Importance Weighted Autoencoders

2020. 2. 20

Eunhyuk Shin

Contents

- Review of VAEs
- Importance Weighted Autoencoders (IWAEs)
- What to do & implementation details

Probabilistic Generative Model

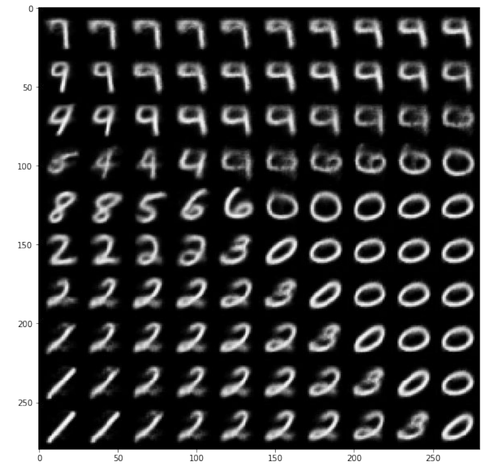
random latent code Z

$Z \sim p(z)$

trainable 'decoder'

$$X|Z \sim p_\theta(x|Z)$$

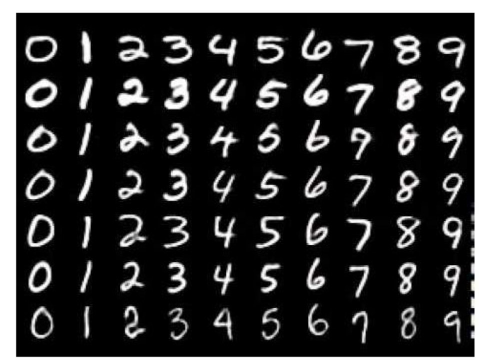
generated sample X



$$p_\theta(x) = \mathbb{E}_{Z\sim p(z)}[p_\theta(x|Z))]$$

find θ s.t. these two distributions are '*close*'

$p_{data}(x)$



Log-Likelihood Objective

- One choice is to **maximize log-likelihood of data**
  - Equivalent to minimizing KLD of data and model distributions

$$\text{maximize}_\theta \mathbb{E}_{X \sim p_{data}(x)} [\log p_\theta(X)]$$

$$\equiv \text{minimize}_\theta D_{\text{KL}}(p_{data}(x) || p_\theta(x))$$

- However, it is intractable to use log-likelihood objective directly
  - Have expectation term inside log
  - Analytic computation: often impossible
  - MC sampling: need too many z samples

$$\log p_\theta(x) = \log \mathbb{E}_{Z \sim p(z)} [p_\theta(x|Z)]$$

Evidence Lower Bound (ELBO)

- Importance sampling: draw better samples & reweight

$$p_\theta(x) = \mathbb{E}_{Z\sim p(z)}[p_\theta(x|Z)] = \mathbb{E}_{Z\sim q(z)}[\frac{p_\theta(x|Z)p(Z)}{q(Z)}]$$

- Exchange log and expectation: creates bias, but **allows single sample estimator**

$$\underbrace{\mathbb{E}_{Z\sim q(z)}[\log \frac{p_\theta(x|Z)p(Z)}{q(Z)}]}_{\mathrm{ELBO}(x,\theta,q)} \leq \log \mathbb{E}_{Z\sim q(z)}[\frac{p_\theta(x|Z)p(Z)}{q(Z)}]$$

Jensen's inequality gap

- Given x and θ, how to get close to LL?

$$\mathrm{ELBO}(x,\theta,q) = \log p_\theta(x) - D_{\mathrm{KL}}(q(z)||p_\theta(z|x))$$

maximize ELBO w.r.t. q

so that the gap is minimized

Variational Autoencoders (VAEs)

- Our objective so far...

$$\text{maximize}_\theta \mathbb{E}_{X \sim p_{data}(x)}[\log p_\theta(X)]$$

$$\approx \text{maximize}_\theta \mathbb{E}_{X \sim p_{data}(x)}[\max_{q \in \mathcal{Q}} \text{ELBO}(X, \theta, q)]$$

cannot do this inner optimization for every single instance of X

- Introduce a learnable recognition model (a.k.a 'encoder')
  - Learn to predict the solution of inner optimization (the ELBO-maximizing q) given X

$$\approx \text{maximize}_{\theta,\phi} \mathbb{E}_{X \sim p_{data}(x)}[\text{ELBO}(X, \theta, q_\phi(z|X))]$$

