The (Local) Reparameterization Trick

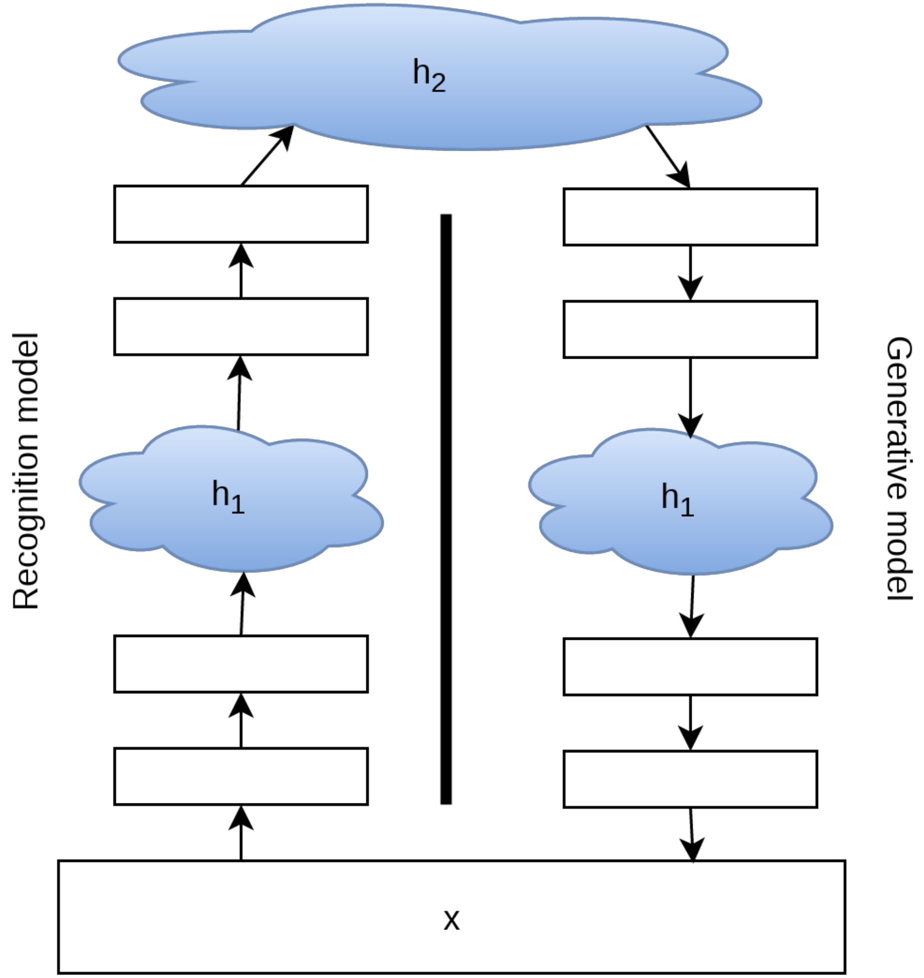
D. Kingma, T. Salimans, M. Welling https://arxiv.org/abs/1506.02557, https://arxiv.org/abs/1312.6114

Slides by Ray Xiao tianrui.xiao@mail.utoronto.ca

- Helmholtz machine
- The general reparameterization trick
- Look at the formulation of variational inference and SGVB
- Discuss the drawback of the SGVB estimator
- Introduce an alternative estimator (local reparameterization trick)

Before VAEs

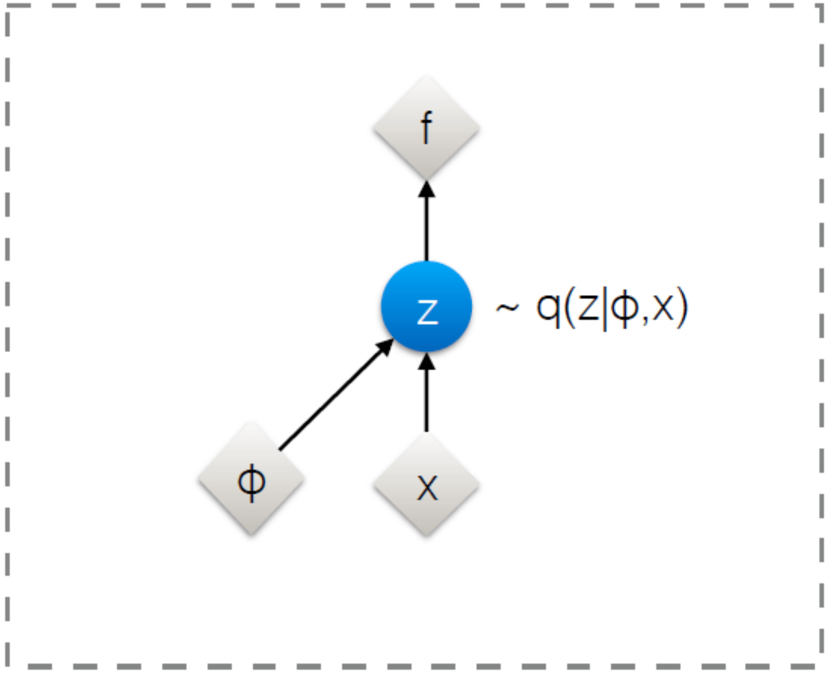
- Helmholtz Machine
  - Multiple stochastic latent layers, each specified by $p(h_{k-1}|h_k)$ in the generative model; reverse in the recognition model
  - Trained using the wake-sleep algorithm
  - Minimize the cost function $C(d) = \sum_\alpha Q(\alpha|x)C(\alpha,x) - (-\sum_\alpha Q(\alpha|x)logQ(\alpha|x))$
  - $C(\alpha,x)$ is the cost of describing the input vector x using the “total representations” α
  - $Q(\alpha|x)$ is the conditional distribution of the recognition weights over total representations
- The cost function is analogous to the Helmholtz free energy of a physical system



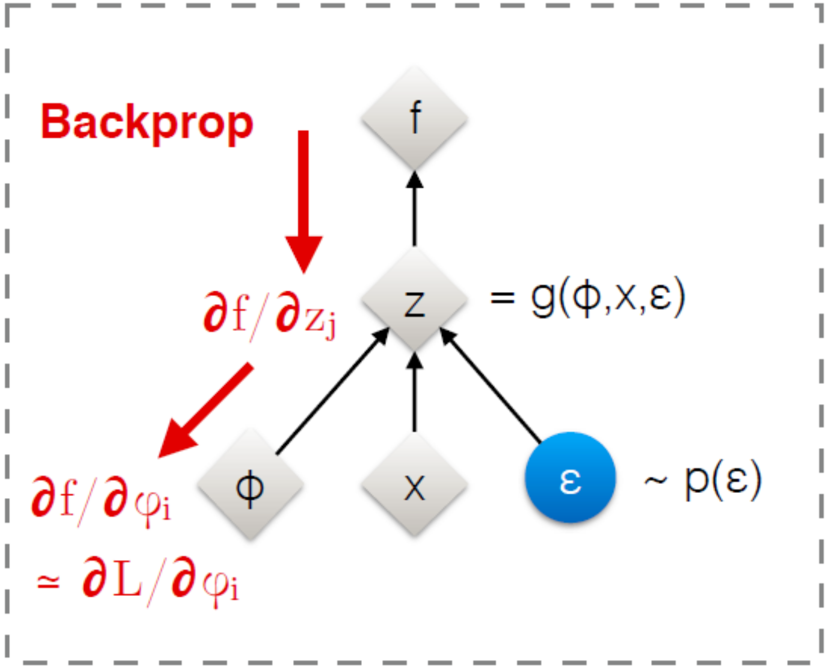
- For a dataset of N observations $\mathrm{D} = \{x^{(n)}\}_{n=1}^{n=N}$
- There is a simple latent space z:
  - $z \sim p_\theta(z)$
  - $x|z \sim p_\theta(x|z)$
  - The exact posterior distribution from Bayes' Rule is intractable
- Goal
  - Learn approx. parameters
  - Infer latent variables based on new observations
  - Parameterize model $q_\phi(z|x)$ to approximate $p(z|x)$
  - Measure by maximizing the variational lower bound: $\log p(x) \geq L(\theta, \phi) = -D_{KL}(q_\phi(z|x)||p_\theta(z|x)) + E_{q_\phi}[\log p_\theta(x|z)] = \mathrm{E}_{\mathrm{q}_\phi}[-\log \mathrm{q}_\phi(\mathrm{z}|\mathrm{x}) + \log \mathrm{p}_\theta(\mathrm{x}, \mathrm{z})]$
  - Wish to optimize the variational lower bound w.r.t. the two parameters
  - The MC gradient estimator for ϕ is $\nabla_\phi E_{q_\phi}[f(z)]$ which exhibits high variance

