Overview

- Some statistics stuff
- Latent variable models, EM, VAE
- Rate vs Distortion
- Improving VAEs
  - Reparameterization
  - A better prior
  - A better encoder through auxiliary variables
  - A better encoder through normalizing flows
  - A better objective (?): Wasserstein VAE
- Variational Bayes
- Local Reparameterization
- Conclusions

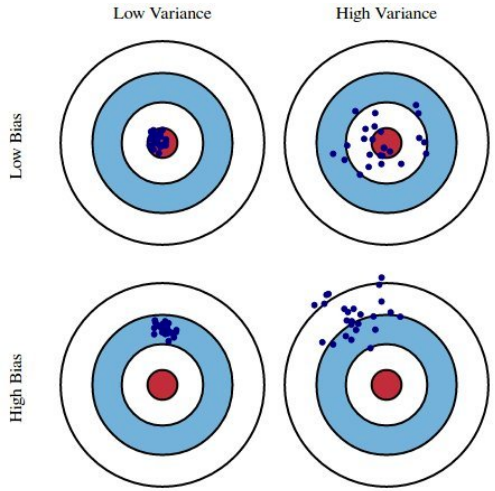
ML as Statistics

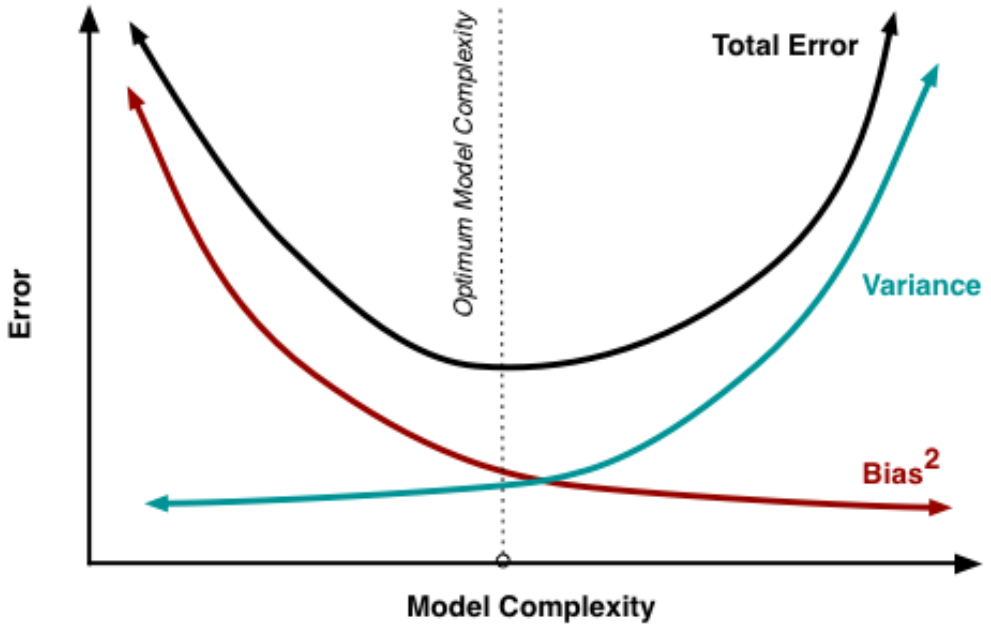
- Data: $\{X_1, ..X_n, Y_1, .., Y_n\} \sim P(X_1, ..X_n, Y_1, .., Y_n)$ (X input, Y label)
- Optimize objective:
  - maximize log likelihood: $\max_{\Theta} \log P(X_1, ...X_n|\Theta)$ (unsupervised)

    $\max_{\Theta} \log P(Y_1, ..Y_n|X_1, ...X_n, \Theta)$ (supervised)
  - minimize loss: $\min_{\Theta} \sum_i Loss(Y_i, \hat{Y}(X_i, \Theta))$ (supervised)

- *ML is more than an optimization problem: it's a statistical inference problem*.
  - E.g.: you should not optimize parameters more precisely than the scale at which the MLE fluctuates under resampling the data: $\Theta_{mle}(X, Y) \approx \Theta'_{mle}(X', Y')$, or risk overfitting.

Bias Variance Tradeoff





$$Y = f(X) + \epsilon \qquad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon).$$

$$Err(x) = E\left[(Y - \hat{f}(x))^2\right]$$

$$Err(x) = \left(E[\hat{f}(x)] - f(x)\right)^2 + E\left[\left(\hat{f}(x) - E[\hat{f}(x)]\right)^2\right] + \sigma_e^2$$

$$Err(x) = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

http://scott.fortmann-roe.com/docs/BiasVariance.html

Bayesian Inference

The image cannot be displayed. Your computer may not have enough memory to open the image, or the image may have been corrupted. Restart your computer, and then open the file again. If the red x still appears, you may have to delete the image and then insert it again.

(model evidence)

$$P(\Theta|X,M) = \frac{P(X|\Theta,M)P(\Theta|M)}{P(X|M)}$$

(posterior)

$$P(x|X,M) = \int d\Theta\ P(x|\Theta,M)P(\Theta|X,M)$$

(prediction)

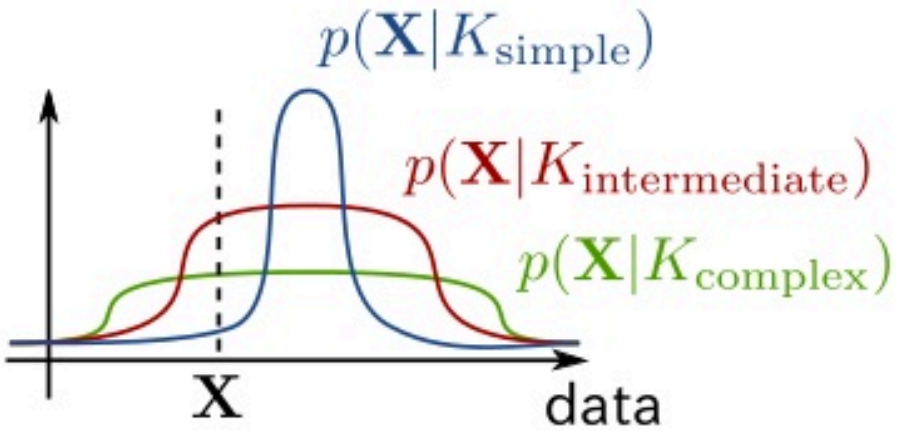
$$P(X) = \sum_M P(X|M)P(M)$$

(evidence)

$$P(M|X) = \frac{P(X|M)P(M)}{P(X)}$$

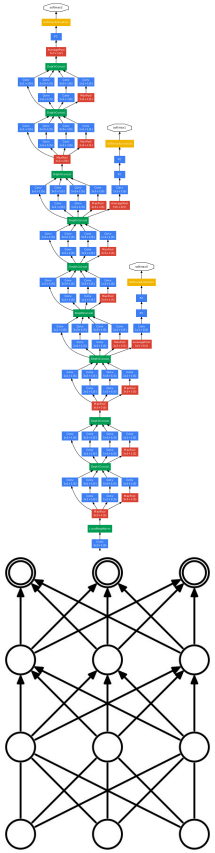
(model selection)

