

- It should be expressive enough to capture the complicated unnormalized probability density of data.
- It should be differentiable to enable CD-based learning.

We already have the answer, i.e., deep neural networks!

Deep EBMs

How to parameterize the energy function using deep neural networks?

$$p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$$

Deep EBMs

How to parameterize the energy function using deep neural networks?

$$p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$$

For image generation, U-Net architecture is crucial [9, 10].

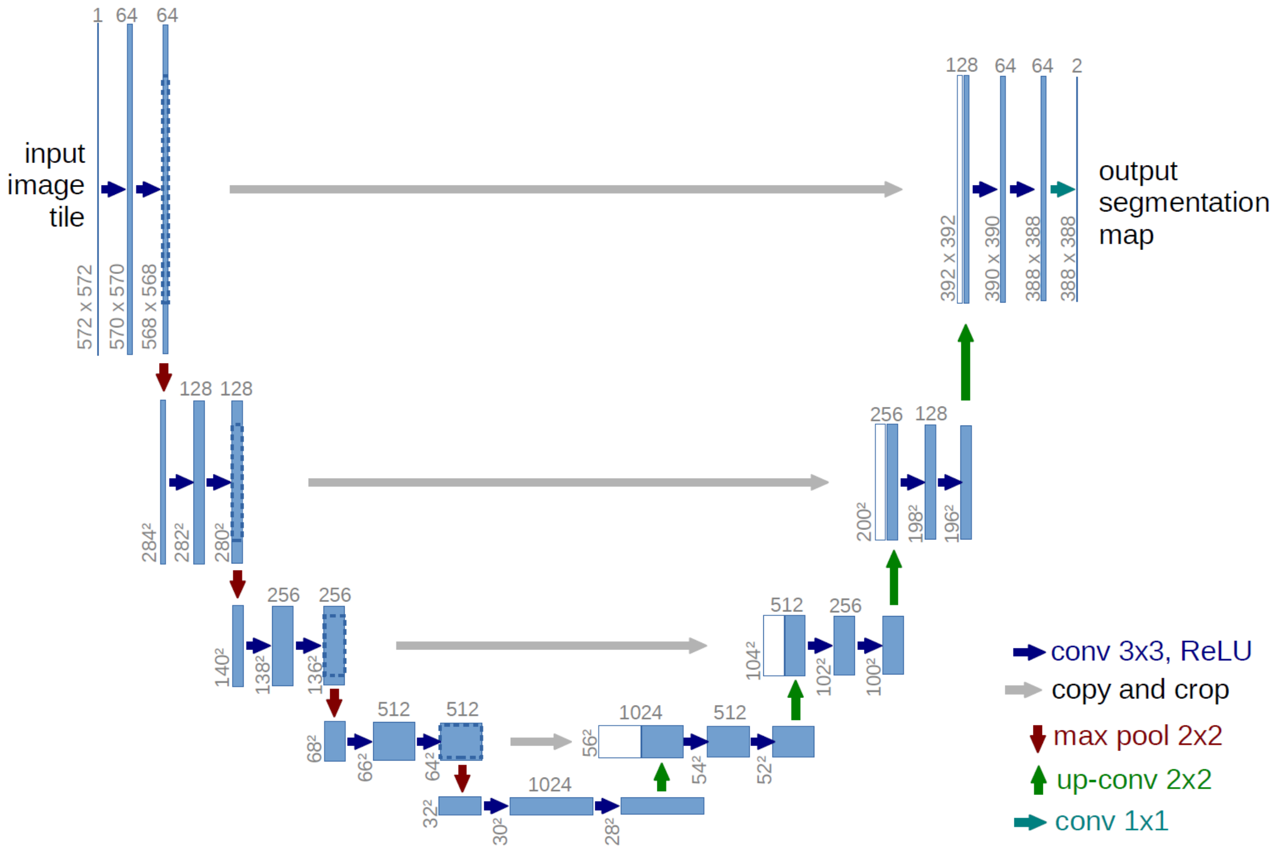


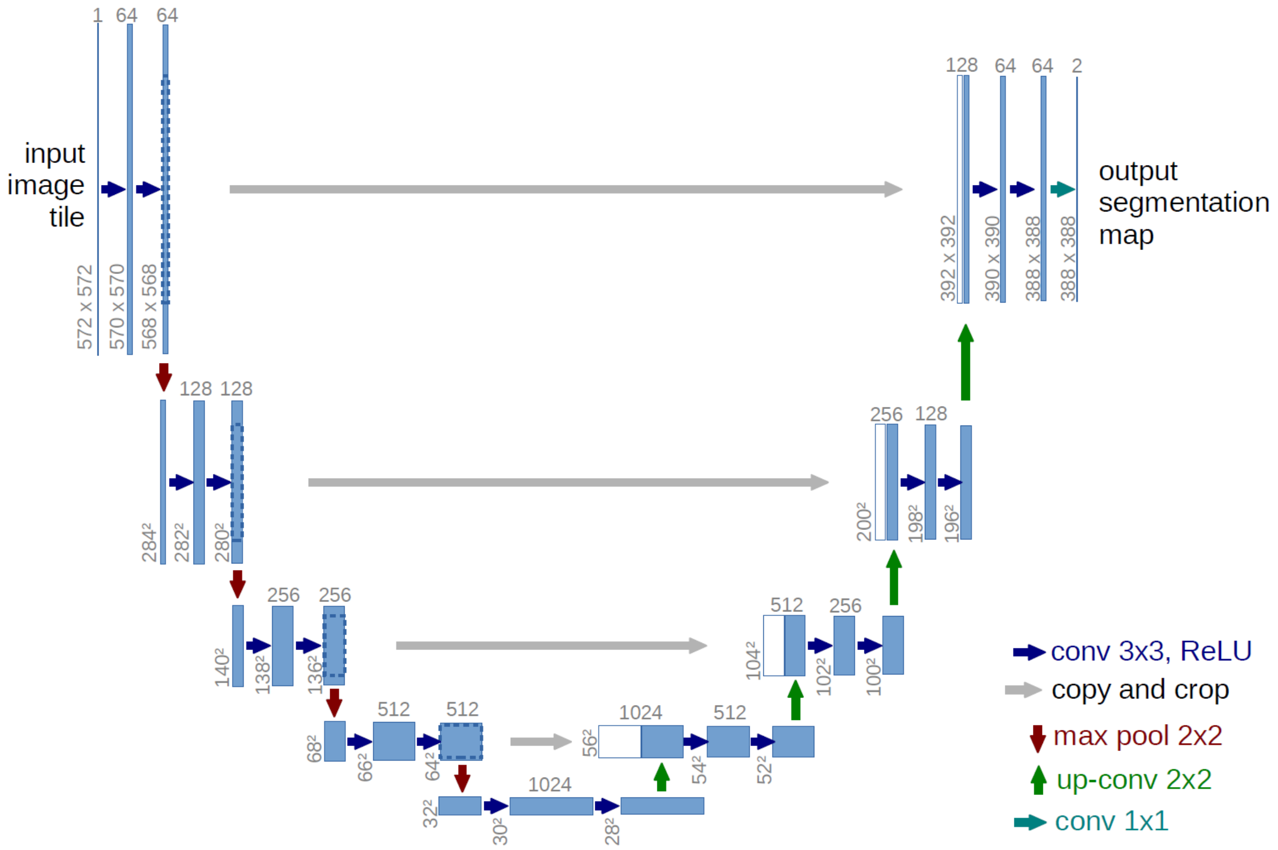
Image Credit: [11]

Deep EBMs

How to parameterize the energy function using deep neural networks?

$$p_\theta(\mathbf{x}) = \frac{1}{Z}\exp\left(-E_\theta(\mathbf{x})\right)$$

For image generation, U-Net architecture is crucial [9, 10].



Recall energy is a scalar, we have several design choices:

$$E_\theta(\mathbf{x}) = \mathbf{x}^T f_\theta(\mathbf{x})$$

$$E_\theta(\mathbf{x}) = (\mathbf{x} - f_\theta(\mathbf{x}))^2$$

$$E_\theta(\mathbf{x}) = f_\theta^2(\mathbf{x})$$

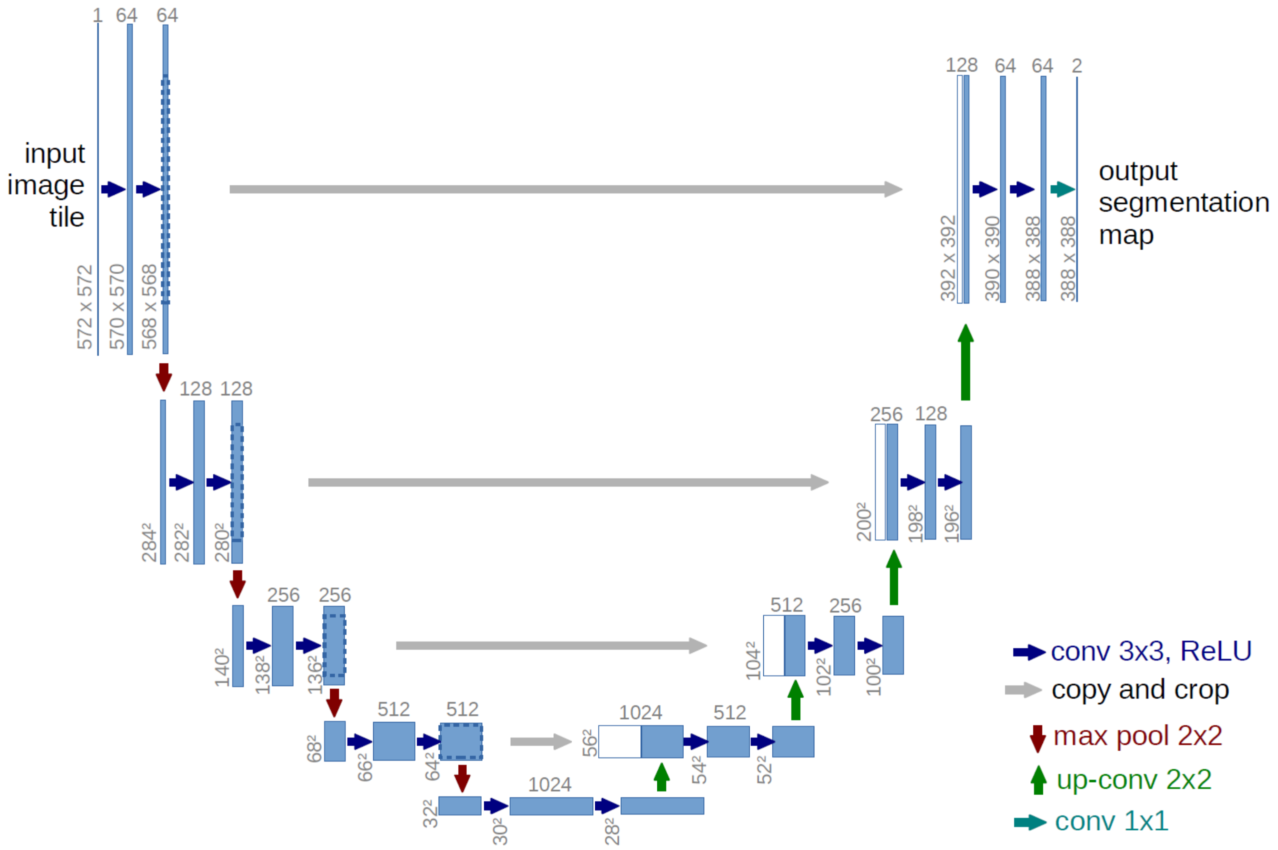
Image Credit: [11]

Deep EBMs

How to parameterize the energy function using deep neural networks?

$$p_\theta(\mathbf{x}) = \frac{1}{Z}\exp\left(-E_\theta(\mathbf{x})\right)$$

For image generation, U-Net architecture is crucial [9, 10].



Recall energy is a scalar, we have several design choices:

$$\boxed{E_\theta(\mathbf{x}) = \mathbf{x}^T f_\theta(\mathbf{x})}$$

$$E_\theta(\mathbf{x}) = (\mathbf{x} - f_\theta(\mathbf{x}))^2$$

$$E_\theta(\mathbf{x}) = f_\theta^2(\mathbf{x})$$

The inner-product version works the best empirically [10]!