Reparameterization trick

Original form



Reparameterised form



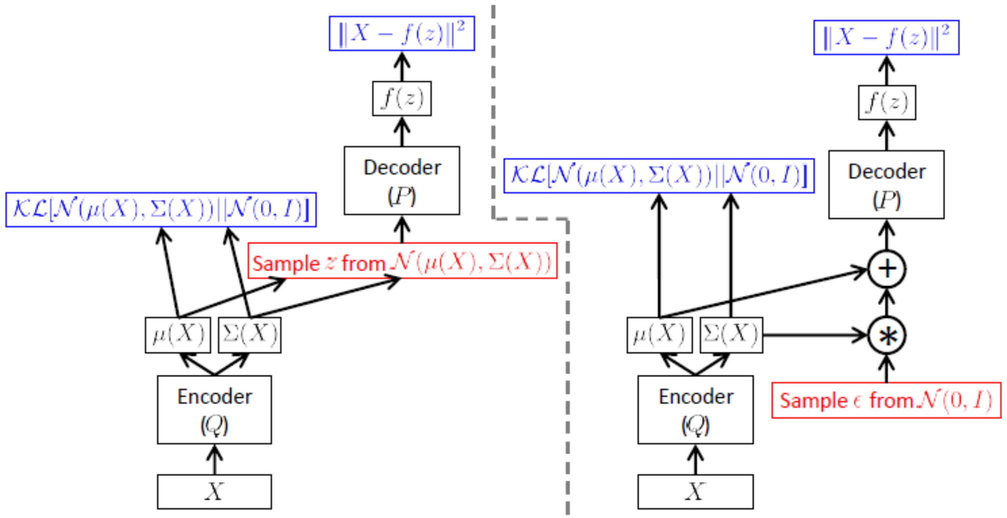
: Deterministic node

: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Kingma & Welling, NIPS workshop 2015

Auto-encoding VB algorithm



Parameterize as $p_\theta(z), q_\phi(z|x)$

Repeat

- Sample x (datapoint/minibatch)
- Sample $\epsilon \sim p(\epsilon)$
- Calculate gradients $g_\theta, g_\phi = \nabla_{\theta,\phi} L(\theta, \phi; x, g(\epsilon, \phi))$ of the minibatch estimator
- Update parameters

Until convergence

By parameterizing the latent variable z as a deterministic function of a random variable drawn from a prior, we can backprop the gradient to the parameters ϕ.

Stochastic Gradient Variational Bayes

- Trick is to reparameterize $\mathrm{z} = f_{\phi}(\epsilon, x)$ where f is differentiable and $\epsilon \sim p(\epsilon)$ is a random noise variable
- $L\left(\theta, \phi; x^{i}\right) = -D_{KL}(q_{\phi}\left(z \middle| x^{i}\right) || p_{\theta}(z)) + \frac{1}{L}\sum_{l=1}^{L}(\log p_{\theta}(x^{i}|z^{i,l}))$
- For a minibatch of size M, we have a Monte Carlo estimator for the full dataset: $L_{D}(\phi) \approx L_{D}^{SGVB}(\phi) = \frac{N}{M}\sum_{i=1}^{M}(-D_{KL}(q_{\phi}\left(z \middle| x^{i}\right) || p_{\theta}(z)) + \log p_{\theta}(x^{i}|z^{i} = f_{\phi}(\epsilon, x))) = \frac{N}{M}\sum_{i=1}^{M} L_{i}$
- This estimator is unbiased and differentiable w.r.t. ϕ, so the gradient is also unbiased. Now we have a Monte Carlo estimate, and assuming we can calculate the KL divergence similarly or analytically

Variance of the SGVB estimator

- $Var[L_D^{SGVB}(\phi)] = \frac{N^2}{M^2}(\sum_{i=1}^{M} Var[L_i] + \sum_{i=1}^{M}\sum_{j=i+1}^{M} Cov[L_i, L_j]) = N^2(\frac{1}{M}Var[L_i] + \frac{M-1}{M}Cov[L_i, L_j])$
- Note from above that the variance can be dominated by the covariance terms
- If the variance is large, the stochastic gradient descent won't converge to local optimum
- Thus we want an estimator with zero covariance

Local Reparameterization Trick

- Propose an estimator with zero covariance
- We sample from the intermediate variables $f(\epsilon)$ instead of sampling from the noise distribution directly
- Example: fully connected NN with 1 hidden layer, 1000 units
- Input matrix A (M x 1000), then neuron activation B = AW
- Suppose posterior approx. of weights to be a fully factorized Gaussian: $q_\phi\left(w_{i,j}\right) = N(\mu_{i,j}, \sigma_{i,j}^2)$
- This way we can ensure zero covariance by sampling a separate W for each example, but it's inefficient

- Note the weights influence B, which are of lower dimension, so we can sample random activations B instead
- For a factorized Gaussian posterior on W, we also have a factorized Gaussian for activations:
  - $q_\phi\left(b_{m,j}|A\right) = N\left(\gamma_{m,j}, \delta_{m,j}\right)$
  - $\gamma_{m,j} = \sum_{i=1}^{1000} a_{m,i}\mu_{i,j} \quad \delta_{m,j} = \sum_{i=1}^{1000} a_{m,i}^2 \delta_{i,j}^2$
  - So we can sample the activations from the implied Gaussian distribution directly as $b_{m,j} = \gamma_{m,j} + \sqrt{\delta_{m,j}}\zeta_{m,j} \quad \zeta_{m,j} \sim N(0,1)$

Importance Weighted Autoencoders

Yuri Burda, Roger Grosse, & Ruslan Salakhutdinov (2016)

"Get a tighter lower bound by sampling k times from the approximate posterior"

Geoffrey Roeder (roeder@cs.toronto.edu)

October 14, 2016

Outline

Motivations

- Limitations of Vanilla VAEs
- VAE Training Refresher
- Why Vanilla VAEs are Suboptimal

IWAE Solution

- Inspiration: Importance Sampling
- Importance Weighted Autoencoder
- Benefits

Motivations

Outline

Motivations

- Limitations of Vanilla VAEs
- VAE Training Refresher
- Why Vanilla VAEs are Suboptimal

IWAE Solution

- Inspiration: Importance Sampling
- Importance Weighted Autoencoder
- Benefits

Importance Weighted Autoencoder (IWAE) Idea

- We want to learn flexible approximate posteriors to have a rich probability model

Importance Weighted Autoencoder (IWAE) Idea

- We want to learn flexible approximate posteriors to have a rich probability model
- Vanilla VAE (Auto-Encoding Variational Bayes) uses a single sample of the latent variable for each datapoint when estimating the gradient

Importance Weighted Autoencoder (IWAE) Idea

- We want to learn flexible approximate posteriors to have a rich probability model
- Vanilla VAE (Auto-Encoding Variational Bayes) uses a single sample of the latent variable for each datapoint when estimating the gradient
- IWAE ("eye-way") learns a more flexible posterior by averaging multiple samples of the posterior scaled according to importance weights

Outline

Motivations

- Limitations of Vanilla VAEs
- VAE Training Refresher
- Why Vanilla VAEs are Suboptimal

IWAE Solution

- Inspiration: Importance Sampling
- Importance Weighted Autoencoder
- Benefits

Vanilla VAEs Refresher: Stochastic Backpropagation

- Train VAEs by optimizing lower bound $\mathcal{L}$ on log-posterior, equivalent to minimizing $KL(q_\phi(z|x)||p(z|x))$

$$\log p(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}\Big] = \mathcal{L}(\theta,\phi;x) \qquad (1)$$

Vanilla VAEs Refresher: Stochastic Backpropagation

- Train VAEs by optimizing lower bound $\mathcal{L}$ on log-posterior, equivalent to minimizing $KL(q_\phi(z|x)||p(z|x))$

$$\log p(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}\Big] = \mathcal{L}(\theta,\phi;x) \qquad (1)$$

- Reparameterize $z = g_\phi(\epsilon, x)$ and obtain gradient by auto-diff

$$\nabla_{\theta,\phi}\ \mathcal{L}(\theta,\phi;x) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\Big[\nabla_{\theta,\phi} \log \frac{p_\theta(g_\phi(\epsilon,x),x)}{q_\phi(g_\phi(\epsilon,x)|x)}\Big] \qquad (2)$$

Vanilla VAEs Refresher: Stochastic Backpropagation

- Train VAEs by optimizing lower bound $\mathcal{L}$ on log-posterior, equivalent to minimizing $KL(q_\phi(z|x)||p(z|x))$

$$\log p(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}\Big[\log \frac{p_\theta(z,x)}{q_\phi(z|x)}\Big] = \mathcal{L}(\theta,\phi;x) \qquad (1)$$

- Reparameterize $z = g_\phi(\epsilon, x)$ and obtain gradient by auto-diff

$$\nabla_{\theta,\phi}\ \mathcal{L}(\theta,\phi;x) = \mathbb{E}_{\epsilon\sim\mathcal{N}(0,1)}\Big[\nabla_{\theta,\phi}\log \frac{p_\theta(g_\phi(\epsilon,x),x)}{q_\phi(g_\phi(\epsilon,x)|x)}\Big] \qquad (2)$$

