

Generative Models: Recent Progresses and Applications

CEDL 2016

- Auto-Encoding Variational Bayes
- Bidirectional Helmholtz Machines
- Energy-Based Generative Adversarial Network

Auto-Encoding Variational Bayes

Diederik P. Kingma

Machine Learning Group
Universiteit van Amsterdam
dpkingma@gmail.com

Max Welling

Machine Learning Group
Universiteit van Amsterdam
welling.max@gmail.com

Abstract

How can we perform efficient inference and learning in directed probabilistic models, in the presence of continuous latent variables with intractable posterior distributions, and large datasets? We introduce a stochastic variational inference and learning algorithm that scales to large datasets and, under some mild differentiability conditions, even works in the intractable case. Our contributions is two-fold. First, we show that a reparameterization of the variational lower bound yields a lower bound estimator that can be straightforwardly optimized using standard stochastic gradient methods. Second, we show that for i.i.d. datasets with continuous latent variables per datapoint, posterior inference can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator. Theoretical advantages are reflected in experimental results.

Directed Graphical Models with Continuous Latent Variables

$$\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^N$$

dataset

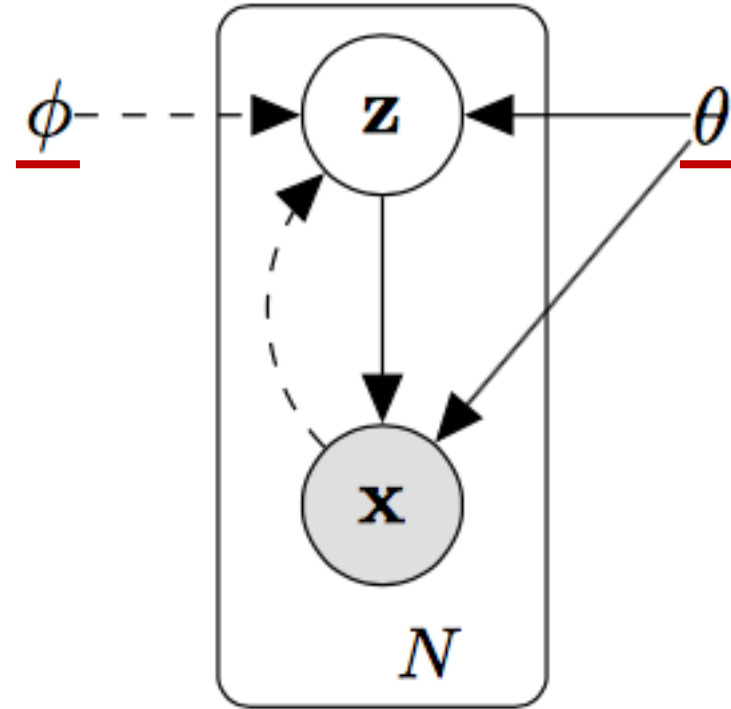


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model $p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})$, dashed lines denote the variational approximation $q_{\phi}(\mathbf{z}|\mathbf{x})$ to the intractable posterior $p_{\theta}(\mathbf{z}|\mathbf{x})$. The variational parameters ϕ are learned jointly with the generative model parameters θ .

Intractability and large datasets

Marginal likelihood $p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})$

Likelihood function:
neural network with non-linear hidden layers

Posterior $p_{\theta}(\mathbf{z}|\mathbf{x}) = p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})/p_{\theta}(\mathbf{x})$

Recognition model $q_{\phi}(\mathbf{z}|\mathbf{x})$

Intractability and large datasets

Marginal likelihood $p_{\theta}(\mathbf{x}) = \int p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x}|\mathbf{z})$

Decoder


Likelihood function:
neural network with non-linear hidden layers


Posterior $p_{\theta}(\mathbf{z}|\mathbf{x}) = p_{\theta}(\mathbf{x}|\mathbf{z})p_{\theta}(\mathbf{z})/p_{\theta}(\mathbf{x})$


Recognition model $q_{\phi}(\mathbf{z}|\mathbf{x})$ Encoder

Variational Bound

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}) = \sum_{i=1}^N \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})$$


$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) = D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$


$$\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} [-\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})]$$


$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})} [\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z})]$$