Image Credit: [11]

Deep EBMs

We can also use deep EBMs for supervised learning tasks like classification [13,14].

$$p_\theta(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x}, \mathbf{y})\right)$$

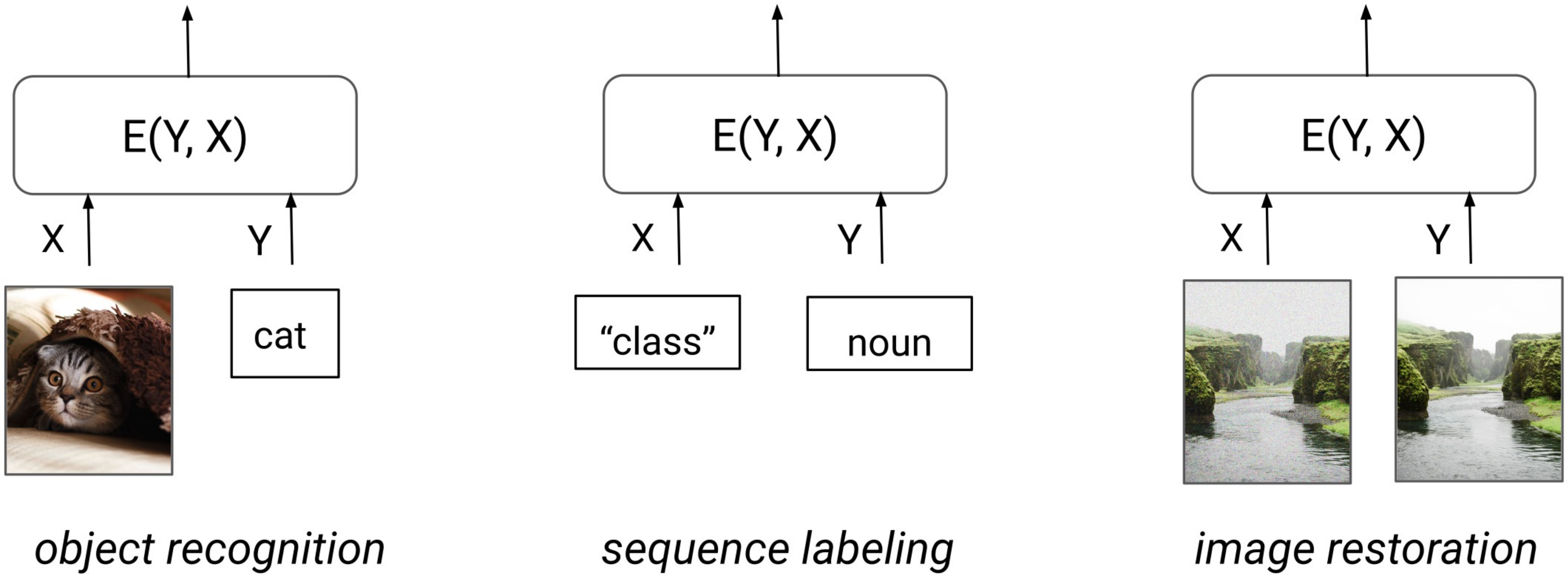


Image Credit: [12]

Outline

- Classic EBMs
  - EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs
  - Inference: Gibbs Sampling
  - Learning: Contrastive Divergence
  - EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs
- Modern EBMs
  - EBMs with Learnable Energy Functions
  - **Inference: Langevin Monte Carlo (LMC)**
  - Learning: Contrastive Divergence

Sampling from Deep EBMs

Suppose we have a deep EBM over continuous random variables, how can we draw samples from it?

$$p_\theta(\mathbf{x}) = \frac{1}{Z}\exp\left(-E_\theta(\mathbf{x})\right)$$

Langevin Monte Carlo

Suppose we have a deep EBM over continuous random variables, how can we draw samples from it?

$$p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$$

One popular approach is Langevin Monte Carlo [15,16] originated from Langevin Diffusion [17].

$$\mathrm{d}\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{2}\mathrm{d}B_t}_{\text{diffusion term}}$$

This is a stochastic differential equation (SDE), known as *Itô diffusion*.

Langevin Monte Carlo

Suppose we have a deep EBM over continuous random variables, how can we draw samples from it?

$$p_\theta(\mathbf{x}) = \frac{1}{Z}\exp\left(-E_\theta(\mathbf{x})\right)$$

One popular approach is Langevin Monte Carlo [15,16] originated from Langevin Diffusion [17].

$$\mathrm{d}\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{2}\mathrm{d}\boxed{B_t}}_{\text{diffusion term}}$$

standard Brownian motion

This is a stochastic differential equation (SDE), known as *Itô diffusion*.