- Estimate by MC: generate K samples of ϵ and evaluate

$$\frac{1}{K}\sum_{i=1}^{K}\nabla_{\theta,\phi}\log \frac{p_\theta(g_\phi(\epsilon^{(i)},x),x)}{q_\phi(g_\phi(\epsilon^{(i)},x)|x)} \qquad (3)$$

Outline

Motivations

- Limitations of Vanilla VAEs
- VAE Training Refresher
- Why Vanilla VAEs are Suboptimal

IWAE Solution

- Inspiration: Importance Sampling
- Importance Weighted Autoencoder
- Benefits

The Problem

- Optimizer maximizes $\nabla_{\theta,\phi}\ \mathcal{L}(\theta, \phi; x)$ w.r.t. ϕ, θ where each step estimates gradient based on a single sample $z \sim q_\phi(z|x)$

The Problem

- Optimizer maximizes $\nabla_{\theta,\phi}\ \mathcal{L}(\theta, \phi; x)$ w.r.t. ϕ, θ where each step estimates gradient based on a single sample $z \sim q_\phi(z|x)$
- This harshly penalizes any stochastic samples that don't explain the data well

The Problem

- Optimizer maximizes $\nabla_{\theta,\phi}\ \mathcal{L}(\theta, \phi; x)$ w.r.t. ϕ, θ where each step estimates gradient based on a single sample $z \sim q_\phi(z|x)$
- This harshly penalizes any stochastic samples that don't explain the data well
- Encourages learned approximate posterior to be approximately factorial (random variables are statistically independent so that $p(y_1, ..., y_L) = \prod_{j=1}^{L} p(y_j)$), with parameters that are learnable through nonlinear regression (as the final layer of a neural network)

IWAE Solution

Outline

Motivations

Limitations of Vanilla VAEs

VAE Training Refresher

Why Vanilla VAEs are Suboptimal

IWAE Solution

Inspiration: Importance Sampling

Importance Weighted Autoencoder

Benefits

Importance Sampling in Monte Carlo estimation

- Consider an arbitrary $\mathbb{E}_{p(z)}[f(z)]$ where $p(z)$ is a complicated distribution, $f(z)$ is **any** function, and $q(z)$ **any** easy-to-sample and easy-to-evaluate distribution

Importance Sampling in Monte Carlo estimation

- Consider an arbitrary $\mathbb{E}_{p(z)}[f(z)]$ where $p(z)$ is a complicated distribution, $f(z)$ is **any** function, and $q(z)$ **any** easy-to-sample and easy-to-evaluate distribution
- Sampling from $p(z)$ is hard, so we'd rather sample from $q(z)$. This introduces a bias, because the expectation is w.r.t. $p(z)$.

Importance Sampling in Monte Carlo estimation

- Consider an arbitrary $\mathbb{E}_{p(z)}[f(z)]$ where $p(z)$ is a complicated distribution, $f(z)$ is **any** function, and $q(z)$ **any** easy-to-sample and easy-to-evaluate distribution
- Sampling from $p(z)$ is hard, so we'd rather sample from $q(z)$. This introduces a bias, because the expectation is w.r.t. $p(z)$.
- By using importance weights, we correct that bias. The derivation is straightforward:

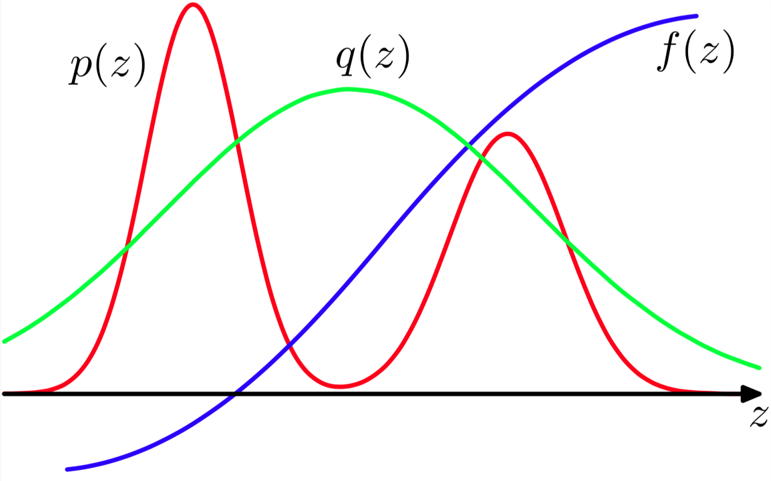
$$\mathbb{E}_p(z)[f(z)] = \int p(z)f(z)dz = \int q(z)\frac{p(z)}{q(z)}f(z)dz \tag{4}$$

$$= \int q(z)w(z)f(z)dz \tag{5}$$

$$= \mathbb{E}_q(z)[w(z)f(z)] \tag{6}$$

$$\approx \frac{1}{K}\sum_{i=1}^{K} w(z^{(i)})f(z^{(i)}) \tag{7}$$

Importance Sampling in Monte Carlo estimation



Bishop PRML, Fig. 11.8

Outline

Motivations

Limitations of Vanilla VAEs

VAE Training Refresher

Why Vanilla VAEs are Suboptimal

IWAE Solution

Inspiration: Importance Sampling

Importance Weighted Autoencoder

Benefits

IWAE The What

- Take k independent samples to evaluate the loss:

$$\mathcal{L}_k = \mathbb{E}_{z^1,\ldots,z^k \sim q(z|x)}\left[log \frac{1}{K}\sum_{i=1}^{K} \frac{p_\theta(x, z^i)}{q_\phi(z^i|x)}\right] \tag{8}$$

IWAE The What

- Take k independent samples to evaluate the loss:

$$\mathcal{L}_k = \mathbb{E}_{z^1,\ldots,z^k \sim q(z|x)}\left[log \frac{1}{K} \sum_{i=1}^{K} \frac{p_\theta(x, z^i)}{q_\phi(z^i|x)} \right] \tag{8}$$

- With respect to reparameterization, take k samples of $\epsilon^i \sim N(0, \mathbb{I})$ and proceed as before on each e^i an independent sample

IWAE The What

- Take k independent samples to evaluate the loss:

$$\mathcal{L}_k = \mathbb{E}_{z^1,\ldots,z^k \sim q(z|x)}\Big[log \frac{1}{K}\sum_{i=1}^{K} \frac{p_\theta(x,z^i)}{q_\phi(z^i|x)}\Big] \tag{8}$$

- With respect to reparameterization, take k samples of $\epsilon^i \sim N(0,\mathbb{I})$ and proceed as before on each e^i an independent sample
- For simplicity, call $w^i = \frac{p_\theta(x,z^i)}{q_\phi(z^i|x)}$. Then, it's still true that

$$\mathcal{L}_k = \mathbb{E}_{q_\phi}\Big[\log \frac{1}{k}\sum_{i=1}^{k} w^i\Big] \le \log \mathbb{E}_{q_\phi}\Big[\frac{1}{k}\sum_{i=1}^{k} w^i\Big] = \log p(x) \tag{9}$$

(Hint: to derive this, work backwards from the definitions of w^i expectation, applying Jensen's inequality)

IWAE: The Weights

- How does this relate to important sampling? Let's look at the gradient of $\mathcal{L}_k$, defining $\psi = (\theta, \phi)$ for convenience:

$$\nabla_\psi \mathcal{L}_k = \nabla_\psi \mathbb{E}_{z^1,\ldots,z^k}\left[\frac{1}{k}\sum_{i=1}^{k} w^i\right] \tag{10}$$

IWAE: The Weights

- How does this relate to important sampling? Let's look at the gradient of $\mathcal{L}_k$, defining $\psi = (\theta, \phi)$ for convenience:

$$\nabla_\psi \mathcal{L}_k = \nabla_\psi \mathbb{E}_{z^1,\dots,z^k}\left[\frac{1}{k}\sum_{i=1}^{k} w^i\right] \tag{10}$$

- Reparameterizing $z^i = g_\phi(\epsilon^i)$ let us move the gradient inside the expectation. Rewriting w^i as $w(\epsilon^i)$ gives:

IWAE: The Weights

- How does this relate to important sampling? Let's look at the gradient of $\mathcal{L}_k$, defining $\psi = (\theta, \phi)$ for convenience:

$$\nabla_\psi \mathcal{L}_k = \nabla_\psi \mathbb{E}_{z^1,\ldots,z^k}\left[\frac{1}{k}\sum_{i=1}^{k} w^i\right] \tag{10}$$

- Reparameterizing $z^i = g_\phi(\epsilon^i)$ let us move the gradient inside the expectation. Rewriting w^i as $w(\epsilon^i)$ gives:

$$\mathbb{E}_{\epsilon^1,\ldots,\epsilon^k}\left[\nabla_\psi \log \frac{1}{k}\sum_{i=1}^{k} w(\epsilon^i)\right] \tag{11}$$

$$= \mathbb{E}_{\epsilon^1,\ldots,\epsilon^k}\left[\frac{1}{k}\sum_{i=1}^{k} \widetilde{w}(\epsilon^i)\nabla_\psi \log w(\epsilon^i)\right], \tag{12}$$

where $\widetilde{w}(\epsilon^i) = \frac{w(\epsilon^i)}{\frac{1}{k}\sum_{j=1}^{k} w(\epsilon^j)}$, weighting the samples.

