Langevin Autoencoders for Learning Deep Latent Variable Models

IE 506 Course Project

Team Name: Supernova

Members

Tarun Bisht

Indian Institute of Technology Bombay

IEOR

March 24, 2023

Presentation Outline

Introduction

Motivation of the Study

- In Unsupervised learning, the main task is to learn useful representations from unlabeled data.
- These tasks include finding the underlying structure or interesting patterns, clustering, statistical correlations, and generating data.
- It is harder than a supervised learning task as instead of just predicting the target label, now the model has to learn to describe the input itself.
- Supervised Learning requires a large labeled dataset to generalize better, but obtaining a large labeled dataset is costly
- An unsupervised that explains data well can be used for semi-supervised learning with the small labeled dataset.

Background

- One of the goals of unsupervised learning is to model the distribution of a given dataset $x \sim D$, i.e., model distribution $p(x)$ such that $p(x) \sim p_D$. Here x contains samples from unknown distributions D. These models are called generative models.
- After we have learned the distribution we can
  - find probability of arbitary data point x, $p(x)$
  - sample point x from distribution, $x \sim p(x)$.
- To generate those data points, the model has to learn to analyze and understand the essence of the dataset. This ability makes these models to be used as learning representations from data.

Applications

- Representation Learning
- Mapping of one domain into another ex. Text to Speech, Text to Image etc.
- Conditional Synthesis ex. generating music, image based on text prompt etc.
- Generate novel data

Training a generative model

- Recall, the goal is to model p_D using samples $(x_1, x_2, ...x_m)$
- One approach to learn this distribution is by using function approximation. Learn θ such that $p_\theta(x) \approx p_D(x)$
- Optimal value of θ can be find using Maximum Likelihood estimation. Given dataset x find θ that maximizes probability of data.

$$\underset{\theta}{\text{argmax}}\ p_\theta(x)$$

$$\underset{\theta}{\text{argmax}}\ \log p_\theta(x)$$

$$\underset{\theta}{\text{argmin}}\ -\log p_\theta(x)$$

Background

Types of Generative models

- **Implicit Density Models** maps the dataset's underlying distribution without explicitly computing it. Ex. GAN
- **Explicit Density Models** computes the dataset's exact or approximate distribution functions. These models are generally referred to as Latent Variable Models. Ex. VAE, LAE
- **Autoregressive Models** model 1-dimensional conditional distribution. $p(x) = \prod_{i=1}^{D} p(x_i|x_1, x_2, ..., x_{i-1})$. Ex. GPT (generative pre-trained transformers).

Types of Generative models

- **Implicit Density Models** maps the dataset's underlying distribution without explicitly computing it. Ex. GAN
- **Explicit Density Models** computes the dataset's exact or approximate distribution functions. These models are generally referred to as Latent Variable Models. Ex. VAE, LAE
- **Autoregressive Models** model 1-dimensional conditional distribution. $p(x) = \prod_{i=1}^{D} p(x_i|x_1, x_2, ..., x_{i-1})$. Ex. GPT (generative pre-trained transformers).

We will focus only on Latent Variable models in this presentation.

Latent Variable Models

- We know that given dataset $x \sim D$, the generative model aims to model distribution $p(x)$.
- Idea of the latent variable model is instead of modeling $p(x)$ directly, an unobserved/hidden variable called latent variable is used, and we define a conditional distribution $p(x|z)$ for data also known as likelihood.
- The idea of latent variable is that in data can be represented using simple and lower dimensional representations, we hope that latent variables will try to find those hidden representations.
- The latent variable z can be continuous or discrete based on the models. Ex. VAE, LAE uses z as continuous while models like Vector Quantized VAE (VQ-VAE) assume z as discrete.
- We also introduce a prior distribution $p(z)$ over the latent variables, and compute the joint distribution over observed and latent variables as $p(x, z)$

Latent Variable Models

Evaluate Likelihood

Calculate likelihood of a sample

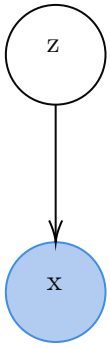
$$p(x) = \sum_z p(z)p(x|z)$$

Sample (Generation)

