10-708: Probabilistic Graphical Models, Spring 2020

12:Deep generative models: overview of the theoretical basis and connection

Lecturer: Eric P. Xing *Scribe: Shangbang Long Ankit Shrivastava, Prakhar Gupta*

1 Deep generative models

Deep generative models (DGMs) are probably the most popular research topic nowadays (CVPR, ICML, NeurIPS, etc.). This lecture is about a unifying theoretical perspective of DGMs.

DGMs can perform: style transfer/fusion, music/image generations, and etc..

DGMs are probabilistic distributions of a set of variables. *Deep* means having multiple layers of hidden variables.

1.1 Early forms of deep generative models

1.1.1 Sigmoid belief nets

One example is the Sigmoid Belief Networks in which lower layers connect to upper layers via sigmoid functions:

$$p(X_{n,k} = 1|\theta_{n,k}, X_{n-1}) = \sigma(\theta_{n,k}^T X_{n-1})$$

where X_n is a vector representing the hidden variables of the $n-$th layer, $\theta_{n,k}$ is a vector that represents how the $k-$th unit of layer n is connected to the lower layer.

These are authentic and genuine PGMs, based on the Bayesian rules.

1.1.2 Helmholtz machines

Helmholtz machines express a dual, alternative process that unifies inference and generative process:

$$X_n = G_\theta(X_{n-1})$$

and

$$X_{n-1} = F_\phi(X_n)$$

1.1.3 Predictability minimization

PM defines a training procedure. The encoding models represent the input data as latent variables. There is an extra predictability layer on top of the latent variables that learns to predict the latent variables. At the same time, PM learns with a constraint that minimizes the predictability of the latent variables.

1.1.4 Learning procedure

The training of such models follows an EM style framework, vias sampling and inference alternatively.

Another option is to optimize a variational lower bound:

$$log(p(x)) \geq E_{q_\phi(z|x)}[log(p_\theta(x,z))] - KL(q_\phi(z|x)||p(z)) := L(\theta,\phi;x)$$

and optimization target:

$$max_{\theta,\phi} L(\theta,\phi;x)$$

1.2 Resurgence of deep generative models

Variational autoencoders (VAEs) is the first modern, actively used deep generative models. VAEs consist of a generative model and an inference model that are trained in a variational way. Nearly at the same period, Generative adversarial networks (GANs) are proposed. GANs consist of a generator that transforms latent variables into data domain and a discriminator that learns to distinguish real data and generated (fake) data.

2 Theoretical basis of deep generative models

2.1 Recap: Variational Inference

Consider a generative model $p_\theta(x|z)$ and prior $p(z)$. The joint distribution is computed as: $p_\theta(x,z) = p_\theta(x|z)p(z)$. Assume variational distribution $q_\phi(z_x)$. Then the objective is to maximize lower bound for log likelihood:

$$\begin{aligned}
&\log p(x) \\
&= KL\left(q_\phi(\mathbf{z}|x)\|p_\theta(\mathbf{z}|x)\right) + \textstyle\int_z q_\phi(\mathbf{z}|x)\log\frac{p_\theta(x,z)}{q_\phi(z|x)} \\
&\geq \textstyle\int_z q_\phi(z|x)\log\frac{p_\theta(x,z)}{q_\phi(z|x)} \\
&:= L(\theta,\phi,x)
\end{aligned}$$

which is equivalently to minimize free energy:

$$F(\boldsymbol{\theta},\phi;x) = -\log p(x) + KL\left(q_\phi(z|x)\|p_\theta(z|x)\right)$$

During training, the following two EM steps are taken alternatively:

•$E-step:$ maximize $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ wrt. $\boldsymbol{\phi}$, with $\boldsymbol{\theta}$ fixed .

If closed form solutions exist, then $q_\phi^*(z|x) \propto \exp[\log p_\theta(x, z)]$.

