Theoretic Fundamentals of Machine and Deep Learning

Variational Inference, Autoencoder, Variational Autoencoder, Conditional Variational Autoencoder

Aleksandr Petiushko

Lomonosov MSU, Faculty of Mechanics and Mathematics
MIPT, RAIRI
Nuro, Autonomy Interaction Research

Winter-Spring, 2023

AP

Content

AP

Variety of generative models

Main representatives of generative models:

- Mixture models (e.g., GMM)
- Hidden Markov Models (HMMs)
- Bayesian Network (and Autoregressive Models)
- Flow-based Models
- Energy-based Models (EBMs)
- Diffusion Models
- Variational Autoencoder (VAE)
- Generative Adversarial Networks (GANs)

AP

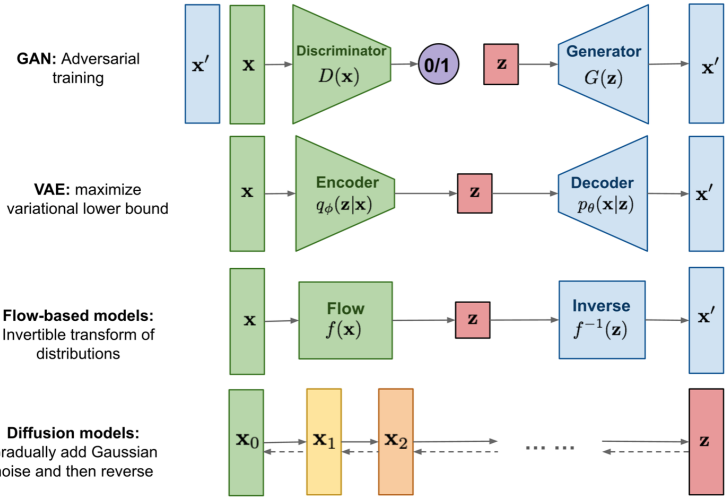
Variety of generative models

Main representatives of generative models:

- Mixture models (e.g., GMM)
- Hidden Markov Models (HMMs)
- Bayesian Network (and Autoregressive Models)
- Flow-based Models
- Energy-based Models (EBMs)
- Diffusion Models
- **Variational Autoencoder (VAE)**
- Generative Adversarial Networks (GANs)

AP

Variety of generative models: illustration[1]



[1] Lil'log blog

Main formulas

- Conditional probability: $p(x|y) = \frac{p(x,y)}{p(y)}$
- Law of total probability: $p(x) = \int_y p(x,y)dy$
- Expectation through integral: $E_{p(x)}f(x) = \int_x p(x)f(x)dx$
- Bayes' theorem: $p(x|y) = \frac{p(y|x)p(x)}{p(y)}$

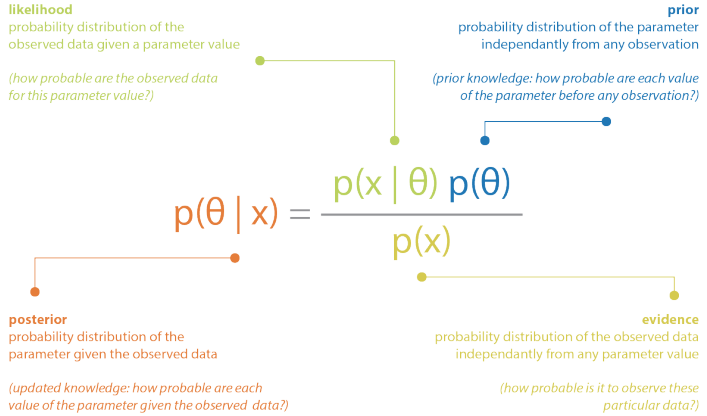
AP

Bayesian inference: preliminaries

- Observed input data: x
- Unobserved parameters of the model z : $z \sim p(z)$ (prior distribution)
- A parametric model = joint distribution of latent variable z and observed x: $p(x, z)$
- Likelihood – distribution of the observed data conditioned on parameters $p(x|z)$
- Marginal likelihood: $p(x) = \int p(x, z)dz = \int p(x|z)p(z)dz$