- Then we end up with VAE!

$$=: \text{maximize}_{\theta,\phi} \mathbb{E}_{X \sim p_{data}(x)}[\mathcal{L}_{\text{VAE}}(X, \theta, \phi)]$$

Reducing the Jensen Gap

- Consider the gap between VAE objective and LL:

$$\mathcal{L}_{\text{VAE}}(x,\theta,\phi) = \mathbb{E}_{Z\sim q_\phi(z|x)}[\log \frac{p_\theta(x|Z)p(Z)}{q_\phi(Z|x)}]$$

$$=: \mathbb{E}_{R|x,\theta,\phi}[\log R]$$

Jensen gap

importance sampling: consistent

$$\log p_\theta(x) = \log \mathbb{E}_{R|x,\theta,\phi}[R]$$

- Jensen gap is due to variance of R.V.
  - R.V. whose distribution concentrated around its expectation will have smaller gap
  - What has same expectation and lower variance? **sample mean**

$$\mathcal{L}_k(x,\theta,\phi) = \mathbb{E}_{\{R_i\}|x,\theta,\phi}[\log \frac{1}{k}\sum_{i=1}^{k} R_i]$$

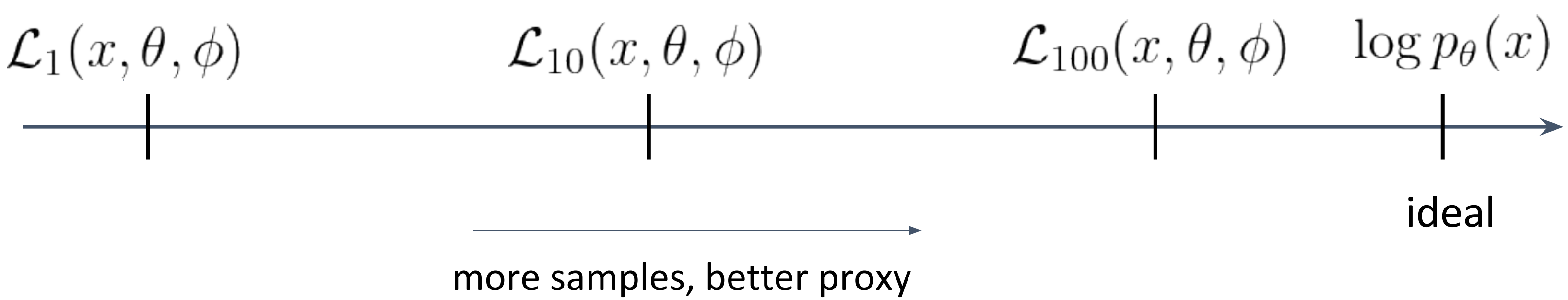
MLILAB Machine Learning & Intelligence

Importance Weighted Autoencoders (IWAEs)

- IWAE objective using k-sample mean

$$\mathcal{L}_k(x,\theta,\phi) = \mathbb{E}_{\{Z_i\}\sim q_\phi(z|x)}[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(x|Z_i)p(Z_i)}{q_\phi(Z_i|x)}]$$

- Correctness
  - Theorem: As k increases, the k-sample IWAE objective becomes a tighter lower bound for the log-likelihood.



Our Setup

- Dataset
  - Binarized MNIST (dim 784)
  - Randomly sampled from Bernoulli where probability = pixel value
- p(z)
  - (Factorized) unit Gaussian
- p(x|z)
  - Input: z (dim: 10)
  - Output: factorized Bernoulli params (mean: dim 784)
  - Simple MLP decoder
- q(z|x)
  - Input: x (dim: 784)
  - Output: factorized Gaussian params (mean: dim 10, variance: dim 10)
  - Simple MLP encoder

What to Implement

- Autoencoder objective **(Provided)**

$$\mathcal{L}_{\mathrm{AE}}(x,\theta,\phi) = \log p_\theta(x|z = \bar{q}_\phi(x))$$

Step 1: Change it to VAE

- VAE objective

$$\mathcal{L}_{\mathrm{VAE}}(x,\theta,\phi) = \mathbb{E}_{Z\sim q_\phi(z|x)}[\log \frac{p_\theta(x|Z)p(Z)}{q_\phi(Z|x)}]$$