Gradient Estimator

- Impractical due to high variance
- Needs many samples

$$\nabla_{\phi} \mathbb{E}_{q_{\phi}(\mathbf{z})} [f(\mathbf{z})] = \mathbb{E}_{q_{\phi}(\mathbf{z})} [f(\mathbf{z}) \nabla_{q_{\phi}(\mathbf{z})} \log q_{\phi}(\mathbf{z})]$$

$$\simeq \frac{1}{L} \sum_{l=1}^L f(\mathbf{z}) \nabla_{q_{\phi}(\mathbf{z}^{(l)})} \log q_{\phi}(\mathbf{z}^{(l)})$$

$$\mathbf{z}^{(l)} \sim q_{\phi}(\mathbf{z} | \mathbf{x}^{(i)})$$

Stochastic Gradient Variational Bayes

- Differential transform

$$\tilde{\mathbf{z}} = g_{\phi}(\boldsymbol{\epsilon}, \mathbf{x}) \quad \text{with} \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$$

$$\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} [f(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})} [f(g_{\phi}(\boldsymbol{\epsilon}, \mathbf{x}^{(i)}))] \simeq \frac{1}{L} \sum_{l=1}^L f(g_{\phi}(\boldsymbol{\epsilon}^{(l)}, \mathbf{x}^{(i)})) \quad \text{where} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

$$\tilde{\mathcal{L}}^A(\boldsymbol{\theta}, \phi; \mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^L \log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, \mathbf{z}^{(i,l)}) - \log q_{\phi}(\mathbf{z}^{(i,l)} | \mathbf{x}^{(i)})$$

$$\text{where} \quad \mathbf{z}^{(i,l)} = g_{\phi}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)}) \quad \text{and} \quad \boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$$

Another Version of SGVB

$$\tilde{\mathcal{L}}^B(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)}) = -D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\boldsymbol{\theta}}(\mathbf{z})) + \frac{1}{L} \sum_{l=1}^L (\log p_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}))$$

where $\mathbf{z}^{(i,l)} = g_{\boldsymbol{\phi}}(\boldsymbol{\epsilon}^{(i,l)}, \mathbf{x}^{(i)})$ and $\boldsymbol{\epsilon}^{(l)} \sim p(\boldsymbol{\epsilon})$

Minibatch

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}) \simeq \tilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M) = \frac{N}{M} \sum_{i=1}^M \tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}^{(i)})$$

$$\mathbf{X}^M = \{\mathbf{x}^{(i)}\}_{i=1}^M$$

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\theta, \phi \leftarrow$ Initialize parameters

repeat

$\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)

$\epsilon \leftarrow$ Random samples from noise distribution $p(\epsilon)$

$\mathbf{g} \leftarrow \nabla_{\theta, \phi} \tilde{\mathcal{L}}^M(\theta, \phi; \mathbf{X}^M, \epsilon)$ (Gradients of minibatch estimator (8))

$\theta, \phi \leftarrow$ Update parameters using gradients \mathbf{g} (e.g. SGD or Adagrad [DHS10])

until convergence of parameters (θ, ϕ)

return θ, ϕ

Re-parameterization Trick

In order to solve our problem we invoked an alternative method for generating samples from $q_\phi(\mathbf{z}|\mathbf{x})$. The essential parameterization trick is quite simple. Let \mathbf{z} be a continuous random variable, and $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$ be some conditional distribution. It is then often possible to express the random variable \mathbf{z} as a deterministic variable $\mathbf{z} = g_\phi(\epsilon, \mathbf{x})$, where ϵ is an auxiliary variable with independent marginal $p(\epsilon)$, and $g_\phi(\cdot)$ is some vector-valued function parameterized by ϕ .

$$\int q_\phi(\mathbf{z}|\mathbf{x}) f(\mathbf{z}) d\mathbf{z} \simeq \frac{1}{L} \sum_{l=1}^L f(g_\phi(\mathbf{x}, \epsilon^{(l)}))$$