AP

Bayesian inference terms illustration[2]



[2] Joseph Rocca's blog

Bayesian inference

- Inference = **posterior** probability (Bayes' rule):

$$p(z|x) = \frac{p(x,z)}{p(x)} = \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz}$$

- **Posterior predictive distribution** for the new input x': $p(x'|x) = \int p(x'|z)p(z|x)dz$
- **Prior predictive distribution** for the new input x': $p(x') = \int p(x'|z)p(z)dz$

AP

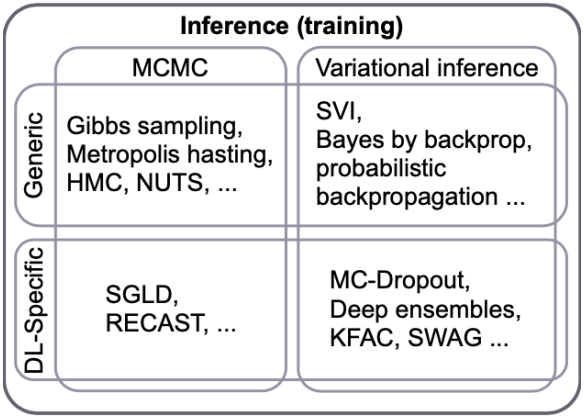
Posterior inference

- Inference = **posterior** probability: $p(z|x) = \frac{p(x,z)}{p(x)}$
- Due to unknown $p(x) = \int p(x|z)p(z)dz$, $p(z|x)$ is done via **approximate posterior inference**
  - We're going to work with posteriors known up to a normalization constant $p(x)$
- The most widely used ways to do it:
  - By Variational Inference
    - Bayes by Backprop (VAE)
    - Deep Ensembles / Monte Carlo Dropout
  - By Markov Chain Monte Carlo
    - Samplers (Gibbs, Metropolis)
    - Stochastic Gradient Langevin Dynamics

AP

Posterior inference

Diagram[3] shows important means of Bayesian Inference:

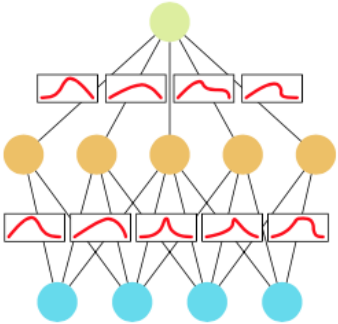


[3] Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., and Bennamoun, M. Hands-on Bayesian neural networks — A tutorial for deep learning users. 2020.

Bayesian Neural Nets (BNN)

BNN

BNN = a *stochastic* neural network trained using Bayesian inference.
Stochastic neural network = NN with a probability distribution over weights.



AP

BNN: design choice

To design a BNN, one need to choose:

- A prior over weights $\theta \sim p(\theta)$
- Output distribution $p(y|x, \theta)$ (usually modeled as NN $y = \Phi_\theta(x)$ with finite $|Y| < \infty, y \in Y$ in case of classifiers)

Let's denote $D = (D_x, D_y)$ as the training set. In this case the posterior

$$p(\theta|D) = \frac{p(D_y|D_x, \theta)p(\theta)}{\int_\Theta p(D_y|D_x, \theta')p(\theta')d\theta'}$$

AP

BNN: inference

- BNN inference (posterior predictive distribution):

$$p(y|x, D) = \int_{\Theta} p(y|x, \theta')p(\theta'|D)d\theta'$$

- Is done through Monte Carlo and the output is $\frac{1}{|\Theta|}\sum_{\theta_i \in \Theta} \Phi_{\theta_i}(x)$

Algorithm 1 Inference procedure for a BNN.