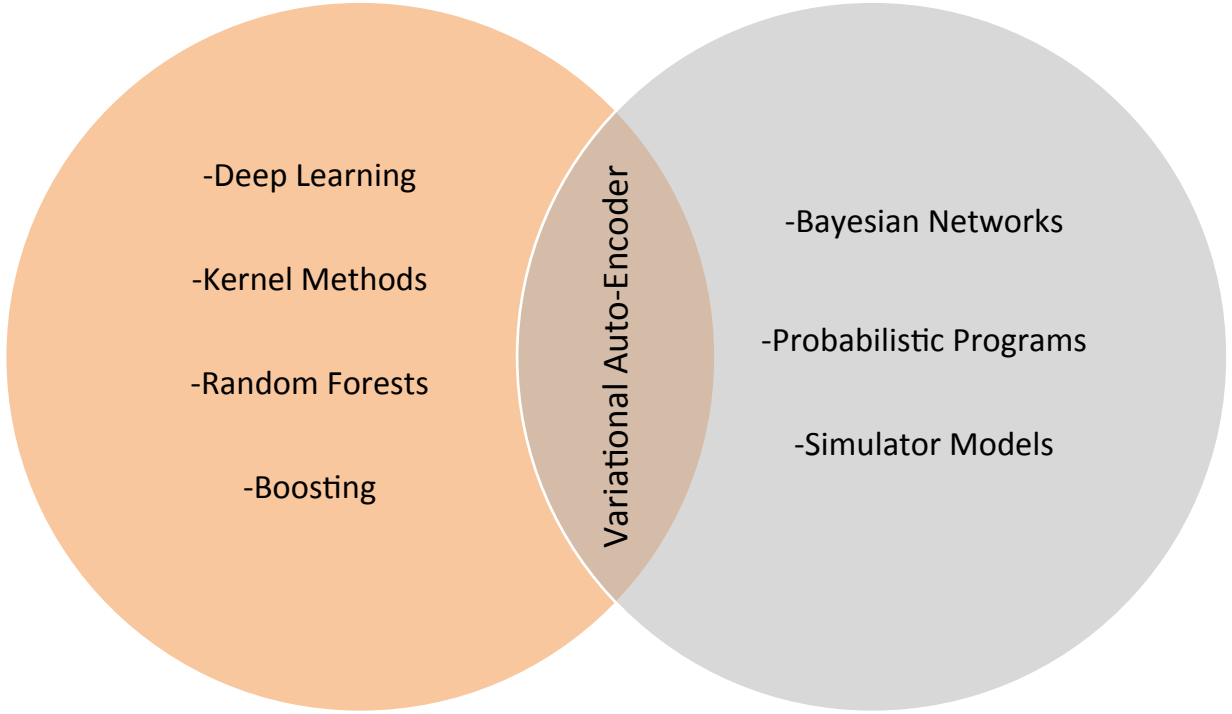
Complex models can have lower marginal likelihood:

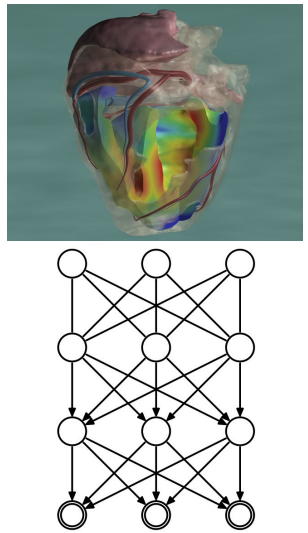


Picture credit: Kohei Hayashi[1,2] Shin-ichi Maeda[3] Ryohei Fujimaki[4]

Discriminative or Generative?





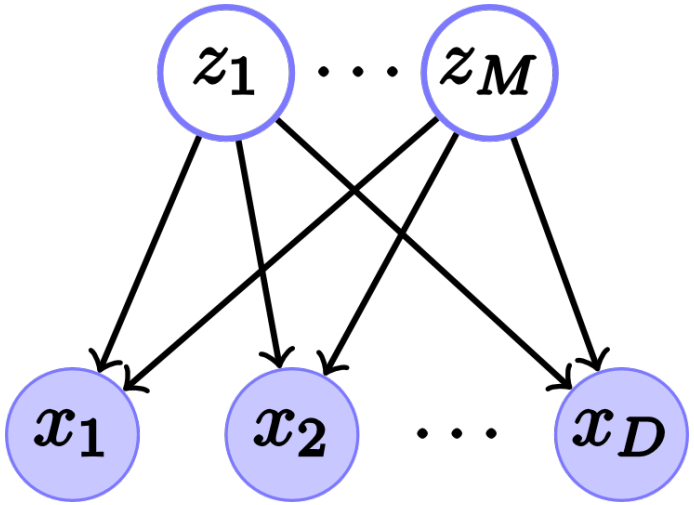


- Advantages discriminative models:
  - Flexible map from input to target (low bias)
  - Efficient training algorithms available
  - Solve the problem you are evaluating on.
  - Very successful and accurate!

- Advantages generative models:
  - Inject expert knowledge
  - Model causal relations, model physics
  - Interpretable
  - Data efficient
  - More robust to domain shift
  - Facilitate un/semi-supervised learning

Latent Variable Models

- Introducing latent (unobserved) variables will dramatically increase the capacity of a model.



$$P(X) = \sum_Z P(X|Z)P(Z)$$

- Problem: P(Z|X) is intractable for most nontrivial models

Learning: Variational Expectation Maximization

$$KL[Q(Z|X)||P(Z||X)]$$

$$\log P(X|\Theta) =$$

$$\log \sum_Z P(X|Z,\Theta)P(Z) \geq$$

$$\sum_Z Q(Z|X,\Phi)\left(\log P(X|Z,\Theta)P(Z) - \log Q(Z|X,\Phi)\right)$$

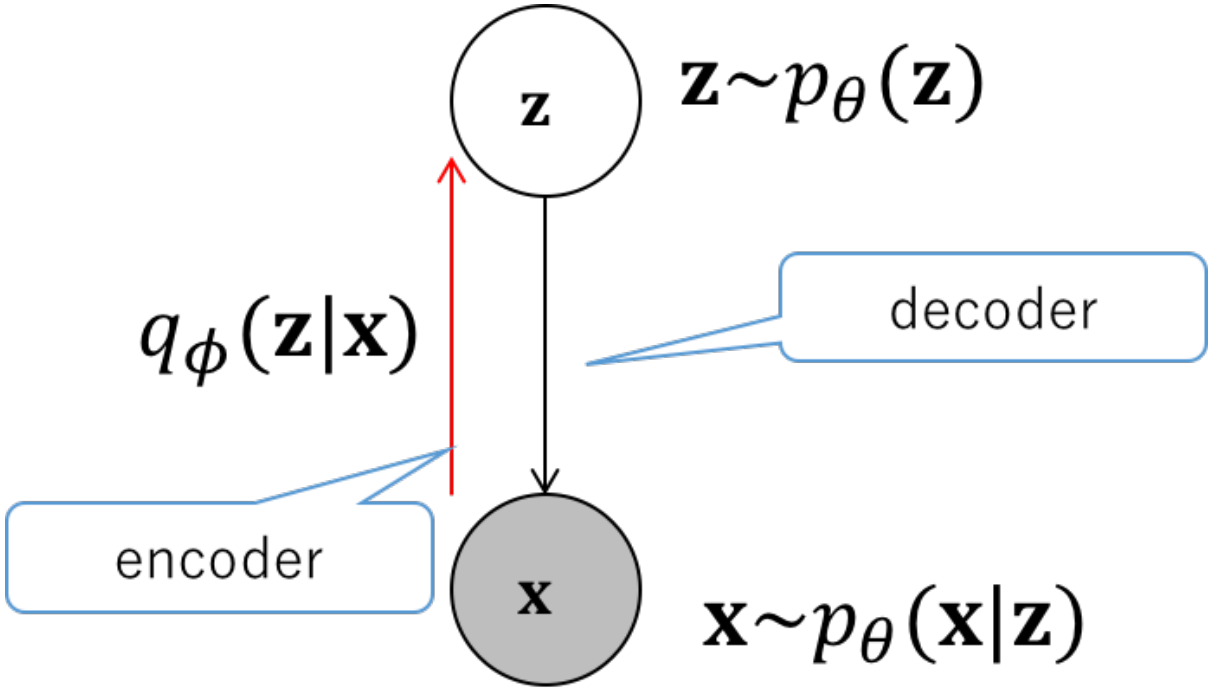
Gap:

Bound $B(\Theta,\Phi)$

E-step: $\arg\max_{\Phi} B(\Theta,\Phi)$ (variational inference)