Example: univariate Gaussian

$$z \sim p(z|x) = \mathcal{N}(\mu, \sigma^2)$$

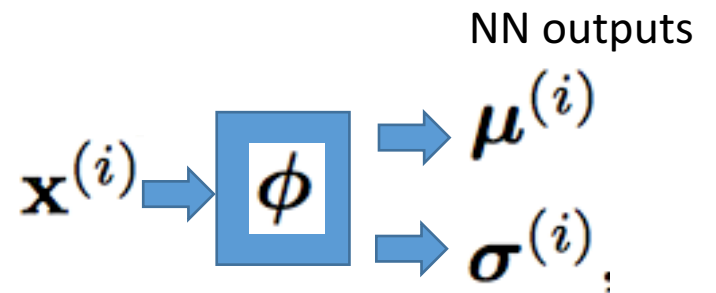
$$z = \mu + \sigma\epsilon$$

$$\epsilon \sim \mathcal{N}(0, 1)$$

$$\mathbb{E}_{\mathcal{N}(z;\mu,\sigma^2)} [f(z)] = \mathbb{E}_{\mathcal{N}(\epsilon;0,1)} [f(\mu + \sigma\epsilon)] \simeq \frac{1}{L} \sum_{l=1}^L f(\mu + \sigma\epsilon^{(l)}) \text{ where } \epsilon^{(l)} \sim \mathcal{N}(0, 1).$$

Vairational Auto-Encoder

$q_\phi(\mathbf{z}|\mathbf{x})$ Neural network Approximation to $p_\theta(\mathbf{x}, \mathbf{z})$

$$\log q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) = \log \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}^{(i)}, \boldsymbol{\sigma}^{2(i)} \mathbf{I})$$


NN outputs

$$\mathbf{z}^{(i,l)} = g_\phi(\mathbf{x}^{(i)}, \boldsymbol{\epsilon}^{(l)}) = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \text{ where } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \simeq \frac{1}{2} \sum_{j=1}^J \left(1 + \log((\sigma_j^{(i)})^2) - (\mu_j^{(i)})^2 - (\sigma_j^{(i)})^2 \right) + \frac{1}{L} \sum_{l=1}^L \log p_\theta(\mathbf{x}^{(i)} | \mathbf{z}^{(i,l)})$$

$$\text{where } \mathbf{z}^{(i,l)} = \boldsymbol{\mu}^{(i)} + \boldsymbol{\sigma}^{(i)} \odot \boldsymbol{\epsilon}^{(l)} \text{ and } \boldsymbol{\epsilon}^{(l)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$$

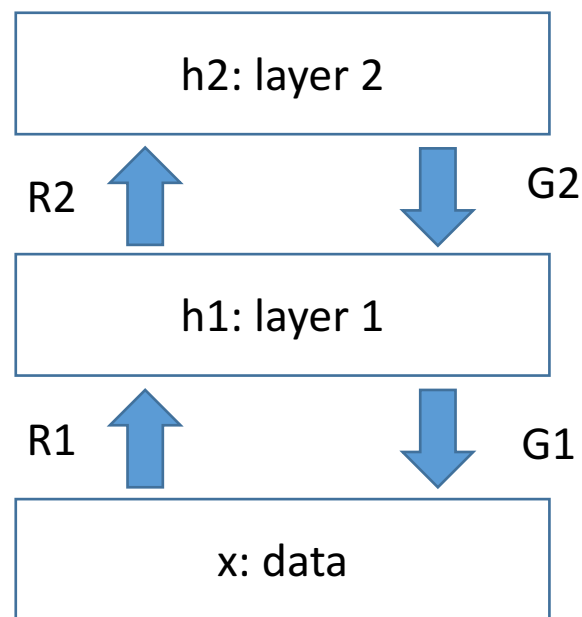
Bidirectional Helmholtz Machines

Jörg Bornschein*
Samira Shabanian
Asja Fischer
Yoshua Bengio*

Dept. Computer Science and Operations Research, University of Montreal

* Canadian Institute for Advanced Research (CIFAR)