Define $p(\boldsymbol{\theta}|D) = \dfrac{p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\theta}} p(D_{\boldsymbol{y}}|D_{\boldsymbol{x}}, \boldsymbol{\theta'})p(\boldsymbol{\theta'})d\boldsymbol{\theta'}}$;
for $i = 0$ **to** N **do**
  Draw $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|D)$;
  $\boldsymbol{y}_i = \Phi_{\boldsymbol{\theta}_i}(\boldsymbol{x})$;
end for
return $Y = \{\boldsymbol{y}_i | i \in [0, N)\}$, $\Theta = \{\boldsymbol{\theta}_i | i \in [0, N)\}$;

AP

BNN: online learning

BNN provide a natural way of **online learning**: *previous posteriors* can be *recycled* as *priors* when new data become available to avoid the so-called problem of catastrophic forgetting

Algorithm 3 Online learning loop with a BNN.

Define $p(\boldsymbol{\theta}) = p(\boldsymbol{\theta})_0$;
while true do
  Define $p(\boldsymbol{\theta}|D_i) = \frac{p(D_{\boldsymbol{y},i}|D_{\boldsymbol{x},i}, \boldsymbol{\theta})p(\boldsymbol{\theta})_i}{\int_{\boldsymbol{\theta}} .p(D_{\boldsymbol{y},i}|D_{\boldsymbol{x},i}, \boldsymbol{\theta'})p(\boldsymbol{\theta'})_i d\boldsymbol{\theta'}}$;
  Define $p(\boldsymbol{\theta})_{i+1} = p(\boldsymbol{\theta}|D_i)$;
end while

AP

Deep ensembles

One of the Deep Learning (DL)-specific methods to simplify working with posterior $p(\theta|D)$ is **Deep ensembles**[4]

- Main idea is to train M neural nets $\Phi_{\theta_i}(x), i = 1 \dots M$
  - Different order of training samples, different initializations of weights
- When making a prediction, just average the outcomes: $\hat{y} = \frac{1}{M} \sum_{i=1}^{M} \Phi_{\theta_i}(x)$
- Can be understood as posterior approximation by distribution parameterized as multiple Dirac deltas, where $\sum_{\theta_i \in \phi} \alpha_{\theta_i} = 1, \alpha_{\theta_i} \geq 0$:

$$q_\phi(\theta) = \sum_{\theta_i \in \phi} \alpha_{\theta_i} \delta_{\theta_i}(\theta)$$

[4] Lakshminarayanan, B., Pritzel, A., and Blundell, C. "Simple and scalable predictive uncertainty estimation using deep ensembles." 2016

MC-dropout

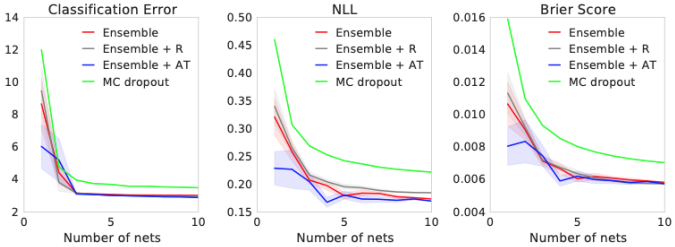
Another (DL)-specific methods to simplify working with posterior $p(\theta|D)$ is **Monte Carlo (MC)-dropout**[5]

- Suppose that layer i of MLP (for other types of NN the approach is quite similar) is a matrix $W_i, i = 1 \dots L$, of size $K_i \times K_{i-1}$
- In usual NN training dropout is applied only during training
- For MC-dropout it is applied during inference for distribution $q(\theta)$:
  - $z_{i,j} \sim Bernoulli(p_i), i = 1 \dots L, j = 1 \dots K_{i-1}$
  - $z_{i,j} = 0$ corresponds to unit j in layer $i-1$ being dropped out as an input to layer i
  - $\Theta_i = diag([z_{i,j}]_{j=1}^{K_i})$, $\theta = \{\Theta_i\}_{i=1}^{L}$
- Then we sample $\theta \sim q(\theta)$ and estimate posterior through Monte Carlo

[5] Gal, Y., and Ghahramani, Z. "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." 2015

Deep ensembles vs MC-dropout

MC-dropout can be understood as Deep ensembles via single NN. But even having only one NN (this is pro) the performance is worse (con):



(b) SVHN using VGG-style convnet

AP

Variational Inference (VI)

VI

VI = a method of approximating a complex distribution with some family of parameterized distributions.