Step 2: Use sample mean

- IWAE objective

$$\mathcal{L}_k(x,\theta,\phi) = \mathbb{E}_{\{Z_i\}\sim q_\phi(z|x)}[\log \frac{1}{k}\sum_{i=1}^{k} \frac{p_\theta(x|Z_i)p(Z_i)}{q_\phi(Z_i|x)}]$$

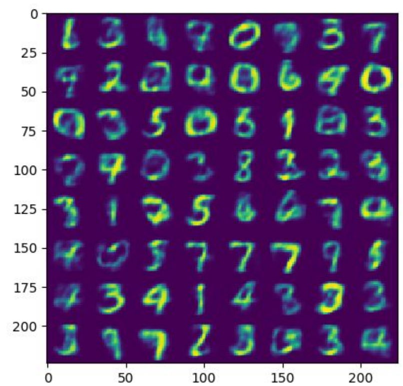
What to Experiment

- **Step 3: Verify that using sample mean helps maximizing the log-likelihood**

$\mathcal{L}_1(x, \theta, \phi)$	$\mathcal{L}_{10}(x, \theta, \phi)$	$\mathcal{L}_{100}(x, \theta, \phi)$	$\log p_\theta(x)$
train VAE	train IWAE	eval VAE, IWAE	true objective

- Example results:

	VAE	IWAE
train k	1	10
train loss	165	140
estimated test NLL	145	129



Sample visualizer is provided (for sanity-check)

- Base code link: https://github.com/silvershine157/IWAE-base

MLILAB Machine Learning & Intelligence

IWAE Algorithmic View

Algorithm 1: IWAE

while *not converged* **do**

- $x \sim p_{data}$;
- $\mu, \sigma := f_\phi(x)$;
- **for** i=1...k **do**
  - $\epsilon_i \sim \text{Gaussian}(0, I)$;
  - $z_i := \mu + \sigma \odot \epsilon_i$;
  - $\hat{x}_i := g_\theta(z_i)$;
  - $\log p_\theta(x|z_i) := \text{BernoulliLL}(x; \hat{x}_i)$;
  - $\log p_\theta(z_i) := \text{GaussianLL}(z_i; 0, I)$;
  - $\log q_\phi(z_i|x) := \text{GaussianLL}(z_i; \mu, \text{diag}(\sigma))$;
  - $\frac{p_\theta(x,z_i)}{q_\phi(z_i|x)} := \exp(\log p_\theta(x|z_i) + \log p_\theta(z_i) - \log q_\phi(z_i|x))$;
- $\mathcal{L}_{\text{IWAE}}(x, \phi, \theta) := \log(\frac{1}{k} \sum_{i=1}^{k} \frac{p_\theta(x,z_i)}{q_\phi(z_i|x)})$;
- $[\phi, \theta] := [\phi, \theta] + \alpha \nabla_{[\phi,\theta]} \mathcal{L}_{\text{IWAE}}(x, \phi, \theta)$

Caution: Consider Backpropagation

- Prevent underflow
  - Ex) forward pass of: log(mean(exp[**-300**, **-200**]) + 1E-7)
  - Don't
    - => log(mean[0., 0.] + 1E-7) <-- too small, simply becomes 0
    - => log 1E-7 <-- gradient path of input is lost
  - Do
    - => -200+log(mean(exp[-100, 0])+1E-7) <-- exploit log-exp relation
    - => -200+log(mean[0., 1]+1E-7)
    - => **-200**+log(0.5) <-- has gradient path

- Reparametrization trick
  - Ex) forward pass of: z ~ Gaussian(**mu, sigma**)
  - Don't
    - z = sample_mvn(mu, sigma) <-- cannot differentiate z w.r.t. mu & sigma
  - Do
    - epsilon = sample_mvn(0, I)
    - z = **mu** + **sigma** * epsilon <-- can differentiate z w.r.t. mu & sigma

References

- “Importance Weighted Autoencoders”, Y. Bruda et al., arXiv preprint, 2015
- “Importance Weighted Variational Inference”, J. Domke et al., NeurIPS, 2018
- “Auto-Encoding Variational Bayes”, D. Kingma et al., arXiv preprint, 2013
- “Reinterpreting Importance-Weighted Autoencoders”, C. Cremer et al., ICLR Workshop, 2017
- “Tighter Variational Bounds are Not Necessarily Better”, S. Arik et al., ICML, 2018