BORNJ@IRO.UMONTREAL.CA
SHABANIS@IRO.UMONTREAL.CA
FISCHER@IRO.UMONTREAL.CA
BENGIOY@IRO.UMONTREAL.CA

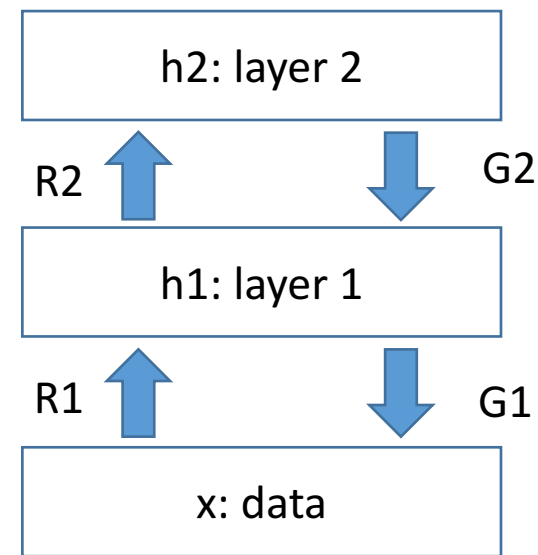


$$p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = \frac{1}{Z} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)}$$

$$p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = p(\mathbf{h}_2) p(\mathbf{h}_1 | \mathbf{h}_2) p(\mathbf{x} | \mathbf{h}_1)$$

$$q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) = q(\mathbf{x}) q(\mathbf{h}_1 | \mathbf{x}) q(\mathbf{h}_2 | \mathbf{h}_1)$$

$$\begin{aligned} q(\mathbf{x}) &= p^*(\mathbf{x}) = \sum_{\mathbf{h}_1, \mathbf{h}_2} p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \\ &= \frac{\sqrt{q(\mathbf{x})}}{Z} \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{h}_1 | \mathbf{x}) q(\mathbf{h}_2 | \mathbf{h}_1)} \\ &= \left(\frac{1}{Z} \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{h}_1 | \mathbf{x}) q(\mathbf{h}_2 | \mathbf{h}_1)} \right)^2. \end{aligned}$$



- Cauchy-Schwarz inequality

$$|\sum_y f(y)g(y)|^2 \leq \sum_y |f(y)|^2 \times \sum_y |g(y)|^2$$

$$Z = \sum_{\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)} \leq 1$$

- Lower bound

$$\begin{aligned} \tilde{p}^*(\mathbf{x}) &= \left(\sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2)q(\mathbf{h}_1|\mathbf{x})q(\mathbf{h}_2|\mathbf{h}_1)} \right)^2 \\ &= Z^2 p^*(\mathbf{x}) \leq p^*(\mathbf{x}) . \end{aligned}$$

Optimization

pressured to find a maximum likelihood solution that yields
 $p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \approx q(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) \approx p^*(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2).$

Alternative View: Bhattacharyya Distance

- Bhattacharyya distance $D_B(p, q) = -\log \sum_y \sqrt{p(y)q(y)}$

$$\begin{aligned} \log p^*(\mathbf{x}) &= 2 \log \sum_{\mathbf{h}_1, \mathbf{h}_2} \sqrt{p(\mathbf{x}, \mathbf{h}_1, \mathbf{h}_2) q(\mathbf{h}_1 | \mathbf{x}) q(\mathbf{h}_2 | \mathbf{h}_1)} \\ &\quad - 2 \log \sum_{\mathbf{x}', \mathbf{h}'_1, \mathbf{h}'_2} \sqrt{p(\mathbf{x}', \mathbf{h}'_1, \mathbf{h}'_2) q(\mathbf{x}' | \mathbf{h}'_1, \mathbf{h}'_2)} \\ &= \log \tilde{p}^*(\mathbf{x}) + 2 D_B(p, q) \geq \log \tilde{p}^*(\mathbf{x}) \end{aligned}$$

Compared with Variational Bayes

$$D_{KL}(q(\mathbf{h}|\mathbf{x}) || p(\mathbf{h}|\mathbf{x})) = \sum_h q(\mathbf{h}|\mathbf{x}) \log \frac{q(\mathbf{h}|\mathbf{x})}{p(\mathbf{h}|\mathbf{x})} \geq 0$$

$$\begin{aligned} \log p(\mathbf{x}) &= \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x})} [\log p(\mathbf{x}, \mathbf{h}) - \log q(\mathbf{h}|\mathbf{x})] \\ &\quad + D_{KL}(q(\mathbf{h}|\mathbf{x}) || p(\mathbf{h}|\mathbf{x})) \\ &\geq \mathbb{E}_{\mathbf{h} \sim q(\mathbf{h}|\mathbf{x})} [\log p(\mathbf{x}, \mathbf{h}) - \log q(\mathbf{h}|\mathbf{x})] \end{aligned}$$