•$M-step:$ maximize $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ wrt. $\boldsymbol{\theta}$, with $\boldsymbol{\phi}$ fixed

2.2 Wake sleep algorithm

Wake sleep algorithm maximizes data log-likelihood with two steps of loss relaxation:

2.2.1 *Wake* phase

Maximize the variational lower bound of log-likelihood, or equivalently, minimize the free energy: $F(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\right)$

In the wake phase, the algorithm gets samples from $q_\phi(z|x)$ through inference on hidden variables, and then use the samples as targets for updating the generative model $p_\theta(\mathbf{Z}|\boldsymbol{x})$. This phase corresponds to the variational M step, i.e.:

$$\max_{\boldsymbol{\theta}} \mathrm{E}_{\boldsymbol{q}_{\boldsymbol{\phi}}(\mathbf{z}|\boldsymbol{x})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}, \mathbf{z})\right]$$

2.2.2 *Sleep* phase

Minimize a different objective (reversed KLD) wrt) to ease the optimization: $F'(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = -\log p(\boldsymbol{x}) + KL\left(p_\theta(\boldsymbol{z}|\boldsymbol{x}) \| q_\phi(\boldsymbol{z}|\boldsymbol{x})\right)$.

The sleep phase corresponds to the E step, i.e.:

$$\max_{\phi} \mathrm{E}_{q_\phi(\mathbf{z}|x)}\left[\log p_\theta(x, z)\right]$$

However, one major difficulty is that the gradient for the distribution q_ϕ is difficult to compute. Instead, we use the log-derivative trick:

$$\nabla_\phi E_{q_\phi}\left[\log p_\theta\right] = \int \nabla_\phi q_\phi \log p_\theta = \int q_\phi \log p_\theta \nabla_\phi \log q_\phi = E_{q_\phi}\left[\log p_\theta \nabla_\phi \log q_\phi\right]$$

Then we can use Monte Carlo to estimate the gradients.

2.3 Variational autoencoders

Variational lower bound:

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x}) = \mathrm{E}_{q_\phi(\mathbf{Z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}, \mathbf{z})\right] - \mathrm{KL}\left(q_\phi(\mathbf{z}|\boldsymbol{x}) \| p(\mathbf{z})\right)$$

Similar to Wake-Sleep algorithm, the procedure optimizes $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{x})$ w.r.t. θ and ϕ alternatively. VAEs also use reparametrization trick to reduce variance.

Recall:

$$\nabla_\phi E_{q_\phi}\left[\log p_\theta\right] = E_{q_\phi}\left[\log p_\theta \nabla_\phi \log q_\phi\right]$$

The scale factor $\log p_\theta$ has high variance.

Instead of sampling with $\mathbf{z} \sim q_\phi(\mathbf{z}|\boldsymbol{x})$, we parametrize z as a noise added to a function, $\mathbf{z} = \mathbf{g}_\phi(\boldsymbol{\epsilon}, \boldsymbol{x}), \quad \boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$.

Then the estimated gradients then become: $\nabla_\phi \mathrm{E}_{q_\phi(\mathbf{z}|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}, \mathbf{z})\right] = \mathrm{E}_{\epsilon\sim p(\epsilon)}\left[\nabla_\phi \log p_\theta\left(\boldsymbol{x}, \mathbf{z}_\phi(\boldsymbol{\epsilon})\right)\right]$. Empirically, it has lower variance.

2.4 Variational autoencoders Algorithm

One of the Variational Autoencoder algorithm is a minibath Auto-encoding variational Bayes proposed by [1]

Algorithm 1: Minibatch version of the Auto-encoding VB (AEVB) algorithm.

Result: Write here the result
$(\theta, \phi) \leftarrow$ initialize parameters;
while *repeat until convergence of parameters* (θ, ϕ) **do**
  $\boldsymbol{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset) ;
  $\epsilon \leftarrow$ Random samples from noise distribution $p(\epsilon)$;
  $g \leftarrow \widetilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \boldsymbol{X}^M, \boldsymbol{\epsilon})$ (Gradient of minibatch estimator);
  $(\theta, \phi) \leftarrow$ Update parameters using gradients g (e.g. SGD or Adagrad)
end

3 Generative adversarial networks

Generative adversarial networks are another type of model complementary to variational autoencoders. It does not have any particular inference model but consist to two main components a Generative model and a Discriminator. The generative model map latent variable $\mathbf{z}$ coming from some prior i.e. $\mathbf{z} \sim p(\mathbf{z})$ to data space $\mathbf{x}$ using a transformation function G_θ, that is $\mathbf{x} = G_\theta(\mathbf{x})$. The discriminator component $D_\phi(\mathbf{x})$ outputs the probability that $\mathbf{x}$ came from data rather than the generator.