Generate new data points, given prior distribution of latent variable $p(z)$

$$z \sim p(z)$$

$$x \sim p(x|z)$$



Latent Variable Models

Data Representation (Inference)

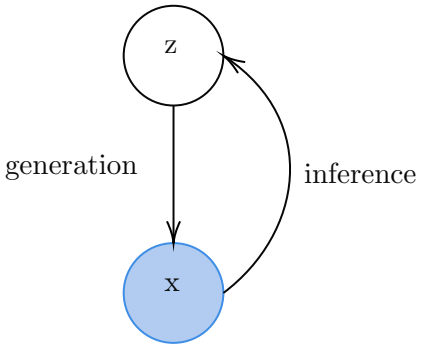
Representing a sample in lower dimension.

$$x \mapsto z$$

Inference is finding latent variable z given data point x and is formulated by posterior distribution $p(z|x)$

$$x \sim p(x)$$

$$z \sim p(z|x)$$



Latent Variable Models

- Using Bayes Rule we can connect inference and generation as,

$$p(z|x) = \frac{p(x|z)p(z)}{p(x)} = \frac{p(x,z)}{\int p(x,z)dz}$$

- $p(x|z)p(z) = p(x,z) = p(z|x)p(x)$

Training Latent Variable Models

- Training of model is done by Maximum Likelihood estimation.

$$\hat{\theta} = \underset{\theta}{\text{argmin}} \; -\sum_{i=1}^{m} \log p_\theta(x^i)$$

- This is a standard optimization problem and can be solved using gradient descent based algorithms. To apply gradient descent, gradients of objective are required with respect to model parameters.

Training Latent Variable Models

- Let's compute gradient for a single data point x

$$\nabla_\theta \log p_\theta(x) = \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)} = \frac{\int \nabla_\theta p_\theta(x,z)dz}{p_\theta(x)}$$

$$\nabla_\theta \log p_\theta(x) = \frac{\int p_\theta(x,z)\nabla_\theta \log p_\theta(x,z)dz}{p_\theta(x)}$$

using, $\nabla_\theta \log p_\theta(x) = \frac{\nabla_\theta p_\theta(x)}{p_\theta(x)}$

$$\nabla_\theta \log p_\theta(x) = \int p_\theta(z|x)\nabla_\theta \log p_\theta(x,z)dz$$

- We need to compute posterior $p(z|x)$ to compute gradient.

Training Latent Variable Models

- The posterior calculation is intractable due to the lack of an analytic solution to the integral.
- We try to approximate the posterior distribution, there are two classes of methods for this:
  - **Variational Inference** approximate the posterior with a tractable distribution. Ex. Variational Autoencoder
  - **Markov Chain Monte Carlo(MCMC)** provide sample based approximation of posterior distribution. It does it by constructing a markov chain in such a way that its stationary distribution is similar to posterior distribution. Ex. Langevin Autoencoder

Training Latent Variable Models

- The posterior calculation is intractable due to the lack of an analytic solution to the integral.
- We try to approximate the posterior distribution, there are two classes of methods for this:
  - **Variational Inference** approximate the posterior with a tractable distribution. Ex. Variational Autoencoder (VAE)
  - **Markov Chain Monte Carlo(MCMC)** provide sample based approximation of posterior distribution. Ex. Langevin Autoencoder (LAE)

This paper uses MCMC based method (Langevin Dynamics) to approximate posterior.

Langevin Dynamics Autoencoder

Langevin Dynamics

- Langevin Monte Carlo is a class of Markov Chain Monte Carlo (MCMC) algorithm that generate samples from a probability distribution of interest by simulating the Langevin Equation, which is inspired from physics.

$$dz = -\nabla_z U(x, z; \theta)dt + \sqrt{2\beta^{-1}}dB$$

  where U is some potential/energy function in our case its a loss function and $\sqrt{2\beta^{-1}}dB$ is noise term, B is brownian motion.