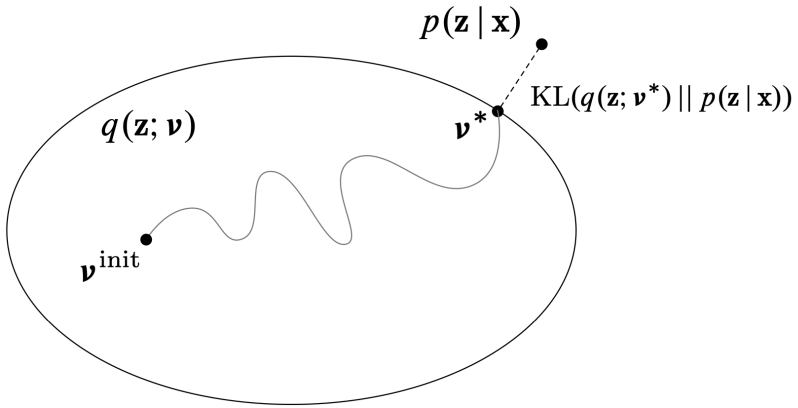
- VI treats posterior inference as optimization problem
- For this purpose a family of parameterized (by ϕ) distributions $q_\phi(z)$ is introduced
  - No any guarantee that $q_\phi(z)$ can cover (or even be very close) to the real posterior
  - Usually this family is simple Gaussian $q_\phi = N(\mu_\phi, \sigma^2_\phi)$
- Optimization objective:

$$D_{KL}(q_\phi(z)||p(z|x)) \to \min_\phi$$

AP

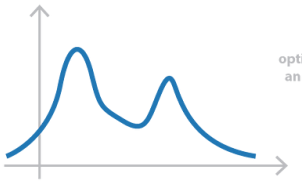
VI scheme

- Optimization process scheme: can never lead to a perfect solution[6]

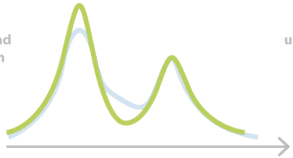


[6] David Blei, Rajesh Ranganath, Shakir Mohamed. Variational Inference: Foundations and Modern Methods

VI illustration[7]



optimisation to find an approximation



use of the approximate distribution

$$\int x.\pi(x)\,dx \approx \int x.f(x)\,dx$$

Unnormalised distribution (π)
whose normalisation factor computation is intractable

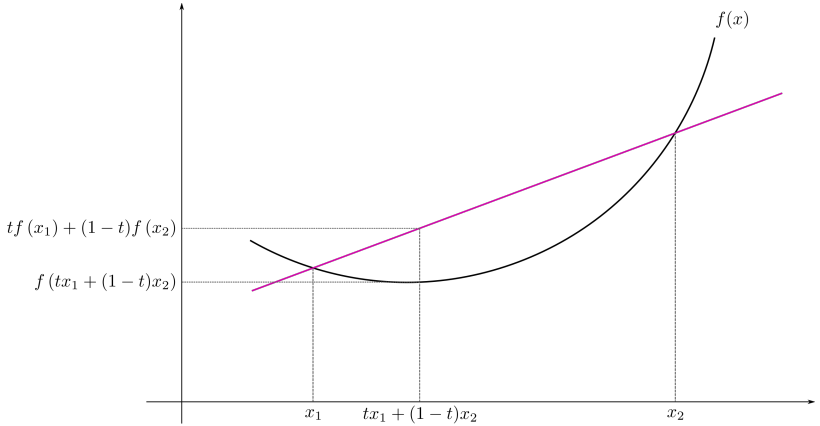
Best approximation (f)
among a parametrised family, obtained by optimising over the family parameters and without proceeding to the normalisation

Approximate computations
obtained by replacing the exact distribution by the approximate one

[7] Joseph Rocca's blog

Jensen's inequality

- Jensen's inequality for a convex function f can be written as a $f(E[x]) \leq E[f(x)]$ [8]
- For a concave function (e.g., log) we have the opposite: $f(E[x]) \geq E[f(x)]$