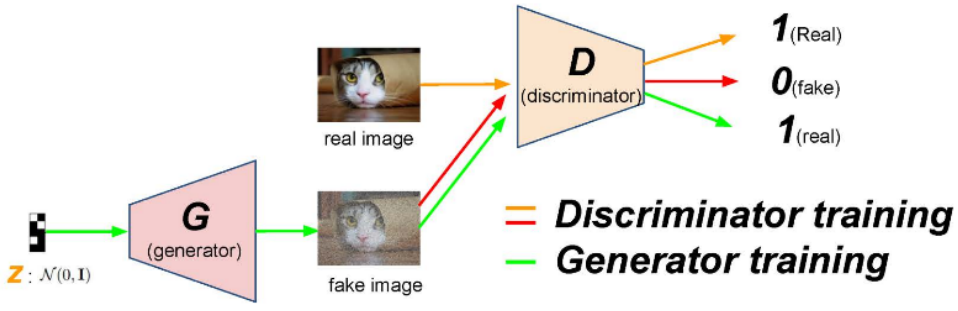

Figure 1: General architecture of GANS

Figure 1 shows a general GANS model. The model is trained by competing generator and discriminator against each other.

- It trains Discriminator to maximize the probability of correctly classifying real training examples and fake generated examples by

$$\max_D \mathcal{L}_D = \mathrm{E}_{\mathbf{x}\sim p_{data(\mathbf{x})}}[\log D(\mathbf{x})] + \mathrm{E}_{\mathbf{x}\sim G(\mathbf{z}),\mathbf{z}\sim p(\mathbf{z})}[\log 1 - D(\mathbf{x})]$$

- At the same time train Generator to beat the discriminator by

$$\min_G \mathcal{L}_G = \mathrm{E}_{\mathbf{x}\sim G(\mathbf{z}),\mathbf{z}\sim p(\mathbf{z})}[\log 1 - D(\mathbf{x})]$$

  Since the original loss suffers from vanishing gradient when Discriminator is too strong, in practice the Generator is trained by

$$\max_G \mathcal{L}_G = \mathrm{E}_{\mathbf{x}\sim G(\mathbf{z}),\mathbf{z}\sim p(\mathbf{z})}[\log D(\mathbf{x})]$$

The goal while training GANs is to achieve optimal state when generator and discriminator are competing among each other i.e.

- Generator produces the data using the distribution which is same as of the data distribution

$$p_G(\mathbf{x}) = p_{data}(\mathbf{x})$$

- And, discriminator assigning the probability randomly with probability of .5

$$D(\mathbf{x}) = \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_G(\mathbf{x})} = \frac{1}{2}$$

3.1 GANs objective in variational-EM format

Since the first GANs model has been proposed there has been multiple proposed GANs to improve desired results but the practice of designing of models are often compared to alchemy as it is difficult to explain why certain model is working. Hence, it is needed to view these models in different way. One of the ways to view GANs is using variational format and try to connect them with other deep generative models such as VAE and its variant

To understand GANs in terms of variational inference format, we first redefine the model such that the goal is to find implicit distribution over $\mathbf{x} \sim p_\theta(\mathbf{x}|y)$ where,

$$p_\theta(\mathbf{x}|y) = \begin{cases} p_{g_\theta}(\mathbf{x}) & y = 0 \quad \textbf{(distribution of generated images)} \\ p_{data}(\mathbf{x}) & y = 1 \quad \textbf{(distribution of data)} \end{cases}$$

Also, $\mathbf{x} \sim p_{g_\theta}(\mathbf{x}) \Leftrightarrow \mathbf{x} = G_\theta(\mathbf{z}), \mathbf{z} \sim p(\mathbf{z}|y = 0)$.

Now recall the conventional GANs formualtion as

$$\max_\phi \mathcal{L}_\phi = \mathrm{E}_{\mathbf{x}\sim p_{data(\mathbf{x})}}[\log D_\phi(\mathbf{x})] + \mathrm{E}_{\mathbf{x}=G_{\boldsymbol{\theta}(\mathbf{z})},\mathbf{z}\sim p(\mathbf{z}|y=0)}[\log 1 - D_\phi(\mathbf{x})]$$

$$\max_\theta \mathcal{L}_\theta = \mathrm{E}_{\mathbf{x}=G_{\boldsymbol{\theta}(\mathbf{z})},\mathbf{z}\sim p(\mathbf{z}|y=0)}[\log D_\phi(\mathbf{x})]$$

By rewriting formulation of GANs, it can be reviewed as

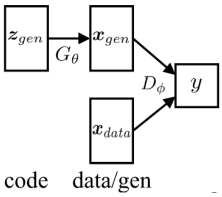

Figure 2: General architecture of GANS

- Introduce implicit distribution over $\mathbf{x} \sim p_\theta(\mathbf{x}|y)$, where

$$\mathbf{x} = G_{\boldsymbol{\theta}(\mathbf{z}),\mathbf{z}\sim p(\mathbf{z}|y)}$$