Differences

1. The KL-divergence in variational methods measures the distance between distributions given some training data. The Bhattacharyya distance in contrast quantifies a property of the model $p^*(x, h_1, h_2)$ independently of any training data. $D_B(p, q) = -\log Z$.
2. The variational lower bound is typically used to construct approximate inference algorithms. The bound $\tilde{p}^*(x)$ is just used to remove the normalization constant from the target distribution $p^*(x)$

Advantage

Goal: to learn a generative model p^* that is regularized to be close to the model q which we use to perform approximate inference for p^*

two equally well trained models $\mathbb{E} [\log p_{\theta_1}^*(\mathbf{x})] = \mathbb{E} [\log p_{\theta_2}^*(\mathbf{x})]$

$$\mathbb{E} [\log \tilde{p}_{\theta_1}^*(\mathbf{x})] > \mathbb{E} [\log \tilde{p}_{\theta_2}^*(\mathbf{x})] \Rightarrow D_B(p_{\theta_1}, q_{\theta_1}) < D_B(p_{\theta_2}, q_{\theta_2})$$

Algorithm 1 Training $p^*(\mathbf{x})$ using K importance samples

for number of training iterations **do**

- Sample \mathbf{x} from the training distribution (i.e. $\mathbf{x} \sim \mathcal{D}$)

for $k = 1, 2, \dots, K$ **do**

- Sample $\mathbf{h}_1^{(k)} \sim q(\mathbf{h}_1^{(k)} | \mathbf{x}); \mathbf{h}_l^{(k)} \sim q(\mathbf{h}_l^{(k)} | \mathbf{h}_{l-1}^{(k)})$
(for layers $l = 2$ to L)
- Compute $q(\mathbf{h}^{(k)} | \mathbf{x})$ and $p(\mathbf{x}, \mathbf{h}^{(k)})$
(for $\mathbf{h}^{(k)} = (\mathbf{h}_1^{(k)}, \dots, \mathbf{h}_L^{(k)})$)

end for

- Compute unnormalized importance weights

$$\omega_k = \sqrt{p(\mathbf{x}, \mathbf{h}^{(k)}) / q(\mathbf{h}^{(k)} | \mathbf{x})}$$
- Normalize the weights $\tilde{\omega}_k = \omega_k / \sum_{k'} \omega_{k'}$
- Update parameters of p and q : gradient descent
with gradient estimator

$$\sum_k \tilde{\omega}_k \frac{\partial}{\partial \theta} \log p(\mathbf{x}, \mathbf{h}^{(k)}) q(\mathbf{h}^{(k)} | \mathbf{x})$$

end for

Basic Sampling Algorithms

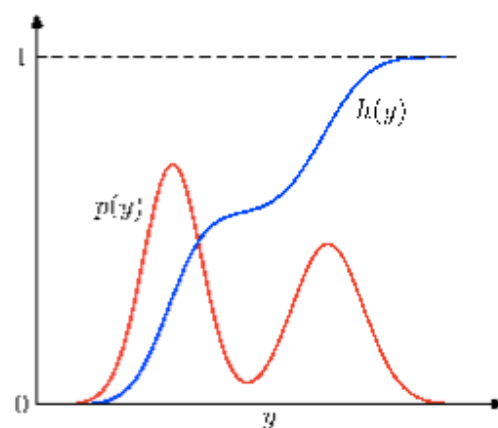
Pseudo-random number generator: $z \sim U(0, 1)$
 $U(0, 1)$ distributed uniformly over $(0, 1)$.

Transform z by $f(\cdot) \Rightarrow y = f(z)$

$$p(y) = p(z) \left| \frac{dz}{dy} \right|$$

$$z = h(y) \equiv \int_{-\infty}^y p(\hat{y}) d\hat{y}$$

$$y = h^{-1}(z)$$

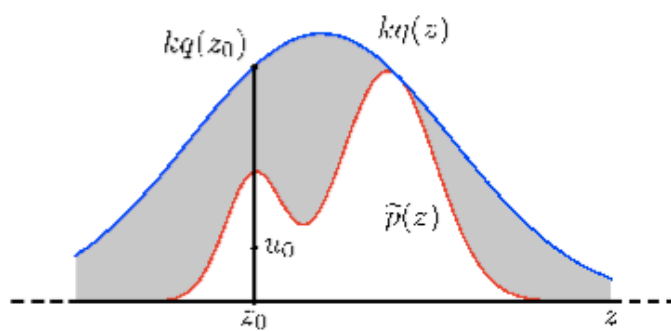


Rejection Sampling

We want to sample z from $p(z)$.

$$p(z) = \frac{1}{Z_p} \tilde{p}(z),$$

where $\tilde{p}(z)$ can easily be evaluated, but $Z_p = \int \tilde{p}(z) dz$ is unknown.