Langevin Monte Carlo

Suppose we have a deep EBM over continuous random variables, how can we draw samples from it?

$$p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$$

One popular approach is Langevin Monte Carlo [15,16] originated from Langevin Diffusion [17].

$$\mathrm{d}\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{2}\mathrm{d}\boxed{B_t}}_{\text{diffusion term}}$$

standard Brownian motion

This is a stochastic differential equation (SDE), known as *Itô diffusion*.

One can prove Langevin Diffusion is *irreducible*, *strong Feller*, and *aperiodic* [18].

Langevin Monte Carlo

Suppose we have a deep EBM over continuous random variables, how can we draw samples from it?

$$p_\theta(\mathbf{x}) = \frac{1}{Z} \exp\left(-E_\theta(\mathbf{x})\right)$$

One popular approach is Langevin Monte Carlo [15,16] originated from Langevin Diffusion [17].

$$\mathrm{d}\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{2}\mathrm{d}\boxed{B_t}}_{\text{diffusion term}}$$

standard Brownian motion

This is a stochastic differential equation (SDE), known as *Itô diffusion*.

One can prove Langevin Diffusion is *irreducible*, *strong Feller*, and *aperiodic* [18].

In other words, $p_\theta(\mathbf{x})$ is the stationary distribution of Langevin Diffusion. Therefore, we can use it as a Markov chain Monte Carlo sampling method.

Langevin Monte Carlo

To turn the Langevin Diffusion into a sampling algorithm, we need to discretize (Euler-Maruyama method) it:

$$\mathrm{d}\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{2}\mathrm{d}B_t}_{\text{diffusion term}}$$

Langevin Monte Carlo

To turn the Langevin Diffusion into a sampling algorithm, we need to discretize (Euler-Maruyama method) it:

$$\mathrm{d}\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{2}\mathrm{d}B_t}_{\text{diffusion term}}$$

$$\Downarrow$$

$$\mathbf{x}_{t+\eta} - \mathbf{x}_t = \nabla \log p_\theta(\mathbf{x}_t)(t+\eta-t) + \sqrt{2}(B_{t+\eta} - B_t)$$

Langevin Monte Carlo

To turn the Langevin Diffusion into a sampling algorithm, we need to discretize (Euler-Maruyama method) it:

$$\mathrm{d}\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})\mathrm{d}t}_{\text{drift term}} + \underbrace{\sqrt{2}\mathrm{d}B_t}_{\text{diffusion term}}$$

⬇

$$\mathbf{x}_{t+\eta} - \mathbf{x}_t = \nabla \log p_\theta(\mathbf{x}_t)(t + \eta - t) + \sqrt{2}(B_{t+\eta} - B_t)$$

⬇

$$\mathbf{x}_{t+\eta} = \mathbf{x}_t + \eta \nabla \log p_\theta(\mathbf{x}_t) + \sqrt{2}\tilde{\epsilon}$$

Increments of Brownian motion satisfy:

$$\tilde{\epsilon} = B_{t+\eta} - B_t \sim \mathcal{N}(0, \eta I)$$

Langevin Monte Carlo

To turn the Langevin Diffusion into a sampling algorithm, we need to discretize (Euler-Maruyama method) it:

$$d\mathbf{x} = \underbrace{\nabla \log p_\theta(\mathbf{x})dt}_{\text{drift term}} + \underbrace{\sqrt{2}dB_t}_{\text{diffusion term}}$$

⇓

$$\mathbf{x}_{t+\eta} - \mathbf{x}_t = \nabla \log p_\theta(\mathbf{x}_t)(t + \eta - t) + \sqrt{2}(B_{t+\eta} - B_t)$$

⇓

Increments of Brownian motion satisfy:

$$\mathbf{x}_{t+\eta} = \mathbf{x}_t + \eta \nabla \log p_\theta(\mathbf{x}_t) + \sqrt{2}\tilde{\epsilon} \qquad \tilde{\epsilon} = B_{t+\eta} - B_t \sim \mathcal{N}(0, \eta I)$$

⇓

$$\mathbf{x}_{t+\eta} = \mathbf{x}_t + \eta \nabla \log p_\theta(\mathbf{x}_t) + \sqrt{2\eta}\epsilon \qquad \epsilon \sim \mathcal{N}(0, I)$$