  We can call it as generative model since the data is generated based on hidden variable $\mathbf{z}$

- And Discriminator distribution $q_\phi(y|\mathbf{x})$ and $q^r_\phi(y|\mathbf{x}) = q_\phi(1-y|\mathbf{x})$ The discriminator model does not exist in variational inference literature, hence it should not be confused with inference model used in earlir vm

- the objective functions of GANs then becomes
  Learn parameters of discriminator model

$$\max_\phi \mathcal{L}_\phi = E_{p\theta(\mathbf{x}|y)p(y)}[\log q_\phi(y|\mathbf{x})]$$

  this kind of look like a posterior inference of hidden states because y is always observed as we need to determine whether it is true or false label. Now next task is to learn generative model parameters but with the loss function is reversed

$$\max_\theta \mathcal{L}_\theta = E_{p\theta(\mathbf{x}|y)p(y)}[\log q^r_\phi(y|\mathbf{x})]$$

  Hence the formulation look similar to wake sleep. This is further discussed n section 3.3 and section 3.4

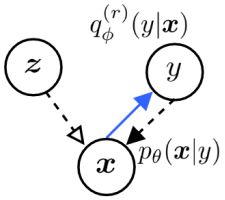

Figure 3: GANS viewed in form of Variational format

3.2 GANs: Minimizing KLD

Recall that in variational EM we minimize $-\log p(\mathbf{x}) + \mathrm{KL}(q_\phi)(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x}))$, that is, we minimize the KLD from the inference model to the posterior. We can similarly rewrite the objective of GAN in the form of minimizing KLD.

For each optimization step of $p_{\boldsymbol{\theta}(\mathbf{x}|y)}$ starting from an initial point $(\boldsymbol{\theta}_0, \boldsymbol{\phi}_0)$, let $p(y)$ be a uniform prior distribution, and

$$p_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}(\mathbf{x}) = E_{p(y)}[p_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}(\mathbf{x}|y)]$$
$$q^{\mathrm{r}}(\mathbf{x}|y) \propto q^{\mathrm{r}}(y|\mathbf{x}) p_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}(\mathbf{x})$$

The update rule for $\boldsymbol{\theta}$ is in Lemma 1:

Lemma 1:

$$\nabla_\theta(-E_{p_\theta(\mathbf{x}|y)p(y)}[\log q^{\mathrm{r}}_{\phi=\phi_0}(y|\mathbf{x})]) \mid_{\theta=\theta_0} =$$
$$\nabla_\theta(E_{p(y)}[\mathrm{KL}(p_\theta(\mathbf{x}|y) \parallel q^{\mathrm{r}}(\mathbf{x}|y))] - \mathrm{JSD}(p_\theta(\mathbf{x}|y=0) \parallel p_\theta(\mathbf{x}|y=1))) \mid_{\theta=\theta_0} .$$

Here the generative model $p_\theta(\mathbf{x}|y)$ is equivalent to the variational distribution for the posterior $q^{\mathrm{r}}(\mathbf{x}|y)$. $p_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}(\mathbf{x})$ is the prior.

Now, minimizing the KLD drives the generator $p_{g_{\boldsymbol{\theta}}(\mathbf{x})}$ to the true data distribution $p_{\mathrm{data}}(\mathbf{x})$. For a uniform y (being equally real or generated), by definition,

$$p_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}(\mathbf{x}) = E_{p(y)}[p_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}(\mathbf{x}|y)] = \frac{p_{g_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}}(\mathbf{x}) + p_{\mathrm{data}}(\mathbf{x})}{2},$$

and thus we could break the KLD into two terms:

$$\mathrm{KL}(p_\theta(\mathbf{x}|y=1) \parallel q^{\mathrm{r}}(\mathbf{x}|y=1)) = \mathrm{KL}(p_{\mathrm{data}}(\mathbf{x}) \parallel q^{\mathrm{r}}(\mathbf{x}|y=1)) = \text{const.},$$

and

$$\mathrm{KL}(p_\theta(\mathbf{x}|y=0) \parallel q^{\mathrm{r}}(\mathbf{x}|y=0)) = \mathrm{KL}(p_{g_\theta}(\mathbf{x}) \parallel q^{\mathrm{r}}(\mathbf{x}|y=0)),$$

where

$$q^{\mathrm{r}}(\mathbf{x}|y=0) \propto q^{\mathrm{r}}(y=0|\mathbf{x}) p_{\boldsymbol{\theta}=\boldsymbol{\theta}_0}(\mathbf{x})$$