[8] Wiki: Jensen's inequality

Evidence Lower BOund (ELBO)

- Let's recall: $p(x)$ is called the **evidence**
- Then using the marginalization property and expectation through integral notation:

$$\log p(x) = \log \int_z p(x, z) = \log \int_z p(x, z) \frac{q_\phi(z)}{q_\phi(z)} = \log E_{q_\phi} \frac{p(x, z)}{q_\phi(z)}$$

- And using the Jensen's inequality for concave function:

$$\log p(x) \geq E_{q_\phi} \log \frac{p(x, z)}{q_\phi(z)}$$

- $E_{q_\phi} \log \frac{p(x,z)}{q_\phi(z)}$ is called Evidence Lower BOund (**ELBO**), as it is the lower bound for the evidence $\log p(x)$

AP

ELBO and KL-divergence

- Conditional probability property: $p(z|x) = \frac{p(x,z)}{p(x)}$
- Let's work on $D_{KL}(q_\phi(z)||p(z|x))$:

$$D_{KL}(q_\phi(z)||p(z|x)) = E_{q_\phi} \log \frac{q_\phi(z)}{p(z|x)} = E_{q_\phi} \log \frac{q_\phi(z)p(x)}{p(x,z)} = E_{q_\phi} \log \frac{q_\phi(z)}{p(x,z)} + E_{q_\phi} \log p(x)$$

- Replacing $E_{q_\phi} \log p(x) = \log p(x)$ and $E_{q_\phi} \log \frac{q_\phi(z)}{p(x,z)} = -E_{q_\phi} \log \frac{p(x,z)}{q_\phi(z)}$, we get:

$$D_{KL}(q_\phi(z)||p(z|x)) = \log p(x) - E_{q_\phi} \log \frac{p(x,z)}{q_\phi(z)} \tag{1}$$

- $q_\phi(z) \to p(z|x) \Rightarrow D_{KL}(q_\phi(z)||p(z|x)) \to 0 \Rightarrow E_{q_\phi} \log \frac{p(x,z)}{q_\phi(z)} \to \max_\phi$
- In other words, we need to **maximize** ELBO

AP

ELBO and normalization

- ELBO is equivalent to: $E_{q_\phi} \log \frac{p(x,z)}{q_\phi(z)} = E_{q_\phi} \log \frac{p(x|z)p(z)}{q_\phi(z)}$
- Suppose the unnormalized distribution $\hat{p}(x|z)$ (usually implemented as a Neural Net) is one that $p(x|z) = \frac{\hat{p}(x|z)}{C}$, where C is the normalization constant
- Then the ELBO becomes:

$$E_{q_\phi} \log \frac{p(x|z)p(z)}{q_\phi(z)} = E_{q_\phi} \log \frac{\hat{p}(x|z)p(z)}{q_\phi(z)} - E_{q_\phi} \log C = E_{q_\phi} \log \frac{\hat{p}(x|z)p(z)}{q_\phi(z)} - \log C$$

- And we can still work on maximization of ELBO using the unnormalized $\hat{p}(x|z)$ as $\log C$ being constant is not changing the optimization target

AP

Another optimization form

- Let's work on Eq. 1 in a different way: $0 \leq D_{KL}(q_\phi(z)||p(z|x)) =$

$$= \log p(x) - E_{q_\phi(z)} \log \frac{p(x,z)}{q_\phi(z)} = \log p(x) + E_{q_\phi(z)} \log \frac{q_\phi(z)}{p(x|z)p(z)} =$$

$$= \log p(x) + D_{KL}(q_\phi(z)||p(z)) - E_{q_\phi(z)} \log p(x|z) \Rightarrow$$

$$\log p(x) \geq E_{q_\phi(z)} p(x|z) - D_{KL}(q_\phi(z)||p(z))$$

- Maximization of ELBO $\Leftrightarrow$

$$L(x, z; \phi) = D_{KL}(q_\phi(z)||p(z)) - E_{q_\phi(z)} \log p(x|z) \rightarrow \min_\phi \tag{2}$$