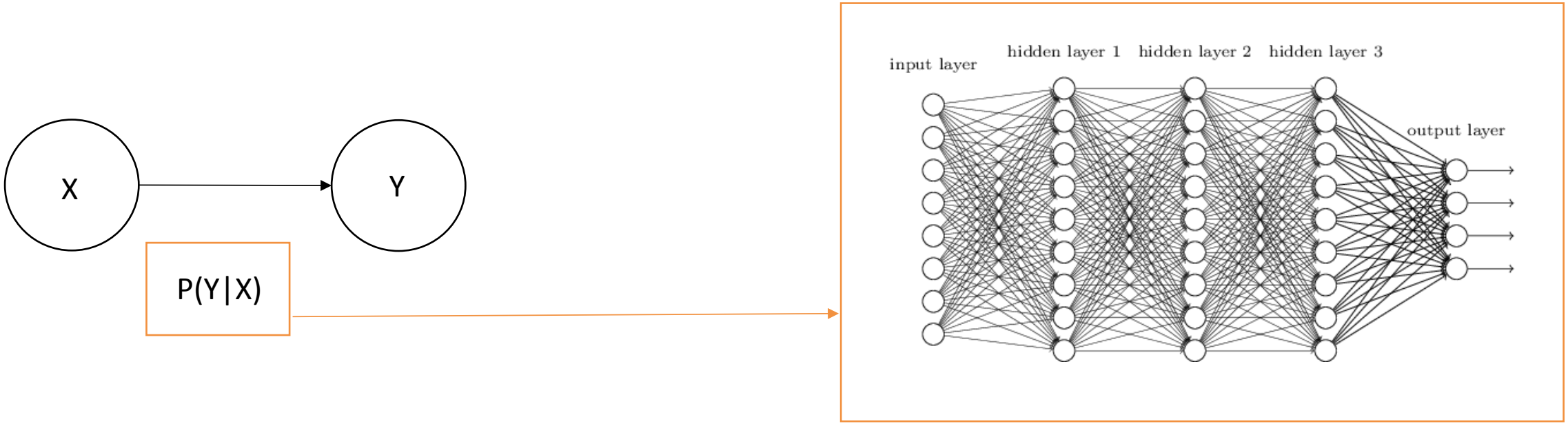
M-step: $\arg\max_{\Theta} B(\Theta,\Phi)$ (approximate learning)

Amortized Inference

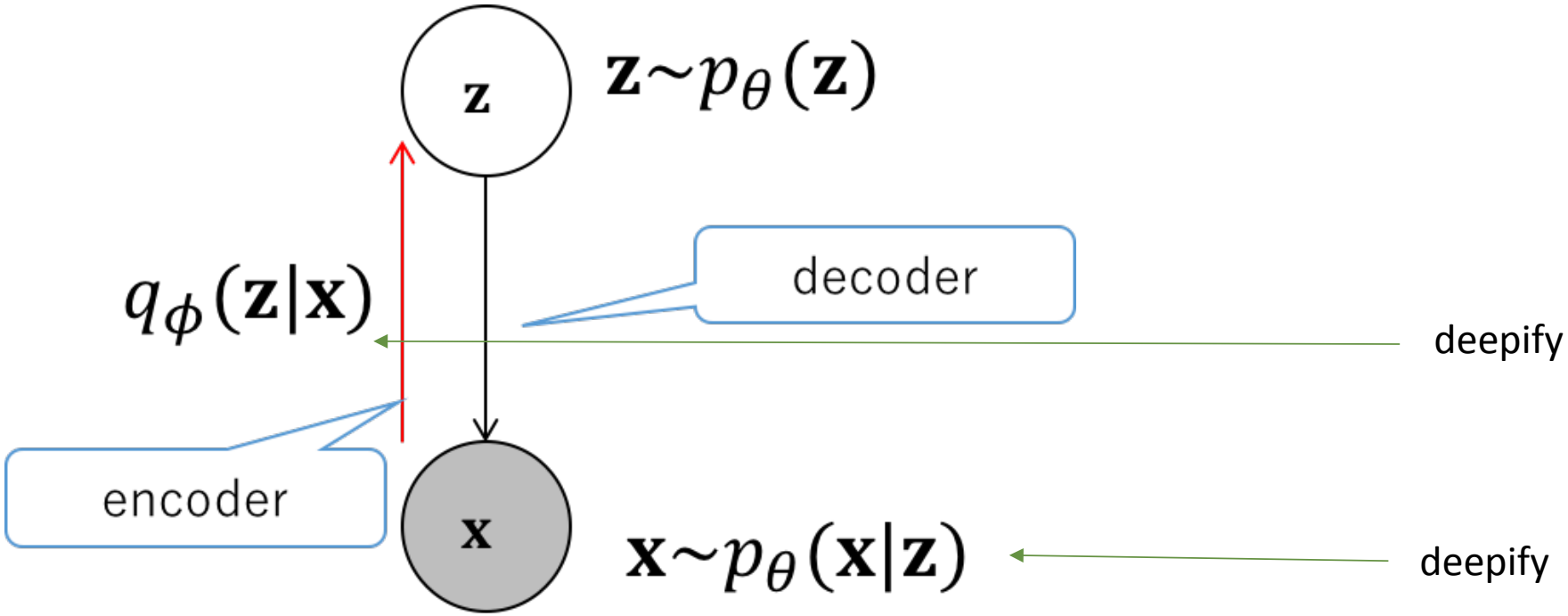


- Bij making q(z|x) a function of x and sharing parameters ϕ , we can do very fast inference at test time (i.e. avoid iterative optimization of $q_{test}(z)$)

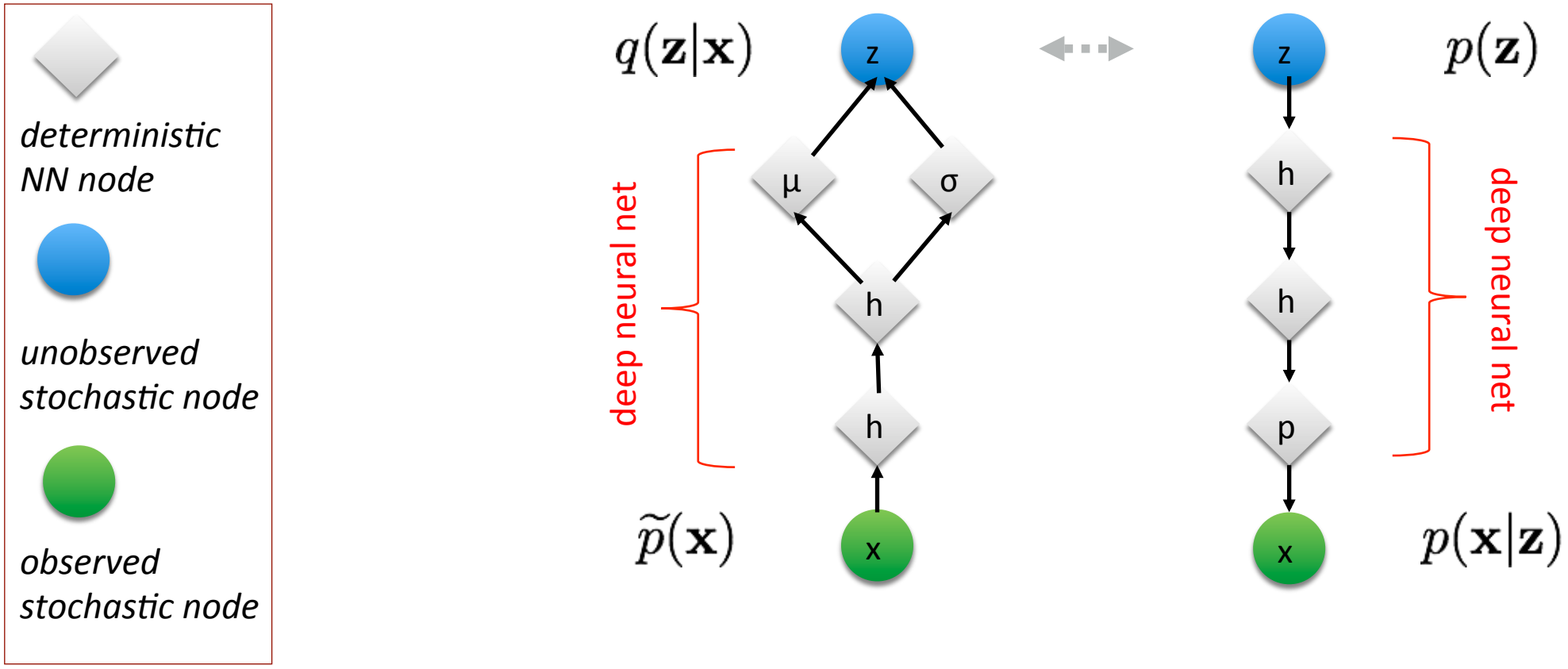
Deep NN as a glorified conditional distribution