This can be seen as a mixture of $p_{g_{\boldsymbol{\theta}}(\mathbf{x})}$ and $p_{\mathrm{data}}(\mathbf{x})$ weighted by $q^{\mathrm{r}}(y=0|\mathbf{x})$. During the update we drive the generator $p_{g_{\boldsymbol{\theta}}(\mathbf{x})}$ towards $q^{\mathrm{r}}(\mathbf{x}|y=0)$ by minimizing the KLD, and drive it towards the true distribution $p_{\mathrm{data}}(\mathbf{x})$, as shown in Figure 4.

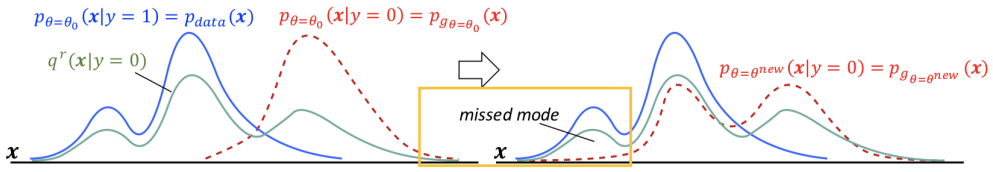

Figure 4: Minimizing KLD in GAN

The KL formulation tends to be large in the regions of x space where $q^{\mathrm{r}}(\mathbf{x}|y=0)$ is small unless $p_{g_{\boldsymbol{\theta}}(\mathbf{x})}$ is also small. Thus, $p_{g_{\boldsymbol{\theta}}(\mathbf{x})}$ tends to be small where $q^{\mathrm{r}}(\mathbf{x}|y=0)$ is small to avoid the penalization. Therefore, this KLD formulation allows GAN to recover the major modes, but also leads GAN to miss some minor modes of $p_{\mathrm{data}}(\mathbf{x})$, where $p_{g_{\boldsymbol{\theta}}(\mathbf{x})}$ and $q^{\mathrm{r}}(\mathbf{x}|y=0)$ are small and already give a small KLD. This phenomenon where some modes are missed but other captured well is what leads GAN to create sharp images.

	GANs (InfoGAN)	VAEs
Generative distribution	$p_\theta(\boldsymbol{x}\|y) = \begin{cases} p_{g_\theta}(\boldsymbol{x}) & y=0 \\ p_{data}(\boldsymbol{x}) & y=1. \end{cases}$	$p_\theta(\boldsymbol{x}\|\boldsymbol{z},y) = \begin{cases} p_\theta(\boldsymbol{x}\|\boldsymbol{z}) & y=0 \\ p_{data}(\boldsymbol{x}) & y=1. \end{cases}$
Discriminator distribution	$q_\phi(y\|\boldsymbol{x})$	$q_*(y\|\boldsymbol{x})$, perfect, degenerated
$\boldsymbol{z}$-inference model	$q_\eta(\boldsymbol{z}\|\boldsymbol{x},y)$ of InfoGAN	$q_\eta(\boldsymbol{z}\|\boldsymbol{x},y)$
KLD to minimize	$\min_\theta \mathrm{KL}\left(p_\theta(\boldsymbol{x}\|y) \,\|\| \, q^r(\boldsymbol{x}\|\boldsymbol{z},y)\right)$ $\sim \min_\theta \mathrm{KL}(P_\theta \,\|\| \, Q)$	$\min_\theta \mathrm{KL}\left(q_\eta(\boldsymbol{z}\|\boldsymbol{x},y)q_*^r(y\|\boldsymbol{x}) \,\|\| \, p_\theta(\boldsymbol{z},y\|\boldsymbol{x})\right)$ $\sim \min_\theta \mathrm{KL}(Q \,\|\| \, P_\theta)$

Figure 5: GAN vs VAE

3.3 GAN vs VAE

Lets recap the VAE objective

$$\max_{\boldsymbol{\theta},\boldsymbol{\eta}} \mathcal{L}^{\text{vae}}_{\theta,\eta} = E_{p_{\text{data}}(\boldsymbol{x})}\left[E_{\tilde{q}_\eta(\boldsymbol{z}|\boldsymbol{x})}\left[\log \tilde{p}_\theta(\boldsymbol{x}|\boldsymbol{z})\right] - \mathrm{KL}\left(\tilde{q}_\eta(\boldsymbol{z}|\boldsymbol{x})\|\tilde{p}(\boldsymbol{z})\right)\right]$$