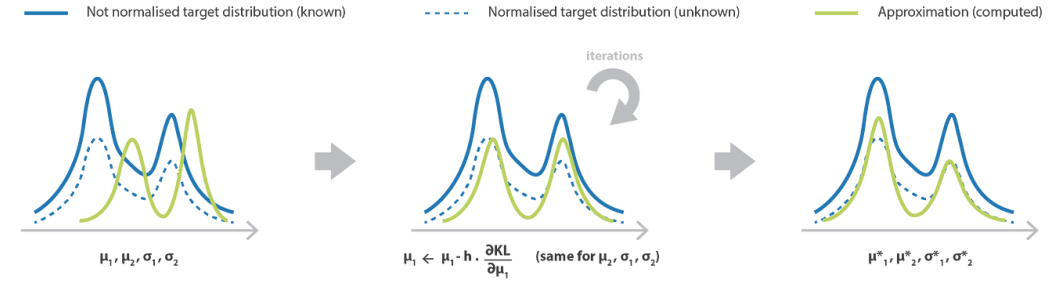
AP

BNN training: Bayes by backprop[9]

- Now we have everything to train the whole BNN
- Let's treat z as a vector of all NN weights
- To train efficiently, we need to apply the reparameterization trick: $\phi = (\mu, \sigma)$, $z = \mu + \sigma \cdot \epsilon$, where $\epsilon \sim N(0, 1) \Rightarrow z \sim_\phi N(\mu, \sigma^2)$
- Prior $z \sim_p N(0, 1)$, and point-wise NN output is $p(x|z)$
- Eq. 2 can be rewritten as $E_{q_\phi(z)}[\log q_\phi(z) - \log p(z) - \log p(x|z)]$ Let $f(z, \phi) = \log q_\phi(z) - \log p(z)p(x|z)$
- Calculate gradient w.r.t. (μ, σ): $\nabla_\mu f(z, \phi) = \frac{\partial f(z,\phi)}{\partial z}\frac{\partial z}{\partial \mu} + \frac{\partial f(z,\phi)}{\partial \mu}$ (and the same for σ)
- And finally, apply SGD! $\phi = \phi - \eta \nabla_\phi f(z)$

[9] Blundell, Charles, et al. "Weight uncertainty in neural network." 2015.

VI optimization process illustration[10]



Initialisation of the family parameters and computation of KL divergence between corresponding approximation and raw target

Iteratively compute derivative of KL divergence with respect to parameters to update and make a step in the opposite direction for these parameters

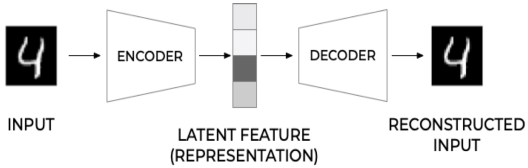
The best approximation among the chosen family is then obtained (up to the efficiency of the optimisation process and local minima)

[10] Joseph Rocca's blog

Auto Encoder (AE)

- Suppose that the data dimensionality is large: $x \in \mathbb{R}^N$
- To extract the lower-dimensional semantic representation of the data the so called **Auto Encoders** (**AE**) are used[11]
- For this purpose, two neural nets – **Encoder** $f : \mathbb{R}^N \to \mathbb{R}^K$, where usually $K \ll N$, and **Decoder**[12] $g : \mathbb{R}^K \to \mathbb{R}^N$ – are trained to minimize the Reconstruction Error:

$$\min_{f,g} \Delta(x, g \circ f(x)) = \min_{f,g} \Delta(x, g(f(x)))$$

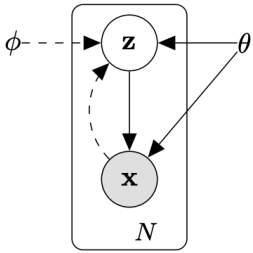


[11] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams. Learning internal representations by error propagation. 1985.

[12] Michelucci, Umberto. "An Introduction to Autoencoders." 2022.