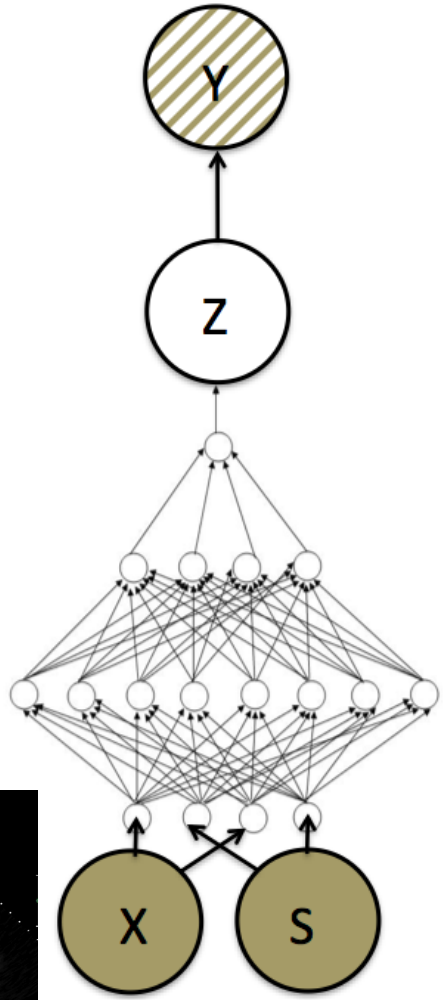
Variational Autoencoder



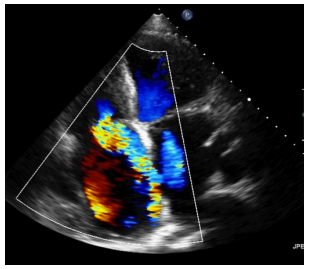
Deep Generative Model: The Variational Auto-Encoder



Combining Generative and Discriminative Models

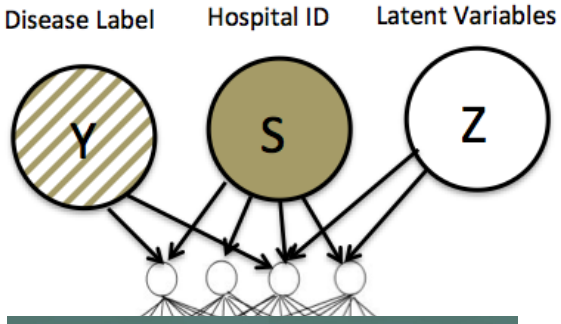


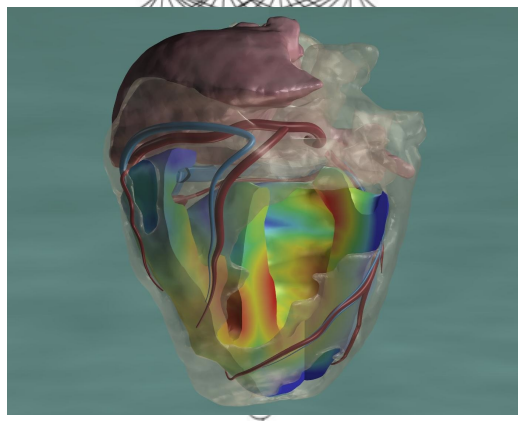
Black box DNN/CNN

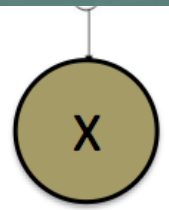


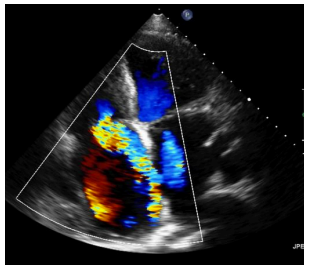
recognition model

generative model









Use physics

Use causality

Use expert knowledge

Stochastic Variational Bayesian Inference

$$B(Q) = \sum_Z Q(Z|X,\Phi)(\log P(X|Z,\Theta) + \log P(Z) - \log Q(Z|X,\Phi))$$

$$\nabla_\Phi B(Q) = \sum_Z Q(Z|X,\Phi)\nabla_\Phi \log Q(Z|X,\Phi)(\log P(X|Z,\Theta) + \log P(Z) - \log Q(Z|X,\Phi))$$

Sample Z

subsample mini-batch X

$$\nabla_\Phi B(Q) = \frac{1}{N}\frac{1}{S}\sum_{i=1}^{N}\sum_{s=1}^{S} \nabla_\Phi \log Q(Z_{is}|X_i,\Phi)(\log P(X_i|Z_{is},\Theta) + \log P(Z_{is}) - \log Q(Z_{is}|X_i,\Phi))$$

very high variance

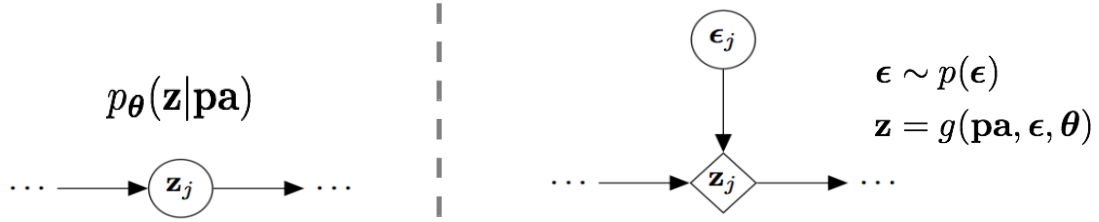
Improving VAEs

- Reducing Variance during learning: e.g. reparameterization, REBAR/RELAX…
- Rate / Distortion tradeoff: e.g. free bits, annealing KL
- More complex models
  - Prior P(z): e.g. VampPrior
  - Decoder P(x|z), e.g. PixelCNN
  - Encoder: e.g. normalizing flows, auxiliary variables

Reducing the Variance: The Reparametrization Trick

Kingma 2013, Bengio 2013, Kingma & Welling 2014

- Reparameterization:



- Applied to VAE:

$$\nabla_\Phi B(\Theta, \Phi) = \nabla_\Phi \int dz\ Q_\Phi(z|x)[\log P_\Theta(x, z) - \log Q_\Phi(z|x)]$$

$$\approx \nabla_\Phi [\log P_\Theta(x, z_s) - \log Q_\Phi(z_s|x)]_{z_s = g(\epsilon_s, \Phi)}, \qquad \epsilon_s \sim P(\epsilon)$$

- Example:

$$\nabla_\mu \int dz \mathcal{N}_z(\mu, \sigma) z$$

$$= \frac{1}{S} \sum_s z_s (z_s - \mu)/\sigma^2, \quad z_s \sim \mathcal{N}_z(\mu, \sigma)$$

$$\text{or } \frac{1}{S} \sum_s 1, \quad \epsilon_s \sim \mathcal{N}_\epsilon(0, 1), \quad z = \mu + \sigma\epsilon$$

Rate vs Distortion

Fixing a Broken ELBO

Alexander A. Alemi [1] Ben Poole [2*] Ian Fischer [1] Joshua V. Dillon [1] Rif A. Saurous [1] Kevin Murphy [1]

- The ELBO objective does not care explicitly about learning good representations Z
- Proposal: look at both distortion and rate.