$q(z)$ such that $kq(z) \geq \tilde{p}(z)$: proposal distribution, from which we can easily draw samples

1. z_0 from $q(z)$
2. u_0 from $U(0, kq(z_0))$
3. if $u_0 > \tilde{p}(z_0)$, reject z_0

$$\begin{aligned}
 P(\text{accept}) &= \int \{\tilde{p}(z)/kq(z)\}q(z)\mathrm{d}z \\
 &= \frac{1}{k} \int \tilde{p}(z)\mathrm{d}z \\
 &= \frac{\int \tilde{p}(z)\mathrm{d}z}{k \int q(z)\mathrm{d}z}
 \end{aligned}$$

Acceptance rate for $\mathcal{N}(\mathbf{0}, \sigma_p^2 \mathbf{I}), \mathcal{N}(\mathbf{0}, \sigma_q^2 \mathbf{I})$

$$kq(\mathbf{z}) \geq p(\mathbf{z})$$

$$\frac{1}{k} = \left(\frac{\sigma_p}{\sigma_q}\right)^D$$

Importance Sampling

Suppose that it is impractical to sample directly from $p(\mathbf{z})$ but that we can evaluate $p(\mathbf{z})$ easily for any given value of \mathbf{z} .

To evaluate expectations by discretizing \mathbf{z} -space into a uniform grid and evaluate the integrand as a sum of the form

$$\mathbb{E}[f] \simeq \sum_{\ell=1}^L p(\mathbf{z}^{(\ell)}) f(\mathbf{z}^{(\ell)}) .$$

$q(\mathbf{z})$ proposal distribution
 $\{\mathbf{z}^{(\ell)}\}_{\ell=1}^L$ from $q(\mathbf{z})$

$$\begin{aligned} \mathbb{E}[f] &= \int f(\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \\ &= \int f(\mathbf{z}) \frac{p(\mathbf{z})}{q(\mathbf{z})} q(\mathbf{z}) d\mathbf{z} \\ &\simeq \frac{1}{L} \sum_{\ell=1}^L \frac{p(\mathbf{z}^{(\ell)})}{q(\mathbf{z}^{(\ell)})} f(\mathbf{z}^{(\ell)}) \end{aligned}$$

For unknown normalization

$$p(\mathbf{z}) = \frac{\tilde{p}(\mathbf{z})}{Z_p}$$

Use an important sampling distribution $q(\mathbf{z}) = \tilde{q}(\mathbf{z})/Z_q$

$$\begin{aligned}\mathbb{E}[f] &= \int f(\mathbf{z})p(\mathbf{z})d\mathbf{z} = \frac{Z_q}{Z_p} \int f(\mathbf{z})\frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})}q(\mathbf{z})d\mathbf{z} \\ &\simeq \frac{Z_q}{Z_p} \frac{1}{L} \sum_{\ell=1}^L \tilde{r}_\ell f(\mathbf{z}^{(\ell)})\end{aligned}$$

where $\tilde{r}_\ell = \tilde{p}(\mathbf{z}^{(\ell)})/\tilde{q}(\mathbf{z}^{(\ell)})$

$$\frac{Z_p}{Z_q} = \frac{1}{Z_q} \int \tilde{p}(\mathbf{z})d\mathbf{z} = \int \frac{\tilde{p}(\mathbf{z})}{\tilde{q}(\mathbf{z})}q(\mathbf{z})d\mathbf{z} \simeq \frac{1}{L} \sum_{\ell=1}^L \tilde{r}_\ell$$

Hence $\mathbb{E}[f] \simeq \sum_{\ell=1}^L w_\ell f(\mathbf{z}^{(\ell)})$ where $w_\ell = \tilde{r}_\ell / \sum_m \tilde{r}_m$.