Langevin Monte Carlo

We can construct the *Unadjusted Langevin Algorithm (ULA)* based on the Euler-Maruyama discretization:

$$\mathbf{x}_{t+\eta} = \mathbf{x}_t + \eta \nabla \log p_\theta(\mathbf{x}_t) + \sqrt{2\eta}\epsilon \qquad \epsilon \sim \mathcal{N}(0, I)$$

Given initial sample $\mathbf{x}_0$, step size η
for $t = 0, \ldots, T-1$ **do**
  $\epsilon_t \sim \mathcal{N}(0, I)$
  $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \nabla \log p_\theta(\mathbf{x}) + \sqrt{2\eta}\epsilon_t$
end
Return $\mathbf{x}_T$

Langevin Monte Carlo

We can construct the *Unadjusted Langevin Algorithm (ULA)* based on the Euler-Maruyama discretization:

$$\mathbf{x}_{t+\eta} = \mathbf{x}_t + \eta \nabla \log p_\theta(\mathbf{x}_t) + \sqrt{2\eta}\epsilon \qquad \epsilon \sim \mathcal{N}(0, I)$$

Given initial sample $\mathbf{x}_0$, step size η
for $t = 0, \ldots, T-1$ **do**
  $\epsilon_t \sim \mathcal{N}(0, I)$
  $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \boxed{\nabla \log p_\theta(\mathbf{x})} + \sqrt{2\eta}\epsilon_t$
end
Return $\mathbf{x}_T$

Score function (in ML):

$$\nabla \log p_\theta(\mathbf{x}) = \nabla \left(-E_\theta(\mathbf{x}) - \log Z\right) = -\nabla E_\theta(\mathbf{x})$$

Langevin Monte Carlo

We can construct the *Unadjusted Langevin Algorithm (ULA)* based on the Euler-Maruyama discretization:

$$\mathbf{x}_{t+\eta} = \mathbf{x}_t + \eta \nabla \log p_\theta(\mathbf{x}_t) + \sqrt{2\eta}\epsilon \qquad \epsilon \sim \mathcal{N}(0, I)$$

Given initial sample $\mathbf{x}_0$, step size η
for $t = 0, \ldots, T-1$ **do**
  $\epsilon_t \sim \mathcal{N}(0, I)$
  $\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \boxed{\nabla \log p_\theta(\mathbf{x})} + \sqrt{2\eta}\epsilon_t$
end
Return $\mathbf{x}_T$

Score function (in ML):

$$\nabla \log p_\theta(\mathbf{x}) = \nabla\left(-E_\theta(\mathbf{x}) - \log Z\right) = -\nabla E_\theta(\mathbf{x})$$

One can also perform Metropolis-Hasting to ensure *detailed balance*, which implies stationary distribution, leading to *Metropolis-adjusted Langevin Algorithm (MALA)*.

But the acceptance probability decreases as the dimension increases, making it impractical in deep learning.

Outline

- Classic EBMs
  - EBMs with Discrete Observable Variables and Discrete Latent Variables: RBMs
  - Inference: Gibbs Sampling
  - Learning: Contrastive Divergence
  - EBMs with Continuous Observable Variables and Discrete Latent Variables : GRBMs
- Modern EBMs
  - EBMs with Learnable Energy Functions
  - Inference: Langevin Monte Carlo (LMC)
  - **Learning: Contrastive Divergence**

Learning Deep EBMs

To learn deep EBMs, we still resort to maximum likelihood and contrastive divergence:

$$
\begin{aligned}
\frac{\partial \log p_\theta(x)}{\partial \theta} &= \frac{1}{p_\theta(x)} \frac{\partial p_\theta(x)}{\partial \theta} \\
&= \frac{1}{p_\theta(x)} \frac{\partial \frac{1}{Z} \exp\left(-E_\theta(x)\right)}{\partial \theta} \\
&= \frac{1}{p_\theta(x)} \frac{\left(-\frac{\partial E_\theta(x)}{\partial \theta}\right) \exp\left(-E_\theta(x)\right) Z - \frac{\partial Z}{\partial \theta} \exp\left(-E_\theta(x)\right)}{Z^2} \\
&= \frac{1}{p_\theta(x)} \left(-\frac{\partial E_\theta(x)}{\partial \theta}\right) p_\theta(x) - \frac{1}{p_\theta(x)} \frac{1}{Z} \frac{\partial Z}{\partial \theta} p_\theta(x) \\
&= -\frac{\partial E_\theta(x)}{\partial \theta} - \frac{1}{Z} \frac{\partial Z}{\partial \theta} \\
&= -\frac{\partial E_\theta(x)}{\partial \theta} - \frac{1}{Z} \frac{\partial \int \exp\left(-E_\theta(x)\right) dx}{\partial \theta} \\
&= -\frac{\partial E_\theta(x)}{\partial \theta} - \frac{1}{Z} \int \left(-\frac{\partial E_\theta(x)}{\partial \theta}\right) \exp\left(-E_\theta(x)\right) dx \\
&= -\frac{\partial E_\theta(x)}{\partial \theta} - \int \left(-\frac{\partial E_\theta(x)}{\partial \theta}\right) p_\theta(x) dx
\end{aligned}
$$

Learning Deep EBMs

Since we care about $$\max_\theta \int p_{\text{data}}(x) \log p_\theta(x) dx$$

Learning Deep EBMs

Since we care about $$\max_\theta \int p_{\text{data}}(x) \log p_\theta(x) dx$$

We have the gradient:

$$\begin{aligned}
\int p_{\text{data}}(x) \frac{\partial \log p_\theta(x)}{\partial \theta} dx &= \int p_{\text{data}}(x) \left(-\frac{\partial E_\theta(x)}{\partial \theta} - \int \left(-\frac{\partial E_\theta(x)}{\partial \theta} \right) p_\theta(x) dx \right) dx \\
&= \mathbb{E}_{p_{\text{data}}(x)} \left[-\frac{\partial E_\theta(x)}{\partial \theta} \right] - \mathbb{E}_{p_\theta(x)} \left[-\frac{\partial E_\theta(x)}{\partial \theta} \right]
\end{aligned}$$

Learning Deep EBMs

Since we care about $$\max_{\theta} \int p_{\text{data}}(x) \log p_\theta(x) dx$$

We have the gradient:

$$\int p_{\text{data}}(x) \frac{\partial \log p_\theta(x)}{\partial \theta} dx = \int p_{\text{data}}(x) \left(-\frac{\partial E_\theta(x)}{\partial \theta} - \int \left(-\frac{\partial E_\theta(x)}{\partial \theta} \right) p_\theta(x) dx \right) dx$$
$$= \mathbb{E}_{p_{\text{data}}(x)} \left[-\frac{\partial E_\theta(x)}{\partial \theta} \right] - \mathbb{E}_{p_\theta(x)} \left[-\frac{\partial E_\theta(x)}{\partial \theta} \right]$$

Positive Gradient: sample from the data distribution

Negative Gradient: sample from the model distribution

We can still use *Contrastive Divergence* (CD) [6], with Langevin Monte Carlo sampling.

Inference & Learning Deep EBMs

In summary, we need score function (derivatives of energy w.r.t. data) in sampling:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \left(-\frac{\partial E_\theta(\mathbf{x})}{\partial \mathbf{x}} \right)_{\mathbf{x}_t} + \sqrt{2\eta}\epsilon$$

Inference & Learning Deep EBMs

In summary, we need score function (derivatives of energy w.r.t. data) in sampling:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \left(-\frac{\partial E_\theta(\mathbf{x})}{\partial \mathbf{x}} \right)_{\mathbf{x}_t} + \sqrt{2\eta}\epsilon$$

Inference & Learning Deep EBMs

In summary, we need score function (derivatives of energy w.r.t. data) in sampling:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \left(-\frac{\partial E_\theta(\mathbf{x})}{\partial \mathbf{x}} \right)_{\mathbf{x}_t} + \sqrt{2\eta}\epsilon$$

We need score function and derivatives of energy w.r.t. parameters in learning:

$$\theta_{t+1} = \theta_t + \beta \left(\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[-\frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \right]_{\theta_t} - \mathbb{E}_{p_\theta(\mathbf{x})} \left[-\frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \right]_{\theta_t} \right)$$

Inference & Learning Deep EBMs

In summary, we need score function (derivatives of energy w.r.t. data) in sampling:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \left(-\frac{\partial E_\theta(\mathbf{x})}{\partial \mathbf{x}} \right)_{\mathbf{x}_t} + \sqrt{2\eta}\epsilon$$