IWAE: The Weights

How does this relate to important sampling? Let's look at the gradient of $\mathcal{L}_k$, defining $\psi = (\theta, \phi)$ for convenience:

$$\nabla_\psi \mathcal{L}_k = \nabla_\psi \mathbb{E}_{z^1,\dots,z^k}\left[\frac{1}{k}\sum_{i=1}^{k} w^i\right] \tag{13}$$

Reparameterizing $z^i = g_\phi(\epsilon^i)$ let us move the gradient inside the expectation. Rewriting w^i as $w(\epsilon^i)$ gives:

$$\mathbb{E}_{\epsilon^1,\dots,\epsilon^k}\left[\nabla_\psi \log \frac{1}{k}\sum_{i=1}^{k} w(\epsilon^i)\right] \tag{14}$$

$$= \mathbb{E}_{\epsilon^1,\dots,\epsilon^k}\left[\frac{1}{k}\sum_{i=1}^{k} \widetilde{w}(\epsilon^i)\nabla_\psi \log w(\epsilon^i)\right], \tag{15}$$

where $\widetilde{w}(\epsilon^i) = \frac{w(\epsilon^i)}{\frac{1}{k}\sum_{j=1}^{k} w(\epsilon^j)}$, weighting the samples

IWAE: The Weights

How does this relate to important sampling? Let's look at the gradient of $\mathcal{L}_k$, defining $\psi = (\theta, \phi)$ for convenience:

$$\nabla_\psi \mathcal{L}_k = \nabla_\psi \mathbb{E}_{z^1,\ldots,z^k}\left[\frac{1}{k}\sum_{i=1}^{k} w^i\right] \tag{16}$$

Reparameterizing $z^i = g_\phi(\epsilon^i)$ let us move the gradient inside the expectation. Rewriting w^i as $w(\epsilon^i)$ gives:

$$\mathbb{E}_{\epsilon^1,\ldots,\epsilon^k}\left[\nabla_\psi \log \frac{1}{k}\sum_{i=1}^{k} w(\epsilon^i)\right] \tag{17}$$

$$= \mathbb{E}_{\epsilon^1,\ldots,\epsilon^k}\left[\frac{1}{k}\sum_{i=1}^{k} \widetilde{w}(\epsilon^i)\nabla_\psi \log \frac{p_\theta(x, z^i)}{q_\phi(z^i|x)}\right], \tag{18}$$

where $\widetilde{w}(\epsilon^i) = \frac{w(\epsilon^i)}{\frac{1}{k}\sum_{j=1}^{k} w(\epsilon^j)}$, weighting the samples

IWAE: Weighted ELBO

- Recalling that $w(\epsilon^i) = \frac{p_\theta(x, g_\phi(\epsilon^i))}{q_\phi(g_\phi(\epsilon^i)|x)}$, we can analyze how $\widetilde{w}(\epsilon^i)\nabla_\psi \log w(\epsilon^i)$ behaves as an optimization objective:

$$\widetilde{w}(\epsilon^i)\nabla_\psi(\log p_\theta(x, g_\phi(\epsilon^i)) - \log q_\phi(g_\phi(\epsilon^i)|x)) \tag{19}$$

IWAE: Weighted ELBO

- Recalling that $w(\epsilon^i) = \frac{p_\theta(x, g_\phi(\epsilon^i))}{q_\phi(g_\phi(\epsilon^i)|x)}$, we can analyze how $\widetilde{w}(\epsilon^i)\nabla_\psi \log w(\epsilon^i)$ behaves as an optimization objective:

$$\widetilde{w}(\epsilon^i)\nabla_\psi(\log p_\theta(x, g_\phi(\epsilon^i)) - \log q_\phi(g_\phi(\epsilon^i)|x)) \tag{19}$$

- The log joint term encourages the recognition network q_ϕ to adjust the latent representations so the generator network makes better predictions

IWAE: Weighted ELBO

- Recalling that $w(\epsilon^i) = \frac{p_\theta(x, g_\phi(\epsilon^i))}{q_\phi(g_\phi(\epsilon^i)|x)}$, we can analyze how $\widetilde{w}(\epsilon^i)\nabla_\psi \log w(\epsilon^i)$ behaves as an optimization objective:

$$\widetilde{w}(\epsilon^i)\nabla_\psi(\log p_\theta(x, g_\phi(\epsilon^i)) - \log q_\phi(g_\phi(\epsilon^i)|x)) \qquad (19)$$

- The log joint term encourages the recognition network q_ϕ to adjust the latent representations so the generator network makes better predictions
- The log approximate posterior term encourages the network to have a spread out distribution over the latent variables.

IWAE: Weighted ELBO

- Recalling that $w(\epsilon^i) = \frac{p_\theta(x, g_\phi(\epsilon^i))}{q_\phi(g_\phi(\epsilon^i)|x)}$, we can analyze how $\widetilde{w}(\epsilon^i)\nabla_\psi \log w(\epsilon^i)$ behaves as an optimization objective:

$$\widetilde{w}(\epsilon^i)\nabla_\psi(\log p_\theta(x, g_\phi(\epsilon^i)) - \log q_\phi(g_\phi(\epsilon^i)|x)) \tag{19}$$

- The log joint term encourages the recognition network q_ϕ to adjust the latent representations so the generator network makes better predictions
- The log approximate posterior term encourages the network to have a spread out distribution over the latent variables.
- Both terms are scaled by an importance weight for that sample of the latent variables, which ensures that non-representative samples incur smaller penalties

Outline

Motivations

- Limitations of Vanilla VAEs
- VAE Training Refresher
- Why Vanilla VAEs are Suboptimal

IWAE Solution

- Inspiration: Importance Sampling
- Importance Weighted Autoencoder
- Benefits

IWAE: Theoretical Result

Theorem

For all k, the lower bounds satisfy

$$\log p(x) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k \tag{20}$$

Morever, if $p(z, x)/q(z|x)$ *is bounded, then* $\mathcal{L}_k$ *approaches* $\log p(x)$ *as k goes to infinity.*

IWAE: Theoretical Result

Theorem

For all k, the lower bounds satisfy

$$\log p(x) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k \tag{20}$$

Morever, if $p(z, x)/q(z|x)$ is bounded, then $\mathcal{L}_k$ approaches $\log p(x)$ as k goes to infinity.

- Under reasonable conditions and given sufficient computing time, we can make the lower bound on the marginal likelihood as tight as we want

IWAE: Theoretical Result

Theorem

For all k, the lower bounds satisfy

$$\log p(x) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k \tag{20}$$

Morever, if $p(z,x)/q(z|x)$ *is bounded, then* $\mathcal{L}_k$ *approaches* $\log p(x)$ *as* k *goes to infinity.*

- Under reasonable conditions and given sufficient computing time, we can make the lower bound on the marginal likelihood as tight as we want
- This comes at the cost of a linear increase in complexity in the training of the recognition network (we need an extra k forward and backwards passes of the recognition network)

IWAE: Theoretical Result

Theorem

For all k, the lower bounds satisfy

$$\log p(x) \geq \mathcal{L}_{k+1} \geq \mathcal{L}_k \tag{20}$$

Morever, if $p(z,x)/q(z|x)$ is bounded, then $\mathcal{L}_k$ approaches $\log p(x)$ as k goes to infinity.

- Under reasonable conditions and given sufficient computing time, we can make the lower bound on the marginal likelihood as tight as we want
- This comes at the cost of a linear increase in complexity in the training of the recognition network (we need an extra k forward and backwards passes of the recognition network)
- Empirical results on MNIST and OMNIGLOT confirm the theoretical result here: compared to a VAE with equivalent

IWAE: Summary

- Method is agnostic as to choice of recognition / generative distributions

IWAE: Summary

- Method is agnostic as to choice of recognition / generative distributions
- Con: not cheap computationally. See paper for an extra trick to reduce the linear complexity by a constant factor.

IWAE: Summary

- Method is agnostic as to choice of recognition / generative distributions
- Con: not cheap computationally. See paper for an extra trick to reduce the linear complexity by a constant factor.
- Useful to quantify progress towards a theoretical lower bound. I.e., run for $k = 5000$ times, evaluate how much better your new method works compared to previous methods in terms of their progress to the baseline

Thanks for listening!