ENERGY-BASED GENERATIVE ADVERSARIAL NETWORK

Junbo Zhao, Michael Mathieu and Yann LeCun

Department of Computer Science, New York University

Facebook Artificial Intelligence Research

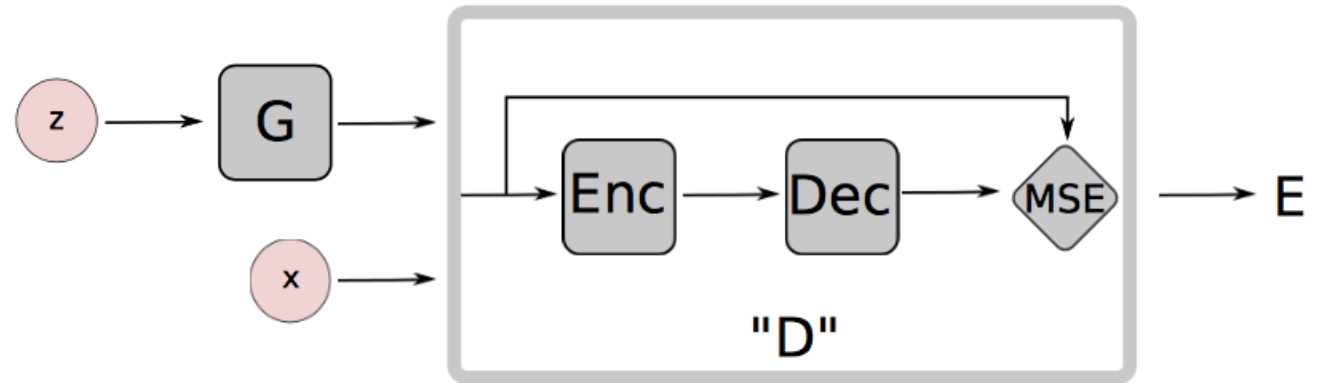
{jakezhao, mathieu}@nyu.edu, yann@fb.com

ABSTRACT

We introduce the “Energy-based Generative Adversarial Network” model (EBGAN) which views the discriminator as an energy function that associates low energies with the regions near the data manifold and higher energies with other regions. Similar to the probabilistic GANs, a generator is trained to produce contrastive samples with minimal energies, while the discriminator is trained to assign high energies to these generated samples. Viewing the discriminator as an energy function allows to use a wide variety of architectures and loss functionals in addition to the usual binary classifier with logistic output. Among them, an instantiation of EBGAN is to use an auto-encoder architecture, with the energy being the reconstruction error. We show that this form of EBGAN exhibits more stable behavior than regular GANs during training. We also show that a single-scale architecture can be trained to generate high-resolution images.

EBGAN Auto-Encoder Model

- Estimate the energy E



- margin m
 $[\cdot]^+ = \max(0, \cdot)$

$$\begin{aligned} f_D(x, z) &= D(x) + [m - D(G(z))]^+ \\ &= \|Dec(Enc(x)) - x\| + [m - \|Dec(Enc(G(z))) - G(z)\|]^+ \end{aligned}$$

$$\begin{aligned} f_G(z) &= \|D(G(z))\| \\ &= \|Dec(Enc(G(z))) - G(z)\| \end{aligned}$$

Repelling Regularizer (Pull-away Term)

- Problem of GAN: samples are clustered in one or a few modes of the regions of high data density, instead of spanning the whole range
 - Minibatch discrimination as a solution
 - Alternative: repelling regularizer
 - bs: batch size

$$f_{PT}(S) = \frac{1}{bs(bs-1)} \sum_i \sum_{j \neq i} \left(\frac{S_i^\top S_j}{\|S_i\| \|S_j\|} \right)^2$$

$$S = Enc(G(z))$$



Figure 5: Exemplar generation from the grid search on MNIST. Left(a): Best GAN generation. Middle(b): Best EBGAN generation. Right(c): Best EBGAN-PT generation.

