Adversarially Learned Inference

authors: Montreal,Stanford,New York

published: 2016-12 ICLR-2017

> We introduce the adversarially learned inference model, which jointly learns a generation network and an inference network using an adversarial process. The generation network maps samples from stochastic latent variables to the data space while the inference network maps training examples in data space to the space of latent variables.

> A discriminator is trained to discriminate joint samples of the data and the corresponding latent variable from the encoder(or approximate posterior) from joint samples from the decoder while in opposition, the encoder and the decoder are trained together to fool the discriminator. Not only are we asking the discriminator to distinguish synthetic samples from real data, but we are requiring it to distinguish between two joint distributions over the data space and the latent variables.

- the encoder joint distribution $q(x, z) = q(x)q(z|x)$
- the decoder joint distribution $p(x, z) = p(z)p(x|z)$

> The encoder marginal $q(x)$ is the empirical data distribution and the decoder marginal $p(z)$ is usually defined to be a simple, factorized distribution, such as the standard Normal distribution $p(z) = N(0, I)$.

ALI's objective is to match the two joint distributions

> Joint pairs (x, z) are drawn either from $q(x, z)$ or $p(x, z)$, and a discriminator network learns to discriminate between the two, while the encoder and decoder networks are trained to fool the discriminator.

$$\min_G \max_D V(D,G) = E_{q(x)}[\log(D(x, G_z(x)))] + E_{p(z)}[\log(1 - D(G_x(z), z))] = \iint q(x)q(z|x)log(D(x,z))dxdz + \iint p(z)p(x$$

Algorithm 1 The ALI training procedure.

$\theta_g, \theta_d \leftarrow$ initialize network parameters

repeat

$\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(M)} \sim q(\boldsymbol{x})$ ▷ Draw M samples from the dataset and the prior

$\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(M)} \sim p(\boldsymbol{z})$

$\hat{\boldsymbol{z}}^{(i)} \sim q(\boldsymbol{z} \mid \boldsymbol{x} = \boldsymbol{x}^{(i)}), \quad i = 1, \ldots, M$ ▷ Sample from the conditionals

$\tilde{\boldsymbol{x}}^{(j)} \sim p(\boldsymbol{x} \mid \boldsymbol{z} = \boldsymbol{z}^{(j)}), \quad j = 1, \ldots, M$

$\rho_q^{(i)} \leftarrow D(\boldsymbol{x}^{(i)}, \hat{\boldsymbol{z}}^{(i)}), \quad i = 1, \ldots, M$ ▷ Compute discriminator predictions

$\rho_p^{(j)} \leftarrow D(\tilde{\boldsymbol{x}}^{(j)}, \boldsymbol{z}^{(j)}), \quad j = 1, \ldots, M$

$\mathcal{L}_d \leftarrow -\frac{1}{M}\sum_{i=1}^{M} \log(\rho_q^{(i)}) - \frac{1}{M}\sum_{j=1}^{M} log(1 - \rho_p^{(j)})$ ▷ Compute discriminator loss

$\mathcal{L}_g \leftarrow -\frac{1}{M}\sum_{i=1}^{M} \log(1 - \rho_q^{(i)}) - \frac{1}{M}\sum_{j=1}^{M} \log(\rho_p^{(j)})$ ▷ Compute generator loss

$\theta_d \leftarrow \theta_d - \nabla_{\theta_d}\mathcal{L}_d$ ▷ Gradient update on discriminator network

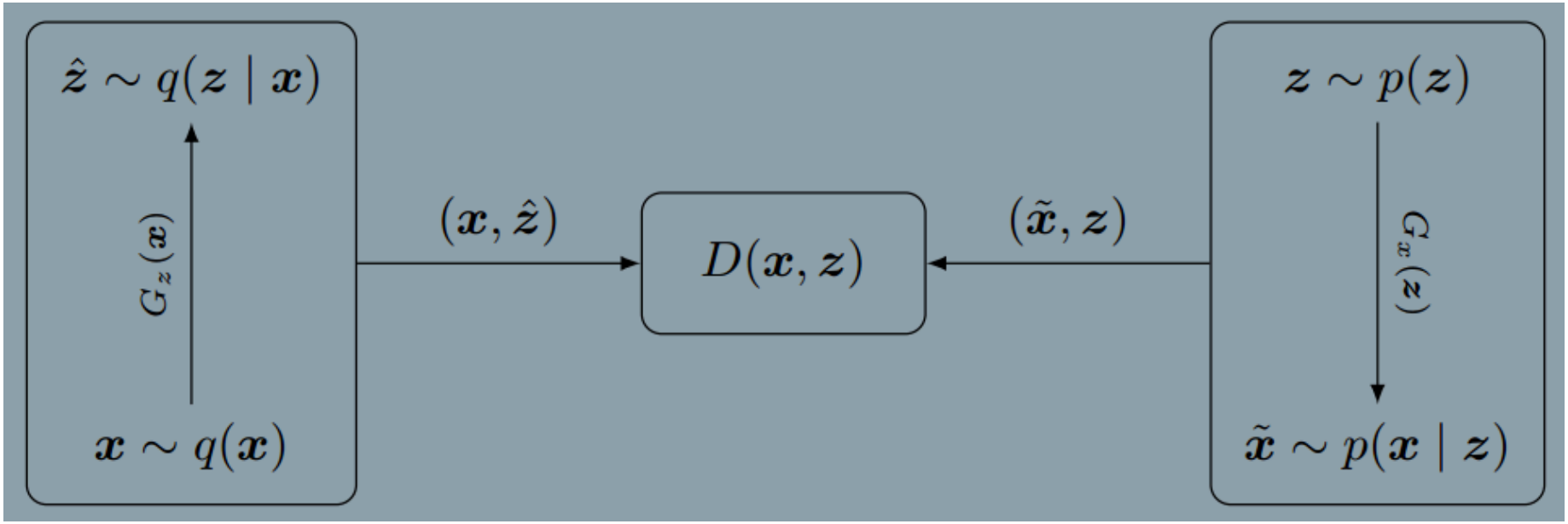
$\theta_g \leftarrow \theta_g - \nabla_{\theta_g}\mathcal{L}_g$ ▷ Gradient update on generator networks

until convergence

> An attractive property of adversarial approaches is that they do not require that the conditional densities can be computed;

> they only require that they can be sampled from in a way that allows gradient backpropagation.

> The discriminator is trained to distinguish between samples from the $encoder(x, \hat{z}) \sim q(x, z)$ and samples from the decoder $(\hat{x}, z) \sim p(x, z)$. The generator is trained to fool the discriminator, i.e., to generate x, z pairs from $q(x, z)$ or $p(x, z)$ that are indistinguishabel one from another.



training method

The generator is trained to maximize:

$$V'(D, G) = E_{q(x)}[log(1 - D(x, G_z(x)))] + E_{p(z)}[log(D(G_x(z), z))]$$

> However, gradient propagation into the encoder and decoder networks relies on the reparametrization trick, which means that ALI is not directly applicable to either applications with discrete data or to models with discrete latent variables.