If you want a copy of these slides, email me at
roeder@cs.toronto.edu

Structured encoding/decoding

We'll be talking about different ways of:

- adding structure to the encoder and decoder,
- make $q(z|x)$ or $p(x|z)$ more complex
- "interpret" the latent space

Variational Inference with Normalizing Flows

CSC 2541 Seminar: Structured Encoders/Decoders

Slides by Lisa Zhang

October 14, 2016

The elephant in the room

Figure 1:
https://commons.wikimedia.org/w/index.php?curid=22613018

The elephant in the room

One problem with vanilla variational inference is that we can never recover the true posterior distribution.

$$q(z|x) \neq p(z|x)$$

Not even in the asymptotic regime.

We know that a more faithful $q(z|x)$ gives better result, if we can sample from it.

Normalizing Flows

Transform a (simple) probability density through a sequence of invertible mappings.

Hopefully we will

- still be able to sample from this distribution
- better approximate the complex $p(z|x)$ distributions
- recover the true distribution as length of sequence $\rightarrow \infty$

Normalizing Flows

- $f : \mathbb{R}^d \to \mathbb{R}^d$ smooth, invertible, with $f^{-1} = g$.
- z random variable with density $q(z)$
- $z' = f(z)$ random variable

$$q(z') = q(z) \left| \det \frac{\partial f^{-1}}{\partial z'} \right| = q(z) \left| \det \frac{\partial f}{\partial z'} \right|^{-1}$$

Successively applying for a chain of K transformation

$$z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0)$$

$$\ln q_K(z_K) = \ln q_0(z_0) - \sum_{k=1}^{K} \ln \left| \det \frac{\partial f_k}{\partial z_{k-1}} \right|$$

Law of the Unconscious Statistician

The expectation w.r.t. density q_K can be computed without explicitly writing down q_K; such an expectation can be rewritten as:

$$\mathbb{E}_{q_K}[h(z)]] = \mathbb{E}_{q_0}[h(f_K \circ \cdots \circ f_2 \circ f_1(z_0))]$$

So we can specify a complex $q(z|x)$ starting with a simple distribution (e.g. Gaussian), then apply normalizing flows to get a complex/multimodel distribution.

What it looks like

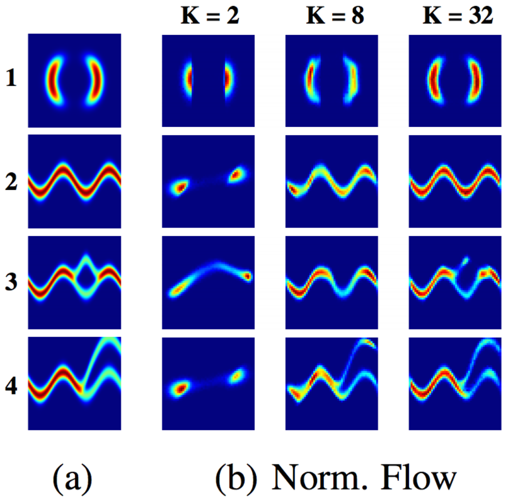


Figure 2: https://arxiv.org/abs/1505.05770

Architecture

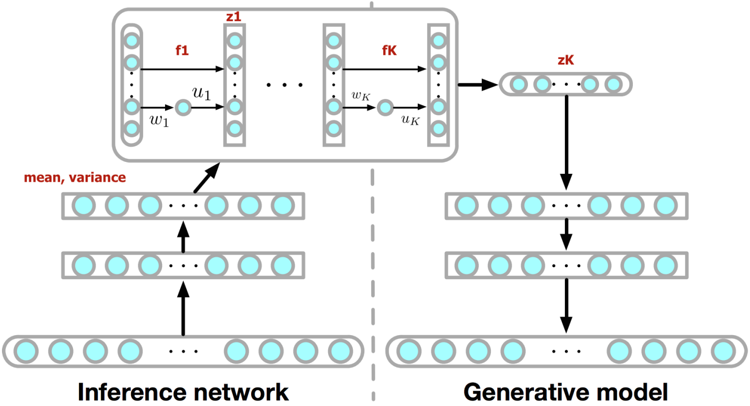


Figure 3: https://arxiv.org/abs/1505.05770

Invertible linear-time transformations

For scalable inference, we need f_l that allows for low-cost computation of the determinant of the Jacobian.

Example

- Planar Flow: contractions/expansions around a hyperplane
- Radial Flow: contractions/expansions around a point

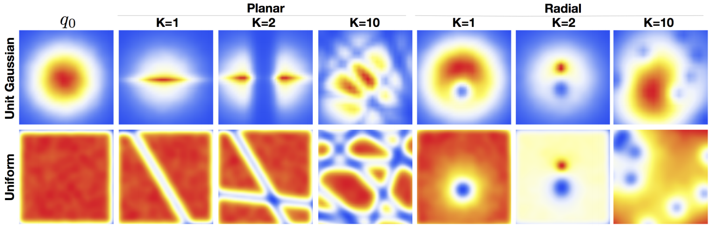


Figure 4: https://arxiv.org/abs/1505.05770

Planar Flow

Transformations of form

$$f(z) = z + uh(w^T z + b)$$

Can show that

$$\left| \det \frac{\partial f}{\partial z} \right| = \left| 1 + u^T \psi(z) \right|$$

where $\psi(z) = h'(w^T z + b)w$.

Modifies the density by a series of contractions and expansions in the direction perpendicular to the hyperplane $w^T z + b = 0$.

Radial Flow

Transformations of form

$$f(z) = z + \beta h(\alpha, r)(z - z_0)$$

Can also compute $\left|\det \frac{\partial f}{\partial z}\right|$ in $O(D)$ time.

Modifies the density by a series radial contractions and expansions around the reference point z_0.

Results: MNIST

Table 2. Comparison of negative log-probabilities on the test set for the binarised MNIST data.

Model	$-\ln p(\mathbf{x})$
DLGM diagonal covariance	≤ 89.9
DLGM+NF (k = 10)	≤ 87.5
DLGM+NF (k = 20)	≤ 86.5
DLGM+NF (k = 40)	≤ 85.7
DLGM+NF (k = 80)	≤ 85.1
DLGM+NICE (k = 10)	≤ 88.6
DLGM+NICE (k = 20)	≤ 87.9
DLGM+NICE (k = 40)	≤ 87.3
DLGM+NICE (k = 80)	≤ 87.2
Results below from (Salimans et al., 2015)	
DLGM + HVI (1 leapfrog step)	88.08
DLGM + HVI (4 leapfrog steps)	86.40
DLGM + HVI (8 leapfrog steps)	85.51
Results below from (Gregor et al., 2014)	
DARN $n_h = 500$	84.71
DARN $n_h = 500$, adaNoise	84.13

Figure 5: https://arxiv.org/abs/1505.05770

Results: CIFAR-10

Table 3. Test set performance on the CIFAR-10 data.

	$K = 0$	$K = 2$	$K = 5$	$K = 10$
$-\ln p(\mathbf{x})$	-293.7	-308.6	-317.9	-320.7

Figure 6: https://arxiv.org/abs/1505.05770

Conclusion

- The distribution $p(z|x)$ may be highly non-Gaussian
- We take $q(z|x)$ to be something that starts off as a simple density, then transforms through normalizing flows
- We recover $p(z|x)$ in the limit as $K \to \infty$

Convolutional/Deconvolutional VAEs

Learning to Generate Chairs, Tables and Cars with Convolutional Networks

University of Toronto

Friday October 14^{th}, 2016

Overview

1. Motivation
2. Problem Description
3. Experiments

Recall the ELBO

- The ELBO can be written as: $\mathbb{E}_{z\sim Q_\phi}[\log P_\theta(X|z)] - \mathcal{D}[Q_\phi(z|X)||P(z)]$
- The decoder is any function $f : Z \times \theta \rightarrow X$ that is continuous in θ and allows us to evaluate $P_\theta(X|z)$. Similarly the encoder can be any function continuous in ϕ that allows quick evaluation of $Q_\phi(z|X)$
- We can use 'structured' encoders and decoders (such as RNNs/CNNs) that are better suited to our specific problem.

Dealing with image inputs to VAEs

- We saw the demo of a VAE trained on MNIST that uses an MLP as its encoder and decoder
- A CNN seems like a more natural choice for encoding the latent state especially for **large images**
- How do we invert the convolution and pooling operations to get a decoder?

Up-convolution

- In order to map a dense representation to a high dimensional image, we need to **unpool** the feature maps (i.e. increase their spatial span)
- One simple approach is to replace each entry of a feature map by an $s \times s$ block with the entry value in the top left corner and zeros elsewhere, followed by a convolution step.

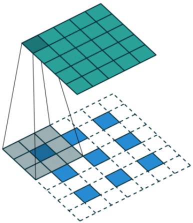


FIGURE 1: Unpooling and Convolution

Source: What are deconvolutional layers?