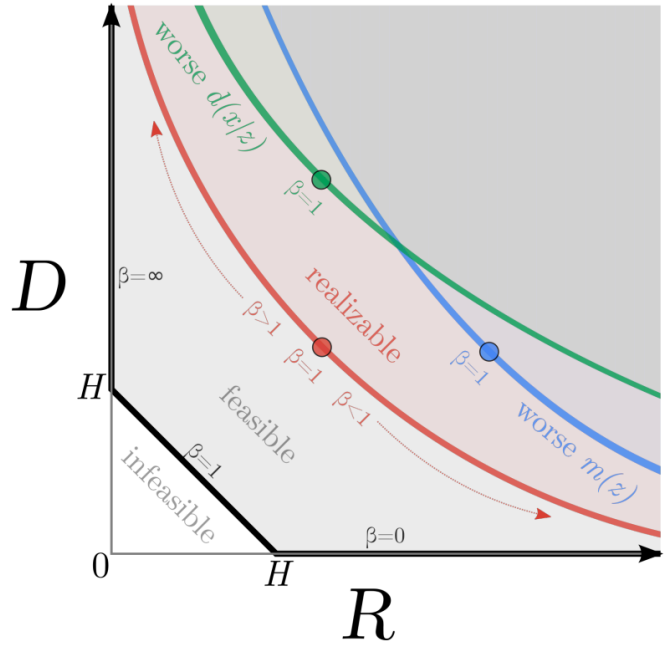
$$\min_{e(z|x),m(z),d(x|z)} D + \beta R.$$

$$\min_{e(z|x),m(z),d(x|z)} \int dx\, p^*(x) \int dz\, e(z|x) \left[-\log d(x|z) + \beta \log \frac{e(z|x)}{m(z)} \right].$$

$$H \equiv -\int dx\, p^*(x) \log p^*(x)$$

$$D \equiv -\int dx\, p^*(x) \int dz\, e(z|x) \log d(x|z)$$

$$R \equiv \int dx\, p^*(x) \int dz\, e(z|x) \log \frac{e(z|x)}{m(z)}$$



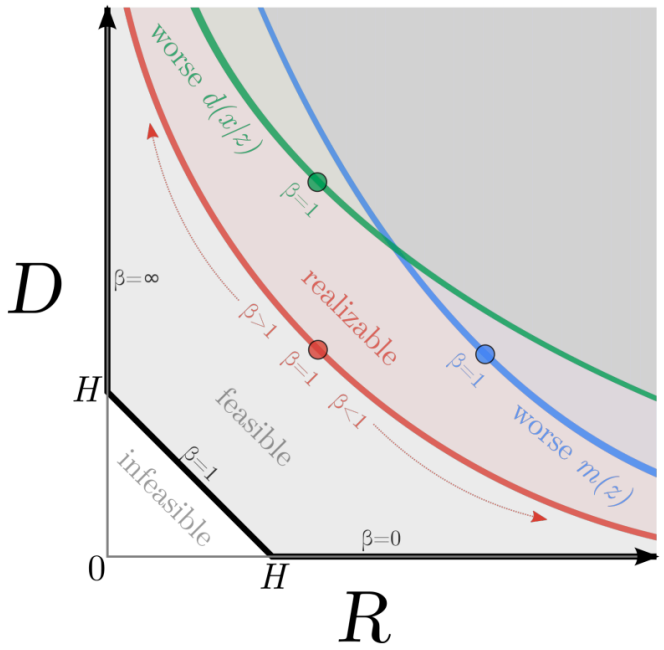
$$H - D \leq \mathrm{I}_e(X;Z) \leq R$$

$$\mathrm{I}_e(X;Z) = \iint dx\, dz\, p_e(x,z) \log \frac{p_e(x,z)}{p^*(x) p_e(z)}.$$

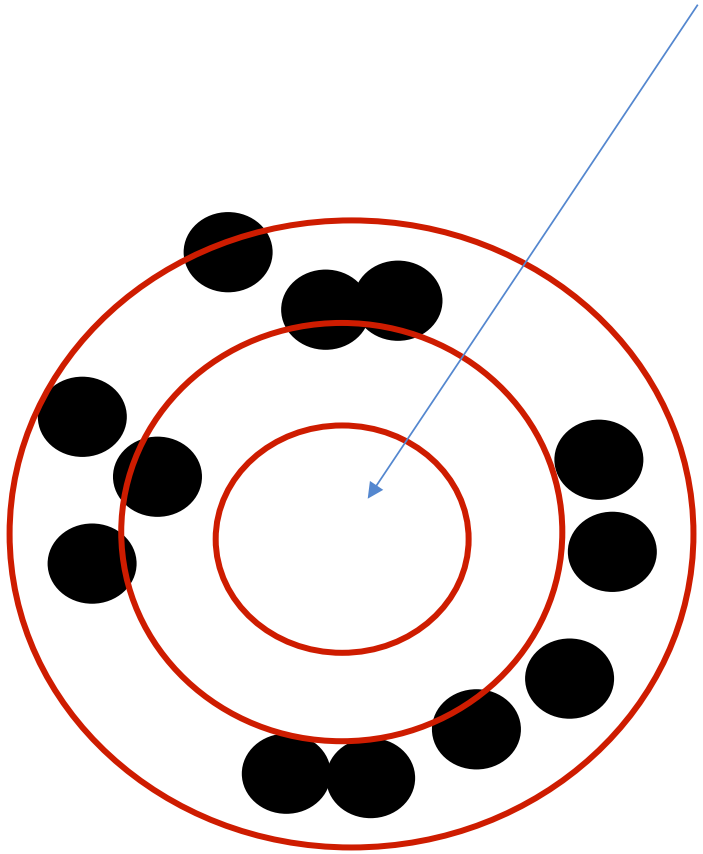
$$p_e(x,z) = e(z|x)p^*(x)\ ;\ \ p_e(z) = \int dx\, p^*(x) e(z|x)$$

Rate vs Distortion Conclusions

- ELBO is too one dimensional, it does not care about representations.
- We can improve R-D by changing encoder, decoder, or prior.



Improving the Prior



“hole”

Model was not trained to generate from the hole.

Improving the Prior (Tomczak & W, 2018)

- We can simply solve for the best prior p(.):

$$p^*(z) = \arg\max_{p(\cdot)} \sum_n \int dz_n q(z_n|x_n) \left[\log p(x_n|z_n) + \log p(z_n) - \log q(z_n|x_n)\right]$$

$$= \frac{1}{N}\sum_{n=1}^{N} q(z|x_n)$$ "Variational Mixture of Posteriors Prior" or VampPrior

- Too expensive because it scales with N. Approximate using K pseudo data points {u_k}:

$$p^*(z) \approx \frac{1}{K}\sum_{k=1}^{K} q(z|u_k) \quad \text{or} \quad p^*(z) \approx \frac{1}{K}\sum_{k=1}^{K} w_k q(z|u_k); \quad \sum_k w_k = 1$$ (Alemi et al 2018)

- Learn the {w_k, u_k} alongside the parameters of the posterior q(z|u)

Improving the Encoder: Auxiliary Variables

(Salimans & Knowles, 2013

- Introduce new latent variables y and distributions:

$$p(x,z) \rightarrow p(x,z)r(y|x,z)$$
$$q(z|x) \rightarrow q(z,y|x)$$

- The variational posterior is now a mixture distribution: Note that q(z|x) can now be very complicated and intractable.

$$q(z|x) = \int dy\ q(z,y|x)$$

- New, but looser bound, on joint space : Note that this bound can still be better than the usual ELBO with a simpler q(z|x).

$$\begin{aligned}
&\mathcal{L}_{\text{aux}} \qquad (3)\\
&= \mathbb{E}_{q(y,z_T|x)}[\log[p(x,z_T)r(y|x,z_T)] - \log q(y,z_T|x)]\\
&= \mathcal{L} - \mathbb{E}_{q(z_T|x)}\{D_{KL}[q(y|z_T,x)||r(y|z_T,x)]\}\\
&\leq \mathcal{L} \ \leq \ \log[p(x)],
\end{aligned}$$