We now assume a perfect discriminator $q_\star(y|\mathbf{x})$ telling whether $\mathbf{x}$ is real or generated, and $q^{\text{r}}_\star(y|\mathbf{x}) = q_\star(1 - y|\mathbf{x})$. Notice that q_* is degenerate since we are always generating fake $\mathbf{x}$. We can write $\mathcal{L}^{\text{vae}}_{\boldsymbol{\theta},\boldsymbol{\eta}}$ as

Lemma 2:

$$\begin{aligned}\mathcal{L}^{\text{vae}}_{\theta,\eta} &= 2 \cdot E_{p_{\theta_0}(\boldsymbol{x})}\left[E_{q_\eta(\boldsymbol{z}|\boldsymbol{x},y)q^r_*(y|\boldsymbol{x})}\left[\log p_\theta(\boldsymbol{x}|\boldsymbol{z},y)\right] - \mathrm{KL}\left(q_\eta(\boldsymbol{z}|\boldsymbol{x},y)q^r_*(y|\boldsymbol{x})\|p(\boldsymbol{z}|y)p(y)\right)\right] \\ &= 2 \cdot E_{p_{\theta_0}(\boldsymbol{x})}\left[-KL\left(q_\eta(\boldsymbol{z}|\boldsymbol{x},y)q^r_*(y|\boldsymbol{x})\|p_\theta(\boldsymbol{z},y|\boldsymbol{x})\right)\right]\end{aligned}$$

where the posterior is given as

$$p_\theta(\mathbf{z}|\mathbf{x},y) \propto p_\theta(\mathbf{x}|\mathbf{z},y)p(\mathbf{z}|y)p(y)$$

and is determined by the generative model $p_{\boldsymbol{\theta}(\mathbf{z}|\mathbf{x},y)}$ and the other two terms are fixed priors. In this way, VAE has the generative model on the right side of KLD, different to GAN.

Therefore, GAN while provides a "sharp" distribution of covering major modes while missing minor modes of the true distribution p_{data}, VAE provides a blurred or approximate distribution to cover all the modes while less sharp approximation of modes where p_{data} is small.

The Figure 5 summarizes the comparison between GAN and VAE's new formulation.

3.4 Linking GAN and VAE to Wake-sleep

Lets look back at the wake-sleep algorithm:

$$\text{Wake} : \max_{\theta} E_{q_\lambda(\mathbf{h}|\mathbf{x})p_{\text{data}}(\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{h})],$$
$$\text{Sleep} : \max_{\theta} E_{p_\theta(\mathbf{x}|\mathbf{h})p(\mathbf{h})}[\log q_\lambda(\mathbf{h}|\mathbf{x})].$$

We now discuss how VAE and GAN relate to the wake-sleep procedure.

VAE only deals with the wake phase and also learns the inference model $\boldsymbol{\eta}$, That is, the KLD term in the original variational free energy:

$$\max_{\theta,\eta} \mathcal{L}^{\text{VAE}}_{\theta,\eta} = E_{q_\eta(\mathbf{z}|\mathbf{x})p_{\text{data}}(\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] - E_{p_{\text{data}}(\mathbf{x})}[\text{KL}(q_\eta(\mathbf{z}|\mathbf{x}) \parallel p(\mathbf{z}))]$$

It does not involve the sleep-phase objective.

GAN only deals with the sleep phase and extends it by learning the generative parameter $\boldsymbol{\theta}$:

$$\max_{\phi} \mathcal{L}_\phi = E_{p_\theta(\mathbf{x}|y)p(y)}[\log q_\phi(y|\mathbf{x})],$$
$$\max_{\phi} \mathcal{L}_\theta = E_{p_\theta(\mathbf{x}|y)p(y)}[\log q^{\text{r}}_\phi(y|\mathbf{x})].$$

It does not involve the wake-phase objective.

3.5 Conclusion

- GANs and VAEs are essentially minimizing KLD in opposite directions and extend two phases of classic wake sleep algorithm, respectively
- This lecture discussed a general formulation useful for analyzing a broad class of existing DGM modelsm and can inspire new models and algorithms.

References

[1] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.