Generating 2-D projections from 3-D models

Given a set of 3D models (of chairs, tables, or cars), , train a neural network capable of generating 2D projec- tions of the models given:

- Model number (defining the style)
- Viewpoint (azimuth and elevation)
- Transformation parameters(color, brightness, saturation, zoom, etc)

FIGURE 2: 2-D projections of chairs, tables and cars

Generating 2-D projections from 3-D models

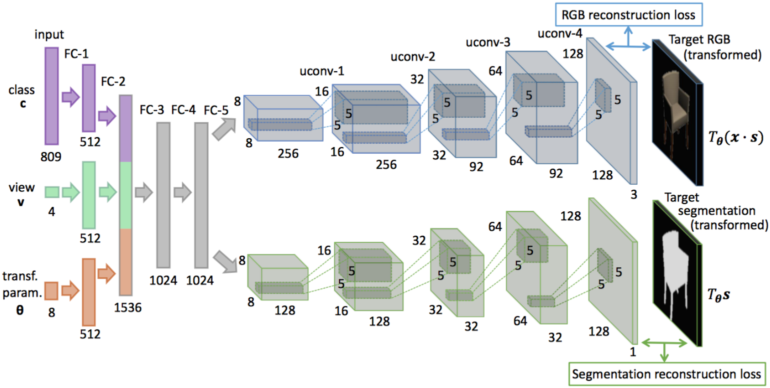


FIGURE 3: Architecture of network that generates 128x128 images

Network Architecture

- The class identity is passed through an inference network that **encodes** it to a latent state z such that $z^i \sim q_\phi(z^i|c)$
- The up-convolutional network then **decodes** the latent state to give the mean of the Gaussian distribution for the image

$$p(T_\theta^i(\mathbf{x}^i.\mathbf{s}^i)|\mathbf{z}^i,\theta^i,\mathbf{v}^i) = \mathcal{N}(u_{RGB}(\hat{h}(\mathbf{z}^i,\mathbf{v}^i,\theta^i),\mathbf{\Sigma}) \quad (1)$$

- The objective is the variational lower bound:

$$\mathbb{E}_z[\log p(T_{\theta_i}(\mathbf{x}^i.\mathbf{s}^i|\mathbf{z}^i)) + \log p(T_{\theta_i}\mathbf{s}^i|\mathbf{z}^i)] - KL(q(\mathbf{z}|\mathbf{c}^i)||p(\mathbf{z}^i)) \quad (2)$$

with $z \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$

Experiments

FIGURE 4: Interpolating between chairs

Experiments

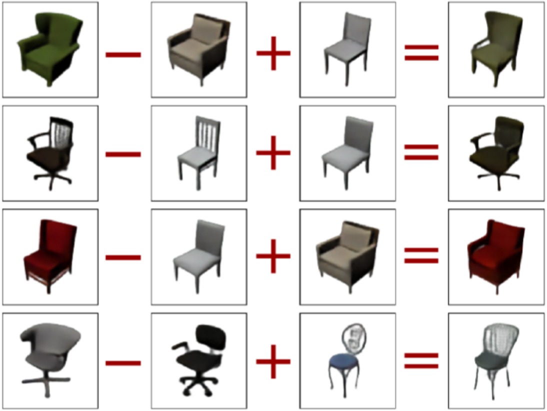


Figure 5: Feature arithmetics

Experiments

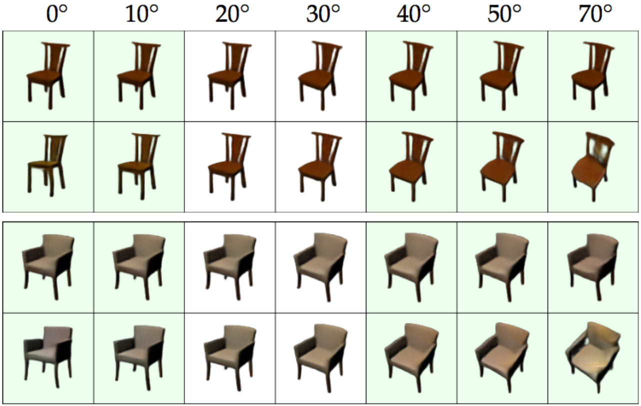


FIGURE 6: Knowledge transfer

Generating sentences from a continuous space

YULIA RUBANOVA

Class presentation for CSC 2541

Sequence autoencoder

Both encoder and decoder are RNNs

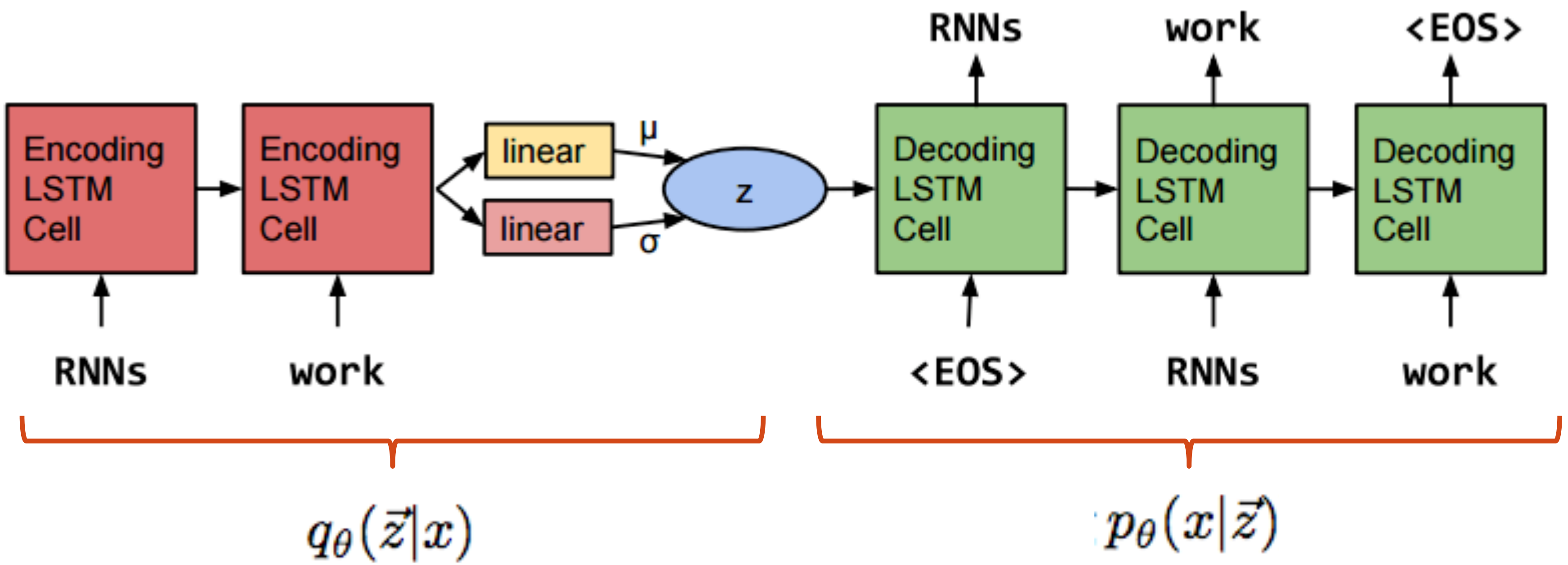
- Generates one word at a time
- Can model complex distributions over sentences with long-term dependences

- Does not incorporate global features (style, topic, high-level syntactic features)

i went to the store to buy some groceries
i store to buy some groceries .
i were to buy any groceries .
horses are to buy any groceries .
horses are to buy any animal .
horses the favorite any animal .
horses the favorite favorite animal .
horses are my favorite animal .

Sentences produced by greedily decoding from points between two sentence encodings with a conventional autoencoder.