We need score function and derivatives of energy w.r.t. parameters in learning:

$$\theta_{t+1} = \theta_t + \beta \left(\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[-\frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \right]_{\theta_t} - \mathbb{E}_{p_\theta(\mathbf{x})} \left[-\frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \right]_{\theta_t} \right)$$

Inference & Learning Deep EBMs

In summary, we need score function (derivatives of energy w.r.t. data) in sampling:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + \eta \left(-\frac{\partial E_\theta(\mathbf{x})}{\partial \mathbf{x}} \right)_{\mathbf{x}_t} + \sqrt{2\eta}\epsilon$$

We need score function and derivatives of energy w.r.t. parameters in learning:

$$\theta_{t+1} = \theta_t + \beta \left(\mathbb{E}_{p_{\text{data}}(\mathbf{x})} \left[-\frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \right]_{\theta_t} - \mathbb{E}_{p_\theta(\mathbf{x})} \left[-\frac{\partial E_\theta(\mathbf{x})}{\partial \theta} \right]_{\theta_t} \right)$$

They are available as long as the energy function is differentiable!

Image Generation of Deep EBMs

Results on CIFAR10 and LSUN datasets [19]

Image Completion of Deep EBMs

Results on LSUN and CelebA [19]:

References

[1] https://en.wikipedia.org/wiki/Ludwig_Boltzmann

[2] Hinton, G.E. and Sejnowski, T.J., 1983, May. Analyzing cooperative computation. In Proceedings of the fifth annual conference of the cognitive science society (pp. 2554-2558).

[3] Hinton, G.E. and Sejnowski, T.J., 1983, June. Optimal perceptual inference. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (Vol. 448, pp. 448-453).

[4] Smolensky, P., 1986. Information processing in dynamical systems: Foundations of harmony theory. Colorado Univ at Boulder Dept of Computer Science.

[5] Hinton, G.E. and Salakhutdinov, R.R., 2006. Reducing the dimensionality of data with neural networks. science, 313(5786), pp.504-507.

[6] Hinton, G.E., 2002. Training products of experts by minimizing contrastive divergence. Neural computation, 14(8), pp.1771-1800.

[7] Welling, M., Rosen-Zvi, M. and Hinton, G.E., 2004. Exponential family harmoniums with an application to information retrieval. Advances in neural information processing systems, 17.

[8] Liao, R., Kornblith, S., Ren, M., Fleet, D.J. and Hinton, G., 2022. Gaussian-bernoulli rbms without tears. arXiv preprint arXiv:2210.10318.

[9] Salimans, T. and Ho, J., 2021, April. Should EBMs model the energy or the score?. In Energy Based Models Workshop-ICLR 2021.

[10] Liang, Z.Y, Vicol, P. and Liao, R., 2023. Training Energy-Based Models with Denoising Diffusion Herding. Under Review.

[11] https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/

References

[12] https://uvadlc-notebooks.readthedocs.io/en/latest/tutorial_notebooks/tutorial8/Deep_Energy_Models.html

[13] Du, Y. and Mordatch, I., 2019. Implicit generation and generalization in energy-based models. arXiv preprint arXiv:1903.08689.

[14] Grathwohl, W., Wang, K.C., Jacobsen, J.H., Duvenaud, D., Norouzi, M. and Swersky, K., 2019. Your classifier is secretly an energy based model and you should treat it like one. arXiv preprint arXiv:1912.03263.

[15] Roberts, G.O. and Tweedie, R.L., 1996. Exponential convergence of Langevin distributions and their discrete approximations. Bernoulli, pp.341-363.

[16] Karagulyan, A., 2021. Sampling with the Langevin Monte-Carlo (Doctoral dissertation, Institut polytechnique de Paris).

[17] Langevin, P. (1908). "Sur la théorie du mouvement brownien [On the Theory of Brownian Motion]". C. R. Acad. Sci. Paris. 146: 530–533.

[18] Bhattacharya, R.N., 1978. Criteria for recurrence and existence of invariant measures for multidimensional diffusions. The Annals of Probability, pp.541-553.

[19] Gao, R., Song, Y., Poole, B., Wu, Y.N. and Kingma, D.P., 2020. Learning energy-based models by diffusion recovery likelihood. arXiv preprint arXiv:2012.08125.

Questions?