- (Homework: derive (3) by first proving the following general triangle inequality for KL-divergences:

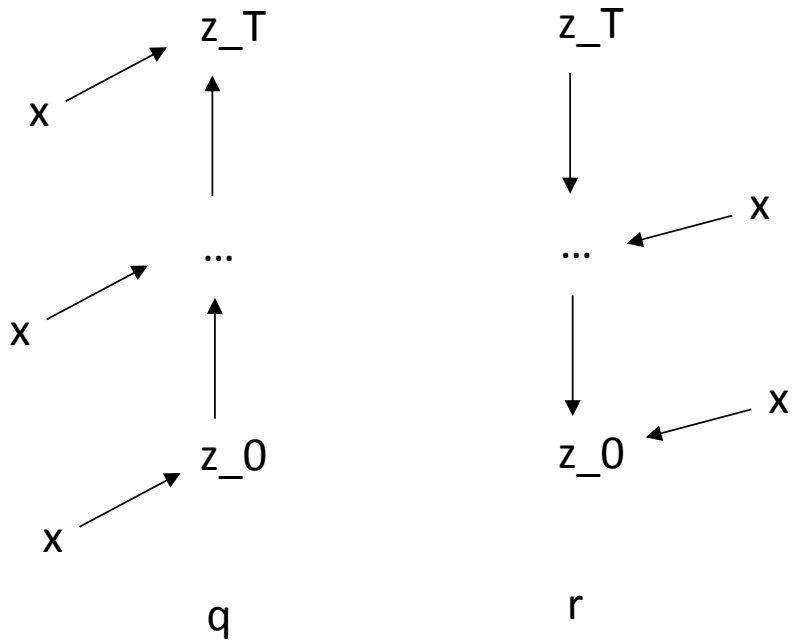
$$KL[q(x,z)||p(x,z)] = KL[q(x)||p(x)] + E_{q(x)}\left[KL[q(z|x)||p(z|x)]\right]\ )$$

Auxiliary Variables

Salimans et al 2015

Choose Markov Chains:

$$r(y|z,x) = r(z_{T-1}|x,z_T)r(z_{T-2}|z_{T-1},x)...r(z_0|z_1,x)$$
$$q(z,y|x) = q(z_0|x)q(z_1|z_0,x)...q(z_T|z_{T-1},x)$$



$$\begin{aligned}\log p(x) &\geq \mathbb{E}_q[\log p(x,z_T) - \log q(z_0,\ldots,z_T|x) \\ &\quad + \log r(z_0,\ldots,z_{t-1}|x,z_T)] \\ &= \mathbb{E}_q\big[\log[p(x,z_T)/q(z_0|x)] \\ &\quad + \sum_{t=1}^{T} \log[r_t(z_{t-1}|x,z_t)/q_t(z_t|x,z_{t-1})]\big].\end{aligned}$$

Algorithm to compute bound:

Draw an initial random variable $z_0 \sim q(z_0|x)$
Initialize the lower bound estimate as
$L = \log p(x,z_0) - \log q(z_0|x)$
for $t = 1:T$ **do**
  Perform random transition $z_t \sim q_t(z_t|x,z_{t-1})$
  Calculate the ratio $\alpha_t = \frac{p(x,z_t)r_t(z_{t-1}|x,z_t)}{p(x,z_{t-1})q_t(z_t|x,z_{t-1})}$
  Update the lower bound $L = L + \log[\alpha_t]$
end for

Improving the Encoder: Normalizing Flows (I)

- The usual VAE predicts mean and log-variance functions $\mu(x), \log \sigma(x)$, and then samples $z_n \sim N(z; \mu(x_n), \sigma(x_n))$ This is unimodal and this quite limited.
- How can we produce more complicated distributions?
- Consider an invertible transformation f(z): $\mathbf{z}' = f(\mathbf{z}),\ \mathbf{z}, \mathbf{z}' \in \mathbb{R}^D \text{ and } f : \mathbb{R}^D \mapsto \mathbb{R}^D$
- The density then transforms as: $p_1(\mathbf{z}') = p_0(\mathbf{z}) \left| \det \left(\frac{\partial f(\mathbf{z})}{\partial \mathbf{z}} \right) \right|^{-1}$
- Let's take a simple distribution q0 for z0 and then transform K times z0→z1→z2→…→zK.
- The final log distribution is now given as:

$$\log q_K(\mathbf{z}_K|\mathbf{x}) = \log q_0(\mathbf{z}_0|\mathbf{x}) - \sum_{k=1}^{K} \log \left| \det \left(\frac{\partial f_k(\mathbf{z}_{k-1}; \lambda_k(\mathbf{x}))}{\partial \mathbf{z}_{k-1}} \right) \right|$$

Improving the Encoder: Normalizing Flows (II)

- Final variational objective is now:

$$\begin{aligned}\mathcal{F}(\theta, \phi) &= \mathbb{E}_{q_\phi}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})] \\ &= \mathbb{E}_{q_0}[\log q_0(\mathbf{z}_0|\mathbf{x}) - \log p_\theta(\mathbf{x}, \mathbf{z})] \\ &- \mathbb{E}_{q_0}\left[\sum_{k=1}^{K} \log \left|\det \left(\frac{\partial f_k(\mathbf{z}_{k-1}; \lambda_k(\mathbf{x}))}{\partial \mathbf{z}_{k-1}}\right)\right|\right]\end{aligned}$$

- Planar flows (Rezende & Mohamed, 2015) are convenient functions: $\mathbf{z}' = \mathbf{z} + \mathbf{u}h(\mathbf{w}^T\mathbf{z} + b)$

  with efficiently computed Jacobian:

$$\begin{aligned}\det \frac{\partial \mathbf{z}'}{\partial \mathbf{z}} &= \det\left(\mathbf{I} + \mathbf{u}h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w}^T\right) \\ &= 1 + \mathbf{u}^T h'(\mathbf{w}^T\mathbf{z} + b)\mathbf{w},\end{aligned}$$

- Note: h is a scalar function and acts as a one-neuron bottleneck.

Improving the Encoder: Sylvester Flows

Van den Berg et al 2018

- A more general flow without the bottleneck is: $\mathbf{z}' = \mathbf{z} + \mathbf{A}h(\mathbf{B}\mathbf{z} + \mathbf{b})$,
  with $\mathbf{A} \in \mathbb{R}^{D \times M}$, $\mathbf{B} \in \mathbb{R}^{M \times D}$, $\mathbf{b} \in \mathbb{R}^{M}$, and $M \leq D$

- We can now the following theorem:

Theorem 1 (Sylvester's determinant identity)**.** *For all* $\mathbf{A} \in \mathbb{R}^{D \times M}$, $\mathbf{B} \in \mathbb{R}^{M \times D}$,

$$\det\left(\mathbf{I}_D + \mathbf{A}\mathbf{B}\right) = \det\left(\mathbf{I}_M + \mathbf{B}\mathbf{A}\right), \qquad (11)$$

where $\mathbf{I}_M$ *and* $\mathbf{I}_D$ *are* M *and* D*-dimensional identity matrices, respectively.*

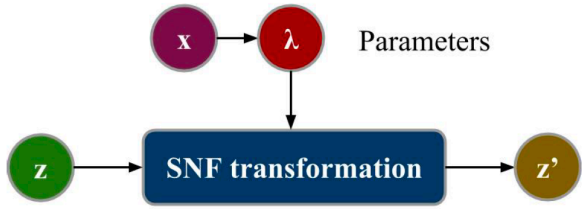
Improving the Encoder: Sylvester Flows

$$\mathbf{z}' = \mathbf{z} + \mathbf{A}h(\mathbf{B}\mathbf{z} + \mathbf{b}),$$

- To achieve invertibility we use: $A = QR$ and $B = \tilde{R}Q^T$ with R & $\tilde{R}$ upper-triangular and

  $\mathbf{Q} = (\mathbf{q}_1 \dots \mathbf{q}_M)$ with q_i an orthonormal set of vectors.