RNN variational autoencoder



Optimization challenge

Model encodes useful information in **Z** when:

- KL divergence between posterior and prior is non-zero
- Reconstruction error is relatively small

Otherwise model learns to ignore z and sets q(z|x) to be equal to p(z)

$$
\begin{aligned}
\mathcal{L}(\theta; x) = & -\mathrm{KL}(q_\theta(\vec{z}|x) || p(\vec{z})) \\
& + \mathbb{E}_{q_\theta(\vec{z}|x)}[\log p_\theta(x|\vec{z})] \\
\leq & \log p(x) \quad .
\end{aligned}
$$

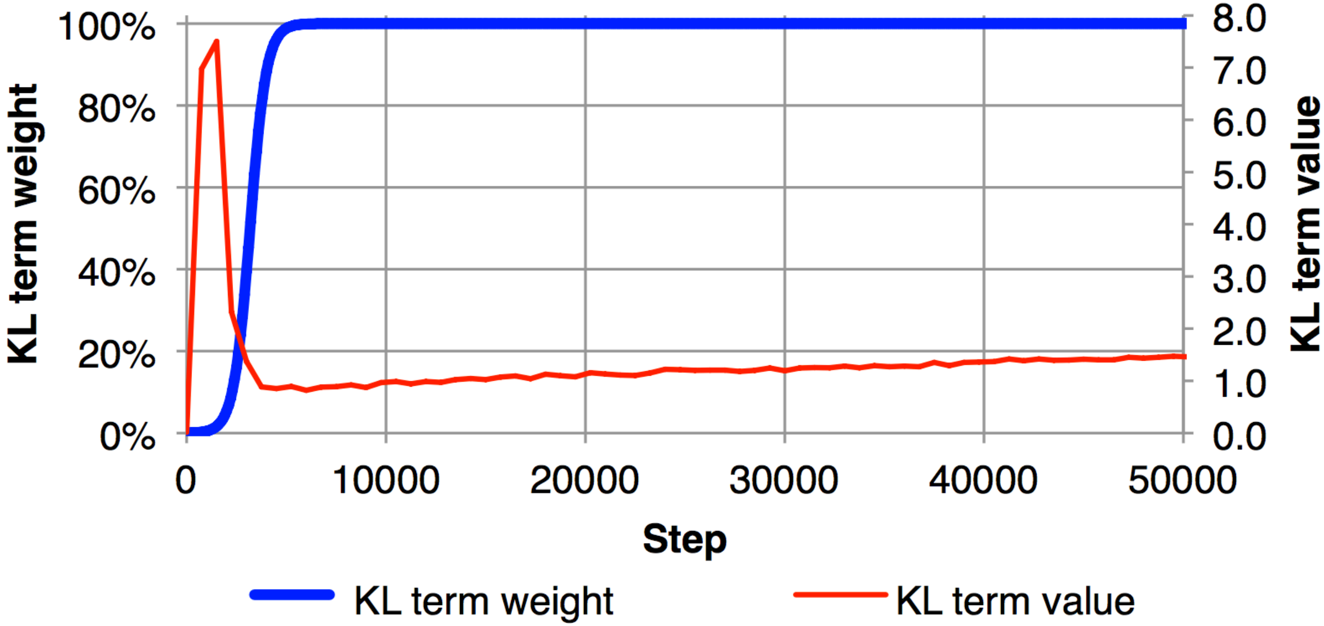
KL divergence of the posterior from the prior

Reconstruction error

Solution to optimization issue

During training, add variable weight **w** to the KL term.

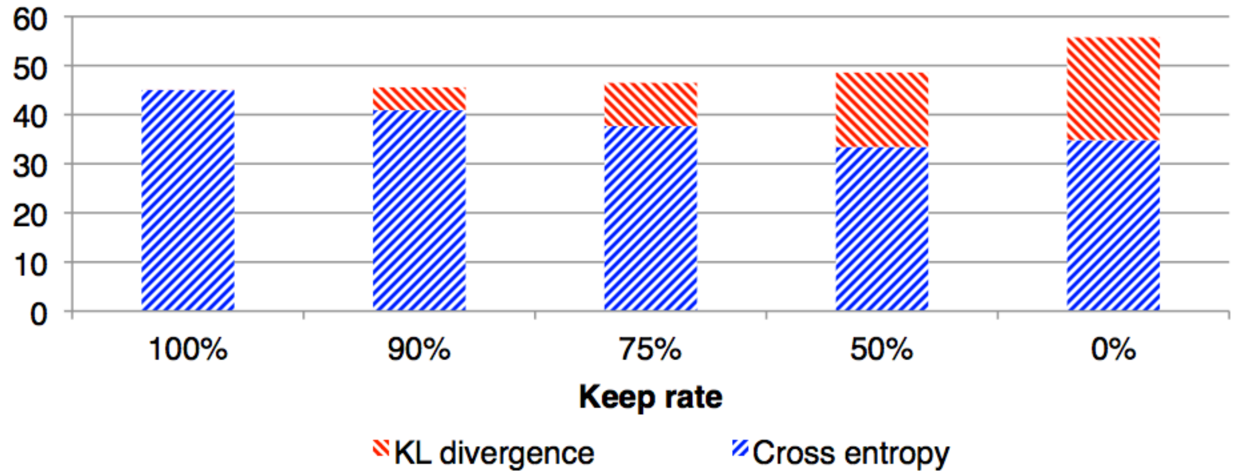
At start set **w** zero, then increase this weight.



Word dropout

Randomly replace some fraction of the conditioned-on word tokens with the generic unknown word token UNK

This forces the model to rely on **Z** to make good predictions



Word dropout

100% word keep	**75% word keep**
“ no , ” he said . *“ thank you , ” he said .*	*“ love you , too . ”* *she put her hand on his shoulder and followed him to the door .*
50% word keep	**0% word keep**
“ maybe two or two . ” *she laughed again , once again , once again , and thought about it for a moment in long silence .*	*i i hear some of of of* *i was noticed that she was holding the in in of the the in*

Interpolation

$\vec{z}(t) = \vec{z}_1 * (1-t) + \vec{z}_2 * t$ with $t \in [0, 1]$.

" i want to talk to you . "
"i want to be with you . "
"i do n't want to be with you . "
i do n't want to be with you .
she did n't want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .

Paths between pairs of random points in VAE space

More examples

Three sentences which were used as inputs to the VAE, presented with greedy decodes from the mean of the posterior distribution, and from three samples from that distribution.

INPUT	**we looked out at the setting sun .**	**i went to the kitchen .**	**how are you doing ?**
MEAN	*they were laughing at the same time .*	*i went to the kitchen .*	*what are you doing ?*
SAMP. 1	*ill see you in the early morning .*	*i went to my apartment .*	*" are you sure ?*
SAMP. 2	*i looked up at the blue sky .*	*i looked around the room .*	*what are you doing ?*
SAMP. 3	*it was down on the dance floor .*	*i turned back to the table .*	*what are you doing ?*

Summary

Variational autoencoder:

- Can decode plausible sentences from every reasonable point in the latent space
- Produces coherent new sentences that interpolate between known sentences
- Models global features in a continuous latent variable **Z**
- Has similar performance to RNN, when global representation is not necessary

DeepMind DRAW

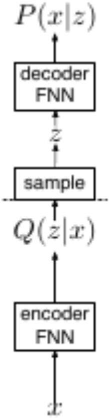
Deep Recurrent Attentive Writer

CSC2541 Structured Encoder/Decoders

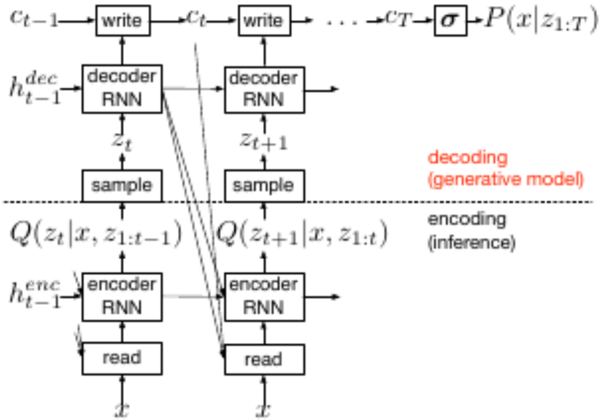
October 14, 2016

Generative Modelling

Conventional VAEs



DRAW Architecture



DRAW: Network

Iterative equations

For $t = 1, ..., T$, DRAW computes

$$\hat{x}_t = x - \boldsymbol{\sigma}(c_{t-1})$$

$$r_t = read(x, \hat{x}_t, h_{t-1}^{dec})$$

$$h_t^{enc} = RNN^{enc}(h_{t-1}^{enc}, [r_t, h_{t-1}^{dec}])$$

$$z_t \sim Q(Z_t | h_t^{enc})$$

$$h_t^{dec} = RNN^{dec}(h_{t-1}^{dec}, z_t)$$

$$c_t = c_{t-1} + write(h_t^{dec})$$

DRAW: Network

Latent distribution

- Diagonal gaussian $Q(Z_t|h_t^{enc}) = \mathcal{N}(Z_t|\mu_t, \sigma_t)$ parametrised by

$$\mu_t = W(h_t^{enc})$$
$$\sigma_t = \exp(\mu_t)$$

DRAW: Network

Latent distribution

- Diagonal gaussian $Q(Z_t|h_t^{enc}) = \mathcal{N}(Z_t|\mu_t, \sigma_t)$ parametrised by

$$\mu_t = W(h_t^{enc})$$
$$\sigma_t = \exp(\mu_t)$$

- latent prior $P(Z_t) = \mathcal{N}(0, I)$

DRAW: Network

Latent distribution

- Diagonal gaussian $Q(Z_t|h_t^{enc}) = \mathcal{N}(Z_t|\mu_t, \sigma_t)$ parametrised by

$$\mu_t = W(h_t^{enc})$$
$$\sigma_t = \exp(\mu_t)$$

- latent prior $P(Z_t) = \mathcal{N}(0, I)$
- easy to sample;