Variational Auto Encoder (VAE)

- Suppose that the we would like to sample from the real data distribution $p(x), x \in \mathbb{R}^N$
- We could do it from the lower dimension $K \ll N$ with the help of the Decoder g
- But we need to know the distribution of the latent $z \in \mathbb{R}^K$ to apply $g(z)$
- It can be done by **Variational Auto Encoder (VAE)**[13]
  - Using the generative model $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$, we would like to approximate the posterior $p_\theta(z|x)$ by $q_\phi(z|x)$, as working solely on unconditional $q_\phi(z)$ is very hard



[13] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." 2013.

VAE objective

- Let's work on $q_\phi(z|x)$ instead of $q_\phi(z)$ in Eq. 2
- And use $p_\theta(z)$, $p_\theta(z|x)$ instead of $p(z)$, $p(z|x)$:

$$\log p(x) \geq E_{q_\phi(z|x)} p_\theta(x|z) - D_{KL}(q_\phi(z|x)||p_\theta(z))$$

- Maximization of ELBO $\Leftrightarrow$

$$L(x, z; \phi, \theta) = D_{KL}(q_\phi(z|x)||p_\theta(z)) - E_{q_\phi(z|x)} \log p_\theta(x|z) \to \min_{\phi,\theta} \tag{3}$$

AP

VAE loss

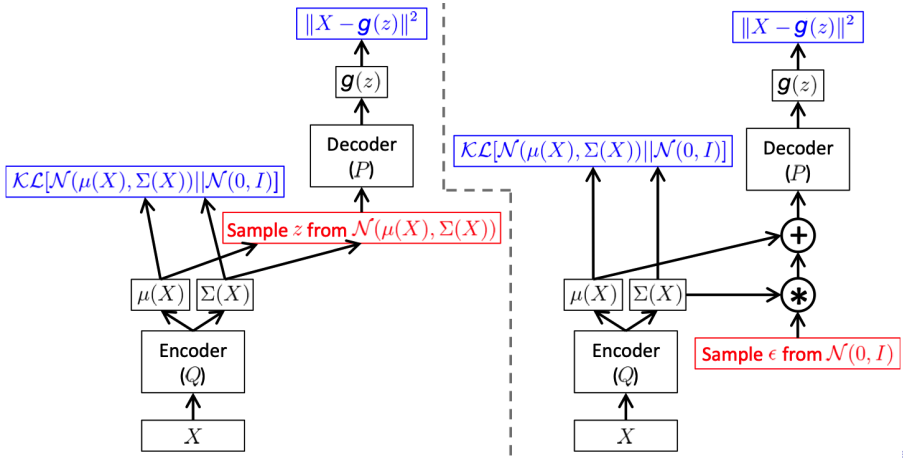
- Common design choices:
  - $p_\theta(z) = N(0, 1)$
  - $q_\phi(z|x) = N(\mu_\phi(x), \sigma^2_\phi(x))$
  - $p_\theta(x|z) = N(g(z), c^2 I)$
- Then
  - $E_{q_\phi(z|x)} \log p_\theta(x|z) = -E_{q_\phi(z|x)} \frac{(x-g(z))^2}{2c^2}$ (upto some constant)
  - $D_{KL}(q_\phi(z|x)||p_\theta(z)) = \frac{1}{2} \sum_{j=1}^{K} \left(\mu_{j,\phi}(x)^2 + \sigma^2_{j,\phi}(x) - 1 - \log \sigma^2_{j,\phi}(x) \right)$
- And the final loss $L(x, z; \phi, \theta) = D_{KL}(q_\phi(z|x)||p_\theta(z)) - E_{q_\phi(z|x)} \log p_\theta(x|z) =$

$$= \frac{1}{2} \sum_{j=1}^{K} \left(\mu_{j,\phi}(x)^2 + \sigma^2_{j,\phi}(x) - 1 - \log \sigma^2_{j,\phi}(x) \right) + E_{q_\phi(z|x)} \frac{(x - g(z))^2}{2c^2}$$