- The parameters of R, Q are all functions of x.



- We then finally have:

Theorem 2. *Let* $\mathbf{R}$ *and* $\tilde{\mathbf{R}}$ *be upper triangular matrices. Let* $h : \mathbb{R} \longrightarrow \mathbb{R}$ *be a smooth function with bounded, positive derivative. Then, if the diagonal entries of* $\mathbf{R}$ *and* $\tilde{\mathbf{R}}$ *satisfy* $r_{ii}\tilde{r}_{ii} > -1/\|h'\|_\infty$ *and* $\tilde{\mathbf{R}}$ *is invertible, the transformation given by* (13) *is invertible.*

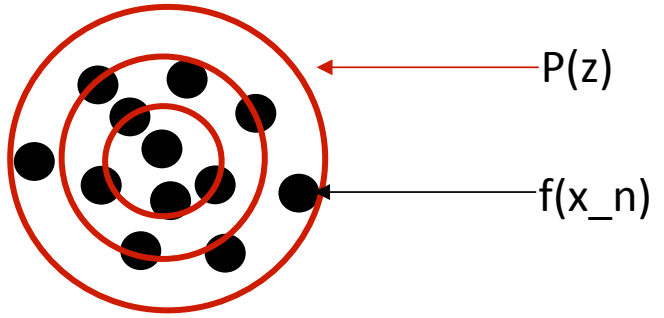
Changing the Objective: Wasserstein AE

- The term $E_{p(x)}\left[KL[q(z|x)||p(z)]\right]$ puts pressure on keeping q(z|x) close to p(z) for every value of x!
- An alternative requirement is to only force the aggregated posterior q(z) close to p(z):

$$E_{p(x)}\left[KL[q(z|x)||p(z)]\right] \longrightarrow KL[E_{p(x)}q(z|x)||p(z)]$$

- This says that the embedded data-points should distribute according to p(z), a much weaker condition!
- Amazingly, with this new KL the objective the ELBO becomes the dual formulation of the Wasserstein GAN! (Tolstikhin et al 2018)

$$\mathrm{POT}_{\|\cdot\|}(\mathbb{P}_{\text{data}}, \mathbb{P}_G) = \inf_{\mathbb{Q}_{Z|X}} \mathop{\mathbb{E}}_{\substack{\mathbf{x}\sim\mathbb{P}_{\text{data}} \\ \mathbf{z}\sim\mathbb{Q}_{Z|X=\mathbf{x}}}} \|\mathbf{x} - G(\mathbf{z})\| + \lambda_Z \cdot \mathfrak{D}(\mathbb{Q}_Z \,\|\, \mathbb{P}_Z)$$



(Wasserstein autoencoder, if D is enforced using a GAN this is a adversarial autoencoder Makhzani et al)

Learning as Bayesian Inference

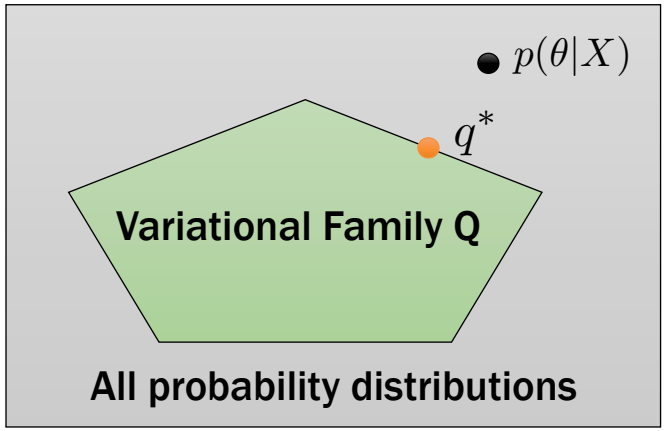
Bayes vs ML

- What's the difference between a prior p(z) over latent variables and a prior over p(w) over parameters?
- Why is Bayesian learning conceptually very different from maximum likelihood learning?
- Is a "Bayesian Network" Bayesian?
- Why Bayes:
  - Less overfitting
  - Uncertainties over predictions due to model
  - Compression

Bayesian Posterior Inference

Variational Inference

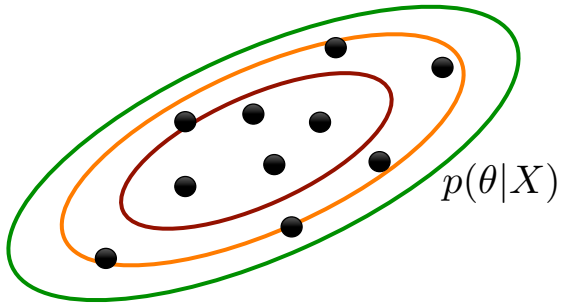
$$q^* = \min_{q \in Q} \mathrm{KL}[q(\theta)||p(\theta|X)]$$



- Deterministic
- Biased
- Local minima
- Easy to assess convergence

Sampling

$$\mathbb{E}_{p(\theta|X)}[f(\theta)] \approx \frac{1}{T}\sum_{t=1}^{T} f(\theta_t) \quad : \quad \theta_t \sim p(\theta|X)$$



- Stochastic (sample error)
- Unbiased
- Hard to mix between modes
- Hard to assess convergence

Hinton

Hinton & Van Camp, 1993,
"Keeping Neural Networks Simple by Minimizing the Description Length of the Weights"

Neal & Hinton, 1998
"A view of the EM algorithm that justifies incremental, sparse, and other variants"

$$\log P(X) = \int \mathrm{d}\Theta\ Q(\Theta) \log P(X|\Theta) - \mathrm{KL}[Q(\Theta)||P(\Theta)] + \mathrm{KL}[Q(\Theta)||P(\Theta|X)]$$

$$\log P(X) \geq \int \mathrm{d}\Theta\ Q(\Theta) \log P(X|\Theta) - \mathrm{KL}[Q(\Theta)||P(\Theta)]$$

$$= \int \mathrm{d}\Theta\ Q(\Theta) \log P(X|\Theta)P(\Theta) - \int \mathrm{d}\Theta\ Q(\Theta) \log Q(\Theta)$$

- Free Energy = - Energy + Entropy

Stochastic Variational Bayes

$$B(Q(\Theta)|X) = \int_{\Theta} d\Theta \;\; Q(\Theta) \left[\log P(X|\Theta) + \log P(\Theta) - \log Q(\Theta)\right]$$

$$\nabla_{\Phi} B = \int_{\Theta} d\Theta \;\; Q_{\Phi}(\Theta) \; \nabla_{\Phi} \log Q_{\Phi}(\Theta) \left[\log P(X|\Theta) + \log P(\Theta) - \log Q_{\Phi}(\Theta)\right]$$

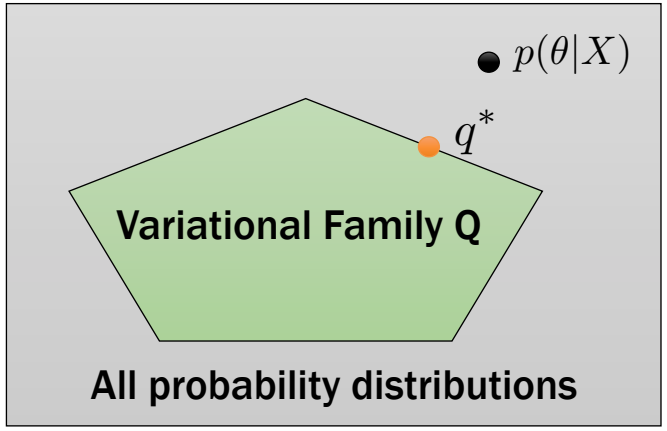
sample

subsample mini-batch X

$$\nabla_{\Phi} B = \frac{1}{S} \sum_{s=1}^{S} \nabla_{\Phi} \log Q_{\Phi}(\Theta_s) \left[\frac{N}{n} \sum_{i=1}^{n} \log P(x_i|\Theta_s) + \log P(\Theta_s) - \log Q_{\Phi}(\Theta_s)\right]$$

very high variance 

Variational Bayesian Learning



$$\Phi^* = \text{argmax}_\Phi \int \mathrm{d}\Theta\ Q_\Phi(\Theta)[\log P(X|\Theta)P(\Theta) - \log Q_\Phi(\Theta)]$$

$$= \text{argmax}_\Phi \int \mathrm{d}\Omega\ P_0(\Omega)[\log P(X|f(\Omega,\Phi))P(f(\Omega,\Phi)) - \log Q_\Phi(f(\Omega,\Phi))]$$

$$\Theta = f(\Omega,\Phi), \quad \text{s.t.} \quad Q_\Phi(\Theta)\mathrm{d}\Theta = P_0(\Omega)\mathrm{d}\Omega$$