DRAW: Network

Latent distribution

- Diagonal gaussian $Q(Z_t|h_t^{enc}) = \mathcal{N}(Z_t|\mu_t, \sigma_t)$ parametrised by

$$\mu_t = W(h_t^{enc})$$
$$\sigma_t = \exp(\mu_t)$$

- latent prior $P(Z_t) = \mathcal{N}(0, I)$
- easy to sample; simplified loss

DRAW: Network

Latent distribution

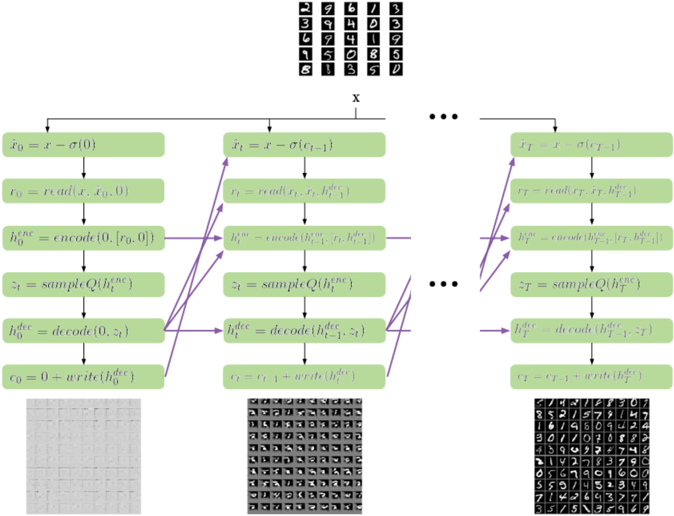
- Diagonal gaussian $Q(Z_t|h_t^{enc}) = \mathcal{N}(Z_t|\mu_t, \sigma_t)$ parametrised by

$$\mu_t = W(h_t^{enc})$$
$$\sigma_t = \exp(\mu_t)$$

- latent prior $P(Z_t) = \mathcal{N}(0, I)$
- easy to sample; simplified loss
- $reparameterization$ trick enables efficient gradient propagation

DRAW: Network

Unrolled graph



DRAW: Network

Loss

- *reconstruction loss* given model $D(X|c_T)$

$$\mathcal{L}^x = -\log D(x|c_T)$$

DRAW: Network

Loss

- *reconstruction loss* given model $D(X|c_T)$

$$\mathcal{L}^x = -\log D(x|c_T)$$

- *latent loss*

$$\mathcal{L}^z = \sum_{t=1}^{T} KL(Q(Z_t|h_t^{enc})||P(Z_t))$$

- *reconstruction loss* given model $D(X|c_T)$

$$\mathcal{L}^x = -\text{log } D(x|c_T)$$

- *latent loss*

$$\mathcal{L}^z = \sum_{t=1}^{T} KL(Q(Z_t|h_t^{enc})||P(Z_t))$$

- Gaussian latents $\Rightarrow \mathcal{L}^z = \frac{1}{2}\left(\sum_{t=1}^{T} \mu_t^2 + \sigma_t^2 - \text{log}\sigma_t^2\right) - T/2$
- *total loss*

$$\mathcal{L} = \left\langle \mathcal{L}^x + \mathcal{L}^z \right\rangle_{z \sim Q}$$

DRAW without attention

- *read* and *write* operations

$$read(x, \hat{x}_t, h_{t-1}^{dec}) = [x, x_t]$$

$$write(h_t^{dec}) = W(h_t^{dec})$$

DRAW without attention

- *read* and *write* operations

$$read(x, \hat{x}_t, h^{dec}_{t-1}) = [x, x_t]$$

$$write(h^{dec}_t) = W(h^{dec}_t)$$

- network doesn't know "where to read" and "where to write"

DRAW with attention

- attention parameters

$$(\hat{g}_X, \hat{g}_Y, \log \sigma^2, \log \hat{\delta}, \log \gamma) = W(h_t^{dec})$$

- scale centre location and stride

$$g_X = \frac{A+1}{2}(\hat{g}_X + 1)$$

$$g_Y = \frac{B+1}{2}(\hat{g}_Y + 1)$$

$$\delta = \frac{\max(A, B) - 1}{N - 1}\hat{\delta}$$

- mean shift (translation) vectors

$$\mu_X^i = g_X + (i - N/2 - 0.5)\delta$$

$$\mu_Y^j = g_Y + (j - N/2 - 0.5)\delta$$

DRAW with attention

- N × N patches extracted from both image and error image

$$F_X[i,a] = \frac{1}{Z_X}\text{exp}\left(-\frac{(a-\mu_X^i)^2}{2\sigma^2}\right)$$

$$F_X[j,b] = \frac{1}{Z_Y}\text{exp}\left(-\frac{(b-\mu_Y^j)^2}{2\sigma^2}\right)$$

$$read(x,\hat{x}_t,h_{t-1}^{dec}) = \gamma[F_Y x F_X^T, F_Y \hat{x}_t F_X^T]$$

$$w_t = W(h_t^{dec})$$

$$write(h_t^{dec}) = \frac{1}{\hat{\gamma}}\hat{F}_Y^T w_t \hat{F}_X$$

DRAW with attention

Visualization

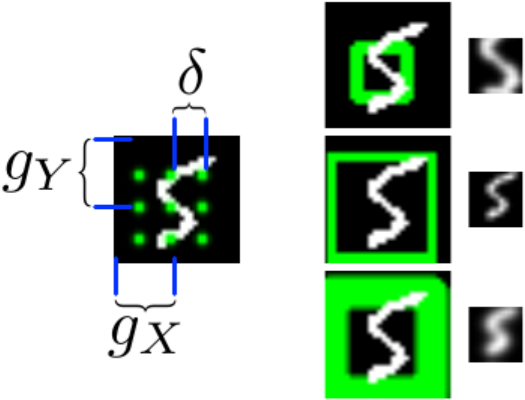
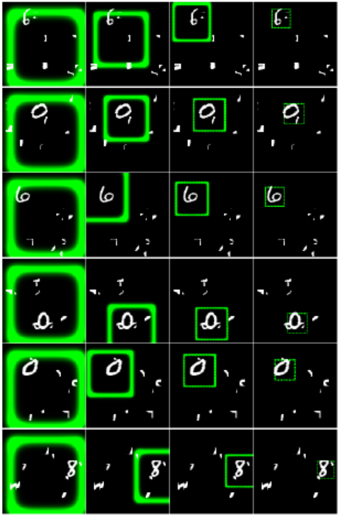


Figure: Effect of varying δ and σ

DRAW Cluttered MNIST Classification



DRAW Cluttered MNIST Classification

Table 1. Classification test error on 100×100 Cluttered Translated MNIST.

Model	Error
Convolutional, 2 layers	14.35%
RAM, 4 glimpses, 12×12, 4 scales	9.41%
RAM, 8 glimpses, 12×12, 4 scales	8.11%
Differentiable RAM, 4 glimpses, 12×12	4.18%
Differentiable RAM, 8 glimpses, 12×12	**3.36%**

Temporal refinement

RNN $\Rightarrow$ joint distribution factorizes as product of conditionals, iterative refinement, as opposed to single step emission

Spatial attention

dynamic attention mechanism increases capability by attending to smaller regions

Complexity

spatio-temporal properties reduce complexity burden that the autoencoder learns, allowing for handling of more complex, larger distributions

Further reading

- DRAW paper
- Eric Jang's blog

Conclusion

In our presentation today we discussed several ways of improving VAEs and making them more suited to several classes of problems

- How to efficiently sample from our approximate posterior while being able to learn the parameters of the distribution using the reparametrization trick
- Approaches for modelling complicated posterior distributions using IWAE and Normalizing flows.
- Using encoders and decoders that utilize the structure of the specific problem such as:
  - Convolutional/Deconvolutional network for images
  - RNN as encoder/decoder for language models
  - RNN with attention for generating images

References I

Diederik P Kingma, Tim Salimans, and Max Welling.
Variational dropout and the local reparameterization trick.
arXiv preprint arXiv:1506.02557, 2015.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov.
Importance weighted autoencoders.
arXiv preprint arXiv:1509.00519, 2015.

Danilo Jimenez Rezende and Shakir Mohamed.
Variational inference with normalizing flows.
arXiv preprint arXiv:1505.05770, 2015.

References II

Alexey Dosovitskiy, Jost Springenberg, Maxim Tatarchenko, and Thomas Brox.
Learning to generate chairs, tables and cars with convolutional networks.
2016.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio.
Generating sentences from a continuous space.
arXiv preprint arXiv:1511.06349, 2015.

Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra.
Draw: A recurrent neural network for image generation.
arXiv preprint arXiv:1502.04623, 2015.