- The above stochastic differential(SDE) equation has a stationary distribution (SD), if we somehow make posterior distribution same as SD of above SDE then we can sample points from posterior by simulating above equation.

Claim: This stocastic differential equation has $p^{\beta}(z|x;\theta) \propto exp(-\beta U(x, z; \theta))$

Langevin Dynamics

- By setting $\beta = 1$ we have, $p(z|x;\theta) \propto exp(-U(x,z;\theta))$

$$exp(-U(x,z;\theta)) = p(z|x;\theta)p(x)$$

- Therefore, by setting $U(x,z;\theta) = -\log p(z|x;\theta)p(x)$ we can obtain posterior $p(z|x;\theta)$ as stationary distribution.

$$U(x,z;\theta) = -\log p(x,z;\theta)$$

- We are now interested in simulating the following SDE to sample from its steady state distribution $p(z|x;\theta)$

$$dz = -\nabla_z \log p(x,z;\theta)dt + \sqrt{2}dB$$

Langevin Dynamics

- We can obtain samples from the posterior by simulating the above SDE using the Euler–Maruyama method.

$$z_{t+\eta} - z_t = -\eta \nabla_z \log p(x, z; \theta) + \sqrt{2}(B_{t+\eta} - B_t)$$

here $(B_{t+\eta} - B_t)$ becomes, $(B_{t+\eta} - B_t) \sim N(0, \eta)$

$$z_{t+\eta} = z_t - \eta \nabla_z \log p(x, z; \theta) + \sqrt{2}(B_{t+\eta} - B_t)$$

$$z_{t+1} \sim q(z_{t+1}|z_t)$$

$$q(z'|z) = \mathcal{N}(z - \eta \nabla_z U(x, z; \theta), 2\eta I)$$

Estimating gradients

- After obtaining samples the gradient $\nabla_\theta \log p_\theta(x, z)$ is approximated as

$$\int p_\theta(z|x) \nabla_\theta \log p_\theta(x, z) dz = \mathbb{E}_{p_\theta(z|x)}[\nabla_\theta \log p_\theta(x, z)] \approx \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log p_\theta(x, z)$$

- For N samples gradients are calculated as

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{p_\theta(z^i|x^i)}[\nabla_\theta \log p_\theta(x^i, z^i)] \approx \frac{1}{N} \sum_{i=1}^{N} \frac{1}{T} \sum_{t=1}^{T} \nabla_\theta \log p_\theta(x^i, z^i)$$

Algorithm 2 Langevin Autoencoders

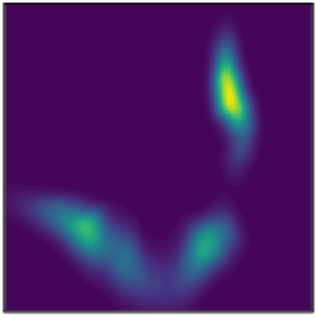
$\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{\psi} \leftarrow$ Initialize parameters
repeat
  $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \sim \hat{p}(\mathbf{x})$ ▷ Sample a minibatch of n examples from the training data.
  for $t = 0, \ldots, T-1$ **do** ▷ Run ALD iterations.
    $V_t = -\sum_{i=1}^{n} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} = \boldsymbol{\Phi} g\left(\boldsymbol{x}^{(i)}; \boldsymbol{\psi}\right); \boldsymbol{\theta}\right)$
    $\boldsymbol{\Phi}' \sim q\left(\boldsymbol{\Phi}' \mid \boldsymbol{\Phi}\right) := \mathcal{N}\left(\boldsymbol{\Phi}'; \boldsymbol{\Phi} - \eta \nabla_{\boldsymbol{\Phi}} V_t, 2\eta \boldsymbol{I}\right)$
    $V_t' = -\sum_{i=1}^{n} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} = \boldsymbol{\Phi}' g\left(\boldsymbol{x}^{(i)}; \boldsymbol{\psi}\right); \boldsymbol{\theta}\right)$
    $\boldsymbol{\Phi} \leftarrow \boldsymbol{\Phi}'$ with probability $\min\left\{1, \frac{\exp\left(-V_t'\right) q\left(\boldsymbol{\Phi} \mid \boldsymbol{\Phi}'\right)}{\exp\left(-V_t\right) q\left(\boldsymbol{\Phi}' \mid \boldsymbol{\Phi}\right)}\right\}$. ▷ MH rejection step.
  end for
  $V_T = -\sum_{i=1}^{n} \log p\left(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)} = \boldsymbol{\Phi} g\left(\boldsymbol{x}^{(i)}; \boldsymbol{\psi}\right); \boldsymbol{\theta}\right)$
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \frac{1}{T} \sum_{t=1}^{T} V_t$ ▷ Update the decoder.
  $\boldsymbol{\psi} \leftarrow \boldsymbol{\psi} - \eta \nabla_{\boldsymbol{\psi}} \frac{1}{T} \sum_{t=1}^{T} V_t$ ▷ Update the encoder.
until convergence of parameters
return $\boldsymbol{\theta}, \boldsymbol{\Phi}, \boldsymbol{\psi}$