("reparameterization trick", Kingma & W. 2013)

High Variance Gradient Estimator

$$L_{\mathcal{D}}(\phi) \simeq L_{\mathcal{D}}^{\mathrm{SGVB}}(\phi) = \frac{N}{M} \sum_{i=1}^{M} \log p(\mathbf{y}^i | \mathbf{x}^i, \mathbf{w} = f(\epsilon, \phi)),$$

- Sample epsilon once, then compute gradients on each data-case and sum.
- This has high variance because the data terms are now correlated because they share single epsilon
- Solution (?): draw different samples epsilon for every datacase
- Problem: very expensive: at every layer we have to multiply M times (M=nr. data in minibatch) a KxK matrix of random weights (K is nr. neurons in layer).

Local Reparameterization for Deep NNs

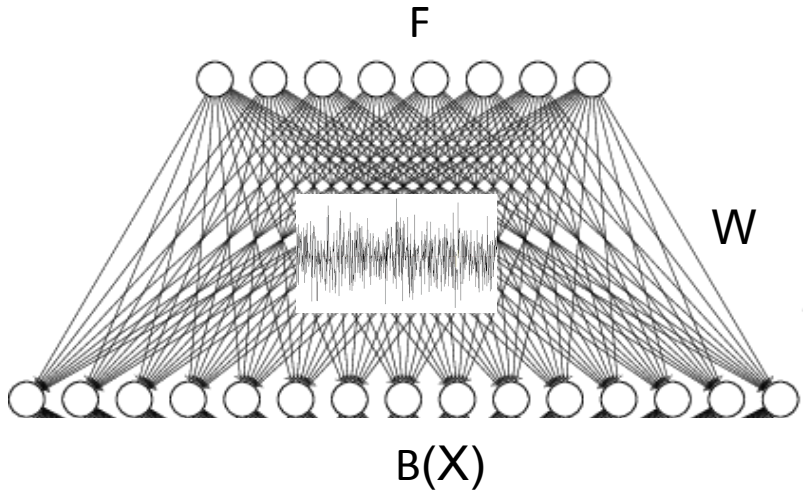
Kingma, Salimans & W. 2015

$$\int dW Q(W) \log P(Y|X,W) - \mathrm{KL}(Q(W)||P(W))$$

Reparameterize: compute exactly

$$\sum_{i=1}^{N} \int dW \; Q(W) \log P(y_i|W\phi(b_i)) \int df_i \; I(f_i - W\phi(b_i))$$

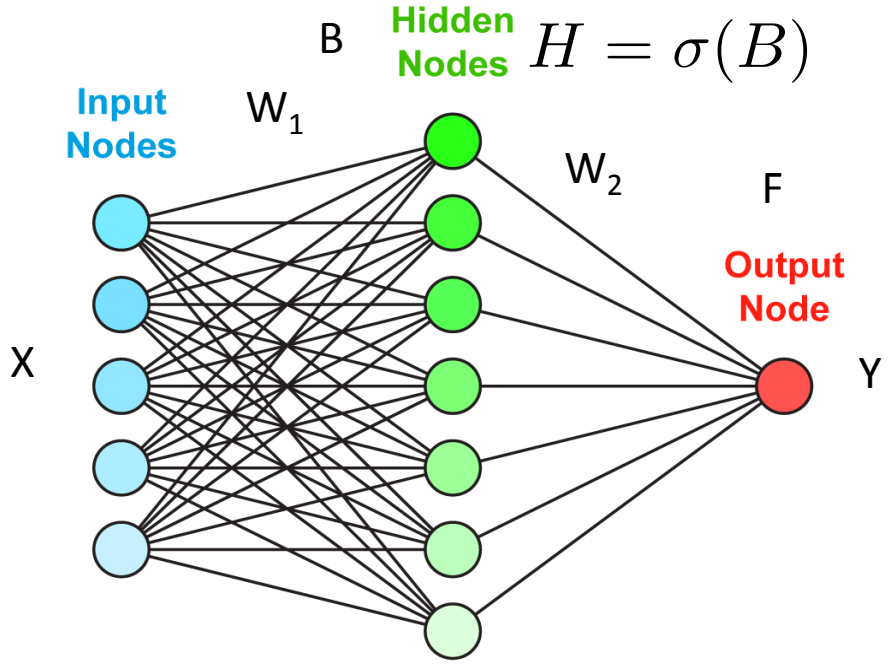
$$\sum_{i=1}^{N} \int df_i \; Q(f_i|b_i) \log P(y_i|f_i)$$



- Sampling f (KxM) for every data case is cheaper than sampling W for every data case (KxKxM)
- This trick reduces also variance

Bayesian Neural Networks

$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}\left[\mathbb{E}_{\tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X})\tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B})}\left[\log p(\mathbf{Y}|\mathbf{F})\right] - \sum_{i=1}^{2} KL(q_{\phi_i}(\mathbf{W}_i)||p_{\theta_i}(\mathbf{W}_i))\right]$$



Interesting relations to Deep GPs and Information Bottleneck

Extensions

$$\mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}[\log p(\mathbf{Y}|\mathbf{X})] \leq \mathbb{E}_{\tilde{p}(\mathbf{x},\mathbf{y})}\left[\mathbb{E}_{\tilde{q}_{\phi_1}(\mathbf{B}|\mathbf{X})\tilde{q}_{\phi_2}(\mathbf{F}|\mathbf{B})}\left[\log p(\mathbf{Y}|\mathbf{F})\right] - \sum_{i=1}^{2} KL(q_{\phi_i}(\mathbf{W}_i)||p_{\theta_i}(\mathbf{W}_i))\right]$$

- Use matrix normal priors and posteriors $p(\mathbf{W}) = \mathcal{MN}(\mathbf{M}, \mathbf{U}, \mathbf{V})$
$$= \frac{\exp\left(-\frac{1}{2}\operatorname{tr}\left[\mathbf{V}^{-1}(\mathbf{W}-\mathbf{M})^T\mathbf{U}^{-1}(\mathbf{W}-\mathbf{M})\right]\right)}{(2\pi)^{np/2}|\mathbf{V}|^{n/2}|\mathbf{U}|^{n/2}}$$
- Use normalizing flows: $q_\phi(\mathbf{W}|\mathbf{z}) = \prod_{i=1}^{D_{in}}\prod_{j=1}^{D_{out}}\mathcal{N}(z_i\mu_{ij}, \sigma_{ij}^2)$, $\mathbf{z} \sim q_\phi(\mathbf{z})$ = normalizing flow

Louizos & W. "Structured and efficient variational deep learning with matrix Gaussian posteriors 2016.

Louizos & W. "Multiplicative Normalizing Flows for Variational Bayesian Neural Networks, 2017

Conclusions

- Think about variance and bias for parameter estimators or gradient estimators.
- There is more to representation learning than just the ELBO
- VAEs can be improved along many dimensions (prior, encoder, decoder, objective)
- A model is not Bayesian when there is a prior over latent variables
- Local reparameterization can help reduce variance and computation in Variational Bayesian Learning