AP

VAE as a Neural Net — training

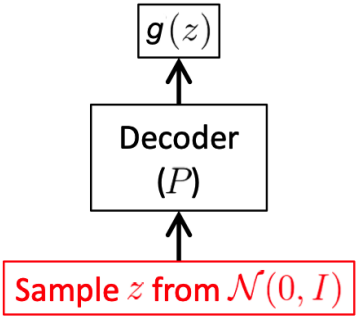
When implementing VAE as Neural Net, we need to apply Reparametrization Trick in order to have backpropagation of gradients through Decoder **and** Encoder (left – w/o, right – w/ this trick; no backprop; loss):



AP

VAE as a Neural Net — inference

For inference[14], we just sample $z \sim N(0, 1)$ and apply the learned Decoder $g(z)$:



[14] Doersch, C. "Tutorial on variational autoencoders." 2016

Conditional VAE (cVAE)

- With VAE we cannot guide the sampling process: if data has multiple classes, we don't know in advance which class we are sampling with prior $z \sim p_\theta(z)$
- To add this ability, we need to sample based on some guidance signal (e.g., class label) y
- This leads to so called **Conditional** VAE (**cVAE**)[15]
- ELBO for cVAE can be done by conditioning all the distribution terms inside $L(x, z; \phi, \theta)$ on y
  - Usually $p_\varphi(x|z, y)$ is modeled by another Neural Net with parameters φ
- Final objective is obtained in the similar manner as Eq. 3:

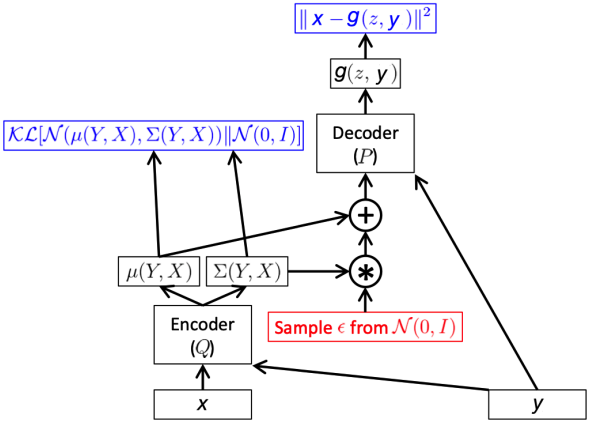
$$L(x, y, z; \phi, \varphi, \theta) = D_{KL}(q_\phi(z|x, y)||p_\theta(z|y)) - E_{q_\phi(z|x,y)} \log p_\varphi(x|z, y) \to \min_{\phi,\varphi,\theta}$$

[15] Sohn, K., Lee, H., and Yan, X. "Learning structured output representation using deep conditional generative models." 2015

cVAE — training

Common design choices:

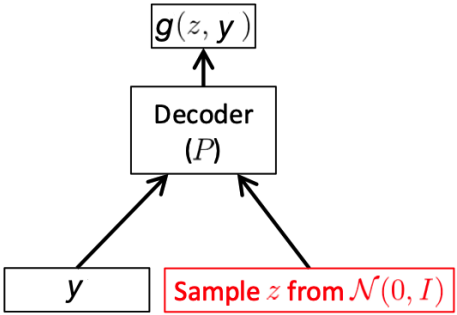
- Sometimes (not always!) conditioning on y for sampling of z in prior is omitted: $p_\theta(z|y) = N(0, 1)$
- $q_\phi(z|x, y) = N(\mu_\phi(x, y), \sigma^2_\phi(x, y))$
- $p_\varphi(x|z, y) = N(g(z, y), c^2 I)$



AP

cVAE — inference

For inference, we just sample $z \sim N(0, 1)$ and apply the learned Decoder $g(z, y)$:



AP

Takeaway notes

- Bayesian Neural Nets: weights are distributions
- Variational Inference is just an approximating of one distribution by another parameterized distribution
- Posterior usually contains intractable constant $\Rightarrow$ VI to the rescue
- VI is sample-efficient during inference because we sample from the converged distribution (that we learned)
- VI can provide the low-variance (see above item) but probably highly biased solution (because it is hard to provide a good family of distributions to cover the original one)
- ELBO is a good bound because there is no need in normalization
- VAE is a classical example of VI
- cVAE (as well as cGAN) is much more widely used in practice

AP

Thank you!