Experiments by Author

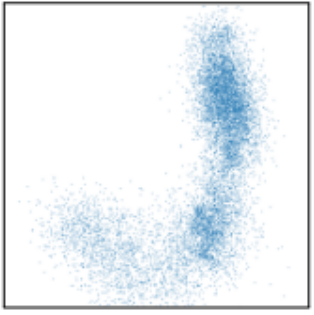
Experiments by Author

- Author run multiple experiments on different dataset, to demonstrate that LAE can properly obtain samples from target distributions and can reconstruct input.
  - Toy Datasets (2D normal distributions with some defined mean and variance)
  - MNIST
  - CIFAR10
  - CelebA
  - SVHN
- The experiments are performed on Computing Nodes of ABCI, each of which has four NVIDIA V100 GPU accelerators, two Intel Xeon Gold 6148, one NVMe SSD, 384GiB memory, two InfiniBand EDR ports (100Gbps each).

Experiments Results



GT



ALD (ours)

Experiments Results

For image generation task author used ELBO score, using different seed values and averging ELBO scores.

	MNIST	SVHN	CIFAR-10	CelebA
VAE	1.189 ± 0.002	4.442 ± 0.003	4.820 ± 0.005	4.671 ± 0.001
VAE-flow	1.183 ± 0.001	4.454 ± 0.016	4.828 ± 0.005	4.667 ± 0.005
Hoffman [2017]	1.189 ± 0.002	4.440 ± 0.007	4.831 ± 0.005	4.662 ± 0.011
LAE (ours)	$\mathbf{1.177} \pm 0.001$	$\mathbf{4.412} \pm 0.002$	$\mathbf{4.773} \pm 0.003$	$\mathbf{4.636} \pm 0.003$

Lower is Better

Experiments by Team

Experiments by Team

Experimental Setup

- For testing Linux Machine CPU: Intel I3, GPU: Geforce 940MX, 12GB RAM.
- For longer training session Kaggle was used, GPU: P400, 16GB RAM.

Dataset

- MNIST
- CIFAR10

Programming Setup

- PyTorch
- PyTorch distributions
- Tensorboard (logging)

Experiments by Team

- Author's code was getting some error so code is written from scratch, the written code also allow to use generic loss functions and models.
- Kaggle notebook was used as it provide 30 hours of GPU time per week and background code run for 12 hours.
- Default configurations and parameters used by authors were used.
- Benchmark using MSE reconstruction loss.

Experiments Results

	MNIST	CIFAR10
VAE	0.0717 ± 0.002	0.0853 ± 0.001
LAE	0.0718 ± 0.0002	0.0823 ± 0.006

Further Investigations

- Analyze the representation learning capability of VAE and LAE.
- Integrate much more powerful encoder and decoder and check generation capability comparing VAE and LAE latent space interpolation results.

References

- https://theaisummer.com/latent-variable-models
- https://abdulfatir.com/blog/2020/Langevin-Monte-Carlo/
- https://www.youtube.com/watch?v=FMuvUZXMzKM
- https://arxiv.org/abs/2209.07036

Thank You!