SUPER-RESOLUTION WITH DEEP CONVOLUTIONAL SUFFICIENT STATISTICS

Joan Bruna

Department of Statistics
University of California, Berkeley
joan.bruna@berkeley.edu

Pablo Sprechmann

Courant Institute
New York University
pablo@cims.nyu.edu

Yann LeCun

Facebook, Inc., &
Courant Institute
New York University
yann@cims.nyu.edu

$$\min_{\Theta} \sum_i \|\Phi(x_i, \Theta) - y_i\|^2$$

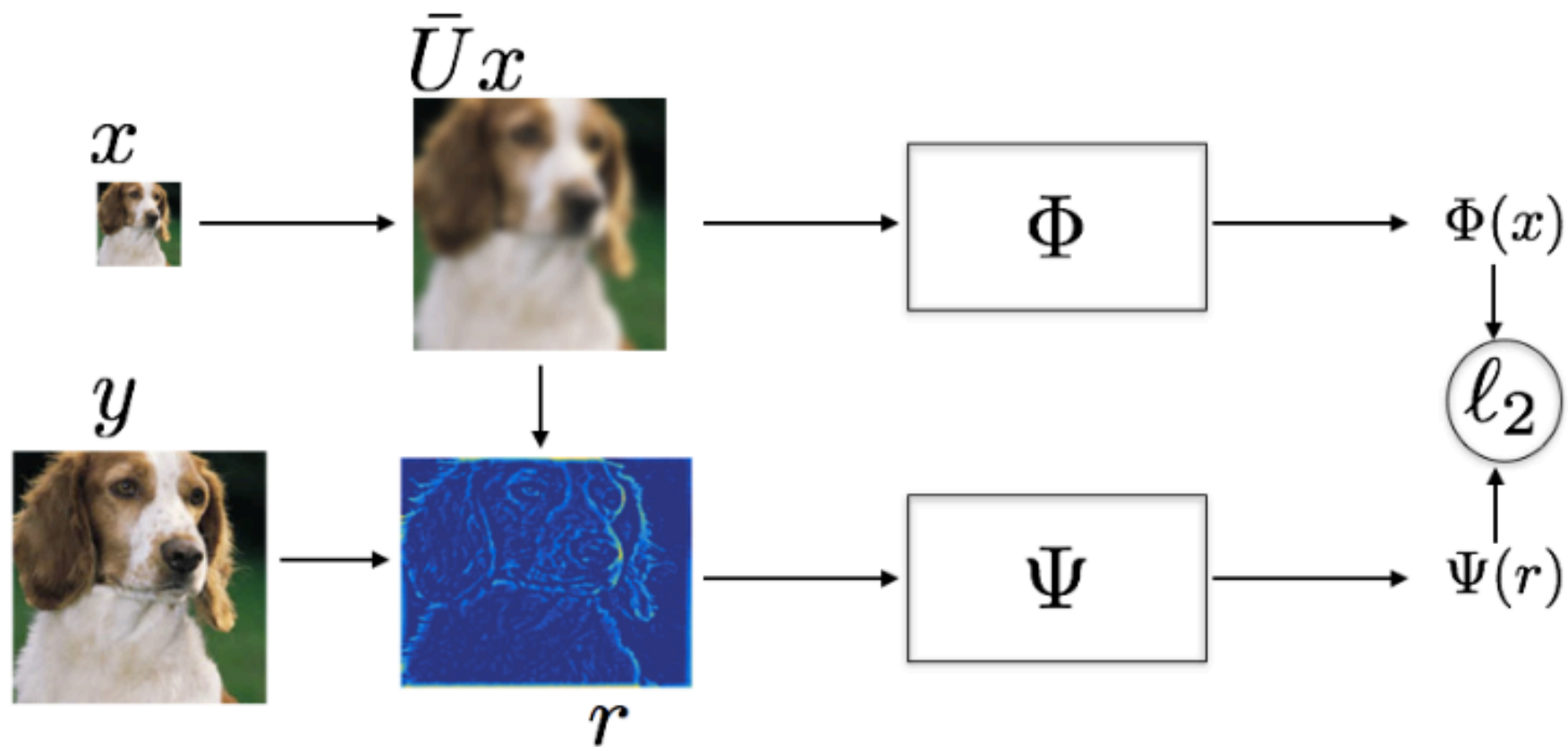
$$\hat{y} = \arg \min_y \frac{1}{2} \|U(y) - x\|^2 + \lambda \mathcal{R}(y)$$

↑
downsampling

Typical regularization:
sparsity

$$\mathcal{R}(y) = \|\nabla y\|_1$$

(A)



(B)

